```
// ---------------------------------------------
// ------ EnString.c - START --------------------
// ---------------------------------------------


/**
 * \defgroup Strings Strings
 * @{
 */
class string
{
■static const string Empty;
■■■
■/**
■\brief Converts string to integer
■■\return \p int - Converted \p string.
■■@code
■■■string str = "56";
■■■int i = str.ToInt();
■■■Print(i);

■■■>> i = 56
■■@endcode
■*/
■proto native int ToInt();
■
■/**
■\brief Converts string to integer
■■\return \p int - Converted \p string.
■■@code
■■■string str = "0xFF";
■■■int i = str.HexToInt();
■■■Print(i);

■■■>> i = 255
■■@endcode
■*/
■proto native int HexToInt();
■
■/**
■\brief Converts string to float
■■\return \p float - Converted \p string \p in float.
■■@code
■■■string str = "56.6";
■■■float f = str.ToFloat();
■■■Print(f);

■■■>> f = 56.6
■■@endcode
■*/
■proto native float ToFloat();
■■■
■/**
■\brief Returns a vector from a string
■■\return \p vector Converted s as vector
■■@code
■■■string str = "1 0 1";
■■■vector v = str.ToVector();
■■■Print(v);

■■■>> v = <1,0,1>
■■@endcode
■*/
```

```
// ---------------------------------------------
// ------ EnSystem.c - START --------------------
// ---------------------------------------------


/**
 * \defgroup System System
 * @{
 */

/**
\brief Returns world time
■\param[out] hour \p int Hour
■\param[out] minute \p int Minute
■\param[out] second \p int Second
■\return \p void
■@code
■■int hour = 0;
■■int minute = 0;
■■int second = 0;

■■GetHourMinuteSecondUTC(hour, minute, second);

■■Print(hour);
■■Print(minute);
■■Print(second);

■■>> hour = 16
■■>> minute = 38
■■>> second = 7
■@endcode
*/
proto void GetHourMinuteSecond(out int hour, out int minute, out int second);

/**
\brief Returns world date
■\param[out] year \p int Year
■\param[out] month \p int Month
■\param[out] day \p int Day
■\return \p void
■@code
■■int year = 0;
■■int month = 0;
■■int day = 0;

■■GetYearMonthDay(year, month, day);

■■Print(year);
■■Print(month);
■■Print(day);

■■>> year = 2015
■■>> month = 3
■■>> day = 24
■@endcode
*/
proto void GetYearMonthDay(out int year, out int month, out int day);

/**
\brief Returns UTC world time
■\param[out] hour \p int Hour
■\param[out] minute \p int Minute
■\param[out] second \p int Second
```

```
// ----------------------------------------------
// ------ entities.c - START --------------------
// ----------------------------------------------


// Include c ---------------
// #include "Scripts/Entities/scriptedEntities.c"

// Include h (temporal) ---------------
// #include "Scripts/Entities/Game/Super/InventoryItem.c"
// #include "Scripts/Entities/Core/Inherited/InventoryItem.c"

// #include "Scripts/Entities/Game/Super/Man.c"
// #include "Scripts/Entities/Core/Inherited/Man.c"

// #include "Scripts/Entities/Game/Super/LightAI.c"
// #include "Scripts/Entities/Core/Inherited/LightAI.c"

// #include "Scripts/Entities/Core/Inherited/Building.c"

// #include "Scripts/Entities/Game/Super/Plant.c"
// #include "Scripts/Entities/Core/Inherited/Plant.c"

// #include "Scripts/Entities/Game/Super/Building.c"

// #include "Scripts/Static/_Includes.c"

// Include c 2 ---------------


// #include "Scripts/Classes/_Includes.c"
// #include "Scripts/Modules/PluginManager.c"

// #include "Scripts/Entities/_Includes.c"




// ----------------------------------------------
// ------ entities.c - END ----------------------
// ----------------------------------------------


// ----------------------------------------------
// ------ Entity.c - START --------------------
// ----------------------------------------------


class Entity extends ObjectTyped
{
■proto native void DisableSimulation(bool disable);

■//! Returns whether simulation is disabled
■proto native bool GetIsSimulationDisabled();
■
■//! Returns simulation timestamp
■proto native int GetSimulationTimeStamp();

■//! Return animation phase of animation on object.
■proto native float GetAnimationPhase(string animation);
■
■//! Process animation on object. Animation is defined in config file. Wanted animation phase is set to pha
■proto native void SetAnimationPhase(string animation, float phase);
```

```
// ---------------------------------------------
// ------ EntityAI.c - START --------------------
// ---------------------------------------------





// ---------------------------------------------
// ------ EntityAI.c - END ----------------------
// ---------------------------------------------


// ---------------------------------------------
// ------ EntityLightSource.c - START --------------------
// ---------------------------------------------


enum LightSourceType
{
■NotDef,
■PointLight,
■SpotLight
};

class EntityLightSource extends Entity
{
■//----------------------------------------------------------------------------
■// Generic interface
■//----------------------------------------------------------------------------
■proto native void ■■■SetLightType(int pType); // This works only if it's caled in the light's constructor!
■proto native int  ■■■GetLightType();

■proto native void ■■■SetEnabled(bool pState);
■proto native bool ■■■IsEnabled();

■proto native void ■■■SetCastShadow(bool pState);
■proto native bool ■■■GetCastShadow();

■proto native bool ■■■EnableSpecular(bool pState);
■proto native bool ■■■EnableLinear(bool pState);

■proto native void ■■■SetPulseCoef(float pState);

■proto native void ■■■SetVisibleDuringDaylight(bool pState);
■proto native bool ■■■IsVisibleDuringDaylight();

■proto native void ■■■SetRadius(float pValue);
■proto native float ■■■GetRadius();
■
■//----------------------------------------------------------------------------
■// heat haze
■//----------------------------------------------------------------------------
■proto native void ■■■EnableHeatHaze(bool pState);
■
■proto native void ■■■SetHeatHazeRadius(float pValue);
■proto native float ■■■GetHeatHazeRadius();

■proto native void ■■■SetHeatHazePower(float pValue);
■proto native float ■■■GetHeatHazePower();
■
■//----------------------------------------------------------------------------
■// colors & brightness
```

```
// ---------------------------------------------
// ------ EntranceLight.c - START --------------------
// ---------------------------------------------


class EntranceLight extends PointLightBase
{
■protected float m_DefaultBrightness = 0.5;
■protected float m_DefaultRadius = 5.5;
■
■void EntranceLight()
■{
■■SetVisibleDuringDaylight(true);
■■SetRadiusTo(m_DefaultRadius);
■■SetBrightnessTo(m_DefaultBrightness);
■■FadeIn(1);
■■SetFadeOutTime(2);
■■SetFlareVisible(false);
■■SetCastShadow(false);
■■SetAmbientColor(0.9, 0.9, 0.7);
■■SetDiffuseColor(0.9, 0.9, 0.7);
■}
}

class EntranceLightStairs1 : EntranceLight
{
■void EntranceLightStairs1()
■{■■■■
■■SetRadiusTo(10);
■■SetBrightnessTo(0.5);
■■SetCastShadow(true);
■■SetAmbientColor(1.0, 0.4, 0.4);
■■SetDiffuseColor(1.0, 0.4, 0.4);
■}
}

class EntranceLightStairs2 : EntranceLight { }

class EntranceLightMain1 : EntranceLight { }

class EntranceLightMain2 : EntranceLight { }


// ---------------------------------------------
// ------ EntranceLight.c - END ----------------------
// ---------------------------------------------


// ---------------------------------------------
// ------ Environment.c - START --------------------
// ---------------------------------------------


class Environment
{
■const float RAIN_LIMIT_LOW■■= 0.05;

■const float WATER_LEVEL_HIGH■= 1.5;
■const float WATER_LEVEL_MID ■= 1.2;
■const float WATER_LEVEL_LOW ■= 0.5;
■const float WATER_LEVEL_NONE■= 0.15;
■
■protected float■■■■■m_WetDryTick; //ticks passed since last clothing wetting or drying
```

```
// ----------------------------------------------
// ------ EnVisual.c - START --------------------
// ----------------------------------------------


/**
 * \defgroup Visual Visual objects
 * @{
 */

//! Loads object from data, or gets it from cache. Object must be released when not used
proto native vobject GetObject(string name);

/*!
Release object. When there are not any other references, object is stored into cache and ready to be victe
\param object Object handle
\param flag If RF_RELEASE is used, the object is evicted immediatelly, if not used by anyone else
*/
proto native void ReleaseObject(vobject object, int flag = 0);

//! Returns number of frames, if the object is animation
proto native int GetNumAnimFrames(vobject anim);

//! Returns name of visual object
proto string vtoa(vobject vobj);

/**
 * \defgroup MeshObject Mesh object (XOB)
 * @{
 */

proto int GetObjectMaterials(vobject object, string materials[]);

// dynamic model creation (for dynamic aabb triggers)
//proto void CreateModel(IEntity ent, vector mins, vector maxs);
//proto void RemoveModel(IEntity ent);

//! Dynamic MeshObject
proto vobject CreateXOB(int nsurfaces, int nverts[], int numindices[], string materials[]);
proto void UpdateVertsEx(notnull IEntity ent, int surf, vector verts[], float uv[]);
proto void UpdateIndices(vobject obj, int surf, int indices[]);

proto native void■SetBone(notnull IEntity ent, int bone, vector angles, vector trans, float scale);
proto native bool■SetBoneMatrix(notnull IEntity ent, int bone, vector mat[4]);
proto native void■SetBoneGlobal(notnull IEntity ent, int bone, vector mat[4]);
proto native bool■GetBoneMatrix(notnull IEntity ent, int bone, vector mat[4]);
proto native bool■GetBoneLocalMatrix(notnull IEntity ent, int bone, vector mat[4]);

proto native void■SetAnimFrame(notnull IEntity ent, int slot, float frame);

//! BoneMask == NULL means that all bits are set
//! WARNING: Non-managed, needs manual delete call, should not be ref'd
class BoneMask
{
■int Mask[8]
}

enum AnimFlags
{
■//! animation is played only once and then if freezes at the last frame. EntityEvent.ANIMEND is called
■ONCE,
/*! defaultne zustava animace pri prehravani a AF_ONCE po skonceni na posledni frame "zamrzla", dokud
 nebo neni kanal vynulovan. Pokud se nastavi AF_BLENDOUT, postara se engine o vyhozeni animace sa
```

```
// ---------------------------------------------
// ------ EnVRDevice.c - START --------------------
// ---------------------------------------------


#ifdef GAME_TEMPLATE

/**
* \defgroup VRDeviceAPI generic and platform specific devices
* @{
*/

enum VRDeviceType
{
■DEVICE_TYPE_OCULUS,
■DEVICE_TYPE_PS4
}

enum VRStatus
{
■VR_STATUS_UNKNOWN,■■■■■//< Unknown state, probably uninitialized.
■VR_STATUS_VISIBLE,■■■■■//< The HMD is being used for rendering.
■VR_STATUS_PRESENT,■■■■■//< The HMD port is open.
■VR_STATUS_MOUNTED,■■■■■//<■The HMD is mounted on users head.
■VR_STATUS_DISPLAY_LOST,■■■//< The HMD was present and disappeared.
■VR_STATUS_SHOULD_QUIT,■■■//< Application requested exit.
■VR_STATUS_SHOULD_RECENTER,■//< The HMD Recenter request event was triggered.
■VR_STATUS_TRACKED,■■■■■//< The tracking data for the HMD are up to date.
■VR_STATUS_CALIBRATING,■■■//< The HMD is being calibrated.
}

enum VRHandEnum
{
■VR_HAND_LEFT, //< HMD controller left hand.
■VR_HAND_RIGHT //< HMD controller right hand.
}

enum VREye
{
■VR_EYE_LEFT, // HMD left eye.
■VR_EYE_RIGHT // HMD right eye.
}

/*!
■VRDevice interface common for all VR implementations.

■DO NOT INHERIT FROM THIS CLASS!
*/
class VRDevice : Managed
{
■/*!
■■\brief returns VRDevice instance.
■■\return device instance.
■*/
■proto native static VRDevice GetInstance();

■/*!
■■\brief returns status flags.

■■\return current device state flags.
■*/
■proto native VRStatus ■■■■GetStatusFlags();
■/*!
```

```
// ---------------------------------------------
// ------ EnWidgets.c - START --------------------
// ---------------------------------------------


/**
 * \defgroup WidgetAPI Widget UI system
 * @{
 */

#ifdef DOXYGEN
/** @name WidgetType
 *  Following WidgetType constants are available to script
 */
///@{

■//!Single-line text. See enf::TextWidget
■TextWidgetTypeID,
■//!Multi-line text. See enf::MultilineTextWidget
■MultilineTextWidgetTypeID,
■//!Multi-line edit box. See enf::MultilineTextWidget
■MultilineEditBoxWidgetTypeID,
■//!Multi-line text with images in text. See enf::RichTextWidget
■RichTextWidgetTypeID,
■//! Render target for enf::BaseWorld. See enf::RenderTargetWidget
■RenderTargetWidgetTypeID,
■//! Picture, or multiple picture. See enf::ImageWidget
■ImageWidgetTypeID,
■//!Console. See enf::ConsoleWidget
■ConsoleWidgetTypeID,
■//!Video player. See enf::VideoWidget
■VideoWidgetTypeID,
■//! Texture used as render target for children widgets. See enf::RTTextureWidget
■RTTextureWidgetTypeID,
■//! Dummy frame, used as hierarchy node and clipper
■FrameWidgetTypeID,
■//! Dummy frame, used for embedding another layout and as hierarchy node and clipper
■EmbededWidgetTypeID,
■ButtonWidgetTypeID,
■CheckBoxWidgetTypeID,
■WindowWidgetTypeID,
■ComboBoxWidgetTypeID,
■SimpleProgressBarWidgetTypeID,
■ProgressBarWidgetTypeID,
■SliderWidgetTypeID,
■BaseListboxWidgetTypeID,
■TextListboxWidgetTypeID,
■GenericListboxWidgetTypeID,
■EditBoxWidgetTypeID,
■WorkspaceWidgetTypeID,
■GridSpacerWidgetTypeID,
■WrapSpacerWidgetTypeID,
■ScrollWidgetTypeID,■
///@}
#else
■typedef TypeID WidgetType;
#endif

typedef TypeID EventType;

enum WidgetFlags
{
■SOURCEALPHA, //< takes alpha from texture * alpha from color. If not set, considers texture as non-tran
```

```
// ----------------------------------------------
// ------ EnWorld.c - START --------------------
// ----------------------------------------------


/**■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■ /**
 * \defgroup World World
 * @{
 */

//----------------------------------------------
/**
 * \defgroup WorldCommon World
 * @{
 */

typedef int[] WorldHandle;

proto native float GetWorldTime();

/*! Sets current world. It allows to work with entities outside world processing
// return previous world
*/
proto native WorldHandle SetCurrentWorld(WorldHandle world);

//proto native void SchedulePreload(vector pos, float radius);

proto native IEntity FindEntityByName(IEntity worldEnt, string name);
proto native IEntity FindEntityByID(IEntity worldEnt, int ID);

//!returns number of active (simulated) Entities in the world
proto native int GetNumActiveEntities(IEntity worldEntity);
//!returns active entity
proto native IEntity GetActiveEntity(IEntity worldEntity, int index);
//@}

//----------------------------------------------
/**
 * \defgroup Camera Camera
 * @{
 */

enum CameraType
{
■PERSPECTIVE,
■ORTHOGRAPHIC
};

//! sets which camera will be a listener (for sound engine)
proto native void SetListenerCamera(int camera);

/*!
Changes camera position
\param cam Index of camera
\param origin■position
\param angle■orientation
*/
proto native void SetCamera(int cam, vector origin, vector angle);

//! Changes camera matrix
proto native void SetCameraEx(int cam, const vector mat[4]);

//!Returns current camera transformation
```

```
// ---------------------------------------------
// ------ Epinephrine.c - START --------------------
// ---------------------------------------------


class Epinephrine: Inventory_Base
{
■override void SetActions()
■{
■■super.SetActions();
■■
■■AddAction(ActionInjectEpinephrineTarget);
■■AddAction(ActionInjectEpinephrineSelf);
■}
■
■override void OnApply(PlayerBase player)
■{
■■if (!player)
■■■return;
■■if( player.GetModifiersManager().IsModifierActive(eModifiers.MDF_EPINEPHRINE ) )//effectively resets
■■{
■■■player.GetModifiersManager().DeactivateModifier( eModifiers.MDF_EPINEPHRINE );
■■}
■■player.GetModifiersManager().ActivateModifier( eModifiers.MDF_EPINEPHRINE );
■}
};



// ---------------------------------------------
// ------ Epinephrine.c - END ---------------------
// ---------------------------------------------


// ---------------------------------------------
// ------ EpinephrineMdfr.c - START --------------------
// ---------------------------------------------


class EpinephrineMdfr: ModifierBase
{
■const int LIFETIME = 60;
■const float STAMINA_DEPLETION_MULTIPLIER = 0;
■override void Init()
■{
■■m_TrackActivatedTime = true;
■■m_IsPersistent = true;
■■m_ID ■■■■■= eModifiers.MDF_EPINEPHRINE;
■■m_TickIntervalInactive ■= DEFAULT_TICK_TIME_INACTIVE;
■■m_TickIntervalActive ■= 1;
■■DisableActivateCheck();
■}

■override bool ActivateCondition(PlayerBase player)
■{
■■return false;
■}
■
■override void OnReconnect(PlayerBase player)
■{
■■OnActivate(player);
■}
■
```

```
// ---------------------------------------------
// ------ EPlayerStates.c - START --------------------
// ---------------------------------------------


enum EPlayerStates
{
■ALIVE,
■DEAD,
}


// ---------------------------------------------
// ------ EPlayerStates.c - END ----------------------
// ---------------------------------------------


// ---------------------------------------------
// ------ EpoxyPutty.c - START --------------------
// ---------------------------------------------


class EpoxyPutty: Inventory_Base
{
■override void SetActions()
■{
■■super.SetActions();
■■AddAction(ActionRepairCarPart); //Heals ONLY global health for now
■■AddAction(ActionRepairCarChassis);
■}
};


// ---------------------------------------------
// ------ EpoxyPutty.c - END ----------------------
// ---------------------------------------------


// ---------------------------------------------
// ------ EPulseType.c - START --------------------
// ---------------------------------------------


enum EPulseType
{
■REGULAR,
■IRREGULAR,
}


// ---------------------------------------------
// ------ EPulseType.c - END ----------------------
// ---------------------------------------------


// ---------------------------------------------
// ------ ERPCs.c - START --------------------
// ---------------------------------------------


enum ERPCs
{
■PB_START = -1
```

```
// ---------------------------------------------
// ------ ErrorHandlerModule.c - START --------------------
// ---------------------------------------------


//-----------------------------------------------------------------------------
// Definition
//-----------------------------------------------------------------------------
//! Definition and API of an ErrorHandlerModule - Do not insert any logic here! (as this class is not moddab
class ErrorHandlerModule
{
■//! Returns the category the module handles.
■proto native ErrorCategory GetCategory();
■
■//! Set the category the module handles.
■proto native void SetCategory(ErrorCategory category);
■
■//! Event that gets triggered when an error of the owned category is thrown.
■protected void OnErrorThrown(int errorCode, owned string additionalInfo = "")
■{
■■#ifdef DEVELOPER
■■Print(ErrorModuleHandler.GetErrorHex(errorCode));
■■#endif
■}
■
■//! Retrieve the message shown on Client
■string GetClientMessage(int errorCode, string additionalInfo = "")
■{
■■return GetSimpleMessage(errorCode, additionalInfo);
■}
■
■//! Retrieve the message shown on Client
■string GetLastClientMessage(int errorCode)
■{
■■return GetSimpleMessage(errorCode);
■}
■
■//! Retrieve the message shown on Server
■string GetServerMessage(int errorCode, string additionalInfo = "")
■{
■■return GetSimpleMessage(errorCode, additionalInfo);
■}
■
■//! Retrieve the message shown on Server
■string GetLastServerMessage(int errorCode)
■{
■■return GetSimpleMessage(errorCode);
■}
■
■//! Simple message of just code and info
■string GetSimpleMessage(int errorCode, string additionalInfo = "")
■{
■■return string.Format("[%1]: %2", ErrorModuleHandler.GetErrorHex(errorCode), additionalInfo);
■}
■
■//! Event called by ErrorModuleHandler
■void OnEvent(EventType eventTypeId, Param params)
■{
■}
}

//-----------------------------------------------------------------------------
// Script override
// ---------------------------------------------
// ------ ErrorModuleHandler.c - START --------------------
// ---------------------------------------------


/**
  \brief ErrorCategory - To decide what ErrorHandlerModule needs to be called and easily identify where it
*/
enum ErrorCategory
{
■Unknown/* = -1*/,
■//! Generic error group
■Generic,
■//! Error group for when something went wrong while trying to connect to the server
■ConnectErrorClient,
■//! Error group for when the Client did connect to the Server, but the server rejected the connection
■ConnectErrorServer,
■//! Error group for connect errors thrown from Script
■ConnectErrorScript,
■//! Error group for when Client is kicked from server
■ClientKicked,
■//! Error group for BIOS errors
■BIOSError,
};

/**
  \brief The error handler itself, for managing and distributing errors to modules
■* Manages the ErrorHandlerModule instances inserted in Init.
■* API comes with several functions to Create, Throw and extract data from error codes.
■* The format used is an int which is made up of two shorts, one that holds the category and one that hold
■* Therefore when looking at an error code, it is much easier to identify when looking at the hex value.
*/
class ErrorModuleHandler
{
■/**
■\brief Creates and throws the error code, sending it to the handler of the category.
■■\param category \p ErrorCategory Category the error is thrown from
■■\param code \p int The code that the error belongs to inside the category between [-32768, 32767]
■■\param additionalInfo \p string Any additional info regarding the error, usually data
■■\return \p int The full error code
■■@code
■■■int errorCode = ErrorModuleHandler.ThrowError( ErrorCategory.ConnectErrorClient, -1 );
■■■Print( errorCode );

■■■>> errorCode = 196607
■■@endcode
■*/■
■static proto int ■■■■■■ThrowError(ErrorCategory category, int code, string additionalInfo = "");
■
■/**
■\brief Throws the error code and sends it to the handler of the category.
■■\param errorCode \p int The full error code
■■\param additionalInfo \p string Any additional info regarding the error, usually data
■■\return \p int The full error code
■■@code
■■■int errorCode = ErrorModuleHandler.ThrowErrorCode( 0x0002FFFF );
■■■Print( errorCode );

■■■>> errorCode = 196607
■■@endcode
■*/
■static proto int■■■■■■ThrowErrorCode(int errorCode, string additionalInfo = "");
■
```

```
// ---------------------------------------------
// ------ ErrorProperties.c - START --------------------
// ---------------------------------------------


//! Class which holds the properties and handling of an error
class ErrorProperties
{
■const string EP_HEADER_FORMAT_STRING = "%1 (%2)"; //!< Formating for header (%1 = Header; %2
■const string EP_MESSAGE_FORMAT_STRING = "%1\n(%2)"; //!< Formating for message (%1 = Messa
■
■protected string m_Message; //!< Message which will appear on Client
■protected string m_ServerMessage; //!< Message which will appear on Server
■
■void ErrorProperties(string message, string serverMessage)
■{
■■m_Message = message;
■■m_ServerMessage = serverMessage;
■}
■■
■void HandleError(int errorCode, string additionalInfo = "") {}
■
■string GetClientMessage(string additionalInfo = "")
■{
■■if ( additionalInfo != "" )
■■■return string.Format(EP_MESSAGE_FORMAT_STRING, m_Message, additionalInfo);
■■else
■■■return m_Message;
■}
■
■string GetServerMessage(string additionalInfo = "")
■{
■■if ( additionalInfo != "" )
■■■return string.Format(EP_MESSAGE_FORMAT_STRING, m_ServerMessage, additionalInfo);
■■else
■■■return m_ServerMessage;
■}
}

//! Error which shows a generic Dialogue UI
class DialogueErrorProperties : ErrorProperties
{
■protected string m_Header;
■protected int m_DialogButtonType;
■protected int m_DefaultButton;
■protected int m_DialogMeaningType;
■protected UIScriptedMenu m_Handler;
■protected bool m_DisplayAdditionalInfo;
■
■void DialogueErrorProperties(string message, string serverMessage, string header, UIScriptedMenu han
■{
■■m_Header = header;
■■m_DialogButtonType = dialogButtonType;
■■m_DefaultButton = defaultButton;
■■m_DialogMeaningType = dialogMeaningType;
■■m_Handler = handler;
■■m_DisplayAdditionalInfo = displayAdditionalInfo;
■}
■
■override void HandleError(int errorCode, string additionalInfo = "")
■{
#ifdef NO_GUI
■■return; //do not display error if GUI is disabled
```

```
// ---------------------------------------------
// ------ EStaminaConsumers.c - START --------------------
// ---------------------------------------------


enum EStaminaConsumers
{
■HOLD_BREATH,
■MELEE_HEAVY,
■MELEE_EVADE,
■SPRINT,
■JUMP,
■VAULT,
■CLIMB,
■ROLL,
■DROWN,
}


// ---------------------------------------------
// ------ EStaminaConsumers.c - END ----------------------
// ---------------------------------------------


// ---------------------------------------------
// ------ EStaminaModifiers.c - START --------------------
// ---------------------------------------------


enum EStaminaModifiers
{
■HOLD_BREATH,
■JUMP,
■MELEE_LIGHT,
■MELEE_HEAVY,
■MELEE_EVADE,
■OVERALL_DRAIN,
■VAULT,
■CLIMB,
■ROLL,
■DROWN,
}


// ---------------------------------------------
// ------ EStaminaModifiers.c - END ----------------------
// ---------------------------------------------


// ---------------------------------------------
// ------ EStatLevels.c - START --------------------
// ---------------------------------------------


enum EStatLevels
{
■GREAT,
■HIGH,
■MEDIUM,
■LOW,
■CRITICAL,
}
```

```
// ---------------------------------------------
// ------ ESyncEvent.c - START --------------------
// ---------------------------------------------


enum ESyncEvent
{
■PlayerList,
■EntityKill,
■PlayerIgnateFireplayce
}


// ---------------------------------------------
// ------ ESyncEvent.c - END ----------------------
// ---------------------------------------------


// ---------------------------------------------
// ------ ETimeOfDay.c - START --------------------
// ---------------------------------------------


enum ETimeOfDay
{
■Day,
■Night,
}


// ---------------------------------------------
// ------ ETimeOfDay.c - END ----------------------
// ---------------------------------------------


// ---------------------------------------------
// ------ Events.c - START --------------------
// ---------------------------------------------


/**@enum■WeaponEventID
 * @brief■identifier for events.
 * mainly for rpc purposes
 **/
enum WeaponEventID
{
■UNKNOWN,
■MECHANISM,
■TRIGGER,
■TRIGGER_JAM,
■TRIGGER_AUTO_START,
■TRIGGER_AUTO_END,
■LOAD1_BULLET,
■CONTINUOUS_LOADBULLET_START,
■CONTINUOUS_LOADBULLET_END,
■UNJAM,
■ATTACH_MAGAZINE,
■UNJAMMING_FAILED_TIMEOUT,
■UNJAMMING_TIMEOUT,
■DETACH_MAGAZINE,
■SWAP_MAGAZINE,
■HUMANCOMMAND_ACTION_FINISHED,
■HUMANCOMMAND_ACTION_ABORTED
```

```
// ---------------------------------------------
// ------ EWaterLevels.c - START --------------------
// ---------------------------------------------


enum EWaterLevels
{
■LEVEL_LOW,
■LEVEL_CROUCH, //high enough to disallow prone
■LEVEL_ERECT, //high enough to disallow prone and crouch
■LEVEL_SWIM_START //high enough
};


// ---------------------------------------------
// ------ EWaterLevels.c - END ----------------------
// ---------------------------------------------


// ---------------------------------------------
// ------ ExhaustSmoke.c - START --------------------
// ---------------------------------------------


class EffExhaustSmoke : EffVehicleSmoke
{
■override void SetParticleStateLight()
■{
■■SetParticleState( ParticleList.HATCHBACK_EXHAUST_SMOKE );
■■
■■Car parent = Car.Cast( GetAttachmentParent() );
■■Particle p = GetParticle();
■■
■■if ( parent && p )
■■{
■■■float speed = parent.GetSpeedometerAbsolute();
■■■float lifetime_scale;
■■■■
■■■if (speed < 100)
■■■■lifetime_scale = (100 - speed) / 100;
■■■else
■■■■lifetime_scale = 0.1;
■■■■
■■■float birthrate_scale = 1 + (speed * 0.1 );

■■■p.ScaleParticleParamFromOriginal( EmitorParam.LIFETIME, lifetime_scale );
■■■p.ScaleParticleParamFromOriginal( EmitorParam.LIFETIME_RND, lifetime_scale );
■■■p.ScaleParticleParamFromOriginal( EmitorParam.BIRTH_RATE, birthrate_scale );
■■■p.ScaleParticleParamFromOriginal( EmitorParam.BIRTH_RATE_RND, birthrate_scale );
■■}
■}
}


// ---------------------------------------------
// ------ ExhaustSmoke.c - END ----------------------
// ---------------------------------------------
// ---------------------------------------------
// ------ Explosion.c - START --------------------
// ---------------------------------------------


// WIP

class Explosion
{
■void SpawnEffect( vector position, Effect eff, vector pos, vector ori)
■{
■■SEffectManager.PlayInWorld(eff, pos);
■}
}

class ExplosionTest : House
{
■ref Timer m_Delay;
■
■void ExplosionTest()
■{
■■m_Delay = new Timer;
■■m_Delay.Run(1, this, "ExplodeNow", null, false);
■}
■
■void ExplodeNow()
■{
■■Explode(DT_EXPLOSION, "Explosion_NonLethal");
■}
}



// ---------------------------------------------
// ------ Explosion.c - END ----------------------
// ---------------------------------------------


// ---------------------------------------------
// ------ ExplosivesBase.c - START --------------------
// ---------------------------------------------


class ExplosiveLight : PointLightBase
{
■protected static float m_DefaultBrightness■= 10;
■protected static float m_DefaultRadius■■= 30;
■
■void ExplosiveLight()
■{
■■SetVisibleDuringDaylight(false);
■■SetRadiusTo(m_DefaultRadius);
■■SetBrightnessTo(m_DefaultBrightness);
■■SetFlareVisible(false);
■■SetAmbientColor(1.0, 1.0, 0.3);
■■SetDiffuseColor(1.0, 1.0, 0.3);
■■SetLifetime(0.15);
■■SetDisableShadowsWithinRadius(-1);
■}■
}

class ExplosivesBase : ItemBase
{
```

```
// ---------------------------------------------
// ------ ExtinguishTorch.c - START --------------------
// ---------------------------------------------


class ExtinguishTorch extends RecipeBase■
{■
■override void Init()
■{
■■m_Name = "#extinguish";
■■m_IsInstaRecipe = false;//should this recipe be performed instantly without animation
■■m_AnimationLength = 0.5;//animation length in relative time units
■■m_Specialty = 0.02;// value > 0 for roughness, value < 0 for precision
■■
■■
■■//conditions
■■m_MinDamageIngredient[0] = -1;//-1 = disable check
■■m_MaxDamageIngredient[0] = 3;//-1 = disable check
■■
■■m_MinQuantityIngredient[0] = -1;//-1 = disable check
■■m_MaxQuantityIngredient[0] = -1;//-1 = disable check
■■
■■m_MinDamageIngredient[1] = -1;//-1 = disable check
■■m_MaxDamageIngredient[1] = 3;//-1 = disable check
■■//TODO SS 2.0 should modify MinQuantity of liquid
■■m_MinQuantityIngredient[1] = 100;//-1 = disable check
■■m_MaxQuantityIngredient[1] = -1;//-1 = disable check
■■//----------------------------------------------------------------------------------------------------------------------
■■
■■//INGREDIENTS
■■//ingredient 1
■■InsertIngredient(0,"Torch");//you can insert multiple ingredients this way
■■InsertIngredient(0,"Broom");//you can insert multiple ingredients this way

■■m_IngredientAddHealth[0] = 0;// 0 = do nothing
■■m_IngredientSetHealth[0] = -1; // -1 = do nothing
■■m_IngredientAddQuantity[0] = 0;// 0 = do nothing
■■m_IngredientDestroy[0] = false;//true = destroy, false = do nothing
■■m_IngredientUseSoftSkills[0] = false;// set 'true' to allow modification of the values by softskills on this i
■■
■■//ingredient 2
■■InsertIngredient(1,"Pot");//you can insert multiple ingredients this way
■■InsertIngredient(1,"CanisterGasoline");//you can insert multiple ingredients this way
■■InsertIngredient(1,"DisinfectantAlcohol");//you can insert multiple ingredients this way
■■InsertIngredient(1,"Canteen");//you can insert multiple ingredients this way
■■InsertIngredient(1,"WaterBottle");//you can insert multiple ingredients this way
■■InsertIngredient(1,"Vodka");//you can insert multiple ingredients this way
■■InsertIngredient(1,"WaterPouch_ColorBase");//you can insert multiple ingredients this way
■■InsertIngredient(1,"Barrel_ColorBase");//you can insert multiple ingredients this way
■■
■■m_IngredientAddHealth[1] = 0;// 0 = do nothing
■■m_IngredientSetHealth[1] = -1; // -1 = do nothing
■■m_IngredientAddQuantity[1] = -100;// 0 = do nothing
■■m_IngredientDestroy[1] = false;// false = do nothing
■■m_IngredientUseSoftSkills[1] = true;// set 'true' to allow modification of the values by softskills on this in
■■//----------------------------------------------------------------------------------------------------------------------
■■
■■//result1
■■//AddResult("");//add results here

■■m_ResultSetFullQuantity[0] = false;//true = set full quantity, false = do nothing
■■m_ResultSetQuantity[0] = -1;//-1 = do nothing
■■m_ResultSetHealth[0] = -1;//-1 = do nothing
```

```
// ---------------------------------------------
// ------ EyeMask_Colorbase.c - START --------------------
// ---------------------------------------------


class EyeMask_ColorBase : Clothing
{
■//=============================================================
■// IGNITION ACTION
■//=============================================================
■override bool HasFlammableMaterial()
■{
■■return true;
■}
■
■override bool CanBeIgnitedBy(EntityAI igniter = null)
■{
■■return !GetHierarchyParent();
■}
■
■override bool CanIgniteItem(EntityAI ignite_target = null)
■{
■■return false;
■}
■
■override void OnIgnitedTarget(EntityAI ignited_item) {}
■
■override void OnIgnitedThis(EntityAI fire_source)
■{
■■Fireplace.IgniteEntityAsFireplace(this, fire_source);
■}

■override bool IsThisIgnitionSuccessful(EntityAI item_source = null)
■{
■■return Fireplace.CanIgniteEntityAsFireplace(this);
■}
■//=============================================================
■
■override bool CanPutAsAttachment(EntityAI parent)
■{
■■if (!super.CanPutAsAttachment(parent))
■■■return false;
■■
■■Clothing mask = Clothing.Cast(parent.FindAttachmentBySlotName("Mask"));
■■if (mask && mask.ConfigGetBool("noEyewear"))
■■{
■■■return false;
■■}
■■
■■return true;
■}
■
■override void SetActions()
■{
■■super.SetActions();

■■AddAction(ActionCreateIndoorFireplace);
■■AddAction(ActionCreateIndoorOven);
■■AddAction(ActionAttach);
■■AddAction(ActionDetach);
■}
}
```

```
// ---------------------------------------------
// ------ EyePatch_improvised.c - START --------------------
// ---------------------------------------------


class EyePatch_Improvised extends Clothing
{
■override array<int> GetEffectWidgetTypes()
■{
■■return {EffectWidgetsTypes.EYEPATCH_OCCLUDER};
■}
■
■override bool CanPutAsAttachment(EntityAI parent)
■{
■■if (!super.CanPutAsAttachment(parent))
■■■return false;
■■
■■Clothing mask = Clothing.Cast(parent.FindAttachmentBySlotName("Mask"));
■■if (mask && mask.ConfigGetBool("noEyewear"))
■■{
■■■return false;
■■}
■■
■■return true;
■}
■
■override void SetActions()
■{
■■super.SetActions();
■■AddAction(ActionWringClothes);
■}
};




// ---------------------------------------------
// ------ EyePatch_improvised.c - END ----------------------
// ---------------------------------------------


// ---------------------------------------------
// ------ FaceCover_improvised.c - START --------------------
// ---------------------------------------------


class FaceCover_Improvised extends Clothing
{
■override void SetActions()
■{
■■super.SetActions();
■■AddAction(ActionWringClothes);
■}
};



// ---------------------------------------------
// ------ FaceCover_improvised.c - END ----------------------
// ---------------------------------------------
// ---------------------------------------------
// ------ FAL.c - START --------------------
// ---------------------------------------------


class FAL_Base : RifleBoltLock_Base
{
■override RecoilBase SpawnRecoilObject()
■{
■■return new FALRecoil(this);
■}
■■
■//Debug menu Spawn Ground Special
■override void OnDebugSpawn()
■{
■■GameInventory inventory = GetInventory();

■■inventory.CreateInInventory( "Fal_OeBttstck" );
■■inventory.CreateInInventory( "M4_T3NRDSOptic" );■
■■inventory.CreateInInventory( "Battery9V" );
■■
■■SpawnAttachedMagazine("Mag_FAL_20Rnd");
■}
};


// ---------------------------------------------
// ------ FAL.c - END ----------------------
// ---------------------------------------------


// ---------------------------------------------
// ------ FALRecoil.c - START --------------------
// ---------------------------------------------


class FALRecoil: RecoilBase
{
■override void Init()
■{
■■vector point_1;
■■vector point_2;
■■vector point_3;
■■vector point_4;
■■point_1[0] = m_Player.GetRandomGeneratorSyncManager().GetRandomInRange(RandomGeneratorS
■■point_2[0] = m_Player.GetRandomGeneratorSyncManager().GetRandomInRange(RandomGeneratorS
■■point_3[0] = m_Player.GetRandomGeneratorSyncManager().GetRandomInRange(RandomGeneratorS
■■point_4[0] = m_Player.GetRandomGeneratorSyncManager().GetRandomInRange(RandomGeneratorS
■■m_HandsCurvePoints.Insert(point_1);//forms a 2 dimensional spline(z is ignored)
■■m_HandsCurvePoints.Insert(point_2);
■■m_HandsCurvePoints.Insert(point_3);
■■m_HandsCurvePoints.Insert(point_4);
■■m_HandsCurvePoints.Insert("0 0 0");
■■m_HandsOffsetRelativeTime = 1;
■■
■■m_MouseOffsetRangeMin = 50;//in degrees min
■■m_MouseOffsetRangeMax = 120;//in degrees max
■■m_MouseOffsetDistance = 1.8;//how far should the mouse travel
■■m_MouseOffsetRelativeTime = 0.5;//[0..1] a time it takes to move the mouse the required distance rela
■
■■m_CamOffsetDistance = 0.01;
■■m_CamOffsetRelativeTime = 1;■
■}
```

```
// ---------------------------------------------
// ------ FAMAS.c - START --------------------
// ---------------------------------------------


class Famas_Base : RifleBoltFree_Base
{
■override RecoilBase SpawnRecoilObject()
■{
■■return new FamasRecoil(this);
■}■■
■■
■//Debug menu Spawn Ground Special
■override void OnDebugSpawn()
■{
■■SpawnAttachedMagazine("Mag_Famas_25Rnd");
■}
■
■/*override int GetWeaponSpecificCommand(int weaponAction ,int subCommand)
■{
■■if ( weaponAction == WeaponActions.RELOAD)
■■{
■■■switch (subCommand)
■■■{
■■■■case WeaponActionReloadTypes.RELOADSRIFLE_MAGAZINE_BULLET:
■■■■■return WeaponActionReloadTypes.RELOADRIFLE_MAGAZINE_BULLET;
■■■■
■■■■case WeaponActionReloadTypes.RELOADSRIFLE_NOMAGAZINE_BULLET:
■■■■■return WeaponActionReloadTypes.RELOADRIFLE_NOMAGAZINE_BULLET;
■■■■
■■■■case WeaponActionReloadTypes.RELOADSRIFLE_MAGAZINE_NOBULLET:
■■■■■return WeaponActionReloadTypes.RELOADRIFLE_MAGAZINE_NOBULLET;
■■■■
■■■■case WeaponActionReloadTypes.RELOADSRIFLE_NOMAGAZINE_NOBULLET:
■■■■■return WeaponActionReloadTypes.RELOADRIFLE_NOMAGAZINE_NOBULLET;
■■■■
■■■■default:
■■■■■return subCommand;
■■■}
■■
■■}
■■return subCommand;
■}*/
};

class FAMAS : Famas_Base {};
class SawedOffFAMAS : Famas_Base {};


// ---------------------------------------------
// ------ FAMAS.c - END ----------------------
// ---------------------------------------------


// ---------------------------------------------
// ------ FamasRecoil.c - START --------------------
// ---------------------------------------------


class FamasRecoil: RecoilBase
{
■override void Init()
■{
```

```
// ---------------------------------------------
// ------ FangeKnife.c - START --------------------
// ---------------------------------------------


class FangeKnife extends ToolBase
{
■override bool IsMeleeFinisher()
■{
■■return true;
■}
■
■override array<int> GetValidFinishers()
■{
■■return {EMeleeHitType.FINISHER_LIVERSTAB,EMeleeHitType.FINISHER_NECKSTAB};
■}
■
■override void SetActions()
■{
■■super.SetActions();
■■
■■AddAction(ActionBurnSewTarget);
■■AddAction(ActionUnrestrainTarget);
■■AddAction(ActionSkinning);
■■AddAction(ActionMineBush);
■■AddAction(ActionMineTreeBark);
■■AddAction(ActionBurnSewSelf);
■■AddAction(ActionDigWorms);
■■AddAction(ActionShaveTarget);
■■AddAction(ActionShave);
■}
}


// ---------------------------------------------
// ------ FangeKnife.c - END ---------------------
// ---------------------------------------------


// ---------------------------------------------
// ------ FarmingHoe.c - START --------------------
// ---------------------------------------------


class FarmingHoe: ItemBase
{
■override bool CanMakeGardenplot()
■{
■■return true;
■}
■
■override void SetActions()
■{
■■super.SetActions();
■■
■■AddAction(ActionClapBearTrapWithThisItem);
■■AddAction(ActionTogglePlaceObject);
■■AddAction(ActionDigGardenPlot);
■■AddAction(ActionDismantleGardenPlot);
■■AddAction(ActionDismantlePart);
■■AddAction(ActionBuildPart);
■■AddAction(ActionBuryBody);
■■AddAction(ActionBuryAshes);
```

```
// ---------------------------------------------
// ------ Fatigue.c - START --------------------
// ---------------------------------------------


class FatigueMdfr: ModifierBase
{■
■private float m_Time;
■private float m_NextEvent;
■
■static const float FATIGUE_EVENT_INTERVAL_MIN = 5;
■static const float FATIGUE_EVENT_INTERVAL_MAX = 12;
■static const float STAMINA_RECOVERY_MULTIPLIER = 0.33;
■static const float STAMINA_DEPLETION_MULTIPLIER = 1.33;
■
■override void Init()
■{
■■m_TrackActivatedTime = false;
■■m_ID ■■■■■= eModifiers.MDF_FATIGUE;
■■m_TickIntervalInactive ■= DEFAULT_TICK_TIME_INACTIVE_LONG;
■■m_TickIntervalActive ■= DEFAULT_TICK_TIME_ACTIVE;
■}
■
■override bool ActivateCondition(PlayerBase player)
■{
■■return (player.GetModifiersManager().IsModifierActive(eModifiers.MDF_WOUND_INFECTION1) || play
■}

■override void OnActivate(PlayerBase player)
■{
■■if( player.m_NotifiersManager )
■■■player.m_NotifiersManager.ActivateByType(eNotifiers.NTF_FEVERISH);
■■

■■
■■player.GetStaminaHandler().ActivateRecoveryModifier(EStaminaMultiplierTypes.FATIGUE);
■■player.GetStaminaHandler().ActivateDepletionModifier(EStaminaMultiplierTypes.FATIGUE);

■}

■override void OnReconnect(PlayerBase player)
■{
■■this.OnActivate(player);
■}


■override void OnDeactivate(PlayerBase player)
■{
■■//player.GetStaminaHandler().SetDepletionMultiplier(1);
■■player.GetStaminaHandler().DeactivateRecoveryModifier(EStaminaMultiplierTypes.FATIGUE);
■■player.GetStaminaHandler().DeactivateDepletionModifier(EStaminaMultiplierTypes.FATIGUE);
■}


■override bool DeactivateCondition(PlayerBase player)
■{
■■return  !ActivateCondition(player);
■}

■override void OnTick(PlayerBase player, float deltaT)
■{

■}
```

```
// ---------------------------------------------
// ------ Feet.c - START --------------------
// ---------------------------------------------


class MaleFeet_Base extends InventoryItem {};
class FemaleFeet_Base extends InventoryItem {};

class MaleAdamFeet extends MaleFeet_Base {};
class MaleBorisFeet extends MaleFeet_Base {};
class MaleCyrilFeet extends MaleFeet_Base {};
class MaleDenisFeet extends MaleFeet_Base {};
class MaleEliasFeet extends MaleFeet_Base {};
class MaleFrancisFeet extends MaleFeet_Base {};
class MaleGuoFeet extends MaleFeet_Base {};
class MaleHassanFeet extends MaleFeet_Base {};
class MaleIndarFeet extends MaleFeet_Base {};
class MaleJoseFeet extends MaleFeet_Base {};
class MaleKaitoFeet extends MaleFeet_Base {};
class MaleLewisFeet extends MaleFeet_Base {};
class MaleManuaFeet extends MaleFeet_Base {};
class MaleNikiFeet extends MaleFeet_Base {};
class MaleOliverFeet extends MaleFeet_Base {};
class MalePeterFeet extends MaleFeet_Base {};
class MaleQuinnFeet extends MaleFeet_Base {};
class MaleRolfFeet extends MaleFeet_Base {};
class MaleSethFeet extends MaleFeet_Base {};
class MaleTaikiFeet extends MaleFeet_Base {};
class MaleDeanFeet extends MaleFeet_Base {};

class FemaleEvaFeet extends FemaleFeet_Base {};
class FemaleFridaFeet extends FemaleFeet_Base {};
class FemaleGabiFeet extends FemaleFeet_Base {};
class FemaleHelgaFeet extends FemaleFeet_Base {};
class FemaleIrenaFeet extends FemaleFeet_Base {};
class FemaleJudyFeet extends FemaleFeet_Base {};
class FemaleKeikoFeet extends FemaleFeet_Base {};
class FemaleLindaFeet extends FemaleFeet_Base {};
class FemaleMariaFeet extends FemaleFeet_Base {};
class FemaleNaomiFeet extends FemaleFeet_Base {};


// ---------------------------------------------
// ------ Feet.c - END ----------------------
// ---------------------------------------------


// ---------------------------------------------
// ------ FeetCover_improvised.c - START --------------------
// ---------------------------------------------


class FeetCover_Improvised extends Clothing
{
■override void SetActions()
■{
■■super.SetActions();
■■AddAction(ActionWringClothes);
■}
};
```

```
// ---------------------------------------------
// ------ Fence.c - START --------------------
// ---------------------------------------------


class Fence extends BaseBuildingBase
{
■const int GATE_STATE_NONE ■■■■= 0;
■const int GATE_STATE_PARTIAL ■■■= 1;
■const int GATE_STATE_FULL ■■■■= 2;
■
■const string ATTACHMENT_SLOT_COMBINATION_LOCK ■= "Att_CombinationLock";
■const string SOUND_GATE_OPEN_START■■■■= "DoorWoodTowerOpen_SoundSet";
■const string SOUND_GATE_CLOSE_START■■■■= "DoorWoodTowerClose_start_SoundSet";
■const string SOUND_GATE_CLOSE_END■■■■= "DoorWoodTowerClose_end_SoundSet";

■//gate openining
■const float GATE_ROTATION_ANGLE_DEG ■■■= 100;
■const float GATE_ROTATION_TIME_APPROX■■■= 2000;■■//ms
■
■const float MAX_ACTION_DETECTION_ANGLE_RAD ■■= 1.3;■■//1.3 RAD = ~75 DEG
■const float MAX_ACTION_DETECTION_DISTANCE ■■= 2.0;■■//meters
■
■typename ATTACHMENT_WOODEN_LOG■■■= WoodenLog;
■typename ATTACHMENT_COMBINATION_LOCK■= CombinationLock;
■
■string ATTSLOT_CAMONET■■■■= "Wall_Camonet";
■string ATTSLOT_BARBEDWIRE_DOWN■■= "Wall_Barbedwire_1";
■string ATTSLOT_BARBEDWIRE_UP■■= "Wall_Barbedwire_2";
■
■//protected bool m_HasHinges■■■■= false;
■//protected bool m_GateFullyConstructed ■= false;
■protected bool m_ToDiscard ■■■■= false; //for legacy OnStoreLoad handling
■protected bool m_IsOpened ■■■■= false;
■protected bool m_IsOpenedClient■■■= false;
■protected int m_GateState ■■■■= 0;
■
■protected EffectSound m_SoundGate_Start;
■protected EffectSound m_SoundGate_End;
■
■void Fence()
■{
■■//synchronized variables
■■//RegisterNetSyncVariableBool( "m_HasHinges" );
■■//RegisterNetSyncVariableBool( "m_GateFullyConstructed" );
■■RegisterNetSyncVariableBool( "m_IsOpened" );
■■RegisterNetSyncVariableInt( "m_GateState" );
■}
■
■override string GetConstructionKitType()
■{
■■return "FenceKit";
■}
■
■override int GetMeleeTargetType()
■{
■■return EMeleeTargetType.NONALIGNABLE;
■}
■
■//Gate
■bool HasHinges()
■{
■■return m_GateState > GATE_STATE_NONE;
```

```
// ---------------------------------------------
// ------ FenceKit.c - START --------------------
// ---------------------------------------------


class FenceKit extends KitBase
{
■override bool CanReceiveAttachment(EntityAI attachment, int slotId)
■{
■■if ( !super.CanReceiveAttachment(attachment, slotId) )
■■■return false;
■■
■■ItemBase att = ItemBase.Cast(GetInventory().FindAttachment(slotId));
■■if (att)
■■■return false;
■■
■■return true;
■}

■//=================================================================
■// ADVANCED PLACEMENT
■//=================================================================■■■
■■
■override void OnPlacementComplete( Man player, vector position = "0 0 0", vector orientation = "0 0 0" )
■{
■■super.OnPlacementComplete( player, position, orientation );
■■
■■if ( GetGame().IsServer() )
■■{
■■■//Create fence
■■■
■■■Fence fence = Fence.Cast( GetGame().CreateObjectEx( "Fence", GetPosition(), ECE_PLACE_ON_S
■■■fence.SetPosition( position );
■■■fence.SetOrientation( orientation );
■■■
■■■//make the kit invisible, so it can be destroyed from deploy UA when action ends
■■■HideAllSelections();
■■■
■■■SetIsDeploySound( true );
■■}■
■}
■
■override bool DoPlacingHeightCheck()
■{
■■return true;
■}
■
■override float HeightCheckOverride()
■{
■■return 2.54;
■}
■
■override void DisassembleKit(ItemBase item)
■{
■■if (!IsHologram())
■■{
■■■ItemBase stick = ItemBase.Cast(GetGame().CreateObjectEx("WoodenStick",GetPosition(),ECE_PLA
■■■MiscGameplayFunctions.TransferItemProperties(this, stick);
■■■stick.SetQuantity(2);
■■■Rope rope = Rope.Cast(item);
■■■CreateRope(rope);
■■}
■}
```

```
// ---------------------------------------------
// ------ Fever.c - START --------------------
// ---------------------------------------------


class FeverMdfr: ModifierBase
{
■private float m_Time;
■private float m_NextEvent;
■
■static const float EVENT_INTERVAL_MIN = 12;
■static const float EVENT_INTERVAL_MAX = 18;
■
■
■override void Init()
■{
■■m_TrackActivatedTime = false;
■■m_ID ■■■■■= eModifiers.MDF_FEVER;
■■m_TickIntervalInactive ■= DEFAULT_TICK_TIME_INACTIVE;
■■m_TickIntervalActive ■= DEFAULT_TICK_TIME_ACTIVE_SHORT;
■}
■override bool ActivateCondition(PlayerBase player)
■{
■■return (player.GetModifiersManager().IsModifierActive(eModifiers.MDF_CHOLERA) || player.GetModifi
■}

■override void OnActivate(PlayerBase player)
■{
■■if( player.m_NotifiersManager )
■■■player.m_NotifiersManager.ActivateByType(eNotifiers.NTF_FEVERISH);
■■
■■player.GetSymptomManager().QueueUpSecondarySymptom(SymptomIDs.SYMPTOM_FEVERBLUR)
■}

■override void OnReconnect(PlayerBase player)
■{
■■this.OnActivate(player);
■}


■override void OnDeactivate(PlayerBase player)
■{
■■if( player.m_NotifiersManager )
■■■player.m_NotifiersManager.DeactivateByType(eNotifiers.NTF_FEVERISH);
■■player.GetSymptomManager().RemoveSecondarySymptom(SymptomIDs.SYMPTOM_FEVERBLUR);
■}


■override bool DeactivateCondition(PlayerBase player)
■{
■■return !ActivateCondition(player);
■}

■override void OnTick(PlayerBase player, float deltaT)
■{
■■float water_loss = deltaT * PlayerConstants.WATER_LOSS_FEVER;
■■player.GetStatWater().Add(-water_loss);
■■
■■m_Time += deltaT;
■
■■if ( m_Time >= m_NextEvent )
■■{
■■■m_Time = 0;
```

```
// ---------------------------------------------
// ------ FeverBlurState.c - START --------------------
// ---------------------------------------------


class FeverBlurSymptom extends SymptomBase
{
■Material m_MatGauss;

■float m_BlurDuration;
■float m_BlurStrength;
■float m_EffectTime;
■float m_EffectStartTime;
■float m_Time;
■protected PPERequester_FeverEffects ■m_Requester;
■
■const float BLUR_STRENGTH_MIN = 0.15;
■const float BLUR_STRENGTH_MAX = 0.25;
■
■const int BLUR_DURATION_TIME_MIN = 1.5;
■const int BLUR_DURATION_TIME_MAX = 2.5;
■
■const int MIN_TIME_BETWEEN_EFFECTS = 25.0;
■const int MAX_TIME_BETWEEN_EFFECTS = 35.0;

■//this is just for the Symptom parameters set-up and is called even if the Symptom doesn't execute, don't
■override void OnInit()
■{
■■m_SymptomType = SymptomTypes.SECONDARY;
■■m_Priority = 0;
■■m_ID = SymptomIDs.SYMPTOM_FEVERBLUR;
■■m_DestroyOnAnimFinish = true;
■■m_IsPersistent = false;
■■m_SyncToClient = true;
■■
■■if ( !GetGame().IsDedicatedServer() )
■■{
■■■Class.CastTo(m_Requester,PPERequesterBank.GetRequester(PPERequester_FeverEffects));
■■}
■}
■
■//!gets called every frame
■override void OnUpdateServer(PlayerBase player, float deltatime)
■{
■■//int i = 1 + 1;
■}

■
■override void OnUpdateClient(PlayerBase player, float deltatime)
■{
■■m_Time += deltatime;
■■if ( m_EffectStartTime <= 0 )
■■{
■■■m_EffectStartTime = Math.RandomFloatInclusive(MIN_TIME_BETWEEN_EFFECTS, MAX_TIME_B
■■■m_BlurDuration = Math.RandomFloatInclusive(BLUR_DURATION_TIME_MIN, BLUR_DURATION_T
■■■m_BlurStrength = Math.RandomFloat(BLUR_STRENGTH_MIN, BLUR_STRENGTH_MAX);
■■■//PrintString("m_BlurDuration=" +m_BlurDuration.ToString());
■■}
■■
■■if ( m_EffectStartTime > 0 && m_Time > m_EffectStartTime )
■■{
■■■m_EffectTime += deltatime / m_BlurDuration;
■■■float cos_value = Math.Sin(m_EffectTime  * Math.PI);
```

```
// ---------------------------------------------
// ------ FeverNotfr.c - START --------------------
// ---------------------------------------------


class FeverNotfr: NotifierBase
{
■void FeverNotfr(NotifiersManager manager)
■{
■■m_Active = false;
■}

■override int GetNotifierType()
■{
■■return eNotifiers.NTF_FEVERISH;
■}

■override void DisplayBadge()
■{
■//■Print("FEVER");
■■////GetVirtualHud().SetStatus(eDisplayElements.DELM_NTFR_FEVER,DELM_LVL_1);
■}

■override void HideBadge()
■{
■■
■■////GetVirtualHud().SetStatus(eDisplayElements.DELM_NTFR_FEVER,DELM_LVL_0);
■}

};


// ---------------------------------------------
// ------ FeverNotfr.c - END ----------------------
// ---------------------------------------------


// ---------------------------------------------
// ------ FieldShovel.c - START --------------------
// ---------------------------------------------


class FieldShovel extends ItemBase
{
■override bool CanMakeGardenplot()
■{
■■return true;
■}

■override void SetActions()
■{
■■super.SetActions();
■■
■■AddAction(ActionClapBearTrapWithThisItem);
■■AddAction(ActionTogglePlaceObject);
■■AddAction(ActionDigGardenPlot);
■■AddAction(ActionDismantleGardenPlot);
■■AddAction(ActionDismantlePart);
■■AddAction(ActionBuildPart);
■■AddAction(ActionBuryBody);
■■AddAction(ActionBuryAshes);
■■AddAction(ActionDigInStash);
■■AddAction(ActionFillObject);
```

```
// ---------------------------------------------
// ------ FillGasMask_Filter.c - START --------------------
// ---------------------------------------------


class FillGasMask_Filter extends RecipeBase■
{■
■override void Init()
■{
■■m_Name = "#STR_FillFilter";
■■m_IsInstaRecipe = false;//should this recipe be performed instantly without animation
■■m_AnimationLength = 0.5;//animation length in relative time units
■■m_Specialty = 0.02;// value > 0 for roughness, value < 0 for precision
■■
■■
■■//conditions
■■m_MinDamageIngredient[0] = -1;//-1 = disable check
■■m_MaxDamageIngredient[0] = 3;//-1 = disable check
■■
■■m_MinQuantityIngredient[0] = 1;//-1 = disable check
■■m_MaxQuantityIngredient[0] = -1;//-1 = disable check
■■
■■m_MinDamageIngredient[1] = -1;//-1 = disable check
■■m_MaxDamageIngredient[1] = 3;//-1 = disable check
■■
■■m_MinQuantityIngredient[1] = -1;//-1 = disable check
■■m_MaxQuantityIngredient[1] = -1;//-1 = disable check
■■//----------------------------------------------------------------------------------------------------------------------
■■
■■//INGREDIENTS
■■//ingredient 1
■■InsertIngredient(0,"CharcoalTablets");//you can insert multiple ingredients this way

■■m_IngredientAddHealth[0] = 0;// 0 = do nothing
■■m_IngredientSetHealth[0] = -1; // -1 = do nothing
■■m_IngredientAddQuantity[0] = 0;// 0 = do nothing
■■m_IngredientDestroy[0] = false;//true = destroy, false = do nothing
■■m_IngredientUseSoftSkills[0] = false;// set 'true' to allow modification of the values by softskills on this i
■■
■■//ingredient 2
■■InsertIngredient(1,"GasMask_Filter");//you can insert multiple ingredients this way
■■InsertIngredient(1,"GasMask_Filter_Improvised");
■■
■■m_IngredientAddHealth[1] = 0;// 0 = do nothing
■■m_IngredientSetHealth[1] = -1; // -1 = do nothing
■■m_IngredientAddQuantity[1] = 0;// 0 = do nothing
■■m_IngredientDestroy[1] = false;// false = do nothing
■■m_IngredientUseSoftSkills[1] = false;// set 'true' to allow modification of the values by softskills on this i
■■//----------------------------------------------------------------------------------------------------------------------
■■/*
■■//result1
■■//AddResult("GasMask_Filter_Improvised");//add results here

■■m_ResultSetFullQuantity[0] = false;//true = set full quantity, false = do nothing
■■m_ResultSetQuantity[0] = 0;//-1 = do nothing
■■m_ResultSetHealth[0] = -1;//-1 = do nothing
■■m_ResultInheritsHealth[0] = 1;// (value) == -1 means do nothing; a (value) >= 0 means this result will in
■■m_ResultInheritsColor[0] = -1;// (value) == -1 means do nothing; a (value) >= 0 means this result classn
■■m_ResultToInventory[0] = -2;//(value) == -2 spawn result on the ground;(value) == -1 place anywhere in
■■m_ResultUseSoftSkills[0] = false;// set 'true' to allow modification of the values by softskills on this resu
■■m_ResultReplacesIngredient[0] = -1;// value == -1 means do nothing; a value >= 0 means this result wi
■■*/
■}
```

```
// ---------------------------------------------
// ------ FillSyringe.c - START --------------------
// ---------------------------------------------


class FillSyringe extends RecipeBase
{■
■override void Init()
■{
■■m_Name = "#fill";
■■m_IsInstaRecipe = false;//should this recipe be performed instantly without animation
■■m_AnimationLength = 1;//animation length in relative time units
■■m_Specialty = -0.02;// value > 0 for roughness, value < 0 for precision
■■
■■
■■//conditions
■■m_MinDamageIngredient[0] = -1;//-1 = disable check
■■m_MaxDamageIngredient[0] = 3;//-1 = disable check
■■
■■m_MinQuantityIngredient[0] = -1;//-1 = disable check
■■m_MaxQuantityIngredient[0] = -1;//-1 = disable check
■■
■■m_MinDamageIngredient[1] = -1;//-1 = disable check
■■m_MaxDamageIngredient[1] = 3;//-1 = disable check
■■
■■m_MinQuantityIngredient[1] = -1;//-1 = disable check
■■m_MaxQuantityIngredient[1] = -1;//-1 = disable check
■■//----------------------------------------------------------------------------------------------------------------------
■■
■■//INGREDIENTS
■■//ingredient 1
■■InsertIngredient(0,"InjectionVial");//you can insert multiple ingredients this way
■■
■■m_IngredientAddHealth[0] = 0;// 0 = do nothing
■■m_IngredientSetHealth[0] = -1; // -1 = do nothing
■■m_IngredientAddQuantity[0] = 0;// 0 = do nothing
■■m_IngredientDestroy[0] = true;//true = destroy, false = do nothing
■■m_IngredientUseSoftSkills[0] = false;// set 'true' to allow modification of the values by softskills on this i
■■
■■//ingredient 2
■■InsertIngredient(1,"Syringe");//you can insert multiple ingredients this way
■■
■■m_IngredientAddHealth[1] = 0;// 0 = do nothing
■■m_IngredientSetHealth[1] = -1; // -1 = do nothing
■■m_IngredientAddQuantity[1] = 0;// 0 = do nothing
■■m_IngredientDestroy[1] = true;// false = do nothing
■■m_IngredientUseSoftSkills[1] = false;// set 'true' to allow modification of the values by softskills on this i
■■//----------------------------------------------------------------------------------------------------------------------
■■
■■//result1
■■AddResult("ClearSyringe");//add results here

■■m_ResultSetFullQuantity[0] = false;//true = set full quantity, false = do nothing
■■m_ResultSetQuantity[0] = -1;//-1 = do nothing
■■m_ResultSetHealth[0] = -1;//-1 = do nothing
■■m_ResultInheritsHealth[0] = 1;// (value) == -1 means do nothing; a (value) >= 0 means this result will in
■■m_ResultInheritsColor[0] = -1;// (value) == -1 means do nothing; a (value) >= 0 means this result class
■■m_ResultToInventory[0] = -2;//(value) == -2 spawn result on the ground;(value) == -1 place anywhere i
■■m_ResultUseSoftSkills[0] = false;// set 'true' to allow modification of the values by softskills on this resu
■■m_ResultReplacesIngredient[0] = -1;// value == -1 means do nothing; a value >= 0 means this result wi
■}

■override bool CanDo(ItemBase ingredients[], PlayerBase player)//final check for recipe's validity
```

```
// ---------------------------------------------
// ------ FirearmActionAttachMagazine.c - START --------------------
// ---------------------------------------------


class AttachMagazineActionReciveData : ActionReciveData
{
■ref InventoryLocation  m_ilOldMagazine;
}
class AttachMagazineActionData : ActionData
{
■ref InventoryLocation  m_ilOldMagazine;
■Magazine m_oldMagazine;
}

class FirearmActionAttachMagazine : FirearmActionBase
{■
■//-----------------------------------------------------
■// ■Action events and methods
■//-----------------------------------------------------
■void FirearmActionAttachMagazine()
■{
■■m_Text = "#attach";
■}■
■
■override int GetActionCategory()
■{
■■return AC_SINGLE_USE;
■}
■
■override bool ActionCondition( PlayerBase player, ActionTarget target, ItemBase item ) //condition for ac
■{
■■if (!super.ActionCondition( player, target, item ))
■■■return false;
■■
■■HumanCommandWeapons hcw = player.GetCommandModifier_Weapons();
■■Magazine mag = Magazine.Cast(target.GetObject());■
■■Weapon_Base wpn = Weapon_Base.Cast(item);■
■■return mag && (player.GetWeaponManager().CanAttachMagazine(wpn,mag) || player.GetWeaponMan
■}
■
■override bool ActionConditionContinue( ActionData action_data )
■{
■■return true;
■}
■
■override ActionData CreateActionData()
■{
■■AttachMagazineActionData action_data = new AttachMagazineActionData;
■■return action_data;
■}
■
■
■override void WriteToContext(ParamsWriteContext ctx, ActionData action_data)
■{
■■super.WriteToContext(ctx, action_data);
■■
■■AttachMagazineActionData action_data_am = AttachMagazineActionData.Cast(action_data);
■■
■■action_data_am.m_ilOldMagazine.WriteToContext(ctx);
■}
■
■override bool ReadFromContext(ParamsReadContext ctx, out ActionReciveData action_recive_data )
```

```
// ---------------------------------------------
// ------ FirearmActionBase.c - START --------------------
// ---------------------------------------------


class FirearmActionBase : ActionBase
{■
■void FirearmActionBase()
■{
■}
■
■override int GetStanceMask(PlayerBase player)
■{
■■return DayZPlayerConstants.STANCEMASK_ALL;
■}
■
■override int GetActionCategory()
■{
■■return AC_SINGLE_USE;
■}

■override typename GetInputType()
■{
■■return DefaultActionInput;
■}

■override bool ActionConditionContinue( ActionData action_data ) //condition for action
■{
■■Weapon_Base wpn = Weapon_Base.Cast(action_data.m_MainItem);
■■return !wpn.IsIdle();
■}
■
■override bool ActionCondition( PlayerBase player, ActionTarget target, ItemBase item )
■{
■■Weapon_Base wpn = Weapon_Base.Cast(item);
■■return wpn && wpn.CanProcessWeaponEvents() && !player.GetDayZPlayerInventory().IsProcessing();
■}
■
■override void Start( ActionData action_data )
■{
■■super.Start( action_data );
■■action_data.m_State = UA_PROCESSING;
■}
■
■override bool CanBePerformedFromInventory()
■{
■■return true;
■}
■
■override bool CanBeUsedOnBack()
■{
■■return true;
■}
■
■override bool CanBeUsedRaised()
■{
■■return true;
■}

■override void OnUpdate( ActionData action_data )
■{
■■super.OnUpdate( action_data );
■■Weapon_Base wpn = Weapon_Base.Cast(action_data.m_MainItem);
```

```
// ---------------------------------------------
// ------ FirearmActionDetachMagazine.c - START --------------------
// ---------------------------------------------


class DetachMagActionReciveData : ActionReciveData
{
■ref InventoryLocation  m_ilMagazine;
}
class DetachMagActionData : ActionData
{
■ref InventoryLocation  m_ilMagazine;
}

class FirearmActionDetachMagazine_Old : FirearmActionBase
{■
■void FirearmActionDetachMagazine_Old()
■{
■}■
■
■override bool HasTarget()
■{
■■return true;
■}
■
■override typename GetInputType()
■{
■■return QuickaBarActionInput;
■}
■
■override bool CanBePerformedFromQuickbar()
■{
■■return true;
■}
■
■override void CreateConditionComponents()
■{
■■m_ConditionItem = new CCINonRuined();
■■m_ConditionTarget = new CCTSelf;
■}
■
■override bool HasProgress()
■{
■■return false;
■}
■
■override bool InventoryReservation( ActionData action_data)
■{
■■Magazine mag = Magazine.Cast(action_data.m_Target.GetObject());
■■InventoryLocation il = new InventoryLocation();
■■if( !action_data.m_Player.GetInventory().FindFreeLocationFor(mag, FindInventoryLocationType.ANY_
■■■return false;
■■
■■if( !super.InventoryReservation( action_data) )
■■■return false;

■■DetachMagActionData action_data_dm = DetachMagActionData.Cast(action_data);
■■if( !action_data.m_Player.GetInventory().AddInventoryReservationEx(mag,il,10000) )
■■■return false;
■■■
■■action_data_dm.m_ReservedInventoryLocations.Insert(il);
■■action_data_dm.m_ilMagazine = il;
```

```
// ---------------------------------------------
// ------ FirearmActionLoadBullet.c - START --------------------
// ---------------------------------------------


class FirearmActionLoadBullet : FirearmActionBase
{■
■//-----------------------------------------------------
■// ■Action events and methods
■//-----------------------------------------------------
■void FirearmActionLoadBullet()
■{
■■m_Text = "#load_bullet";
■}■
■
■override int GetActionCategory()
■{
■■return AC_SINGLE_USE;
■}
■
■/*string GetTargetDescription()
■{
■■return "default target description";
■}*/
■
■/*protected bool ActionConditionContinue( ActionData action_data ) //condition for action
■{
■■return true;
■}*/
■
■override bool ActionCondition( PlayerBase player, ActionTarget target, ItemBase item ) //condition for ac
■{
■■if (!super.ActionCondition( player, target, item ))
■■■return false;
■■
■■HumanCommandWeapons hcw = player.GetCommandModifier_Weapons();■■
■■Magazine mag = Magazine.Cast(target.GetObject());■
■■Weapon_Base wpn = Weapon_Base.Cast(item);
■■return mag && player.GetWeaponManager().CanLoadBullet(wpn,mag) && hcw && hcw.GetRunningAc
■}
■
■override void Start( ActionData action_data )
■{
■■super.Start( action_data );
■■Magazine mag = Magazine.Cast(action_data.m_Target.GetObject());■

■■action_data.m_Player.GetWeaponManager().LoadBullet(mag, this);
■}
■
■override bool CanBePerformedFromInventory()
■{
■■return true;
■}
■
■override bool CanBePerformedFromQuickbar()
■{
■■return true;
■}
■
■// action need first have permission from server before can start
■/*bool UseAcknowledgment()
■{
■■return true;
```

```
// ---------------------------------------------
// ------ FirearmActionLoadMultiBullet.c - START --------------------
// ---------------------------------------------


class FirearmActionLoadMultiBullet : FirearmActionBase
{■
■//-----------------------------------------------------
■// ■Action events and methods
■//-----------------------------------------------------
■void FirearmActionLoadMultiBullet()
■{
■■m_Text = "#load_bullets";
■}■
■
■override int GetActionCategory()
■{
■■return AC_CONTINUOUS;
■}
■
■/*string GetTargetDescription()
■{
■■return "default target description";
■}*/
■
■/*protected bool ActionConditionContinue( ActionData action_data ) //condition for action
■{
■■return true;
■}*/
■
■override bool ActionCondition( PlayerBase player, ActionTarget target, ItemBase item ) //condition for ac
■{
■■if (!super.ActionCondition( player, target, item ))
■■■return false;
■■
■■Weapon_Base wpn = Weapon_Base.Cast(item);
■■Magazine mag = Magazine.Cast(target.GetObject());■■
■■return mag && player.GetWeaponManager().CanLoadBullet(wpn,mag);
■}
■
■override void Start( ActionData action_data )
■{
■■super.Start( action_data );
■■Magazine mag = Magazine.Cast(action_data.m_Target.GetObject());■

■■action_data.m_Player.GetWeaponManager().LoadMultiBullet(mag, this);

■}
■
■override bool CanBePerformedFromInventory()
■{
■■return false;
■}
■
■override bool CanBeSetFromInventory()
■{
■■return false;
■}

■override void OnEndInput( ActionData action_data )
■{
■■action_data.m_Player.GetWeaponManager().LoadMultiBulletStop();
■}
```

```
// ---------------------------------------------
// ------ FirearmActionMechanicManipulate.c - START --------------------
// ---------------------------------------------


class FirearmActionMechanicManipulate : FirearmActionBase
{■
■void FirearmActionLoadBulletQuick()
■{
■}■
■
■override bool HasTarget()
■{
■■return false;
■}
■
■override typename GetInputType()
■{
■■return WeaponManipulationActionInput;
■}
■
■override void CreateConditionComponents()
■{
■■m_ConditionItem = new CCINonRuined();
■■m_ConditionTarget = new CCTSelf;
■}
■
■override bool HasProgress()
■{
■■return false;
■}
■
■
■override bool ActionCondition( PlayerBase player, ActionTarget target, ItemBase item ) //condition for ac
■{
■■if (!super.ActionCondition( player, target, item ))
■■■return false;
■■
■■bool result = false;
■■Weapon_Base wpn = Weapon_Base.Cast(item);
■■if ( player.GetWeaponManager().CanEjectBullet(wpn))
■■{
■■■if( GetGame().IsServer() && GetGame().IsMultiplayer() )
■■■{
■■■■result = true;
■■■}
■■■else
■■■{
■■■■if( player.GetWeaponManager().CanEjectBulletVerified() )
■■■■{
■■■■■result = true;
■■■■}
■■■■player.GetWeaponManager().SetEjectBulletTryTimestamp();
■■■}
■■}
■■return result;
■}
■
■override void Start( ActionData action_data )
■{
■■super.Start( action_data );
■■
■■action_data.m_Player.GetWeaponManager().EjectBulletVerified( this );
```

```
// ---------------------------------------------
// ------ FirearmActionUnjam.c - START --------------------
// ---------------------------------------------


class FirearmActionUnjam : FirearmActionBase
{■
■void FirearmActionUnjam()
■{
■}■
■
■override bool HasTarget()
■{
■■return false;
■}
■
■override typename GetInputType()
■{
■■return ContinuousWeaponManipulationActionInput;
■}
■
■override void CreateConditionComponents()
■{
■■m_ConditionItem = new CCINonRuined();
■■m_ConditionTarget = new CCTSelf;
■}
■
■override bool HasProgress()
■{
■■return false;
■}
■
■
■override bool ActionCondition( PlayerBase player, ActionTarget target, ItemBase item ) //condition for ac
■{
■■if (!super.ActionCondition( player, target, item ))
■■■return false;
■■
■■Weapon_Base wpn = Weapon_Base.Cast(item);
■■return player.GetWeaponManager().CanUnjam(wpn);
■}
■
■override void Start( ActionData action_data )
■{
■■super.Start( action_data );
■■
■■action_data.m_Player.GetWeaponManager().Unjam( this );
■}
};


// ---------------------------------------------
// ------ FirearmActionUnjam.c - END ----------------------
// ---------------------------------------------


// ---------------------------------------------
// ------ FireConsumable.c - START --------------------
// ---------------------------------------------


class FireConsumable
{
```

```
// ----------------------------------------------
// ------ FireConsumableType.c - START --------------------
// ----------------------------------------------


class FireConsumableType
{
■typename ■m_ItemType;
■float ■■m_Energy;
■bool ■■m_IsKindling;
■string ■■m_AttSlot;
■
■void FireConsumableType ( typename item_type, float energy, bool is_kindling, string att_slot )
■{
■■m_ItemType = item_type;
■■m_Energy = energy;
■■m_IsKindling = is_kindling;
■■m_AttSlot = att_slot;
■}
■
■//Item typename
■typename GetItemType()
■{
■■return m_ItemType;
■}
■
■//Energy
■float GetEnergy()
■{
■■return m_Energy;
■}
■
■//Is Kindling
■bool IsKindling()
■{
■■return m_IsKindling;
■}
■
■//Attachment slot
■string GetAttSlot()
■{
■■return m_AttSlot;
■}
}



// ----------------------------------------------
// ------ FireConsumableType.c - END ----------------------
// ----------------------------------------------


// ----------------------------------------------
// ------ FireExtinguisher.c - START --------------------
// ----------------------------------------------


class FireExtinguisher: Inventory_Base
{
■override void SetActions()
■{
■■super.SetActions();
■■
```

```
// ---------------------------------------------
// ------ FirefighterAxe.c - START --------------------
// ---------------------------------------------


class FirefighterAxe extends ToolBase
{
■void FirefighterAxe()
■{
■}

■override void SetActions()
■{
■■super.SetActions();
■■AddAction(ActionMineTree);
■■AddAction(ActionMineTreeBark);
■■AddAction(ActionMineBush);
■■//AddAction(ActionRepairPart);
■■AddAction(ActionDismantlePart);
■■//AddAction(ActionBuildPart);
■■//AddAction(ActionDestroyPart);
■■//AddAction(ActionSawPlanks);
■■AddAction(ActionUnrestrainTarget);
■■AddAction(ActionSkinning);
■}
}


// ---------------------------------------------
// ------ FirefighterAxe.c - END ----------------------
// ---------------------------------------------


// ---------------------------------------------
// ------ FirefighterJacket_ColorBase.c - START --------------------
// ---------------------------------------------


class FirefighterJacket_ColorBase extends Clothing
{
■override void SetActions()
■{
■■super.SetActions();
■■AddAction(ActionWringClothes);
■}
};
class FirefighterJacket_Beige extends FirefighterJacket_ColorBase {};
class FirefighterJacket_Black extends FirefighterJacket_ColorBase {};


// ---------------------------------------------
// ------ FirefighterJacket_ColorBase.c - END ----------------------
// ---------------------------------------------


// ---------------------------------------------
// ------ FirefightersHelmet_ColorBase.c - START --------------------
// ---------------------------------------------


class FirefightersHelmet_ColorBase extends HelmetBase {};
class FirefightersHelmet_Red extends FirefightersHelmet_ColorBase {};
class FirefightersHelmet_White extends FirefightersHelmet_ColorBase {};
```

```
// ---------------------------------------------
// ------ FirefightersPants_ColorBase.c - START --------------------
// ---------------------------------------------


class FirefightersPants_ColorBase extends Clothing
{
■override void SetActions()
■{
■■super.SetActions();
■■AddAction(ActionWringClothes);
■}
};
class FirefightersPants_Beige extends FirefightersPants_ColorBase {};
class FirefightersPants_Black extends FirefightersPants_ColorBase {};


// ---------------------------------------------
// ------ FirefightersPants_ColorBase.c - END ----------------------
// ---------------------------------------------



// ---------------------------------------------
// ------ Fireplace.c - START --------------------
// ---------------------------------------------


class Fireplace extends FireplaceBase
{
■bool m_ContactEventProcessing;
■
■void Fireplace()
■{
■■//Particles - default for FireplaceBase
■■PARTICLE_FIRE_START ■= ParticleList.CAMP_FIRE_START;
■■PARTICLE_SMALL_FIRE ■= ParticleList.CAMP_SMALL_FIRE;
■■PARTICLE_NORMAL_FIRE■= ParticleList.CAMP_NORMAL_FIRE;
■■PARTICLE_SMALL_SMOKE ■= ParticleList.CAMP_SMALL_SMOKE;
■■PARTICLE_NORMAL_SMOKE■= ParticleList.CAMP_NORMAL_SMOKE;
■■PARTICLE_FIRE_END ■■= ParticleList.CAMP_FIRE_END;
■■PARTICLE_STEAM_END■■= ParticleList.CAMP_STEAM_2END;
■■
■■SetEventMask( EntityEvent.CONTACT | EntityEvent.TOUCH );
■}
■
■override bool IsBaseFireplace()
■{
■■return true;
■}
■
■override void EOnTouch( IEntity other, int extra )
■{
■■ContactEvent( other, GetPosition() );
■}
■
■override void EOnContact( IEntity other, Contact extra )
■{
■■ContactEvent( other, extra.Position );
■}
■
■void ContactEvent( IEntity other, vector position )
■{■■
■■if ( GetGame().IsServer() && !m_ContactEventProcessing && dBodyIsActive(this) && !IsSetForDeletion
// ---------------------------------------------
// ------ FireplaceBase.c - START --------------------
// ---------------------------------------------


enum FireplaceFireState
{
■NO_FIRE■■■■= 1,
■START_FIRE■■■= 2,
■SMALL_FIRE■■■= 3,■
■NORMAL_FIRE■■■= 4,
■END_FIRE■■■= 5,
■EXTINGUISHING_FIRE■= 6,
■EXTINGUISHED_FIRE■= 7,
■REIGNITED_FIRE■■= 8,
■
■COUNT■■■■= 9■■■//for net sync purposes
}

enum FirePlaceFailure
{
■WIND■■■= 0,
■WET■■■■= 1,■
}

class FireplaceBase extends ItemBase
{
■//State params
■protected bool m_IsBurning ■■■■■■= false;
■protected bool m_HasAshes■■■■■■= false;
■protected bool m_IsOven■■■■■■■= false;
■protected bool m_HasStoneCircle ■■■■= false;
■protected bool m_RoofAbove ■■■■■■= false;
■protected bool m_NoIgnite ■■■■■■= false;
■protected int m_OvenAttachmentsLockState ■■= -1;
■protected FireplaceFireState m_FireState■ ■= FireplaceFireState.NO_FIRE;
■protected FireplaceFireState m_LastFireState ■= FireplaceFireState.NO_FIRE;■■//for synchronization p
■protected vector m_HalfExtents;
■
■//Fireplace params
■protected float m_TemperatureLossMP■■■■= 1.0;■■//! determines how fast will the fireplace loose its
■protected float m_FuelBurnRateMP■■■■= 1.0;■■//! determines how fast will the fuel item burn before s
■
■//cooking
■protected ref Cooking m_CookingProcess;
■
■//
■const float PARAM_SMALL_FIRE_TEMPERATURE ■■= 150;■■//! maximum fireplace temperature of a
■const float PARAM_NORMAL_FIRE_TEMPERATURE ■■= 1000;■■//! maximum fireplace temperature
■const float PARAM_MIN_FIRE_TEMPERATURE ■■■= 30;■■//! minimum fireplace temperature under
■const float■PARAM_TEMPERATURE_INCREASE ■■■= 9;■■//! how much will temperature increase w
■const float■PARAM_TEMPERATURE_DECREASE ■■■= 20;■■//! how much will temperature decreas
■const float■PARAM_MAX_WET_TO_IGNITE ■■■= 0.2;■■//! maximum wetness value when the firepla
■const float PARAM_MIN_TEMP_TO_REIGNITE ■■■= 30;■■//! minimum fireplace temperature under w
■const float■PARAM_IGNITE_RAIN_THRESHOLD ■■= 0.1;■■//! maximum rain value when the fireplac
■const float■PARAM_BURN_WET_THRESHOLD ■■■= 0.40;■■//! maximum wetness value when the fi
■const float■PARAM_WET_INCREASE_COEF ■■■= 0.02;■■//! value for calculating of  wetness that fir
■const float■PARAM_WET_HEATING_DECREASE_COEF ■= 0.01;■■//! value for calculating wetness l
■const float■PARAM_WET_COOLING_DECREASE_COEF ■= 0.002;■//! value for calculating wetness l
■const float■PARAM_FIRE_CONSUM_RATE_AMOUNT■■= 0.5;■■//! base value of fire consumption ra
■const float■PARAM_BURN_DAMAGE_COEF■■■■= 5.0;■■//! value for calculating damage on items lo
■const float■PARAM_ITEM_HEAT_TEMP_INCREASE_COEF■= 10;■■//! value for calculating temperat
■const float■PARAM_ITEM_HEAT_MIN_TEMP■■■= 40;■■//! minimum temperature for items that can b
```

```
// ---------------------------------------------
// ------ FireplaceIndoor.c - START --------------------
// ---------------------------------------------


class FireplaceIndoor extends FireplaceBase
{
■protected float ■■■■m_SmokePosX;
■protected float ■■■■m_SmokePosY;
■protected float ■■■■m_SmokePosZ;
■protected int■■■■■m_FirePointIndex = 1;■//limited to 1 decimal place (1-9)
■
■static const string FIREPOINT_ACTION_SELECTION■= "fireplace_action";
■static const string FIREPOINT_FIRE_POSITION ■= "fireplace_point";
■static const string FIREPOINT_PLACE_ROT ■■= "fireplace_rot";
■static const string FIREPOINT_SMOKE_POSITION ■= "fireplace_smoke";
■
■void FireplaceIndoor()
■{
■■//Particles - default for FireplaceBase
■■PARTICLE_FIRE_START ■= ParticleList.HOUSE_FIRE_START;
■■PARTICLE_SMALL_FIRE ■= ParticleList.HOUSE_SMALL_FIRE;
■■PARTICLE_NORMAL_FIRE■= ParticleList.HOUSE_NORMAL_FIRE;
■■PARTICLE_SMALL_SMOKE ■= ParticleList.HOUSE_SMALL_SMOKE;
■■PARTICLE_NORMAL_SMOKE■= ParticleList.HOUSE_NORMAL_SMOKE;
■■PARTICLE_FIRE_END ■■= ParticleList.HOUSE_FIRE_END;
■■PARTICLE_STEAM_END■■= ParticleList.HOUSE_FIRE_STEAM_2END;■■
■■
■■//register sync variables
■■RegisterNetSyncVariableFloat( "m_SmokePosX", 0, 0, 2 );
■■RegisterNetSyncVariableFloat( "m_SmokePosY", 0, 0, 2 );
■■RegisterNetSyncVariableFloat( "m_SmokePosZ", 0, 0, 2 );
■■RegisterNetSyncVariableInt( "m_FirePointIndex", 0, 9 );
■■
■■m_LightDistance = 50;
■■m_RoofAbove = true;
■}
■
■//================================================================
■// ONSTORESAVE/LOAD/AFTERLOAD
■//================================================================
■override void OnStoreSave( ParamsWriteContext ctx )
■{
■■super.OnStoreSave( ctx );

■■//fire point name
■■ctx.Write( m_FirePointIndex );
■■
■■//smoke position
■■ctx.Write( m_SmokePosX );
■■ctx.Write( m_SmokePosY );
■■ctx.Write( m_SmokePosZ );
■}

■override bool OnStoreLoad( ParamsReadContext ctx, int version )
■{
■■if ( !super.OnStoreLoad( ctx, version ) )
■■■return false;

■■//--- Fireplace Indoor data ---
■■//fire point name
■■if ( !ctx.Read( m_FirePointIndex ) )
■■{
```

```
// ---------------------------------------------
// ------ FireplaceLight.c - START --------------------
// ---------------------------------------------


class FireplaceLight extends PointLightBase
{
■static float m_FireplaceRadius = 25;
■static float m_FireplaceBrightness = 4.75;
■
■void FireplaceLight()
■{
■■SetVisibleDuringDaylight(false);
■■SetRadiusTo( m_FireplaceRadius );
■■SetBrightnessTo(m_FireplaceBrightness);
■■SetCastShadow(true);
■■SetFadeOutTime(5);
■■SetDiffuseColor(1.0, 0.5, 0.3);
■■SetAmbientColor(1.0, 0.5, 0.3);
■■SetFlareVisible(false);
■■SetFlickerAmplitude(0.3);
■■SetFlickerSpeed(0.9);
■■SetExteriorMode();
■■EnableHeatHaze(true);
■■SetHeatHazeRadius(0.23);
■■SetHeatHazePower(0.010);
■}
■
■// Use this mode when the light is in tight space like barrel, chimney or improvsed oven
■void SetInteriorMode()
■{
■■SetDancingShadowsMovementSpeed(0.05);
■■SetDancingShadowsAmplitude(0.05);
■}
■
■// Use this mode when fireplace is on the ground with plenty of space for the light to wiggle around
■void SetExteriorMode()
■{
■■SetDancingShadowsMovementSpeed(0.25);
■■SetDancingShadowsAmplitude(0.10);
■}
■
■/*override void OnFrameLightSource(IEntity other, float timeSlice)
■{
■■
■}*/
}


// ---------------------------------------------
// ------ FireplaceLight.c - END ----------------------
// ---------------------------------------------


// ---------------------------------------------
// ------ Firewood.c - START --------------------
// ---------------------------------------------


class Firewood extends ItemBase
{
■override void SetActions()
■{
```

```
// ---------------------------------------------
// ------ FireworksBase.c - START --------------------
// ---------------------------------------------



enum EFireworksState
{
■DEFAULT,
■PLACED,
■IGNITED,
■FIRING,
■FINISHED
}

class FireworksBase: Inventory_Base
{
■protected  EFireworksState m_State;
■protected  EFireworksState m_StatePrev;
■protected ref Timer m_TimerEvent;
■protected int m_RandomSeed;

■void FireworksBase()
■{
■■Init();
■}
■
■override void EEOnCECreate()
■{
■■StandUp();
■}
■
■protected void Init();
■

■
■override protected void SetActions()
■{
■■super.SetActions();
■■
■■AddAction(ActionTogglePlaceObject);
■■AddAction(ActionDeployObject);
■■AddAction(ActionLightItemOnFire);
■}
■
■override bool HasFlammableMaterial()
■{
■■return true;
■}
■
■protected float GetMaxAllowedWetness()
■{
■■return 0.1;
■}
■
■protected EFireworksState GetState()
■{
■■return m_State;
■}
■
■protected void SetState(EFireworksState state)
■{
■■if (state != m_StatePrev && GetGame().IsServer())
```

```
// ---------------------------------------------
// ------ FireworksLauncher.c - START --------------------
// ---------------------------------------------


class ExplosionLight : PointLightBase
{
■void ExplosionLight()
■{
■■SetVisibleDuringDaylight(true);
■■SetRadiusTo(80);
■■SetBrightnessTo(0.05);
■■SetFlareVisible(false);
■■SetAmbientColor(1.0, 1.0, 1.0);
■■SetDiffuseColor(1.0, 1.0, 1.0);
■■SetLifetime(2.1);
■■SetDisableShadowsWithinRadius(-1);
■■SetFadeOutTime(1);
■■m_FadeInTime = 0.25;
■■SetFlickerSpeed(7);
■■//SetFlickerAmplitude(0.5);
■■SetFlickerAmplitudeMax(3);
■■SetFlickerAmplitudeMin(0);
■}
}

class FireworksLauncherClientEventBase
{
■void OnFired();
}

class FireworksLauncherClientEvent : FireworksLauncherClientEventBase
{
■protected ref Timer ■■■m_Timer = new Timer();
■protected FireworksLauncher ■m_Item;
■protected int ■■■■■m_Index;
■protected vector■■■■m_ExplosionPos;
■protected EffectSound ■■■m_FireSound;
■protected EffectSound ■■■m_ExplosionSound;
■protected ParticleSource ■■m_ParticleShot;
■protected ParticleSource ■■m_ParticleExplosion;
■protected vector■■■■m_ShotDir;
■protected ExplosionLight■■m_ExplosionLight;
■protected ParticleSource ■■m_ParticleAfterBurnEnd;
■protected string■■■■m_Color;
■protected int ■■■■■m_RemainingExplosions = GetSecondaryExplosionCount();
■protected ref array<ref FireworksLauncherClientEventBase> m_Events = new ref array<ref FireworksLau
■#ifdef DEVELOPER
■Shape ■■■■■■■m_ShotTrajectory;
■#endif
■
■
■// -----------------------------
■// ------- Tweaking Start-------
■// -----------------------------
■
■protected int GetSoundDelay()
■{
■■return 0;
■}
■
■protected int GetLightDelay()
■{
```

```
// ---------------------------------------------
// ------ FishingRod.c - START --------------------
// ---------------------------------------------


class FishingRod : FishingRod_Base_New
{
■void FishingRod()
■{
■}
■
■override float GetFishingEffectivityBonus()
■{
■■return 0.1;
■}
■
■override void AnimateFishingRod(bool state)
■{
■■SetAnimationPhase("AnimateRod",state);
■}
■
■override void SetActions()
■{
■■super.SetActions();
■}■
};



// ---------------------------------------------
// ------ FishingRod.c - END ----------------------
// ---------------------------------------------


// ---------------------------------------------
// ------ FishingRod_Base.c - START --------------------
// ---------------------------------------------


class FishingRod_Base_New : ItemBase
{
■void FishingRod_Base_New()
■{
■}
■
■override void SetActions()
■{
■■super.SetActions();
■■
■■AddAction(ActionFishingNew);
■}
■
■float GetFishingEffectivityBonus()
■{
■■return 0.0;
■}
■
■override bool IsOpen()
■{
■■return false;
■}
■
■void AnimateFishingRod(bool state) {}■
```

```
// ---------------------------------------------
// ------ Flag_Base.c - START --------------------
// ---------------------------------------------


class Flag_Base extends ItemBase
{
■void Flag_Base()
■{
■■//synchronized variables
■■//RegisterNetSyncVariableBool( "m_IsMounted" );
■■ShowSelection("folded");
■■HideSelection("unfolded");
■}
■
■void ~Flag_Base()
■{
■}
■
■// --- SYNCHRONIZATION
■void Synchronize()
■{
■■if ( GetGame().IsServer() )
■■{
■■■SetSynchDirty();
■■}
■}
■
■override void OnVariablesSynchronized()
■{
■■super.OnVariablesSynchronized();
■■
■■
■}
■
■// --- EVENTS
■override void OnStoreSave( ParamsWriteContext ctx )
■{
■■super.OnStoreSave( ctx );
■}
■
■override bool OnStoreLoad( ParamsReadContext ctx, int version )
■{
■■if ( !super.OnStoreLoad( ctx, version ) )
■■■return false;
■■
■■return true;
■}
■
■override void AfterStoreLoad()
■{■
■■super.AfterStoreLoad();
■■■■
■■//set mounted state based on locked slot after everything is loaded
■■//SetMountedState( GetSlotLockedState() );
■}
■
■/*override void OnInventoryEnter(Man player)
■{
■■super.OnInventoryEnter(player);
■■
■■ShowSelection("folded");
■■HideSelection("unfolded");
```

```
// ---------------------------------------------
// ------ Flaregun.c - START --------------------
// ---------------------------------------------


enum FLAREAnimState
{
■COCKED ■■= 0, ■///< default weapon state, closed and discharged
■UNCOCKED ■= 1, ■///< default weapon state, closed and discharged
};

enum FLAREStableStateID
{
■UNKNOWN■■■■=  0,
■Empty■■■■=  1,
■Fireout■■■■=  2,
■Loaded■■■■=  3,
}

class FLAREEmpty extends WeaponStableState
{
■override void OnEntry (WeaponEventBase e) { if (LogManager.IsWeaponLogEnable()) { wpnPrint("[wpnf
■override void OnExit (WeaponEventBase e) { super.OnExit(e); if (LogManager.IsWeaponLogEnable()) { w
■override int GetCurrentStateID () { return FLAREStableStateID.Empty; }
■override bool HasBullet () { return false; }
■override bool HasMagazine () { return false; }
■override bool IsJammed () { return false; }
■override bool IsBoltOpen () { return false; }
■override bool IsRepairEnabled () { return true; }
■override void InitMuzzleArray () { m_muzzleHasBullet = {MuzzleState.E}; }
};
class FLAREFireout extends WeaponStableState
{
■override void OnEntry (WeaponEventBase e) { if (LogManager.IsWeaponLogEnable()) { wpnPrint("[wpnf
■override void OnExit (WeaponEventBase e) { super.OnExit(e); if (LogManager.IsWeaponLogEnable()) { w
■override int GetCurrentStateID () { return FLAREStableStateID.Fireout; }
■override bool HasBullet () { return true; }
■override bool HasMagazine () { return false; }
■override bool IsJammed () { return false; }
■override bool IsBoltOpen () { return false; }
■override bool IsRepairEnabled () { return true; }
■override void InitMuzzleArray () { m_muzzleHasBullet = {MuzzleState.F}; }
};
class FLARELoaded extends WeaponStableState
{
■override void OnEntry (WeaponEventBase e) { if (LogManager.IsWeaponLogEnable()) { wpnPrint("[wpnf
■override void OnExit (WeaponEventBase e) { super.OnExit(e); if (LogManager.IsWeaponLogEnable()) { w
■override int GetCurrentStateID () { return FLAREStableStateID.Loaded; }
■override bool HasBullet () { return true; }
■override bool HasMagazine () { return false; }
■override bool IsJammed () { return false; }
■override bool IsBoltOpen () { return false; }
■override bool IsRepairEnabled () { return true; }
■override void InitMuzzleArray () { m_muzzleHasBullet = {MuzzleState.L}; }
};

class Flaregun: Weapon_Base
{
■override void InitStateMachine()
■{
■■// setup abilities
■■m_abilities.Insert(new AbilityRecord(WeaponActions.MECHANISM, WeaponActionMechanismTypes.M
■■m_abilities.Insert(new AbilityRecord(WeaponActions.MECHANISM, WeaponActionMechanismTypes.M
// ---------------------------------------------
// ------ FlareLight.c - START --------------------
// ---------------------------------------------


class FlareLight extends PointLightBase
{
■static float ■m_FlareRadius = 100;
■static float ■m_FlareBrightness = 10.0;
■static string ■m_MemoryPoint = "light";
■
■void FlareLight()
■{
■■SetVisibleDuringDaylight( true );
■■SetRadiusTo( m_FlareRadius );
■■SetBrightnessTo( m_FlareBrightness );
■■FadeIn( 1 );
■■SetFadeOutTime( 0.2 );
■■SetDiffuseColor( 0.7, 0.7, 0.3 );
■■SetAmbientColor( 0.7, 0.7, 0.3 );
■■SetFlareVisible( false );
■■SetFlickerAmplitude( 0.9 );
■■SetFlickerSpeed( 0.3 );
■■SetDancingShadowsMovementSpeed( 0.5 );
■■SetDancingShadowsAmplitude( 0.15 );
■■//SetLifetime( 50 );
■■EnableLinear( true );
■■EnableHeatHaze( true );
■■SetHeatHazeRadius( 0.1 );
■■SetHeatHazePower( 0.02 );
■■#ifdef PLATFORM_WINDOWS
■■■SetCastShadow( true );
■■#else
■■■SetCastShadow( false );
■■#endif
■■
■}
}

class FlareLightRed extends FlareLight
{
■
■void FlareLightRed()
■{
■■SetDiffuseColor( 1.0, 0.3, 0.3 );
■■SetAmbientColor( 1.0, 0.3, 0.3 );
■}
}

class FlareLightGreen extends FlareLight
{■
■void FlareLightGreen()
■{
■■SetDiffuseColor( 0.3, 1.0, 0.3 );
■■SetAmbientColor( 0.3, 1.0, 0.3 );
■}
}

class FlareLightBlue extends FlareLight
{
■void FlareLightBlue()
■{
■■SetDiffuseColor( 0.3, 0.3, 1.0 );
```

```
// ---------------------------------------------
// ------ FlareSimulation.c - START --------------------
// ---------------------------------------------


class FlareSimulation : Entity
{
■protected Particle ■■■m_ParMainFire;
■protected EffectSound ■■m_BurningSound;
■protected FlareLight■■m_FlareLight;
■const static float■■■MAX_FARLIGHT_DIST = 40;
■const static float■■■MIN_FARLIGHT_DIST = 5;
■
■static ref NoiseParams ■■m_NoisePar; // Contains the noise data ( template and strength )
■float ■■■■■■m_LastNoiseTime = -1;
■float ■■■■■■m_NoiseTimer = 0;
■const float ■■■■NOISE_DELAY = 5; // How much time between two consecutive noise pings
■
■static protected typename ■m_ScriptedLight;
■static protected int ■■m_ParticleId;
■
■void FlareSimulation()
■{
■■m_ScriptedLight■ = FlareLight;
■■m_ParticleId■ = ParticleList.FLAREPROJ_ACTIVATE;
■}
■
■void OnActivation(Entity flare)
■{
■■if ( !GetGame().IsServer() || !GetGame().IsMultiplayer() )
■■{
■■■m_FlareLight = FlareLight.Cast( ScriptedLightBase.CreateLight( m_ScriptedLight, Vector(0,0,0) ) );
■■■if ( m_FlareLight )
■■■■m_FlareLight.AttachOnObject( flare );
■■■
■■■if (m_ParMainFire)
■■■■m_ParMainFire.Stop();
■
■■■m_ParMainFire = ParticleManager.GetInstance().PlayOnObject( m_ParticleId, flare );
■■■m_ParMainFire.SetWiggle( 7, 0.3 );
■■■
■■■flare.PlaySoundSetLoop( m_BurningSound, "roadflareLoop_SoundSet", 0, 0 );
■■}
■■
■■if ( GetGame().IsServer() )
■■{
■■■// Create and load noise parameters
■■■m_NoisePar = new NoiseParams();
■■■m_NoisePar.LoadFromPath("cfgWeapons Flaregun_Base NoiseFlare");
■■}
■}
■
■void OnTermination(Entity flare)
■{
■■//MiscGameplayFunctions.GenerateAINoiseAtPosition(flare.GetPosition(), NOISE_DELAY, m_NoisePa
■}
■
■void OnFire( Entity flare)
■{
■■//m_ParMainFire = ParticleManager.GetInstance().PlayOnObject( ParticleList.FLAREPROJ_FIRE, flare
■■//m_ParMainFire.SetWiggle( 7, 0.3);
■}
■
```

```
// ---------------------------------------------
// ------ FlashbangEffect.c - START --------------------
// ---------------------------------------------


class FlashbangEffect
{
■protected const float ■■■■■■ALPHA_MIN = 0.0;
■protected const float ■■■■■■ALPHA_MAX = 1.0;
■
■protected const float ■■■■■■SOUND_DEFER_TIME = 0.4;■■■//! SFX will be played ~0.5s AFTER V

■protected float ■■■■■■■m_HitDuration;
■protected float ■■■■■■■m_BreakPoint;
■protected float ■■■■■■■m_TimeActive;
■protected float■■■■■■■■m_DayTimeToggle;
■
■protected float■■■■■■■■m_AlphaMaxActual; //actual max alpha of the effect
■protected float■■■■■■■■m_SoundMaxActual; //actual max volume of the sound
■protected float■■■■■■■■m_ProgressMultiplier;
■
■protected bool■■■■■■■■m_Visual;
■protected bool ■■■■■■■■m_Initialized;
■
■protected PlayerBase ■■■■■■m_Player;
■protected EffectSound ■■■■■■m_FlashbangEffectSound;
■protected float ■■■■■■■m_SoundStopTime;
■
■protected ref Timer■■■■■■■m_DeferAttenuation;
■
■protected PPERequester_FlashbangEffects ■m_Requester;
■
■void FlashbangEffect(PlayerBase player, bool visual = true)
■{
■■m_Player = player;
■■m_Visual = visual;
■■m_Initialized = false;

■■m_HitDuration = 8.0;
■■m_BreakPoint = 2.5;
■■m_AlphaMaxActual = ALPHA_MAX;
■■m_SoundMaxActual = 1.0;
■■m_ProgressMultiplier = 1.0;
■■
■■m_FlashbangEffectSound = null;
■■
■■if (m_Visual)
■■{
■■■Class.CastTo(m_Requester,PPERequesterBank.GetRequester(PPERequester_FlashbangEffects));
■■■m_Requester.Start();
■■}
■■
■■m_DeferAttenuation = new ref Timer();
■■m_DeferAttenuation.Run(SOUND_DEFER_TIME, this, "PlaySound", null, false);
■■
■■//! naive time of the day selector
■■m_DayTimeToggle = 5; //! -1: night; 1: day
■■if ( g_Game.GetDayTime() >= 22.0 || g_Game.GetDayTime() < 7.0 )
■■{
■■■m_DayTimeToggle = 10;
■■}
■}
■
```

```
// ---------------------------------------------
// ------ FlashGrenade.c - START --------------------
// ---------------------------------------------


class FlashGrenade extends Grenade_Base
{
■const float FX_RANGE_MAX_MULT = 1.0;
■
■override void OnExplosionEffects(Object source, Object directHit, int componentIndex, string surface, ve
■{
■■super.OnExplosionEffects(source, directHit, componentIndex, surface, pos, surfNormal, energyFactor,

■■PlayerBase player = PlayerBase.Cast(GetGame().GetPlayer());

■■if (player)
■■{
■■■vector headPos ■■■= player.GetDamageZonePos("Head"); // animated position in the middle of the
■■■float ammoRangeKill■ ■= GetGame().ConfigGetFloat(string.Format("CfgAmmo %1 indirectHitRange
■■■float ammoRangeMaxMult ■= 4.0;
■■■
■■■string indirectHitRangeMultiplier = string.Format("CfgAmmo %1 indirectHitRangeMultiplier", ammoTy
■■■if (GetGame().ConfigIsExisting(indirectHitRangeMultiplier))
■■■{
■■■■//! values less than 1.0 make no sense
■■■■ammoRangeMaxMult = Math.Clamp(GetGame().ConfigGetFloat(indirectHitRangeMultiplier), 1.0, flo
■■■}

■■■float ammoRangeMax = ammoRangeKill * ammoRangeMaxMult;
■■■ammoRangeMax = Math.Clamp(ammoRangeMax * FX_RANGE_MAX_MULT, ammoRangeKill, amm
■■■
■■■float dist■■■= vector.Distance(headPos, pos);
■■■float distSq ■■= vector.DistanceSq(headPos, pos);
■■■float radiusMaxSq = Math.SqrFloat(ammoRangeMax);
■■■
■■■if (distSq <= radiusMaxSq)
■■■{
■■■■// ignore collisions with parent if fireplace
■■■■InventoryItem invItem = InventoryItem.Cast(source);
■■■■EntityAI parent = invItem.GetHierarchyParent();
■■■■array<Object> excluded = new array<Object>;
■■■■
■■■■if (!parent || !parent.IsFireplace())
■■■■■parent = null;
■■■■else if (parent)
■■■■■excluded.Insert(parent);
■■■■
■■■■array<ref RaycastRVResult> results = new array<ref RaycastRVResult>;
■■■■excluded.Insert(this); //Ignore self for visibility check
■■■■
■■■■//There shouldn't be cases justifying we go further than first entry (if in fireplace, self does not impac
■■■■RaycastRVParams rayParams■= new RaycastRVParams(pos, headPos, excluded[0]);
■■■■rayParams.flags■■■■= CollisionFlags.ALLOBJECTS;
■■■■DayZPhysics.RaycastRVProxy(rayParams, results, excluded);

■■■■//! removes possible obstacles made by items around the grenade(or on the same position)
■■■■array<Object> hitObjects = new array<Object>;
■■■■for (int i = 0; i < results.Count(); i++)
■■■■{
■■■■■if (results[i].obj && !results[i].obj.IsInherited(ItemBase))
■■■■■{
■■■■■■hitObjects.Insert(results[i].obj);
■■■■■}
```

```
// ---------------------------------------------
// ------ Flashlight.c - START --------------------
// ---------------------------------------------


class Flashlight extends ItemBase
{
■FlashlightLight m_Light;
■
■static int■■REFLECTOR_ID = 1;
■static int■■GLASS_ID = 2;
■
■static string ■LIGHT_OFF_GLASS = "dz\\gear\\tools\\data\\flashlight_glass.rvmat";
■static string ■LIGHT_OFF_REFLECTOR = "dz\\gear\\tools\\data\\flashlight.rvmat";
■static string ■LIGHT_ON_GLASS = "dz\\gear\\tools\\data\\flashlight_glass_on.rvmat";
■static string ■LIGHT_ON_REFLECTOR = "dz\\gear\\tools\\data\\flashlight_glass_on.rvmat";
■
■override void OnWorkStart()
■{
■■if ( !GetGame().IsServer()  ||  !GetGame().IsMultiplayer() ) // Client side
■■{
■■■m_Light = FlashlightLight.Cast(  ScriptedLightBase.CreateLight(FlashlightLight, "0 0 0", 0.08)  ); // Po
■■■m_Light.AttachOnMemoryPoint(this, "beamStart", "beamEnd");
■■}
■■
■■SetObjectMaterial(GLASS_ID, LIGHT_ON_GLASS);
■■SetObjectMaterial(REFLECTOR_ID, LIGHT_ON_REFLECTOR);
■■}
■
■override void OnWork( float consumed_energy )
■{
■■if ( !GetGame().IsServer()  ||  !GetGame().IsMultiplayer() ) // Client side
■■{
■■■Battery9V battery = Battery9V.Cast( GetCompEM().GetEnergySource() );
■■■
■■■if (battery  &&  m_Light)
■■■{
■■■■float efficiency = battery.GetEfficiency0To1();
■■■■
■■■■if ( efficiency < 1 )
■■■■{
■■■■■m_Light.SetIntensity( efficiency, GetCompEM().GetUpdateInterval() );
■■■■}
■■■■else
■■■■{
■■■■■m_Light.SetIntensity( 1, 0 );
■■■■}
■■■}
■■}
■}
■
■override void OnWorkStop()
■{
■■if ( !GetGame().IsServer()  ||  !GetGame().IsMultiplayer() ) // Client side
■■{
■■■if (m_Light)
■■■■m_Light.FadeOut();
■■■
■■■m_Light = NULL;
■■}
■■
■■SetObjectMaterial(GLASS_ID, LIGHT_OFF_GLASS);
■■SetObjectMaterial(REFLECTOR_ID, LIGHT_OFF_REFLECTOR);
```

```
// ---------------------------------------------
// ------ FlashlightLight.c - START --------------------
// ---------------------------------------------


class FlashlightLight extends SpotLightBase
{
■private static float m_DefaultBrightness = 4;
■private static float m_DefaultRadius = 25;
■
■void FlashlightLight()
■{
■■SetVisibleDuringDaylight( true );
■■SetRadiusTo( m_DefaultRadius );
■■SetSpotLightAngle( 110 );
■■SetCastShadow( true );
■■EnableSpecular( true );
■■SetBrightnessTo( m_DefaultBrightness );
■■SetFadeOutTime( 0.15 );
■■SetAmbientColor( 0.9, 0.85, 0.75 );
■■SetDiffuseColor( 0.9, 0.85, 0.75 );
■}
■
■void SetIntensity( float coef, float time )
■{
■■FadeBrightnessTo(m_DefaultBrightness * coef, time);
■}
}


// ---------------------------------------------
// ------ FlashlightLight.c - END ----------------------
// ---------------------------------------------


// ---------------------------------------------
// ------ FlatCap_ColorBase.c - START --------------------
// ---------------------------------------------


class FlatCap_ColorBase extends Clothing
{
■override void SetActions()
■{
■■super.SetActions();
■■AddAction(ActionWringClothes);
■}
};
class FlatCap_Black extends FlatCap_ColorBase {};
class FlatCap_Blue extends FlatCap_ColorBase {};
class FlatCap_Red extends FlatCap_ColorBase {};
class FlatCap_Brown extends FlatCap_ColorBase {};
class FlatCap_Grey extends FlatCap_ColorBase {};
class FlatCap_BrownCheck extends FlatCap_ColorBase {};
class FlatCap_GreyCheck extends FlatCap_ColorBase {};
class FlatCap_BlackCheck extends FlatCap_ColorBase {};


// ---------------------------------------------
// ------ FlatCap_ColorBase.c - END ----------------------
// ---------------------------------------------
// ---------------------------------------------
// ------ Flies.c - START --------------------
// ---------------------------------------------


class FliesMdfr: ModifierBase
{
■const float DISTANCE_SENSITIVITY_SQR = Math.SqrFloat(0.05/*actual distance in meters*/);
■const int TICK_FREQUENCY = 15;
■const int IDLE_COUNT_THRESHOLD = 60;
■vector m_PrevPosition;
■int m_IdleCount;
■
■
■override void Init()
■{
■■m_TrackActivatedTime = false;
■■m_ID ■■■■■= eModifiers.MDF_FLIES;
■■m_TickIntervalInactive ■= DEFAULT_TICK_TIME_INACTIVE;
■■m_TickIntervalActive ■= TICK_FREQUENCY;
■■DisableActivateCheck();
■■DisableDeactivateCheck();
■}■

■override bool ActivateCondition(PlayerBase player)
■{
■■return false;
■}

■override bool DeactivateCondition(PlayerBase player)
■{
■■return false;
■}

■override void OnTick(PlayerBase player, float deltaT)
■{■
■■float dist_sqr = vector.DistanceSq(player.GetPosition(), m_PrevPosition);
■■if( dist_sqr < DISTANCE_SENSITIVITY_SQR)//has the player stayed still since last check
■■{
■■■m_IdleCount++;
■■}
■■else
■■{
■■■if(m_IdleCount >= IDLE_COUNT_THRESHOLD)//disable the effect
■■■{
■■■■player.m_CorpseState = -PlayerConstants.CORPSE_STATE_DECAYED;
■■■■player.SetSynchDirty();
■■■}
■■■
■■■m_IdleCount = 0;//player moved, reset the count
■■}
■■m_PrevPosition = player.GetPosition();
■■
■■if( m_IdleCount == IDLE_COUNT_THRESHOLD)//should we play the effect ?
■■{
■■■player.m_CorpseState = PlayerConstants.CORPSE_STATE_DECAYED;
■■■player.SetSynchDirty();
■■}
■■
■■
■}
■
■override void OnReconnect(PlayerBase player)
```

```
// ---------------------------------------------
// ------ FNX45.c - START --------------------
// ---------------------------------------------


class FNX45_Base : Pistol_Base
{
■override RecoilBase SpawnRecoilObject()
■{
■■return new Fnx45Recoil(this);
■}
■■
■//Debug menu Spawn Ground Special
■override void OnDebugSpawn()
■{
■■GameInventory inventory = GetInventory();
■■inventory.CreateInInventory( "PistolSuppressor" );
■■inventory.CreateInInventory( "FNP45_MRDSOptic" );
■■inventory.CreateInInventory( "TLRLight" );
■■inventory.CreateInInventory( "Battery9V" );
■■inventory.CreateInInventory( "Battery9V" );
■■
■■SpawnAttachedMagazine("Mag_FNX45_15Rnd");
■}
■
};


// ---------------------------------------------
// ------ FNX45.c - END ----------------------
// ---------------------------------------------


// ---------------------------------------------
// ------ Fnx45Recoil.c - START --------------------
// ---------------------------------------------


class Fnx45Recoil: RecoilBase
{
■override void Init()
■{
■■vector point_1;
■■vector point_2;
■■vector point_3;
■■vector point_4;
■■point_1[0] = m_Player.GetRandomGeneratorSyncManager().GetRandomInRange(RandomGeneratorS
■■point_2[0] = m_Player.GetRandomGeneratorSyncManager().GetRandomInRange(RandomGeneratorS
■■point_3[0] = m_Player.GetRandomGeneratorSyncManager().GetRandomInRange(RandomGeneratorS
■■point_4[0] = m_Player.GetRandomGeneratorSyncManager().GetRandomInRange(RandomGeneratorS
■■m_HandsCurvePoints.Insert(point_1);//forms a 2 dimensional spline(z is ignored)
■■m_HandsCurvePoints.Insert(point_2);
■■m_HandsCurvePoints.Insert(point_3);
■■m_HandsCurvePoints.Insert(point_4);
■■m_HandsCurvePoints.Insert("0 0 0");
■■m_HandsOffsetRelativeTime = 1;
■■
■■m_MouseOffsetRangeMin = 45;//in degrees min
■■m_MouseOffsetRangeMax = 105;//in degrees max
■■m_MouseOffsetDistance = 1.2;//how far should the mouse travel
■■m_MouseOffsetRelativeTime = 0.5;//[0..1] a time it takes to move the mouse the required distance rela
■
■■m_CamOffsetDistance = 0.04;
```

```
// ---------------------------------------------
// ------ FoodPoisonAgent.c - START --------------------
// ---------------------------------------------


class FoodPoisonAgent extends AgentBase
{
■override void Init()
■{
■■m_Type ■■■■■= eAgents.FOOD_POISON;
■■m_Invasibility ■■■= 1;
■■m_TransferabilityIn■■= 1;
■■m_TransferabilityOut■= 0;
■■m_AntibioticsResistance = 0.5;
■■m_MaxCount ■■■■= 400;
■■m_Potency ■■■■= EStatLevels.CRITICAL;
■■m_DieOffSpeed ■■■= 1;
■■m_Digestibility ■■= 1;
■}
}


// ---------------------------------------------
// ------ FoodPoisonAgent.c - END ----------------------
// ---------------------------------------------


// ---------------------------------------------
// ------ FoodStage.c - START --------------------
// ---------------------------------------------


enum FoodStageType
{
■NONE■■= 0,■■■//food stage not define
■RAW■■■= 1,■■■//default
■BAKED■■= 2,
■BOILED■■= 3,
■DRIED■■= 4,
■BURNED■■= 5,
■ROTTEN■■= 6,
■
■COUNT■■■■■■//for net sync purposes
}

// Used to make getting data more readable
enum eCookingPropertyIndices
{
■MIN_TEMP ■= 0,
■COOK_TIME ■= 1,
■MAX_TEMP ■= 2
}

class FoodStage
{
■protected FoodStageType m_FoodStageTypeClientLast;
■protected FoodStageType m_FoodStageType;
■protected Edible_Base m_FoodItem;
■
■protected int m_SelectionIndex;■■■■//visual properties
■protected int m_TextureIndex;
■protected int m_MaterialIndex;
■
```

```
// ---------------------------------------------
// ------ FoxSteakMeat.c - START --------------------
// ---------------------------------------------


class FoxSteakMeat extends Edible_Base
{
■override bool CanBeCooked()
■{
■■return true;
■}■■
■
■override bool CanBeCookedOnStick()
■{
■■return true;
■}
■
■override bool IsMeat()
■{
■■return true;
■}
■
■override bool CanDecay()
■{
■■return true;
■}
■
■override void SetActions()
■{
■■super.SetActions();
■■
■■AddAction(ActionForceFeed);
■■AddAction(ActionEatMeat);
■■
■■AddAction(ActionCreateIndoorFireplace);
■■AddAction(ActionCreateIndoorOven);
■}
}



// ---------------------------------------------
// ------ FoxSteakMeat.c - END ----------------------
// ---------------------------------------------


// ---------------------------------------------
// ------ FracturedLegNotfr.c - START --------------------
// ---------------------------------------------


class FracturedLegNotfr: NotifierBase
{
■void FracturedLegNotfr(NotifiersManager manager)
■{
■■m_Active = false;
■}

■override int GetNotifierType()
■{
■■return eNotifiers.NTF_FRACTURE;
■}
```

```
// ---------------------------------------------
// ------ FreezeState.c - START --------------------
// ---------------------------------------------


class FreezeSymptom extends SymptomBase
{
■//this is just for the Symptom parameters set-up and is called even if the Symptom doesn't execute, don't
■override void OnInit()
■{
■■m_SymptomType = SymptomTypes.PRIMARY;
■■m_Priority = 1;
■■m_ID = SymptomIDs.SYMPTOM_FREEZE;
■■m_DestroyOnAnimFinish = true;
■■m_SyncToClient = false;
■■m_MaxCount = 2;
■}
■
■//!gets called every frame
■override void OnUpdateServer(PlayerBase player, float deltatime)
■{

■}

■override void OnUpdateClient(PlayerBase player, float deltatime)
■{
■}
■
■override void OnAnimationPlayFailed()
■{
■■
■}
■
■override bool CanActivate()
■{
■■return true;
■}
■
■//!gets called once on an Symptom which is being activated
■override void OnGetActivatedServer(PlayerBase player)
■{■
■■if (LogManager.IsSymptomLogEnable()) Debug.SymptomLog("n/a", this.ToString(), "n/a", "OnGetActiva
■■HumanMovementState hms = new HumanMovementState();
■■player.GetMovementState(hms);
■■ItemBase item = m_Player.GetItemInHands();

■■if(!(item && item.IsHeavyBehaviour()) && m_Manager.GetCurrentCommandID() == DayZPlayerConsta
■■{
■■■PlayAnimationADD(2);
■■}
■■else
■■{
■■■PlaySound(EPlayerSoundEventID.FREEZING);
■■}
■}

■//!gets called once on a Symptom which is being activated
■override void OnGetActivatedClient(PlayerBase player)
■{
■■if (LogManager.IsSymptomLogEnable()) Debug.SymptomLog("n/a", this.ToString(), "n/a", "OnGetActiva
■}

■//!only gets called once on an active Symptom that is being deactivated
```

```
// ---------------------------------------------
// ------ FreezingSoundHandler.c - START --------------------
// ---------------------------------------------


/*
class FreezingSoundHandlerBase extends SoundHandlerBase
{
■override void Init()
■{
■■m_Id = eSoundHandlers.FREEZING;
■}
■
}

//---------------------------
// Client
//---------------------------
class FreezingSoundHandlerClient extends FreezingSoundHandlerBase
{
■const float SOUND_INTERVALS_LIGHT_MIN = 5;■const float SOUND_INTERVALS_LIGHT_MAX = 20
■float m_SoundTime;
■EffectSound m_Sound;
■ref HumanMovementState■hms = new HumanMovementState();
■override void Update()
■{
■■if( m_Player.GetShakeLevel() > 0 )
■■{
■■■ProcessSound();
■■}
■}
■
■void ProcessSound()
■{
■■m_Player.GetMovementState(hms);
■■if (hms.m_iMovement == DayZPlayerConstants.MOVEMENTIDX_IDLE)
■■{
■■■//return;
■■}
■■
■■if( GetGame().GetTime() > m_SoundTime )
■■{
■■■float offset_time = Math.RandomFloatInclusive(SOUND_INTERVALS_LIGHT_MIN, SOUND_INTERV
■■■m_SoundTime = GetGame().GetTime() + offset_time;
■■■PlaySound();
■■}
■}
■
■void PlaySound()
■{
■■//Print("------------- freezing  --------------");
■■m_Player.PlaySoundEvent(EPlayerSoundEventID.FREEZING);
■}
}


//---------------------------
// Server
//---------------------------
class FreezingSoundHandlerServer extends HungerSoundHandlerBase
{
■
}*/
```

```
// ---------------------------------------------
// ------ FryingPan.c - START --------------------
// ---------------------------------------------


class FryingPan extends Inventory_Base
{
■// Cooking data
■protected CookingMethodType ■■m_CookingMethod;
■protected bool ■■■■■■m_CookingIsDone;
■protected bool ■■■■■■m_CookingIsEmpty;
■protected bool ■■■■■■m_CookingIsBurned;

■// Particles
■protected Particle ■m_ParticleCooking;
■protected int ■■m_ParticlePlaying ■= ParticleList.INVALID;

■protected int PARTICLE_BAKING_START ■= ParticleList.COOKING_BAKING_START;
■protected int PARTICLE_BAKING_DONE ■■= ParticleList.COOKING_BAKING_DONE;
■protected int PARTICLE_DRYING_START ■= ParticleList.COOKING_DRYING_START;
■protected int PARTICLE_DRYING_DONE ■■= ParticleList.COOKING_DRYING_DONE;
■protected int PARTICLE_BURNING_DONE■■= ParticleList.COOKING_BURNING_DONE;

■// Sounds
■protected SoundOnVehicle■m_SoundCooking;■//! DEPRECATED
■protected EffectSound■■m_SoundEffectCooking;
■protected string■■■m_SoundPlaying = "";

■const string SOUND_BAKING_START ■= "Baking_SoundSet";■■//! DEPRECATED
■const string SOUND_BAKING_DONE ■■= "Baking_Done_SoundSet";■//! DEPRECATED
■const string SOUND_DRYING_START ■= "Drying_SoundSet";■■//! DEPRECATED
■const string SOUND_DRYING_DONE ■■= "Drying_Done_SoundSet";  ■//! DEPRECATED
■const string SOUND_BURNING_DONE ■= "Food_Burning_SoundSet"; ■//! DEPRECATED

■void FryingPan()
■{
■■RegisterNetSyncVariableInt( "m_CookingMethod", CookingMethodType.NONE, CookingMethodType.C
■■RegisterNetSyncVariableBool( "m_CookingIsDone" );
■■RegisterNetSyncVariableBool( "m_CookingIsEmpty" );
■■RegisterNetSyncVariableBool( "m_CookingIsBurned" );
■}
■void ~FryingPan() {}

■override bool IsContainer()
■{
■■return true;■■
■}
■
■override bool CanHaveTemperature()
■{
■■return true;
■}

■override bool CanPutInCargo( EntityAI parent )
■{
■■if ( !super.CanPutInCargo( parent ) )
■■■return false;
■■
■■if ( parent && IsCargoException4x3( parent ) )
■■■return false;

■■return true;
■}
```

```
// ---------------------------------------------
// ------ FSMBase.c - START --------------------
// ---------------------------------------------


void fsmbDebugPrint (string s)
{
■//Print("" + s); // comment/uncomment to hide/see debug logs
}
void fsmbDebugSpam (string s)
{
■//Print("" + s); // comment/uncomment to hide/see debug spam
}


/**@class■FSMTransition
 * @brief■represents transition src ---- event[guard]/action ----|> dst
 **/
class FSMTransition<Class FSMStateBase, Class FSMEventBase, Class FSMActionBase, Class FSMGu
{
■ref FSMStateBase m_srcState;
■ref FSMEventBase m_event; // @NOTE: NULL event means "completion transition" in UML speak
■ref FSMStateBase m_dstState; // @NOTE: NULL dst state == UML terminate pseudonode
■ref FSMActionBase m_action;
■ref FSMGuardBase m_guard;

■void FSMTransition (FSMStateBase src, FSMEventBase e, FSMStateBase dst, FSMActionBase a = NU
■{
■■m_srcState = src;
■■m_event = e;
■■m_dstState = dst;
■■m_action = a;
■■m_guard = g;
■}
};

enum ProcessEventResult
{
■FSM_OK,
■FSM_TERMINATED,
■FSM_ABORTED,
■FSM_NO_TRANSITION,
};

/**@class■■FSMBase
 * @brief■■base class for finite state machine
 *
 * stores current state (m_state) and transition table with possible transitions from each state
 * to another state via event
 **/
class FSMBase<Class FSMStateBase, Class FSMEventBase, Class FSMActionBase, Class FSMGuardB
{
■protected ref FSMStateBase m_state; /// current fsm state
■protected ref FSMStateBase m_initialState; /// configurable initial state of the machine
■protected ref FSMEventBase m_initialEvent; /// configurable initial event to start the machine (null by def
■protected ref array<ref FSMTransition<FSMStateBase, FSMEventBase, FSMActionBase, FSMGuardBa

■void FSMBase ()
■{
■■m_transitions = new array<ref FSMTransition<FSMStateBase, FSMEventBase, FSMActionBase, FSM
■}

■/**@fn■■■GetCurrentState
```

```
// ----------------------------------------------
// ------ FuelChainsaw.c - START --------------------
// ----------------------------------------------


class FuelChainsaw extends RecipeBase
{■
■override void Init()
■{
■■m_Name = "#refuel";
■■m_IsInstaRecipe = false;//should this recipe be performed instantly without animation
■■m_AnimationLength = 1.5;//animation length in relative time units
■■m_Specialty = 0.01;// value > 0 for roughness, value < 0 for precision
■■
■■
■■//conditions
■■m_MinDamageIngredient[0] = -1;//-1 = disable check
■■m_MaxDamageIngredient[0] = 3;//-1 = disable check
■■
■■m_MinQuantityIngredient[0] = -1;//-1 = disable check
■■m_MaxQuantityIngredient[0] = -1;//-1 = disable check
■■
■■m_MinDamageIngredient[1] = -1;//-1 = disable check
■■m_MaxDamageIngredient[1] = 3;//-1 = disable check
■■
■■m_MinQuantityIngredient[1] = 0;//-1 = disable check
■■m_MaxQuantityIngredient[1] = -1;//-1 = disable check
■■//----------------------------------------------------------------------------------------------------------------------
■■
■■//INGREDIENTS
■■//ingredient 1
■■InsertIngredient(0,"Chainsaw");//you can insert multiple ingredients this way
■■
■■m_IngredientAddHealth[0] = 0;// 0 = do nothing
■■m_IngredientSetHealth[0] = -1; // -1 = do nothing
■■m_IngredientAddQuantity[0] = 0;// 0 = do nothing
■■m_IngredientDestroy[0] = false;//true = destroy, false = do nothing
■■m_IngredientUseSoftSkills[0] = false;// set 'true' to allow modification of the values by softskills on this i
■■
■■//ingredient 2
■■InsertIngredient(1,"Pot");//you can insert multiple ingredients this way
■■InsertIngredient(1,"CanisterGasoline");//you can insert multiple ingredients this way
■■InsertIngredient(1,"DisinfectantAlcohol");//you can insert multiple ingredients this way
■■InsertIngredient(1,"Canteen");//you can insert multiple ingredients this way
■■InsertIngredient(1,"WaterBottle");//you can insert multiple ingredients this way
■■InsertIngredient(1,"Vodka");//you can insert multiple ingredients this way
■■InsertIngredient(1,"WaterPouch_ColorBase");//you can insert multiple ingredients this way
■■InsertIngredient(1,"Barrel_ColorBase");//you can insert multiple ingredients this way
■■
■■m_IngredientAddHealth[1] = 0;// 0 = do nothing
■■m_IngredientSetHealth[1] = -1; // -1 = do nothing
■■m_IngredientAddQuantity[1] = 0;// 0 = do nothing
■■m_IngredientDestroy[1] = false;// false = do nothing
■■m_IngredientUseSoftSkills[1] = false;// set 'true' to allow modification of the values by softskills on this i
■■//----------------------------------------------------------------------------------------------------------------------
■■
■■//result1
■■//AddResult("");//add results here

■■m_ResultSetFullQuantity[0] = false;//true = set full quantity, false = do nothing
■■m_ResultSetQuantity[0] = -1;//-1 = do nothing
■■m_ResultSetHealth[0] = -1;//-1 = do nothing
■■m_ResultInheritsHealth[0] = -1;// (value) == -1 means do nothing; a (value) >= 0 means this result will i
```

```
// ---------------------------------------------
// ------ FuelStation.c - START --------------------
// ---------------------------------------------


class FuelStation extends BuildingSuper
{
■override bool IsBuilding()
■{
■■return false;
■}
■
■override bool IsFuelStation()
■{
■■return true;
■}
■
■override float GetLiquidThroughputCoef()
■{
■■return LIQUID_THROUGHPUT_FUELSTATION;
■}
}


// ---------------------------------------------
// ------ FuelStation.c - END ----------------------
// ---------------------------------------------


// ---------------------------------------------
// ------ FurCourierBag.c - START --------------------
// ---------------------------------------------


class FurCourierBag : Clothing {};



// ---------------------------------------------
// ------ FurCourierBag.c - END ----------------------
// ---------------------------------------------


// ---------------------------------------------
// ------ FurImprovisedBag.c - START --------------------
// ---------------------------------------------


class FurImprovisedBag : Clothing {};



// ---------------------------------------------
// ------ FurImprovisedBag.c - END ----------------------
// ---------------------------------------------


// ---------------------------------------------
// ------ Game.c - START --------------------
// ---------------------------------------------


// #include "Scripts/DayZGame.c"
// ---------------------------------------------
// ------ gameLib.c - START --------------------
// ---------------------------------------------


#ifdef GAME_TEMPLATE
Game g_Game;

Game GetGame()
{
■return g_Game;
}

class Game
{
■ScriptModule GameScript;
■
■ScriptModule GetScriptModule()
■{
■■return GameScript;
■}
■
■void SetDebug(bool isDebug) {}
■//!
■/**
  ■\brief Called when some system event occur.
  ■@param eventTypeId event type.
■@param params Param object, cast to specific param class to get parameters for particular event.
■*/
■void OnEvent(EventType eventTypeId, Param params)
■{
■■Print("OnEvent");
■}

■/**
■\brief Called after full initialization of Game instance
■*/
■void OnAfterInit()
■{
■■Print("OnAfterInit");
■}
■
■/**
  ■\brief Called on World update
■@param timeslice time elapsed from last call
■*/
■void OnUpdate(float timeslice)
■{

■}

■/**
  ■\brief Sets world file to be loaded. Returns false if file doesn't exist.
■@param path Path to the ent file
■@param reload Force reload the world
■*/
■proto native bool SetWorldFile(string path, bool reload);

■/**
  ■\brief Returns path of world file loaded
■*/
■proto native owned string GetWorldFile();
```

```
// ---------------------------------------------
// ------ gameLibComponents.c - START --------------------
// ---------------------------------------------


#ifdef COMPONENT_SYSTEM
//Generic components from GameLib (script side of c++ classes)

typedef int[] IEntityComponentSource;
class IEntityComponentSource: BaseContainer
{
};

//!Entity touch event types
enum TouchEvent
{
■ON_ENTER,
■ON_STAY,
■ON_EXIT
};

//!Builtin component types
//TypeID MeshObjectTypeID;
//TypeID HierarchyTypeID;
//TypeID RigidBodyTypeID;
//TypeID SphereGeometryTypeID;
//TypeID BoxGeometryTypeID;


class GenericComponent : Managed
{
■/**
■* Gets current eventmask of the component.
■* \return Returns bitmask of events the component accepts
■*/
■proto native int GetEventMask();
■
■/**
■* Sets eventmask. Component accepts only events which has set bits in eventmask.
■* Bits are or'ed with existing bitmask. See enf::EntityEvents.
■* When this method is called in the constructor of the component, it will not properly set the eventmask to
■* \param mask Mask of those bits, which will be set.
■* \return Returns bitmask of events the component accepts (result of mask|GetEventMask())
■*/
■proto native int SetEventMask(IEntity owner, int mask);
■
■/**
■* Clears bitmask. Component accepts only events which has set bits in eventmask.
■* Only bits set in the mask are cleared. See enf::EntityEvents
■* \param mask Mask of those bits, which will be cleared.
■* \return returns these bits that were set before and now are cleared.
■*/
■proto native int ClearEventMask(IEntity owner, int mask);

■/**
■* Activate component and calls EOnActivate().
■*/
■proto native void Activate(IEntity owner);

■/**
■* Deactivate component and calls EOnDectivate().
■*/
■proto native void Deactivate(IEntity owner);
```

```
// ---------------------------------------------
// ------ gameLibEntities.c - START --------------------
// ---------------------------------------------


//Generic entities from GameLib (script side of c++ classes)

#ifdef COMPONENT_SYSTEM
class GenericEntity extends IEntity
{
■//native method implemented on c++ side
■proto native void Show(bool show);
■
■/*!
■Finds first occurance of the coresponding component.
■\param typeName type of the component
■*/
■proto native GenericComponent FindComponent(typename typeName);
■
■/*!
■inserts instance of the script component to the entity.
■Calls OnComponentInsert on all entity's components to inform that this new component was inserted. Th
■Calls EOnInit on the component if component sets EV_INIT mask.
■Calls EOnActivate on the component if the component is active (default)
■\param component instance
■*/
■proto native void InsertComponent(GenericComponent component);
■
■/*!
■Removes component from entity. Doesn't delete the entity.
■Calls EOnDeactivate on the component.
■Calls OnComponentRemove on all entity's components to inform that this new component was removed
■\param component instance
■*/
■proto native void RemoveComponent(GenericComponent component);
■
■/*!
■Removes and deletes component from entity.
■Calls EOnDeactivate on the component.
■Calls OnComponentRemove on all entity's components to inform that this new component was removed
■Calls OnDelete on the component.
■\param component instance
■*/
■proto native void DeleteComponent(GenericComponent component);

#ifdef WORKBENCH
■/*!
■Called after updating world in Workbench. The entity must be selected.
■*/
■void _WB_AfterWorldUpdate(float timeSlice) {};
#endif
}

class GenericWorldEntity extends GenericEntity
{
}

class GenericTerrainEntity extends GenericEntity
{
}

class LightEntity extends GenericEntity
{
```

```
// ----------------------------------------------
// ------ gameplay.c - START --------------------
// ----------------------------------------------



/** @file */

/// tree traversal type, for more see http://en.wikipedia.org/wiki/Tree_traversal
enum InventoryTraversalType
{
■PREORDER,
■INORDER,
■POSTORDER,
■LEVELORDER
};

const int INDEX_NOT_FOUND = -1;
//---------------------------------------------------------------------------
typedef Serializer ParamsReadContext;
typedef Serializer ParamsWriteContext;

/**
\brief Class for sending RPC over network
■@code
■■// example sending
■■void Send()
■■{
■■■ScriptRPC rpc = new ScriptRPC();
■
■■■rpc.Write(645);
■■■rpc.Write("hello");
■■■
■■■array<float> farray = {1.2, 5.6, 8.1};
■■■rpc.Write(farray);
■
■■■rpc.Send(m_Player, ERPCs.RPC_TEST, true, m_Player.GetIdentity());
■■}

■■// example receive
■■void OnRPC(ParamsReadContext ctx)
■■{
■■■int num;
■■■string text;
■■■array<float> farray;

■■■ctx.Read(num);
■■■ctx.Read(text);
■■■ctx.Read(farray);
■■}
■@endcode
*/

class JsonSerializer: Serializer
{
■void JsonSerializer() {}
■void ~JsonSerializer() {}
■
■/**
■\brief Script variable serialization to json string
■@param variable_out script variable to be serialized to string
■@param nice if the string should be formated for human readability
■@param result from the serialization, output or error depending on the return value
```

```
// ----------------------------------------------
// ------ GameplayEffectWidgets.c - START --------------------
// ----------------------------------------------


/*
TODO - doxygen formating
*/

//! grouped gameplay effect widgets and their handling
class GameplayEffectWidgets extends GameplayEffectWidgets_base
{
■protected ref Widget ■■■■■■■■■m_Root; //dummy parent node
■protected ref map<int,ref Widget> ■■■■■m_Layouts;
■protected ref set<ref Widget> ■■■■■■m_UniqueLayouts;
■protected ref GameplayEffectDataMap ■■■■m_WidgetDataMap;
■protected ref set<int> ■■■■■■■■■m_RunningEffects;
■protected ref set<int> ■■■■■■■■■m_RunningEffectsPrevious;
■protected ref array<int> ■■■■■■■■m_UpdatingEffects;
■protected ref array<ref Widget> ■■■■■m_UpdatedWidgetsCheck; //to make sure widgets are not upda
■protected ref array<int> ■■■■■■■■m_UpdatedWidgetSetsCheck; //to make sure sets are not updated o
■protected ref set<int> ■■■■■■■■■m_SuspendRequests;
■protected ref map<int,typename> ■■■■■m_IDToTypeMap;
■
■protected float ■■■■■■■■■■m_TimeProgBreath;
■protected float ■■■■■■■■■■m_BreathMultStamina;
■protected float ■■■■■■■■■■m_BreathResidue;
■
■//UserID's for widget containers that use something different from 'EffectWidgetsTypes' defaults
■protected const int ■■■■■■■■■WIDGETSET_BREATH = 100;
■
■//effect values
■protected int ■■■■■■■■■■■m_BreathColor;
■protected float ■■■■■■■■■■m_BreathAlphaVal;
■protected float ■■■■■■■■■■m_FlashbangCoverAlphaVal;
■
■void GameplayEffectWidgets()
■{
■■m_Root = GetGame().GetWorkspace().CreateWidget(FrameWidgetTypeID,0,0,1.0,1.0,WidgetFlags.VIS
■■m_Layouts = new map<int,ref Widget>;
■■m_UniqueLayouts = new set<ref Widget>;
■■m_WidgetDataMap = new GameplayEffectDataMap;
■■m_RunningEffects = new set<int>;
■■m_RunningEffectsPrevious = new set<int>;
■■m_UpdatingEffects = new array<int>;
■■m_UpdatedWidgetsCheck = new array<ref Widget>;
■■m_UpdatedWidgetSetsCheck = new array<int>;
■■m_SuspendRequests = new set<int>;
■■m_IDToTypeMap = new map<int,typename>;
■■
■■m_TimeProgBreath = 0.0;
■■m_BreathMultStamina = 1.0;
■■
■■PairIDToTypes();
■■
■■RegisterLayouts("gui/layouts/gameplay/CameraEffects.layout",CompileEffectListing());
■■RegisterLayouts("gui/layouts/gameplay/BleedingEffects.layout",{EffectWidgetsTypes.BLEEDING_LAYE
■■
■■InitWidgetSet(EffectWidgetsTypes.MASK_BREATH,true,WIDGETSET_BREATH);
■■InitWidgetSet(EffectWidgetsTypes.HELMET_BREATH,true,WIDGETSET_BREATH);
■■InitWidgetSet(EffectWidgetsTypes.MOTO_BREATH,true,WIDGETSET_BREATH);
■■
■■InitWidgetSet(EffectWidgetsTypes.MASK_OCCLUDER,false,EffectWidgetsTypes.MASK_OCCLUDER
// ---------------------------------------------
// ------ GameplayEffectWidgets_base.c - START --------------------
// ---------------------------------------------


class GameplayEffectWidgets_base extends Managed
{
■int m_MaskWidgetUpdateCount;//number of times the widget was updated through a single voice event

■void IncreaseMaskUpdateCount(){m_MaskWidgetUpdateCount++};
■void ResetMaskUpdateCount(){m_MaskWidgetUpdateCount = 0};
■
■bool IsAnyEffectRunning(){}
■bool AreEffectsSuspended(){}
■void AddActiveEffects(array<int> effects){}
■void RemoveActiveEffects(array<int> effects){}
■void StopAllEffects(){}
■void AddSuspendRequest(int request_id){}
■void RemoveSuspendRequest(int request_id){}
■void ClearSuspendRequests(){}
■int GetSuspendRequestCount(){}
■void UpdateWidgets(int type = -1, float timeSlice = 0, Param p = null, int handle = -1){}
■void Update(float timeSlice){}
■void OnVoiceEvent(float breathing_resistance01){}
■void SetBreathIntensityStamina(float stamina_cap, float stamina_current){}
■
■void RegisterGameplayEffectData(int id, Param p){}
}


// ---------------------------------------------
// ------ GameplayEffectWidgets_base.c - END ----------------------
// ---------------------------------------------


// ---------------------------------------------
// ------ GardenBase.c - START --------------------
// ---------------------------------------------


class GardenBase extends ItemBase //BuildingSuper
{
■// Paths to slot textures. Slots can have multiple states, so multiple textures must be generated
■static const string SLOT_TEXTURE_DIGGED_WET_LIME■■= "dz\\gear\\cultivation\\data\\soil_digged_
■static const string SLOT_TEXTURE_DIGGED_WET_PLANT■■= "dz\\gear\\cultivation\\data\\soil_digged
■
■// Wet/dry material
■static const string SLOT_MATERIAL_WET■■■■■= "dz\\gear\\cultivation\\data\\soil_cultivated_wet.rvma
■static const string SLOT_MATERIAL_DRY■■■■■= "dz\\gear\\cultivation\\data\\soil_cultivated.rvmat";
■
■static const string SLOT_MATERIAL_LIMED_WET■■■■= "dz\\gear\\cultivation\\data\\soil_cultivated_lin
■static const string SLOT_MATERIAL_LIMED_DRY■■■■= "dz\\gear\\cultivation\\data\\soil_cultivated_lin
■static const string SLOT_MATERIAL_COMPOST_WET■■■= "dz\\gear\\cultivation\\data\\soil_cultivated_
■static const string SLOT_MATERIAL_COMPOST_DRY■■■= "dz\\gear\\cultivation\\data\\soil_cultivated_
■
■// slot names -> MUST BE LOWERCASE
■private static const string SLOT_SELECTION_DIGGED_PREFIX ■= "seedbase_";
■private static const string SLOT_SELECTION_COVERED_PREFIX ■= "slotCovered_";
■private static const string SLOT_MEMORY_POINT_PREFIX ■■= "slot_";
■private static const string SLOT_SEEDBASE_PREFIX ■■■= "seedbase_";
■
■
■private static const int ■CHECK_RAIN_INTERVAL ■■■= 15;
```

```
// ----------------------------------------------
// ------ GardenLime.c - START --------------------
// ----------------------------------------------


class GardenLime extends ItemBase
{
■override bool CanPutAsAttachment( EntityAI parent )
■{
■■if(!super.CanPutAsAttachment(parent)) {return false;}
■■const int SLOTS_ARRAY = 8;
■■bool is_barrel = false;
■■bool is_opened_barrel = false;
■■bool slot_test = true;
■■string slot_names[SLOTS_ARRAY] = { "BerryR", "BerryB", "Plant", "OakBark", "BirchBark", "Nails", "Di

■■
■■// is barrel
■■if ( parent.IsKindOf("Barrel_ColorBase") )
■■{
■■■is_barrel = true;
■■}

■■// is opened barrel■■■■
■■if ( is_barrel && parent.GetAnimationPhase("Lid") == 1 )
■■{
■■■is_opened_barrel = true;
■■}

■■// all of the barrel slots are empty
■■for ( int i = 0; i < SLOTS_ARRAY ; i++ )
■■{
■■■if ( parent.FindAttachmentBySlotName(slot_names[i]) != NULL )
■■■{
■■■■slot_test = false;
■■■■break;
■■■}
■■}
■■
■■if ( ( is_opened_barrel && slot_test ) || !is_barrel )
■■{
■■■return true;
■■}
■■return false;
■}

■override bool CanDetachAttachment( EntityAI parent )
■{
■■
■■bool is_barrel = false;
■■bool is_opened_barrel = false;
■■
■■// is barrel
■■if ( parent.IsKindOf("Barrel_ColorBase") )
■■{
■■■is_barrel = true;
■■}

■■// is opened barrel■■■■
■■if ( is_barrel && parent.GetAnimationPhase("Lid") == 1 )
■■{
■■■is_opened_barrel = true;
■■}
```

```
// ---------------------------------------------
// ------ GardenPlot.c - START --------------------
// ---------------------------------------------


class GardenPlot extends GardenBase
{
■Object ■m_ClutterCutter;
■private const int GARDEN_SLOT_COUNT = 9;
■
■void GardenPlot()
■{
■■SetBaseFertility(0.5);
■}
■
■override void EEInit()
■{■
■■super.EEInit();
■}■
■
■override bool OnStoreLoad( ParamsReadContext ctx, int version )
■{■■■■
■■if ( !super.OnStoreLoad(ctx, version) )
■■■return false;

■■if ( !m_ClutterCutter )
■■{■■
■■■m_ClutterCutter = GetGame().CreateObjectEx( "ClutterCutter6x6", GetPosition(), ECE_PLACE_ON_
■■■m_ClutterCutter.SetOrientation( GetOrientation() );
■■}

■■return true;
■}

■override void EEDelete(EntityAI parent)
■{
■■super.EEDelete(parent);
■■
■■if (m_ClutterCutter  &&  GetGame())
■■{
■■■GetGame().ObjectDelete(m_ClutterCutter);
■■■m_ClutterCutter = NULL;
■■}
■}
■
■override bool IsInventoryVisible()
■{
■■return true;
■}

■override int GetGardenSlotsCount()
■{
■■return GARDEN_SLOT_COUNT;
■}

■void RefreshSlots()
■{
■■HideSelection("SeedBase_1");
■■HideSelection("SeedBase_2");
■■HideSelection("SeedBase_3");
■■HideSelection("SeedBase_4");
■■HideSelection("SeedBase_5");
■■HideSelection("SeedBase_6");
```

```
// ---------------------------------------------
// ------ GasMask.c - START --------------------
// ---------------------------------------------


class GasMask extends MaskBase
{
■override bool CanPutAsAttachment( EntityAI parent )
■{
■■if (!super.CanPutAsAttachment(parent)) {return false;}
■■
■■Clothing headgear = Clothing.Cast(parent.FindAttachmentBySlotName("Headgear"));
■■if ( headgear && headgear.ConfigGetBool("noMask") )
■■{
■■■return false;
■■}
■■
■■Clothing eyewear = Clothing.Cast(parent.FindAttachmentBySlotName("Eyewear"));
■■if ( eyewear && SportGlasses_ColorBase.Cast(eyewear) )
■■{
■■■return false;
■■}
■■
■■return true;
■}
■
■override bool IsObstructingVoice()
■{
■■return true;
■}
■
■override int GetVoiceEffect()
■{
■■return VoiceEffectObstruction;
■}
■
■override void EEHealthLevelChanged(int oldLevel, int newLevel, string zone)
■{
■■if (GetGame().IsServer())
■■{
■■■if( newLevel == GameConstants.STATE_RUINED )
■■■{
■■■■SetQuantity(0);
■■■}
■■}
■}
}



// ---------------------------------------------
// ------ GasMask.c - END ----------------------
// ---------------------------------------------


// ---------------------------------------------
// ------ GasMask_Filter.c - START --------------------
// ---------------------------------------------


class GasMask_Filter : ItemBase
{
■/*
```

```
// ---------------------------------------------
// ------ GeneratorSmoke.c - START --------------------
// ---------------------------------------------


class EffGeneratorSmoke : EffectParticle
{
■void EffGeneratorSmoke()
■{
■■SetParticleID(ParticleList.POWER_GENERATOR_SMOKE);
■}
}


// ---------------------------------------------
// ------ GeneratorSmoke.c - END ----------------------
// ---------------------------------------------


// ---------------------------------------------
// ------ GesturesMenu.c - START --------------------
// ---------------------------------------------


enum GestureCategories
{
■CATEGORIES,■■■■//special category selection
■CATEGORY_1,
■CATEGORY_2,
■CATEGORY_3,
■CATEGORY_4,
■CATEGORY_5,
■CONSOLE_GESTURES
}

class GestureMenuItem
{
■protected int ■■■■■m_ID;
■protected string ■■■■m_Name;
■protected GestureCategories ■m_Category;
■protected EmoteBase ■■■m_EmoteClass;
■protected bool ■■■■■m_CanPerformEmote;
■//radial menu
■protected Widget ■■■■m_RadialMenuSelector;
■protected Widget■■■■m_RadialMenuItemCard;
■
■void GestureMenuItem(int id, string name, GestureCategories category)
■{
■■m_ID■■■■= id;
■■m_Name ■■■■= name;
■■m_Category ■■■= category;
■■m_CanPerformEmote ■= true;
■■
■■PlayerBase player;
■■if (Class.CastTo(player,GetGame().GetPlayer()) && category != GestureCategories.CATEGORIES)
■■{
■■■m_EmoteClass = player.GetEmoteManager().GetNameEmoteMap().Get(id);
■■}
■}
■
■string GetName()
■{
■■return m_Name;
```

```
// ---------------------------------------------
// ------ GEWidgetsMetaData.c - START --------------------
// ---------------------------------------------


/*
TODO - doxygen formating
*/

//! generic metadata class
class GameplayEffectsData extends Managed
{
■protected ref array<ref Widget> ■m_WidgetArray;
■int ■■■■■■■■m_Type;
■int ■■■■■■■■m_WidgetSetIdentifier;
■Widget ■■■■■■■■m_LayoutRoot;
■
■void GameplayEffectsData(array<ref Widget> input, int type, int user_override = -1)
■{
■■m_WidgetArray = input;
■■m_Type = type;
■■m_WidgetSetIdentifier = type;
■■
■■if (user_override != -1)
■■{
■■■m_WidgetSetIdentifier = user_override;
■■}
■}
■
■void Init(array<ref Widget> input, int type, Widget layout_root, int user_override = -1)
■{
■■m_WidgetArray = input;
■■m_Type = type;
■■m_WidgetSetIdentifier = type;
■■m_LayoutRoot = layout_root;
■■
■■if (user_override != -1)
■■{
■■■m_WidgetSetIdentifier = user_override;
■■}
■}
■
■array<ref Widget> GetWidgetSet()
■{
■■return m_WidgetArray;
■}
■
■int GetWidgetSetType()
■{
■■return m_Type;
■}
■
■int GetWidgetSetID()
■{
■■return m_WidgetSetIdentifier;
■}
■
■//! Returns 'true' if this class contains update info
■bool HasDefinedHandle()
■{
■■return false;
■}
■
```

```
// ---------------------------------------------
// ------ GEWidgetsMetaDataBleeding.c - START --------------------
// ---------------------------------------------



//! Manages all bleeding indicators and their updates
class GameplayEffectsDataBleeding extends GameplayEffectsData
{
■protected bool m_Initialized; //tied to initialization of 'BleedingSourcesManagerBase' on player object, ski
■protected bool m_Visible; //overall visibility
■protected ref map<int,ref BleedingIndicator> m_RegisteredInstances;
■protected ref array<int> m_CleanupQueue;
■protected ref array<int> m_RunningIndicators;
■protected int m_LastDropIdx;
■protected int m_ImageWidgetCount; //number of available blood drop image widgets
■protected ref map<int,ref array<float>> m_ProbabilityMap;
■protected ref array<Widget> m_PositioningFramesArray;
■protected Widget m_BloodDropsFrame;
■
■protected int m_LastPositionFrameUsed;
■
■void GameplayEffectsDataBleeding(array<ref Widget> input, int type, int user_override = -1)
■{
■■m_RegisteredInstances = new map<int,ref BleedingIndicator>;
■■m_CleanupQueue = new array<int>;
■■m_RunningIndicators = new array<int>;
■■m_Initialized = false;
■}
■
■override void Init(array<ref Widget> input, int type, Widget layout_root, int user_override = -1)
■{
■■super.Init(input, type, layout_root, user_override);
■■
■■m_WidgetArray.ShuffleArray(); //shuffles order of the widgets on every init
■■m_ImageWidgetCount = m_WidgetArray.Count();
■■m_BloodDropsFrame = m_LayoutRoot.FindAnyWidgetById(EffectWidgetsTypes.BLEEDING_LAYER);
■■m_Visible = g_Game.GetProfileOption(EDayZProfilesOptions.BLEEDINGINDICATION);
■■m_BloodDropsFrame.Show(m_Visible);
■■m_LastDropIdx = -1;
■■m_LastPositionFrameUsed = -1;
■■
■■BuildProbabilityData(BleedingIndicationConstants.INDICATOR_SEVERITY_LOW,BleedingIndicationC
■■BuildProbabilityData(BleedingIndicationConstants.INDICATOR_SEVERITY_MEDIUM,BleedingIndicati
■■BuildProbabilityData(BleedingIndicationConstants.INDICATOR_SEVERITY_HIGH,BleedingIndicationC
■■BuildPositioningData();
■}
■
■override bool HasDefinedHandle()
■{
■■return true;
■}
■
■override bool DataInitialized()
■{
■■return m_Initialized;
■}
■
■override void RegisterData(Param p)
■{
■■if (m_Initialized)
■■{
■■■#ifdef DEVELOPER
```

```
// ---------------------------------------------
// ------ GhillieBushrag_ColorBase.c - START --------------------
// ---------------------------------------------


class GhillieBushrag_ColorBase extends Clothing {};
class GhillieBushrag_Tan extends GhillieBushrag_ColorBase {};
class GhillieBushrag_Woodland extends GhillieBushrag_ColorBase {};
class GhillieBushrag_Mossy extends GhillieBushrag_ColorBase {};


// ---------------------------------------------
// ------ GhillieBushrag_ColorBase.c - END ----------------------
// ---------------------------------------------


// ---------------------------------------------
// ------ GhillieHood_ColorBase.c - START --------------------
// ---------------------------------------------


class GhillieHood_ColorBase extends Clothing {};
class GhillieHood_Tan extends GhillieHood_ColorBase {};
class GhillieHood_Woodland extends GhillieHood_ColorBase {};
class GhillieHood_Mossy extends GhillieHood_ColorBase {};


// ---------------------------------------------
// ------ GhillieHood_ColorBase.c - END ----------------------
// ---------------------------------------------


// ---------------------------------------------
// ------ GhillieSuit_ColorBase.c - START --------------------
// ---------------------------------------------


class GhillieSuit_ColorBase extends Clothing {};
class GhillieSuit_Tan extends GhillieSuit_ColorBase {};
class GhillieSuit_Woodland extends GhillieSuit_ColorBase {};
class GhillieSuit_Mossy extends GhillieSuit_ColorBase {};


// ---------------------------------------------
// ------ GhillieSuit_ColorBase.c - END ----------------------
// ---------------------------------------------


// ---------------------------------------------
// ------ GhillieTop_ColorBase.c - START --------------------
// ---------------------------------------------


class GhillieTop_ColorBase extends Clothing {};
class GhillieTop_Tan extends GhillieTop_ColorBase {};
class GhillieTop_Woodland extends GhillieTop_ColorBase {};
class GhillieTop_Mossy extends GhillieTop_ColorBase {};


// ---------------------------------------------
// ------ GhillieTop_ColorBase.c - END ----------------------
// ---------------------------------------------
// ---------------------------------------------
// ------ GiftBox_Base.c - START --------------------
// ---------------------------------------------


class GiftBox_Base extends Container_Base
{
■protected vector m_HalfExtents; // The Y value contains a heightoffset and not the halfextent !!!
■
■void GiftBox_Base()
■{
■■m_HalfExtents = vector.Zero;
■■m_Openable = new OpenableBehaviour(false);
■■RegisterNetSyncVariableBool("m_Openable.m_IsOpened");
■}
■
■protected ref OpenableBehaviour m_Openable;
■
■override bool CanReceiveItemIntoCargo( EntityAI item )
■{
■■if( !super.CanReceiveItemIntoCargo(item) ) {return false;}
■■
■■if (GameInventory.GetInventoryCheckContext() == InventoryCheckContext.DEFAULT )
■■{
■■■if(!GetGame().IsDedicatedServer())
■■■■return IsOpen();
■■}
■■
■■return true;
■}
■
■
■override void Open()
■{
■■m_Openable.Open();
■■SetSynchDirty();
■}
■
■override void Close()
■{
■■m_Openable.Close();
■■SetSynchDirty();
■}
■
■override bool IsOpen()
■{
■■return m_Openable.IsOpened();
■}
■
■override void SetActions()
■{
■■super.SetActions();
■■AddAction(ActionUnpackGift);
■}
■
■override void OnDebugSpawn()
■{
■■EntityAI entity;
■■if ( Class.CastTo(entity, this) )
■■{
■■■entity.GetInventory().CreateInInventory( "Chemlight_Green" );■
■■}
■}
```

```
// ---------------------------------------------
// ------ GiftWrapPaper.c - START --------------------
// ---------------------------------------------


class GiftWrapPaper extends ItemBase
{
■//=================================================================
■// IGNITION ACTION
■//=================================================================
■override bool HasFlammableMaterial()
■{
■■return true;
■}
■
■override bool CanBeIgnitedBy( EntityAI igniter = NULL )
■{
■■if ( GetHierarchyParent() ) return false;
■■
■■return true;
■}
■
■override bool CanIgniteItem( EntityAI ignite_target = NULL )
■{
■■return false;
■}
■
■override void OnIgnitedTarget( EntityAI ignited_item )
■{
■}
■
■override void OnIgnitedThis( EntityAI fire_source )
■{
■■Fireplace.IgniteEntityAsFireplace( this, fire_source );
■}

■override bool IsThisIgnitionSuccessful( EntityAI item_source = NULL )
■{
■■return Fireplace.CanIgniteEntityAsFireplace( this );
■}
■
■override void SetActions()
■{
■■AddAction(ActionPackGift);
■■super.SetActions();

■■AddAction(ActionCreateIndoorFireplace);
■■AddAction(ActionCreateIndoorOven);
■■AddAction(ActionAttach);
■■AddAction(ActionDetach);
■}
}


// ---------------------------------------------
// ------ GiftWrapPaper.c - END ---------------------
// ---------------------------------------------
// ---------------------------------------------
// ------ Glock.c - START --------------------
// ---------------------------------------------


class Glock19_Base : Pistol_Base
{
■override RecoilBase SpawnRecoilObject()
■{
■■return new GlockRecoil(this);
■}
■
■//Debug menu Spawn Ground Special
■override void OnDebugSpawn()
■{
■■GameInventory inventory = GetInventory();
■■inventory.CreateInInventory( "PistolSuppressor" );
■■inventory.CreateInInventory( "FNP45_MRDSOptic" );
■■inventory.CreateInInventory( "TLRLight" );
■■inventory.CreateInInventory( "Battery9V" );
■■inventory.CreateInInventory( "Battery9V" );
■■
■■SpawnAttachedMagazine("Mag_Glock_15Rnd");
■}
■
};


// ---------------------------------------------
// ------ Glock.c - END ---------------------
// ---------------------------------------------


// ---------------------------------------------
// ------ GlockRecoil.c - START --------------------
// ---------------------------------------------


class GlockRecoil: RecoilBase
{
■override void Init()
■{
■■vector point_1;
■■vector point_2;
■■vector point_3;
■■vector point_4;
■■point_1[0] = m_Player.GetRandomGeneratorSyncManager().GetRandomInRange(RandomGeneratorS
■■point_2[0] = m_Player.GetRandomGeneratorSyncManager().GetRandomInRange(RandomGeneratorS
■■point_3[0] = m_Player.GetRandomGeneratorSyncManager().GetRandomInRange(RandomGeneratorS
■■point_4[0] = m_Player.GetRandomGeneratorSyncManager().GetRandomInRange(RandomGeneratorS
■■m_HandsCurvePoints.Insert(point_1);//forms a 2 dimensional spline(z is ignored)
■■m_HandsCurvePoints.Insert(point_2);
■■m_HandsCurvePoints.Insert(point_3);
■■m_HandsCurvePoints.Insert(point_4);
■■m_HandsCurvePoints.Insert("0 0 0");
■■m_HandsOffsetRelativeTime = 1;
■■
■■m_MouseOffsetRangeMin = 40;//in degrees min
■■m_MouseOffsetRangeMax = 90;//in degrees max
■■m_MouseOffsetDistance = 1;//how far should the mouse travel
■■m_MouseOffsetRelativeTime = 0.5;//[0..1] a time it takes to move the mouse the required distance rela
■
■■m_CamOffsetDistance = 0.04;
```

```
// ----------------------------------------------
// ------ GoatSteakMeat.c - START --------------------
// ----------------------------------------------


class GoatSteakMeat extends Edible_Base
{
■override bool CanBeCooked()
■{
■■return true;
■}■■
■
■override bool CanBeCookedOnStick()
■{
■■return true;
■}
■
■override bool IsMeat()
■{
■■return true;
■}

■override bool CanDecay()
■{
■■return true;
■}
■
■override void SetActions()
■{
■■super.SetActions();
■■
■■AddAction(ActionForceFeed);
■■AddAction(ActionEatMeat);
■■
■■AddAction(ActionCreateIndoorFireplace);
■■AddAction(ActionCreateIndoorOven);
■}
}



// ----------------------------------------------
// ------ GoatSteakMeat.c - END ----------------------
// ----------------------------------------------


// ----------------------------------------------
// ------ GorkaEJacket_ColorBase.c - START --------------------
// ----------------------------------------------


class GorkaEJacket_ColorBase extends Clothing
{
■override void SetActions()
■{
■■super.SetActions();
■■AddAction(ActionWringClothes);
■}
};
class GorkaEJacket_Summer extends GorkaEJacket_ColorBase {};
class GorkaEJacket_Flat extends GorkaEJacket_ColorBase {};
class GorkaEJacket_Autumn extends GorkaEJacket_ColorBase {};
class GorkaEJacket_PautRev extends GorkaEJacket_ColorBase {};
```

```
// ---------------------------------------------
// ------ GorkaHelmetComplete.c - START --------------------
// ---------------------------------------------


class GorkaHelmet extends HelmetBase
{
■//Debug menu Spawn Ground Special
■override void OnDebugSpawn()
■{
■■EntityAI entity;
■■if ( Class.CastTo(entity, this) )
■■{
■■■entity.GetInventory().CreateInInventory( "GorkaHelmetVisor" );
■■}
■}
}


// ---------------------------------------------
// ------ GorkaHelmetComplete.c - END ----------------------
// ---------------------------------------------


// ---------------------------------------------
// ------ GorkaPants_ColorBase.c - START --------------------
// ---------------------------------------------


class GorkaPants_ColorBase extends Clothing
{
■override void SetActions()
■{
■■super.SetActions();
■■AddAction(ActionWringClothes);
■}
};
class GorkaPants_Summer extends GorkaPants_ColorBase {};
class GorkaPants_Autumn extends GorkaPants_ColorBase {};
class GorkaPants_Flat extends GorkaPants_ColorBase {};
class GorkaPants_PautRev extends GorkaPants_ColorBase {};


// ---------------------------------------------
// ------ GorkaPants_ColorBase.c - END ----------------------
// ---------------------------------------------


// ---------------------------------------------
// ------ GP5GasMask.c - START --------------------
// ---------------------------------------------


class GP5GasMask extends MaskBase
{
■override bool CanPutAsAttachment( EntityAI parent )
■{
■■if (!super.CanPutAsAttachment(parent)) {return false;}
■■
■■Clothing headgear = Clothing.Cast(parent.FindAttachmentBySlotName("Headgear"));
■■if ( headgear && headgear.ConfigGetBool("noMask") )
■■{
■■■return false;
```

```
// ---------------------------------------------
// ------ GPSReceiver.c - START --------------------
// ---------------------------------------------


class GPSReceiver : ItemGPS
{

■const string DISPLAY_TEXTURE_PATH_FORMAT ■= "dz\\gear\\navigation\\data\\GPS_%1_ca.paa";
■const string ANIM_PHASE_DISPLAY_HIDE ■■= "DisplayState";

■protected ref set<string> ■■m_DisplayGridSelections;
■protected ref set<string> ■■m_DisplayAltSelections;
■protected ref map<int, string> ■m_DisplayNumericSignTextureMap;
■
■//! cache
■protected ref array<int> ■■m_OrderedPositionNumbersLast;
■protected ref array<int> ■■m_AltitudeNumbersLast;
■
■void GPSReceiver()
■{
■■m_OrderedPositionNumbersLast ■= new array<int>;
■■m_AltitudeNumbersLast ■■■= new array<int>;
■■
■■int i;
■■for (i = 0; i < MapNavigationBehaviour.DISPLAY_GRID_POS_MAX_CHARS_COUNT * 2; ++i)
■■{
■■■m_OrderedPositionNumbersLast.Insert(0);
■■}
■■
■■for (i = 0; i < MapNavigationBehaviour.DISPLAY_ALT_MAX_CHARS_COUNT; ++i)
■■{
■■■m_AltitudeNumbersLast.Insert(0);
■■}■
■■
■■m_DisplayGridSelections = new set<string>();
■■m_DisplayGridSelections.Insert("grid_1_0");
■■m_DisplayGridSelections.Insert("grid_1_1");
■■m_DisplayGridSelections.Insert("grid_1_2");
■■m_DisplayGridSelections.Insert("grid_2_0");
■■m_DisplayGridSelections.Insert("grid_2_1");
■■m_DisplayGridSelections.Insert("grid_2_2");
■■
■■m_DisplayAltSelections = new set<string>();
■■m_DisplayAltSelections.Insert("alt_0");
■■m_DisplayAltSelections.Insert("alt_1");
■■m_DisplayAltSelections.Insert("alt_2");
■■m_DisplayAltSelections.Insert("alt_3");
■■
■■m_DisplayNumericSignTextureMap = new map<int, string>();
■■for (i = -1; i < 11; i++)
■■{
■■■
■■■string texturePath = string.Format(DISPLAY_TEXTURE_PATH_FORMAT, i);
■■■
■■■//! translate numerical -1 to `-` texture
■■■if (i == -1)
■■■{
■■■■texturePath = string.Format(DISPLAY_TEXTURE_PATH_FORMAT, "dash");
■■■}

■■■m_DisplayNumericSignTextureMap.Insert(i, texturePath);
■■}
```

```
// ---------------------------------------------
// ------ GreatHelm.c - START --------------------
// ---------------------------------------------


class GreatHelm extends ClothingBase
{
■override array<int> GetEffectWidgetTypes()
■{
■■return {EffectWidgetsTypes.HELMET_OCCLUDER/*,EffectWidgetsTypes.HELMET_BREATH*/};
■}
■
■override bool CanPutAsAttachment( EntityAI parent )
■{
■■if(!super.CanPutAsAttachment(parent)) {return false;}
■■
■■Clothing eyewear = Clothing.Cast(parent.FindAttachmentBySlotName("Eyewear"));
■■if ( eyewear && eyewear.ConfigGetBool("isStrap") )
■■{
■■■return false;
■■}
■■
■■Clothing mask = Clothing.Cast(parent.FindAttachmentBySlotName("Mask"));
■■if ( mask && mask.ConfigGetBool("noHelmet") ) //TODO
■■{
■■■return false;
■■}
■■
■■return true;
■}
■
■override bool IsObstructingVoice()
■{
■■return true;
■}
■
■override int GetVoiceEffect()
■{
■■return VoiceEffectObstruction;
■}
}


// ---------------------------------------------
// ------ GreatHelm.c - END ----------------------
// ---------------------------------------------


// ---------------------------------------------
// ------ GreenBellPepper.c - START --------------------
// ---------------------------------------------


class GreenBellPepper : Edible_Base
{
■override bool CanBeCookedOnStick()
■{
■■return true;
■}

■override bool CanBeCooked()
■{
■■return true;
```

```
// ---------------------------------------------
// ------ Grenade_Base.c - START --------------------
// ---------------------------------------------


enum EGrenadeType
{
■FRAGMENTATION = 0,
■CHEMICAL,
■ILLUMINATING,
■NON_LETHAL
}

//! For backward compatibility
class GrenadeLight : ExplosiveLight {}

class FlashGrenadeLight : PointLightBase
{
■protected static float m_DefaultBrightness■= 50;
■protected static float m_DefaultRadius■■= 20;
■
■void FlashGrenadeLight()
■{
■■SetVisibleDuringDaylight(true);
■■SetRadiusTo(m_DefaultRadius);
■■SetBrightnessTo(m_DefaultBrightness);
■■SetFlareVisible(false);
■■SetAmbientColor(1.0, 1.0, 1.0);
■■SetDiffuseColor(1.0, 1.0, 1.0);
■■SetLifetime(0.35);
■■SetDisableShadowsWithinRadius(-1);
■}
}

class Grenade_Base : ExplosivesBase
{
■protected const float DEFAULT_FUSE_DELAY ■= 10;

■protected ref Timer■■m_FuseTimer;
■protected float■■■m_FuseDelay;
■protected float ■■m_RemainingFuseTime;
■
■protected bool■■■m_Pinned;
■protected bool■■■m_Pinnable;
■//protected bool■■m_Explodable;■//! DEPRECATED; not used anywhere
■
■protected EGrenadeType■m_GrenadeType;

■void Pin()
■{
■■if (!m_Pinned && m_Pinnable)
■■{
■■■OnPin();
■■}
■}
■
■void Unpin()
■{
■■if (m_Pinned)
■■{
■■■OnUnpin();
■■}
■}■■
```

```
// ---------------------------------------------
// ------ Grenade_ChemGas.c - START --------------------
// ---------------------------------------------


class Grenade_ChemGas : Grenade_Base
{
■protected bool ■■■■■m_Exploded;
■protected ParticleSource ■■m_ParticleExploded;
■protected EffectSound ■■■m_ExplosionSound;
■
■
■void Grenade_ChemGas()
■{
■■SetParticleExplosion(ParticleList.RGD5);
■■SetGrenadeType(EGrenadeType.CHEMICAL);
■■m_Pinned = false;
■■SetPinnable(false);
■■Arm();
■}

■void ~Grenade_ChemGas();

■override protected void OnExplode()
■{
■■m_Exploded = true;
■■
■■if (GetGame().IsServer())
■■{
■■■GetGame().CreateObject("ContaminatedArea_Local", GetPosition());
■■}
■}
■
■protected string GetExplosionSoundSet()
■{
■■return "Grenade_detonation_SoundSet";
■}
■
■override void EOnContact(IEntity other, Contact extra)
■{
■■if (GetGame().IsServer())
■■{
■■■if (!m_Exploded)
■■■{
■■■■OnActivateFinished();
■■■}
■■}
■}
■
■override void EEKilled(Object killer)
■{
■■super.EEKilled(killer);
■}
■
■override void OnDamageDestroyed(int oldLevel)
■{
■■if (GetGame().IsServer() || !GetGame().IsMultiplayer())
■■{
■■■GetGame().GetCallQueue(CALL_CATEGORY_SYSTEM).CallLater(Delete, 1000);
■■}
■■#ifndef SERVER
■■ClearFlags(EntityFlags.VISIBLE, false);
■■m_ParticleExploded = ParticleManager.GetInstance().PlayInWorld(ParticleList.GRENADE_CHEM_BR
// ---------------------------------------------
// ------ GrozaGL_LowerReceiver.c - START --------------------
// ---------------------------------------------


class GrozaGL_LowerReceiver extends ItemBase
{
■override bool CanPutAsAttachment( EntityAI parent )
■{
■■if(!super.CanPutAsAttachment(parent)) {return false;}
■■if ( parent.FindAttachmentBySlotName("weaponMuzzleAK") == NULL )
■■{
■■■return true;
■■}
■■return true;
■}
}


// ---------------------------------------------
// ------ GrozaGL_LowerReceiver.c - END ----------------------
// ---------------------------------------------


// ---------------------------------------------
// ------ Groza_Barrel_Grip.c - START --------------------
// ---------------------------------------------


class Groza_Barrel_Grip extends SuppressorBase
{
■override bool CanPutAsAttachment( EntityAI parent )
■{
■■if(!super.CanPutAsAttachment(parent)) {return false;}
■■if ( !parent.FindAttachmentBySlotName("weaponButtstockAK").IsKindOf("GrozaGL_LowerReceiver") )
■■{
■■■return true;
■■}

■■return false;
■}
}


// ---------------------------------------------
// ------ Groza_Barrel_Grip.c - END ----------------------
// ---------------------------------------------


// ---------------------------------------------
// ------ Groza_Barrel_Short.c - START --------------------
// ---------------------------------------------


class Groza_Barrel_Short extends SuppressorBase
{
■override bool CanPutAsAttachment( EntityAI parent )
■{
■■if(!super.CanPutAsAttachment(parent)) {return false;}
■■if ( !parent.FindAttachmentBySlotName("weaponButtstockAK") || !parent.FindAttachmentBySlotName("
■■{
■■■return true;
■■}
```

```
// ---------------------------------------------
// ------ Groza_Barrel_Suppressor.c - START --------------------
// ---------------------------------------------


class Groza_Barrel_Suppressor extends SuppressorBase
{
■override bool CanPutAsAttachment( EntityAI parent )
■{
■■if(!super.CanPutAsAttachment(parent)) {return false;}
■■if ( !parent.FindAttachmentBySlotName("weaponButtstockAK").IsKindOf("GrozaGL_LowerReceiver") )
■■{
■■■return true;
■■}

■■return false;
■}
}


// ---------------------------------------------
// ------ Groza_Barrel_Suppressor.c - END ----------------------
// ---------------------------------------------


// ---------------------------------------------
// ------ Guards.c - START --------------------
// ---------------------------------------------


/**@class■WeaponGuardBase
 * @brief■represents guard on a transition from state to state
 **/
class WeaponGuardBase
{
■/**@fn■■GuardCondition
■ * @brief enable or disable transition based on condition
■ * the guard is a boolean operation executed first and which can prevent the transition from firing by retur
■ * @return■true if transition is allowed
■ **/
■bool GuardCondition (WeaponEventBase e) { return true; }
};

class GuardAnd extends WeaponGuardBase
{
■ref WeaponGuardBase m_arg0;
■ref WeaponGuardBase m_arg1;

■void GuardAnd (WeaponGuardBase arg0 = NULL, WeaponGuardBase arg1 = NULL) { m_arg0 = arg0; m

■override bool GuardCondition (WeaponEventBase e)
■{
■■bool result = m_arg0.GuardCondition(e) && m_arg1.GuardCondition(e);
■■if (LogManager.IsWeaponLogEnable()) { wpnDebugPrint("[wpnfsm] guard - " + m_arg0.Type() + " && "
■■return result;
■}
};

class GuardNot extends WeaponGuardBase
{
■ref WeaponGuardBase m_arg0;

■void GuardNot (WeaponGuardBase arg0 = NULL) { m_arg0 = arg0; }
```

```
// ---------------------------------------------
// ------ Guts.c - START --------------------
// ---------------------------------------------


class Guts extends Edible_Base
{
■override bool CanPutAsAttachment( EntityAI parent )
■{
■■if(!super.CanPutAsAttachment(parent)) {return false;}
■■const int SLOTS_ARRAY = 8;
■■bool is_barrel = false;
■■bool is_opened_barrel = false;
■■bool slot_test = true;
■■string slot_names[SLOTS_ARRAY] = { "BerryR", "BerryB", "Nails", "OakBark", "BirchBark", "Lime", "Dis

■■
■■// is barrel
■■if ( parent.IsKindOf("Barrel_ColorBase") )
■■{
■■■is_barrel = true;
■■}

■■// is opened barrel■■■■
■■if ( is_barrel && parent.GetAnimationPhase("Lid") == 1 )
■■{
■■■is_opened_barrel = true;
■■}

■■// all of the barrel slots are empty
■■for ( int i = 0; i < SLOTS_ARRAY ; i++ )
■■{
■■■if ( parent.FindAttachmentBySlotName(slot_names[i]) != NULL )
■■■{
■■■■slot_test = false;
■■■■break;
■■■}
■■}
■■
■■if ( ( is_opened_barrel && slot_test ) || !is_barrel )
■■{
■■■return true;
■■}
■■return false;
■}

■override bool CanDetachAttachment( EntityAI parent )
■{
■■
■■bool is_barrel = false;
■■bool is_opened_barrel = false;
■■
■■// is barrel
■■if ( parent.IsKindOf("Barrel_ColorBase") )
■■{
■■■is_barrel = true;
■■}

■■// is opened barrel■■■■
■■if ( is_barrel && parent.GetAnimationPhase("Lid") == 1 )
■■{
■■■is_opened_barrel = true;
■■}
```

```
// ---------------------------------------------
// ------ GuyFawkesMask.c - START --------------------
// ---------------------------------------------


class GuyFawkesMask extends ClothingBase
{
■override bool CanPutAsAttachment( EntityAI parent )
■{
■■if(!super.CanPutAsAttachment(parent)) {return false;}
■■bool headgear_present = false;
■■
■■if ( parent.FindAttachmentBySlotName( "Headgear" ) )
■■{
■■■headgear_present = parent.FindAttachmentBySlotName( "Headgear" ).ConfigGetBool( "noMask" );
■■}
■■
■■if ( ( GetNumberOfItems() == 0 || !parent || parent.IsMan() ) && !headgear_present )
■■{
■■■return true;
■■}
■■return false;
■}
}


// ---------------------------------------------
// ------ GuyFawkesMask.c - END ----------------------
// ---------------------------------------------


// ---------------------------------------------
// ------ Hacksaw.c - START --------------------
// ---------------------------------------------


class Hacksaw extends ToolBase
{
■override void SetActions()
■{
■■super.SetActions();
■■
■■AddAction(ActionUnrestrainTarget);
■■AddAction(ActionMineBush);
■■AddAction(ActionSawPlanks);
■■AddAction(ActionDismantlePart);
■■//AddAction(ActionDestroyCombinationLock);
■■//AddAction(ActionDestroyPart);
■■AddAction(ActionSkinning);
■■AddAction(ActionMineTreeBark);
■}
}


// ---------------------------------------------
// ------ Hacksaw.c - END ----------------------
// ---------------------------------------------
// ---------------------------------------------
// ------ Hammer.c - START --------------------
// ---------------------------------------------


class Hammer extends Inventory_Base
{
■override void SetActions()
■{
■■super.SetActions();
■■
■■AddAction(ActionRepairPart);
■■AddAction(ActionMineRock1H);
■■AddAction(ActionBuildPart);
■}
}


// ---------------------------------------------
// ------ Hammer.c - END ----------------------
// ---------------------------------------------


// ---------------------------------------------
// ------ HandAnimatedForceSwapping.c - START --------------------
// ---------------------------------------------


// When editing this, take a look at HandAnimatedMoveToDst_W4T_Basic as well
// They can not be inherited from each other because of different inheritance
class HandAnimatedMoveToDst_W4T_Basic extends HandStateBase
{
■ref InventoryLocation m_Dst;

■override void OnEntry(HandEventBase e)
■{
■■Man player = e.m_Player;
■■if (m_Dst && m_Dst.IsValid())
■■{
■■■EntityAI item = m_Dst.GetItem();
■■■InventoryLocation src = new InventoryLocation;
■■■if (item.GetInventory().GetCurrentInventoryLocation(src))
■■■{
■■■■if (GameInventory.LocationCanMoveEntity(src, m_Dst))
■■■■{
■■■■■GameInventory.LocationSyncMoveEntity(src, m_Dst);
■■■■■player.OnItemInHandsChanged();
■■■■}
■■■■else
■■■■{
■■■■■#ifdef DEVELOPER
■■■■■if ( LogManager.IsInventoryHFSMLogEnable() )
■■■■■{■
■■■■■■Debug.InventoryHFSMLog("[hndfsm] HandAnimatedMoveToDst_W4T_Basic - not allowed");
■■■■■}
■■■■■#endif
■■■■}
■■■}
■■■else
■■■■Error("[hndfsm] " + Object.GetDebugName(e.m_Player) + " STS = " + e.m_Player.GetSimulationTi
■■}
■■else
■■■Error("[hndfsm] HandAnimatedMoveToDst_W4T_Basic - event has no valid m_Dst");
```

```
// ---------------------------------------------
// ------ HandAnimatedMovingToAtt.c - START --------------------
// ---------------------------------------------


class HandStartHidingAnimated extends HandStartAction
{
■void HandStartHidingAnimated(Man player = null, HandStateBase parent = null, WeaponActions action =
■{ }

■override void OnEntry(HandEventBase e)
■{
■■super.OnEntry(e);
■}

■override void OnAbort(HandEventBase e)
■{
■■super.OnAbort(e);
■}

■override void OnExit(HandEventBase e)
■{
■■super.OnExit(e);
■}
■
■override bool IsWaitingForActionFinish() { return m_ActionType == -1; }
};

// When editing this, take a look at HandAnimatedMoveToDst_W4T_Basic as well
// They can not be inherited from each other because of different inheritance
class HandAnimatedMoveToDst_W4T extends HandStartAction
{
■ref InventoryLocation m_Dst;

■override void OnEntry(HandEventBase e)
■{
■■Man player = e.m_Player;
■■if (m_Dst && m_Dst.IsValid())
■■{
■■■EntityAI item = m_Dst.GetItem();
■■■InventoryLocation src = new InventoryLocation;
■■■if (item.GetInventory().GetCurrentInventoryLocation(src))
■■■{
■■■■if (GameInventory.LocationCanMoveEntity(src, m_Dst))
■■■■{
■■■■■GameInventory.LocationSyncMoveEntity(src, m_Dst);
■■■■■player.OnItemInHandsChanged();
■■■■}
■■■■else
■■■■{
■■■■■#ifdef DEVELOPER
■■■■■if ( LogManager.IsInventoryHFSMLogEnable() )
■■■■■{■
■■■■■■Debug.InventoryHFSMLog("[hndfsm] HandAnimatedMoveToDst_W4T - not allowed");
■■■■■}
■■■■■#endif
■■■■}
■■■}
■■■else
■■■■Error("[hndfsm] " + Object.GetDebugName(e.m_Player) + " STS = " + e.m_Player.GetSimulationTi
■■}
■■else
■■■Error("[hndfsm] HandAnimatedMoveToDst_W4T - event has no valid m_Dst");
```

```
// ---------------------------------------------
// ------ HandAnimatedSwapping.c - START --------------------
// ---------------------------------------------


class HandSwappingAnimated_Show extends HandForceSwappingAnimated_Show
{
};


class HandAnimatedSwapping extends HandStateBase
{
■ref InventoryLocation m_Src1 = null;
■ref InventoryLocation m_Src2 = null;
■ref InventoryLocation m_Dst1 = null;
■ref InventoryLocation m_Dst2 = null;

■ref HandStartHidingAnimated m_Hide;
■ref HandSwappingAnimated_Show m_Show;

■void HandAnimatedSwapping(Man player = null, HandStateBase parent = null)
■{
■■// setup nested state machine
■■m_Hide = new HandStartHidingAnimated(player, this, WeaponActions.HIDE, -1);
■■m_Show = new HandSwappingAnimated_Show(player, this, WeaponActions.SHOW, -1);

■■// events:
■■HandEventBase _fin_ = new HandEventHumanCommandActionFinished;
■■HandEventBase _AEh_ = new HandAnimEventChanged;

■■m_FSM = new HandFSM(this); // @NOTE: set owner of the submachine fsm

■■m_FSM.AddTransition(new HandTransition(   m_Hide, _AEh_,   m_Show ));
■■m_FSM.AddTransition(new HandTransition(   m_Show, _fin_,     null ));

■■m_FSM.SetInitialState(m_Hide);
■}

■override void OnEntry(HandEventBase e)
■{
■■HandEventSwap es = HandEventSwap.Cast(e);
■■if (es)
■■{
■■■m_Src1 = es.GetSrc();
■■■m_Src2 = es.m_Src2;
■■■m_Dst1 = es.GetDst();
■■■m_Dst2 = es.m_Dst2;

■■■m_Show.m_Src1 = m_Src1;
■■■m_Show.m_Src2 = m_Src2;
■■■m_Show.m_Dst1 = m_Dst1;
■■■m_Show.m_Dst2 = m_Dst2;

■■■m_Hide.m_ActionType = es.m_AnimationID;
■■■m_Show.m_ActionType = es.m_Animation2ID;
■■■
■■■if (!GetGame().IsDedicatedServer())
■■■{
■■■■e.m_Player.GetHumanInventory().AddInventoryReservationEx(m_Dst2.GetItem(), m_Dst2, GameI
■■■■e.m_Player.GetHumanInventory().AddInventoryReservationEx(m_Dst1.GetItem(), m_Dst1, GameI
■■■}
■■}
```

```
// ---------------------------------------------
// ------ HandAnimatedTakingFromAtt.c - START --------------------
// ---------------------------------------------


class HandTakingAnimated_Hide extends HandStartAction
{ };

class HandTakingAnimated_Show extends HandStartAction
{
■ref InventoryLocation m_Src;
■ref InventoryLocation m_Dst;

■void HandTakingAnimated_Show(Man player = null, HandStateBase parent = null, WeaponActions actio
■{
■■m_Src = null;
■■m_Dst = null;
■}

■override void OnEntry(HandEventBase e)
■{
■■super.OnEntry(e);

■■if (m_Src)
■■{
■■■if (m_Src.IsValid())
■■■{
■■■■#ifdef DEVELOPER
■■■■if ( LogManager.IsInventoryHFSMLogEnable() )
■■■■{■
■■■■■Debug.InventoryHFSMLog("Action - STS = " + e.m_Player.GetSimulationTimeStamp(), e.ToString
■■■■}
■■■■#endif
■■■■
■■■■//if (GameInventory.LocationCanMoveEntity(m_Src, m_Dst))
■■■■//{
■■■■■GameInventory.LocationSyncMoveEntity(m_Src, m_Dst);
■■■■■e.m_Player.OnItemInHandsChanged();
■■■■//}
■■■■//else
■■■■//{
■■■■//■hndDebugPrint("[hndfsm] HandTakingAnimated_Show - not allowed");
■■■■//}
■■■}
■■■else
■■■{
■■■■Error("[hndfsm] " + Object.GetDebugName(e.m_Player) + " STS = " + e.m_Player.GetSimulationTi
■■■}
■■}
■■else
■■■Error("[hndfsm] HandTakingAnimated_Show, error - m_Src not configured");
■}

■override void OnAbort(HandEventBase e)
■{
■■m_Src = null;
■■m_Dst = null;
■■super.OnAbort(e);
■}

■override void OnExit(HandEventBase e)
■{
■■m_Src = null;
```

```
// ---------------------------------------------
// ------ HandAnimated_Guards.c - START --------------------
// ---------------------------------------------


int SlotToAnimType(notnull Man player, notnull InventoryLocation src, InventoryLocation dst = null)
{
	//Print("src.GetType() " + src.GetType());
	InventoryLocation invloc1 = new InventoryLocation;
	//InventoryLocation invloc2 = new InventoryLocation;
	
	if ( (dst && dst.GetParent() && !dst.GetParent().GetHierarchyRootPlayer()) || (src && src.GetParent() &&
		return -1;
	
	if (dst && (dst.GetType() == InventoryLocationType.ATTACHMENT || dst.GetType() == InventoryLocatio
	{
		invloc1.Copy(dst);
		//invloc2.Copy(src);
	}
	else if (src.GetType() == InventoryLocationType.ATTACHMENT || src.GetType() == InventoryLocationTy
	{
		invloc1.Copy(src);
		//invloc2.Copy(dst);
	}
	else
	{
		return -1;
	}
	
	int val = -1;
	if ( invloc1.GetItem() && invloc1.GetItem().GetInventoryHandAnimation(invloc1,val) )
	{
		return val;
	}
	
	if (invloc1.GetType() == InventoryLocationType.ATTACHMENT /*|| src.GetType() == InventoryLocationT
	{
		//return WeaponHideShowTypes.HIDESHOW_SLOT_KNIFEBACK;
		switch (invloc1.GetSlot())
		{
			case InventorySlots.SHOULDER:
			{
				if (invloc1.GetItem() && invloc1.GetItem().IsWeapon())
				{
					return WeaponHideShowTypes.HIDESHOW_SLOT_RFLLEFTBACK;
				}
				else if (invloc1.GetItem() && invloc1.GetItem().IsOneHandedBehaviour())
				{
					return WeaponHideShowTypes.HIDESHOW_SLOT_1HDLEFTBACK;
				}
				return WeaponHideShowTypes.HIDESHOW_SLOT_2HDLEFTBACK;
			}
			case InventorySlots.MELEE:
			{
				if (invloc1.GetItem() && invloc1.GetItem().IsWeapon())
				{
					return WeaponHideShowTypes.HIDESHOW_SLOT_RFLRIGHTBACK;
				}
				else if (invloc1.GetItem() && invloc1.GetItem().IsOneHandedBehaviour())
				{
					return WeaponHideShowTypes.HIDESHOW_SLOT_1HDRIGHTBACK;
				}
				return WeaponHideShowTypes.HIDESHOW_SLOT_2HDRIGHTBACK;
```

```
// ---------------------------------------------
// ------ HandcuffKeys.c - START --------------------
// ---------------------------------------------


class HandcuffKeys: Inventory_Base
{
■override void SetActions()
■{
■■super.SetActions();
■■
■■AddAction(ActionUnrestrainTarget);
■}
};



// ---------------------------------------------
// ------ HandcuffKeys.c - END ----------------------
// ---------------------------------------------


// ---------------------------------------------
// ------ Handcuffs.c - START --------------------
// ---------------------------------------------


class Handcuffs: Inventory_Base
{
■override void SetActions()
■{
■■super.SetActions();
■■
■■AddAction(ActionRestrainTarget);
■■AddAction(ActionRestrainSelf);
■}
};



// ---------------------------------------------
// ------ Handcuffs.c - END ----------------------
// ---------------------------------------------


// ---------------------------------------------
// ------ HandcuffsLocked.c - START --------------------
// ---------------------------------------------


class RestrainingToolLocked extends ItemBase
{
■void ~RestrainingToolLocked()
■{
■■PlayerBase player = PlayerBase.Cast(GetHierarchyRootPlayer());
■■if ( player && player.IsRestrained() )
■■{
■■■player.SetRestrained(false);
■■}
■}
■
■override void EEItemLocationChanged(notnull InventoryLocation oldLoc, notnull InventoryLocation newL
■{
```

```
// ---------------------------------------------
// ------ HandDrillKit.c - START --------------------
// ---------------------------------------------


class HandDrillKit extends ItemBase
{
■//! Override this method and and check if the given item can be ignited. It is not necesarry to check here i
■override bool CanIgniteItem( EntityAI ignite_target = NULL )
■{
■■return true;
■}
■
■override void OnIgnitedTarget( EntityAI ignited_item )
■{
■■if ( GetGame().IsServer() )
■■{
■■■DecreaseHealth( 20 );
■■}
■}■
■
■override void OnIgnitedTargetFailed( EntityAI target_item )
■{
■■if ( GetGame().IsServer() )
■■{
■■■DecreaseHealth( 20 );
■■}
■}
■
■override void SetActions()
■{
■■super.SetActions();
■■
■■AddAction(ActionLightItemOnFire);
■}
■
■/*
■override bool IsTargetIgnitionSuccessful( EntityAI item_target = NULL )
■{
■}
■*/■
}


// ---------------------------------------------
// ------ HandDrillKit.c - END ----------------------
// ---------------------------------------------


// ---------------------------------------------
// ------ HandFSM.c - START --------------------
// ---------------------------------------------


void hndDebugPrint (string s)
{
#ifdef INV_DEBUG
■PrintToRPT("" + s); // comment/uncomment to hide/see debug logs
#else
■//Print("" + s); // comment/uncomment to hide/see debug logs
#endif
}
void hndDebugSpam (string s)
```

```
// ---------------------------------------------
// ------ HandReplacingItemElsewhereWithNewInHands.c - START --------------------
// ---------------------------------------------


class HandStartReplacingItemElsewhereWithNewInHands extends HandStateBase
{
■void HandStartReplacingItemElsewhereWithNewInHands (Man player = NULL, HandStateBase parent =
■{ }

■override void OnEntry (HandEventBase e)
■{
■■super.OnEntry(e);
■■
■■Man player = e.m_Player;
■■HandEventReplaceWithNewBase edr = HandEventReplaceWithNewBase.Cast(e);
■■if (edr)
■■{
■■■hndDebugPrint("[hndfsm] HandStartReplacingItemElsewhereWithNewInHands about to execute lamb

■■■HumanInventoryWithFSM inv = HumanInventoryWithFSM.Cast(player.GetInventory());
■■■edr.m_Lambda.Execute(inv);
■■■return;
■■}
■■else
■■■Error("[hndfsm] HandStartReplacingItemElsewhereWithNewInHands - not a HandEvenReplaceWithN
■}

■override void OnAbort (HandEventBase e)
■{
■■super.OnAbort(e);
■}

■override void OnExit (HandEventBase e)
■{
■■super.OnExit(e);
■}

■override bool IsWaitingForActionFinish () { return true; }
};


class HandReplacingItemElsewhereWithNewInHands extends HandStateBase
{
■ref HandStartReplacingItemElsewhereWithNewInHands m_Replacing;

■void HandReplacingItemElsewhereWithNewInHands (Man player = NULL, HandStateBase parent = NUL
■{
■■// setup nested state machine
■■m_Replacing = new HandStartReplacingItemElsewhereWithNewInHands(player, this);

■■// events:
■■HandEventBase _fin_ = new HandEventHumanCommandActionFinished;

■■m_FSM = new HandFSM(this); // @NOTE: set owner of the submachine fsm

■■m_FSM.AddTransition(new HandTransition(   m_Replacing, _fin_,     NULL ));

■■m_FSM.SetInitialState(m_Replacing);
■}
};
```

```
// ---------------------------------------------
// ------ HandReplacingItemInHands.c - START --------------------
// ---------------------------------------------


class HandStartReplacingItemInHands extends HandStateBase
{
	void HandStartReplacingItemInHands (Man player = NULL, HandStateBase parent = NULL)
	{ }

	override void OnEntry (HandEventBase e)
	{
		super.OnEntry(e);
		
		Man player = e.m_Player;
		EntityAI itemInHands = player.GetHumanInventory().GetEntityInHands();

		InventoryLocation src = new InventoryLocation;
		if (itemInHands.GetInventory().GetCurrentInventoryLocation(src))
		{
			HandEventDestroyAndReplaceWithNew edr = HandEventDestroyAndReplaceWithNew.Cast(e);
			if (edr)
			{
				hndDebugPrint("[hndfsm] HandStartReplacingItemInHands about to execute lambda");

				HumanInventoryWithFSM inv = HumanInventoryWithFSM.Cast(player.GetInventory());
				edr.m_Lambda.Execute(inv);
				//player.GetItemAccessor().OnItemInHandsChanged();
				return;
			}
			else
				Error("[hndfsm] HandStartReplacingItemInHands - not a HandEventDestroyAndReplaceWithNew e
		}
		else
			Error("[hndfsm] HandStartReplacingItemInHands - itemInHands has no InventoryLocation");
	}

	override void OnAbort (HandEventBase e)
	{
		super.OnAbort(e);
	}

	override void OnExit (HandEventBase e)
	{
		super.OnExit(e);
	}

	override bool IsWaitingForActionFinish () { return true; }
};


class HandReplacingItemInHands extends HandStateBase
{
	ref HandStartReplacingItemInHands m_Replacing;

	void HandReplacingItemInHands (Man player = NULL, HandStateBase parent = NULL)
	{
		// setup nested state machine
		m_Replacing = new HandStartReplacingItemInHands(player, this);

		// events:
		HandEventBase _fin_ = new HandEventHumanCommandActionFinished;
```

```
// ---------------------------------------------
// ------ Hands.c - START --------------------
// ---------------------------------------------


class MaleHands_Base extends InventoryItem {};
class FemaleHands_Base extends InventoryItem {};

class MaleAdamHands extends MaleHands_Base {};
class MaleBorisHands extends MaleHands_Base {};
class MaleCyrilHands extends MaleHands_Base {};
class MaleDenisHands extends MaleHands_Base {};
class MaleEliasHands extends MaleHands_Base {};
class MaleFrancisHands extends MaleHands_Base {};
class MaleGuoHands extends MaleHands_Base {};
class MaleHassanHands extends MaleHands_Base {};
class MaleIndarHands extends MaleHands_Base {};
class MaleJoseHands extends MaleHands_Base {};
class MaleKaitoHands extends MaleHands_Base {};
class MaleLewisHands extends MaleHands_Base {};
class MaleManuaHands extends MaleHands_Base {};
class MaleNikiHands extends MaleHands_Base {};
class MaleOliverHands extends MaleHands_Base {};
class MalePeterHands extends MaleHands_Base {};
class MaleQuinnHands extends MaleHands_Base {};
class MaleRolfHands extends MaleHands_Base {};
class MaleSethHands extends MaleHands_Base {};
class MaleTaikiHands extends MaleHands_Base {};
class MaleDeanHands extends MaleHands_Base {};

class FemaleEvaHands extends FemaleHands_Base {};
class FemaleFridaHands extends FemaleHands_Base {};
class FemaleGabiHands extends FemaleHands_Base {};
class FemaleHelgaHands extends FemaleHands_Base {};
class FemaleIrenaHands extends FemaleHands_Base {};
class FemaleJudyHands extends FemaleHands_Base {};
class FemaleKeikoHands extends FemaleHands_Base {};
class FemaleLindaHands extends FemaleHands_Base {};
class FemaleMariaHands extends FemaleHands_Base {};
class FemaleNaomiHands extends FemaleHands_Base {};


// ---------------------------------------------
// ------ Hands.c - END ----------------------
// ---------------------------------------------



// ---------------------------------------------
// ------ HandsArea.c - START --------------------
// ---------------------------------------------


class HandsArea: Container
{
■protected ScrollWidget■■■m_Scroller;
■protected ref HandsContainer■m_HandsContainer;
■protected ref SizeToChild■■m_HandsResizer;
■
■protected bool■■■■■m_ShouldChangeSize = true;

■void HandsArea( LayoutHolder parent )
■{
■■m_HandsContainer = new HandsContainer( this );
```

```
// ---------------------------------------------
// ------ HandSaw.c - START --------------------
// ---------------------------------------------


class HandSaw: Inventory_Base
{
■override void SetActions()
■{
■■super.SetActions();
■■
■■AddAction(ActionUnrestrainTarget);
■■AddAction(ActionMineBush);
■■AddAction(ActionSawPlanks);
■■AddAction(ActionDismantlePart);
■■//AddAction(ActionDestroyCombinationLock);
■■//AddAction(ActionDestroyPart);
■■AddAction(ActionSkinning);
■■AddAction(ActionMineTreeBark);
■}
};




// ---------------------------------------------
// ------ HandSaw.c - END ----------------------
// ---------------------------------------------


// ---------------------------------------------
// ------ HandsContainer.c - START --------------------
// ---------------------------------------------


class HandsContainer: Container
{
■protected bool■■■■■■■■■■■■m_Hidden;
■protected ref HandsHeader■■■■■■■■■m_CollapsibleHeader;
■protected ref HandsPreview■■■■■■■■■m_HandsPreview;
■
■protected ref Attachments■■■■■■■■■m_Atts;
■protected ref CargoContainer■■■■■■■■m_CargoGrid;
■
■protected ref map<EntityAI, ref CargoContainer>■■■m_AttachmentCargos;
■protected ref map<EntityAI, AttachmentsWrapper>■■■m_AttachmentAttachmentsContainers;
■protected ref map<EntityAI, ref Attachments>■■■m_AttachmentAttachments;
■protected ref array<int>■■■■■■■■■m_AttachmentSlotsSorted;
■
■protected int■■■■■■■■■■■■m_StaticAttCount = 0;
■protected int■■■■■■■■■■■■m_StaticCargoCount = 0;
■
■protected ScrollWidget■■■■■■■■■■m_ScrollWidget;

■void HandsContainer( LayoutHolder parent )
■{
■■m_AttachmentCargos■■■■■= new map<EntityAI, ref CargoContainer>;
■■m_AttachmentAttachmentsContainers■= new map<EntityAI, AttachmentsWrapper>;
■■m_AttachmentAttachments■■■■= new map<EntityAI, ref Attachments>;
■■
■■m_CollapsibleHeader = new HandsHeader( this, "CollapseButtonOnMouseButtonDown" );
■■GetMainWidget().SetFlags( WidgetFlags.IGNOREPOINTER );
■■m_MainWidget = m_MainWidget.FindWidget( "body" );
■■GetMainWidget().SetFlags( WidgetFlags.IGNOREPOINTER );
```

```
// ---------------------------------------------
// ------ HandsCover_improvised.c - START --------------------
// ---------------------------------------------


class HandsCover_Improvised extends Clothing
{
■override void SetActions()
■{
■■super.SetActions();
■■AddAction(ActionWringClothes);
■}
};




// ---------------------------------------------
// ------ HandsCover_improvised.c - END ----------------------
// ---------------------------------------------


// ---------------------------------------------
// ------ HandsHeader.c - START --------------------
// ---------------------------------------------


class HandsHeader: Header
{
■protected bool■■m_ForceHideCollapseButtons;
■protected Widget■m_ItemHeader;
■
■void HandsHeader( LayoutHolder parent, string function_name )
■{
■■m_DefaultFontSize■= 24;
■■m_ItemHeader■■= m_MainWidget.FindAnyWidget( "hands_item_header" );
■■
■■m_MainWidget■■= GetMainWidget().FindAnyWidget( "height_wrapper" );
■■m_DefaultColor■■= GetMainWidget().GetColor();
■■#ifdef PLATFORM_CONSOLE
■■ShowCollapseButtons( false, true );
■■#endif
■}
■
■override void SetLayoutName()
■{
■■m_LayoutName = WidgetLayoutName.HandsHeader;
■}
■
■override void UpdateInterval()
■{
■■PlayerBase p = PlayerBase.Cast( GetGame().GetPlayer() );
■■if (!p)
■■■return;
■■
■■EntityAI item = p.GetHumanInventory().GetEntityInHands();
■■if ( !m_ForceHideCollapseButtons )
■■{
■■■if ( item && !item.GetInventory().IsInventoryLockedForLockType( HIDE_INV_FROM_SCRIPT ) && ite
■■■{
■■■■ShowCollapseButtons( true );
■■■}
■■■else
■■■{
```

```
// ---------------------------------------------
// ------ HandShivers.c - START --------------------
// ---------------------------------------------



class HandShiversSymptom extends SymptomBase
{
■int m_ShakeLevel;
■//this is just for the Symptom parameters set-up and is called even if the Symptom doesn't execute, don't
■override void OnInit()
■{
■■m_SymptomType = SymptomTypes.SECONDARY;
■■m_Priority = 0;
■■m_ID = SymptomIDs.SYMPTOM_HAND_SHIVER;
■■m_SyncToClient = false;

■}
■
■override void SetParam(Param p)
■{
■■Param1<int> p1 = Param1<int>.Cast(p);
■■if ( p1 )
■■{
■■■m_ShakeLevel = p1.param1;
■■■
■■■if ( m_ShakeLevel > PlayerBase.SHAKE_LEVEL_MAX )
■■■{
■■■■m_ShakeLevel = PlayerBase.SHAKE_LEVEL_MAX;
■■■}
■■}
■}

■
■override void OnUpdateClient(PlayerBase player, float deltatime)
■{
■}

■
■override void OnGetActivatedServer(PlayerBase player)
■{
■■player.SetShakesForced(m_ShakeLevel);
■}
■
■override void OnGetDeactivatedServer(PlayerBase player)
■{
■■player.SetShakesForced(0);
■}

■override void OnGetActivatedClient(PlayerBase player)
■{

■■
■}

■//!only gets called once on an active Symptom that is being deactivated
■override void OnGetDeactivatedClient(PlayerBase player)
■{

■}
■
}
```

```
// ---------------------------------------------
// ------ HandsPreview.c - START --------------------
// ---------------------------------------------


class HandsPreview: Container
{
■protected ref Icon■m_Icon;
■protected EntityAI■m_AttachmentsInitialized;
■
■protected float■■m_IconSize;

■protected ItemBase■m_Item

■void HandsPreview( LayoutHolder parent )
■{
■■GetGame().GetPlayer().GetOnItemAddedToHands().Insert( CreateNewIcon );
■■GetGame().GetPlayer().GetOnItemRemovedFromHands().Insert( DeleteIcon );
■}
■
■void ~HandsPreview()
■{
■■GetGame().GetPlayer().GetOnItemAddedToHands().Remove( CreateNewIcon );
■■GetGame().GetPlayer().GetOnItemRemovedFromHands().Remove( DeleteIcon );
■}

■void RefreshQuantity( EntityAI m_Item_to_refresh )
■{
■■if ( m_Icon )
■■{
■■■m_Icon.SetQuantity();
■■}
■}
■
■override EntityAI GetFocusedItem()
■{
■■return m_Item;
■}

■Icon GetIcon()
■{
■■return m_Icon;
■}
■
■override void SetDefaultFocus(bool while_micromanagment_mode = false)
■{
■■super.SetDefaultFocus(while_micromanagment_mode);
■■if ( m_Icon )
■■■m_Icon.SetActive(true);
■}
■
■override void SetLastFocus()
■{
■■super.SetLastFocus();
■■if ( m_Icon )
■■■m_Icon.SetActive(true);
■}
■
■override void Unfocus()
■{
■■super.Unfocus();
■■if ( m_Icon )
■■■m_Icon.SetActive(false);
```

```
// ---------------------------------------------
// ------ HandStableState.c - START --------------------
// ---------------------------------------------


/**@class■■HandStableState
 * @brief■  represents stable state (i.e. the basic states that the fsm will spend the most time in)
 **/
class HandStableState extends HandStateBase
{
■int m_AnimState;

■void HandStableState (Man player = NULL, HandStateBase parent = NULL, int anim_state = -1) { m_Ani

■void SyncAnimState () { }

■override void OnEntry (HandEventBase e)
■{
■■super.OnEntry(e);
■■SyncAnimState();

■■//m_weapon.OnStableStateEntry();
■}
■override void OnUpdate (float dt)
■{
■■super.OnUpdate(dt);
■■SyncAnimState();
■}
■override void OnAbort (HandEventBase e)
■{
■■super.OnAbort(e);
■}
■override void OnExit (HandEventBase e)
■{
■■//m_weapon.ResetWeaponAnimState();
■■super.OnExit(e);
■}

■override bool IsIdle () { return true; }

■int GetCurrentStateID () { return 0; }

■/// query for entity in hands
■bool HasEntityInHands () { return false; }
};




// ---------------------------------------------
// ------ HandStableState.c - END ----------------------
// ---------------------------------------------


// ---------------------------------------------
// ------ HandStartAction.c - START --------------------
// ---------------------------------------------


/**@class ■HandStartAction
 * @brief■simple class starting animation action specified by m_action and m_actionType
 */
class HandStartAction extends HandStateBase
```

```
// ---------------------------------------------
// ------ HandStateBase.c - START --------------------
// ---------------------------------------------


/**@class■HandStateBase
 * @brief■represent hand state base
 *
 * Class comes with entry/update/exit hooks that can be overriden in custom states
 **/
class HandStateBase
{
■Man m_Player; /// entity that this state relates to
■HandStateBase m_parentState; /// hierarchical parent state of this state (or null)
■ref HandFSM m_FSM; /// nested state machine (or null)
■void HandStateBase (Man player = NULL, HandStateBase parent = NULL) { m_Player = player; m_parer

■/**@fn■■SetParentState
■ * @brief■allows construction of hierarchical state machine
■ **/
■void SetParentState (HandStateBase parent) { m_parentState = parent; }
■/**@fn■■GetParentState
■ * @return■state that owns this sub-state (or null if plain state)
■ **/
■HandStateBase GetParentState () { return m_parentState; }

■bool HasFSM () { return m_FSM != NULL; }
■HandFSM GetFSM () { return m_FSM; }

■bool ProcessEvent (HandEventBase e)
■{
■■if (HasFSM())
■■■return m_FSM.ProcessEvent(e);
■■return false;
■}

■/**@fn■■AddTransition
■ * @brief■adds transition into m_FSM transition table
■ **/
■void AddTransition (HandTransition t)
■{
■■if (HasFSM())
■■■m_FSM.AddTransition(t);
■■else
■■■Error("[hndfsm] adding transition to state without FSM. Configure FSM first.");
■}


■/**@fn■■OnEntry
■ * @brief■called upon entry to state
■ * @NOTE■if state has (non-running) sub-machine, it's started on entry
■ * @param[in] e■the event that triggered transition to this state
■ **/
■void OnEntry (HandEventBase e)
■{
■■if (HasFSM() && !m_FSM.IsRunning())
■■{
■■■if (e)
■■■■hndDebugPrint("[hndfsm] { " + Object.GetDebugName(e.m_Player) + " STS = " + e.m_Player.GetSi
■■■else
■■■hndDebugPrint("[hndfsm] { " + this.Type().ToString() + "  Has Sub-FSM! Starting submachine...");
■■■m_FSM.Start(e);
■■}
```

```
// ---------------------------------------------
// ------ Hand_Actions.c - START --------------------
// ---------------------------------------------


///@{ actions

/**@class■HandActionBase
 * @brief■represents action executed on transition just between OnExit from old state and OnEntry to new
 **/
class HandActionBase
{
■/**@fn■■■Action
■ * @brief■■executed when transition occurs
■ **/
■void Action (HandEventBase e) { }
};


class HandActionCreated extends HandActionBase
{
■override void Action (HandEventBase e)
■{
■■#ifdef DEVELOPER
■■if ( LogManager.IsInventoryHFSMLogEnable() )
■■{■
■■■Debug.InventoryHFSMLog("Action - STS = " + e.m_Player.GetSimulationTimeStamp(), e.ToString() ,
■■}
■■#endif

■■e.m_Player.OnItemInHandsChanged();
■}
};

class HandActionTake extends HandActionBase
{
■override void Action (HandEventBase e)
■{
■■#ifdef DEVELOPER
■■if ( LogManager.IsInventoryHFSMLogEnable() )
■■{■
■■■Debug.InventoryHFSMLog("Action - STS = " + e.m_Player.GetSimulationTimeStamp(), e.ToString() ,
■■}
■■#endif

■■GameInventory.LocationSyncMoveEntity(e.GetSrc(), e.GetDst());
■■e.m_Player.OnItemInHandsChanged();
■}
};

class HandActionDrop extends HandActionBase
{
■override void Action (HandEventBase e)
■{
■■#ifdef DEVELOPER
■■if ( LogManager.IsInventoryHFSMLogEnable() )
■■{■
■■■Debug.InventoryHFSMLog("Action - STS = " + e.m_Player.GetSimulationTimeStamp(), e.ToString() ,
■■}
■■#endif

■■GameInventory.LocationSyncMoveEntity(e.GetSrc(), e.GetDst());
 ■■e.m_Player.OnItemInHandsChanged();
```

```
// ---------------------------------------------
// ------ Hand_Events.c - START --------------------
// ---------------------------------------------


///@{ events

/**@enum■HandEventID
 * @brief■identifier for events mainly for rpc purposes
 **/
enum HandEventID
{
■UNKNOWN,
■TAKE,
■MOVETO,
■DROP,
■THROW,
■SWAP,
■FORCESWAP,
■DESTROY,
■CREATED,
■DESTROYED,
■REPLACE,
■REPLACE2,
■REPLACE3,
■REPLACED,
■HUMANCOMMAND_ACTION_FINISHED,
■HUMANCOMMAND_ACTION_ABORTED,
■ANIMEVENT_CHANGE_HIDE
};

enum JunctureRequestResult
{
■JUNCTURE_NOT_REQUIRED,
■JUNCTURE_ACQUIRED,
■JUNCTURE_DENIED,
■ERROR,
}

/**@class■HandEventBase
 * @brief■represents event that triggers transition from state to state
 **/
class HandEventBase
{
■int m_EventID = 0;
■int m_AnimationID = -1;
■Man m_Player;
■ref InventoryLocation m_Src;

■void HandEventBase (Man p = null, InventoryLocation src = null) { m_Player = p; m_Src = src; }
■HandEventID GetEventID () { return m_EventID; }

■void ReadFromContext (ParamsReadContext ctx) { } // actual read is in CreateHandEventFromContext

■void WriteToContext (ParamsWriteContext ctx)
■{
■■ctx.Write(m_EventID);
■■ctx.Write(m_Player);
■■OptionalLocationWriteToContext(m_Src, ctx);
■■ctx.Write(m_AnimationID);
■}

■InventoryLocation GetSrc () { return m_Src; }
```

```
// ---------------------------------------------
// ------ Hand_Guards.c - START --------------------
// ---------------------------------------------


///@{ guards

/**@class■HandGuardBase
 * @brief■represents guard on a transition from state to state
 **/
class HandGuardBase
{
■/**@fn■■■GuardCondition
■ * @brief■■enable or disable transition based on condition
■ * the guard is a boolean operation executed first and which can prevent the transition from firing by retur
■ * @return■■true if transition is allowed
■ **/
■bool GuardCondition(HandEventBase e) { return true; }
};


class HandGuardAnd extends HandGuardBase
{
■ref HandGuardBase m_arg0;
■ref HandGuardBase m_arg1;

■void HandGuardAnd(HandGuardBase arg0 = null, HandGuardBase arg1 = null) { m_arg0 = arg0; m_arg

■override bool GuardCondition(HandEventBase e)
■{
■■bool result = m_arg0.GuardCondition(e) && m_arg1.GuardCondition(e);
■■
■■#ifdef DEVELOPER
■■if ( LogManager.IsInventoryHFSMLogEnable() )
■■{■
■■■Debug.InventoryHFSMLog("GuardCondition result: " + result + " - " + m_arg0.Type() + " && " + m_arg
■■}
■■#endif
■■
■■return result;
■}
};

class HandGuardNot extends HandGuardBase
{
■ref HandGuardBase m_arg0;

■void HandGuardNot(HandGuardBase arg0 = null) { m_arg0 = arg0; }

■override bool GuardCondition(HandEventBase e)
■{
■■bool result = !m_arg0.GuardCondition(e);
■■
■■#ifdef DEVELOPER
■■if ( LogManager.IsInventoryHFSMLogEnable() )
■■{■
■■■Debug.InventoryHFSMLog("GuardCondition result: " + result + " - " + m_arg0.Type(), "HandGuardNo
■■}
■■#endif
■■return result;
■}
};
```

```
// ---------------------------------------------
// ------ Hand_States.c - START --------------------
// ---------------------------------------------


///@{ states

enum HandStateID
{
■UNKNOWN■■■■=  0,
■Empty■■■■=  1,
■Equipped■■■=  2,
}

class HandStateEmpty : HandStableState
{
■override void OnEntry(HandEventBase e)
■ {
■■if(e)
■■{
■■■switch (e.m_EventID)
■■■{
■■■■case HandEventID.MOVETO:
■■■■■if (HumanInventory.HasInventoryReservation(e.GetSrcEntity(), e.GetDst()))
■■■■■■HumanInventory.ClearInventoryReservation(e.GetSrcEntity(), e.GetDst());
■■■■■break;
■■■■
■■■■default: {};
■■■}
■■}
■■super.OnEntry(e);
■}
■override void OnExit(HandEventBase e) { super.OnExit(e); }
■override int GetCurrentStateID() { return HandStateID.Empty; }
};

class HandStateEquipped : HandStableState
{
■override void OnEntry(HandEventBase e)
■{
■■if(e)
■■{
■■■switch (e.m_EventID)
■■■{
■■■■case HandEventID.MOVETO:
■■■■■if (HumanInventory.HasInventoryReservation(e.GetSrcEntity(), e.GetDst()))
■■■■■■HumanInventory.ClearInventoryReservation(e.GetSrcEntity(), e.GetDst());
■■■■■break;
■■■■
■■■■default: {};
■■■}
■■}
■■super.OnEntry(e);
■}
■override void OnExit(HandEventBase e)
■{
■■/*switch (e.m_EventID)
■■{
■■■case HandEventID.MOVETO:
■■■■if (HumanInventory.HasInventoryReservation(e.GetSrcEntity(), e.GetDst()))
■■■■■HumanInventory.ClearInventoryReservation(e.GetSrcEntity(), e.GetDst());
■■■■break;
■■■■
```

```
// ---------------------------------------------
// ------ Hatchback_02.c - START --------------------
// ---------------------------------------------


class Hatchback_02 extends CarScript
{
■protected ref UniversalTemperatureSource m_UTSource;
■protected ref UniversalTemperatureSourceSettings m_UTSSettings;
■protected ref UniversalTemperatureSourceLambdaEngine m_UTSLEngine;

■void Hatchback_02()
■{
■■//m_dmgContactCoef■■= 0.070;

■■m_EngineStartOK■■■= "Hatchback_02_engine_start_SoundSet";
■■m_EngineStartBattery■= "Hatchback_02_engine_failed_start_battery_SoundSet";
■■m_EngineStartPlug■■= "Hatchback_02_engine_failed_start_sparkplugs_SoundSet";
■■m_EngineStartFuel■■= "Hatchback_02_engine_failed_start_fuel_SoundSet";
■■m_EngineStopFuel■■= "offroad_engine_stop_fuel_SoundSet";
■■
■■m_CarDoorOpenSound■■= "offroad_door_open_SoundSet";
■■m_CarDoorCloseSound■■= "offroad_door_close_SoundSet";
■■
■■m_CarHornShortSoundName = "Hatchback_02_Horn_Short_SoundSet";
■■m_CarHornLongSoundName■= "Hatchback_02_Horn_SoundSet";
■■
■■SetEnginePos("0 0.7 1.4");
■}

■override void EEInit()
■{■■
■■super.EEInit();
■■
■■if (GetGame().IsServer() || !GetGame().IsMultiplayer())
■■{
 ■■■m_UTSSettings ■■■■■= new UniversalTemperatureSourceSettings();
■■■m_UTSSettings.m_ManualUpdate ■= true;
■■■m_UTSSettings.m_TemperatureMin■= 0;
■■■m_UTSSettings.m_TemperatureMax■= 30;
■■■m_UTSSettings.m_RangeFull■■= 0.5;
■■■m_UTSSettings.m_RangeMax■■= 2;
■■■m_UTSSettings.m_TemperatureCap■= 25;
■■■
■■■m_UTSLEngine■■■■■= new UniversalTemperatureSourceLambdaEngine();
■■■m_UTSource■■■■■■= new UniversalTemperatureSource(this, m_UTSSettings, m_UTSLEngine);
■■}■■
■}
■
■override void OnEngineStart()
■{
■■super.OnEngineStart();

■■if (GetGame().IsServer() || !GetGame().IsMultiplayer())
■■{
■■■m_UTSource.SetDefferedActive(true, 20.0);
■■}
■}
■
■override void OnEngineStop()
■{
■■super.OnEngineStop();
```

```
// ---------------------------------------------
// ------ Hatchback_02FrontLight.c - START --------------------
// ---------------------------------------------


class Hatchback_02FrontLight extends CarLightBase
{
■void Hatchback_02FrontLight()
■{
■■m_SegregatedBrightness = 6;
■■m_SegregatedRadius = 65;
■■m_SegregatedAngle = 80;
■■m_SegregatedColorRGB = Vector(0.8, 0.8, 1);
■■
■■m_AggregatedBrightness = 12;
■■m_AggregatedRadius = 90;
■■m_AggregatedAngle = 90;
■■m_AggregatedColorRGB = Vector(0.8, 0.8, 1);
■■
■■FadeIn(0.3);
■■SetFadeOutTime(0.25);
■■
■■SegregateLight();
■}
}


// ---------------------------------------------
// ------ Hatchback_02FrontLight.c - END ----------------------
// ---------------------------------------------


// ---------------------------------------------
// ------ Hatchback_02RearLight.c - START --------------------
// ---------------------------------------------


class Hatchback_02RearLight extends CarRearLightBase
{
■void Hatchback_02RearLight()
■{
■■// Brake light only
■■m_SegregatedBrakeBrightness = 1;
■■m_SegregatedBrakeRadius = 6;
■■m_SegregatedBrakeAngle = 270;
■■m_SegregatedBrakeColorRGB = Vector(1, 0.05, 0.05);
■■
■■// Reverse light only
■■m_SegregatedBrightness = 2;
■■m_SegregatedRadius = 13;
■■m_SegregatedAngle = 180;
■■m_SegregatedColorRGB = Vector(1.0, 1.0, 1.0);
■■
■■// Brake & Revese lights combined
■■m_AggregatedBrightness = 2.5;
■■m_AggregatedRadius = 15;
■■m_AggregatedAngle = 180;
■■m_AggregatedColorRGB = Vector(1.0, 0.5, 0.5);
■■
■■FadeIn(0.1);
■■SetFadeOutTime(0.1);
■■SetVisibleDuringDaylight(false);
■■SetCastShadow(false);
```

```
// ---------------------------------------------
// ------ Hatchet.c - START --------------------
// ---------------------------------------------


class Hatchet extends ToolBase
{
■override void SetActions()
■{
■■super.SetActions();

■■AddAction(ActionMineTree);
■■AddAction(ActionMineTreeBark);
■■AddAction(ActionMineBush);
■■AddAction(ActionRepairPart);
■■AddAction(ActionDismantlePart);
■■AddAction(ActionBuildPart);
■■//AddAction(ActionDestroyPart);
■■//AddAction(ActionSawPlanks);
■■AddAction(ActionUnrestrainTarget);
■■AddAction(ActionSkinning);
■}
}


// ---------------------------------------------
// ------ Hatchet.c - END ----------------------
// ---------------------------------------------


// ---------------------------------------------
// ------ HayHook.c - START --------------------
// ---------------------------------------------


class HayHook: Inventory_Base
{
■override void SetActions()
■{
■■super.SetActions();
■■
■■AddAction(ActionUnrestrainTarget);
■■AddAction(ActionSkinning);
■}
};



// ---------------------------------------------
// ------ HayHook.c - END ----------------------
// ---------------------------------------------


// ---------------------------------------------
// ------ Head.c - START --------------------
// ---------------------------------------------


class Head : ItemBase
{
};
```

```
// ---------------------------------------------
// ------ HeadBandana_ColorBase.c - START --------------------
// ---------------------------------------------


/*
class HeadBandana_ColorBase extends Clothing {};
class HeadBandana_Green extends HeadBandana_ColorBase {};
class HeadBandana_Black extends HeadBandana_ColorBase {};
class HeadBandana_Blue extends HeadBandana_ColorBase {};
class HeadBandana_Red extends HeadBandana_ColorBase {};
class HeadBandana_Yellow extends HeadBandana_ColorBase {};
*/


// ---------------------------------------------
// ------ HeadBandana_ColorBase.c - END ----------------------
// ---------------------------------------------


// ---------------------------------------------
// ------ HeadCover_improvised.c - START --------------------
// ---------------------------------------------


class HeadCover_Improvised extends Clothing
{
■override void SetActions()
■{
■■super.SetActions();
■■AddAction(ActionWringClothes);
■}
};



// ---------------------------------------------
// ------ HeadCover_improvised.c - END ----------------------
// ---------------------------------------------


// ---------------------------------------------
// ------ Header.c - START --------------------
// ---------------------------------------------


class Header: LayoutHolder
{
■protected int■■■m_DefaultColor;
■protected int■■■m_DefaultFontSize;
■
■protected EntityAI■■m_Entity;
■
■protected Widget■■m_CollapseButton;
■protected Widget■■m_UpArrowButton;
■protected Widget■■m_DownArrowButton;
■protected TextWidget■m_HeaderText;
■
■void Header( LayoutHolder parent, string function_name )
■{
■■m_CollapseButton■= GetMainWidget().FindAnyWidget( "collapse_button" );
■■m_UpArrowButton■■= GetMainWidget().FindAnyWidget( "opened" );
■■m_DownArrowButton■= GetMainWidget().FindAnyWidget( "closed" );
```

```
// ---------------------------------------------
// ------ HeadtorchLight.c - START --------------------
// ---------------------------------------------


class HeadtorchLight extends SpotLightBase
{
■private static float m_DefaultBrightness = 3;
■private static float m_DefaultRadius = 20;
■
■void HeadtorchLight()
■{
■■SetVisibleDuringDaylight( true );
■■SetRadiusTo( m_DefaultRadius );
■■SetSpotLightAngle( 95 );
■■SetCastShadow( true );
■■FadeIn( 0.06 );
■■SetBrightnessTo( m_DefaultBrightness );
■■SetAmbientColor( 0.92, 0.85, 0.58 );
■■SetDiffuseColor( 0.92, 0.85, 0.58 );
■■SetFadeOutTime( 0.1 );
■■//SetDisableShadowsWithinRadius(0.25); // Idea for optimization: Uncomment this to disable shadows
■}
■
■void SetColorToWhite()
■{
■■SetAmbientColor( 0.92, 0.85, 0.86 );
■■SetDiffuseColor( 0.92, 0.85, 0.86 );
■}
■
■void SetColorToRed()
■{
■■SetAmbientColor( 1.0, 0.2, 0.2 );
■■SetDiffuseColor( 1.0, 0.2, 0.2 );
■}
■
■void SetIntensity( float coef, float time )
■{
■■FadeBrightnessTo( m_DefaultBrightness * coef, time );
■■FadeRadiusTo( m_DefaultRadius * coef, time );
■}
}


// ---------------------------------------------
// ------ HeadtorchLight.c - END ----------------------
// ---------------------------------------------


// ---------------------------------------------
// ------ Headtorch_Black.c - START --------------------
// ---------------------------------------------


class Headtorch_Black : Headtorch_ColorBase
{
■override void OnLightCreated()
■{
■■m_Light.SetColorToRed();
■}

■override void OnDebugSpawn()
■{
```

```
// ---------------------------------------------
// ------ Headtorch_ColorBase.c - START --------------------
// ---------------------------------------------


class Headtorch_ColorBase extends Clothing
{
■HeadtorchLight m_Light;
■
■static int■■REFLECTOR_ID = 4;
■static int■■GLASS_ID = 5;
■
■static string ■LIGHT_OFF_GLASS ■■= "dz\\characters\\headgear\\data\\HeadTorchGlass.rvmat";
■static string ■LIGHT_OFF_REFLECTOR ■= "dz\\characters\\headgear\\data\\HeadTorch.rvmat";
■static string ■LIGHT_ON_GLASS ■■■= "dz\\characters\\headgear\\data\\HeadTorchGlass_on.rvmat";
■static string ■LIGHT_ON_GLASS_RED■■= "dz\\characters\\headgear\\data\\HeadTorchGlass_on_red.r
■static string ■LIGHT_ON_REFLECTOR ■■= "dz\\characters\\headgear\\data\\HeadTorch_ON.rvmat";
■static string ■LIGHT_ON_REFLECTOR_RED■= "dz\\characters\\headgear\\data\\HeadTorch_ON_red.r
■
■static vector m_OnHeadLocalPos = Vector(0.12,0.15,0);
■static vector m_OnHeadLocalOri = Vector(0,90,0);
■static string m_OffHeadLightPoint = "beamStart";
■static string m_OffHeadLightTarget = "beamEnd";
■
■ref Timer m_Timer;
■
■void Headtorch_ColorBase()
■{
■■if (!m_Timer)
■■■m_Timer = new Timer(CALL_CATEGORY_SYSTEM);
■■■
■■m_Timer.Run(1 , this, "CheckParent", NULL, false);
■}
■
■override bool CanPutAsAttachment(EntityAI parent)
■{
■■if (!super.CanPutAsAttachment(parent))
■■■return false;
■■
■■Clothing helmet = Clothing.Cast(parent.FindAttachmentBySlotName("Headgear"));
■■if (helmet && helmet.ConfigGetBool("noNVStrap"))
■■{
■■■return false;
■■}
■■
■■Clothing mask = Clothing.Cast(parent.FindAttachmentBySlotName("Mask"));
■■if (mask && mask.ConfigGetBool("noEyewear"))
■■{
■■■return false;
■■}
■■
■■return true;
■}
■
■void CheckParent()
■{
■■EntityAI owner = GetHierarchyParent();
■■
■■// Spawn a battery in the headtorch if it's attached on a zombie and switch it on
■■if (owner  &&  owner.IsZombie())
■■{
■■■GetInventory().CreateAttachment("Battery9V");
■■■GetCompEM().SwitchOn();
```

```
// ---------------------------------------------
// ------ Headtorch_Grey.c - START --------------------
// ---------------------------------------------


class Headtorch_Grey : Headtorch_ColorBase
{
■override void OnDebugSpawn()
■{
■■Battery9V.Cast(GetInventory().CreateInInventory("Battery9V"));
■}
};


// ---------------------------------------------
// ------ Headtorch_Grey.c - END ----------------------
// ---------------------------------------------


// ---------------------------------------------
// ------ Health.c - START --------------------
// ---------------------------------------------


class HealthMdfr: ModifierBase
{
■private float■m_LastHealthLevel;
■private float■m_LastBloodLevel;
■
■
■override void Init()
■{
■■m_TrackActivatedTime = false;
■■m_ID ■■■■■= eModifiers.MDF_HEALTH;
■■m_TickIntervalInactive ■= DEFAULT_TICK_TIME_INACTIVE;
■■m_TickIntervalActive ■= DEFAULT_TICK_TIME_ACTIVE;
■■DisableDeactivateCheck();
■}
■
■override bool ActivateCondition(PlayerBase player)
■{
■■return true;
■}


■override bool DeactivateCondition(PlayerBase player)
■{
■■return false;
■}

■override void OnTick(PlayerBase player, float deltaT)
■{■
■■
■■float blood =  player.GetHealth("GlobalHealth", "Blood");
■■float health = player.GetHealth("GlobalHealth", "Health");■

■■float healthdelta  = Math.AbsInt(health - m_LastHealthLevel);
■■if (health <  m_LastHealthLevel) healthdelta = -healthdelta;
■■//if( player.m_NotifiersManager ) player.m_NotifiersManager.FindNotifier(eNotifiers.NTF_LIVES).Displa
■■m_LastHealthLevel = health;
■■
■■float blooddelta  = Math.AbsInt(blood - m_LastBloodLevel);
■■if (blood <  m_LastBloodLevel) blooddelta = -blooddelta;
```

```
// ---------------------------------------------
// ------ HealthNotfr.c - START --------------------
// ---------------------------------------------


class HealthNotfr: NotifierBase
{
■/*
■private const float■ ■HEALTHY_BLOOD_TRESHOLD■= 5000;
■private const float■ ■HEALTHY_TRESHOLD■■= 5000;
■private const float■ ■HEALING_ENERGY_TRESHOLD■= 4000;
■private const float■ ■HEALING_WATER_TRESHOLD■= 2500;
■private const float■ ■HEALING_BLOOD_TRESHOLD■= 5000;
■*/
■private const float ■DEC_TRESHOLD_LOW ■■■= 0;
■private const float ■DEC_TRESHOLD_MED ■■■= -0.7;
■private const float ■DEC_TRESHOLD_HIGH■■■= -1.3;
■private const float ■INC_TRESHOLD_LOW ■■■= 0;
■private const float ■INC_TRESHOLD_MED ■■■= 0.7;
■private const float ■INC_TRESHOLD_HIGH■■■= 1.3;
■
■■
■void HealthNotfr(NotifiersManager manager)
■{
■■m_TickInterval = 3000;
■■m_TendencyBufferSize = 6;
■}

■override int GetNotifierType()
■{
■■return eNotifiers.NTF_HEALTHY;
■}

■override void DisplayBadge()
■{
■}
■
■override void DisplayTendency(float delta)
■{
■■int tendency = CalculateTendency(delta, INC_TRESHOLD_LOW, INC_TRESHOLD_MED, INC_TRES
■■//PrintString("tendency:" + tendency);
■■//GetVirtualHud().SetStatus(eDisplayElements.DELM_TDCY_HEALTH,tendency);
■■
■■//DSLevels level = DetermineLevel( GetObservedValue(), PlayerConstants.THRESHOLD_HEALTH_W
■■
■■EStatLevels health_level = m_Player.GetStatLevelHealth();
■■DisplayElementTendency dis_elm = DisplayElementTendency.Cast(GetVirtualHud().GetElement(eDisp
■■
■■if( dis_elm )
■■{
■■■dis_elm.SetSeriousnessLevel(health_level);
■■■dis_elm.SetTendency(tendency);
■■}
■}

■override void HideBadge()
■{
■■
■■////GetVirtualHud().SetStatus(eDisplayElements.DELM_NTFR_HEALTHY,DELM_LVL_0);
■}

■override protected float GetObservedValue()
■{
```

```
// ---------------------------------------------
// ------ HealthRegen.c - START --------------------
// ---------------------------------------------


class HealthRegenMdfr: ModifierBase
{
■override void Init()
■{
■■m_TrackActivatedTime = false;
■■m_ID ■■■■■= eModifiers.MDF_HEALTH_REGEN;
■■m_TickIntervalInactive ■= DEFAULT_TICK_TIME_INACTIVE;
■■m_TickIntervalActive ■= DEFAULT_TICK_TIME_ACTIVE;
■■DisableDeactivateCheck();
■}
■
■override bool ActivateCondition(PlayerBase player)
■{
■■return true;
■}

■override void OnActivate(PlayerBase player)
■{
■}

■override void OnReconnect(PlayerBase player)
■{

■}
■
■override bool DeactivateCondition(PlayerBase player)
■{
■■return false;
■}

■override void OnTick(PlayerBase player, float deltaT)
■{■
■■if (player.IsAlive())//this needs to be here, some of the other modifiers can kill the character during the s
■■{

■■■float regen_speed = player.GetHealthRegenSpeed() * deltaT;;
■■■player.AddHealth("GlobalHealth", "Health" ,  regen_speed );
■■■
■■■player.AddHealth("RightArm","Health",regen_speed * PlayerConstants.DAMAGE_ZONE_BLOOD_R
■■■player.AddHealth("RightHand","Health",regen_speed * PlayerConstants.DAMAGE_ZONE_BLOOD_
■■■player.AddHealth("LeftArm","Health",regen_speed * PlayerConstants.DAMAGE_ZONE_BLOOD_RE
■■■player.AddHealth("LeftHand","Health",regen_speed * PlayerConstants.DAMAGE_ZONE_BLOOD_R
■■■
■■■//Leg regen when legs are NOT BROKEN
■■■if ( player.GetBrokenLegs() == eBrokenLegs.NO_BROKEN_LEGS )
■■■{
■■■■player.AddHealth("RightLeg","Health", PlayerConstants.LEG_HEALTH_REGEN);
■■■■player.AddHealth("RightFoot","Health", PlayerConstants.LEG_HEALTH_REGEN);
■■■■player.AddHealth("LeftLeg","Health", PlayerConstants.LEG_HEALTH_REGEN);
■■■■player.AddHealth("LeftFoot","Health", PlayerConstants.LEG_HEALTH_REGEN);
■■■}
■■■else
■■■{
■■■■//Leg regen when legs are BROKEN or SPLINTED
■■■■player.AddHealth("RightLeg","Health", PlayerConstants.LEG_HEALTH_REGEN_BROKEN);
■■■■player.AddHealth("RightFoot","Health",PlayerConstants.LEG_HEALTH_REGEN_BROKEN);
■■■■player.AddHealth("LeftLeg","Health", PlayerConstants.LEG_HEALTH_REGEN_BROKEN);
■■■■player.AddHealth("LeftFoot","Health", PlayerConstants.LEG_HEALTH_REGEN_BROKEN);
```

```
// ---------------------------------------------
// ------ HeartAttack.c - START --------------------
// ---------------------------------------------


class HeartAttackMdfr: ModifierBase
{
■private const float ■HEALTH_DECREMENT_PER_SEC = -0.2;
■private const float ■SHOCK_DECREMENT_PER_SEC = -2;
■private const float ■SHOCK_LIMIT = 0;
■■
■override void Init()
■{
■■m_TrackActivatedTime = false;
■■m_ID ■■■■■= eModifiers.MDF_HEART_ATTACK;
■■m_TickIntervalInactive ■= DEFAULT_TICK_TIME_INACTIVE;
■■m_TickIntervalActive ■= DEFAULT_TICK_TIME_ACTIVE;
■■m_IsPersistent = true;
■■DisableDeactivateCheck();
■■DisableActivateCheck();
■}
■override bool ActivateCondition(PlayerBase player)
■{
■■return false;
■}

■override void OnActivate(PlayerBase player)
■{
■}

■override bool DeactivateCondition(PlayerBase player)
■{
■■return false;
■}

■override void OnTick(PlayerBase player, float deltaT)
■{■
■■if ( player.GetHealth("GlobalHealth","Shock") <= SHOCK_LIMIT )
■■{
■■■float currenthealth = player.GetHealth("GlobalHealth", "Health");
■■■player.AddHealth("GlobalHealth", "Health" , HEALTH_DECREMENT_PER_SEC * deltaT );
■■}
■■else
■■{
■■■float currentshock =  player.GetHealth("GlobalHealth", "Shock");
■■■player.AddHealth("GlobalHealth", "Shock", SHOCK_DECREMENT_PER_SEC * deltaT);
■■}
■}
};


// ---------------------------------------------
// ------ HeartAttack.c - END ---------------------
// ---------------------------------------------


// ---------------------------------------------
// ------ HeartbeatNotfr.c - START --------------------
// ---------------------------------------------


class HeartbeatNotfr: NotifierBase
{
```

```
// ---------------------------------------------
// ------ HeatBuffer.c - START --------------------
// ---------------------------------------------


class HeatBufferMdfr: ModifierBase
{
■protected const float HEATBUFFER_SHOW = 25.0;
■protected const float HEATBUFFER_HIDE = 20.0;

■override void Init()
■{
■■m_TrackActivatedTime = false;
■■m_ID ■■■■■= eModifiers.MDF_HEATBUFFER;
■■m_TickIntervalInactive ■= 1;
■■m_TickIntervalActive ■= DEFAULT_TICK_TIME_ACTIVE;
■}
■
■override void OnTick(PlayerBase player, float deltaT)
■{

■}
■
■override bool ActivateCondition(PlayerBase player)
■{
■■float heatbuffer = player.GetStatHeatBuffer().Get();
■■if ( heatbuffer >= HEATBUFFER_SHOW )
■■{
■■■return true;
■■}
■■else
■■{
■■■return false;
■■}
■}

■override void OnActivate(PlayerBase player)
■{
■■if( player.GetNotifiersManager() )
■■{
■■■//player.GetNotifiersManager().ActivateByType(eNotifiers.NTF_HEATBUFFER);
■■■player.ToggleHeatBufferVisibility( true );
■■}
■}

■override void OnDeactivate(PlayerBase player)
■{
■■if( player.GetNotifiersManager() )
■■{
■■■//player.GetNotifiersManager().DeactivateByType(eNotifiers.NTF_HEATBUFFER);
■■■player.ToggleHeatBufferVisibility( false );
■■}
■}

■override bool DeactivateCondition(PlayerBase player)
■{
■■float heatbuffer = player.GetStatHeatBuffer().Get();
■■if ( heatbuffer < HEATBUFFER_HIDE )
■■{
■■■return true;
■■}
■■else
■■{
```

```
// ---------------------------------------------
// ------ HeatComfortAnimHandler.c - START --------------------
// ---------------------------------------------


class HeatComfortAnimHandler
{
■const float TICK_INTERVAL  = 2;
■float m_TimeSinceLastTick;
■float m_ProcessTimeAccuFreeze;
■float m_ProcessTimeAccuHot;
■
■PlayerBase m_Player;
■float m_EventTimeFreeze = -1;
■float m_EventTimeHot = -1;
■protected ref HumanMovementState m_MovementState = new HumanMovementState();
■
■const float TIME_INTERVAL_HC_MINUS_LOW_MIN = 5; const float TIME_INTERVAL_HC_MINUS_LO
■const float TIME_INTERVAL_HC_MINUS_HIGH_MIN = 15; const float TIME_INTERVAL_HC_MINUS_H
■
■const float TIME_INTERVAL_HC_PLUS_LOW_MIN = 5; const float TIME_INTERVAL_HC_PLUS_LOW_
■const float TIME_INTERVAL_HC_PLUS_HIGH_MIN = 15; const float TIME_INTERVAL_HC_PLUS_HIG
■
■void HeatComfortAnimHandler(PlayerBase player)
■{
■■m_Player = player;
■}

■void Update(float delta_time, HumanMovementState hms)
■{
■■m_TimeSinceLastTick += delta_time;

■■if( m_TimeSinceLastTick > TICK_INTERVAL )
■■{
■■■Process(m_TimeSinceLastTick);
■■■m_TimeSinceLastTick = 0;
■■}
■}
■
■float GetEventTime(float hc_value ,float threshold_low, float threshold_high, float low_min, float high_mir
■{
■■float inv_value = Math.InverseLerp(threshold_low, threshold_high, hc_value);
■■float value_min = Math.Lerp(low_min, high_min,inv_value);
■■float value_max = Math.Lerp(low_max,high_max,inv_value);

■■return Math.RandomFloatInclusive(value_min,value_max);
■}
■
■
■void Process(float delta_time)
■{
■■if( GetGame().IsServer() )
■■{
■■■
■■■float hc = m_Player.GetStatHeatComfort().Get();
■■■float inv_value;
■■■float value_min;
■■■float value_max;
■■■float offset_time;
■■■
■■■if ( hc <= PlayerConstants.THRESHOLD_HEAT_COMFORT_MINUS_WARNING )
■■■{
■■■■m_ProcessTimeAccuFreeze++;
```

```
// ---------------------------------------------
// ------ HeatComfortEvents.c - START --------------------
// ---------------------------------------------


class HeatComfortEventsBase extends PlayerSoundEventBase
{
■
■void HeatComfortEventsBase()
■{
■■m_HasPriorityOverTypes = EPlayerSoundEventType.DUMMY | EPlayerSoundEventType.INJURY | EP
■■m_Type = EPlayerSoundEventType.GENERAL;
■}
■
■override bool HasPriorityOverCurrent(PlayerBase player, EPlayerSoundEventID other_state_id,EPlayerS
■{
■■return true;
■}
}

class FreezingSoundEvent extends HeatComfortEventsBase
{
■void FreezingSoundEvent()
■{
■■m_ID = EPlayerSoundEventID.FREEZING;
■■m_SoundVoiceAnimEventClassID = 11;
■}
}

class HotSoundEvent extends HeatComfortEventsBase
{
■void HotSoundEvent()
■{
■■m_ID = EPlayerSoundEventID.HOT;
■■m_SoundVoiceAnimEventClassID = 1111111;
■}
}


// ---------------------------------------------
// ------ HeatComfortEvents.c - END ----------------------
// ---------------------------------------------


// ---------------------------------------------
// ------ HeatComfortMdfr.c - START --------------------
// ---------------------------------------------


class HeatComfortMdfr: ModifierBase
{
■override void Init()
■{
■■m_TrackActivatedTime ■= false;
■■m_ID ■■■■■= eModifiers.MDF_TEMPERATURE;
■■m_TickIntervalInactive ■= DEFAULT_TICK_TIME_INACTIVE;
■■m_TickIntervalActive ■= DEFAULT_TICK_TIME_ACTIVE;
■■DisableDeactivateCheck();
■}

■override bool ActivateCondition(PlayerBase player)
■{
■■return true;
```

```
// ---------------------------------------------
// ------ Heatpack.c - START --------------------
// ---------------------------------------------


class Heatpack : ItemBase
{
■override bool CanHaveTemperature()
■{
■■return true;
■}
■
■override void OnWorkStart()
■{
■■if (GetGame().IsServer())
■■{
■■■SetTemperature(60);
■■}
■}
■
■override void OnWork( float consumed_energy )
■{
■■if (GetGame().IsServer())
■■{
■■■SetTemperature(60);
■■}
■}
■
■override void OnWorkStop()
■{
■■if (GetGame().IsServer())
■■{
■■■SetHealth(0);
■■}
■}
■
■override void SetActions()
■{
■■super.SetActions();
■■
■■AddAction(ActionTurnOnHeatpack);
■}
};



// ---------------------------------------------
// ------ Heatpack.c - END ----------------------
// ---------------------------------------------


// ---------------------------------------------
// ------ Helicopter.c - START --------------------
// ---------------------------------------------


/*!
■Base native class for helicopter.
*/
class Helicopter extends Transport
{
};
```

```
// ---------------------------------------------
// ------ HelicopterScript.c - START --------------------
// ---------------------------------------------


/*!
■Base script class for helicopters.
*/
class HelicopterScript extends HelicopterAuto
{
■void HelicopterScript()
■{
■■SetEventMask(EntityEvent.POSTSIMULATE);
■}

■override void EOnPostSimulate(IEntity other, float timeSlice)
■{
■}

■/*!
■■Gets called everytime the game wants to start the engine.
■■This callback is called on server only.

■■\return true if the engine can start, false otherwise.
■*/
■bool OnBeforeEngineStart()
■{
■■return true;
■}

■//! Gets called everytime the engine starts.
■void OnEngineStart()
■{
■}

■//! Gets called everytime the engine stops.
■void OnEngineStop()
■{
■}
};



// ---------------------------------------------
// ------ HelicopterScript.c - END ----------------------
// ---------------------------------------------


// ---------------------------------------------
// ------ HelmetBase.c - START --------------------
// ---------------------------------------------


class HelmetBase extends ClothingBase
{
■
};


// ---------------------------------------------
// ------ HelmetBase.c - END ----------------------
// ---------------------------------------------
// ---------------------------------------------
// ------ HelpScreen.c - START --------------------
// ---------------------------------------------


class HelpScreen extends UIScriptedMenu
{
■TextListboxWidget m_KeyBindingsTextListboxWidget;
■TextListboxWidget m_MouseBindingsTextListboxWidget;

■ButtonWidget m_CloseConsoleButton;

■void HelpScreen()
■{
■■
■}

■void ~HelpScreen()
■{■
■}

■override Widget Init()
■{
■■PluginKeyBinding module_keybinding = PluginKeyBinding.Cast( GetPlugin(PluginKeyBinding) );
■■
■■layoutRoot = GetGame().GetWorkspace().CreateWidgets("gui/layouts/help_screen.layout");
■■m_KeyBindingsTextListboxWidget = TextListboxWidget.Cast( layoutRoot.FindAnyWidget("KeyBindings
■■m_MouseBindingsTextListboxWidget = TextListboxWidget.Cast( layoutRoot.FindAnyWidget("MouseBi
■■
■■array<ref KeyBinding> keybindings = module_keybinding.GetKeyBindings();
■■for ( int i = 0; i < keybindings.Count(); i++ )
■■{
■■■m_KeyBindingsTextListboxWidget.AddItem( keybindings.Get(i).GetInfoBind(), NULL, 0 );
■■■m_KeyBindingsTextListboxWidget.SetItem( i, keybindings.Get(i).GetInfoDescription(), NULL, 1 );
■■}
■■
■■array<ref MouseBinding> mousebindings = module_keybinding.GetMouseBindings();
■■for ( i = 0; i < mousebindings.Count(); i++ )
■■{
■■■m_MouseBindingsTextListboxWidget.AddItem( mousebindings.Get(i).GetInfoBind(), NULL, 0 );
■■■m_MouseBindingsTextListboxWidget.SetItem( i, mousebindings.Get(i).GetInfoDescription(), NULL, 1
■■}
■■
■■m_CloseConsoleButton = ButtonWidget.Cast( layoutRoot.FindAnyWidget("CloseButtonWidget") );
■■
■■return layoutRoot;
■}

■override bool OnClick(Widget w, int x, int y, int button)
■{
■■super.OnClick(w, x, y, button);
■■
■■if ( w == m_CloseConsoleButton )
■■{
■■■Close();
■■■return true;
■■}
■■
■■return false;
■}
}
```

```
// ---------------------------------------------
// ------ HemolyticReaction.c - START --------------------
// ---------------------------------------------


class HemolyticReactionMdfr: ModifierBase
{
■private float■■m_RunningTime;
■
■override void Init()
■{
■■m_TrackActivatedTime = true;
■■m_ID ■■■■■= eModifiers.MDF_HEMOLYTIC_REACTION;
■■m_TickIntervalInactive ■= DEFAULT_TICK_TIME_INACTIVE;
■■m_TickIntervalActive ■= DEFAULT_TICK_TIME_ACTIVE;
■■m_IsPersistent = true;
■■m_RunningTime = CalculateRunTime();
■■
■■DisableActivateCheck();
■}
■override bool ActivateCondition(PlayerBase player)
■{
■■return false;
■}

■override void OnActivate(PlayerBase player)
■{
■■player.IncreaseDiseaseCount();
■}

■override void OnDeactivate(PlayerBase player)
■{
■■player.DecreaseDiseaseCount();
■}

■override void OnReconnect(PlayerBase player)
■{
■■OnActivate(player);
■}

■override bool DeactivateCondition(PlayerBase player)
■{
■■float attached_time = GetAttachedTime();
■■
■■if(attached_time > m_RunningTime )
■■{
■■■return true;■
■■}
■■else
■■{
■■■return false;
■■}
■}

■override void OnTick(PlayerBase player, float deltaT)
■{
■■player.AddHealth("","Blood",-PlayerConstants.HEMOLYTIC_BLOOD_DRAIN_PER_SEC * deltaT);
■}
■
■float CalculateRunTime()
■{
■■float time = PlayerConstants.HEMOLYTIC_BLOODLOSS_AMOUNT / PlayerConstants.SALINE_BLOO
■■return time;
```

```
// ----------------------------------------------
// ------ HescoBox.c - START --------------------
// ----------------------------------------------


class HescoBox extends Inventory_Base
{
■static const int FOLDED            = 0;
■static const int UNFOLDED ■■   = 1;
■static const int FILLED ■■   = 2;
■static const int PERCENTUAL_DAMAGE = 1;
■
■ref Timer ■■■■■■m_Timer;
■ref protected EffectSound ■■m_DeployLoopSound;

■protected int m_State;
■
■void HescoBox()
■{
■■m_State = FOLDED;

■■//synchronized variables
■■RegisterNetSyncVariableInt( "m_State", FOLDED, FILLED );
■■RegisterNetSyncVariableBool("m_IsSoundSynchRemote");
■■RegisterNetSyncVariableBool("m_IsDeploySound");
■}
■
■void ~HescoBox()
■{
■■SEffectManager.DestroyEffect( m_DeployLoopSound );
■}

■override bool HasProxyParts()
■{
■■return true;
■}
■
■override bool CanPutIntoHands( EntityAI parent )
■{
■■if( !super.CanPutIntoHands( parent ) )
■■{
■■■return false;
■■}
■■return CanBeManipulated();
■}
■■
■void Synchronize()
■{
■■SetSynchDirty();
■}

■override void OnVariablesSynchronized()
■{
■■super.OnVariablesSynchronized();
■■
■■//refresh visuals
■■RefreshVisuals();
■■■■
■■if ( IsDeploySound() )
■■{
■■■PlayDeploySound();
■■}
■■■■
```

```
// ---------------------------------------------
// ------ HFSMBase.c - START --------------------
// ---------------------------------------------


void fsmDebugPrint (string s)
{
■//Print("" + s); // comment/uncomment to hide/see debug logs
}
void fsmDebugSpam (string s)
{
■//Print("" + s); // comment/uncomment to hide/see debug spam
}


/**@class■■HFSMBase
 * @brief■■base class for hierarchic finite state machine
 *
 * stores current state (m_State) and transition table with possible transitions from each state
 * to another state via event
 *
 * each state can have nested state machine, thus creating hierarchy
 **/
class HFSMBase<Class FSMStateBase, Class FSMEventBase, Class FSMActionBase, Class FSMGuard
{
■protected ref FSMStateBase m_State; /// current fsm state
■protected FSMStateBase m_OwnerState; /// state that owns this fsm (or null if root)
■protected ref FSMStateBase m_InitialState; /// configurable initial state of the machine
■protected ref array<ref FSMTransition<FSMStateBase, FSMEventBase, FSMActionBase, FSMGuardBas
■protected bool m_HasCompletions = false;

■void HFSMBase (FSMStateBase ownerState = NULL)
■{
■■m_OwnerState = ownerState;
■}

■/**@fn■■GetCurrentState
■ * @return■returns currently active state within this machine (i.e. not hierarchic state)
■ **/
■FSMStateBase GetCurrentState ()
■{
■■return m_State;
■}
■/**@fn■■GetOwnerState
■ * @return■returns state that is owner of this fsm submachine. returns null if this is a root machine.
■ **/
■FSMStateBase GetOwnerState ()
■{
■■return m_OwnerState;
■}
■/**@fn■■GetHierarchyPath
■ * @brief■■returns hierarchic state (path to root) of a state
■ * @param[in] state \p state the path is starting
■ * @param[out] path  \p  current hierarchic state
■ * @return■true if current state returned in path argument, false otherwise
■ **/
■bool GetHierarchyPath (FSMStateBase state, out array<FSMStateBase> path)
■{
■■FSMStateBase curr = state;
■■while (curr)
■■{
■■  path.Insert(curr);
■■  curr = curr.GetParentState();
```

```
// ---------------------------------------------
// ------ HiddenSelectionsData.c - START --------------------
// ---------------------------------------------


class HiddenSelectionsData
{
■ref array<string> m_HiddenSelections = new array<string>;
■ref array<string> m_HiddenSelectionsTextures = new array<string>;
■ref array<string> m_HiddenSelectionsMaterials = new array<string>;
■
■ref map<string, int> m_HiddenSelectionNameToIndex = new map<string, int>;
■
■void HiddenSelectionsData(string type)
■{
■■GetGame().ConfigGetTextArray( string.Format("CfgVehicles %1 hiddenSelections", ■■■type), m_Hidd
■■GetGame().ConfigGetTextArray( string.Format("CfgVehicles %1 hiddenSelectionsTextures", ■type), m
■■GetGame().ConfigGetTextArray( string.Format("CfgVehicles %1 hiddenSelectionsMaterials", type), m_
■■
■■for (int i = 0; i < m_HiddenSelections.Count(); ++i)
■■■m_HiddenSelectionNameToIndex.Insert(m_HiddenSelections[i], i);
■}
■
■int GetHiddenSelectionIndex(string selection)
■{
■■int index;
■■if (m_HiddenSelectionNameToIndex.Find(selection, index))
■■■return index;
■■
■■return -1;
■}
■
■TStringArray GetHiddenSelectionsTextures()
■{
■■return m_HiddenSelectionsTextures;
■}
■
■TStringArray GetHiddenSelectionsMaterials()
■{
■■return m_HiddenSelectionsMaterials;
■}
■
■static array<string> GetHiddenSelectionsConfig(string type)
■{
■■array<string> hiddenSelections = new array<string>;
■■GetGame().ConfigGetTextArray( string.Format("CfgVehicles %1 hiddenSelections", type), hiddenSelec
■■return hiddenSelections;
■}
■
■static array<string> GetHiddenSelectionsTexturesConfig(string type)
■{
■■array<string> hiddenSelectionsTextures = new array<string>;
■■GetGame().ConfigGetTextArray( string.Format("CfgVehicles %1 hiddenSelectionsTextures", type), hidd
■■return hiddenSelectionsTextures;
■}
■
■static array<string> GetHiddenSelectionsMaterialsConfig(string type)
■{
■■array<string> hiddenSelectionsMaterials = new array<string>;
■■GetGame().ConfigGetTextArray( string.Format("CfgVehicles %1 hiddenSelectionsMaterials", type), hid
■■return hiddenSelectionsMaterials;
■}
}
```

```
// --------------------------------------------
// ------ HighCapacityVest_ColorBase.c - START --------------------
// --------------------------------------------


class HighCapacityVest_ColorBase extends Clothing {};
class HighCapacityVest_Black extends HighCapacityVest_ColorBase {};
class HighCapacityVest_Olive extends HighCapacityVest_ColorBase {};


// --------------------------------------------
// ------ HighCapacityVest_ColorBase.c - END ----------------------
// --------------------------------------------


// --------------------------------------------
// ------ HikingBootsLow_ColorBase.c - START --------------------
// --------------------------------------------


class HikingBootsLow_ColorBase extends Clothing {};
class HikingBootsLow_Black extends HikingBootsLow_ColorBase {};
class HikingBootsLow_Blue extends HikingBootsLow_ColorBase {};
class HikingBootsLow_Beige extends HikingBootsLow_ColorBase {};
class HikingBootsLow_Grey extends HikingBootsLow_ColorBase {};


// --------------------------------------------
// ------ HikingBootsLow_ColorBase.c - END ----------------------
// --------------------------------------------


// --------------------------------------------
// ------ HikingBoots_ColorBase.c - START --------------------
// --------------------------------------------


class HikingBoots_ColorBase extends Clothing {};
class HikingBoots_Brown extends HikingBoots_ColorBase {};
class HikingBoots_Black extends HikingBoots_ColorBase {};


// --------------------------------------------
// ------ HikingBoots_ColorBase.c - END ----------------------
// --------------------------------------------


// --------------------------------------------
// ------ HikingJacket_ColorBase.c - START --------------------
// --------------------------------------------


class HikingJacket_ColorBase extends Clothing
{
■override void SetActions()
■{
■■super.SetActions();
■■AddAction(ActionWringClothes);
■}
};
class HikingJacket_Black extends HikingJacket_ColorBase {};
class HikingJacket_Blue extends HikingJacket_ColorBase {};
class HikingJacket_Red extends HikingJacket_ColorBase {};
```

```
// ---------------------------------------------
// ------ HintPage.c - START --------------------
// ---------------------------------------------


/*
■Data model class for Hint json
■Location:
*/
class HintPage
{
■private string ■m_Headline;■■■// Headline hint text■
■private string ■m_Description;■■// Hint description text
■private string ■m_ImagePath;■■// Hint image, can be null

■string GetHeadlineText()
■{
■■return m_Headline;
■}
■string GetDescriptionText()
■{
■■return m_Description;
■}
■string GetImagePath()
■{
■■return m_ImagePath;
■}
}


// ---------------------------------------------
// ------ HintPage.c - END ----------------------
// ---------------------------------------------


// ---------------------------------------------
// ------ HitDirectionArrow.c - START --------------------
// ---------------------------------------------



class HitDirectionEffectArrow extends HitDirectionEffectBase
{
■override HitDirectionImagesBase GetImageData()
■{
■■typename type = HitDirectionImagesArrow;
■■HitDirectionImagesArrow data = HitDirectionImagesArrow.Cast(type.Spawn());
■■return data;
■}
■
■override void FinalizePositionCalculation()
■{
■■//blobs move around the edges, the rest on elypsis
■■float distance_x = (m_DistanceAdjust * m_SizeXEnf) + (m_SizeXEnf / 2.5);
■■float distance_y = (m_DistanceAdjust * m_SizeYEnf) + (m_SizeYEnf / 2.5);
■■m_PosX = Math.Sin(m_AngleRad) * distance_x;
■■m_PosY = -Math.Cos(m_AngleRad) * distance_y;
■}
■
■override void SetIndicatorRotation(float timeslice = -1.0)
■{
■■if (m_RotationOverride == HitDirectionConstants.ROTATION_DEFAULT)
■■{
```

```
// ---------------------------------------------
// ------ HitDirectionBase.c - START --------------------
// ---------------------------------------------




class HitDirectionEffectBase
{
■const float DURATION_COEF_MIN = 0.6;
■const float INTENSITY_MIN = 0.6;
■
■float m_HitDirection;
■float m_Duration;
■float m_BreakPoint;
■float m_TimeActive;
■float m_IntensityMax;
■
■Widget m_LayoutRoot;
■Widget m_Image;
■
■DayZPlayer m_Player;
■
■bool m_Initialized;
■int m_SizeXEnf;
■int m_SizeYEnf;
■float m_PosX;
■float m_PosY;
■float m_PosXScreenEdge;
■float m_PosYScreenEdge;
■float m_AngleRad;
■float m_AngleRadPrev;
■float m_SmoothVel[1];
■
■ref HitDirectionImagesBase m_ImageData;
■
■void HitDirectionEffectBase()
■{
■■m_Initialized = false;
■■m_PosX = 0.0;
■■m_PosY = 0.0;
■■m_AngleRad = 0.0;
■■m_SmoothVel[0] = 0.0;
■■
■■m_ImageData = GetImageData();
■■m_ImageData.GetCurrentImageData(m_LayoutRoot,m_Image);
■}
■
■//! Called manually after object spawn
■void Init(DayZPlayer player, float hit_direction, float intensity_max)
■{
■■m_Player = player;
■■float duration_coef = Math.Clamp(intensity_max,DURATION_COEF_MIN,1);
■■m_IntensityMax = Math.Clamp(intensity_max,INTENSITY_MIN,1);
■■m_Duration = m_DurationMax * duration_coef;
■■m_BreakPoint = Math.Clamp(m_BreakPointBase * m_Duration,0,m_Duration);
■■m_Scatter = Math.Clamp(m_Scatter,0.0,180.0);
■■m_HitDirection = hit_direction + (Math.RandomFloatInclusive(0,m_Scatter) * Math.Pow(-1.0,Math.Rand
■■
■■Widget w = m_LayoutRoot;
■■w = w.GetChildren();
■■while (w)
■■{
```

```
// ---------------------------------------------
// ------ HitDirectionImagesArrow.c - START --------------------
// ---------------------------------------------



class HitDirectionImagesArrow_Static
{
■static ref array<string> ■m_ImagePathsLayouts;
■static ref array<string> ■m_ImagePathsImages;
■
■static void InitArrays()
■{
■■if ( !m_ImagePathsLayouts && !m_ImagePathsImages )
■■{
■■■m_ImagePathsLayouts = new array<string>;
■■■m_ImagePathsImages = new array<string>;
■■}
■}
}

class HitDirectionImagesArrow extends HitDirectionImagesBase
{
■override protected void PerformRegisterImages()
■{
■■HitDirectionImagesArrow_Static.InitArrays();
■■
■■RegisterImage("gui/layouts/gameplay/HitDirectionIndication.layout","Single_indicator_Arrow_0");
■}
■
■override protected ref array<string> GetTypeLayoutPathArray()
■{
■■return HitDirectionImagesArrow_Static.m_ImagePathsLayouts;
■}
■
■override protected ref array<string> GetTypeImagePathArray()
■{
■■return HitDirectionImagesArrow_Static.m_ImagePathsImages;
■}
}


// ---------------------------------------------
// ------ HitDirectionImagesArrow.c - END ----------------------
// ---------------------------------------------


// ---------------------------------------------
// ------ HitDirectionImagesBase.c - START --------------------
// ---------------------------------------------



class HitDirectionImagesBase
{
■int ■■■■■■m_ImageIndex;
■
■void HitDirectionImagesBase()
■{
■■if ( (!GetTypeLayoutPathArray() && !GetTypeImagePathArray()) )
■■{
■■■PerformRegisterImages();
■■}
```

```
// ---------------------------------------------
// ------ HitDirectionImagesSpike.c - START --------------------
// ---------------------------------------------



class HitDirectionImagesSpike_Static
{
■static ref array<string> ■m_ImagePathsLayouts;
■static ref array<string> ■m_ImagePathsImages;
■
■static void InitArrays()
■{
■■if ( !m_ImagePathsLayouts && !m_ImagePathsImages )
■■{
■■■m_ImagePathsLayouts = new array<string>;
■■■m_ImagePathsImages = new array<string>;
■■}
■}
}

class HitDirectionImagesSpike extends HitDirectionImagesBase
{
■override protected void PerformRegisterImages()
■{
■■HitDirectionImagesSpike_Static.InitArrays();
■■
■■RegisterImage("gui/layouts/gameplay/HitDirectionIndication.layout","Single_indicator_Spike_0");
■}
■
■override protected ref array<string> GetTypeLayoutPathArray()
■{
■■return HitDirectionImagesSpike_Static.m_ImagePathsLayouts;
■}
■
■override protected ref array<string> GetTypeImagePathArray()
■{
■■return HitDirectionImagesSpike_Static.m_ImagePathsImages;
■}
}


// ---------------------------------------------
// ------ HitDirectionImagesSpike.c - END ----------------------
// ---------------------------------------------


// ---------------------------------------------
// ------ HitDirectionImagesSplash.c - START --------------------
// ---------------------------------------------



class HitDirectionImagesSplash_Static
{
■static ref array<string> ■m_ImagePathsLayouts;
■static ref array<string> ■m_ImagePathsImages;
■
■static void InitArrays()
■{
■■if ( !m_ImagePathsLayouts && !m_ImagePathsImages )
■■{
■■■m_ImagePathsLayouts = new array<string>;
```

```
// ---------------------------------------------
// ------ HitDirectionSpike.c - START --------------------
// ---------------------------------------------



class HitDirectionEffectSpike extends HitDirectionEffectBase
{
■override HitDirectionImagesBase GetImageData()
■{
■■typename type = HitDirectionImagesSpike;
■■HitDirectionImagesSpike data = HitDirectionImagesSpike.Cast(type.Spawn());
■■return data;
■}
■
■override void FinalizePositionCalculation()
■{
■■//blobs move around the edges, the rest on elypsis
■■float distance_x = (m_DistanceAdjust * m_SizeXEnf) + (m_SizeXEnf / 2.5);
■■float distance_y = (m_DistanceAdjust * m_SizeYEnf) + (m_SizeYEnf / 2.5);
■■m_PosX = Math.Sin(m_AngleRad) * distance_x;
■■m_PosY = -Math.Cos(m_AngleRad) * distance_y;
■}
■
■override void SetIndicatorRotation(float timeslice = -1.0)
■{
■■if (m_RotationOverride == HitDirectionConstants.ROTATION_DEFAULT)
■■{
■■■m_LayoutRoot.SetRotation(0,0,Math.RAD2DEG * m_AngleRad,true);
■■}
■■else
■■{
■■■m_LayoutRoot.SetRotation(0,0,m_RotationOverride,true);
■■}
■}
}


// ---------------------------------------------
// ------ HitDirectionSpike.c - END ----------------------
// ---------------------------------------------


// ---------------------------------------------
// ------ HitDirectionSplash.c - START --------------------
// ---------------------------------------------



class HitDirectionEffectSplash extends HitDirectionEffectBase
{
■override HitDirectionImagesBase GetImageData()
■{
■■typename type = HitDirectionImagesSplash;
■■HitDirectionImagesSplash data = HitDirectionImagesSplash.Cast(type.Spawn());
■■return data;
■}
■
■override void FinalizePositionCalculation()
■{
■■//blobs move around the edges, the rest on elypsis
■■m_PosX = m_PosXScreenEdge;
■■m_PosY = m_PosYScreenEdge;
```

```
// ---------------------------------------------
// ------ HitInfo.c - START --------------------
// ---------------------------------------------


class HitInfo
{
■proto native float GetSurfaceNoiseMultiplier();
■proto native string GetAmmoType();
■proto native vector GetPosition();
■proto native vector GetSurfaceNormal();
■proto native string GetSurface();
■proto native bool IsWater();
}


// ---------------------------------------------
// ------ HitInfo.c - END ----------------------
// ---------------------------------------------


// ---------------------------------------------
// ------ Hit_Concrete.c - START --------------------
// ---------------------------------------------


class Hit_Concrete : EffBulletImpactBase
{
■void Hit_Concrete()
■{
■■SetEnterParticle(ParticleList.IMPACT_CONCRETE_ENTER);
■■SetExitParticle(ParticleList.IMPACT_CONCRETE_EXIT);
■■SetRicochetParticle(ParticleList.IMPACT_CONCRETE_RICOCHET);
■■
■■m_AngledEnter = 0.50;
■}
}


// ---------------------------------------------
// ------ Hit_Concrete.c - END ----------------------
// ---------------------------------------------


// ---------------------------------------------
// ------ Hit_Dirt.c - START --------------------
// ---------------------------------------------


class Hit_Dirt : EffBulletImpactBase
{
■void Hit_Dirt()
■{
■■SetEnterParticle(ParticleList.IMPACT_DIRT_ENTER);
■■SetExitParticle(ParticleList.IMPACT_DIRT_EXIT);
■■SetRicochetParticle(ParticleList.IMPACT_DIRT_RICOCHET);
■}
}


// ---------------------------------------------
// ------ Hit_Dirt.c - END ----------------------
// ---------------------------------------------
// ---------------------------------------------
// ------ Hit_Error.c - START --------------------
// ---------------------------------------------


class Hit_ErrorNoMaterial : EffBulletImpactBase
{
■void Hit_ErrorNoMaterial()
■{
■■#ifdef DEVELOPER
■■SetSingleParticle(ParticleList.IMPACT_TEST_NO_MATERIAL_ERROR);
■■#else
■■SetSingleParticle(ParticleList.IMPACT_TEST);
■■#endif
■}
}


// ---------------------------------------------
// ------ Hit_Error.c - END ----------------------
// ---------------------------------------------


// ---------------------------------------------
// ------ Hit_Foliage.c - START --------------------
// ---------------------------------------------


class Hit_Foliage : EffBulletImpactBase
{
■void Hit_Foliage()
■{
■■SetEnterParticle(ParticleList.IMPACT_FOLIAGE_ENTER);
■■SetExitParticle(ParticleList.IMPACT_FOLIAGE_EXIT);
■■SetRicochetParticle(ParticleList.IMPACT_FOLIAGE_RICOCHET);
■}
■
■override float CalculateStoppingForce(float in_speedf, float out_speedf, string ammoType, float weight)
■{
■■if ( m_ImpactType == ImpactTypes.MELEE )
■■{
■■■return 500;
■■}
■■
■■float projectile_weight_coef = weight / DEFAULT_PROJECTILE_WEIGHT;
■■
■■float stopping_force = in_speedf * projectile_weight_coef * 0.5;
■■■■
■■return stopping_force;
■}
}


// ---------------------------------------------
// ------ Hit_Foliage.c - END ----------------------
// ---------------------------------------------
// ---------------------------------------------
// ------ Hit_Glass.c - START --------------------
// ---------------------------------------------


class Hit_Glass : EffBulletImpactBase
{
■void Hit_Glass()
■{
■■SetEnterParticle(ParticleList.IMPACT_GLASS_ENTER);
■■SetExitParticle(ParticleList.IMPACT_GLASS_EXIT);
■■SetRicochetParticle(ParticleList.IMPACT_GLASS_RICOCHET);
■}
}


// ---------------------------------------------
// ------ Hit_Glass.c - END ----------------------
// ---------------------------------------------


// ---------------------------------------------
// ------ Hit_Glass_Thin.c - START --------------------
// ---------------------------------------------


class Hit_Glass_Thin : EffBulletImpactBase
{
■void Hit_Glass_Thin()
■{
■■SetEnterParticle(ParticleList.IMPACT_GLASS_ENTER);
■■SetExitParticle(ParticleList.IMPACT_GLASS_EXIT);
■■SetRicochetParticle(ParticleList.IMPACT_GLASS_RICOCHET);
■}
}


// ---------------------------------------------
// ------ Hit_Glass_Thin.c - END ----------------------
// ---------------------------------------------


// ---------------------------------------------
// ------ Hit_Grass.c - START --------------------
// ---------------------------------------------


class Hit_Grass : EffBulletImpactBase
{
■void Hit_Grass()
■{
■■SetEnterParticle(ParticleList.IMPACT_GRASS_ENTER);
■■SetExitParticle(ParticleList.IMPACT_GRASS_ENTER);
■■SetRicochetParticle(ParticleList.IMPACT_GRASS_RICOCHET);
■■
■■m_AngledEnter = 0.6;
■}
}


// ---------------------------------------------
// ------ Hit_Grass.c - END ----------------------
// ---------------------------------------------
// ---------------------------------------------
// ------ Hit_Gravel.c - START --------------------
// ---------------------------------------------


class Hit_Gravel : EffBulletImpactBase
{
■void Hit_Gravel()
■{
■■SetEnterParticle(ParticleList.IMPACT_GRAVEL_ENTER);
■■SetExitParticle(ParticleList.IMPACT_GRAVEL_EXIT);
■■SetRicochetParticle(ParticleList.IMPACT_GRAVEL_RICOCHET);
■}
}


// ---------------------------------------------
// ------ Hit_Gravel.c - END ----------------------
// ---------------------------------------------


// ---------------------------------------------
// ------ Hit_MeatBones.c - START --------------------
// ---------------------------------------------


class Hit_MeatBones : EffBulletImpactBase
{
■float m_ScalingByDistance;
■
■void Hit_MeatBones()
■{
■■SetEnterParticle(ParticleList.IMPACT_MEATBONES_ENTER);
■■SetExitParticle(ParticleList.IMPACT_MEATBONES_EXIT);
■■SetRicochetParticle(ParticleList.IMPACT_MEATBONES_RICOCHET);
■■
■■m_AngledEnter = 10;
■■m_EnterSplashCoef = 0.002;
■■m_ExitSplashCoef = 0.006;
■■m_ScalingByDistance = 0.05;
■■
■■MIN_SCALING_PARAM = 0.2;
■}
■
■override float CalculateStoppingForce(float in_speedf, float out_speedf, string ammoType, float weight)
■{
■■if ( m_ImpactType == ImpactTypes.MELEE )
■■{
■■■return 400;
■■}
■■
■■float projectile_weight_coef = weight / DEFAULT_PROJECTILE_WEIGHT;
■■
■■float stopping_force = in_speedf * projectile_weight_coef;
■■
■■if (m_DirectHit)
■■{
■■■/*
■■■// TO DO: Doesn't work because IsAlive() is false, even when it shouldn't be.
■■■if ( m_DirectHit.IsMan()  &&  m_componentIndex == SURVIVOR_HEAD  &&  m_DirectHit.IsAlive() ) /
■■■{
■■■■stopping_force = stopping_force * 2;
■■■}
```

```
// ---------------------------------------------
// ------ Hit_MeatBones_MeleeFist.c - START --------------------
// ---------------------------------------------


class Hit_MeatBones_MeleeFist : Hit_MeatBones
{
■void Hit_MeatBones_MeleeFist()
■{
■■SetSingleParticle(ParticleList.INVALID);
■}
}


// ---------------------------------------------
// ------ Hit_MeatBones_MeleeFist.c - END ----------------------
// ---------------------------------------------


// ---------------------------------------------
// ------ Hit_MeatBones_MeleePipeWrench.c - START --------------------
// ---------------------------------------------


class Hit_MeatBones_MeleePipeWrench : Hit_MeatBones
{
■void Hit_MeatBones_MeleePipeWrench()
■{

■}
}


// ---------------------------------------------
// ------ Hit_MeatBones_MeleePipeWrench.c - END ----------------------
// ---------------------------------------------


// ---------------------------------------------
// ------ Hit_MeatBones_MeleeShovel.c - START --------------------
// ---------------------------------------------


class Hit_MeatBones_MeleeShovel : Hit_MeatBones
{
■void Hit_MeatBones_MeleeShovel()
■{

■}
}


// ---------------------------------------------
// ------ Hit_MeatBones_MeleeShovel.c - END ----------------------
// ---------------------------------------------


// ---------------------------------------------
// ------ Hit_MeatBones_MeleeWrench.c - START --------------------
// ---------------------------------------------


class Hit_MeatBones_MeleeWrench : Hit_MeatBones
```

```
// ---------------------------------------------
// ------ Hit_Metal.c - START --------------------
// ---------------------------------------------

class Hit_Metal : EffBulletImpactBase
{
■void Hit_Metal()
■{
■■SetEnterParticle(ParticleList.IMPACT_METAL_ENTER);
■■SetExitParticle(ParticleList.IMPACT_METAL_EXIT);
■■SetRicochetParticle(ParticleList.IMPACT_METAL_RICOCHET);
■}
}

// ---------------------------------------------
// ------ Hit_Metal.c - END ----------------------
// ---------------------------------------------

// ---------------------------------------------
// ------ Hit_Plaster.c - START --------------------
// ---------------------------------------------

class Hit_Plaster : EffBulletImpactBase
{
■void Hit_Plaster()
■{
■■SetEnterParticle(ParticleList.IMPACT_PLASTER_ENTER);
■■SetExitParticle(ParticleList.IMPACT_PLASTER_EXIT);
■■SetRicochetParticle(ParticleList.IMPACT_PLASTER_RICOCHET);
■}
}

// ---------------------------------------------
// ------ Hit_Plaster.c - END ----------------------
// ---------------------------------------------

// ---------------------------------------------
// ------ Hit_Plastic.c - START --------------------
// ---------------------------------------------

class Hit_Plastic : EffBulletImpactBase
{
■void Hit_Plastic()
■{
■■SetEnterParticle(ParticleList.IMPACT_PLASTIC_ENTER);
■■SetRicochetParticle(ParticleList.IMPACT_PLASTIC_ENTER);
■■SetExitParticle(ParticleList.IMPACT_PLASTIC_ENTER);
■}
}

// ---------------------------------------------
// ------ Hit_Plastic.c - END ----------------------
// ---------------------------------------------
// ---------------------------------------------
// ------ Hit_Rubber.c - START --------------------
// ---------------------------------------------


class Hit_Rubber : EffBulletImpactBase
{
■void Hit_Rubber()
■{
■■SetEnterParticle(ParticleList.IMPACT_RUBBER_ENTER);
■■SetExitParticle(ParticleList.IMPACT_RUBBER_EXIT);
■■SetRicochetParticle(ParticleList.IMPACT_RUBBER_RICOCHET);
■}
}


// ---------------------------------------------
// ------ Hit_Rubber.c - END ----------------------
// ---------------------------------------------


// ---------------------------------------------
// ------ Hit_Sand.c - START --------------------
// ---------------------------------------------


class Hit_Sand : EffBulletImpactBase
{
■void Hit_Sand()
■{
■■SetEnterParticle(ParticleList.IMPACT_SAND_ENTER);
■■SetRicochetParticle(ParticleList.IMPACT_SAND_ENTER);
■■SetExitParticle(ParticleList.IMPACT_SAND_ENTER);
■}
}


// ---------------------------------------------
// ------ Hit_Sand.c - END ----------------------
// ---------------------------------------------


// ---------------------------------------------
// ------ Hit_Textile.c - START --------------------
// ---------------------------------------------


class Hit_Textile : EffBulletImpactBase
{
■void Hit_Textile()
■{
■■SetEnterParticle(ParticleList.IMPACT_TEXTILE_ENTER);
■■SetRicochetParticle(ParticleList.IMPACT_TEXTILE_ENTER);
■■SetExitParticle(ParticleList.IMPACT_TEXTILE_EXIT);
■}
}


// ---------------------------------------------
// ------ Hit_Textile.c - END ----------------------
// ---------------------------------------------
// ---------------------------------------------
// ------ Hit_Undefined.c - START --------------------
// ---------------------------------------------


class Hit_Undefined : EffBulletImpactBase
{
■void Hit_Undefined()
■{
■■SetSingleParticle(ParticleList.IMPACT_TEST);
■}
}


// ---------------------------------------------
// ------ Hit_Undefined.c - END ----------------------
// ---------------------------------------------


// ---------------------------------------------
// ------ Hit_Water.c - START --------------------
// ---------------------------------------------


class Hit_Water : EffBulletImpactBase
{
■void Hit_Water()
■{
■■SetSingleParticle(ParticleList.IMPACT_WATER_SMALL_ENTER);
■■
■■m_AngledEnter = 10;
■}
■
■override void EvaluateEffect(Object directHit, int componentIndex, vector pos, int impact_type, vector sur
■{
■■super.EvaluateEffect(directHit, componentIndex, pos, impact_type, surfNormal, exitPos, inSpeed, outS
■■
■■m_SurfNormal = "0 0 0";
■}
■
■override float CalculateStoppingForce(float in_speedf, float out_speedf, string ammoType, float weight)
■{
■■float projectile_weight_coef = weight / DEFAULT_PROJECTILE_WEIGHT;
■■
■■float stopping_force = (in_speedf - out_speedf) * projectile_weight_coef;
■■
■■if ( stopping_force < 350 )
■■■SetEnterParticle(ParticleList.IMPACT_WATER_SMALL_ENTER);
■■
■■if ( stopping_force >= 350  &&  stopping_force < 750 )
■■■SetEnterParticle(ParticleList.IMPACT_WATER_MEDIUM_ENTER);
■■
■■if ( stopping_force >= 750 )
■■■SetEnterParticle(ParticleList.IMPACT_WATER_LARGE_ENTER);
■■
■■return stopping_force;
■}
■
■override void OnEnterCalculations( Particle p )
■{
■■// no particle scaling
■}
■
```

```
// ---------------------------------------------
// ------ Hit_Wood.c - START --------------------
// ---------------------------------------------


class Hit_Wood : EffBulletImpactBase
{
■void Hit_Wood()
■{
■■SetEnterParticle(ParticleList.IMPACT_WOOD_ENTER);
■■SetExitParticle(ParticleList.IMPACT_WOOD_EXIT);
■■SetRicochetParticle(ParticleList.IMPACT_WOOD_RICOCHET);
■}
■
■override void OnEnterCalculations( Particle p )
■{
■■// All values represent scale
■■float velocity_min = MIN_SCALING_PARAM + (m_StoppingForce * m_EnterSplashCoef);
■■float velocity_max = MIN_SCALING_PARAM + (m_StoppingForce * m_EnterSplashCoef);
■■float size = MIN_SCALING_PARAM + ( m_StoppingForce * m_EnterSplashCoef)*0.5;
■■float birth_rate = MIN_SCALING_PARAM + (m_StoppingForce * m_EnterSplashCoef)*0.5;
■■
■■if (velocity_min < MIN_SCALING_PARAM)
■■■velocity_min = MIN_SCALING_PARAM;
■■
■■if (size < MIN_SCALING_PARAM)
■■■size = MIN_SCALING_PARAM;
■■
■■if (birth_rate < MIN_SCALING_PARAM)
■■■birth_rate = MIN_SCALING_PARAM;
■■
■■
■■
■■p.ScaleParticleParam(EmitorParam.VELOCITY, velocity_min);
■■p.ScaleParticleParam(EmitorParam.VELOCITY_RND, velocity_max);
■■p.ScaleParticleParam(EmitorParam.SIZE, size);
■■p.ScaleParticleParam(EmitorParam.BIRTH_RATE, birth_rate);
■}
■
■override void OnExitCalculations(Particle p, float outSpeedf)
■{
■■float velocity_min = 1 + (outSpeedf * m_ExitSplashCoef);
■■float velocity_max = 1 + (outSpeedf * m_ExitSplashCoef);
■■float size = 1 + ( outSpeedf * m_ExitSplashCoef)*0.5;
■■float birth_rate = 1 + (outSpeedf * m_ExitSplashCoef)*0.5;
■■
■■if (velocity_min < MIN_SCALING_PARAM)
■■■velocity_min = MIN_SCALING_PARAM;
■■
■■if (size < MIN_SCALING_PARAM)
■■■size = MIN_SCALING_PARAM;
■■
■■if (birth_rate < MIN_SCALING_PARAM)
■■■birth_rate = MIN_SCALING_PARAM;
■■
■■p.ScaleParticleParam(EmitorParam.VELOCITY, velocity_min);
■■p.ScaleParticleParam(EmitorParam.VELOCITY_RND, velocity_max);
■■p.ScaleParticleParam(EmitorParam.SIZE, size);
■■p.ScaleParticleParam(EmitorParam.BIRTH_RATE, birth_rate);
■}
}
```

```
// ---------------------------------------------
// ------ Hive.c - START --------------------
// ---------------------------------------------



/** @file */

// -----------------------------------------------------------------------
class Hive
{
■private void Hive() {}
■private void ~Hive() {}
■
■proto native void InitOnline( string ceSetup, string host = "" );
■proto native void InitOffline();
■proto native void InitSandbox();

■proto native bool IsIdleMode();

■proto native void SetShardID( string shard );
■proto native void SetEnviroment( string env );

■proto native void CharacterSave( Man player );
■proto native void CharacterKill( Man player );
■proto native void CharacterExit( Man player );

■proto native void CallUpdater( string content );
};

proto native Hive CreateHive();
proto native void DestroyHive();
proto native Hive GetHive();



// ---------------------------------------------
// ------ Hive.c - END ----------------------
// ---------------------------------------------


// ---------------------------------------------
// ------ HockeyHelmet_ColorBase.c - START --------------------
// ---------------------------------------------


class HockeyHelmet_ColorBase extends HelmetBase {};
class HockeyHelmet_Black extends HockeyHelmet_ColorBase {};
class HockeyHelmet_Blue extends HockeyHelmet_ColorBase {};
class HockeyHelmet_Red extends HockeyHelmet_ColorBase {};
class HockeyHelmet_White extends HockeyHelmet_ColorBase {};


// ---------------------------------------------
// ------ HockeyHelmet_ColorBase.c - END ----------------------
// ---------------------------------------------
// ---------------------------------------------
// ------ HockeyMask.c - START --------------------
// ---------------------------------------------


class HockeyMask extends Clothing
{
■override bool CanPutAsAttachment( EntityAI parent )
■{
■■if (!super.CanPutAsAttachment(parent)) {return false;}
■■
■■Clothing headgear = Clothing.Cast(parent.FindAttachmentBySlotName("Headgear"));
■■if ( headgear && (headgear.ConfigGetBool("noMask") && !PumpkinHelmet.Cast(headgear)) )
■■{
■■■return false;
■■}
■■
■■Clothing eyewear = Clothing.Cast(parent.FindAttachmentBySlotName("Eyewear"));
■■if ( eyewear && SportGlasses_ColorBase.Cast(eyewear) )
■■{
■■■return false;
■■}
■■
■■return true;
■}
}


// ---------------------------------------------
// ------ HockeyMask.c - END ----------------------
// ---------------------------------------------


// ---------------------------------------------
// ------ HoldBreathEvents.c - START --------------------
// ---------------------------------------------


class HoldBreathSoundEventBase extends PlayerSoundEventBase
{
■void HoldBreathSoundEventBase()
■{
■■m_HasPriorityOverTypes = -1;
■}
■
■override bool HasHoldBreathException()
■{
■■return true;
■}

}


class HoldBreathSoundEvent extends HoldBreathSoundEventBase
{
■void HoldBreathSoundEvent()
■{
■■m_Type = EPlayerSoundEventType.GENERAL;
■■m_ID = EPlayerSoundEventID.HOLD_BREATH;
■■m_SoundVoiceAnimEventClassID = 20;
■}
■
■override bool HasPriorityOverCurrent(PlayerBase player, EPlayerSoundEventID other_state_id, EPlayer
// ---------------------------------------------
// ------ Hologram.c - START --------------------
// ---------------------------------------------


class Hologram
{
■#ifdef SERVER
■protected const int SPAWN_FLAGS ■= ECE_CREATEPHYSICS;
■#else
■protected const int SPAWN_FLAGS ■= ECE_LOCAL;
■#endif
■
■protected const string SUFFIX_MATERIAL_DEPLOYABLE ■= "_deployable.rvmat";
■protected const string SUFFIX_MATERIAL_UNDEPLOYABLE = "_undeployable.rvmat";
■protected const string SUFFIX_MATERIAL_POWERED ■■= "_powered.rvmat";

■protected ItemBase ■■■m_Parent;
■protected EntityAI ■■■m_Projection;
■protected PlayerBase ■■m_Player;
■protected ProjectionTrigger m_ProjectionTrigger;
■protected string■■■m_ProjectionTypename;

■protected bool ■■■■m_IsColliding;
■protected bool ■■■■m_IsCollidingGPlot;
■protected bool ■■■■m_IsSlope;
■protected bool ■■■■m_IsCollidingPlayer;
■protected bool ■■■■m_IsFloating;
■protected bool ■■■■m_UpdatePosition;
■protected bool ■■■■m_IsHidden;

■protected vector ■■■m_DefaultOrientation;
■protected vector■■■m_Rotation;
■protected vector■■■m_y_p_r_previous;
■protected vector ■■■m_ContactDir;
■protected vector ■■■m_FromAdjusted;
■protected const string ■■ANIMATION_PLACING ■■■■= "Placing";
■protected const string ■■ANIMATION_INVENTORY ■■■= "Inventory";
■protected const string ■■SELECTION_PLACING ■■■■= "placing";
■protected const string ■■SELECTION_INVENTORY ■■■= "inventory";

■protected const float ■■SMALL_PROJECTION_RADIUS ■■= 1;
■protected const float ■■SMALL_PROJECTION_GROUND ■■= 2;
■protected const float■■DISTANCE_SMALL_PROJECTION■■= 1; //! Deprecated
■protected const float■■LARGE_PROJECTION_DISTANCE_LIMIT■= 6;
■protected const float ■■PROJECTION_TRANSITION_MIN■■= 1;
■protected const float ■■PROJECTION_TRANSITION_MAX■■= 0.25;
■protected const float ■■LOOKING_TO_SKY■■■■■= 0.75;
■static const float ■■■DEFAULT_MAX_PLACEMENT_HEIGHT_DIFF = 1.5;
■
■protected float ■■■m_SlopeTolerance;
■protected bool■■■■m_AlignToTerrain;
■protected vector ■■■m_YawPitchRollLimit;
■protected int■■■■m_ContactComponent;
■
■protected ref set<string> ■m_SelectionsToRefresh ■■■= new set<string>;
■
■// Watchtower correction variables■
■// These watchtower component names should be corrected when colliding with them as they are suppos
■static const protected ref array<string>■m_WatchtowerIgnoreComponentNames■= new array<string>;
■
■// These watchtower components are supposed to be trigger boxes, but should block placement on them
■static const protected ref array<string>■m_WatchtowerBlockedComponentNames■= new array<string>
```

```
// ---------------------------------------------
// ------ Hoodie_ColorBase.c - START --------------------
// ---------------------------------------------


class Hoodie_ColorBase extends Clothing
{
■override void SetActions()
■{
■■super.SetActions();
■■AddAction(ActionWringClothes);
■}
};
class Hoodie_Blue extends Hoodie_ColorBase {};
class Hoodie_Black extends Hoodie_ColorBase {};
class Hoodie_Brown extends Hoodie_ColorBase {};
class Hoodie_Green extends Hoodie_ColorBase {};
class Hoodie_Grey extends Hoodie_ColorBase {};
class Hoodie_Red extends Hoodie_ColorBase {};


// ---------------------------------------------
// ------ Hoodie_ColorBase.c - END ----------------------
// ---------------------------------------------


// ---------------------------------------------
// ------ HorizontalSpacer.c - START --------------------
// ---------------------------------------------


// ----------------------------------------------------------
class HorizontalSpacer : SpacerBase
{
■reference int Border;
■reference int Gap;
■
■override protected void UpdateChild(Widget child, float w, float h, int index)
■{
■■float itemWidth = (w - (Border * 2) - ((m_count - 1) * Gap)) / m_count;
■■float itemHeight = h - (2 * Border);
■
■■child.SetPos(Border + ((itemWidth + Gap) * index), Border);
■■child.SetSize(itemWidth, itemHeight);
■■
■■if(child.GetChildren())
■■{
■■■Widget c = child.GetChildren();
■■■RightGap gap;
■■■c.GetScript(gap);
■■■if(gap)
■■■gap.OnUpdate(c);

■■■//gap.Update();
■■■
■■}
■}
};


// ---------------------------------------------
// ------ HorizontalSpacer.c - END ----------------------
// ---------------------------------------------
// ---------------------------------------------
// ------ HorizontalSpacerWithFixedAspect.c - START --------------------
// ---------------------------------------------


// --------------------------------------------------------
class HorizontalSpacerWithFixedAspect: ScriptedWidgetEventHandler
{
■protected Widget m_root;
■reference int border;
■reference int gap;
■reference float coef;
■float itemWidth;
■float itemHeight;
■
■// --------------------------------------------------------
■void OnWidgetScriptInit(Widget w)
■{
■■m_root = w;
■■m_root.SetHandler(this);
■}
■
■// --------------------------------------------------------
■override bool OnUpdate(Widget w)
■{
■■if (w == m_root) UpdateLayout();
■■return false;
■}
■
■protected void UpdateLayout()
■{
■■Widget child = m_root.GetChildren();
■■
■■■int index = 0;
■■■while (child)
■■■{
■■■■if( index == 0 )
■■■■{
■■■■■child.GetScreenSize(itemWidth, itemHeight);
■■■■}
■■■■else
■■■■{
■■■■■child.SetFlags( WidgetFlags.EXACTPOS, false);
■■■■■child.SetPos(itemWidth+(itemWidth*coef), 0);
■■■■}
■■■■
■■■■index++;
■■■■child = child.GetSibling();
■■}
■}
};


// ---------------------------------------------
// ------ HorizontalSpacerWithFixedAspect.c - END ---------------------
// ---------------------------------------------
// ---------------------------------------------
// ------ HotState.c - START --------------------
// ---------------------------------------------


class HotSymptom extends SymptomBase
{
■//this is just for the Symptom parameters set-up and is called even if the Symptom doesn't execute, don't
■override void OnInit()
■{
■■m_SymptomType = SymptomTypes.PRIMARY;
■■m_Priority = 1;
■■m_ID = SymptomIDs.SYMPTOM_HOT;
■■m_DestroyOnAnimFinish = true;
■■m_SyncToClient = false;
■■m_MaxCount = 2;
■}
■
■//!gets called every frame
■override void OnUpdateServer(PlayerBase player, float deltatime)
■{

■}

■override void OnUpdateClient(PlayerBase player, float deltatime)
■{
■}
■
■override void OnAnimationPlayFailed()
■{
■■
■}
■
■override bool CanActivate()
■{
■■return true;
■}
■
■//!gets called once on an Symptom which is being activated
■override void OnGetActivatedServer(PlayerBase player)
■{
■■if (LogManager.IsSymptomLogEnable()) Debug.SymptomLog("n/a", this.ToString(), "n/a", "OnGetActiva
■■HumanMovementState hms = new HumanMovementState();
■■player.GetMovementState(hms);
■■ItemBase item = m_Player.GetItemInHands();
■■
■■if(!(item && item.IsHeavyBehaviour()) && m_Manager.GetCurrentCommandID() == DayZPlayerConsta
■■{
■■■PlayAnimationADD(3);
■■}
■■else
■■{
■■■PlaySound(EPlayerSoundEventID.HOT);
■■}
■}

■//!gets called once on a Symptom which is being activated
■override void OnGetActivatedClient(PlayerBase player)
■{
■■if (LogManager.IsSymptomLogEnable()) Debug.SymptomLog("n/a", this.ToString(), "n/a", "OnGetActiva
■}

■//!only gets called once on an active Symptom that is being deactivated
```

```
// ---------------------------------------------
// ------ HouseBlock_1F1.c - START --------------------
// ---------------------------------------------


class Land_HouseBlock_1F1 extends BuildingWithFireplace
{
}


// ---------------------------------------------
// ------ HouseBlock_1F1.c - END ----------------------
// ---------------------------------------------


// ---------------------------------------------
// ------ HouseBlock_1F3.c - START --------------------
// ---------------------------------------------


class Land_HouseBlock_1F3 extends BuildingWithFireplace
{
}


// ---------------------------------------------
// ------ HouseBlock_1F3.c - END ----------------------
// ---------------------------------------------


// ---------------------------------------------
// ------ HouseBlock_1F4.c - START --------------------
// ---------------------------------------------


class Land_HouseBlock_1F4 extends BuildingWithFireplace
{
}


// ---------------------------------------------
// ------ HouseBlock_1F4.c - END ----------------------
// ---------------------------------------------


// ---------------------------------------------
// ------ HouseBlock_2F1.c - START --------------------
// ---------------------------------------------


class Land_HouseBlock_2F1 extends BuildingWithFireplace
{
}


// ---------------------------------------------
// ------ HouseBlock_2F1.c - END ----------------------
// ---------------------------------------------
// --------------------------------------------
// ------ HouseBlock_2F8.c - START --------------------
// --------------------------------------------

class Land_HouseBlock_2F8 extends BuildingWithFireplace
{
}

// --------------------------------------------
// ------ HouseBlock_2F8.c - END ----------------------
// --------------------------------------------

// --------------------------------------------
// ------ HouseBlock_2F9.c - START --------------------
// --------------------------------------------

class Land_HouseBlock_2F9 extends BuildingWithFireplace
{
}

// --------------------------------------------
// ------ HouseBlock_2F9.c - END ----------------------
// --------------------------------------------

// --------------------------------------------
// ------ HouseBlock_3F2.c - START --------------------
// --------------------------------------------

class Land_HouseBlock_3F2 extends BuildingWithFireplace
{
}

// --------------------------------------------
// ------ HouseBlock_3F2.c - END ----------------------
// --------------------------------------------

// --------------------------------------------
// ------ HouseBlock_5F.c - START --------------------
// --------------------------------------------

class Land_HouseBlock_5F extends BuildingWithFireplace
{
}

// --------------------------------------------
// ------ HouseBlock_5F.c - END ----------------------
// --------------------------------------------
// ---------------------------------------------
// ------ HoverEffect.c - START --------------------
// ---------------------------------------------


// ---------------------------------------------------------
class HoverEffect : ScriptedWidgetEventHandler
{
■reference float speed;
■reference float amount;
■protected float m_orginal_width;
■protected float m_orginal_height;
■protected Widget m_root;
■protected ref AnimatorTimer m_anim;
■
■void HoverEffect()
■{
■■m_anim = new AnimatorTimer();
■}
■
■// ---------------------------------------------------------
■void OnWidgetScriptInit(Widget w)
■{
■■m_root = w;
■■m_root.SetHandler(this);
■}
■
■// ---------------------------------------------------------
■protected void Update()
■{
■■float p = amount * m_anim.GetValue();
■■m_root.SetSize(m_orginal_width + (m_orginal_width * p), m_orginal_height + (m_orginal_height * p));
■
■■float c = 1.0 - (0.5 * m_anim.GetValue());
■■m_root.SetColor(ARGBF(1, 1, c, c));
■}
■
■// ---------------------------------------------------------
■override bool OnMouseEnter(Widget w, int x, int y)
■{
■■if ( !m_anim.IsRunning() ) m_root.GetSize(m_orginal_width, m_orginal_height);
■■m_anim.Animate(1.0, speed);
■■
■■return false;
■}
■
■// ---------------------------------------------------------
■override bool OnMouseLeave(Widget w, Widget enterW, int x, int y)
■{
■■m_anim.Animate(0.0, speed);
■■return false;
■}

};


// ---------------------------------------------
// ------ HoverEffect.c - END ---------------------
// ---------------------------------------------
// ---------------------------------------------
// ------ HoxtonMask.c - START --------------------
// ---------------------------------------------


class HoxtonMask extends ClothingBase
{
■override bool CanPutAsAttachment( EntityAI parent )
■{
■■if(!super.CanPutAsAttachment(parent)) {return false;}
■■bool headgear_present = false;
■■
■■if ( parent.FindAttachmentBySlotName( "Headgear" ) )
■■{
■■■headgear_present = parent.FindAttachmentBySlotName( "Headgear" ).ConfigGetBool( "noMask" );
■■}
■■
■■if ( ( GetNumberOfItems() == 0 || !parent || parent.IsMan() ) && !headgear_present )
■■{
■■■return true;
■■}
■■return false;
■}
}


// ---------------------------------------------
// ------ HoxtonMask.c - END ----------------------
// ---------------------------------------------


// ---------------------------------------------
// ------ HudDebug.c - START --------------------
// ---------------------------------------------


// *************************************************************************************
// ! PluginDayzPlayerDebugUI
// *************************************************************************************
class HudDebugEventHandler extends ScriptedWidgetEventHandler
{
■HudDebug m_HudDebug;
■
■void HudDebugEventHandler( HudDebug hud_debug )
■{
■■m_HudDebug = hud_debug;
■}
■
■HudDebug GetHudDebug()
■{
■■return m_HudDebug;
■}
■
■override bool OnClick( Widget w, int x, int y, int button )
■{
■■super.OnClick( w, x, y, button );
■■return GetHudDebug().OnClick( w, x, y, button );
■}
■
■override bool OnFocus(Widget w, int x, int y)
■{
■■//Print("OnFocus");
■■return true;
```

```
// ----------------------------------------------
// ------ HudDebugWinBase.c - START --------------------
// ----------------------------------------------


class HudDebugWinBase
{
■Widget m_WgtRoot;
■protected bool m_Show;
■protected bool m_RPCSent;
■
■//============================================
■// Constructor
■//============================================
■void HudDebugWinBase( Widget widget_root )
■{
■■m_WgtRoot = widget_root;
■}

■//============================================
■// GetType
■//============================================
■int GetType()
■{
■■return HudDebug.HUD_WIN_UNDEFINED;
■}

■void Init( Widget widget_root );
■
■void SetUpdate( bool state );
■
■void Update()
■{
■■if(m_Show && !m_RPCSent)
■■{
■■■SetUpdate(true);
■■}
■}
■
■
■//============================================
■// SetUpdate
■//============================================
■void SetRPCSent()
■{
■■m_RPCSent = true;
■}
■
■
■//============================================
■// Show
■//============================================
■void Show()
■{
■■m_WgtRoot.Show(true);
■■m_Show = true;
■}
■
■//============================================
■// Hide
■//============================================
■void Hide()
■{
```

```
// --------------------------------------------
// ------ HudDebugWinCharAgents.c - START --------------------
// --------------------------------------------


class DebugAgentData
{
■string m_Name;
■int m_ID;
■
■void DebugAgentData( string name, int id )
■{
■■m_Name = name;
■■m_ID = id;
■}
■
■string GetName()
■{
■■return m_Name;
■}
■
■int GetID()
■{
■■return m_ID;
■}
}


class HudDebugWinCharAgents extends HudDebugWinBase
{■
■protected Widget m_WgtAgents;
■protected ref array<ref Widget>■ m_AgentWidgets = new array<ref Widget>;
■protected ref map<Widget, ref DebugAgentData>■■m_AgentWidgetData = new map<Widget, ref Debug
■//============================================
■// HudDebugWinCharAgents
■//============================================
■void HudDebugWinCharAgents( Widget widget_root )
■{■
■■m_WgtRoot = widget_root;
■■//m_WgtAgents = TextListboxWidget.Cast( m_WgtRoot.FindAnyWidget( "txl_CharAgents_Values" ) );
■■m_WgtAgents = m_WgtRoot.FindAnyWidget( "AgentList" );
■■
■■//FitWindowByContent( m_WgtAgents );
■}

■void ~HudDebugWinCharAgents()
■{■
■■SetUpdate( false );
■}

■//============================================
■// GetWinType
■//============================================
■override int GetType()
■{
■■return HudDebug.HUD_WIN_CHAR_AGENTS;
■}
■
■//============================================
■// Update
■//============================================
■override void SetUpdate( bool state )
■{
```

```
// ---------------------------------------------
// ------ HudDebugWinCharDebug.c - START --------------------
// ---------------------------------------------


class HudDebugWinCharDebug extends HudDebugWinBase
{
■private PluginDeveloper m_ModuleDeveloper;
■
■private TextWidget ■■m_PlayerPosTextWidget;
■private TextWidget ■■m_ClipboardTextWidget;

■//============================================
■// HudDebugWinCharDebug
■//============================================
■void HudDebugWinCharDebug(Widget widget_root)
■{■
■■m_PlayerPosTextWidget = TextWidget.Cast( widget_root.FindAnyWidget("txt_PlayerPos") );
■■m_ClipboardTextWidget = TextWidget.Cast( widget_root.FindAnyWidget("txt_Clipboard") );
■}

■//============================================
■// ~HudDebugWinCharDebug
■//============================================
■void ~HudDebugWinCharDebug()
■{
■}
■
■//============================================
■// Update
■//============================================
■override void Update()
■{
■■super.Update();
■■
■■PlayerBase player = PlayerBase.Cast( GetGame().GetPlayer() );
■■if ( player != NULL )
■■{
■■■vector pos = player.GetPosition();
■■■string pos_str = "Pos: " + pos[0].ToString() + " " + pos[2].ToString();
■■■m_PlayerPosTextWidget.SetText(pos_str);
■■}
■■
■■string clipboard;
■■GetGame().CopyFromClipboard(clipboard);
■■clipboard = clipboard.Substring( 0, Math.Min( clipboard.Length(), 128 ) );■//max 128 chars
■■clipboard = "Clipboard: " + clipboard;
■■m_ClipboardTextWidget.SetText(clipboard);
■}

■//============================================
■// GetWinType
■//============================================
■override int GetType()
■{
■■return HudDebug.HUD_WIN_CHAR_DEBUG;
■}
}



// ---------------------------------------------
// ------ HudDebugWinCharDebug.c - END ----------------------
// ---------------------------------------------
// ------ HudDebugWinCharLevels.c - START --------------------
// ---------------------------------------------


class HudDebugWinCharLevels extends HudDebugWinBase
{
■TextListboxWidget■■■■m_WgtValues;
■
■//=============================================
■// Constructor
■//=============================================
■void HudDebugWinCharLevels(Widget widget_root)
■{■
■■m_WgtValues = TextListboxWidget.Cast( widget_root.FindAnyWidget("txl_CharLevels_Values") );
■■
■■FitWindow();
■}

■//=============================================
■// Destructor
■//=============================================
■void ~HudDebugWinCharLevels()
■{■
■■SetUpdate( false );
■}


■//=============================================
■// GetWinType
■//=============================================
■override int GetType()
■{
■■return HudDebug.HUD_WIN_CHAR_LEVELS;
■}
■
■//=============================================
■// Show
■//=============================================
■override void Show()
■{
■■super.Show();
■■
■■//Print("Show()");
■■
■■SetUpdate( true );
■}

■//=============================================
■// Hide
■//=============================================
■override void Hide()
■{
■■super.Hide();
■■
■■//Print("Hide()");
■■
■■SetUpdate( false );
■}

■//=============================================
■// SetUpdate
■//=============================================
// --------------------------------------------
// ------ HudDebugWinCharModifiers.c - START --------------------
// --------------------------------------------


class DebugModifierData
{
■string m_Name;
■int m_ID;
■
■void DebugModifierData( string name, int id )
■{
■■m_Name = name;
■■m_ID = id;
■}
■
■string GetName()
■{
■■return m_Name;
■}
■
■int GetID()
■{
■■return m_ID;
■}
}

class HudDebugWinCharModifiers extends HudDebugWinBase
{■
■protected Widget■■■■■■■■■■■m_WgtModifiersContent;
■protected ref array<ref Widget>■■■■■■■■m_ModifierWidgets;
■protected ref map<Widget, ref DebugModifierData>■■m_ModifierWidgetData;
■protected PluginDeveloperSync m_PluginDeveloperSync;
■protected Widget ■■■■■■■■■■■m_WgtDetailedInfo;
■protected TextWidget ■■■■■■■■■■m_WgtDetailedInfoText;
■protected int ■■■■■■■■■■■■m_DetailedInfoIndex;

//m_RPCSent
■
■//============================================
■// HudDebugWinCharModifiers
■//============================================
■void HudDebugWinCharModifiers( Widget widget_root )
■{■
■■m_WgtRoot = widget_root;
■■m_WgtModifiersContent = Widget.Cast( m_WgtRoot.FindAnyWidget( "pnl_CharModifiers_Values" ) );
■■m_ModifierWidgets = new array<ref Widget>;
■■m_ModifierWidgetData = new map<Widget, ref DebugModifierData>;
■■m_PluginDeveloperSync = PluginDeveloperSync.Cast( GetPlugin( PluginDeveloperSync ) );
■}

■void ~HudDebugWinCharModifiers()
■{■
■■SetUpdate( false );
■}

■//============================================
■// GetWinType
■//============================================
■override int GetType()
■{
■■return HudDebug.HUD_WIN_CHAR_MODIFIERS;
■}
```

```
// ---------------------------------------------
// ------ HudDebugWinCharStats.c - START --------------------
// ---------------------------------------------


class HudDebugWinCharStats extends HudDebugWinBase
{
■TextListboxWidget■■■■■■m_WgtValues;
■Widget■■■■■■■■■■m_WgtPanel;
■ref array<ref Widget> ■■■■■m_StatWidgets = new array<ref Widget>;
■ref map <ref SliderWidget, string> ■■m_SliderWidgets = new map<ref SliderWidget, string>;
■ref array<ref TextWidget> ■■■■m_StatValues = new array<ref TextWidget>;
■ref map<ref EditBoxWidget, string> ■■m_StatValuesInput = new map<ref EditBoxWidget, string>;
■bool■■■■■■■■■■m_Populated;
■bool■■■■■■■■■■m_ChangingSlider;
■
■//============================================
■// Constructor
■//============================================
■void HudDebugWinCharStats(Widget widget_root)
■{
■■m_WgtRoot = widget_root;
■■m_WgtPanel = Widget.Cast(m_WgtRoot.FindAnyWidget("Stats") );
■■//FitWindow();
■}

■//============================================
■// Destructor
■//============================================
■void ~HudDebugWinCharStats()
■{■
■■SetUpdate( false );
■}


■//============================================
■// GetWinType
■//============================================
■override int GetType()
■{
■■return HudDebug.HUD_WIN_CHAR_STATS;
■}
■
■//============================================
■// Show
■//============================================
■override void Show()
■{
■■super.Show();
■■
■■//Print("Show()");
■■
■■SetUpdate( true );
■}

■//============================================
■// Hide
■//============================================
■override void Hide()
■{
■■super.Hide();
■■
■■//Print("Hide()");
```

```
// ---------------------------------------------
// ------ HudDebugWinCharStomach.c - START --------------------
// ---------------------------------------------


class HudDebugWinCharStomach extends HudDebugWinBase
{
■TextListboxWidget■■■■m_WgtValues;
■TextWidget■■■■■■m_WgtOverall;
■
■//============================================
■// Constructor
■//============================================
■void HudDebugWinCharStomach(Widget widget_root)
■{■
■■m_WgtValues = TextListboxWidget.Cast( widget_root.FindAnyWidget("txl_StomachContents") );
■■m_WgtOverall = TextWidget.Cast( widget_root.FindAnyWidget("InfoOverall") );
■■//FitWindow();
■}

■//============================================
■// Destructor
■//============================================
■void ~HudDebugWinCharStomach()
■{■
■■SetUpdate( false );
■}


■//============================================
■// GetWinType
■//============================================
■override int GetType()
■{
■■return HudDebug.HUD_WIN_CHAR_STOMACH;
■}
■
■//============================================
■// Show
■//============================================
■override void Show()
■{
■■super.Show();
■■
■■//Print("Show()");
■■
■■SetUpdate( true );
■}

■//============================================
■// Hide
■//============================================
■override void Hide()
■{
■■super.Hide();
■■
■■//Print("Hide()");
■■
■■SetUpdate( false );
■}

■//============================================
■// SetUpdate
```

```
// ---------------------------------------------
// ------ HudDebugWinTemperature.c - START --------------------
// ---------------------------------------------


class HudDebugWinTemperature extends HudDebugWinBase
{
■private PluginDeveloper ■■m_ModuleDeveloper;
■
■private TextWidget ■■■■m_EnviroTextWidget;
■protected PluginDeveloperSync ■m_PluginDeveloperSync;
■//============================================
■// HudDebugWinTemperature
■//============================================
■void HudDebugWinTemperature(Widget widget_root)
■{■
■■m_EnviroTextWidget = TextWidget.Cast( widget_root.FindAnyWidget("txt_Temp") );
■■m_PluginDeveloperSync = PluginDeveloperSync.Cast( GetPlugin( PluginDeveloperSync ) );
■■
■}

■//============================================
■// ~HudDebugWinTemperature
■//============================================
■void ~HudDebugWinTemperature()
■{
■}
■
■//============================================
■// Set Update
■//============================================
■override void SetUpdate( bool state )
■{
■■//Disable update on server (PluginDeveloperSync)
■■PlayerBase player = PlayerBase.Cast( GetGame().GetPlayer() );
■■
■■//if client, send RPC
■■if ( GetGame().IsClient() )
■■{
■■■CachedObjectsParams.PARAM1_BOOL.param1 = state;
■■■if ( player )
■■■{
■■■■player.RPCSingleParam( ERPCs.DEV_TEMP_UPDATE, CachedObjectsParams.PARAM1_BOOL,
■■■■SetRPCSent();
■■■}
■■}
■■//else set directly
■■else
■■{
■■■if ( m_PluginDeveloperSync )
■■■{
■■■■m_PluginDeveloperSync.EnableUpdate( state, ERPCs.DEV_TEMP_UPDATE, player );■
■■■}
■■}
■}

■//============================================
■// Show / Hide
■//============================================
■override void Show()
■{
■■super.Show();
■■
```

```
// ---------------------------------------------
// ------ HudDebugWinVersion.c - START --------------------
// ---------------------------------------------


class HudDebugWinVersion extends HudDebugWinBase
{
■private PluginDeveloper m_ModuleDeveloper;
■
■private TextWidget ■■m_VersionTextWidget;

■//============================================
■// HudDebugWinVersion
■//============================================
■void HudDebugWinVersion(Widget widget_root)
■{■
■■m_VersionTextWidget = TextWidget.Cast( widget_root.FindAnyWidget("txt_Version") );
■■
■■string version;
■■g_Game.GetVersion(version);
■■m_VersionTextWidget.SetText(string.Format("exe: %1 | scripts: %2", version, GetPBOAPI().GetPBOVe
■}

■//============================================
■// ~HudDebugWinVersion
■//============================================
■void ~HudDebugWinVersion()
■{
■}
■
■//============================================
■// Update
■//============================================
■override void Update()
■{
■■super.Update();
■}

■//============================================
■// GetWinType
■//============================================
■override int GetType()
■{
■■return HudDebug.HUD_WIN_VERSION;
■}
}



// ---------------------------------------------
// ------ HudDebugWinVersion.c - END ----------------------
// ---------------------------------------------


// ---------------------------------------------
// ------ human.c - START --------------------
// ---------------------------------------------


/** @file

  this file is interface to Human
```

```
// ---------------------------------------------
// ------ HumanInventory.c - START --------------------
// ---------------------------------------------


/**@class■■HumanInventory
 * @brief■■inventory for plain man/human
 *
 * HumanInventory has simple synchronous operations only, i.e.
 * no animations are involved while adding/removing/swapping to/from hands.
 *
 * Animations are added on higher level of hierarchy (DayZPlayerInventory for example)
 **/
class HumanInventory : GameInventory
{■
■int m_syncClearUserReservationindex = -1;
■//int m_UserReservationToClear = -1;
■/**@fn■■■GetEntityInHands
■ * @return■■entity in hands
■ **/
■proto native EntityAI GetEntityInHands ();

■/**@fn■■■CanAddEntityInHands
■ * @brief■■alternative for TestAddEntityInHands(e, true, true, true);
■ **/
■proto native bool CanAddEntityInHands (EntityAI e);

■/**@fn■■■TestAddEntityInHands
■ * @param[in]■e■entity to test for taking in hands
■ * @param[in]■do_item_check■■deny if entity is not InventoryItem
■ * @param[in]■do_occupancy_test■■deny if there is item in hands already
■ * @param[in]■do_script_check■■deny if script conditions fail
■ * @return■■true if item passed all tests
■ **/
■proto native bool TestAddEntityInHands (EntityAI e, bool do_resevation_check, bool do_item_check, boo

■/**@fn■■■CanRemoveEntityInHands
■ * @return■■true if entity can be removed from hands
■ **/
■proto native bool CanRemoveEntityInHands ();

■proto native bool CanOpenInventory();

■/**@fn■■■CreateInHands
■ * @brief■■creates new entity in hands
■ * @param[in]■typeName type name of the entity to be created
■ * @return■■new entity or null otherwise
■ **/
■proto native EntityAI CreateInHands (string typeName);

■proto native int GetUserReservedLocationCount ();
■proto native int FindUserReservedLocationIndex (notnull EntityAI e);
■proto native int FindCollidingUserReservedLocationIndex (notnull EntityAI e, notnull InventoryLocation ds
■proto native void GetUserReservedLocation (int index, out notnull InventoryLocation dst);
■proto native int FindFirstUserReservedLocationIndexForContainer(notnull EntityAI e);
■
■proto native void SetUserReservedLocation (notnull EntityAI eai, notnull InventoryLocation dst);
■proto native void ClearUserReservedLocation (notnull EntityAI eai);
■proto native bool ClearUserReservedLocationAtIndex (int index);
■proto native void ClearUserReservedLocationForContainer (notnull EntityAI eai);
■proto native bool GetDebugFlag ();

■/**
```

```
// ---------------------------------------------
// ------ HumanInventoryWithFSM.c - START --------------------
// ---------------------------------------------


/**
 * @class■HumanInventoryWithFSM
 * @brief■HumanInventory... with FSM (synchronous, no anims)
 **/
class HumanInventoryWithFSM : HumanInventory
{
■protected ref HandFSM m_FSM;  /// hand slot state machine
■protected ref HandStateBase m_Empty;
■protected ref HandStateBase m_Equipped;

■void HumanInventoryWithFSM ()
■{
■■m_FSM = new HandFSM();
■}
■
■void CreateStableStates ()
■{
■■// stable states are created only if they do not exist already (for example created by derived class like D
■■// @NOTE: this cannot be done in constructor, as there is NO owner set yet. GetManOwner() will there
■■if (!m_Empty)
■■■m_Empty = new HandStateEmpty(GetManOwner());
■■if (!m_Equipped)
■■■m_Equipped = new HandStateEquipped(GetManOwner());
■}

■override void Init ()
■{
■■// setup state machine
■■hndDebugPrint("[hndfsm] Initializing Human Inventory FSM");
■■
■■// basic states
■■CreateStableStates();
■■
■■// events
■■HandEventBase __T__ = new HandEventTake;
■■HandEventBase __D__ = new HandEventDrop;
■■HandEventBase __Tw_ = new HandEventThrow;
■■HandEventBase __M__ = new HandEventMoveTo;
■■HandEventBase __W__ = new HandEventSwap;
■■HandEventBase __F__ = new HandEventForceSwap;
■■HandEventBase __X__ = new HandEventDestroy;
■■HandEventBase __Xd_ = new HandEventDestroyed;
■■HandEventBase __R__ = new HandEventDestroyAndReplaceWithNew;
■■HandEventBase __Re_ = new HandEventDestroyAndReplaceWithNewElsewhere;
■■HandEventBase __Rh_ = new HandEventDestroyElsewhereAndReplaceWithNewInHands;
■■HandEventBase __Rd_ = new HandEventReplaced;
■■HandEventBase __Cd_ = new HandEventCreated;
■■HandEventBase _fin_ = new HandEventHumanCommandActionFinished;
■■HandEventBase _abt_ = new HandEventHumanCommandActionAborted;


■■HandReplacingItemInHands replacing = new HandReplacingItemInHands(GetManOwner());
■■HandReplacingItemInHands replacingElsewhere = new HandReplacingItemInHands(GetManOwner());
■■HandReplacingItemElsewhereWithNewInHands replacingElsewhere3 = new HandReplacingItemElsew

■■// setup transitions
■■m_FSM.AddTransition(new HandTransition( m_Empty   , __T__,  m_Equipped, new HandActionTake, 
■■m_FSM.AddTransition(new HandTransition( m_Empty   , __Cd__,  m_Equipped, new HandActionCreate
// ---------------------------------------------
// ------ humansettings.c - START --------------------
// ---------------------------------------------


class SHumanGlobalSettings
{
■private void SHumanGlobalSettings() {}
■private void ~SHumanGlobalSettings() {}
■
■float■■m_fPhysMoveMaxTracedDistance;■■//!<■when actual phys step is lower than this -> distance is
}


class SHumanCommandMoveSettings
{
■private void SHumanCommandMoveSettings() {}
■private void ~SHumanCommandMoveSettings() {}
■
■float ■■m_fRunSpringTimeout;■■■■//!< filter span value■■[s]
■float■■m_fRunSpringMaxChange;■■■■//!< filter speed value■■[val/s]
■float■■m_fDirFilterTimeout;■■■■//!< direction filter timeout [s]
■float■■m_fDirFilterSprintTimeout;■■■//!< sprint direction filter timeout [s]
■float■■m_fDirFilterSpeed;■■■■■//!< direction filter max rot speed [rad/s]
■float■■m_fMaxSprintAngle;■■■■■//!< max sprint angle [rad]

■float ■■m_fTurnAngle;■■■■■■//!< when character starts to turn (45 deg default)
■float ■■m_fTurnEndUpdateNTime;■■■■//!< when turn values are not updated anymore (normalized tin
■float ■■m_fTurnNextNTime;■■■■■//!< when next turn is started from previous turn

■float■■m_fSlidingPoseAngle;■■■■//!< angle (0.707 * PI)■- when angle is bigger - sliding pose is done
■float■■m_fSlidingPoseTrackTime;■■■//!< time change of angle is tracked in the past (sec)
■float■■m_fSlidingPoseRepTime;■■■■//!< not to do sliding pose after another sliding pose (sec)

■float■■m_fHeadingChangeLimiterIdle;
■float■■m_fHeadingChangeLimiterWalk;
■float■■m_fHeadingChangeLimiterRun;
■
■float■■m_fLeaningSpeed;

■float■■m_fWaterLevelSpeedRectrictionLow;■//!< Water level (in meters) - m_iMaxSpeedNormal_Water
■float■■m_fWaterLevelSpeedRectrictionHigh;■//!< Water level (in meters) - m_iMaxSpeedNormal_Water
■
■int■■■m_iMaxSpeedNormal_WaterLevelLow;■//!< Max movement speed for m_fWaterLevelSpeedRec
■int■■■m_iMaxSpeedFast_WaterLevelLow;■■//!< Max movement speed for m_fWaterLevelSpeedRectr
■int■■■m_iMaxSpeedNormal_WaterLevelHigh;■//!< Max movement speed for m_fWaterLevelSpeedRec
■int■■■m_iMaxSpeedFast_WaterLevelHigh;■■//!< Max movement speed for m_fWaterLevelSpeedRect
}

class SHumanCommandSwimSettings
{
■private void SHumanCommandSwimSettings() {}
■private void ~SHumanCommandSwimSettings() {}■
■
■float ■■m_fAlignIdleTimeout;■■■■//! align filters in idle, slow, fast
■float ■■m_fAlignIdleMaxChanged;■
■float ■■m_fAlignSlowTimeout;■■
■float ■■m_fAlignSlowMaxChanged;■
■float ■■m_fAlignFastTimeout;■■
■float ■■m_fAlignFastMaxChanged;■

■float■■m_fMovementSpeed;■■■
■float■■m_fMovementSpeedFltTime;
```

```
// ---------------------------------------------
// ------ HumanSteakMeat.c - START --------------------
// ---------------------------------------------


class HumanSteakMeat extends Edible_Base
{
■void HumanSteakMeat()
■{
■■InsertAgent(eAgents.BRAIN, 1);
■}
■
■override bool CanBeCooked()
■{
■■return true;
■}■■
■
■override bool CanBeCookedOnStick()
■{
■■return true;
■}
■
■override bool IsMeat()
■{
■■return true;
■}
■
■override bool CanDecay()
■{
■■return true;
■}

■override void SetActions()
■{
■■super.SetActions();
■■
■■AddAction(ActionForceFeed);
■■AddAction(ActionEatMeat);
■■
■■AddAction(ActionCreateIndoorFireplace);
■■AddAction(ActionCreateIndoorOven);
■}
}




// ---------------------------------------------
// ------ HumanSteakMeat.c - END ----------------------
// ---------------------------------------------


// ---------------------------------------------
// ------ Hunger.c - START --------------------
// ---------------------------------------------


class HungerMdfr: ModifierBase
{
■protected float■m_EnergyDelta;
■protected float■m_LastEnergyLevel;
■ref HumanMovementState■■m_MovementState■= new HumanMovementState();
■■
■override void Init()
```

```
// ---------------------------------------------
// ------ HungerNotfr.c - START --------------------
// ---------------------------------------------


class HungerNotfr: NotifierBase
{
■protected const float ■DEC_TRESHOLD_LOW ■■■= 0;
■protected const float ■DEC_TRESHOLD_MED ■■■= -0.35;
■protected const float ■DEC_TRESHOLD_HIGH■■■= -0.57;
■protected const float ■INC_TRESHOLD_LOW ■■■= 0;
■protected const float ■INC_TRESHOLD_MED ■■■= 0.35;
■protected const float ■INC_TRESHOLD_HIGH■■■= 0.57;
■
■void HungerNotfr(NotifiersManager manager)
■{
■}

■override int GetNotifierType()
■{
■■return eNotifiers.NTF_HUNGRY;
■}

■override void DisplayTendency(float delta)
■{
■■//PrintString("delta:"+delta.ToString());
■■int tendency = CalculateTendency(delta, INC_TRESHOLD_LOW, INC_TRESHOLD_MED, INC_TRES
■■//GetVirtualHud().SetStatus(eDisplayElements.DELM_TDCY_ENERGY,tendency);
■■
■■//DSLevels level = DetermineLevel( GetObservedValue(), PlayerConstants.THRESHOLD_ENERGY_W
■
■■EStatLevels energy_level = m_Player.GetStatLevelEnergy();
■■DisplayElementTendency dis_elm = DisplayElementTendency.Cast(GetVirtualHud().GetElement(eDisp
■■
■■if( dis_elm )
■■{
■■■dis_elm.SetTendency(tendency);
■■■dis_elm.SetSeriousnessLevel(energy_level);
■■■
■■}
■■
■■/*
■■Print("------------------------------------------------------------");
■■Print("water:"+ typename.EnumToString(EStatLevels,m_Player.GetStatLevelWater()));
■■Print("energy:"+ typename.EnumToString(EStatLevels,m_Player.GetStatLevelEnergy()));
■■Print("health:"+ typename.EnumToString(EStatLevels,m_Player.GetStatLevelHealth()));
■■Print("blood:"+ typename.EnumToString(EStatLevels,m_Player.GetStatLevelBlood()));
■■*/
■}

■override void DisplayBadge()
■{
■■
■}

■override void HideBadge()
■{
■}

■
■override float GetObservedValue()
■{
■■return m_Player.GetStatEnergy().Get();
```

```
// ---------------------------------------------
// ------ HungerSoundHandler.c - START --------------------
// ---------------------------------------------


class HungerSoundHandlerBase extends SoundHandlerBase
{
■override void Init()
■{
■■m_Id = eSoundHandlers.HUNGER;
■}
■
}

//---------------------------
// Client
//---------------------------
class HungerSoundHandlerClient extends HungerSoundHandlerBase
{
■const float SOUND_INTERVALS_LIGHT_MIN = 10;
■const float SOUND_INTERVALS_LIGHT_MAX = 30;
■float m_SoundTime;
■EffectSound m_Sound;
■
■override void Update()
■{
■■if ( m_Player.GetMixedSoundStates() & eMixedSoundStates.HUNGRY )
■■{
■■■ProcessSound();
■■}
■}
■
■void ProcessSound()
■{
■■if ( GetGame().GetTime() > m_SoundTime)
■■{
■■■float offset_time = Math.RandomFloatInclusive(SOUND_INTERVALS_LIGHT_MIN, SOUND_INTERV
■■■m_SoundTime = GetGame().GetTime() + offset_time;
■■■PlaySound();
■■}
■}
■
■void PlaySound()
■{
■■m_Sound = SEffectManager.PlaySoundOnObject("hungry_uni_Voice_Char_SoundSet", m_Player);
■■
■■if( m_Sound )
■■{
■■■m_Sound.SetAutodestroy(true);
■■}
■■else
■■{
■■■Debug.LogError("Missing sounset");
■■}
■}
}


//---------------------------
// Server
//---------------------------
class HungerSoundHandlerServer extends HungerSoundHandlerBase
{
```

```
// ---------------------------------------------
// ------ HunterPants_ColorBase.c - START --------------------
// ---------------------------------------------


class HunterPants_ColorBase extends Clothing
{
■override void SetActions()
■{
■■super.SetActions();
■■AddAction(ActionWringClothes);
■}
};
class HunterPants_Autumn extends HunterPants_ColorBase {};
class HunterPants_Brown extends HunterPants_ColorBase {};
class HunterPants_Spring extends HunterPants_ColorBase {};
class HunterPants_Summer extends HunterPants_ColorBase {};
class HunterPants_Winter extends HunterPants_ColorBase {};


// ---------------------------------------------
// ------ HunterPants_ColorBase.c - END ----------------------
// ---------------------------------------------


// ---------------------------------------------
// ------ HuntingBag.c - START --------------------
// ---------------------------------------------


class HuntingBag : Clothing {};



// ---------------------------------------------
// ------ HuntingBag.c - END ----------------------
// ---------------------------------------------


// ---------------------------------------------
// ------ HuntingJacket_ColorBase.c - START --------------------
// ---------------------------------------------


class HuntingJacket_ColorBase extends Clothing
{
■override void SetActions()
■{
■■super.SetActions();
■■AddAction(ActionWringClothes);
■}
};
class HuntingJacket_Autumn extends HuntingJacket_ColorBase {};
class HuntingJacket_Brown extends HuntingJacket_ColorBase {};
class HuntingJacket_Spring extends HuntingJacket_ColorBase {};
class HuntingJacket_Summer extends HuntingJacket_ColorBase {};
class HuntingJacket_Winter extends HuntingJacket_ColorBase {};


// ---------------------------------------------
// ------ HuntingJacket_ColorBase.c - END ----------------------
// ---------------------------------------------
// ---------------------------------------------
// ------ HuntingKnife.c - START --------------------
// ---------------------------------------------


class HuntingKnife extends ToolBase
{
■override bool IsMeleeFinisher()
■{
■■return true;
■}
■
■override array<int> GetValidFinishers()
■{
■■return {EMeleeHitType.FINISHER_LIVERSTAB,EMeleeHitType.FINISHER_NECKSTAB};
■}
■
■override void SetActions()
■{
■■super.SetActions();
■■
■■AddAction(ActionBurnSewTarget);
■■AddAction(ActionUnrestrainTarget);
■■AddAction(ActionSkinning);
■■AddAction(ActionMineBush);
■■AddAction(ActionMineTreeBark);
■■AddAction(ActionBurnSewSelf);
■■AddAction(ActionDigWorms);
■■AddAction(ActionShaveTarget);
■■AddAction(ActionDisarmMine);
■■AddAction(ActionDisarmExplosive);
■■AddAction(ActionShave);
■}
}


// ---------------------------------------------
// ------ HuntingKnife.c - END ----------------------
// ---------------------------------------------


// ---------------------------------------------
// ------ HuntingOptic.c - START --------------------
// ---------------------------------------------


class HuntingOptic extends ItemOptics
{
■EntityAI m_Parent;
■
■void HuntingOptic()
■{
■■HideSelection("rings_ris");
■■HideSelection("rings_ris_pilot");
■}
■
■override void OnWasAttached( EntityAI parent, int slot_id )
■{
■■super.OnWasAttached(parent, slot_id);
■■m_Parent = parent;
■■if (!ParentUsesWinchesterTypeMount())
■■{
■■■HideSelection("rings_winchester");
```

```
// ---------------------------------------------
// ------ HuntingVest.c - START --------------------
// ---------------------------------------------


class HuntingVest extends Clothing {};


// ---------------------------------------------
// ------ HuntingVest.c - END ----------------------
// ---------------------------------------------


// ---------------------------------------------
// ------ Iceaxe.c - START --------------------
// ---------------------------------------------


class Iceaxe: Inventory_Base
{
■override void SetActions()
■{
■■super.SetActions();
■■
■■AddAction(ActionBuryAshes);
■■AddAction(ActionDigGardenPlot);
■■AddAction(ActionDismantleGardenPlot);
■■//AddAction(ActionMineRock1H);
■■AddAction(ActionDigWorms);
■■AddAction(ActionDismantlePart);
■■AddAction(ActionBuildPart);
■■AddAction(ActionSkinning);
■■AddAction(ActionCreateGreenhouseGardenPlot);
■}
};



// ---------------------------------------------
// ------ Iceaxe.c - END ----------------------
// ---------------------------------------------


// ---------------------------------------------
// ------ Icon.c - START --------------------
// ---------------------------------------------


class Icon: LayoutHolder
{
■protected int■■■■■m_SizeX;
■protected int■■■■■m_SizeY;
■protected int■■■■■m_PosX;
■protected int■■■■■m_PosY;
■
■protected EntityAI■■■■m_Lock;
■protected bool■■■■■m_IsWeapon■■■= false;
■protected bool■■■■■m_IsMagazine■■= false;
■protected bool■■■■■m_HasTemperature■= false;
■protected bool■■■■■m_HasQuantity■■= false;
■protected float■■■■■m_CurrQuantity■■= -1;
■
■protected EntityAI■■■■m_Obj;
```

```
// ---------------------------------------------
// ------ IconsContainer.c - START --------------------
// ---------------------------------------------


class IconsContainer: Container
{
■protected ref map<int, ref Icon> m_EntitiesMap = new map<int, ref Icon>;

■void IconsContainer( LayoutHolder parent )
■{
■}

■override void UnfocusAll()
■{
■■for ( int i = 0; i < Count(); i++ )
■■{
■■■for ( int j = 0; j < ITEMS_IN_ROW; j++ )
■■■{
■■■■Get( i ).GetMainWidget().FindAnyWidget( "Cursor" + j ).Show( false );
■■■}
■■}
■■
■■
■■for ( i = 0; i < m_EntitiesMap.Count(); ++i )
■■{
■■■m_EntitiesMap.GetElement(i).SetActive(false);
■■}
■}
■
■int GetItemCount()
■{
■■return m_EntitiesMap.Count();
■}
■
■void AddItem( Icon icon )
■{■
■■if( !m_EntitiesMap.Contains( icon.GetObject().GetID() ) )
■■{
■■■m_EntitiesMap.Insert( icon.GetObject().GetID(), icon);
■■}
■}
■
■Icon GetIcon( int entity_id )
■{
■■return m_EntitiesMap.Get( entity_id  );
■}
■
■Icon GetIconByIndex( int index )
■{
■■if( index < m_EntitiesMap.Count() && index > -1 )
■■■return m_EntitiesMap.GetElement( index  );
■■return null;
■}

■void RemoveItem( Icon icon )
■{
■■if( icon )
■■{
■■■Icon icon_copy = icon; // for some reason garbage collector collects icon too soon, so copy has to be
■■■if( icon.GetObject() )
■■■■m_EntitiesMap.Remove( icon.GetObject().GetID() );
■■■else
```

```
// ----------------------------------------------
// ------ ImmuneSystem.c - START --------------------
// ----------------------------------------------


class ImmuneSystemMdfr: ModifierBase
{
■bool m_HasDisease;
■bool m_HasHealings;
■override void Init()
■{
■■m_TrackActivatedTime = false;
■■m_ID ■■■■■= eModifiers.MDF_IMMUNE_SYSTEM;
■■m_TickIntervalInactive ■= DEFAULT_TICK_TIME_INACTIVE;
■■m_TickIntervalActive ■= DEFAULT_TICK_TIME_ACTIVE;
■■
■■DisableDeactivateCheck();
■}
■
■override bool ActivateCondition(PlayerBase player)
■{
■■return true;
■}

■override void OnActivate(PlayerBase player)
■{
■}
■
■override void OnReconnect(PlayerBase player)
■{

■}

■override bool DeactivateCondition(PlayerBase player)
■{
■■return false;
■}

■override void OnTick(PlayerBase player, float deltaT)
■{
//■■Debug.Log("ticking immune system", "agent");
■■float result ■= player.GetImmunity() * deltaT;
■■player.ImmuneSystemTick(result, deltaT);
//■■Debug.Log("result: "+result.ToString(), "agent");
■■
■■if( m_HasDisease != player.HasDisease() )
■■{
■■■if(player.HasDisease())
■■■{
■■■■if( player.GetNotifiersManager() ) player.GetNotifiersManager().ActivateByType(eNotifiers.NTF_SIC
■■■}
■■■else
■■■{
■■■■if( player.GetNotifiersManager() ) player.GetNotifiersManager().DeactivateByType(eNotifiers.NTF_S
■■■}
■■■m_HasDisease = player.HasDisease();
■■}■

■■■
■■if( m_HasHealings != player.HasHealings() )
■■{
■■
■■■if( player.HasHealings() )
```

```
// ---------------------------------------------
// ------ ImmunityBoost.c - START --------------------
// ---------------------------------------------


class ImmunityBoost: ModifierBase
{
■float ■■■m_LastWaterLevel;
■ref HumanMovementState■■m_MovementState■= new HumanMovementState();
■float m_RegenTime;
■
■override void Init()
■{
■■m_TrackActivatedTime = true;
■■m_IsPersistent = true;
■■m_ID ■■■■■= eModifiers.MDF_IMMUNITYBOOST;
■■m_TickIntervalInactive ■= DEFAULT_TICK_TIME_INACTIVE;
■■m_TickIntervalActive ■= 1;
■■m_RegenTime = PlayerConstants.VITAMINS_LIFETIME_SECS;
■}■■

■override bool ActivateCondition(PlayerBase player)
■{
■■return false;
■}
■
■override void OnReconnect(PlayerBase player)
■{
■■OnActivate(player);
■}
■
■override string GetDebugText()
■{
■■return (m_RegenTime - GetAttachedTime()).ToString();
■}
■
■override void OnActivate(PlayerBase player)
■{
■■player.SetImmunityBoosted(true);
■■player.IncreaseHealingsCount();
■■/*
■■if( player.GetNotifiersManager() )
■■■player.GetNotifiersManager().ActivateByType(eNotifiers.NTF_PILLS);
■■*/
■}
■
■override void OnDeactivate(PlayerBase player)
■{
■■player.SetImmunityBoosted(false);
■■player.DecreaseHealingsCount();
■■/*
■■if( player.GetNotifiersManager() )
■■■player.GetNotifiersManager().DeactivateByType(eNotifiers.NTF_PILLS);
■■*/
■}
■
■override bool DeactivateCondition(PlayerBase player)
■{
■■float attached_time = GetAttachedTime();
■■
■■if( attached_time >= m_RegenTime )
■■{
■■■return true;■
```

```
// ---------------------------------------------
// ------ ImpactEffects.c - START --------------------
// ---------------------------------------------


enum ImpactTypes
{
■UNKNOWN;
■STOP;
■PENETRATE;
■RICOCHET;
■MELEE;
}

class ImpactMaterials
{
■ref static map<string, typename> ■m_ImpactEffect;
■static int ■■■■■■■m_LastRegisteredMaterial = 0;
■
■//! Map of ammo which will not spawn any impact effect
■ref static map<string, int>■■■m_IgnoredAmmo;
■static int ■■■■■■■m_LastRegisteredIgnoredAmmo = 0;
■
■/** \name  Surface effects
■*■Register all hit materials here!
■*■When modding, these can be unregistered with 'UnregisterSurface' if so desired
■*/
■//@{
■static int PLASTIC■ ■■■■■= RegisterSurface("Hit_Plastic");
■static int SAND ■■■■■■= RegisterSurface("Hit_Sand");
■static int TEXTILE ■■■■■■= RegisterSurface("Hit_Textile");
■static int CONCRETE ■■■■■= RegisterSurface("Hit_Concrete");
■static int GRAVEL ■■■■■■= RegisterSurface("Hit_Gravel");
■static int DIRT ■■■■■■= RegisterSurface("Hit_Dirt");
■static int FOLIAGE ■■■■■■= RegisterSurface("Hit_Foliage");
■static int GRASS ■■■■■■= RegisterSurface("Hit_Grass");
■static int WOOD ■■■■■■= RegisterSurface("Hit_Wood");
■static int METAL ■■■■■■= RegisterSurface("Hit_Metal");
■static int GLASS ■■■■■■= RegisterSurface("Hit_Glass");
■static int GLASS_THIN ■■■■■= RegisterSurface("Hit_Glass_Thin");
■static int WATER ■■■■■■= RegisterSurface("Hit_Water");
■static int RUBBER ■■■■■■= RegisterSurface("Hit_Rubber");
■static int PLASTER ■■■■■■= RegisterSurface("Hit_Plaster");
■static int MEATBONES■■■■■= RegisterSurface("Hit_MeatBones");
■static int MEATBONES_SHOVEL■■■■= RegisterSurface("Hit_MeatBones_MeleeShovel");
■static int MEATBONES_PIPEWRENCH■■■= RegisterSurface("Hit_MeatBones_MeleePipeWrench");
■static int MEATBONES_WRENCH■■■■= RegisterSurface("Hit_MeatBones_MeleeWrench");
■static int MEATBONES_FIST■■■■= RegisterSurface("Hit_MeatBones_MeleeFist");
■static int UNDEFINED ■■■■■= RegisterSurface("Hit_Undefined");
■static int ERROR_NO_MATERIAL■■■= RegisterSurface("Hit_ErrorNoMaterial");
■//@}
■
■/** \name  Ignored ammo
■*■Register ammo which will not spawn impact effects here
■*■When modding, these can be unregistered with 'UnregisterIgnoredAmmo' if so desired
■*/
■//@{
■static int FIST■■■■■■■= RegisterIgnoredAmmo("MeleeFist");
■static int FIST_HEAVY■■■■■= RegisterIgnoredAmmo("MeleeFist_Heavy");
■static int SOFT■■■■■■■= RegisterIgnoredAmmo("MeleeSoft");
■static int SOFT_HEAVY■■■■■= RegisterIgnoredAmmo("MeleeSoft_Heavy");
■//@}
■
```

```
// ---------------------------------------------
// ------ ImprovisedBag.c - START --------------------
// ---------------------------------------------


class ImprovisedBag : Clothing {};



// ---------------------------------------------
// ------ ImprovisedBag.c - END ---------------------
// ---------------------------------------------


// ---------------------------------------------
// ------ ImprovisedExplosive.c - START --------------------
// ---------------------------------------------


class ImprovisedExplosive : ExplosivesBase
{
■protected const float TIME_TRIGGER_INITIAL_DELAY_SECS ■■■= 0.1;
■protected const float TIME_TRIGGER_TIMER_BASED_DELAY_SECS ■■= 1.0;
■protected const float TIME_TRIGGER_DELAY_SECS ■■■■■= 0.3;
■
■protected static const string SLOT_TRIGGER_ALARM_CLOCK■ ■■= "TriggerAlarmClock";
■protected static const string SLOT_TRIGGER_KITCHEN_TIMER■■= "TriggerKitchenTimer";
■protected static const string SLOT_TRIGGER_REMOTE■■■■= "TriggerRemoteDetonator_Receiver";
■
■protected static const string SLOT_EXPLOSIVE_A■■■■■= "IEDExplosiveA";
■protected static const string SLOT_EXPLOSIVE_B■■■■■= "IEDExplosiveB";
■
■protected const int SLOT_EXPLOSIVE_COUNT■■■■■■= 2;
■protected const string SLOT_EXPLOSIVES[SLOT_EXPLOSIVE_COUNT] ■= {
■■SLOT_EXPLOSIVE_A,
■■SLOT_EXPLOSIVE_B
■};
■
■protected const int SLOT_TRIGGERS_COUNT■■■■■■■= 3;
■protected const string SLOT_TRIGGERS[SLOT_TRIGGERS_COUNT]■■= {
■■SLOT_TRIGGER_ALARM_CLOCK,
■■SLOT_TRIGGER_KITCHEN_TIMER,
■■SLOT_TRIGGER_REMOTE
■};
■
■protected const string ANIM_PHASE_TRIGGER_EMPTY■■■■■= "TriggerEmpty";
■protected const string ANIM_PHASE_TRIGGER_TIMER■■■■■= "TriggerTimer";
■protected const string ANIM_PHASE_TRIGGER_CLOCK■■■■■= "TriggerClock";
■protected const string ANIM_PHASE_TRIGGER_REMOTE■■■■= "TriggerRemote";
■
■protected ref RemotelyActivatedItemBehaviour m_RAIB;
■
■void ImprovisedExplosive()
■{
■■m_RAIB = new RemotelyActivatedItemBehaviour(this);

■■RegisterNetSyncVariableInt("m_RAIB.m_PairDeviceNetIdLow");
■■RegisterNetSyncVariableInt("m_RAIB.m_PairDeviceNetIdHigh");
■}

■override void EOnInit(IEntity other, int extra)
■{
■■LockTriggerSlots();
```

```
// ---------------------------------------------
// ------ ImprovisedFishingRod.c - START --------------------
// ---------------------------------------------


class ImprovisedFishingRod: FishingRod_Base_New
{
■override void AnimateFishingRod(bool state)
■{
■■SetAnimationPhase("AnimateRod",state);
■}
};




// ---------------------------------------------
// ------ ImprovisedFishingRod.c - END ----------------------
// ---------------------------------------------


// ---------------------------------------------
// ------ ImprovisedSuppressor.c - START --------------------
// ---------------------------------------------


class ImprovisedSuppressor extends SuppressorBase
{
■
■const int SLOTS_ARRAY = 9;
■
■string slot_names[SLOTS_ARRAY] = { /*"weaponMuzzleAK", */"weaponBayonetAK", "weaponBayonet",
■■

■override bool CanPutAsAttachment( EntityAI parent )
■{
■■bool cond_state = true;
■■if(!super.CanPutAsAttachment(parent)) {return false;}
■■for ( int i = 0; i < SLOTS_ARRAY ; i++ )
■■{
■■■if ( parent.FindAttachmentBySlotName(slot_names[i]) != NULL )
■■■{
■■■■cond_state = false;
■■■■break;
■■■}
■■}

■■if ( cond_state && !parent.IsKindOf("PlateCarrierHolster") && !parent.IsKindOf("PlateCarrierComplete")
■■{
■■■return true;
■■}
■■return false;
■}
}


// ---------------------------------------------
// ------ ImprovisedSuppressor.c - END ----------------------
// ---------------------------------------------
// ---------------------------------------------
// ------ InfectedSoundEventBase.c - START --------------------
// ---------------------------------------------


enum EInfectedSoundEventType
{
■GENERAL,
}

class InfectedSoundEventBase extends SoundEventBase
{
■ZombieBase m_Infected;

■void InfectedSoundEventBase()
■{
■■m_Type = EInfectedSoundEventType.GENERAL;
■}
■
■void ~InfectedSoundEventBase()
■{
■■if (m_SoundSetCallback) m_SoundSetCallback.Stop();
■}

■void Init(ZombieBase pInfected)
■{
■■m_Infected = pInfected;
■}
■
■void SoftStop()
■{
■■if (m_SoundSetCallback)
■■{
■■■m_SoundSetCallback.Loop(false);
■■■m_SoundSetCallback = null;
■■}
■}

■override void Stop()
■{
■■if (m_SoundSetCallback)
■■{
■■■m_SoundSetCallback.Stop();
■■■m_SoundSetCallback = null;
■■}

■■GetGame().GetCallQueue(CALL_CATEGORY_GAMEPLAY).RemoveByName(this, "PosUpdate");
■}
■
■void PosUpdate()
■{
■■if (m_SoundSetCallback)
■■{
■■■m_SoundSetCallback.SetPosition(m_Infected.GetPosition());
■■}
■}
■
■override bool Play()
■{
■■string soundset_name;
■■■
■■soundset_name = string.Format("%1_%2_SoundSet", m_Infected.ClassName(), m_SoundSetNameRo
■■m_SoundSetCallback = m_Infected.ProcessVoiceFX(soundset_name);
```

```
// ---------------------------------------------
// ------ InfectedSoundEventHandler.c - START --------------------
// ---------------------------------------------


enum EInfectedSoundEventID
{
■MINDSTATE_CALM_IDLE = 1,
■MINDSTATE_CALM_MOVE,
■MINDSTATE_DISTURBED_IDLE,
■//MINDSTATE_CHASE_IDLE,
■MINDSTATE_CHASE_MOVE,
■MINDSTATE_ALERTED_IDLE,
■MINDSTATE_ALERTED_MOVE,
■//--------------
■ENUM_COUNT,
}

class InfectedSoundEventHandler extends SoundEventHandler
{
■const int SOUND_EVENTS_MAX = EInfectedSoundEventID.ENUM_COUNT;
■static ref InfectedSoundEventBase m_AvailableStates[SOUND_EVENTS_MAX];
■ref InfectedSoundEventBase m_CurrentState;
■ZombieBase m_Infected;
■
■void InfectedSoundEventHandler(ZombieBase pInfected)
■{
■■m_Infected = pInfected;
■■m_AvailableStates[EInfectedSoundEventID.MINDSTATE_CALM_IDLE] = new CalmIdleSoundEvent();
■■m_AvailableStates[EInfectedSoundEventID.MINDSTATE_CALM_MOVE] = new CalmMoveSoundEven
■■m_AvailableStates[EInfectedSoundEventID.MINDSTATE_DISTURBED_IDLE] = new DisturbedIdleSou
■■//m_AvailableStates[EInfectedSoundEventID.MINDSTATE_CHASE_IDLE] = new ChaseIdleSoundEve
■■m_AvailableStates[EInfectedSoundEventID.MINDSTATE_CHASE_MOVE] = new ChaseMoveSoundE
■■m_AvailableStates[EInfectedSoundEventID.MINDSTATE_ALERTED_IDLE] = new AlertedIdleSoundE
■■m_AvailableStates[EInfectedSoundEventID.MINDSTATE_ALERTED_MOVE] = new AlertedMoveSoun
■}
■
■override static int GetSoundEventType(int id)
■{
■■return m_AvailableStates[id].GetSoundEventType();
■}
■
■override int GetCurrentStateEventID()
■{
■■if (m_CurrentState)
■■{
■■■return m_CurrentState.GetSoundEventID();
■■}
■■return -1;
■}
■
■override int GetCurrentStateEventType()
■{
■■if (m_CurrentState)
■■{
■■■return m_CurrentState.GetSoundEventType();
■■}
■■return -1;
■}
■
■void Stop()
■{
■■if ( m_CurrentState )
```

```
// ---------------------------------------------
// ------ Influenza.c - START --------------------
// ---------------------------------------------


class InfluenzaMdfr: ModifierBase
{
■const int AGENT_THRESHOLD_ACTIVATE = 300;
■const int AGENT_THRESHOLD_DEACTIVATE = 200;
■
■override void Init()
■{
■■m_TrackActivatedTime = false;
■■m_ID ■■■■■= eModifiers.MDF_INFLUENZA;
■■m_TickIntervalInactive ■= DEFAULT_TICK_TIME_INACTIVE;
■■m_TickIntervalActive ■= DEFAULT_TICK_TIME_ACTIVE;
■}
■
■override string GetDebugText()
■{
■■return ("Activate threshold: "+AGENT_THRESHOLD_ACTIVATE + "| " +"Deativate threshold: "+AGEN
■}
■
■override protected bool ActivateCondition(PlayerBase player)
■{
■■if(player.GetSingleAgentCount(eAgents.INFLUENZA) >= AGENT_THRESHOLD_ACTIVATE)
■■{
■■■return true;
■■}
■■return false;
■}

■override protected void OnActivate(PlayerBase player)
■{
■■player.IncreaseDiseaseCount();
■}

■override protected void OnDeactivate(PlayerBase player)
■{
■■player.DecreaseDiseaseCount();
■}

■override protected bool DeactivateCondition(PlayerBase player)
■{
■■return (player.GetSingleAgentCount(eAgents.INFLUENZA) <= AGENT_THRESHOLD_DEACTIVATE);
■}

■override protected void OnTick(PlayerBase player, float deltaT)
■{
■■float chance_of_cough = player.GetSingleAgentCountNormalized(eAgents.INFLUENZA);
■■
■■if( Math.RandomFloat01() < chance_of_cough / Math.RandomInt(5,20) )
■■{
■■■player.GetSymptomManager().QueueUpPrimarySymptom(SymptomIDs.SYMPTOM_COUGH);
■■}
■}
};


// ---------------------------------------------
// ------ Influenza.c - END ----------------------
// ---------------------------------------------
// ---------------------------------------------
// ------ InfluenzaAgent.c - START --------------------
// ---------------------------------------------


class InfluenzaAgent extends AgentBase
{
■const float INFLUENZA_AGENT_AUTOINFECT_THRESHOLD_HC = PlayerConstants.THRESHOLD_H

■override void Init()
■{
■■m_Type ■■■■■= eAgents.INFLUENZA;
■■m_Invasibility ■■■= 0.33;
■■m_TransferabilityIn■■= 1;
■■m_TransferabilityOut■= 1;
■■m_MaxCount ■■■■= 1000;
■■m_Digestibility■■■= 0.1;
■■m_AntibioticsResistance = 0;
■■m_AutoinfectProbability = CalculateAutoinfectProbability( 0.40 );
■■m_TransferabilityAirOut = 1;
■■m_Potency ■■■■= EStatLevels.MEDIUM;
■■m_DieOffSpeed ■■■= 0.66;
■}

■override bool CanAutoinfectPlayer(PlayerBase player)
■{
■■if ( player.GetStatHeatComfort().Get() < INFLUENZA_AGENT_AUTOINFECT_THRESHOLD_HC )
■■{
■■■return true;■■■
■■}
■■else
■■{
■■■return false;■
■■}
■}
}


// ---------------------------------------------
// ------ InfluenzaAgent.c - END ----------------------
// ---------------------------------------------


// ---------------------------------------------
// ------ IngameHud.c - START --------------------
// ---------------------------------------------


class IngameHud extends Hud
{
■protected const float■■■■■■FADE_IN_TIME = 0.3;
■protected const float■■■■■■FADE_OUT_TIME = 0.3;
■protected const float■■■■■■HIDE_MENU_TIME = 5;

■protected ref map<int,string>■■■■m_StatesWidgetNames;
■protected ref map<int,ImageWidget>■■■m_StatesWidgets;  // [key] ImageWidget

■protected ref map<ImageWidget, int>■■■m_TendencyStatusCritical;■//array of icons that are blinking d
■protected const float■■■■■■TENDENCY_BLINK_TIME = 0.25;
■protected float■■■■■■■■m_BlinkTime;
■
■protected ref map<int,string>■■■■m_BadgesWidgetNames;
■protected ref map<int,int>■■■■■m_BadgesWidgetDisplay;
```

```
// ---------------------------------------------
// ------ InGameMenu.c - START --------------------
// ---------------------------------------------


class InGameMenu extends UIScriptedMenu
{
■string ■■■■■■m_ServerInfoText;
■
■protected Widget■■■m_ContinueButton;
■protected Widget■■■m_SeparatorPanel;
■protected Widget■■■m_ExitButton;
■protected Widget■■■m_RestartButton;
■protected Widget ■■■m_RespawnButton;
■protected Widget■■■m_RestartDeadRandomButton;
■protected Widget■■■m_RestartDeadCustomButton;
■protected Widget■■■m_OptionsButton;
■protected Widget ■■■m_ServerInfoPanel;
■protected Widget ■■■m_FavoriteButton;
■protected Widget ■■■m_FavoriteImage;
■protected Widget ■■■m_UnfavoriteImage;
■protected Widget ■■■m_CopyInfoButton;
■
■protected ref TextWidget■m_ModdedWarning;
■protected ref TextWidget ■m_ServerIP;
■protected ref TextWidget ■m_ServerPort;
■protected ref TextWidget ■m_ServerName;
■
■protected ref UiHintPanel m_HintPanel;
■
■void ~InGameMenu()
■{
■■HudShow(true);
■■
■■Mission mission = g_Game.GetMission();
■■if (mission)
■■{
■■■mission.Continue();
■■}
■}

■override Widget Init()
■{
■■layoutRoot = GetGame().GetWorkspace().CreateWidgets("gui/layouts/day_z_ingamemenu.layout");
■■■■
■■m_ContinueButton■■■= layoutRoot.FindAnyWidget("continuebtn");
■■m_SeparatorPanel■■■= layoutRoot.FindAnyWidget("separator_red");
■■m_ExitButton■■■■= layoutRoot.FindAnyWidget("exitbtn");
■■m_RestartButton■■■■= layoutRoot.FindAnyWidget("restartbtn");
■■m_RespawnButton ■■■= layoutRoot.FindAnyWidget("respawn_button");
■■m_RestartDeadRandomButton■= layoutRoot.FindAnyWidget("respawn_button_random");
■■m_RestartDeadCustomButton■= layoutRoot.FindAnyWidget("respawn_button_custom");
■■m_OptionsButton■■■■= layoutRoot.FindAnyWidget("optionsbtn");
■■m_ModdedWarning■■■■= TextWidget.Cast(layoutRoot.FindAnyWidget("ModdedWarning"));
■■m_HintPanel■■■■■= new UiHintPanel(layoutRoot.FindAnyWidget("hint_frame"));
■■m_ServerInfoPanel ■■■= layoutRoot.FindAnyWidget("server_info");
■■m_ServerIP ■■■■■= TextWidget.Cast(layoutRoot.FindAnyWidget("server_ip"));
■■m_ServerPort ■■■■= TextWidget.Cast(layoutRoot.FindAnyWidget("server_port"));
■■m_ServerName ■■■■= TextWidget.Cast(layoutRoot.FindAnyWidget("server_name"));
■■m_FavoriteImage ■■■= layoutRoot.FindAnyWidget("favorite_image");
■■m_UnfavoriteImage ■■■= layoutRoot.FindAnyWidget("unfavorite_image");
■■m_CopyInfoButton ■■■= layoutRoot.FindAnyWidget("copy_button");
■■
```

```
// ---------------------------------------------
// ------ InGameMenuXbox.c - START --------------------
// ---------------------------------------------


class InGameMenuXbox extends UIScriptedMenu
{
//#ifdef PLATFORM_CONSOLE
■
■// Widgets texts id
■protected  string ■■■■■■m_MuteButtonTextID;
■protected  string ■■■■■■m_UnmuteButtonTextID;
■protected  string ■■■■■■m_BackButtonTextID;
■protected  string ■■■■■■m_SelectButtonTextID;
■protected  string ■■■■■■m_OpenGameCardButtonTextID;
■protected  string ■■■■■■m_CurrentMuteButtonText;
■
■protected bool ■■■■■■■m_SelectAvailable;
■protected bool ■■■■■■■m_MuteAvailable;
■protected bool ■■■■■■■m_BackAvailable;
■protected bool ■■■■■■■m_GamercardAvailable;
■
■protected bool ■■■■■■■m_PlayerAlive;
■
■protected ref PlayerListScriptedWidget■m_ServerInfoPanel;
■
■protected Widget■■■■■■m_OnlineMenu;
■
■protected ButtonWidget■■■■■m_ContinueButton;
■protected ButtonWidget■■■■■m_ExitButton;
■protected ButtonWidget■■■■■m_RestartDeadButton;
■protected ButtonWidget■■■■■m_RestartButton;
■protected ButtonWidget■■■■■m_OptionsButton;
■protected ButtonWidget■■■■■m_ControlsButton;
■protected ButtonWidget■■■■■m_OnlineButton;
■protected ButtonWidget■■■■■m_TutorialsButton;
■
■protected TextWidget■■■■■m_Version;
■
■const int ■■■■■■■■BUTTON_XBOX_CONTROLS = 201;
■
■void InGameMenuXbox()
■{
■■GetGame().GetMission().GetOnInputPresetChanged().Insert(OnInputPresetChanged);
■■GetGame().GetMission().GetOnInputDeviceChanged().Insert(OnInputDeviceChanged);
■}

■void ~InGameMenuXbox()
■{
■■ClientData.SyncEvent_OnPlayerListUpdate.Remove(SyncEvent_OnRecievedPlayerList);
■■OnlineServices.m_PermissionsAsyncInvoker.Remove(OnPermissionsUpdate);
■■
■■Mission mission = GetGame().GetMission();
■■if (mission)
■■{
■■■IngameHud hud = IngameHud.Cast(mission.GetHud());
■■■if (hud)
■■■{
■■■■hud.ShowHudUI(true);
■■■■hud.ShowQuickbarUI(true);
■■■}
■■■
■■■mission.Continue();
```

```
// ---------------------------------------------
// ------ InjectionVial.c - START --------------------
// ---------------------------------------------


class InjectionVial: Inventory_Base {};



// ---------------------------------------------
// ------ InjectionVial.c - END ---------------------
// ---------------------------------------------


// ---------------------------------------------
// ------ InjuryEvents.c - START --------------------
// ---------------------------------------------


class InjurySoundEvents extends PlayerSoundEventBase
{
■
■void InjurySoundEvents()
■{
■■m_HasPriorityOverTypes = -1;//-1 for all
■■m_Type = EPlayerSoundEventType.INJURY;
■}
■
■override bool CanPlay(PlayerBase player)
■{
■■return true;
■}
■
■override bool HasPriorityOverCurrent(PlayerBase player, EPlayerSoundEventID other_state_id,EPlayerS
■{
■■return true;
■}
}

class InjuryLightSoundEvent extends InjurySoundEvents
{
■void InjuryLightSoundEvent()
■{
■■
■■m_ID = EPlayerSoundEventID.INJURED_LIGHT;
■■m_SoundVoiceAnimEventClassID = 28;
■}
}

class InjuryMediumSoundEvent extends InjurySoundEvents
{
■void InjuryMediumSoundEvent()
■{
■■
■■m_ID = EPlayerSoundEventID.INJURED_MEDIUM;
■■m_SoundVoiceAnimEventClassID = 28;
■}
}

class InjuryHeavySoundEvent extends InjurySoundEvents
{
■void InjuryHeavySoundEvent()
■{
```

```
// ---------------------------------------------
// ------ InjuryHandler.c - START --------------------
// ---------------------------------------------


class InjuryHandlerThresholds
{
■const float WORN = 0.5;
■const float DAMAGED = 0.3;
■const float BADLY_DAMAGED = 0.2;
■const float RUINED = 0.1;
};

class InjuryAnimValues
{
■const float LVL0 = 0;
■const float LVL1 = 0.3;
■const float LVL2 = 0.6;
■const float LVL3 = 0.8;
■const float LVL4 = 1;
};

enum eInjuryHandlerLevels
{
■PRISTINE,
■WORN,
■DAMAGED,
■BADLY_DAMAGED,
■RUINED,
}

enum eInjuryOverrides
{
■//! MUST BE POW2
■NONE = 0,
■MORPHINE = 1,
■PAIN_KILLERS_LVL0 = 2,
■PAIN_KILLERS_LVL1 = 4,
■BROKEN_LEGS = 8, //New
■BROKEN_LEGS_SPLINT = 16, //New
■PRONE_ANIM_OVERRIDE = 32, //Prevent non pristine animation when prone
}

class InjuryAnimationHandler
{

■const float VALUE_CHECK_INTERVAL■ ■= 5;
■const float SENSITIVTY_PERCENTAGE ■■= 1; //how much the value needs to change up/down from p

■ref ScriptInvoker m_ChangedStateInvoker■= new ScriptInvoker();
■
■float m_TimeSinceLastTick = VALUE_CHECK_INTERVAL + 1;
■float m_LastUpdate;
■eInjuryHandlerLevels m_LastHealthUpdate;
■float m_HealthMaxValue;
■
■
■private PlayerBase m_Player; //! owner
■private bool m_AnimationChange = false;
■private bool m_InjuryAnimEnabled = false;
■private float m_InjuryAnimDamageValue = 0;
■int m_ForceInjuryAnimMask;
■
```

```
// ---------------------------------------------
// ------ InjurySoundHandler.c - START --------------------
// ---------------------------------------------


enum eInjurySoundZones
{
■NONE = 0,
■LIGHT,
■MEDIUM,
■HEAVY,
■
■// PUT NEW STATES ABOVE
■COUNT
}

class InjurySoundHandlerBase extends SoundHandlerBase
{
■override void Init()
■{
■■m_Id = eSoundHandlers.INJURY;
■}
■
}

//---------------------------
// Client
//---------------------------
class InjurySoundHandlerClient extends InjurySoundHandlerBase
{
■const float SOUND_INTERVALS_LIGHT_MIN ■= 15; const float SOUND_INTERVALS_LIGHT_MAX = 3
■const float SOUND_INTERVALS_MEDIUM_MIN ■= 10; const float SOUND_INTERVALS_MEDIUM_MA
■const float SOUND_INTERVALS_HEAVY_MIN■= 3; const float SOUND_INTERVALS_HEAVY_MAX = 
■
■ref HumanMovementState m_MovementState = new HumanMovementState;
■eInjurySoundZones m_InjurySoundZone;
■eInjuryHandlerLevels m_InjuryLevel;
■float m_SoundTime;

■
■eInjurySoundZones DetermineInjuryZone(eInjuryHandlerLevels level)
■{
■■if( level == eInjuryHandlerLevels.PRISTINE )
■■■return eInjurySoundZones.NONE;
■■
■■m_Player.GetMovementState(m_MovementState);
■■int speed = m_MovementState.m_iMovement;
■■int stance = m_MovementState.m_iStanceIdx;
■■
■■//int stance_lvl_down = DayZPlayerConstants.STANCEMASK_PRONE | DayZPlayerConstants.STANC
■■
■■if( speed == 0 )
■■■return eInjurySoundZones.NONE;
■■
■■level--;// shift the level so that we play from higher damage
■■
■■if( stance ==  DayZPlayerConstants.STANCEIDX_PRONE || stance ==  DayZPlayerConstants.STANC
■■{
■■■level--;
■■}
■■
■■if( speed == DayZPlayerConstants.MOVEMENTIDX_WALK )
■■{
```

```
// ---------------------------------------------
// ------ input.c - START --------------------
// ---------------------------------------------



enum EInputDeviceType
{
■UNKNOWN,
■MOUSE_AND_KEYBOARD,
■CONTROLLER
};

//-----------------------------------------------------------------------------
class Input
{
■// input locking
■
■/**
■\brief Change game focus number
■@param add number to add to focus number
■@param input_device if equals -1, works globally on all devices, see INPUT_DEVICE_* values in consta
■@see HasGameFocus()
■*/
■proto native void■■ChangeGameFocus(int add, int input_device = -1);
■
■/**
■\brief Reset game focus number (set to 0)
■@param input_device if equals -1, works globally on all devices, see INPUT_DEVICE_* values in consta
■@see HasGameFocus()
■*/
■proto native void■■ResetGameFocus(int input_device = -1);
■
■/**
■\brief Check if game should have focus
■@param input_device if equals -1, checks all devices, see INPUT_DEVICE_* values in constants.h
■@return true if focus number == 0, false otherwise
■*/
■proto native bool■■HasGameFocus(int input_device = -1);

■// actions
■proto native int■■GetActionGroupsCount();
■proto native int■■GetActionGroupSize(int group_index);
■proto int■■■■GetActionGroupName(int group_index, out string name);
■proto int■■■■GetActionDesc(int action_index, out string desc);

■// getting action state
■/**
■\brief Get action state
■@param action id of action, defined in \ref 4_World/Classes/UserActionsComponent/_constants.c
■@param check_focus if true and game is unfocused, returns 0; otherwise returns actual value
■@return actual action state as float, for regular two state buttons returns 0 or 1, for analog buttons/axes r
■@see LocalValue()
■*/
■proto native float■LocalValue_ID(int action, bool check_focus = true);
■proto native float■LocalValue(string action, bool check_focus = true);
■■
■/**
■\brief Returns true just in frame, when action was invoked (button was pressed)
■\note if the input is limited (click, hold, doubleclick), 'Press' event is limited as well, and reacts to the limite
■@param action id of action, defined in \ref 4_World/Classes/UserActionsComponent/_constants.c
■@param check_focus if true and game is unfocused, returns 0; otherwise returns actual value
■@return true if action was invoked in that frame, false otherwise
```

```
// ---------------------------------------------
// ------ InputDeviceDisconnectWarningMenu.c - START --------------------
// ---------------------------------------------


class InputDeviceDisconnectWarningMenu extends UIScriptedMenu
{
■private ref array<int> m_DisconnectedDevices;
■private ref array<int> m_DeviceOrder = {EUAINPUT_DEVICE_CONTROLLER,EUAINPUT_DEVICE_MO
■private ref map<int,string> m_DeviceMessages;
■private bool m_CanClose;
■private bool m_Initialized;
■private string m_DevicesText;
■private string m_CaptionText;
■private string m_ConfirmationText;
■
■private WrapSpacerWidget ■m_WrapperWidget;
■private TextWidget ■■■m_CaptionWidget;
■private RichTextWidget ■■m_DeviceListWidget;
■private RichTextWidget ■■m_ActionTextWidget;
■
■void InputDeviceDisconnectWarningMenu()
■{
■■g_Game.GetMission().AddActiveInputExcludes({"gamepaddisconnect"});
■■
■■m_DisconnectedDevices = new array<int>;
■■m_CanClose = false;
■■
■■BuildMessages();
■}
■
■void ~InputDeviceDisconnectWarningMenu()
■{
■■if (g_Game)
■■{
■■■PPERequesterBank.GetRequester(PPERequester_ControllerDisconnectBlur).Stop();
■■■g_Game.DeleteTitleScreen(); // Removes any outstanding gamepad identification propmt; the gamep
■■■if (g_Game.GetMission())
■■■■g_Game.GetMission().RemoveActiveInputExcludes({"gamepaddisconnect"},true);
■■}
■}
■
■override Widget Init()
■{
■■layoutRoot = g_Game.GetWorkspace().CreateWidgets("gui/layouts/new_ui/notifications/day_z_input_d
■■m_WrapperWidget = WrapSpacerWidget.Cast(layoutRoot.FindAnyWidget("Wrapper"));
■■m_CaptionWidget = TextWidget.Cast(layoutRoot.FindAnyWidget("Caption"));
■■m_DeviceListWidget = RichTextWidget.Cast(layoutRoot.FindAnyWidget("DeviceList"));
■■m_ActionTextWidget = RichTextWidget.Cast(layoutRoot.FindAnyWidget("ActionText"));
■■
■■g_Game.GetMission().GetOnInputDeviceConnected().Insert(UpdateDisconnectedDevices);
■■g_Game.GetMission().GetOnInputDeviceDisconnected().Insert(UpdateDisconnectedDevices);
■■
■■UpdateDisconnectedDevices();
■■if (g_Game.GetUIManager() && g_Game.GetUIManager().IsDialogVisible())
■■{
■■■g_Game.GetUIManager().CloseDialog();
■■}
■■
■■PPERequesterBank.GetRequester(PPERequester_ControllerDisconnectBlur).Start();
■■
■■return layoutRoot;
■}
```

```
// ---------------------------------------------
// ------ inputManager.c - START --------------------
// ---------------------------------------------


#ifdef GAME_TEMPLATE

enum InputTrigger
{
■UP, ■■■///< call listener when button/key is released
■DOWN,■■ ■///< call listener when button/key is pressed
■PRESSED, ■■///< call listener in each tick when button/key is pressed
■VALUE■■ ■///< call listener in each tick with current value
};

class ActionManager
{
■void ActionManager(ActionManager parent);
■proto native external bool RegisterAction(string actionName);
■proto native external bool RegisterContext(string contextName);
■
■proto native external float LocalValue(string actionName);
■proto native external bool GetActionTriggered(string actionName);
■
■proto native external bool ActivateAction(string actionName, int duration = 0);■
■proto native external bool IsActionActive(string actionName);
■
■proto native external bool ActivateContext(string contextName, int duration = 0);■
■proto native external bool IsContextActive(string contextName);
■
■proto external void AddActionListener(string actionName, InputTrigger trigger,  func callback);
■
■proto native external void SetContextDebug(string contextName, bool bDebug);
■
■proto native void SetParent(ActionManager parent);
■proto native void SetDebug(bool bDebug);
};

class InputManager: ActionManager
{
■private void InputManager(ActionManager parent) {};
■private void ~InputManager() {};
■
■proto native external void ResetAction(string actionName);
■proto native void SetCursorPosition(int x, int y);
■proto native external bool RegisterActionManager(ActionManager pManager);
■proto native external bool UnregisterActionManager(ActionManager pManager);
};

#endif




// ---------------------------------------------
// ------ inputManager.c - END ---------------------
// ---------------------------------------------
// ---------------------------------------------
// ------ InputUtils.c - START --------------------
// ---------------------------------------------


class InputUtils
{
■//useful for console preset differentiation
■static const string PRESET_OLD ■■■= "#STR_UAPRESET_0";
■static const string PRESET_NEW ■■■= "#STR_UAPRESET_1";
■static const int VARIANT_OLD ■■■= 0;
■static const int VARIANT_NEW ■■■= 1;
■static int m_CurrentPresetIDConsole ■= -1;
■//
■
■static protected ref map<int, ref array<int>> m_InputActionSortingMap; //<sorting_idx,<input_ID>>
■static protected ref array<int> m_UnsortedInputActions;
■
■static const float ICON_SCALE_NORMAL■= 1.21;
■static const float ICON_SCALE_TOOLBAR■= 1.81;
■
■static string GetButtonNameFromInput(string pInputName, int pInputDeviceType)
■{
■■UAInput inp = GetUApi().GetInputByName(pInputName);
■■for (int i = 0; i < inp.AlternativeCount(); i++)
■■{
■■■inp.SelectAlternative(i);
■■■if (inp.CheckBindDevice(0, pInputDeviceType))
■■■{
■■■■return GetUApi().GetButtonName(inp.GetBindKey(0));
■■■}
■■}
■■
■■return "";
■}
■
■//! returns a map of button names, combo or not
■static map<int,ref TStringArray> GetComboButtonNamesFromInput(string pInputName, int pInputDevice
■{
■■UAInput inp = GetUApi().GetInputByName(pInputName);
■■TStringArray buttons;
■■map<int,ref TStringArray> output;//<alternativeIDX<button_array>>
■■for (int i = 0; i < inp.AlternativeCount(); i++)
■■{
■■■inp.SelectAlternative(i);
■■■if (inp.BindingCount() > 0 && inp.Binding(0) != 0 && inp.CheckBindDevice(0,pInputDeviceType))
■■■{
■■■■buttons = new TStringArray;
■■■■if (inp.IsCombo())
■■■■{
■■■■■buttons.Insert(GetUApi().GetButtonName( inp.Binding(0) ));
■■■■■
■■■■■for (int j = 1; j < inp.BindingCount(); j++)
■■■■■{
■■■■■■if (inp.Binding(j) != 0 && inp.CheckBindDevice(j,pInputDeviceType))
■■■■■■{
■■■■■■■buttons.Insert(GetUApi().GetButtonName( inp.Binding(j) ));
■■■■■■}
■■■■■}
■■■■■//return buttons;
■■■■}
■■■■else
■■■■{
```

```
// ---------------------------------------------
// ------ InspectMenuNew.c - START --------------------
// ---------------------------------------------


//--------------------------------------------------------------------------
class InspectMenuNew extends UIScriptedMenu
{
■private ItemPreviewWidget m_item_widget;
■private ItemPreviewWidget m_slot_widget;
■private int m_characterRotationX;
■private int m_characterRotationY;
■private int m_characterScaleDelta;
■private vector m_characterOrientation;

■
■void InspectMenuNew()
■{
■■
■}
■
■//--------------------------------------------------------------------------
■void ~InspectMenuNew()
■{
■■GetGame().GetDragQueue().RemoveCalls(this);
■■if (GetGame().GetMission())
■■{
■■■GetGame().GetMission().GetHud().ShowHudUI( true );
■■■GetGame().GetMission().GetHud().ShowQuickbarUI( true );
■■}
■}
■
■//--------------------------------------------------------------------------
■override Widget Init()
■{
■■layoutRoot = GetGame().GetWorkspace().CreateWidgets("gui/layouts/inventory_new/day_z_inventory_
■■
■■
■■return layoutRoot;
■}
■
■//--------------------------------------------------------------------------
■override bool OnClick(Widget w, int x, int y, int button)
■{
■■super.OnClick(w, x, y, button);
■■
■■switch (w.GetUserID())
■■{
■■■case IDC_CANCEL:
■■■■Close();
■■■■return true;
■■}
■
■■return false;
■}
■
■//--------------------------------------------------------------------------
■void SetItem(EntityAI item)
■{
■■if (item)
■■{
■■■InspectMenuNew.UpdateItemInfo(layoutRoot, item);
■
```

```
// ---------------------------------------------
// ------ IntroSceneCharacter.c - START --------------------
// ---------------------------------------------


class IntroSceneCharacter extends Managed
{
■protected int■■■m_CharacterId;
■protected string■■m_CharacterType;
■protected MenuData■■m_CharacterDta;
■protected PlayerBase■m_CharacterObj;
■protected vector■■m_CharacterPos;
■protected vector■■m_CharacterRot;
■■
■protected ref TStringArray ■m_CharGenderList■■■■= new TStringArray;
■protected ref TStringArray■m_CharShirtList■■■■■= new TStringArray; //legacy
■protected ref TStringArray ■m_CharPantsList■■■■■= new TStringArray; //legacy
■protected ref TStringArray■m_CharShoesList■■■■■= new TStringArray; //legacy
■
■protected ref map<ECharGender, ref array<string>> m_Characters = new map<ECharGender, ref array
■
■protected ECharGender■■m_CharGender;
■
■void IntroSceneCharacter()
■{
■■m_CharacterId = GameConstants.DEFAULT_CHARACTER_MENU_ID;
■}
■
■void ~IntroSceneCharacter()
■{
■■CharacterUnload();
■}
■
■bool IsDefaultCharacter()
■{
■■return m_CharacterId == GameConstants.DEFAULT_CHARACTER_MENU_ID;
■}
■
■void SetToDefaultCharacter()
■{
■■m_CharacterId = GameConstants.DEFAULT_CHARACTER_MENU_ID;
■}
■//==============================================
■// SetcharacterID
■//==============================================
■void SetCharacterID(int char_id)
■{
■■m_CharacterId = char_id;
■}
■
■//==============================================
■// GetCharacterID
■//==============================================
■int GetCharacterID()
■{
■■return m_CharacterId;
■}
■
■//==============================================
■// GetCharacterObj
■//==============================================
■PlayerBase GetCharacterObj()
■{
```

```
// ---------------------------------------------
// ------ Inventory.c - START --------------------
// ---------------------------------------------


//-----------------------------------------------------
enum InventoryCommandType
{
■MOVE,■■■■■///< generic move, may involve animations
■SYNC_MOVE,■■■■///< synchronous move. action is finished immeadiately, no animations involved
■HAND_EVENT,■■■■///< event for hands
■SWAP,■■■■■///< swap two entities (simple swap of compatible objects)
■FORCESWAP,■■■■///< Forced swap two entities. First goes to second's place, second goes "somewhe
■DESTROY,■■■■///< destroy of entity right in inventory
■REPLACE,■■■■///< replace of entity in inventory (@NOTE: hands goes through HAND_EVENT)
■USER_RESERVATION_CANCEL■///< Clear user reserved inventory space
};
enum InventoryJunctureType
{
■TAKE,■■■///< taking from ground
■SWAP,■■■///< swapping from ground
■//LOAD,■■■///< load mag from ground
};
enum InventoryMode
{
■PREDICTIVE,■■///< 'Predictive' means that the operation uses Client-Side Prediction, i.e. the action run
■LOCAL,■■■///< 'Local' operation executes from where it is run
■JUNCTURE,■■///< 'Juncture' operation is used whenever there is possibility of race condition, i.e. two pl
■SERVER,■■■///< 'Server' mode operation is required if and only if the operation runs only on server (cre
};

enum InventoryCheckContext
{
■DEFAULT,
■SYNC_CHECK,
}

enum FindInventoryReservationMode
{
■//! The original logic, finds anything depending on the parameters, item or dst required
■LEGACY,
■//! Find a reservation for the item EXCLUDING the dst, item and dst required
■ITEM,
■//! Find a reservation for the dst EXCLUDING the item, item and dst required
■DST,
■//! Find an exact reservation for item and dst, item and dst required
■EQUAL,
};

/**@class■■GameInventory
 * @brief■■script counterpart to engine's class Inventory
 **/
class GameInventory
{
■protected static int m_inventory_check_context = InventoryCheckContext.DEFAULT;■
//-----------------------------------------------------
///////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
///@{ Engine native functions

■/**
■ * @fn■■GetInventoryOwner
■ * @return■entity associated with this inventory
■ **/
```

```
// ---------------------------------------------
// ------ InventoryActionHandler.c - START --------------------
// ---------------------------------------------


//! Client only - manage set up crafting on client
class InventoryActionHandler
{
■ActionBase m_action;
■ActionTarget m_target;
■ItemBase m_mainItem;
■bool ■m_useItemInHands;

■PlayerBase m_player;
■
■bool m_isActive;
■vector m_actionStartPos;
■
■const float MIN_DISTANCE_TO_INTERRUPT = 1.0;■
■const int IAH_SINGLE_USE = 1;
■const int IAH_CONTINUOUS = 2;


■void InventoryActionHandler(PlayerBase player)
■{
■■m_player = player;
■■m_isActive = false;
■■m_action = null;
■■m_target = null;
■■m_mainItem = null;
■■m_useItemInHands = false;

■}
■
■void SetAction(ActionBase action, ItemBase target_item, ItemBase main_item )
■{
■■Object target_parent = null;
■■if(target_item)
■■{
■■■target_parent = target_item.GetHierarchyParent();
■■}
■■ActionTarget at = new ActionTarget(target_item, target_parent, -1, vector.Zero, -1);
■■
■■SetAction(action, at, main_item );
■}
■
■void SetAction(ActionBase action, ActionTarget target, ItemBase main_item )
■{
■■ActionManagerClient mngr;
■■Class.CastTo(mngr, m_player.GetActionManager());
■■
■■m_action = action;
■■m_target = target;
■■m_mainItem = main_item;
■■
■■ItemBase itemInHand = m_player.GetItemInHands();
■■m_useItemInHands = main_item == itemInHand;
■■
■■m_actionStartPos = m_player.GetPosition();
■■m_isActive = true;
■■
■■//mngr.InjectAction( action, target, main_item );
■■mngr.ForceTarget(m_target.GetObject());
```

```
// ---------------------------------------------
// ------ InventoryCombinationFlags.c - START --------------------
// ---------------------------------------------


class InventoryCombinationFlags
{
■static const int NONE ■■■■■= 0;
■static const int ADD_AS_ATTACHMENT ■■= 1;
■static const int ADD_AS_CARGO ■■■= 2;
■static const int SWAP ■■■■■= 4;
■static const int COMBINE_QUANTITY■■= 8;
■static const int CRAFT■■■■■= 16;
■static const int ACTIONS■■■■= 32;
■static const int REPAIR■■■■■= 64;
■static const int TAKE_TO_HANDS■■■= 128;
■static const int RECIPE_HANDS■■■= 256;
■static const int COMBINE_QUANTITY2■■= 512;
■static const int RECIPE_ANYWHERE■■= 1024;
■static const int PERFORM_ACTION■■■= 2048;
■static const int LOAD_CHAMBER■■■= 4096;
■static const int DETACH_MAGAZINE■■= 8192;
■static const int ATTACH_MAGAZINE■■= 16384;
■static const int SET_ACTION■■■■= 32768;
■//static const int SWAP_MAGAZINE■■■= 65536;
■
}



// ---------------------------------------------
// ------ InventoryCombinationFlags.c - END ----------------------
// ---------------------------------------------


// ---------------------------------------------
// ------ InventoryGrid.c - START --------------------
// ---------------------------------------------


// ---------------------------------------------------------
class InventoryGridController extends ScriptedWidgetEventHandler
{
■void OnItemEnter(InventoryGrid grid, Widget w, int row, int col) {}
■void OnItemLeave(InventoryGrid grid, Widget w) {}
■void OnItemDrag(InventoryGrid grid, Widget w, int row, int col) {}
■void OnItemDraggingOver(InventoryGrid grid, Widget w, int row, int col) {}
■void OnItemDrop(InventoryGrid grid, Widget w, int row, int col) {}
■void OnItemDropReceived(InventoryGrid grid, Widget w, int row, int col) {}
■void OnItemClick(InventoryGrid grid, Widget w, int row, int col) {}
■void OnItemLeftClick(InventoryGrid grid, Widget w, int row, int col) {}
■void OnItemRightClick(InventoryGrid grid, Widget w, int row, int col) {}
■void OnItemDoubleClick(InventoryGrid grid, Widget w, int row, int col) {}
■// float GetItemQuantity(InventoryGrid grid, InventoryItem item) {}
■int GetItemColor(ScriptedWidgetEventHandler grid, InventoryItem item)
■{
■■return 0;
■}
■int GetQuickbarItemColor(InventoryGrid grid, InventoryItem item) {}
■vector GetItemSize(InventoryGrid grid, InventoryItem item)
■{
■■return Vector(0, 1, 1);
■}
```

```
// ---------------------------------------------
// ------ InventoryInputUserData.c - START --------------------
// ---------------------------------------------



class InventoryInputUserData
{
■///@{ move
■static void SerializeMove(ParamsWriteContext ctx, int type, notnull InventoryLocation src, notnull Invento
■{
■■ctx.Write(INPUT_UDT_INVENTORY);
■■ctx.Write(type);
■■src.WriteToContext(ctx);
■■dst.WriteToContext(ctx);
■}

■static void SendInputUserDataMove(int type, notnull InventoryLocation src, notnull InventoryLocation dst
■{
■■if (GetGame().IsClient())
■■{
■■■syncDebugPrint("[syncinv] t=" + GetGame().GetTime() + "ms sending cmd=" + typename.EnumToStr
■■■ScriptInputUserData ctx = new ScriptInputUserData;
■■■SerializeMove(ctx, type, src, dst);
■■■ctx.Send();
■■}
■}

■static void SendServerMove(Man player, int type, notnull InventoryLocation src, notnull InventoryLocatio
■{
■■if (GetGame().IsServer())
■■{
■■■syncDebugPrint("[syncinv] server sending cmd=" + typename.EnumToString(InventoryCommandTyp
■■■ScriptInputUserData ctx = new ScriptInputUserData;
■■■SerializeMove(ctx, type, src, dst);
■■■GameInventory.ServerLocationSyncMoveEntity(player, src.GetItem(), ctx);
■■}
■}
■///@} move

■///@{ swap
■static void SerializeSwap(ParamsWriteContext ctx, notnull InventoryLocation src1, notnull InventoryLoca
■{
■■ctx.Write(INPUT_UDT_INVENTORY);
■■ctx.Write(InventoryCommandType.SWAP);
■■src1.WriteToContext(ctx);
■■src2.WriteToContext(ctx);
■■dst1.WriteToContext(ctx);
■■dst2.WriteToContext(ctx);
■■ctx.Write(skippedSwap);
■}

■static void SendInputUserDataSwap(notnull InventoryLocation src1, notnull InventoryLocation src2, notnu
■{
■■if (GetGame().IsClient())
■■{
■■■syncDebugPrint("[syncinv] t=" + GetGame().GetTime() + "ms sending cmd=SWAP src1=" + Inventory
■■■ScriptInputUserData ctx = new ScriptInputUserData;
■■■SerializeSwap(ctx, src1, src2, dst1, dst2, skippedSwap);
■■■ctx.Send();
■■}
■}
```

```
// ---------------------------------------------
// ------ InventoryItem.c - START --------------------
// ---------------------------------------------


// #include "Scripts\Entities\ItemBase.c"

typedef ItemBase InventoryItemBase;
typedef ItemBase InventoryItemSuper;




// ---------------------------------------------
// ------ InventoryItem.c - END ----------------------
// ---------------------------------------------


// ---------------------------------------------
// ------ InventoryItemType.c - START --------------------
// ---------------------------------------------


class CarWheelType extends InventoryItemType
{
};

class ClothingType extends InventoryItemType
{
};

class ItemWatchType extends InventoryItemType
{
};

class ItemRadioType extends InventoryItemType
{
};

class ItemTransmitterType extends ClothingType
{
};

class ItemMegaphoneType extends InventoryItemType
{
};

class ItemCompassType extends InventoryItemType
{
};

class ItemGPSType extends InventoryItemType
{
};

class ItemMapType extends InventoryItemType
{
};

class ItemBookType extends InventoryItemType
{
};
```

```
// ---------------------------------------------
// ------ InventoryLocation.c - START --------------------
// ---------------------------------------------


//@NOTE: DO NOT EDIT! enum values are overwritten from c++
/// types of Inventory Location
enum InventoryLocationType
{
■UNKNOWN, ///< unknown, usually freshly created object
■GROUND, /// < ground
■ATTACHMENT, ///< attachment of another entity
■CARGO, ///< cargo of another entity
■HANDS, ///< hands of another entity
■PROXYCARGO, ///< cargo of a large object (building,...)
};

//@NOTE: DO NOT EDIT! enum values are overwritten from c++
/// flags for searching locations in inventory
enum FindInventoryLocationType
{
■ATTACHMENT, ■///< ■ATT
■CARGO,■■■///< ■CGO
■HANDS, ■■■///< ■HND
■PROXYCARGO, ■///< ■PXY
■ANY_CARGO, ■■///< ■CGO | PXY
■ANY, ■■■///< ■ATT | CGO | PXY | HND
■NO_SLOT_AUTO_ASSIGN ///<■skips auto-assign test
};

//! InventoryLocation
class InventoryLocation
{
■/**
■ * @fn■■■IsValid
■ * @brief■■verify current set inventory location
■ * @return■true if valid, false otherwise
■ **/
■proto native bool IsValid ();
■/**
■ * @fn■■GetType
■ * @brief■returns type of InventoryLocation
■ *
■ * @see■InventoryLocationType for known types of inventory locations
■ **/
■proto native int GetType ();
■/**
■ * @fn■■GetParent
■ * @brief■returns parent of current inventory location
■ *
■ * Parent entity can be null if item is on ground.
■ *
■ * @return parent entity, null otherwise
■ **/
■proto native EntityAI GetParent ();
■/**
■ * @fn■■GetItem
■ * @brief■returns item of current inventory location
■ *
■ * Item can be null if and only if this is a query for location
■ * of item that is about to be created (new spawn).
■ *
■ * @return■item
```

```
// ---------------------------------------------
// ------ InventoryMenu.c - START --------------------
// ---------------------------------------------


enum ScreenWidthType
{
■NARROW,
■MEDIUM,
■WIDE
}

class InventoryMenu extends UIScriptedMenu
{
■ref Inventory■■■■■■m_Inventory;
■private ref ContextMenu■■■■m_context_menu;
■protected bool■■■■■■m_IsOpened;
■protected bool■■■■■■m_OnlyFirstTime;
■protected int  ■■■■■■m_LastDisplayLanguage;
■
■protected static ScreenWidthType■m_WidthType;
■protected static int■■■■m_Width;
■protected static int■■■■m_Height;
■
■void InventoryMenu()
■{
■■CheckWidth();
■■m_Inventory = new Inventory(null);
■■m_Inventory.Reset();
■■m_Inventory.UpdateInterval();
■■m_context_menu = new ContextMenu;
■■m_LastDisplayLanguage = g_Game.GetCurrentDisplayLanguageIdx();
■}
■
■override Widget Init()
■{
■■m_Inventory.Init();
■■m_context_menu.Init(layoutRoot);
■■layoutRoot = m_Inventory.GetMainWidget();
■■return layoutRoot;
■}
■
■void CheckWidth()
■{
■■GetScreenSize( m_Width, m_Height );
■■
■■if( m_Height > 0 )
■■{
■■■float ratio = m_Width / m_Height;
■■■if( ratio > 1.75 )
■■■■m_WidthType = ScreenWidthType.WIDE;
■■■else if( ratio > 1.5 )
■■■■m_WidthType = ScreenWidthType.MEDIUM;
■■■else
■■■■m_WidthType = ScreenWidthType.NARROW;
■■}
■}
■
■static ScreenWidthType GetWidthType()
■{
■■return m_WidthType;
■}
■
```

```
// ---------------------------------------------
// ------ InventoryQuickbar.c - START --------------------
// ---------------------------------------------


class InventoryQuickbar extends InventoryGridController
{
■
■protected ref TItemsMap■m_Items;
■protected InventoryGrid■m_Grid;
■protected int■■■m_DraggingIndex;
■
■void InventoryQuickbar(Widget quickbarGridWidget)
■{
■■m_DraggingIndex = INDEX_NOT_FOUND;
■■m_Items = new TItemsMap;
■■UpdateItems( quickbarGridWidget );
■}

■void ~InventoryQuickbar()
■{
■}
■
■void UpdateItems( Widget quickbarGridWidget )
■{
■■if( !quickbarGridWidget )
■■{
■■■return;
■■}
■■
■■PlayerBase player = PlayerBase.Cast( GetGame().GetPlayer() );
■■if ( !player )
■■■return;
■■
■■int i;

■■// create grid and align it to center
■■if( !m_Grid )
■■{
■■■quickbarGridWidget.GetScript( m_Grid );
■■■m_Grid.SetController( this );
■■■m_Grid.GenerateQuickBarBackgroundTiles( 10 );
■■}
■■
■■m_Items.Clear();
■■InventoryItem item;
■■
■■if( m_Grid )
■■{
■■■if( m_Grid.GetGridSize() != player.GetQuickBarSize() )
■■■{
■■■■m_Grid.SetGridSize( player.GetQuickBarSize() );
■■■}
■■}
■■
■■for( i = 0; i < m_Grid.GetGridSize(); i++)
■■{
■■■item = InventoryItem.Cast( player.GetQuickBarEntity(i) );
■■■if( item )
■■■{
■■■■m_Items.Set( item, Vector(i, 1, 1) );
■■■}
■■}
```

```
// ---------------------------------------------
// ------ InventorySlots.c - START --------------------
// ---------------------------------------------



/**@class■■InventorySlots
 * @brief■■provides access to slot configuration
 **/
class InventorySlots
{■
■/**@NOTE: engine pre-populates this class with first 32 slots from CfgSlots (uppercased)
■ **/
■
■/** \name C++ constants
■■Constants filled in from C++
■*/
■//@{
■//! Amount of pre-populated slots (32)
■const int COUNT;
■//! Invalid slot (-1)
■const int INVALID;
■//@}
■
■#ifdef DIAG_DEVELOPER
■private void InventorySlots() {}
■private void ~InventorySlots() {}
■#else
■void InventorySlots() {}
■void ~InventorySlots() {}
■#endif

■/**@fn■■GetSlotIdFromString
■ * @brief■converts string to slot_id
■ * @param[in]■slot_name■\p■slot name to be find.
■ * @return■■slot id or InventorySlots.INVALID
■ *
■ * @example
■ *■■int slot = InventorySlots.GetSlotIdFromString("FireWood");
■ *■■if (slot != InventorySlots.INVALID)
■ *
■ * @NOTE the example looks in the DZ/data/config.cpp and searches for class entry Slot_##slot_name
■ *■■class Slot_Firewood { whatever };
■ *
■ * i.e. it does NOT look at name= or displayName= attributes of the entry!
■ **/
■static proto native int GetSlotIdFromString(string slot_name);
■/**@fn■■GetSlotName
■ * @brief■converts slot_id to string
■ * @param[in]■slot_id■\p■slot id to be find.
■ * @return■■string or null string
■ **/
■static proto native owned string GetSlotName(int id);
■/**@fn■■GetSlotDisplayName
■ * @brief■converts slot_id to string
■ * @param[in]■slot_id■\p■slot id to be find.
■ * @return■■string or null string
■ **/
■static proto native owned string GetSlotDisplayName(int id);
■/**@fn■■IsSlotIdValid
■ * @brief■verifies existence of the slot id
■ * @return■true if slot valid
■ **/
```

```
// ---------------------------------------------
// ------ Inventory_Base.c - START --------------------
// ---------------------------------------------


class Inventory_Base extends ItemBase
{
}


// ---------------------------------------------
// ------ Inventory_Base.c - END ----------------------
// ---------------------------------------------


// ---------------------------------------------
// ------ InviteMenu.c - START --------------------
// ---------------------------------------------


class InviteMenu extends UIScriptedMenu
{■
■private TextWidget■■m_LogoutTimetext;
■private TextWidget■■m_Info;
■private int m_iTime;

■void InviteMenu()
■{
■■m_iTime = 15;
■}
■
■override Widget Init()
■{
■■layoutRoot = GetGame().GetWorkspace().CreateWidgets("gui/layouts/day_z_invite_dialog.layout");
■■
■■m_LogoutTimetext = TextWidget.Cast(layoutRoot.FindAnyWidget("logoutTimeText"));
■■m_Info = TextWidget.Cast(layoutRoot.FindAnyWidget("txtInfo"));
■■m_LogoutTimetext.SetText(m_iTime.ToString());
■■
■■layoutRoot.FindAnyWidget("toolbar_bg").Show(true);
■■RichTextWidget toolbar_b = RichTextWidget.Cast(layoutRoot.FindAnyWidget("BackIcon"));
■■toolbar_b.SetText(InputUtils.GetRichtextButtonIconFromInputAction("UAUIBack", "", EUAINPUT_DEV
■■
■■// player should sit down if possible
■■PlayerBase player = PlayerBase.Cast(GetGame().GetPlayer());
■■if (player && player.GetEmoteManager() && !player.IsRestrained() && !player.IsUnconscious())
■■{
■■■player.GetEmoteManager().CreateEmoteCBFromMenu(EmoteConstants.ID_EMOTE_SITA);
■■■player.GetEmoteManager().GetEmoteLauncher().SetForced(EmoteLauncher.FORCE_DIFFERENT);
■■}
■■
■■GetGame().GetCallQueue(CALL_CATEGORY_SYSTEM).CallLater(UpdateTime, 1000, true);
■■return layoutRoot;
■}
■
■override void OnShow()
■{
■■super.OnShow();
■}

■override void Update(float timeslice)
■{
■■if (GetUApi().GetInputByID(UAUIBack).LocalPress())
```

```
// ---------------------------------------------
// ------ IodineTincture.c - START --------------------
// ---------------------------------------------


class IodineTincture : Edible_Base
{
■
■override void InitItemVariables()
■{
■■super.InitItemVariables();
■■can_this_be_combined = true;
■}
■
■override void SetActions()
■{
■■super.SetActions();
■■
■■AddAction(ActionDisinfectTarget);
■■AddAction(ActionDisinfectSelf);;
■■AddAction(ActionWashHandsItem);
■■//AddAction(ActionForceDrinkDisinfectant);
■■//AddAction(ActionDrinkDisinfectant);
■}
■
■override float GetDisinfectQuantity(int system = 0, Param param1 = null)
■{
■■return (GetQuantityMax() * 0.05);
■}
}


// ---------------------------------------------
// ------ IodineTincture.c - END ----------------------
// ---------------------------------------------


// ---------------------------------------------
// ------ IsBoxCollidingGeometryProxyClasses.c - START --------------------
// ---------------------------------------------


//! Class that holds parameters to feed into CGame.IsBoxCollidingGeometryProxy
class BoxCollidingParams
{
■/**
■\brief Set the parameters
■    \param center \p vector Center of box
■■\param orientation \p vector Orientation of box
■■\param edgeLength \p vector EdgeLength of box
■■\param primaryType \p vector Primary geometry to check
■■\param secondaryType \p vector Secondary geometry to check (if no backup is needed, fill in the same
■■\param fullComponentInfo \p vector When true gives information about every component and proxy in b
■*/
■proto native void SetParams(vector center, vector orientation, vector edgeLength, ObjIntersect primaryTy
}

//! ComponentInfo for BoxCollidingResult
class ComponentInfo
{
■int component;■■■//!< Component index
■vector componentMin; ■//!< Component min in world space
■vector componentMax; ■//!< Component max in world space
```

```
// ---------------------------------------------
// ------ ItemActionsWidget.c - START --------------------
// ---------------------------------------------


class ItemActionsWidget extends ScriptedWidgetEventHandler
{
■protected PlayerBase ■■■■m_Player;
■protected EntityAI ■■■■■m_EntityInHands;
■protected ActionBase■■■■m_Interact;
■protected ActionBase■■■■m_ContinuousInteract;
■protected ActionBase■■■■m_Single;
■protected ActionBase■■■■m_Continuous;
■protected ActionManagerClient■ ■m_AM;
■protected IngameHud■■■■■m_Hud;
■
■protected UAIDWrapper ■■■■m_UseActionWrapper;

■protected int■■■■■■m_InteractActionsNum;
■protected int■■■■■■m_ContinuousInteractActionsNum;
■protected int■■■■■■m_ItemActionsNum;
■protected int■■■■■■m_ContinuousItemActionsNum;
■protected bool ■■■■■■m_HealthEnabled;
■protected bool■■■■■■m_QuantityEnabled;
■
■protected ref WidgetFadeTimer■■m_FadeTimer;
■protected bool■■■■■■m_Faded;
■protected bool ■■■■■■m_Hidden;

■protected Widget■■■■■m_Root;
■protected Widget ■■■■■m_ItemLeft;

■//! widget width
■protected float m_MaxWidthChild;
■protected float m_RootWidth;
■protected float m_RootHeight;
■
■void ItemActionsWidget()
■{
■■m_Interact■■■■= null;
■■m_ContinuousInteract ■= null;
■■m_Single ■■■■= null;
■■m_Continuous ■■■= null;
■■m_AM ■■■■■= null;

■■m_FadeTimer ■■■= new WidgetFadeTimer;
■■m_Faded ■■■■= true;
■■
■■m_HealthEnabled ■■= true;
■■m_QuantityEnabled ■■= true;
■■
■■m_Hud■■■■■= GetHud();
■■m_Hidden ■■■■= true;
■■
■■m_UseActionWrapper ■■= GetUApi().GetInputByID(UAAction).GetPersistentWrapper();
■■
■■GetGame().GetUpdateQueue(CALL_CATEGORY_GUI).Insert(Update);
■■GetGame().GetMission().GetOnInputPresetChanged().Insert(OnInputPresetChanged);
■■GetGame().GetMission().GetOnInputDeviceChanged().Insert(OnInputDeviceChanged);
■}
■
■void ~ItemActionsWidget()
■{
```

```
// ---------------------------------------------
// ------ ItemBase.c - START --------------------
// ---------------------------------------------


typedef ItemBase Inventory_Base;

class DummyItem extends ItemBase
{
■override bool CanPutAsAttachment(EntityAI parent)
■{
■■return true;
■}
};

//const bool QUANTITY_DEBUG_REMOVE_ME = false;

class ItemBase extends InventoryItem
{
■static ref map<typename, ref TInputActionMap> m_ItemTypeActionsMap = new map<typename, ref TInp
■TInputActionMap m_InputActionMap;
■bool■m_ActionsInitialize;
■
■static int■m_DebugActionsMask;
■bool■■m_RecipesInitialized;
■// ============================================
■// Variable Manipulation System
■// ============================================
■//ref map<string,string> ■m_ItemVarsString;
■
■int■■m_VariablesMask;//this holds information about which vars have been changed from their default v
■// Quantity
■
■float ■m_VarQuantity;
■float ■m_VarQuantityPrev;//for client to know quantity changed during synchronization
■int■■m_VarQuantityInit;
■int■■m_VarQuantityMin;
■int■■m_VarQuantityMax;
■int■■m_Count;
■float■m_VarStackMax;
■float■m_StoreLoadedQuantity = float.LOWEST;
■// Temperature
■float ■m_VarTemperature;
■float ■m_VarTemperatureInit;
■float ■m_VarTemperatureMin;
■float ■m_VarTemperatureMax;
■// Wet
■float ■m_VarWet;
■float ■m_VarWetPrev;//for client to know wetness changed during synchronization
■float ■m_VarWetInit;
■float ■m_VarWetMin;
■float ■m_VarWetMax;
■// Cleanness
■int■■m_Cleanness;
■int■■m_CleannessInit;
■int■■m_CleannessMin;
■int■■m_CleannessMax;
■// impact sounds
■bool■m_WantPlayImpactSound;
■bool■m_CanPlayImpactSound = true;
■float■m_ImpactSpeed;
■int■■m_ImpactSoundSurfaceHash;
■//
```

```
// ---------------------------------------------
// ------ ItemBook.c - START --------------------
// ---------------------------------------------


class ItemBook extends InventoryItemSuper
{
■override event bool OnUseFromInventory(Man owner)
■{
■■return false;
■}
■
■//===================================================================
■// IGNITION ACTION
■//===================================================================
■override bool HasFlammableMaterial()
■{
■■return true;
■}
■
■override bool CanBeIgnitedBy( EntityAI igniter = NULL )
■{
■■if ( GetHierarchyParent() ) return false;
■■
■■return true;
■}
■
■override bool CanIgniteItem( EntityAI ignite_target = NULL )
■{
■■return false;
■}
■
■override void OnIgnitedTarget( EntityAI ignited_item )
■{
■}
■
■override void OnIgnitedThis( EntityAI fire_source )
■{
■■Fireplace.IgniteEntityAsFireplace( this, fire_source );
■}

■override bool IsThisIgnitionSuccessful( EntityAI item_source = NULL )
■{
■■return Fireplace.CanIgniteEntityAsFireplace( this );
■}■
}


// ---------------------------------------------
// ------ ItemBook.c - END ----------------------
// ---------------------------------------------


// ---------------------------------------------
// ------ ItemDropWarningMenu.c - START --------------------
// ---------------------------------------------


class WarningMenuBase extends UIScriptedMenu
{
■void WarningMenuBase()
■{
■■GetGame().GetMission().AddActiveInputExcludes({"menu"});
```

```
// ---------------------------------------------
// ------ ItemManager.c - START --------------------
// ---------------------------------------------


class ItemManager
{
■private ref static ItemManager■■m_Instance;
■protected bool■■■■■■m_IsDragging;
■protected EntityAI■■■■■m_HoveredItem;
■protected bool■■■■■■m_SlotInfoShown;
■protected EntityAI■■■■■m_DraggedItem;
■protected Icon■■■■■■m_DraggedIcon;
■protected ref Widget■■■■m_TooltipWidget;
■protected ref Widget■■■■m_TooltipSlotWidget;
■protected ref Widget■■■■m_TooltipCategoryWidget;
■protected ItemPreviewWidget■■■m_ItemPreviewWidget;
■protected Widget■■■■■m_RootWidget;
■protected ref map<string, bool>■■m_DefautOpenStates;
■protected ref map<string, bool>■■m_DefautHeaderOpenStates;
 ■protected int■■■■■■m_HandsDefaultOpenState;
■protected ref Timer■■■■■m_ToolTipTimer;
■protected ref Timer■■■■■m_TooltipSlotTimer;

■protected EntityAI■■■■■m_SelectedItem;
■protected Container■■■■■m_SelectedContainer;
■protected Widget■■■■■m_SelectedWidget;
■protected SlotsIcon■■■■■m_SelectedIcon;
■
■protected HandsPreview■■■■m_HandsPreview;
■
■protected bool■■■■■■m_ItemMicromanagmentMode;
■
■protected Widget■■■■■m_LeftDropzone;
■protected Widget■■■■■m_CenterDropzone;
■protected Widget■■■■■m_RightDropzone;
■
■protected int ■■■■■■m_TooltipPosX;
■protected int■ ■■■■■m_TooltipPosY;
■protected Widget ■■■■■m_TooltipSourceWidget; //stored here for tooltip position updates
■
■#ifndef PLATFORM_CONSOLE
■protected const float TOOLTIP_DELAY = 0.25; // in seconds
■#else
■protected const float TOOLTIP_DELAY = 1.5; // in seconds
■#endif
■
■void ItemManager( Widget root )
■{
■■m_Instance■■■■■= this;
■■m_RootWidget■■■■= root;
■■m_DefautOpenStates■■■= new map<string, bool>;
■■m_DefautHeaderOpenStates■= new map<string, bool>;
■■m_SlotInfoShown■■■■= false;
■■
■■#ifdef PLATFORM_CONSOLE
■■■m_TooltipWidget■■■= GetGame().GetWorkspace().CreateWidgets("gui/layouts/inventory_new/day_
■■■m_TooltipSlotWidget■■= GetGame().GetWorkspace().CreateWidgets("gui/layouts/inventory_new/da
■■#else
■■■m_TooltipWidget■■■= GetGame().GetWorkspace().CreateWidgets("gui/layouts/inventory_new/day_
■■■m_TooltipSlotWidget■■= GetGame().GetWorkspace().CreateWidgets("gui/layouts/inventory_new/da
■■#endif
■■m_TooltipWidget.Show( false );
```

```
// ---------------------------------------------
// ------ ItemOptics.c - START --------------------
// ---------------------------------------------


class ItemOptics extends InventoryItemSuper
{
■bool ■■■■m_data_set;
■bool ■■■■m_allowsDOF; //true if optics DOES NOT have magnification (FOV >= DZPLAYER_CAMERA
■bool ■■■■m_reddot_displayed
■bool ■■■■m_isNVOptic = false;
■int ■■■■m_CurrentOpticMode; //generic optic mode, currently used for NV optics only (could be expand
■int ■■■■m_CurrentOpticModeLocal; //local mirror for sync purposes;
■int ■■■■m_reddot_index;
■float ■■■■m_blur_float;
■float ■■■■m_nearplane_override; //override value for DayZPlayerCameraOptics only!
■string ■■■■m_optic_sight_texture;
■string ■■■■m_optic_sight_material;
■string ■■■■m_2D_preload_type;
■ref array<float> ■m_mask_array;
■ref array<float> ■m_lens_array;
■ref array<float> ■m_OpticsDOFProperties;
■■
■void ItemOptics()
■{
■■m_mask_array = new array<float>;
■■m_lens_array = new array<float>;
■■m_OpticsDOFProperties = new array<float>;
■■
■■InitReddotData();
■■InitOpticsPPInfo();
■■InitCameraOverrideProperties();
■■InitOpticsDOFProperties(m_OpticsDOFProperties);
■■Init2DPreloadType();
■■InitOpticMode();
■■
■■m_CurrentOpticModeLocal = -1;
■■RegisterNetSyncVariableInt( "m_CurrentOpticMode", 0, 63 );
■}
■
■/**@fn■■EnterOptics
■ * @brief■switches to optics mode if possible
■ * @return true if success, false otherwise
■ **/
■proto native bool EnterOptics ();
■
■/**@fn■■IsInOptics
■ * @brief■is weapon in optics mode or not
■ * @return true if in optics mode, false otherwise
■ **/
■proto native bool IsInOptics ();
■
■/**@fn■■ExitOptics
■ * @brief■switches out of optics mode (if possible)
■ * @return true if success, false otherwise
■ **/
■proto native bool ExitOptics ();

■/**@fn■■HasWeaponIronsightsOverride
■ * @brief■is weapon in optics mode or not
■ * @return true if optics has defined override optics info for weapon
■ **/
■proto native bool HasWeaponIronsightsOverride ();
```

```
// ---------------------------------------------
// ------ ItemsCounter.c - START --------------------
// ---------------------------------------------


// ----------------------------------------------------------
class ItemsCounter : ScriptedWidgetEventHandler
{
■bool NumberOfItems;
■
■void OnWidgetScriptInit(Widget w)
■{
   Widget child = w.GetChildren();

   while (child)
   {
      child = child.GetSibling();
      NumberOfItems++;
   }
■}
};


// ---------------------------------------------
// ------ ItemsCounter.c - END ----------------------
// ---------------------------------------------


// ---------------------------------------------
// ------ Izh18.c - START --------------------
// ---------------------------------------------


/**@class■■Izh18_Base
 * @brief■■base for Izh18
 * @NOTE■■name copies config base class
 **/
class Izh18_Base extends RifleSingleShot_Base
{
■void Izh18_Base ()
■{
■}
■
■override RecoilBase SpawnRecoilObject()
■{
■■return new Izh18Recoil(this);
■}
■■■■■
■//Debug menu Spawn Ground Special
■override void OnDebugSpawn()
■{
■■super.OnDebugSpawn();
■■
■■EntityAI entity;
■■if ( Class.CastTo(entity, this) )
■■{
■■■entity.SpawnEntityOnGroundPos("Ammo_762x39", entity.GetPosition());
■■}
■}
};
```

```
// ---------------------------------------------
// ------ Izh18Recoil.c - START --------------------
// ---------------------------------------------


class Izh18Recoil: RecoilBase
{
■override void Init()
■{
■■vector point_1;
■■vector point_2;
■■vector point_3;
■■vector point_4;
■■point_1[0] = m_Player.GetRandomGeneratorSyncManager().GetRandomInRange(RandomGeneratorS
■■point_2[0] = m_Player.GetRandomGeneratorSyncManager().GetRandomInRange(RandomGeneratorS
■■point_3[0] = m_Player.GetRandomGeneratorSyncManager().GetRandomInRange(RandomGeneratorS
■■point_4[0] = m_Player.GetRandomGeneratorSyncManager().GetRandomInRange(RandomGeneratorS
■■m_HandsCurvePoints.Insert(point_1);//forms a 2 dimensional spline(z is ignored)
■■m_HandsCurvePoints.Insert(point_2);
■■m_HandsCurvePoints.Insert(point_3);
■■m_HandsCurvePoints.Insert(point_4);
■■m_HandsCurvePoints.Insert("0 0 0");
■■m_HandsOffsetRelativeTime = 0.125;
■■
■■m_MouseOffsetRangeMin = 80;//in degrees min
■■m_MouseOffsetRangeMax = 100;//in degrees max
■■m_MouseOffsetDistance = 1.4;//how far should the mouse travel
■■m_MouseOffsetRelativeTime = 0.0625;//[0..1] a time it takes to move the mouse the required distance
■
■■m_CamOffsetDistance = 0.02;
■■m_CamOffsetRelativeTime = 0.125;
■}
}


// ---------------------------------------------
// ------ Izh18Recoil.c - END ----------------------
// ---------------------------------------------


// ---------------------------------------------
// ------ Izh18SawedOffRecoil.c - START --------------------
// ---------------------------------------------


class Izh18SawedOffRecoil: RecoilBase
{
■override void Init()
■{
■■vector point_1;
■■vector point_2;
■■vector point_3;
■■vector point_4;
■■point_1[0] = m_Player.GetRandomGeneratorSyncManager().GetRandomInRange(RandomGeneratorS
■■point_2[0] = m_Player.GetRandomGeneratorSyncManager().GetRandomInRange(RandomGeneratorS
■■point_3[0] = m_Player.GetRandomGeneratorSyncManager().GetRandomInRange(RandomGeneratorS
■■point_4[0] = m_Player.GetRandomGeneratorSyncManager().GetRandomInRange(RandomGeneratorS
■■m_HandsCurvePoints.Insert(point_1);//forms a 2 dimensional spline(z is ignored)
■■m_HandsCurvePoints.Insert(point_2);
■■m_HandsCurvePoints.Insert(point_3);
■■m_HandsCurvePoints.Insert(point_4);
■■m_HandsCurvePoints.Insert("0 0 0");
■■m_HandsOffsetRelativeTime = 0.125;
```

```
// ---------------------------------------------
// ------ Izh18Shotgun.c - START --------------------
// ---------------------------------------------


/**@class■■Izh18_Base
 * @brief■■base for Izh18Shotgun
 * @NOTE■■name copies config base class
 **/
class Izh18Shotgun_Base extends RifleSingleShot_Base
{
■void Izh18Shotgun_Base ()
■{
■}
■
■override RecoilBase SpawnRecoilObject()
■{
■■return new Izh18ShotgunRecoil(this);
■}
■■■■■
■//Debug menu Spawn Ground Special
■override void OnDebugSpawn()
■{
■■super.OnDebugSpawn();

■    SpawnEntityOnGroundPos("Ammo_12gaPellets", GetPosition());
■    SpawnEntityOnGroundPos("Ammo_12gaSlug", GetPosition());
■    SpawnEntityOnGroundPos("Ammo_12gaRubberSlug", GetPosition());
■}
};



// ---------------------------------------------
// ------ Izh18Shotgun.c - END ----------------------
// ---------------------------------------------


// ---------------------------------------------
// ------ Izh18ShotgunRecoil.c - START --------------------
// ---------------------------------------------


class Izh18ShotgunRecoil: RecoilBase
{
■override void Init()
■{
■■vector point_1;
■■vector point_2;
■■vector point_3;
■■vector point_4;
■■point_1[0] = m_Player.GetRandomGeneratorSyncManager().GetRandomInRange(RandomGeneratorS
■■point_2[0] = m_Player.GetRandomGeneratorSyncManager().GetRandomInRange(RandomGeneratorS
■■point_3[0] = m_Player.GetRandomGeneratorSyncManager().GetRandomInRange(RandomGeneratorS
■■point_4[0] = m_Player.GetRandomGeneratorSyncManager().GetRandomInRange(RandomGeneratorS
■■m_HandsCurvePoints.Insert(point_1);//forms a 2 dimensional spline(z is ignored)
■■m_HandsCurvePoints.Insert(point_2);
■■m_HandsCurvePoints.Insert(point_3);
■■m_HandsCurvePoints.Insert(point_4);
■■m_HandsCurvePoints.Insert("0 0 0");
■■m_HandsOffsetRelativeTime = 0.125;
■■
■■m_MouseOffsetRangeMin = 70;//in degrees min
```

```
// ---------------------------------------------
// ------ IZH43.c - START --------------------
// ---------------------------------------------


class Izh43Shotgun_Base : DoubleBarrel_Base
{
■override RecoilBase SpawnRecoilObject()
■{
■■return new Izh43Recoil(this);
■}
■■■■
■//Debug menu Spawn Ground Special
■override void OnDebugSpawn()
■{
■■super.OnDebugSpawn();
■■
■■EntityAI entity;
■■if ( Class.CastTo(entity, this) )
■■{
■■■entity.SpawnEntityOnGroundPos("Ammo_12gaPellets", entity.GetPosition());
■■■entity.SpawnEntityOnGroundPos("Ammo_12gaSlug", entity.GetPosition());
■■■entity.SpawnEntityOnGroundPos("Ammo_12gaRubberSlug", entity.GetPosition());
■■}
■}
};

class Izh43Shotgun: Izh43Shotgun_Base {};
class SawedoffIzh43Shotgun: Izh43Shotgun_Base {};




// ---------------------------------------------
// ------ IZH43.c - END ----------------------
// ---------------------------------------------


// ---------------------------------------------
// ------ Izh43Recoil.c - START --------------------
// ---------------------------------------------


class Izh43Recoil: RecoilBase
{
■override void Init()
■{
■■vector point_1;
■■vector point_2;
■■vector point_3;
■■vector point_4;
■■point_1[0] = m_Player.GetRandomGeneratorSyncManager().GetRandomInRange(RandomGeneratorS
■■point_2[0] = m_Player.GetRandomGeneratorSyncManager().GetRandomInRange(RandomGeneratorS
■■point_3[0] = m_Player.GetRandomGeneratorSyncManager().GetRandomInRange(RandomGeneratorS
■■point_4[0] = m_Player.GetRandomGeneratorSyncManager().GetRandomInRange(RandomGeneratorS
■■m_HandsCurvePoints.Insert(point_1);//forms a 2 dimensional spline(z is ignored)
■■m_HandsCurvePoints.Insert(point_2);
■■m_HandsCurvePoints.Insert(point_3);
■■m_HandsCurvePoints.Insert(point_4);
■■m_HandsCurvePoints.Insert("0 0 0");
■■m_HandsOffsetRelativeTime = 0.125;
■■
■■m_MouseOffsetRangeMin = 70;//in degrees min
```

```
// ---------------------------------------------
// ------ Jeans_ColorBase.c - START --------------------
// ---------------------------------------------


class Jeans_ColorBase extends Clothing
{
■override void SetActions()
■{
■■super.SetActions();
■■AddAction(ActionWringClothes);
■}
};
class Jeans_Black extends Jeans_ColorBase {};
class Jeans_Blue extends Jeans_ColorBase {};
class Jeans_Brown extends Jeans_ColorBase {};
class Jeans_Green extends Jeans_ColorBase {};
class Jeans_Grey extends Jeans_ColorBase {};
class Jeans_BlueDark extends Jeans_ColorBase {};

class ShortJeans_ColorBase extends Clothing
{
■override void SetActions()
■{
■■super.SetActions();
■■AddAction(ActionWringClothes);
■}
};
class ShortJeans_Black extends ShortJeans_ColorBase {};
class ShortJeans_Blue extends ShortJeans_ColorBase {};
class ShortJeans_Brown extends ShortJeans_ColorBase {};
class ShortJeans_Darkblue extends ShortJeans_ColorBase {};
class ShortJeans_Green extends ShortJeans_ColorBase {};
class ShortJeans_Red extends ShortJeans_ColorBase {};


// ---------------------------------------------
// ------ Jeans_ColorBase.c - END ----------------------
// ---------------------------------------------


// ---------------------------------------------
// ------ JoggingShoes_ColorBase.c - START --------------------
// ---------------------------------------------


class JoggingShoes_ColorBase extends Clothing {};
class JoggingShoes_Black extends JoggingShoes_ColorBase {};
class JoggingShoes_Blue extends JoggingShoes_ColorBase {};
class JoggingShoes_Red extends JoggingShoes_ColorBase {};
class JoggingShoes_Violet extends JoggingShoes_ColorBase {};
class JoggingShoes_White extends JoggingShoes_ColorBase {};


// ---------------------------------------------
// ------ JoggingShoes_ColorBase.c - END ----------------------
// ---------------------------------------------
// ---------------------------------------------
// ------ JsonApi.c - START --------------------
// ---------------------------------------------



/** @file */

// -----------------------------------------------------------------------
// error codes for handle processing
// defined in C++
enum EJsonApiError
{
■ETJSON_UNKNOWN,■■■// invalid code

■ETJSON_OK,■■■■// all fine
■ETJSON_COMMSEND,■■// error during send
■ETJSON_COMMRECV,■■// error during receive
■ETJSON_PARSERERROR,■■// error during parsing
■ETJSON_PACKNOSTART,■■// error - cannot start packing (invalid state)
■ETJSON_TIMEOUT,■■■// failed to send/ store handle due to timeout
■ETJSON_NOBUFFERS,■■// not enough buffers available
■ETJSON_FAILFILELOAD,■// failed to load file
■ETJSON_FAILFILESAVE,■// failed to save file
■ETJSON_NOTARRAY,■■// object is not array (ie. attempt to provide different or none object as array)
};


// -----------------------------------------------------------------------
// JsonApi Handle is container encapsulating real JSON data being sent or recieved via. RESTful/ other se
class JsonApiHandle
{
■private void JsonApiHandle() {}
■private void ~JsonApiHandle() {}
■
■/**
■\brief Length of JSON
■*/
■proto native int Size();

■/**
■\brief Return as string
■*/
■proto native owned string AsString();

■/**
■\brief Invalidate handle and schedule removal
■*/
■proto native void Invalidate();

};


// -----------------------------------------------------------------------
// parent class for handling JsonApi Struct objects
//
//
class JsonApi
{
■private void JsonApi() {}
■private void ~JsonApi() {}

■/**
```

```
// ---------------------------------------------
// ------ JsonApiStruct.c - START --------------------
// ---------------------------------------------



/** @file */

// -----------------------------------------------------------------------
// object which allow to parse upon generic JSON structure and format it back
//
//
class JsonApiStruct : Managed
{

■void JsonApiStruct()
■{
■}

■void ~JsonApiStruct()
■{
■}

■/**
■\brief Event when expand (unpack) process starts
■*/
■void OnExpand()
■{
■}

■/**
■\brief Event when pack starts - you will pack your stuff here
■*/
■void OnPack()
■{
■■Print( "OnPack() ");
■}

■/**
■\brief Verification event after successfull JSON packing
■*/
■void OnBufferReady()
■{
■}

■/**
■\brief Event called when pending store operation is finished - callback from JsonApiHandle before handle
■*/
■void OnSuccess( int errorCode )
■{
■■// errorCode is EJsonApiError
■}

■/**
■\brief Event called when pending store operation is finished - callback from JsonApiHandle before handle
■*/
■void OnError( int errorCode )
■{
■■// errorCode is EJsonApiError
■}
■
■/**
■\brief Called when parsing object
```

```
// ---------------------------------------------
// ------ JsonDataContaminatedArea.c - START --------------------
// ---------------------------------------------


// JSON data structure
class JsonDataContaminatedAreas : Managed
{
■ref array<ref JsonDataContaminatedArea> Areas;
■ref array<ref array<float>> SafePositions;
}

class JsonDataContaminatedArea : Managed
{
■string AreaName;
■string Type;
■string TriggerType;
■JsonDataAreaData Data;
■JsonDataPlayerData PlayerData;
};

class JsonDataAreaData : Managed
{
■ref array<float> Pos;
■float Radius;
■float PosHeight;
■float NegHeight;
■int InnerRingCount;
■int InnerPartDist;
■bool OuterRingToggle;
■int OuterPartDist;
■int OuterOffset;
■int VerticalLayers;
■int VerticalOffset;
■string ParticleName;
};

class JsonDataPlayerData : Managed
{
■string AroundPartName;
■string TinyPartName;
■string PPERequesterType;
}


// ---------------------------------------------
// ------ JsonDataContaminatedArea.c - END ---------------------
// ---------------------------------------------


// ---------------------------------------------
// ------ JsonDataCredits.c - START --------------------
// ---------------------------------------------


class JsonDataCredits : Managed
{
■ref array<ref JsonDataCreditsDepartment> Departments;
};
```

```
// ---------------------------------------------
// ------ JsonDataCreditsDepartment.c - START --------------------
// ---------------------------------------------


class JsonDataCreditsDepartment : Managed
{
■string DepartmentName;
■ref array<ref JsonDataCreditsSection> Sections;
};




// ---------------------------------------------
// ------ JsonDataCreditsDepartment.c - END ----------------------
// ---------------------------------------------


// ---------------------------------------------
// ------ JsonDataCreditsSection.c - START --------------------
// ---------------------------------------------


class JsonDataCreditsSection : Managed
{
■string SectionName;
■ref array<string> SectionLines;
};




// ---------------------------------------------
// ------ JsonDataCreditsSection.c - END ----------------------
// ---------------------------------------------


// ---------------------------------------------
// ------ JsonFileLoader.c - START --------------------
// ---------------------------------------------


class JsonFileLoader<Class T>
{
■protected static ref JsonSerializer m_Serializer = new JsonSerializer;
■
■static void JsonLoadFile( string filename, out T data )
■{
■■if( FileExist( filename ) )
■■{
■■■string file_content;
■■■string line_content;
■■■string error;
■■■
■■■FileHandle handle = OpenFile( filename, FileMode.READ );
■■■if ( handle == 0 )
■■■■return;
■■■
■■■
■■■while ( FGets( handle,  line_content ) >= 0 )
■■■{
■■■■file_content += line_content;
```

```
// ---------------------------------------------
// ------ JsonObject.c - START --------------------
// ---------------------------------------------


class JsonObject
{■
■ref map<string, string> ■■m_Strings;
■ref map<string, int>■■■m_Ints;
■ref map<string, float>■■■m_Floats;
■ref map<string, bool>■■■m_Bools;
■ref map<string, ref Vector2>■m_Vectors2;
■■
■void JsonObject()
■{
■■m_Strings■= new map<string, string>;
■■m_Ints■■= new map<string, int>;
■■m_Floats■= new map<string, float>;
■■m_Bools■■= new map<string, bool>;
■■m_Vectors2■= new map<string, ref Vector2>;
■}
■
■void ~JsonObject()
■{
■■Clear();
■}
■■
■void Clear()
■{
■■m_Strings.Clear();
■■m_Ints.Clear();
■■m_Floats.Clear();
■■m_Bools.Clear();
■■m_Vectors2.Clear();
■}
■
■void AddString(string name, string value)
■{
■■array<string> strgs = new array<string>;
■■value.Split("\"", strgs);
■■
■■if ( strgs.Count() > 1 )
■■{
■■■value = "";
■■■int str_count = strgs.Count();
■■■
■■■for ( int i = 0; i < str_count; ++i )
■■■{
■■■■string s = strgs[i];
■■■■
■■■■int length = s.Length();
■
■■■■if ( s[length - 1] != "\\" )
■■■■{
■■■■■value += s;
■■■■■
■■■■■if (i < str_count - 1 )
■■■■■{
■■■■■■value += "\\";
■■■■■■value += "\"";
■■■■■}
■■■■}
■■■}
```

```
// ---------------------------------------------
// ------ JumpEvents.c - START --------------------
// ---------------------------------------------


class JumpEventsBase extends PlayerSoundEventBase
{
■override bool HasPriorityOverCurrent(PlayerBase player, EPlayerSoundEventID other_state_id, EPlayer
■{
■■return true;
■}
■
■override bool CanPlay(PlayerBase player)
■{
■■if( !super.CanPlay(player) )
■■{
■■■return false;
■■}
■■return true;
■}
■
■override void OnPlay(PlayerBase player)
■{
■■super.OnPlay(player);
■■if( player.CanSpawnBreathVaporEffect() )
■■■player.SpawnBreathVaporEffect();
■}
■
■override void OnEnd()
■{
■■super.OnEnd();
■■//m_Player.GetStaminaSoundHandlerClient().PostponeStamina(1000);
■■if(m_Player)
■■■StaminaSoundHandlerClient.Cast(m_Player.m_PlayerSoundManagerClient.GetHandler(eSoundHand
■}
}

class JumpSoundEvent extends JumpEventsBase
{
■void JumpSoundEvent()
■{
■■m_HasPriorityOverTypes = -1;//-1 for all
■■m_Type = EPlayerSoundEventType.GENERAL;
■■m_ID = EPlayerSoundEventID.JUMP;
■■m_SoundVoiceAnimEventClassID = 18;
■}
}


// ---------------------------------------------
// ------ JumpEvents.c - END ----------------------
// ---------------------------------------------


// ---------------------------------------------
// ------ JumpsuitJacket_ColorBase.c - START --------------------
// ---------------------------------------------


class JumpsuitJacket_ColorBase extends Clothing
{
■override void SetActions()
■{
```

```
// ---------------------------------------------
// ------ JumpsuitPants_ColorBase.c - START --------------------
// ---------------------------------------------


class JumpsuitPants_ColorBase extends Clothing
{
■override void SetActions()
■{
■■super.SetActions();
■■AddAction(ActionWringClothes);
■}
};
class JumpsuitPants_Blue extends JumpsuitPants_ColorBase {};
class JumpsuitPants_Green extends JumpsuitPants_ColorBase {};
class JumpsuitPants_Grey extends JumpsuitPants_ColorBase {};
class JumpsuitPants_Red extends JumpsuitPants_ColorBase {};


// ---------------------------------------------
// ------ JumpsuitPants_ColorBase.c - END ----------------------
// ---------------------------------------------


// ---------------------------------------------
// ------ Junctures.c - START --------------------
// ---------------------------------------------



bool TryAcquireInventoryJunctureFromServer (notnull Man player, notnull InventoryLocation src, notnull In
{
■if (player.NeedInventoryJunctureFromServer(src.GetItem(), src.GetParent(), dst.GetParent()))
■{
■■if ( ( src.GetItem() && src.GetItem().IsSetForDeletion() ) || ( src.GetParent() && src.GetParent().IsSetFo
■■{
■■■return JunctureRequestResult.JUNCTURE_DENIED;
■■}
■■bool test_dst_occupancy = true;
■■if (GetGame().AddInventoryJunctureEx(player, src.GetItem(), dst, test_dst_occupancy, GameInventory
■■{
■■■syncDebugPrint("[syncinv] juncture needed and acquired, player=" + Object.GetDebugName(player)
■■■return JunctureRequestResult.JUNCTURE_ACQUIRED; // ok
■■}
■■else
■■{
■■■syncDebugPrint("[syncinv] juncture request DENIED, player=" + Object.GetDebugName(player) + " S
■■■return JunctureRequestResult.JUNCTURE_DENIED; // permission to perform juncture denied
■■}
■}
■else
■{
■■syncDebugPrint("[syncinv] juncture not required, player=" + Object.GetDebugName(player) + " STS = '
■■return JunctureRequestResult.JUNCTURE_NOT_REQUIRED; // juncture not necessary
■}
}

bool TryAcquireTwoInventoryJuncturesFromServer (notnull Man player, notnull InventoryLocation src1, no
{
■#ifdef DEVELOPER
■if ( LogManager.IsInventoryReservationLogEnable() )
■{
■■Debug.InventoryReservationLog("STS = " + player.GetSimulationTimeStamp() + " src1:" + src1.DumpT
// ---------------------------------------------
// ------ JungleBoots_ColorBase.c - START --------------------
// ---------------------------------------------


class JungleBoots_ColorBase extends Clothing {};
class JungleBoots_Beige extends JungleBoots_ColorBase {};
class JungleBoots_Black extends JungleBoots_ColorBase {};
class JungleBoots_Brown extends JungleBoots_ColorBase {};
class JungleBoots_Green extends JungleBoots_ColorBase {};
class JungleBoots_Olive extends JungleBoots_ColorBase {};


// ---------------------------------------------
// ------ JungleBoots_ColorBase.c - END ----------------------
// ---------------------------------------------


// ---------------------------------------------
// ------ KeyBinding.c - START --------------------
// ---------------------------------------------


class KeyBinding
{
■private int■■m_ActiveUIMenuID;
■private int■■m_Key1;
■private int■■m_Key2;■
■private string■m_CallbackTarget;
■private string■m_CallbackFunction;
■private string■m_InfoKeys;
■private string■m_InfoDescription
■
■void KeyBinding( int ui_id, int key1, int key2, string call_target, string call_function, string info_bind, strin
■{
■■m_ActiveUIMenuID■= ui_id;
■■m_Key1■■■■= key1;
■■m_Key2■■■■= key2;
■■m_CallbackTarget■= call_target;
■■m_CallbackFunction■= call_function;
■■m_InfoKeys■■■= info_bind;
■■m_InfoDescription■= info_description;
■}
■
■int GetUIMenuID()
■{
■■return m_ActiveUIMenuID;
■}
■
■int GetKey1()
■{
■■return m_Key1;
■}
■
■int GetKey2()
■{
■■return m_Key2;
■}
■
■string GetCallbackTarget()
■{
■■return m_CallbackTarget;
■}
```

```
// ---------------------------------------------
// ------ KeybindingElement.c - START --------------------
// ---------------------------------------------


//! Deprecated
class KeybindingElement extends ScriptedWidgetEventHandler
{
■protected KeybindingsGroup■■■■m_Group;
■
■protected Widget■■■■■■m_Root;
■protected TextWidget■■■■■m_ElementName;
■protected TextWidget■■■■■m_ElementModifier;
■protected ButtonWidget■■■■■m_PrimaryBindButton;
■protected ButtonWidget■■■■■m_AlternativeBindButton;
■protected Widget■■■■■■m_PrimaryClear;
■protected Widget■■■■■■m_AlternativeClear;
■
■protected int■■■■■■■m_ElementIndex;
■protected bool■■■■■■■m_IsEdited;
■protected bool■■■■■■■m_IsAlternateEdited;
■
■protected ref array<int>■■■■m_CustomBind;
■protected ref array<int>■■■■m_CustomAlternateBind;
■
■protected ref Timer■■■■■■m_EntryTimer = new Timer( CALL_CATEGORY_GUI );
■
■void KeybindingElement( int key_index, Widget parent, KeybindingsGroup group )
■{
■■m_Root■■■■■= GetGame().GetWorkspace().CreateWidgets( GetLayoutName(), parent );
■■m_ElementName■■■= TextWidget.Cast( m_Root.FindAnyWidget( "setting_label" ) );
■■m_ElementModifier■■= TextWidget.Cast( m_Root.FindAnyWidget( "modifier_label" ) );
■■m_PrimaryBindButton■■= ButtonWidget.Cast( m_Root.FindAnyWidget( "primary_bind" ) );
■■m_AlternativeBindButton■= ButtonWidget.Cast( m_Root.FindAnyWidget( "alternative_bind" ) );
■■m_PrimaryClear■■■= m_Root.FindAnyWidget( "primary_clear" );
■■m_AlternativeClear■■= m_Root.FindAnyWidget( "alternative_clear" );
■■
■■m_Group■■■■■= group;
■■m_ElementIndex■■■= key_index;
■■
■■Reload();
■■m_Root.SetHandler( this );
■}
■
■string GetLayoutName()
■{
■■return "gui/layouts/new_ui/options/keybindings_selectors/keybinding_option.layout";
■}
■
■bool IsChanged()
■{
■■return m_IsEdited;
■}
■
■bool IsAlternateChanged()
■{
■■return m_IsAlternateEdited;
■}
■
■array<int> GetChangedBinds()
■{
■■return m_CustomBind;
■}
```

```
// ---------------------------------------------
// ------ KeybindingElementNew.c - START --------------------
// ---------------------------------------------


class KeybindingElementNew extends ScriptedWidgetEventHandler
{
■protected KeybindingsContainer■■■m_Container;
■
■protected Widget■■■■■■m_Root;
■protected TextWidget■■■■■m_ElementName;
■protected TextWidget■■■■■m_ElementModifier;
■protected ButtonWidget■■■■■m_PrimaryBindButton;
■protected ButtonWidget■■■■■m_AlternativeBindButton;
■protected Widget■■■■■■m_PrimaryClear;
■protected Widget■■■■■■m_AlternativeClear;
■
■protected int■■■■■■■m_ElementIndex;
■protected bool■■■■■■■m_IsEdited;
■protected bool■■■■■■■m_IsAlternateEdited;
■
■protected ref array<int>■■■■m_CustomBind;
■protected ref array<int>■■■■m_CustomAlternateBind;
■
■protected ref Timer■■■■■■m_EntryTimer = new Timer( CALL_CATEGORY_GUI );
■
■void KeybindingElementNew( int key_index, Widget parent, KeybindingsContainer group )
■{
■■m_Root■■■■■= GetGame().GetWorkspace().CreateWidgets( GetLayoutName(), parent );
■■m_ElementName■■■= TextWidget.Cast( m_Root.FindAnyWidget( "setting_label" ) );
■■m_ElementModifier■■= TextWidget.Cast( m_Root.FindAnyWidget( "modifier_label" ) );
■■m_PrimaryBindButton■■= ButtonWidget.Cast( m_Root.FindAnyWidget( "primary_bind" ) );
■■m_AlternativeBindButton■= ButtonWidget.Cast( m_Root.FindAnyWidget( "alternative_bind" ) );
■■m_PrimaryClear■■■= m_Root.FindAnyWidget( "primary_clear" );
■■m_AlternativeClear■■= m_Root.FindAnyWidget( "alternative_clear" );
■■
■■m_Container■■■■■= group;
■■m_ElementIndex■■■= key_index;
■■
■■Reload();
■■m_Root.SetHandler( this );
■}
■
■string GetLayoutName()
■{
■■return "gui/layouts/new_ui/options/keybindings_selectors/keybinding_option.layout";
■}
■
■bool IsChanged()
■{
■■return m_IsEdited;
■}
■
■bool IsAlternateChanged()
■{
■■return m_IsAlternateEdited;
■}
■
■array<int> GetChangedBinds()
■{
■■return m_CustomBind;
■}
■
```

```
// ---------------------------------------------
// ------ KeybindingsContainer.c - START --------------------
// ---------------------------------------------


//container for various subgroups and their elements
class KeybindingsContainer extends ScriptedWidgetEventHandler
{
■protected const int  ■■■■■■■■NO_SORTING = -1;
■protected Widget■■■■■■■■■m_Root;
■protected KeybindingsMenu■■■■■■■m_Menu;
■protected ScrollWidget ■■■■■■■■m_Scroll;
■
■protected ref map<int, ref ElementArray>■■■m_KeyWidgetElements; //<input_action_id,<KeybindingEle
■protected int■■■■■■■■■■m_CurrentSettingKeyIndex = -1;
■protected int■■■■■■■■■■m_CurrentSettingAlternateKeyIndex = -1;
■
■protected ref array<Widget> ■■■■■■m_Subgroups;
■
■void KeybindingsContainer( int index, Input input, Widget parent, KeybindingsMenu menu )
■{
■■m_KeyWidgetElements■= new map<int, ref ElementArray>;
■■m_Menu■■■= menu;
■■
■■m_Root■= GetGame().GetWorkspace().CreateWidgets( GetLayoutName(), parent );
■■m_Scroll = ScrollWidget.Cast(m_Root.FindAnyWidget("group_scroll"));
■■Widget container■= m_Root.FindAnyWidget( "group_scroll" );
■■
■■CreateSubgroups(container,input);
■■
■■container.Update();
■■
■■m_Root.SetHandler( this );
■}
■
■string GetLayoutName()
■{
■■return "gui/layouts/new_ui/options/keybindings_selectors/keybinding_container.layout";
■}
■
■void OnSelectKBPreset( int index )
■{
■■GetUApi().PresetSelect( index );
■■ReloadElements();
■■GetUApi().Export();
■■GetUApi().SaveInputPresetMiscData();
■}
■
■void ReloadElements()
■{
■■foreach ( ElementArray elements : m_KeyWidgetElements )
■■{
■■■foreach (KeybindingElementNew element : elements)
■■■{
■■■■element.Reload();
■■■}
■■}
■}
■
■void AddSubgroup( int sort_index, Widget parent, Input input )
■{
■■Widget subgroup■■■■= GetGame().GetWorkspace().CreateWidgets( "gui/layouts/new_ui/options/key
■■Widget subgroup_content ■= subgroup.FindAnyWidget( "subgroup_content" );
```

```
// ---------------------------------------------
// ------ KeybindingsEntryWindow.c - START --------------------
// ---------------------------------------------





// ---------------------------------------------
// ------ KeybindingsEntryWindow.c - END ----------------------
// ---------------------------------------------


// ---------------------------------------------
// ------ KeybindingsGroup.c - START --------------------
// ---------------------------------------------


//! Deprecated
class KeybindingsGroup extends ScriptedWidgetEventHandler
{
■protected Widget■■■■■■■■■m_Root;
■protected KeybindingsMenu■■■■■■■m_Menu;
■
■protected ref map<int, ref KeybindingElement>■■m_KeyWidgets;
■protected int■■■■■■■■■■m_CurrentSettingKeyIndex = -1;
■protected int■■■■■■■■■■m_CurrentSettingAlternateKeyIndex = -1;
■
■protected ref DropdownPrefab■■■■■■m_KBDropdown;
■
■void KeybindingsGroup( int index, Input input, Widget parent, KeybindingsMenu menu )
■{
■■m_KeyWidgets■= new map<int, ref KeybindingElement>;
■■m_Menu■■■= menu;
■■
■■string group_name;
■■input.GetActionGroupName( index, group_name );
■■
■■m_Root■= GetGame().GetWorkspace().CreateWidgets( GetLayoutName(), parent );
■■Widget subgroup■= m_Root.FindAnyWidget( "group_content" );
■■
//■■for( int i = 0; i < 1; i++ )
//■■{
■■■AddSubgroup( /*index, */subgroup, input );
//■■}
■■
■■InitPresets( index, m_Root.FindAnyWidget( "group_header" ), input );
■■
■■subgroup.Update();
■■
■■m_Root.SetHandler( this );
■}
■
■string GetLayoutName()
■{
■■return "gui/layouts/new_ui/options/keybindings_selectors/keybinding_group.layout";
■}
■
■void InitPresets( int index, Widget parent, Input input )
■{
■■Widget kb_root = parent.FindAnyWidget( "keyboard_dropown" );
■■
■■string profile_text;
```

```
// ---------------------------------------------
// ------ KeybindingsMenu.c - START --------------------
// ---------------------------------------------


class KeybindingsMenu extends UIScriptedMenu
{
■protected TabberUI■■■■■■■m_Tabber;
■protected ref DropdownPrefab■■■■m_KBDropdown; //DEPRECATED
■protected ref OptionSelectorMultistate■■m_PresetSelector;
■protected ref KeybindingsContainer ■■■m_GroupsContainer;
■protected ref array<ref KeybindingsGroup>■m_Tabs; //DEPRECATED
■
■protected TextWidget■■■■■■m_Version;
■protected ButtonWidget■■■■■■m_Apply;
■protected ButtonWidget■■■■■■m_Back;
■protected ButtonWidget■■■■■■m_Undo;
■protected ButtonWidget■■■■■■m_Defaults;
■protected ButtonWidget■■■■■■m_HardReset;
■
■protected int■■■■■■■■m_CurrentSettingKeyIndex = -1;
■protected int■■■■■■■■m_CurrentSettingAlternateKeyIndex = -1;
■protected int■■■■■■■■m_OriginalPresetIndex;
■protected int■■■■■■■■m_TargetPresetIndex;
■protected ref array<int>■■■■■m_SetKeybinds;
■
■const int MODAL_ID_BACK = 1337;
■const int MODAL_ID_DEFAULT = 100;
■const int MODAL_ID_DEFAULT_ALL = 101;
■const int MODAL_ID_PRESET_CHANGE = 200;
■const int MODAL_RESULT_DEFAULT_CURRENT = 0;
■const int MODAL_RESULT_DEFAULT_ALL = 1;
■
■override Widget Init()
■{
■■Input input = GetGame().GetInput();
■■layoutRoot■■■= GetGame().GetWorkspace().CreateWidgets("gui/layouts/new_ui/options/pc/keybindin
■■
■■m_Version■■■= TextWidget.Cast(layoutRoot.FindAnyWidget("version"));
■■m_Apply■■■■= ButtonWidget.Cast(layoutRoot.FindAnyWidget("apply"));
■■m_Back■■■■= ButtonWidget.Cast(layoutRoot.FindAnyWidget("back"));
■■m_Undo■■■■= ButtonWidget.Cast(layoutRoot.FindAnyWidget("undo"));
■■m_Defaults■■■= ButtonWidget.Cast(layoutRoot.FindAnyWidget("reset"));
■■m_HardReset■■■= ButtonWidget.Cast(layoutRoot.FindAnyWidget("reset_all"));
■■
■■layoutRoot.FindAnyWidget("Tabber").GetScript(m_Tabber);
■■
■■string version;
■■GetGame().GetVersion(version);
■■#ifdef PLATFORM_CONSOLE
■■■version = "#main_menu_version" + " " + version + " (" + g_Game.GetDatabaseID() + ")";
■■#else
■■■version = "#main_menu_version" + " " + version;
■■#endif
■■m_Version.SetText(version);

■■#ifdef PLATFORM_PS4
■■■string back = "circle";
■■■if (GetGame().GetInput().GetEnterButton() == GamepadButton.A)
■■■{
■■■■back = "circle";
■■■}
■■■else
```

```
// ----------------------------------------------
// ------ KeysToUIElements.c - START --------------------
// ----------------------------------------------


//
// Keep this Object up-to-date with KeyCodes Enum in EnSystem.c
//
// KeysToUIElements is mapping of KeyCodes to its literal name or icon from ImageSet
// that will be displayed in floating widget for Default Actions - aka Action Target Selection

typedef Param2<string, bool> KeyToUIElement;

class KeysToUIElements
{
■static private ref map<int, ref KeyToUIElement> m_KeysToUIElements;
■
■static private void■RegisterKeyToUIElement(int key_code, string name, bool is_image_set )
■{
■■if ( m_KeysToUIElements == NULL )
■■{
■■■m_KeysToUIElements = new map<int, ref KeyToUIElement>;
■■}
■■
■■if ( m_KeysToUIElements.Contains(key_code) )
■■{
■■■Debug.Log("Template ID: "+string.ToString(key_code)+" is alredy exist!", "KeysToUIElements -> OnI
■■}
■■else
■■{
■■■KeyToUIElement params = new KeyToUIElement(name, is_image_set);
■■■m_KeysToUIElements.Set(key_code, params);
■■}
■}

■static KeyToUIElement GetKeyToUIElement(int key_code)
■{
■■if ( m_KeysToUIElements && m_KeysToUIElements.Contains(key_code) )
■■{
■■■return m_KeysToUIElements.Get(key_code);
■■}
■■
■■Debug.Log("Template ID: "+string.ToString(key_code)+" does not exist!", "KeysToUIElements -> GetK
■■return NULL;
■}

■static void Init()
■{
■■/////////////////////////////////////////////////////////////////////////////////////////////////
■■//////// Register KeysToUIElements ////////////////////////////////////////////////////////////////////
■■//■■■■■  | KeyCode■■■■■| Name■■■■■■■■■■| is_image_set  //
■■RegisterKeyToUIElement( KeyCode.KC_ESCAPE■■, "ESC"■■■■■■■■■■, false■■■);
■■RegisterKeyToUIElement( KeyCode.KC_1■■■, "1"■■■■■■■■■■, false■■■);
■■RegisterKeyToUIElement( KeyCode.KC_2■■■, "2"■■■■■■■■■■, false■■■);
■■RegisterKeyToUIElement( KeyCode.KC_3■■■, "3"■■■■■■■■■■, false■■■);
■■RegisterKeyToUIElement( KeyCode.KC_4■■■, "4"■■■■■■■■■■, false■■■);
■■RegisterKeyToUIElement( KeyCode.KC_5■■■, "5"■■■■■■■■■■, false■■■);
■■RegisterKeyToUIElement( KeyCode.KC_6■■■, "6"■■■■■■■■■■, false■■■);
■■RegisterKeyToUIElement( KeyCode.KC_7■■■, "7"■■■■■■■■■■, false■■■);
■■RegisterKeyToUIElement( KeyCode.KC_8■■■, "8"■■■■■■■■■■, false■■■);
■■RegisterKeyToUIElement( KeyCode.KC_9■■■, "9"■■■■■■■■■■, false■■■);
■■RegisterKeyToUIElement( KeyCode.KC_0■■■, "0"■■■■■■■■■■, false■■■);
■■RegisterKeyToUIElement( KeyCode.KC_MINUS■■, "-"■■■■■■■■■■, false■■■);
```

```
// ---------------------------------------------
// ------ KillerData.c - START --------------------
// ---------------------------------------------


class KillerData
{
■EntityAI ■m_Killer;
■EntityAI ■m_MurderWeapon; //can be fists, so no ItemBase possible
■bool ■■m_KillerHiTheBrain;
}



// ---------------------------------------------
// ------ KillerData.c - END ----------------------
// ---------------------------------------------


// ---------------------------------------------
// ------ KitBase.c - START --------------------
// ---------------------------------------------


class KitBase extends ItemBase
{
■ref protected EffectSound ■■m_DeployLoopSound;
■protected bool ■■■■■m_DeployedRegularly;

■void KitBase()
■{
■■RegisterNetSyncVariableBool("m_IsSoundSynchRemote");
■■RegisterNetSyncVariableBool("m_IsDeploySound");
■}

■void ~KitBase()
■{
■■SEffectManager.DestroyEffect( m_DeployLoopSound );
■}

■override bool IsBasebuildingKit()
■{
■■return true;
■}
■
■override bool HasProxyParts()
■{
■■return true;
■}
■
■override void OnVariablesSynchronized()
■{
■■super.OnVariablesSynchronized();
■■
■■if ( IsDeploySound() )
■■{
■■■PlayDeploySound();
■■}
■■■■
■■if ( CanPlayDeployLoopSound() )
■■{
■■■PlayDeployLoopSound();
■■}
```

```
// ---------------------------------------------
// ------ KitchenKnife.c - START --------------------
// ---------------------------------------------


class KitchenKnife extends ToolBase
{
■override bool IsMeleeFinisher()
■{
■■return true;
■}
■
■override array<int> GetValidFinishers()
■{
■■return {EMeleeHitType.FINISHER_LIVERSTAB,EMeleeHitType.FINISHER_NECKSTAB};
■}
■
■override void SetActions()
■{
■■super.SetActions();
■■
■■AddAction(ActionBurnSewTarget);
■■AddAction(ActionUnrestrainTarget);
■■AddAction(ActionSkinning);
■■AddAction(ActionMineBush);
■■AddAction(ActionMineTreeBark);
■■AddAction(ActionBurnSewSelf);
■■AddAction(ActionDigWorms);
■■AddAction(ActionShaveTarget);
■■AddAction(ActionDisarmMine);
■■AddAction(ActionDisarmExplosive);
■■AddAction(ActionShave);
■}
}


// ---------------------------------------------
// ------ KitchenKnife.c - END ----------------------
// ---------------------------------------------


// ---------------------------------------------
// ------ KitchenTimer.c - START --------------------
// ---------------------------------------------


class KitchenTimer: ClockBase
{
■const string■■RINGING_SOUND ■■■■= "KitchenTimer_Ring_Loop_SoundSet";
■const string■■DESTROYED_SOUND ■■■= "AlarmClock_Destroyed_SoundSet";
■const string■■HIT_SOUND ■■■■■= "AlarmClock_Hit_SoundSet";
■const string■■WORKING_SOUND■■■■= "KitchenTimer_Ticking_Loop_SoundSet";
■
■EffectSound ■■m_RingingStopSound;
■static ref NoiseParams ■m_NoisePar;
■static NoiseSystem■■m_NoiseSystem;

■int ■■■■m_AlarmInSecs;
■
■override void Init()
■{
■■super.Init();
■
```

```
// ---------------------------------------------
// ------ Kiwi.c - START --------------------
// ---------------------------------------------


class Kiwi : Edible_Base
{
■override bool CanBeCooked()
■{
■■return false;
■}■■
■
■override bool CanBeCookedOnStick()
■{
■■return false;
■}■
■
■override bool IsFruit()
■{
■■return true;
■}
■
■override bool CanDecay()
■{
■■return true;
■}
■
■override void SetActions()
■{
■■super.SetActions();
■■
■■AddAction(ActionForceFeed);
■■AddAction(ActionEatFruit);
■■AddAction(ActionCreateIndoorFireplace);
■■AddAction(ActionCreateIndoorOven);
■}
}



// ---------------------------------------------
// ------ Kiwi.c - END ----------------------
// ---------------------------------------------


// ---------------------------------------------
// ------ KnifeHolster.c - START --------------------
// ---------------------------------------------


class KnifeHolster: Clothing {};



// ---------------------------------------------
// ------ KnifeHolster.c - END ----------------------
// ---------------------------------------------
// ---------------------------------------------
// ------ KukriKnife.c - START --------------------
// ---------------------------------------------


class KukriKnife extends ToolBase
{
■override bool IsMeleeFinisher()
■{
■■return true;
■}
■
■override array<int> GetValidFinishers()
■{
■■return {EMeleeHitType.FINISHER_LIVERSTAB,EMeleeHitType.FINISHER_NECKSTAB};
■}
■
■override void SetActions()
■{
■■super.SetActions();
■■
■■AddAction(ActionBurnSewTarget);
■■AddAction(ActionUnrestrainTarget);
■■AddAction(ActionSkinning);
■■AddAction(ActionMineBush);
■■AddAction(ActionMineTreeBark);
■■AddAction(ActionBurnSewSelf);
■■AddAction(ActionDigWorms);
■■AddAction(ActionShaveTarget);
■■AddAction(ActionShave);
■}
}


// ---------------------------------------------
// ------ KukriKnife.c - END ----------------------
// ---------------------------------------------


// ---------------------------------------------
// ------ KuruShake.c - START --------------------
// ---------------------------------------------


class KuruShake
{
■const float RANDOM_RANGE_POINT = 1;
■const float RANDOM_RANGE_DEVIATION = 0.50;
■const float SHAKE_DURATION_PHASE1 = 0.01;
■const float SHAKE_STRENGTH_MIN = 1;
■const float SHAKE_STRENGTH_MAX = 5;
■
■PlayerBase m_Player;
■float m_Time;
■float m_Time2;
■float m_RelativeTime;
■float m_RelativeTime2;
■float m_ShakeStrength;
■
■
■ref array<vector> m_Curve = new array<vector>;
■
■void KuruShake(PlayerBase player, float amount)
```

```
// ---------------------------------------------
// ------ LabCoat.c - START --------------------
// ---------------------------------------------


class LabCoat extends Clothing
{
■override void SetActions()
■{
■■super.SetActions();
■■AddAction(ActionWringClothes);
■}
};


// ---------------------------------------------
// ------ LabCoat.c - END ----------------------
// ---------------------------------------------


// ---------------------------------------------
// ------ LactariusMushroom.c - START --------------------
// ---------------------------------------------


class LactariusMushroom : MushroomBase
{
■override void SetActions()
■{
■■super.SetActions();
■■
■■AddAction(ActionCreateIndoorFireplace);
■■AddAction(ActionCreateIndoorOven);
■}
}



// ---------------------------------------------
// ------ LactariusMushroom.c - END ----------------------
// ---------------------------------------------


// ---------------------------------------------
// ------ LandmineExplosion.c - START --------------------
// ---------------------------------------------


class LandmineExplosion : EffectParticle
{
■void LandmineExplosion()
■{
■■SetParticleID(ParticleList.EXPLOSION_LANDMINE);
■}
}


// ---------------------------------------------
// ------ LandmineExplosion.c - END ----------------------
// ---------------------------------------------
// ---------------------------------------------
// ------ Land_Airfield_Small_Control.c - START --------------------
// ---------------------------------------------


class Land_Airfield_Small_Control extends BuildingWithFireplace
{
}


// ---------------------------------------------
// ------ Land_Airfield_Small_Control.c - END ----------------------
// ---------------------------------------------


// ---------------------------------------------
// ------ Land_FuelStation_Feed.c - START --------------------
// ---------------------------------------------


class Land_FuelStation_Feed extends FuelStation
{
■override void EEKilled(Object killer)
■{
■■super.EEKilled(killer);

■ ■Explode(DT_EXPLOSION, "LandFuelFeed_Ammo");
■}
■
■override void OnExplosionEffects(Object source, Object directHit, int componentIndex, string surface, ve
■{
■■if ( !GetGame().IsDedicatedServer() )
■■{■
■■■vector n = surfNormal.VectorToAngles() + "0 90 0";
■■■Particle p1 = ParticleManager.GetInstance().PlayInWorld(ParticleList.SMOKE_GENERIC_WRECK, p
■■■p1.SetOrientation(n);
■■
■■■Particle p2 = ParticleManager.GetInstance().PlayInWorld(ParticleList.EXPLOSION_LANDMINE, pos)
■■■p2.SetOrientation(n);

■■■Particle p3 = ParticleManager.GetInstance().PlayInWorld(ParticleList.IMPACT_METAL_RICOCHET,
■■■p3.SetOrientation(n);
■■
■■■Particle p4 = ParticleManager.GetInstance().PlayInWorld(ParticleList.IMPACT_GRAVEL_RICOCHET
■■■p4.SetOrientation(n);
■■}
■}
■
■//! Returns true if this stand is functional
■bool HasFuelToGive()
■{
■■return !IsRuined();
■}
};


// ---------------------------------------------
// ------ Land_FuelStation_Feed.c - END ----------------------
// ---------------------------------------------
// ---------------------------------------------
// ------ Land_House_1B01_Pub.c - START --------------------
// ---------------------------------------------

class Land_House_1B01_Pub extends BuildingWithFireplace
{
}

// ---------------------------------------------
// ------ Land_House_1B01_Pub.c - END ----------------------
// ---------------------------------------------

// ---------------------------------------------
// ------ Land_House_1W01.c - START --------------------
// ---------------------------------------------

class Land_House_1W01 extends BuildingWithFireplace
{
}

// ---------------------------------------------
// ------ Land_House_1W01.c - END ----------------------
// ---------------------------------------------

// ---------------------------------------------
// ------ Land_House_1W02.c - START --------------------
// ---------------------------------------------

class Land_House_1W02 extends BuildingWithFireplace
{
}

// ---------------------------------------------
// ------ Land_House_1W02.c - END ----------------------
// ---------------------------------------------

// ---------------------------------------------
// ------ Land_House_1W03.c - START --------------------
// ---------------------------------------------

class Land_House_1W03 extends BuildingWithFireplace
{
}

// ---------------------------------------------
// ------ Land_House_1W03.c - END ----------------------
// ---------------------------------------------
// ---------------------------------------------
// ------ Land_House_1W04.c - START --------------------
// ---------------------------------------------


class Land_House_1W04 extends BuildingWithFireplace
{
}


// ---------------------------------------------
// ------ Land_House_1W04.c - END ----------------------
// ---------------------------------------------


// ---------------------------------------------
// ------ Land_House_1W05.c - START --------------------
// ---------------------------------------------


class Land_House_1W05 extends BuildingWithFireplace
{
}


// ---------------------------------------------
// ------ Land_House_1W05.c - END ----------------------
// ---------------------------------------------


// ---------------------------------------------
// ------ Land_House_1W06.c - START --------------------
// ---------------------------------------------


class Land_House_1W06 extends BuildingWithFireplace
{
}


// ---------------------------------------------
// ------ Land_House_1W06.c - END ----------------------
// ---------------------------------------------


// ---------------------------------------------
// ------ Land_House_1W08.c - START --------------------
// ---------------------------------------------


class Land_House_1W08 extends BuildingWithFireplace
{
}



// ---------------------------------------------
// ------ Land_House_1W08.c - END ----------------------
// ---------------------------------------------
// ---------------------------------------------
// ------ Land_House_1W09.c - START --------------------
// ---------------------------------------------

class Land_House_1W09 extends BuildingWithFireplace
{
}



// ---------------------------------------------
// ------ Land_House_1W09.c - END ----------------------
// ---------------------------------------------


// ---------------------------------------------
// ------ Land_House_1W10.c - START --------------------
// ---------------------------------------------

class Land_House_1W10 extends BuildingWithFireplace
{
}


// ---------------------------------------------
// ------ Land_House_1W10.c - END ----------------------
// ---------------------------------------------


// ---------------------------------------------
// ------ Land_House_1W11.c - START --------------------
// ---------------------------------------------

class Land_House_1W11 extends BuildingWithFireplace
{
}


// ---------------------------------------------
// ------ Land_House_1W11.c - END ----------------------
// ---------------------------------------------


// ---------------------------------------------
// ------ Land_House_1W12.c - START --------------------
// ---------------------------------------------

class Land_House_1W12 extends BuildingWithFireplace
{
}


// ---------------------------------------------
// ------ Land_House_1W12.c - END ----------------------
// ---------------------------------------------
// ---------------------------------------------
// ------ Land_House_2B01.c - START --------------------
// ---------------------------------------------


class Land_House_2B01 extends BuildingWithFireplace
{
}


// ---------------------------------------------
// ------ Land_House_2B01.c - END ----------------------
// ---------------------------------------------


// ---------------------------------------------
// ------ Land_House_2W01.c - START --------------------
// ---------------------------------------------


class Land_House_2W01 extends BuildingWithFireplace
{
}


// ---------------------------------------------
// ------ Land_House_2W01.c - END ----------------------
// ---------------------------------------------


// ---------------------------------------------
// ------ Land_House_2W02.c - START --------------------
// ---------------------------------------------


class Land_House_2W02 extends BuildingWithFireplace
{
}


// ---------------------------------------------
// ------ Land_House_2W02.c - END ----------------------
// ---------------------------------------------


// ---------------------------------------------
// ------ Land_Lamp_City1_amp.c - START --------------------
// ---------------------------------------------


class Land_Lamp_City1_amp extends PASReceiver
{
}


// ---------------------------------------------
// ------ Land_Lamp_City1_amp.c - END ----------------------
// ---------------------------------------------
// ---------------------------------------------
// ------ Land_Misc_Greenhouse.c - START --------------------
// ---------------------------------------------


class Land_Misc_Greenhouse extends BuildingSuper
{
■void Land_Misc_Greenhouse()
■{
■■
■}
}

class Land_Misc_Polytunnel extends BuildingSuper
{
■void Land_Misc_Polytunnel()
■{
■■
■}
}


// ---------------------------------------------
// ------ Land_Misc_Greenhouse.c - END ----------------------
// ---------------------------------------------


// ---------------------------------------------
// ------ Land_Misc_Through_Static.c - START --------------------
// ---------------------------------------------


class Land_Misc_Through_Static : Well {}


// ---------------------------------------------
// ------ Land_Misc_Through_Static.c - END ----------------------
// ---------------------------------------------


// ---------------------------------------------
// ------ Land_Misc_Well_Pump_Blue.c - START --------------------
// ---------------------------------------------


class Land_Misc_Well_Pump_Blue extends Well
{
■
}


// ---------------------------------------------
// ------ Land_Misc_Well_Pump_Blue.c - END ----------------------
// ---------------------------------------------


// ---------------------------------------------
// ------ Land_Misc_Well_Pump_Yellow.c - START --------------------
// ---------------------------------------------


class Land_Misc_Well_Pump_Yellow extends Well
{
```

```
// ---------------------------------------------
// ------ Land_Power_Pole_Conc1_Amp.c - START --------------------
// ---------------------------------------------


class Land_Power_Pole_Conc1_Amp extends PASReceiver
{
}



// ---------------------------------------------
// ------ Land_Power_Pole_Conc1_Amp.c - END ----------------------
// ---------------------------------------------


// ---------------------------------------------
// ------ Land_Power_Pole_Conc4_Lamp_Amp.c - START --------------------
// ---------------------------------------------


class Land_Power_Pole_Conc4_Lamp_Amp extends PASReceiver
{
}


// ---------------------------------------------
// ------ Land_Power_Pole_Conc4_Lamp_Amp.c - END ----------------------
// ---------------------------------------------


// ---------------------------------------------
// ------ Land_Power_Pole_Wood1_Amp.c - START --------------------
// ---------------------------------------------


class Land_Power_Pole_Wood1_Amp extends PASReceiver
{
}



// ---------------------------------------------
// ------ Land_Power_Pole_Wood1_Amp.c - END ----------------------
// ---------------------------------------------


// ---------------------------------------------
// ------ Land_Power_Pole_Wood1_Lamp_Amp.c - START --------------------
// ---------------------------------------------


class Land_Power_Pole_Wood1_Lamp_Amp extends PASReceiver
{
}


// ---------------------------------------------
// ------ Land_Power_Pole_Wood1_Lamp_Amp.c - END ----------------------
// ---------------------------------------------
// ----------------------------------------------
// ------ Land_Radio_PanelBig.c - START --------------------
// ----------------------------------------------


class Land_Radio_PanelBig extends StaticTransmitter
{
■override bool DisableVicinityIcon()
■{
■■return true;
■}
■
■// --- SYSTEM EVENTS
■override void OnStoreSave( ParamsWriteContext ctx )
■{
■■super.OnStoreSave( ctx );
■■
■■//store tuned frequency
■■ctx.Write( GetTunedFrequencyIndex() );
■}
■
■override bool OnStoreLoad( ParamsReadContext ctx, int version )
■{
■■if ( !super.OnStoreLoad( ctx, version ) )
■■■return false;

■■//--- Panel Radio data ---
■■//load and set tuned frequency
■■int tuned_frequency_idx;
■■if ( !ctx.Read( tuned_frequency_idx ) )
■■{
■■■tuned_frequency_idx = 0;■■//set default
■■}
■■SetFrequencyByIndex( tuned_frequency_idx );
■■//---
■■
■■return true;
■}■
■
■//--- BASE
■override bool IsStaticTransmitter()
■{
■■return true;
■}
■
■void SetNextFrequency( PlayerBase player = NULL )
■{
■■SetNextChannel();
■■
■■/*
■■if ( player )
■■{
■■■DisplayRadioInfo( GetTunedFrequency().ToString(), player );
■■}
■■*/
■}

■//--- POWER EVENTS
■override void OnSwitchOn()
■{
■■if ( !GetCompEM().CanWork() )
■■{
■■■GetCompEM().SwitchOff();
```

```
// ---------------------------------------------
// ------ Land_Radio_PanelPAS.c - START --------------------
// ---------------------------------------------


class Land_Radio_PanelPAS extends PASBroadcaster
{
■//Sounds
■const string SOUND_PAS_TURN_ON ■■■= "pastransmitter_turnon_SoundSet";
■const string SOUND_PAS_TURN_OFF ■■= "pastransmitter_turnoff_SoundSet";
■const string SOUND_PAS_TURNED_ON ■■= "pastransmitter_staticnoise_SoundSet";

■protected EffectSound m_Sound;
■protected EffectSound m_SoundLoop;
■
■//--- BASE
■override bool IsStaticTransmitter()
■{
■■return true;
■}
■
■override bool DisableVicinityIcon()
■{
■■return true;
■}
■
■//--- POWER EVENTS
■override void OnSwitchOn()
■{
■■super.OnSwitchOn();
■■
■■if ( !GetCompEM().CanWork() )
■■{
■■■GetCompEM().SwitchOff();
■■}
■■
■■//sound
■■SoundTurnOn();
■}
■
■override void OnSwitchOff()
■{
■■super.OnSwitchOff();
■■
■■//sound
■■SoundTurnOff();
■}■
■
■override void OnWorkStart()
■{
■■super.OnWorkStart();
■■
■■//turn off device
■■SwitchOn ( true ); // start send/receive voice
■■
■■//sound
■■SoundTurnedOnNoiseStart();
■}
■
■override void OnWorkStop()
■{
■■super.OnWorkStop();
■■
```

```
// ---------------------------------------------
// ------ Land_Underground_Entrance.c - START --------------------
// ---------------------------------------------


enum EUndegroundEntranceState
{
■UNINITIALIZED,
■CLOSED,//fully closed
■//opening
■OPENING_A,
■OPENING_B,
■OPENING_C,
■OPENING_D,
■OPENING_E,//fully open
■OPENING_F,
■OPENING_G,
■//closing
■CLOSING_A,
■CLOSING_B,
■CLOSING_C,
■CLOSING_D,
■CLOSING_E,
■CLOSING_F,
■CLOSING_G
}

enum EUndegroundDoorType
{
■MAIN,
■SMALL,
}

class AlarmLight : SpotlightLight
{
■void AlarmLight()
■{
■■SetVisibleDuringDaylight(true);
■■SetRadiusTo(15);
■■SetBrightnessTo(10);
■■SetFlareVisible(false);
■■SetAmbientColor(1.0, 0.0, 0.0);
■■SetDiffuseColor(1.0, 0.0, 0.0);
■■SetLifetime(1000);
■■SetDisableShadowsWithinRadius(-1);
■■SetFadeOutTime(1);
■■SetCastShadow(false);
■■m_FadeInTime = 0.25;
■}
}

//-----------------------------------------------------------------------------------------------------
//----------------------------------- Land_Underground_EntranceBase -------------------------------------
//-----------------------------------------------------------------------------------------------------

class Land_Underground_EntranceBase : House
{
■EUndegroundEntranceState m_DoorState ■■= EUndegroundEntranceState.CLOSED;
■EUndegroundEntranceState m_DoorStatePrev ■= EUndegroundEntranceState.UNINITIALIZED;
■float ■■■■■■■■■m_AnimPhase;
■ref AnimationTimer ■■■■■■m_AnimTimerDoorServer;
■ref Timer■■■■■■■■m_NavmeshTimer;
■ref array<Land_Underground_Panel> ■■m_ConnectedPanels;
```

```
// ----------------------------------------------
// ------ Land_Underground_Panel.c - START --------------------
// ----------------------------------------------


enum ELEDState
{
■OFF,
■BLINKING,
■ON,
}

enum ELEDColors
{
■RED,
■GREEN,
}

class Land_Underground_Panel: House
{
■static ref set<Land_Underground_EntranceBase> ■■m_Entrances;
■static ref set<Land_Underground_Panel>■■■■m_Panels;
■
■Land_Underground_EntranceBase m_LinkedDoor;
■
■const string COLOR_LED_OFF ■■= "#(argb,8,8,3)color(0,0,0,1.0,co)";
■const string COLOR_LED_GREEN■= "#(argb,8,8,3)color(0,1,0,1.0,co)";
■const string COLOR_LED_RED ■■= "#(argb,8,8,3)color(1,0,0,1.0,co)";
■
■const string SELECTION_NAME_LED_RED■■= "LED_Red";
■const string SELECTION_NAME_LED_GREEN■= "LED_Green";
■
■bool m_PanelWasUsed
■bool m_PanelWasUsedPrev;
■bool m_BlinkingFlipFlop;
■ref Timer m_FlipFlopTimer;
■
■EffectSound ■■m_ActivationSound;
■
■ELEDState m_LedStateRed;
■ELEDState m_LedStateGreen;
■
■void Land_Underground_Panel()
■{
■■RegisterNetSyncVariableBool("m_PanelWasUsed");
■■RegisterPanel(this);
■■SetLEDState(ELEDColors.RED, ELEDState.ON);
■■SetLEDState(ELEDColors.GREEN, ELEDState.OFF);
■}
■
■void ~Land_Underground_Panel()
■{
■■UnregisterPanel(this);
■}
■
■bool CanInteract()
■{
■■Land_Underground_EntranceBase door = GetLinkedDoor();
■■if (door)
■■{
■■■return Land_Underground_EntranceBase.Cast(door).CanManipulate();
■■}
■■return false;
```

```
// ---------------------------------------------
// ------ Land_Underground_Panel_Lever.c - START --------------------
// ---------------------------------------------


class Land_Underground_Panel_Lever : Land_Underground_Panel
{
■override void SetActions()
■{
■■super.SetActions();
■■
■■AddAction(ActionUseUndergroundLever);
■}

■void ResetPhase(EntityAI target)
■{
■■target.SetAnimationPhaseNow("PanelLever", 0);
■}
■
■override void OnPanelUsedSynchronized()
■{
■■super.OnPanelUsedSynchronized();
■■SetAnimationPhase("PanelLever", 1);
■■GetGame().GetCallQueue( CALL_CATEGORY_SYSTEM ).CallLater( ResetPhase, 3000, false, this);
■■OnLEDStateChanged();
■}
■
■override void Interact()
■{
■■super.Interact();
■■GetGame().RegisterNetworkStaticObject(this);
■■GetGame().RegisterNetworkStaticObject(GetLinkedDoor());
■}
}



// ---------------------------------------------
// ------ Land_Underground_Panel_Lever.c - END ----------------------
// ---------------------------------------------


// ---------------------------------------------
// ------ Land_Underground_Stairs_Exit.c - START --------------------
// ---------------------------------------------


class Land_Underground_Stairs_Exit : Land_Underground_EntranceBase
{
■EffectSound ■■m_LockingSound;
■
■EffectSound ■■m_OpenSoundIn;
■//EffectSound ■■m_OpenSoundLoop;
■EffectSound ■■m_OpenSoundOut;

■EffectSound ■■m_CloseSoundIn;
■EffectSound ■■m_CloseSoundLoop;
■EffectSound ■■m_CloseSoundOut;

■const string LOCKING_SOUNDSET ■■■= "UndergroundDoor_Lock_SoundSet";
■const string OPENING_SOUNDSET_LOOP_IN ■= "UndergroundSmallExitDoor_Open_Start_SoundSet
■const string OPENING_SOUNDSET_LOOP ■■= "UndergroundSmallExitDoor_Open_Loop_SoundSet";
■const string OPENING_SOUNDSET_LOOP_OUT ■= "UndergroundSmallExitDoor_Open_End_SoundSe
```

```
// ----------------------------------------------
// ------ Land_Underground_WaterReservoir.c - START --------------------
// ----------------------------------------------


class WaterLevelSettings
{
■int WaterLevel;
■float Duration;
■
■void WaterLevelSettings(int pWaterLevel, float pDuration)
■{
■■WaterLevel ■= pWaterLevel;
■■Duration■= pDuration;
■}
}

class PressureLevelSettings
{
■int PressureLevel;
■float Duration;
■
■void PressureLevelSettings(int pPressureLevel, float pDuration)
■{
■■PressureLevel ■= pPressureLevel;
■■Duration■■= pDuration;
■}
}

class WaterLevelSnapshot
{
■float WaterHeight;
■float RemainingDuration;
}

class Land_Underground_WaterReservoir : BuildingBase
{
■protected const string OBJECT_NAME_WATER_PLANE ■■■■= "Land_Underground_WaterReservoir_

■protected const int WL_MIN ■■■■■■■■■= 0;
■protected const int WL_ABOVE_PIPES■■■■■■■= 1;
■protected const int WL_AVERAGE ■■■■■■■■= 2;
■protected const int WL_MAX ■■■■■■■■■= 3;
■
■protected const int PL_MIN ■■■■■■■■■= 0;
■protected const int PL_AVERAGE ■■■■■■■■= 1;
■protected const int PL_MAX ■■■■■■■■■= 2;
■
■protected const int VALVES_COUNT■■■■■■■= 2;
■protected const int VALVE_INDEX_DRAIN ■■■■■■= 0;
■protected const int VALVE_INDEX_FILL ■■■■■■= 1;
■
■protected const int PIPES_BROKEN_COUNT■■■■■■= 2;
■protected const int PIPE_INDEX_BROKEN1■■■■■■= 0;■//! main broken pipe
■protected const int PIPE_INDEX_BROKEN2■■■■■■= 1;■//! tighter broken pipe

■protected const string ANIM_PHASE_VALVE_GAUGE_DRAIN ■■= "ValveGauge1";
■protected const string ANIM_PHASE_VALVE_GAUGE_FILL ■■■= "ValveGauge2";
■protected const string ANIM_PHASE_VALVE_DRAIN ■■■■= "Valve1";
■protected const string ANIM_PHASE_VALVE_FILL ■■■■= "Valve2";
■protected const string VALVE_NAME_DRAIN ■■■■■= "valve1";
■protected const string VALVE_NAME_FILL■■ ■■■■= "valve2";
■protected const string PIPE_NAME_BROKEN1■ ■■■■= "pipe_broken_1";
```

```
// ---------------------------------------------
// ------ Land_Village_Pub.c - START --------------------
// ---------------------------------------------


class Land_Village_Pub extends BuildingWithFireplace
{
}


// ---------------------------------------------
// ------ Land_Village_Pub.c - END ----------------------
// ---------------------------------------------


// ---------------------------------------------
// ------ Lard.c - START --------------------
// ---------------------------------------------


class Lard extends Edible_Base
{
■override bool CanBeCooked()
■{
■■return true;
■}■■
■
■override bool CanBeCookedOnStick()
■{
■■return true;
■}
■
■override bool IsMeat()
■{
■■return true;
■}

■override bool CanDecay()
■{
■■return true;
■}
■
■override void SetActions()
■{
■■super.SetActions();
■■
■■AddAction(ActionForceFeed);
■■AddAction(ActionEatMeat);
■■
■■AddAction(ActionCreateIndoorFireplace);
■■AddAction(ActionCreateIndoorOven);
■}
}



// ---------------------------------------------
// ------ Lard.c - END ---------------------
// ---------------------------------------------
// ---------------------------------------------
// ------ LargeGasCannister.c - START --------------------
// ---------------------------------------------


class LargeGasCannister extends ItemBase
{■
■override bool CanExplodeInFire()
■{
■■return true;
■}
}


// ---------------------------------------------
// ------ LargeGasCannister.c - END ----------------------
// ---------------------------------------------


// ---------------------------------------------
// ------ LargeTent.c - START --------------------
// ---------------------------------------------


enum SoundTypeTent
{
■REPACK■■= 1,
■NONE■■= 0,■
}

class LargeTent extends TentBase
{
■ref protected EffectSound ■m_RepackingLoopSound;
■
■void LargeTent()
■{■■
■■m_ToggleAnimations.Insert( new ToggleAnimations("EntranceO", "EntranceC", OPENING_0), 0 );
■■m_ToggleAnimations.Insert( new ToggleAnimations("Window1O", "Window1C", OPENING_1), 0 );
■■m_ToggleAnimations.Insert( new ToggleAnimations("Window2O", "Window2C", OPENING_2), 0 );
■■m_ToggleAnimations.Insert( new ToggleAnimations("Window3O", "Window3C", OPENING_3), 0 );
■■m_ToggleAnimations.Insert( new ToggleAnimations("Window4O", "Window4C", OPENING_4), 0 );
■■m_ToggleAnimations.Insert( new ToggleAnimations("Window5O", "Window5C", OPENING_5), 0 );
■■m_ToggleAnimations.Insert( new ToggleAnimations("Window6O", "Window6C", OPENING_6), 0 );
■■m_ToggleAnimations.Insert( new ToggleAnimations("Window7O", "Window7C", OPENING_7), 0 );
■■■
■■m_ShowAnimationsWhenPitched.Insert( "Body" );
■■/*m_ShowAnimationsWhenPitched.Insert( "EntranceO" );
■■m_ShowAnimationsWhenPitched.Insert( "Window1O" );
■■m_ShowAnimationsWhenPitched.Insert( "Window2O" );
■■m_ShowAnimationsWhenPitched.Insert( "Window3O" );
■■m_ShowAnimationsWhenPitched.Insert( "Window4O" );
■■m_ShowAnimationsWhenPitched.Insert( "Window5O" );
■■m_ShowAnimationsWhenPitched.Insert( "Window6O" );
■■m_ShowAnimationsWhenPitched.Insert( "Window7O" );*/
■■m_ShowAnimationsWhenPitched.Insert( "Pack" );
■■
■■m_ShowAnimationsWhenPacked.Insert( "Inventory" );
■■
■■m_HalfExtents = Vector(2.2, 0.3, 1.9);
■}
■
■void ~LargeTent()
■{■■
```

```
// ---------------------------------------------
// ------ LargeTentBackPack.c - START --------------------
// ---------------------------------------------


class LargeTentBackPack extends Clothing
{■■
■ref protected EffectSound ■m_RepackingLoopSound;■
■
■void LargeTentBackPack()
■{
■■m_RepackingLoopSound  = new EffectSound;■
■}
■
■void ~LargeTentBackPack()
■{
■■SEffectManager.DestroyEffect( m_RepackingLoopSound );■
■}
■
■override void OnRPC(PlayerIdentity sender, int rpc_type,ParamsReadContext  ctx)
■{
■■super.OnRPC(sender, rpc_type, ctx);
■■
■■Param1<bool> p = new Param1<bool>(false);
■■■■
■■if (!ctx.Read(p))
■■■return;
■■
■■bool play = p.param1;
■■
■■switch (rpc_type)
■■{
■■■case SoundTypeTent.REPACK:
■■■
■■■■if ( play )
■■■■{
■■■■■PlayRepackingLoopSound();
■■■■}
■■■■else
■■■■{
■■■■■StopRepackingLoopSound();
■■■■}
■■■
■■■break;
■■}
■}
■
■void PlayRepackingLoopSound()
■{
■■if ( !m_RepackingLoopSound || !m_RepackingLoopSound.IsSoundPlaying() )
■■{
■■■m_RepackingLoopSound = SEffectManager.PlaySound( "largetent_deploy_SoundSet", GetPosition()
■■}
■}
■
■void StopRepackingLoopSound()
■{
■■m_RepackingLoopSound.SetSoundFadeOut(0.5);
■■m_RepackingLoopSound.SoundStop();
■}
■
■override void SetActions()
■{
```

```
// ---------------------------------------------
// ------ LaughterState.c - START --------------------
// ---------------------------------------------


class LaughterSymptom extends SymptomBase
{
■//this is just for the Symptom parameters set-up and is called even if the Symptom doesn't execute, don't
■override void OnInit()
■{
■■m_SymptomType = SymptomTypes.PRIMARY;
■■m_Priority = 0;
■■m_ID = SymptomIDs.SYMPTOM_LAUGHTER;
■■m_DestroyOnAnimFinish = true;
■■m_SyncToClient = false;
■■m_Duration = 4;
■}
■
■//!gets called every frame
■override void OnUpdateServer(PlayerBase player, float deltatime)
■{

■}

■override void OnUpdateClient(PlayerBase player, float deltatime)
■{
■■int i = 1 + 1;
■}
■
■//!gets called once on an Symptom which is being activated
■override void OnGetActivatedServer(PlayerBase player)
■{
■■if (LogManager.IsSymptomLogEnable()) Debug.SymptomLog("n/a", this.ToString(), "n/a", "OnGetActiva
■■PlaySound(EPlayerSoundEventID.SYMPTOM_LAUGHTER);
■■player.SpreadAgentsEx(3);
■}

■//!gets called once on a Symptom which is being activated
■override void OnGetActivatedClient(PlayerBase player)
■{
■■if (LogManager.IsSymptomLogEnable()) Debug.SymptomLog("n/a", this.ToString(), "n/a", "OnGetActiva
■}

■//!only gets called once on an active Symptom that is being deactivated
■override void OnGetDeactivatedServer(PlayerBase player)
■{
■■if (LogManager.IsSymptomLogEnable()) Debug.SymptomLog("n/a", this.ToString(), "n/a", "OnGetDeac
■}

■//!only gets called once on an active Symptom that is being deactivated
■override void OnGetDeactivatedClient(PlayerBase player)
■{
■■if (LogManager.IsSymptomLogEnable()) Debug.SymptomLog("n/a", this.ToString(), "n/a", "OnGetDeac
■}
}



// ---------------------------------------------
// ------ LaughterState.c - END ----------------------
// ---------------------------------------------
// ---------------------------------------------
// ------ LayoutHolder.c - START --------------------
// ---------------------------------------------


class LayoutHolder extends ScriptedWidgetEventHandler
{
■protected Widget■■■■m_MainWidget;
■protected Widget■■■■m_RootWidget;
■protected Widget■■■■m_ParentWidget;
■protected LayoutHolder■■■m_Parent;
■protected string■■■■m_LayoutName;
■
■protected bool■■■■■m_IsActive;
■protected bool■■■■■m_ImmedUpdate;
■protected bool■■■■■m_TooltipOwner;
■
■protected EntityAI■■■■m_am_entity1, m_am_entity2;

■void UpdateInterval();
■void SetLayoutName();
■
■// Override this and set m_ImmedUpdate to true if you need the widget to update on construction
■// Had to be done this way since adding it to the constructor parameters would break mods..
■void SetImmedUpdate()
■{
■■m_ImmedUpdate = false;
■}
■
■void OnSelectAction(ItemBase item, int actionId)
■{
■■PlayerBase m_player = PlayerBase.Cast( GetGame().GetPlayer() );
■■m_player.GetActionManager().OnInstantAction(ActionDebug,new Param2<ItemBase,int>(item,actionId
■}
■
■void ShowActionMenu(InventoryItem item)
■{
■■PlayerBase m_player = PlayerBase.Cast( GetGame().GetPlayer() );
■■HideOwnedTooltip();
■■m_am_entity1 = item;
■■m_am_entity2 = null;
■■ContextMenu cmenu = GetGame().GetUIManager().GetMenu().GetContextMenu();

■■cmenu.Hide();
■■cmenu.Clear();

■■if (m_am_entity1 == null)
■■■return;

■■TSelectableActionInfoArrayEx customActions = new TSelectableActionInfoArrayEx();
■■ItemBase itemBase = ItemBase.Cast(item);
■■itemBase.GetDebugActions(customActions);

■■if (ItemBase.GetDebugActionsMask() & DebugActionType.PLAYER_AGENTS)
■■{
■■■m_player.GetDebugActions(customActions);
■■}

■■int actionsCount = customActions.Count();
■■for (int i = 0; i < customActions.Count(); i++)
■■{
■■■TSelectableActionInfoWithColor actionInfo = TSelectableActionInfoWithColor.Cast(customActions.Ge
■■■if (actionInfo)
```

```
// ---------------------------------------------
// ------ LeatherBelt_ColorBase.c - START --------------------
// ---------------------------------------------


class LeatherBelt_ColorBase extends Clothing
{
■override bool CanPutInCargo( EntityAI parent )
■{
■■if( !super.CanPutInCargo( parent ) )
■■{
■■■return false;
■■}
■■
■■return IsEmpty();
■}
■
■override bool CanReceiveAttachment( EntityAI attachment,int slotId )
■{
■■if( !super.CanReceiveAttachment( attachment, slotId ) )
■■{
■■■return false;
■■}
■■
■■return !GetInventory().IsInCargo();
■}
}
class LeatherBelt_Beige extends LeatherBelt_ColorBase {};
class LeatherBelt_Natural extends LeatherBelt_ColorBase {};
class LeatherBelt_Brown extends LeatherBelt_ColorBase {};
class LeatherBelt_Black extends LeatherBelt_ColorBase {};


// ---------------------------------------------
// ------ LeatherBelt_ColorBase.c - END ----------------------
// ---------------------------------------------


// ---------------------------------------------
// ------ LeatherGloves_ColorBase.c - START --------------------
// ---------------------------------------------


class LeatherGloves_ColorBase extends Clothing {};
class LeatherGloves_Natural extends LeatherGloves_ColorBase {};
class LeatherGloves_Beige extends LeatherGloves_ColorBase {};
class LeatherGloves_Black extends LeatherGloves_ColorBase {};
class LeatherGloves_Brown extends LeatherGloves_ColorBase {};


// ---------------------------------------------
// ------ LeatherGloves_ColorBase.c - END ----------------------
// ---------------------------------------------


// ---------------------------------------------
// ------ LeatherHat_ColorBase.c - START --------------------
// ---------------------------------------------


class LeatherHat_ColorBase extends ClothingBase
{
■/*override bool CanPutAsAttachment( EntityAI parent )
```

```
// ---------------------------------------------
// ------ LeatherJacket_ColorBase.c - START --------------------
// ---------------------------------------------


class LeatherJacket_ColorBase extends Clothing {};
class LeatherJacket_Natural extends LeatherJacket_ColorBase {};
class LeatherJacket_Beige extends LeatherJacket_ColorBase {};
class LeatherJacket_Brown extends LeatherJacket_ColorBase {};
class LeatherJacket_Black extends LeatherJacket_ColorBase {};


// ---------------------------------------------
// ------ LeatherJacket_ColorBase.c - END ----------------------
// ---------------------------------------------


// ---------------------------------------------
// ------ LeatherJacket_Natural.c - START --------------------
// ---------------------------------------------


/*
class LeatherJacket_Natural extends Clothing
{
}
*/


// ---------------------------------------------
// ------ LeatherJacket_Natural.c - END ----------------------
// ---------------------------------------------


// ---------------------------------------------
// ------ LeatherMoccasinsShoes_Natural.c - START --------------------
// ---------------------------------------------


class LeatherMoccasinsShoes_Natural : Clothing
{
■override bool IsClothing()
■{
■■return true;
■}
}



// ---------------------------------------------
// ------ LeatherMoccasinsShoes_Natural.c - END ----------------------
// ---------------------------------------------


// ---------------------------------------------
// ------ LeatherPants_ColorBase.c - START --------------------
// ---------------------------------------------


class LeatherPants_ColorBase extends Clothing {};
class LeatherPants_Natural extends LeatherPants_ColorBase {};
class LeatherPants_Beige extends LeatherPants_ColorBase {};
class LeatherPants_Brown extends LeatherPants_ColorBase {};
```

```
// ---------------------------------------------
// ------ LeatherPants_Natural.c - START --------------------
// ---------------------------------------------


/*
class LeatherPants_Natural : Clothing
{
■override bool IsClothing()
■{
■■return true;
■}
}
*/


// ---------------------------------------------
// ------ LeatherPants_Natural.c - END ----------------------
// ---------------------------------------------


// ---------------------------------------------
// ------ LeatherSack_ColorBase.c - START --------------------
// ---------------------------------------------


class LeatherSack_ColorBase extends Clothing
{
■override bool IsClothing()
■{
■■return true;
■}

};

class LeatherSack_Natural extends LeatherSack_ColorBase {};
class LeatherSack_Black extends LeatherSack_ColorBase {};
class LeatherSack_Beige extends LeatherSack_ColorBase {};
class LeatherSack_Brown extends LeatherSack_ColorBase {};


// ---------------------------------------------
// ------ LeatherSack_ColorBase.c - END ----------------------
// ---------------------------------------------


// ---------------------------------------------
// ------ LeatherSack_Natural.c - START --------------------
// ---------------------------------------------


/*
class LeatherSack_Natural : Clothing
{
■override bool IsClothing()
■{
■■return true;
■}
}
*/
```

```
// ---------------------------------------------
// ------ LeatherSewingKit.c - START --------------------
// ---------------------------------------------


class LeatherSewingKit: Inventory_Base
{
■override void SetActions()
■{
■■super.SetActions();
■■AddAction(ActionRepairShelter);
■}
};



// ---------------------------------------------
// ------ LeatherSewingKit.c - END ----------------------
// ---------------------------------------------


// ---------------------------------------------
// ------ LeatherShirt_ColorBase.c - START --------------------
// ---------------------------------------------


class LeatherShirt_ColorBase extends Clothing {};
class LeatherShirt_Natural extends LeatherShirt_ColorBase {};


// ---------------------------------------------
// ------ LeatherShirt_ColorBase.c - END ----------------------
// ---------------------------------------------


// ---------------------------------------------
// ------ LeatherShoes_ColorBase.c - START --------------------
// ---------------------------------------------


class LeatherShoes_ColorBase extends Clothing {};
class LeatherShoes_Black extends LeatherShoes_ColorBase {};
class LeatherShoes_Natural extends LeatherShoes_ColorBase {};
class LeatherShoes_Brown extends LeatherShoes_ColorBase {};
class LeatherShoes_Beige extends LeatherShoes_ColorBase {};


// ---------------------------------------------
// ------ LeatherShoes_ColorBase.c - END ----------------------
// ---------------------------------------------


// ---------------------------------------------
// ------ LeatherStorageVest_ColorBase.c - START --------------------
// ---------------------------------------------


class LeatherStorageVest_ColorBase extends Clothing {};
class LeatherStorageVest_Natural extends LeatherStorageVest_ColorBase {};
class LeatherStorageVest_Beige extends LeatherStorageVest_ColorBase {};
class LeatherStorageVest_Brown extends LeatherStorageVest_ColorBase {};
class LeatherStorageVest_Black extends LeatherStorageVest_ColorBase {};
```

```
// ---------------------------------------------
// ------ LeatherStorageVest_Natural.c - START --------------------
// ---------------------------------------------


/*
class LeatherStorageVest_Natural : Clothing
{
■override bool IsClothing()
■{
■■return true;
■}
}
*/


// ---------------------------------------------
// ------ LeatherStorageVest_Natural.c - END ----------------------
// ---------------------------------------------


// ---------------------------------------------
// ------ LeftArea.c - START --------------------
// ---------------------------------------------


class LeftArea: Container
{
■protected Widget■■■■■m_UpIcon;
■protected Widget■■■■■m_DownIcon;
■protected Widget■■■■■m_ContentParent;
■protected ref VicinityContainer■■m_VicinityContainer;
■protected ScrollWidget■■■■m_ScrollWidget;
■protected ref SizeToChild■■■m_ContentResize;
■protected bool■■■■■■m_ShouldChangeSize = true;
■protected bool■■■■■■m_IsProcessing = false; // Prevents refreshing every time a child is added while
■
■void LeftArea(LayoutHolder parent )
■{
■■m_MainWidget.Show(true, false);
■■
■■m_ContentParent■= m_MainWidget.FindAnyWidget("ContentParent");
■■m_ContentParent.GetScript(m_ContentResize);
■■
■■m_ScrollWidget■= ScrollWidget.Cast(m_MainWidget.FindAnyWidget("Scroller"));
■■m_MainWidget■= m_MainWidget.FindAnyWidget("Content");
■■
■■m_UpIcon■■= m_RootWidget.FindAnyWidget("Up");
■■m_DownIcon■■= m_RootWidget.FindAnyWidget("Down");
■■
■■m_VicinityContainer = new VicinityContainer(this, false);
■■m_Body.Insert(m_VicinityContainer);
■■m_ActiveIndex = 0;
■■
■■WidgetEventHandler.GetInstance().RegisterOnChildAdd(m_MainWidget, this, "OnChildAdd");
■■WidgetEventHandler.GetInstance().RegisterOnChildRemove(m_MainWidget, this, "OnChildRemove");
■■RecomputeOpenedContainers();
■}
■
■override void UnfocusGrid()
■{
■■Container active_container;
■■for ( int i = 0; i < m_OpenedContainers.Count(); i++ )
```

```
// ---------------------------------------------
// ------ LegCover_improvised.c - START --------------------
// ---------------------------------------------


class LegsCover_Improvised extends Clothing
{
■override void SetActions()
■{
■■super.SetActions();
■■AddAction(ActionWringClothes);
■}
};




// ---------------------------------------------
// ------ LegCover_improvised.c - END ----------------------
// ---------------------------------------------


// ---------------------------------------------
// ------ Legs.c - START --------------------
// ---------------------------------------------


class MaleLegs_Base extends InventoryItem {};
class FemaleLegs_Base extends InventoryItem {};

class MaleAdamLegs extends MaleLegs_Base {};
class MaleBorisLegs extends MaleLegs_Base {};
class MaleCyrilLegs extends MaleLegs_Base {};
class MaleDenisLegs extends MaleLegs_Base {};
class MaleEliasLegs extends MaleLegs_Base {};
class MaleFrancisLegs extends MaleLegs_Base {};
class MaleGuoLegs extends MaleLegs_Base {};
class MaleHassanLegs extends MaleLegs_Base {};
class MaleIndarLegs extends MaleLegs_Base {};
class MaleJoseLegs extends MaleLegs_Base {};
class MaleKaitoLegs extends MaleLegs_Base {};
class MaleLewisLegs extends MaleLegs_Base {};
class MaleManuaLegs extends MaleLegs_Base {};
class MaleNikiLegs extends MaleLegs_Base {};
class MaleOliverLegs extends MaleLegs_Base {};
class MalePeterLegs extends MaleLegs_Base {};
class MaleQuinnLegs extends MaleLegs_Base {};
class MaleRolfLegs extends MaleLegs_Base {};
class MaleSethLegs extends MaleLegs_Base {};
class MaleTaikiLegs extends MaleLegs_Base {};

class FemaleEvaLegs extends FemaleLegs_Base {};
class FemaleFridaLegs extends FemaleLegs_Base {};
class FemaleGabiLegs extends FemaleLegs_Base {};
class FemaleHelgaLegs extends FemaleLegs_Base {};
class FemaleIrenaLegs extends FemaleLegs_Base {};
class FemaleJudyLegs extends FemaleLegs_Base {};
class FemaleKeikoLegs extends FemaleLegs_Base {};
class FemaleLindaLegs extends FemaleLegs_Base {};
class FemaleMariaLegs extends FemaleLegs_Base {};
class FemaleNaomiLegs extends FemaleLegs_Base {};


// ---------------------------------------------
// ---------------------------------------------
// ------ LifespanLevel.c - START --------------------
// ---------------------------------------------


class LifespanLevel
{
■protected int m_Level;
■protected float m_Threshold;
■protected string m_TextureName;
■protected string m_MaterialName;


■void LifespanLevel( int level, float threshold, string texture_name, string material_name )
■{
■■m_Level = level;
■■m_Threshold = threshold;
■■m_TextureName = texture_name;
■■m_MaterialName = material_name;
■}

■int GetLevel()
■{
■■return m_Level;
■}

■float GetThreshold()
■{
■■return m_Threshold;
■}

■string GetTextureName()
■{
■■return m_TextureName;
■}

■string GetMaterialName()
■{
■■return m_MaterialName;
■}
}


// ---------------------------------------------
// ------ LifespanLevel.c - END ----------------------
// ---------------------------------------------


// ---------------------------------------------
// ------ LightAI.c - START --------------------
// ---------------------------------------------


//-----------------------------------------------------------------------------
class Animal extends LightAISuper
{
};

//-----------------------------------------------------------------------------
class Zombie extends LightAISuper
{
};
```

```
// ---------------------------------------------
// ------ LightAIBase.c - START --------------------
// ---------------------------------------------


class LightAIBase extends LightAI
{

};


// ---------------------------------------------
// ------ LightAIBase.c - END ----------------------
// ---------------------------------------------


// ---------------------------------------------
// ------ Liquid.c - START --------------------
// ---------------------------------------------


class Liquid
{
■static ref map<int, ref NutritionalProfile> m_AllLiquidsByType = new map<int, ref NutritionalProfile>;
■static ref map<string, ref NutritionalProfile> m_AllLiquidsByName = new map<string, ref NutritionalProfile
■
■static bool m_Init = InitAllLiquids();
■
■static string GetLiquidClassname(int liquid_type)
■{
■■NutritionalProfile liquid = m_AllLiquidsByType.Get(liquid_type);
■■if (liquid)
■■{
■■■return liquid.GetLiquidClassname();
■■}
■■
■■return "";
■}
■
■static bool InitAllLiquids()
■{
■■string cfg_classname = "cfgLiquidDefinitions";
■■string property_value = "NULL_PROPERTY";
■■int cfg_item_count = g_Game.ConfigGetChildrenCount(cfg_classname);

■■for ( int i = 0; i < cfg_item_count; i++ )
■■{
■■■string liquid_class_name;
■■■GetGame().ConfigGetChildName(cfg_classname, i, liquid_class_name);
■■■string liquid_full_path = string.Format("%1 %2",cfg_classname, liquid_class_name);
■■■int config_liquid_type = GetGame().ConfigGetInt( string.Format("%1 type", liquid_full_path) );
■■■m_AllLiquidsByType.Insert(config_liquid_type, SetUpNutritionalProfile(config_liquid_type, liquid_class
■■■m_AllLiquidsByName.Insert(liquid_class_name, SetUpNutritionalProfile(config_liquid_type, liquid_cla
■■■
■■}
■■return true;
■}
■
■//-----------------------------------------------------------------------------------------------
■static void Transfer(ItemBase source_ent, ItemBase target_ent, float quantity = -1)
■{
■■if ( !Liquid.CanTransfer(source_ent, target_ent) ) return;
■■
```

```
// ---------------------------------------------
// ------ LoadingMenu.c - START --------------------
// ---------------------------------------------


class LoadingMenu extends UIScriptedMenu
{
■void LoadingMenu()
■{
■}

■void ~LoadingMenu()
■{
■}
■
■override Widget Init()
■{
■■layoutRoot = GetGame().GetWorkspace().CreateWidgets("gui/layouts/loading.layout");
■■
■■m_label = TextWidget.Cast( layoutRoot.FindAnyWidget("TextWidget") );
■■m_progressBar = ProgressBarWidget.Cast( layoutRoot.FindAnyWidget("ProgressBarWidget") );
■■m_image = ImageWidget.Cast( layoutRoot.FindAnyWidget("ImageBackground") );
■■
■■m_image.LoadImageFile( 0, GetRandomLoadingBackground() );
■■layoutRoot.FindAnyWidget("notification_root").Show(false);
■■
■■#ifdef PLATFORM_CONSOLE
■■#ifdef PLATFORM_XBOX
■■#ifdef BUILD_EXPERIMENTAL
■■■Widget exp_notifiaction = layoutRoot.FindAnyWidget("notification_root");
■■■if (exp_notifiaction)
■■■{
■■■■exp_notifiaction.Show(true);
■■■}
■■#endif
■■#endif
■■#endif
■■
■■return layoutRoot;
■}

■TextWidget m_label;
■ProgressBarWidget m_progressBar;
■ImageWidget m_image;
}



// ---------------------------------------------
// ------ LoadingMenu.c - END ----------------------
// ---------------------------------------------


// ---------------------------------------------
// ------ LoadMagazine.c - START --------------------
// ---------------------------------------------


class LoadMagazine extends RecipeBase
{■
■override void Init()
■{
■■m_Name = "#load_magazine";
```

```
// ---------------------------------------------
// ------ Lockpick.c - START --------------------
// ---------------------------------------------


class Lockpick: ToolBase
{
■void Lockpick()
■{
■■m_MineDisarmRate = 90;
■}
■
■override void SetActions()
■{
■■super.SetActions();
■■
■■AddAction(ActionUnrestrainTarget);
■■AddAction(ActionLockDoors);
■■AddAction(ActionUnlockDoors);
■■AddAction(ActionDisarmMine);
■■AddAction(ActionDisarmExplosive);
■}
};



// ---------------------------------------------
// ------ Lockpick.c - END ----------------------
// ---------------------------------------------


// ---------------------------------------------
// ------ LogoutMenu.c - START --------------------
// ---------------------------------------------



class LogoutMenu extends UIScriptedMenu
{■
■private TextWidget m_LogoutTimeText;
■private TextWidget m_DescriptionText;
■private ButtonWidget m_bLogoutNow;
■private ButtonWidget m_bCancel;
■private ButtonWidget m_bCancelConsole;
■private int m_iTime;

■void LogoutMenu()
■{
■■m_iTime = 0;
■■g_Game.SetKeyboardHandle(this);
■}

■void ~LogoutMenu()
■{
■■g_Game.SetKeyboardHandle(NULL);
■}
■
■override Widget Init()
■{
■■layoutRoot = GetGame().GetWorkspace().CreateWidgets("gui/layouts/day_z_logout_dialog.layout");
■■
■■m_LogoutTimeText ■= TextWidget.Cast(layoutRoot.FindAnyWidget("txtLogoutTime"));
■■m_DescriptionText ■= TextWidget.Cast(layoutRoot.FindAnyWidget("txtDescription"));
```

```
// ---------------------------------------------
// ------ LogTemplates.c - START --------------------
// ---------------------------------------------


typedef Param3<string, string, string> LogTemplate;
typedef int LogTemplateID;

class LogTemplates
{
■static private ref map<LogTemplateID, ref LogTemplate> m_LogTemplates;
■
■static private void■RegisterLogTamplate(LogTemplateID template_id, string author, string plugin, string
■{
■■if ( m_LogTemplates == NULL )
■■{
■■■m_LogTemplates■= new map<LogTemplateID, ref LogTemplate>;
■■}
■■
■■if ( m_LogTemplates.Contains(template_id) )
■■{
■■■Debug.Log("Template ID: "+string.ToString(template_id)+" is alredy exist!", "LogTemplate.h -> OnInit
■■}
■■else
■■{
■■■LogTemplate params = new LogTemplate(author, plugin, label);
■■■m_LogTemplates.Set(template_id, params);
■■}
■}
■
■// Steps to register of new log template:
■// 1.) Crete new LogTemplateID in below of this comment.
■// 2.) Set Template Name in format TEMPLATE_[CUSTOM NAME] => TEMPLATE_MY_LOG
■// 3.) Set Template ID which is your id + your custom id =>  + CustomID(0) = 50190
■////////////////////////////////////////////////////////////
■//■■■ ■■■Template Name■■■■Template ID
■static LogTemplateID■TEMPLATE_UNKNOWN■■■= 0;
■static LogTemplateID■TEMPLATE_JANOSIK■■■= 1;
■static LogTemplateID■TEMPLATE_PLAYER_WEIGHT■■= 2;
■static LogTemplateID■TEMPLATE_BROADCAST■■■= 3;
■
■static void Init()
■{
■■///////////////////////////////////////////////////////////////////////////////////////////////////////////////
■■////////Register Log templates///////////////////////////////////////////////////////////////////////////////
■■//■■■■■■| Template Name■■■| author■■■| plugin■■■■| label■■■■■////
■■RegisterLogTamplate(■TEMPLATE_UNKNOWN■■,"Unknown"■■■,"Unknown"■■■■,"Unknown"■■
■■RegisterLogTamplate(■TEMPLATE_JANOSIK■■,"Janosik"■■■,"GUI"■■■■■,"None"■■■■■);//
■■RegisterLogTamplate(■TEMPLATE_PLAYER_WEIGHT■,"Unknown"■■■,"PlayerBase"■■■,"Weight
■■RegisterLogTamplate(■TEMPLATE_BROADCAST■■,"Unknown"■■■,"PluginMessageManager"■,"E
■■
■}
■
■static LogTemplate GetTemplate(LogTemplateID template_id)
■{
■■if ( m_LogTemplates && m_LogTemplates.Contains(template_id) )
■■{
■■■return m_LogTemplates.Get(template_id);
■■}
■■
■■Debug.Log("Template ID: "+string.ToString(template_id)+" does not exist!", "LogTemplate.h -> GetTem
■■return NULL;
■}
```

```
// ---------------------------------------------
// ------ LongHorn.c - START --------------------
// ---------------------------------------------


class LongHorn_Base : SingleShotPistol_Base
{
■override RecoilBase SpawnRecoilObject()
■{
■■return new LongHornRecoil(this);
■}
■
■//Debug menu Spawn Ground Special
■override void OnDebugSpawn()
■{
■■super.OnDebugSpawn();
■■GetInventory().CreateInInventory( "PistolOptic" );
■}■
};

class LongHorn : LongHorn_Base {};


// ---------------------------------------------
// ------ LongHorn.c - END ----------------------
// ---------------------------------------------


// ---------------------------------------------
// ------ LongHornRecoil.c - START --------------------
// ---------------------------------------------


class LongHornRecoil: RecoilBase
{
■
■override void Init()
■{
■■vector point_1;
■■vector point_2;
■■vector point_3;
■■vector point_4;
■■point_1[0] = m_Player.GetRandomGeneratorSyncManager().GetRandomInRange(RandomGeneratorS
■■point_2[0] = m_Player.GetRandomGeneratorSyncManager().GetRandomInRange(RandomGeneratorS
■■point_3[0] = m_Player.GetRandomGeneratorSyncManager().GetRandomInRange(RandomGeneratorS
■■point_4[0] = m_Player.GetRandomGeneratorSyncManager().GetRandomInRange(RandomGeneratorS
■■m_HandsCurvePoints.Insert(point_1);//forms a 2 dimensional spline(z is ignored)
■■m_HandsCurvePoints.Insert(point_2);
■■m_HandsCurvePoints.Insert(point_3);
■■m_HandsCurvePoints.Insert(point_4);
■■m_HandsCurvePoints.Insert("0 0 0");
■■m_HandsOffsetRelativeTime = 0.125;
■■
■■m_MouseOffsetRangeMin = 70;//in degrees min
■■m_MouseOffsetRangeMax = 110;//in degrees max
■■m_MouseOffsetDistance = 4.0;//how far should the mouse travel
■■m_MouseOffsetRelativeTime = 0.0625; //0.03;//[0..1] a time it takes to move the mouse the required dis
■■■■
■■m_CamOffsetDistance = 0.05;
■■m_CamOffsetRelativeTime = 0.5;
■}
}
```

```
// ---------------------------------------------
// ------ LongTorch.c - START --------------------
// ---------------------------------------------


class LongTorch : Torch
{
■void LongTorch()
■{
■■m_DecraftResult = "LongWoodenStick";
■■m_ParticleLocalPos = Vector(0, 0.83, 0);
■}
■
■override void SetActions()
■{
■■super.SetActions();
■■
■■AddAction(ActionClapBearTrapWithThisItem);
■}
};


// ---------------------------------------------
// ------ LongTorch.c - END ----------------------
// ---------------------------------------------


// ---------------------------------------------
// ------ LugWrench.c - START --------------------
// ---------------------------------------------


class LugWrench: Inventory_Base
{
■override void SetActions()
■{
■■super.SetActions();
■■
■■AddAction(ActionLockAttachment);
■}
};



// ---------------------------------------------
// ------ LugWrench.c - END ----------------------
// ---------------------------------------------


// ---------------------------------------------
// ------ M16A2.c - START --------------------
// ---------------------------------------------


class M16A2_Base : RifleBoltLock_Base
{
■override RecoilBase SpawnRecoilObject()
■{
■■return new M16A2Recoil(this);
■}■■
■■
■//Debug menu Spawn Ground Special
■override void OnDebugSpawn()
```

```
// ---------------------------------------------
// ------ M16A2Recoil.c - START --------------------
// ---------------------------------------------


class M16A2Recoil: RecoilBase
{
■override void Init()
■{
■■vector point_1;
■■vector point_2;
■■vector point_3;
■■vector point_4;
■■point_1[0] = m_Player.GetRandomGeneratorSyncManager().GetRandomInRange(RandomGeneratorS
■■point_2[0] = m_Player.GetRandomGeneratorSyncManager().GetRandomInRange(RandomGeneratorS
■■point_3[0] = m_Player.GetRandomGeneratorSyncManager().GetRandomInRange(RandomGeneratorS
■■point_4[0] = m_Player.GetRandomGeneratorSyncManager().GetRandomInRange(RandomGeneratorS
■■m_HandsCurvePoints.Insert(point_1);//forms a 2 dimensional spline(z is ignored)
■■m_HandsCurvePoints.Insert(point_2);
■■m_HandsCurvePoints.Insert(point_3);
■■m_HandsCurvePoints.Insert(point_4);
■■m_HandsCurvePoints.Insert("0 0 0");
■■m_HandsOffsetRelativeTime = 1;
■■
■■m_MouseOffsetRangeMin = 50;//in degrees min
■■m_MouseOffsetRangeMax = 120;//in degrees max
■■m_MouseOffsetDistance = 1.0;//how far should the mouse travel
■■m_MouseOffsetRelativeTime = 0.5;//[0..1] a time it takes to move the mouse the required distance rela
■
■■m_CamOffsetDistance = 0.01;
■■m_CamOffsetRelativeTime = 1;
■}
}


// ---------------------------------------------
// ------ M16A2Recoil.c - END ----------------------
// ---------------------------------------------


// ---------------------------------------------
// ------ M18SmokeGrenade_ColorBase.c - START --------------------
// ---------------------------------------------


class M18SmokeGrenade_ColorBase extends SmokeGrenadeBase
{
■const string SOUND_SMOKE_START = "SmokegGrenades_M18_start_loop_SoundSet";
■const string SOUND_SMOKE_LOOP = "SmokegGrenades_M18_active_loop_SoundSet";
■const string SOUND_SMOKE_END = "SmokegGrenades_M18_end_loop_SoundSet";

■void M18SmokeGrenade_ColorBase()
■{
■■SetAmmoType("");
■■SetFuseDelay(2);
■■SetSoundSmokeStart(SOUND_SMOKE_START);
■■SetSoundSmokeLoop(SOUND_SMOKE_LOOP);
■■SetSoundSmokeEnd(SOUND_SMOKE_END);
■}
■
■void ~M18SmokeGrenade_ColorBase() {}
}
```

```
// ---------------------------------------------
// ------ M249.c - START --------------------
// ---------------------------------------------


class M249 : Weapon_Base
{
};


// ---------------------------------------------
// ------ M249.c - END ----------------------
// ---------------------------------------------


// ---------------------------------------------
// ------ M249_Hndgrd.c - START --------------------
// ---------------------------------------------


class M249_Hndgrd extends Inventory_Base
{
/*
■override bool CanDetachAttachment( EntityAI attachment )
■{
■■return false;
■}
*/
}


// ---------------------------------------------
// ------ M249_Hndgrd.c - END ----------------------
// ---------------------------------------------


// ---------------------------------------------
// ------ M249_RisHndgrd.c - START --------------------
// ---------------------------------------------


class M249_RisHndgrd extends Inventory_Base
{
/*
■override bool CanDetachAttachment( EntityAI attachment )
■{
■■return false;
■}
*/
}


// ---------------------------------------------
// ------ M249_RisHndgrd.c - END ----------------------
// ---------------------------------------------


// ---------------------------------------------
// ------ M4A1.c - START --------------------
// ---------------------------------------------


class M4A1_Base : RifleBoltLock_Base
```

```
// ----------------------------------------------
// ------ M4a1Recoil.c - START --------------------
// ----------------------------------------------


class M4a1Recoil: RecoilBase
{
■override void Init()
■{
■■vector point_1;
■■vector point_2;
■■vector point_3;
■■vector point_4;
■■point_1[0] = m_Player.GetRandomGeneratorSyncManager().GetRandomInRange(RandomGeneratorS
■■point_2[0] = m_Player.GetRandomGeneratorSyncManager().GetRandomInRange(RandomGeneratorS
■■point_3[0] = m_Player.GetRandomGeneratorSyncManager().GetRandomInRange(RandomGeneratorS
■■point_4[0] = m_Player.GetRandomGeneratorSyncManager().GetRandomInRange(RandomGeneratorS
■■m_HandsCurvePoints.Insert(point_1);//forms a 2 dimensional spline(z is ignored)
■■m_HandsCurvePoints.Insert(point_2);
■■m_HandsCurvePoints.Insert(point_3);
■■m_HandsCurvePoints.Insert(point_4);
■■m_HandsCurvePoints.Insert("0 0 0");
■■m_HandsOffsetRelativeTime = 1;
■■
■■m_MouseOffsetRangeMin = 50;//in degrees min
■■m_MouseOffsetRangeMax = 120;//in degrees max
■■m_MouseOffsetDistance = 1.4;//how far should the mouse travel
■■m_MouseOffsetRelativeTime = 0.5;//[0..1] a time it takes to move the mouse the required distance rela
■
■■m_CamOffsetDistance = 0.01;
■■m_CamOffsetRelativeTime = 1;
■}
}


// ----------------------------------------------
// ------ M4a1Recoil.c - END ----------------------
// ----------------------------------------------


// ----------------------------------------------
// ------ M4_CarryHandleOptic.c - START --------------------
// ----------------------------------------------


class M4_CarryHandleOptic extends ItemOptics
{
■override bool CanPutAsAttachment( EntityAI parent )
■{
■■if(!super.CanPutAsAttachment(parent)) {return false;}
■■if ( parent.IsKindOf("M4A1_Base") )
■■{
■■■return true;
■■}

■■return false;
■}
}


// ----------------------------------------------
// ------ M4_CarryHandleOptic.c - END ----------------------
// ----------------------------------------------
// ---------------------------------------------
// ------ M4_MPHndgrd.c - START --------------------
// ---------------------------------------------


class M4_MPHndgrd extends Inventory_Base
{
/*
■override bool CanDetachAttachment( EntityAI attachment )
■{
■■return false;
■}
*/
}


// ---------------------------------------------
// ------ M4_MPHndgrd.c - END ----------------------
// ---------------------------------------------


// ---------------------------------------------
// ------ M4_PlasticHndgrd.c - START --------------------
// ---------------------------------------------


class M4_PlasticHndgrd extends Inventory_Base
{
/*
■override bool CanDetachAttachment( EntityAI attachment )
■{
■■return false;
■}
*/
}


// ---------------------------------------------
// ------ M4_PlasticHndgrd.c - END ----------------------
// ---------------------------------------------


// ---------------------------------------------
// ------ M4_RISHndgrd.c - START --------------------
// ---------------------------------------------


class M4_RISHndgrd extends Inventory_Base
{
■override bool CanDetachAttachment (EntityAI parent)
■{
■■if ( Weapon_Base.Cast(parent) && parent.FindAttachmentBySlotName("weaponFlashlight") )
■■{
■■■return false;
■■}
■■return true;
■}
}


// ---------------------------------------------
// ------ M4_RISHndgrd.c - END ----------------------
// ---------------------------------------------
// ---------------------------------------------
// ------ M4_Suppressor.c - START --------------------
// ---------------------------------------------


class M4_Suppressor extends SuppressorBase
{
■override bool CanPutAsAttachment( EntityAI parent )
■{
■■if(!super.CanPutAsAttachment(parent)) {return false;}
■■if ( parent.FindAttachmentBySlotName("suppressorImpro") == NULL && parent.FindAttachmentBySlot
■■{
■■■return true;
■■}
■■return false;
■}
}


// ---------------------------------------------
// ------ M4_Suppressor.c - END ----------------------
// ---------------------------------------------


// ---------------------------------------------
// ------ M65Jacket_ColorBase.c - START --------------------
// ---------------------------------------------


class M65Jacket_ColorBase extends Clothing
{
■override void SetActions()
■{
■■super.SetActions();
■■AddAction(ActionWringClothes);
■}
};
class M65Jacket_Black extends M65Jacket_ColorBase {};
class M65Jacket_Khaki extends M65Jacket_ColorBase {};
class M65Jacket_Tan extends M65Jacket_ColorBase {};
class M65Jacket_Olive extends M65Jacket_ColorBase {};


// ---------------------------------------------
// ------ M65Jacket_ColorBase.c - END ----------------------
// ---------------------------------------------


// ---------------------------------------------
// ------ M67Grenade.c - START --------------------
// ---------------------------------------------


class M67Grenade extends Grenade_Base
{
■void M67Grenade()
■{
■■SetAmmoType("M67Grenade_Ammo");
■■SetFuseDelay(4);
■■SetParticleExplosion(ParticleList.M67);
■}

■void ~M67Grenade() {}
```

```
// ---------------------------------------------
// ------ M79.c - START --------------------
// ---------------------------------------------


class M79_Base extends RifleSingleShotManual_Base
{
■void M79_Base ()
■{
■}
■
■override bool ShootsExplosiveAmmo()
■{
■■return true;
■}
■
■override RecoilBase SpawnRecoilObject()
■{
■■return new M79Recoil(this);
■}
};

class M79 extends M79_Base
{
■override void AssembleGun()
■{
■■super.AssembleGun();
■■
■■if ( !FindAttachmentBySlotName("weaponOpticsAug") )
■■{
■■■GetInventory().CreateAttachment("M79DummyOptics");
■■}
■}
};

class M79DummyOptics extends ItemOptics
{
■override bool CanPutAsAttachment( EntityAI parent )
■{
■■return true;
■}
};




// ---------------------------------------------
// ------ M79.c - END ----------------------
// ---------------------------------------------


// ---------------------------------------------
// ------ M79Recoil.c - START --------------------
// ---------------------------------------------


class M79Recoil: RecoilBase
{
■override void Init()
■{
■■vector point_1;
■■vector point_2;
■■vector point_3;
```

```
// ---------------------------------------------
// ------ M9A1_Bayonet.c - START --------------------
// ---------------------------------------------


class M9A1_Bayonet extends ToolBase
{
■override bool IsMeleeFinisher()
■{
■■return true;
■}
■
■override array<int> GetValidFinishers()
■{
■■return {EMeleeHitType.FINISHER_LIVERSTAB,EMeleeHitType.FINISHER_NECKSTAB};
■}
■
■override bool CanPutAsAttachment( EntityAI parent )
■{
■■if(!super.CanPutAsAttachment(parent)) {return false;}
■■if ( parent.FindAttachmentBySlotName("suppressorImpro") == null && parent.FindAttachmentBySlotNa
■■{
■■■return true;
■■}
■■return false;
■}

■override void OnWasAttached(EntityAI parent, int slot_id)
■{
■■super.OnWasAttached(parent, slot_id);
■■
■■if( parent.IsWeapon() )
■■{
■■■parent.SetBayonetAttached(true,slot_id);
■■}■■
■}
■
■override void OnWasDetached(EntityAI parent, int slot_id)
■{
■■super.OnWasDetached(parent, slot_id);
■■
■■if( parent.IsWeapon() )
■■{
■■■parent.SetBayonetAttached(false);
■■}
■}
■
■override void SetActions()
■{
■■super.SetActions();
■■
■■AddAction(ActionBurnSewTarget);
■■AddAction(ActionUnrestrainTarget);
■■AddAction(ActionSkinning);
■■AddAction(ActionMineBush);
■■AddAction(ActionMineTreeBark);
■■AddAction(ActionBurnSewSelf);
■■AddAction(ActionDigWorms);
■■AddAction(ActionShaveTarget);
■■AddAction(ActionDisarmMine);
■■AddAction(ActionDisarmExplosive);
■■AddAction(ActionShave);
■}
```

```
// ---------------------------------------------
// ------ Mace.c - START --------------------
// ---------------------------------------------


class Mace: Inventory_Base {};



// ---------------------------------------------
// ------ Mace.c - END ----------------------
// ---------------------------------------------


// ---------------------------------------------
// ------ Machete.c - START --------------------
// ---------------------------------------------


class Machete extends ToolBase
{
■void Machete()
■{
■}

■override bool IsMeleeFinisher()
■{
■■return true;
■}
■
■override array<int> GetValidFinishers()
■{
■■return {EMeleeHitType.FINISHER_LIVERSTAB,EMeleeHitType.FINISHER_NECKSTAB};
■}
■
■override void SetActions()
■{
■■super.SetActions();
■■AddAction(ActionUnrestrainTarget);
■■AddAction(ActionBurnSewTarget);
■■AddAction(ActionSkinning);
■■AddAction(ActionMineBush);
■■AddAction(ActionMineTreeBark);
■■AddAction(ActionBurnSewSelf);
■■AddAction(ActionDigWorms);
■}
}


// ---------------------------------------------
// ------ Machete.c - END ----------------------
// ---------------------------------------------


// ---------------------------------------------
// ------ Mackerel.c - START --------------------
// ---------------------------------------------


class Mackerel extends Edible_Base
{
■override bool CanBeCookedOnStick()
■{
```

```
// ---------------------------------------------
// ------ MackerelFilletMeat.c - START --------------------
// ---------------------------------------------


class MackerelFilletMeat extends Edible_Base
{
■override bool CanBeCooked()
■{
■■return true;
■}■■
■
■override bool CanBeCookedOnStick()
■{
■■return true;
■}
■
■override bool IsMeat()
■{
■■return true;
■}

■override bool CanDecay()
■{
■■return true;
■}
■
■override void SetActions()
■{
■■super.SetActions();
■■
■■AddAction(ActionForceFeed);
■■AddAction(ActionEatMeat);
■■
■■AddAction(ActionCreateIndoorFireplace);
■■AddAction(ActionCreateIndoorOven);
■}
}



// ---------------------------------------------
// ------ MackerelFilletMeat.c - END ----------------------
// ---------------------------------------------


// ---------------------------------------------
// ------ MacrolepiotaMushroom.c - START --------------------
// ---------------------------------------------


class MacrolepiotaMushroom : MushroomBase
{
■override void SetActions()
■{
■■super.SetActions();
■■
■■AddAction(ActionCreateIndoorFireplace);
■■AddAction(ActionCreateIndoorOven);
■}
}
```

```
// ---------------------------------------------
// ------ Magazine.c - START --------------------
// ---------------------------------------------


typedef Magazine Magazine_Base;

enum CartridgeType
{
■None = 0,
■Pistol = 1,
■Intermediate = 2,
■FullPower = 3,
■Shell = 4
}

enum ProjectileType
{
■None = 0,
■Tracer = 1,
■AP = 2
}


class AmmoData
{
■bool m_IsValid;
■CartridgeType m_CartridgeType;
■ProjectileType m_ProjectileType;
■
■void AmmoData( string init_type )
■{
■■m_IsValid = GetGame().ConfigIsExisting( "CfgMagazines " + init_type );
■■if ( m_IsValid )
■■{
■■■m_CartridgeType = GetGame().ConfigGetInt( "CfgMagazines " + init_type + " iconCartridge" );
■■■m_ProjectileType = GetGame().ConfigGetInt( "CfgMagazines " + init_type + " iconType" );
■■}
■}
}

class Magazine : InventoryItemSuper
{
■protected static ref map<string, ref AmmoData>■m_AmmoData;
■ref array<string>■■■■■■■■m_CompatiableAmmo;
■ref array<float> ■■■■■■■■m_ChanceToJam;
■protected float ■■■■■■■■m_ManipulationDamage;
■
■void Magazine ()
■{
■■m_ChanceToJam = new array<float>;
■■InitReliability(m_ChanceToJam);
■■m_ManipulationDamage = ConfigGetFloat("manipulationDamage");
■■m_CompatiableAmmo = new array<string>;
■■ConfigGetTextArray("ammoItems", m_CompatiableAmmo);
■■if ( !GetGame().IsDedicatedServer() )
■■{
■■■if ( !m_AmmoData )
■■■■m_AmmoData = new map<string, ref AmmoData>;
■■■
■■■string classname = ClassName();
■■■if ( !m_AmmoData.Contains(classname) )
■■■{
```

```
// ---------------------------------------------
// ------ MagazineHide.c - START --------------------
// ---------------------------------------------


class MagazineHide extends WeaponStateBase
{
■override void OnEntry (WeaponEventBase e)
■{
■■super.OnEntry(e);
■■//m_weapon.HideMagazine();
■}

■override void OnExit (WeaponEventBase e)
■{
■■super.OnExit(e);
■}
};

class MagazineHide_W4T extends MagazineHide
{
■override bool IsWaitingForActionFinish () { return true; }
};




// ---------------------------------------------
// ------ MagazineHide.c - END ----------------------
// ---------------------------------------------


// ---------------------------------------------
// ------ Magazines.c - START --------------------
// ---------------------------------------------


class Mag_FNX45_15Rnd: MagazineStorage {};
class Mag_357Speedloader_6Rnd: MagazineStorage {};
class Mag_Deagle_9rnd: MagazineStorage {};
class Mag_1911_7Rnd: MagazineStorage {};
class Mag_CZ75_15Rnd: MagazineStorage {};
class Mag_Glock_15Rnd: MagazineStorage {};
class Mag_P1_8Rnd: MagazineStorage {};
class Mag_IJ70_8Rnd: MagazineStorage {};
class Mag_MP5_15Rnd: MagazineStorage {};
class Mag_MP5_30Rnd: MagazineStorage {};
class Mag_PM73_15Rnd: MagazineStorage {};
class Mag_PM73_25Rnd: MagazineStorage {};
class Mag_CZ61_20Rnd: MagazineStorage {};
class Mag_MKII_10Rnd: MagazineStorage {};
class Mag_ShockCartridge: MagazineStorage {};
class Mag_Ruger1022_10Rnd: MagazineStorage {};
class Mag_Ruger1022_15Rnd: MagazineStorage {};
class Mag_Ruger1022_30Rnd: MagazineStorage {};
class Mag_CLIP762x54_5Rnd: MagazineStorage {};
class Mag_762x54Snaploader_2Rnd: MagazineStorage {};
class Mag_308WinSnaploader_2Rnd: MagazineStorage {};
class Mag_CLIP762x39_10Rnd: MagazineStorage {};
class Mag_CLIP9x19_10Rnd: MagazineStorage {};
class Mag_AKM_30Rnd: MagazineStorage {};
class Mag_AKM_Drum75Rnd: MagazineStorage {};
class Mag_AKM_Palm30Rnd: MagazineStorage {};
```

```
// ---------------------------------------------
// ------ MagazineShow.c - START --------------------
// ---------------------------------------------


class MagazineShow extends WeaponStateBase
{
■override void OnEntry (WeaponEventBase e)
■{
■■super.OnEntry(e);
■■if (e)
■■■m_weapon.ShowMagazine();
■}

■override void OnExit (WeaponEventBase e)
■{
■■super.OnExit(e);
■}
};

class MagazineShow_W4T extends MagazineShow
{
■override bool IsWaitingForActionFinish () { return true; }
};




// ---------------------------------------------
// ------ MagazineShow.c - END ----------------------
// ---------------------------------------------


// ---------------------------------------------
// ------ Magnum.c - START --------------------
// ---------------------------------------------


const float MAGNUM_ROTATION_POSITION_M1 = -0.167;
const float MAGNUM_ROTATION_POSITION_0 = 0.0;
const float MAGNUM_ROTATION_POSITION_1 = 0.167;
const float MAGNUM_ROTATION_POSITION_2 = 0.334;
const float MAGNUM_ROTATION_POSITION_3 = 0.500;
const float MAGNUM_ROTATION_POSITION_4 = 0.668;
const float MAGNUM_ROTATION_POSITION_5 = 0.835;
const float MAGNUM_ROTATION_POSITION_6 = 1.0;


enum MagnumAnimState
{
■DEFAULT ■■■= 0, ■///< default weapon state
};

enum MagnumStableStateID
{
■UNKNOWN■■■■=  0,
■DEFAULT■■■■=  1,
}

class Magnum_Static_State extends WeaponStableState
{
■bool init = false;
■override void OnEntry(WeaponEventBase e)
```

```
// ---------------------------------------------
// ------ MagnumRecoil.c - START --------------------
// ---------------------------------------------


class MagnumRecoil: RecoilBase
{
■override void Init()
■{
■■vector point_1;
■■vector point_2;
■■vector point_3;
■■vector point_4;
■■point_1[0] = m_Player.GetRandomGeneratorSyncManager().GetRandomInRange(RandomGeneratorS
■■point_2[0] = m_Player.GetRandomGeneratorSyncManager().GetRandomInRange(RandomGeneratorS
■■point_3[0] = m_Player.GetRandomGeneratorSyncManager().GetRandomInRange(RandomGeneratorS
■■point_4[0] = m_Player.GetRandomGeneratorSyncManager().GetRandomInRange(RandomGeneratorS
■■m_HandsCurvePoints.Insert(point_1);//forms a 2 dimensional spline(z is ignored)
■■m_HandsCurvePoints.Insert(point_2);
■■m_HandsCurvePoints.Insert(point_3);
■■m_HandsCurvePoints.Insert(point_4);
■■m_HandsCurvePoints.Insert("0 0 0");
■■m_HandsOffsetRelativeTime = 1;
■■
■■m_MouseOffsetRangeMin = 60;//in degrees min
■■m_MouseOffsetRangeMax = 100;//in degrees max
■■m_MouseOffsetDistance = 1.5;//how far should the mouse travel
■■m_MouseOffsetRelativeTime = 0.5;//[0..1] a time it takes to move the mouse the required distance rela
■
■■m_CamOffsetDistance = 0.04;
■■m_CamOffsetRelativeTime = 1;
■■
■}
}


// ---------------------------------------------
// ------ MagnumRecoil.c - END ----------------------
// ---------------------------------------------


// ---------------------------------------------
// ------ MainMenu.c - START --------------------
// ---------------------------------------------


class MainMenu extends UIScriptedMenu
{
■protected ref MainMenuStats■■m_Stats;
■protected ref MainMenuVideo■■m_Video;
■
■protected MissionMainMenu■■m_Mission;
■protected DayZIntroScenePC ■■m_ScenePC;
■
■protected TextWidget■■■m_PlayerName;
■protected TextWidget■■■m_Version;
■
■protected Widget■■■■m_CharacterRotationFrame;
■
■protected Widget■■■■m_Play;
■protected Widget■■ ■■m_ChooseServer;
■protected Widget■■■■m_CustomizeCharacter;
■protected Widget■■■■m_PlayVideo;
```

```
// ---------------------------------------------
// ------ MainMenuButtonEffect.c - START --------------------
// ---------------------------------------------


class MainMenuButtonEffect : ScriptedWidgetEventHandler
{
■reference float speed;
■reference float amount;
■protected float m_textProportion;
■protected float m_textProportion2;
■protected ButtonWidget m_root;
■protected ref AnimatorTimer m_anim;
■
■// -----------------------------------------------------------
■void MainMenuButtonEffect()
■{
■■if ( GetGame() )
■■{
■■■GetGame().GetUpdateQueue(CALL_CATEGORY_GUI).Insert(this.Update);
■■}
■■m_anim = new AnimatorTimer();
■}
■
■// -----------------------------------------------------------
■void ~MainMenuButtonEffect()
■{
■■if ( GetGame() && GetGame().GetUpdateQueue(CALL_CATEGORY_GUI) )
■■{
■■■GetGame().GetUpdateQueue(CALL_CATEGORY_GUI).Remove(this.Update);
■■}
■}
■
■// -----------------------------------------------------------
■void OnWidgetScriptInit(ButtonWidget w)
■{
■■m_root = w;
■■m_root.SetHandler(this);
■}
■
■// -----------------------------------------------------------
■protected void Update(float tDelta)
■{
■■m_anim.Tick(tDelta);
■■float p = amount * m_anim.GetValue();
■■//m_root.SetTextProportion( m_textProportion + (p * 0.5) );
■■m_root.SetTextOffset( p * 4, 0 );
■
■■float c = 1.0 - m_anim.GetValue();
■■m_root.SetTextColor(ARGBF(1, 1, c, c));
■}
■
■// -----------------------------------------------------------
■override bool OnFocus(Widget w, int x, int y)
■{
■■//if ( !m_anim.IsRunning() ) m_textProportion = m_root.GetTextProportion();
■■if ( !m_anim.IsRunning() )
■■{
■■■m_root.GetPos( m_textProportion, m_textProportion2 );
■■}
■■m_anim.Animate(1.0, speed);

■■return false;
```

```
// ---------------------------------------------
// ------ MainMenuConsoles.c - START --------------------
// ---------------------------------------------


class MainMenuConsole extends UIScriptedMenu
{
■protected ref MainMenuVideo■■m_Video;
■
■protected MissionMainMenu■■m_Mission;
■protected DayZIntroScenePC ■■m_ScenePC;
■
■protected TextWidget■■■m_PlayerName;
■protected TextWidget■■■m_Version;
■
■protected Widget■■■■m_OpenDLC;
■protected Widget■■■■m_ChangeAccount;
■protected Widget■■■■m_CustomizeCharacter;
■protected Widget■■■■m_PlayVideo;
■protected Widget■■■■m_Tutorials;
■protected Widget■■■■m_Options;
■protected Widget■■■■m_Controls;
■protected Widget■■■■m_Play;
■protected Widget■■■■m_MessageButton;
■
■protected ref Widget■■■m_LastFocusedButton;
■
■protected ref ModsMenuDetailed■m_ModsDetailed;

■override Widget Init()
■{
■■layoutRoot = GetGame().GetWorkspace().CreateWidgets("gui/layouts/new_ui/main_menu_console.lay
■■
■■m_PlayerName■■■■= TextWidget.Cast(layoutRoot.FindAnyWidget("character_name_xbox"));
■■
■■m_OpenDLC■■■■■= layoutRoot.FindAnyWidget("show_dlc");
■■m_ChangeAccount■■■■= layoutRoot.FindAnyWidget("choose_account");
■■m_CustomizeCharacter■■= layoutRoot.FindAnyWidget("customize_character");
■■m_PlayVideo■■■■■= layoutRoot.FindAnyWidget("play_video");
■■m_Tutorials■■■■■= layoutRoot.FindAnyWidget("tutorials");
■■m_Options■■■■■= layoutRoot.FindAnyWidget("options");
■■m_Controls■■■■■= layoutRoot.FindAnyWidget("controls");
■■m_Play■■■■■■= layoutRoot.FindAnyWidget("play");
■■m_MessageButton■■■■= layoutRoot.FindAnyWidget("message_button");
■■
■■m_Version■■■■■= TextWidget.Cast(layoutRoot.FindAnyWidget("version"));
■■m_Mission■■■■■= MissionMainMenu.Cast(GetGame().GetMission());
■■m_LastFocusedButton■■■= m_Play;
■■■■
■■GetGame().GetUIManager().ScreenFadeOut(1);

■■string launch_done;
■■if (!GetGame().GetProfileString("FirstLaunchDone", launch_done) || launch_done != "true")
■■{
■■■GetGame().SetProfileString("FirstLaunchDone", "true");
■■■GetGame().GetUIManager().ShowDialog("#main_menu_tutorial", "#main_menu_tutorial_desc", 555,
■■■GetGame().SaveProfile();
■■}
■■
■■UpdateControlsElements();
■■UpdateControlsElementVisibility();
■■LoadMods();■■
■■Refresh();
```

```
// ---------------------------------------------
// ------ MainMenuNewsfeed.c - START --------------------
// ---------------------------------------------


class MainMenuNewsfeed extends ScriptedWidgetEventHandler
{
■protected Widget■■m_Root;
■
■protected Widget■■m_DLDLC;
■protected Widget■■m_Discord;
■protected Widget■■m_Feedback;
■protected Widget■■m_DayZForum;
■protected Widget■■m_Twitter;
■protected Widget■■m_Youtube;
■
■protected TextWidget■m_MainText1;
■protected TextWidget■m_MainText2;
■protected TextWidget■m_MainText3;
■protected TextWidget■m_SecText1;
■protected TextWidget■m_SecText2;
■protected TextWidget■m_SecText3;
■
■void MainMenuNewsfeed( Widget root )
■{
■■m_Root■■■= root;
■■
■■m_DLDLC■■■= m_Root.FindAnyWidget( "downloaddlc" );
■■m_Discord■■= m_Root.FindAnyWidget( "discord" );
■■m_Feedback■■= m_Root.FindAnyWidget( "feedback_tracker" );
■■m_DayZForum■■= m_Root.FindAnyWidget( "dayz_forums" );
■■m_Twitter■■= m_Root.FindAnyWidget( "twitter" );
■■m_Youtube■■= m_Root.FindAnyWidget( "youtube" );
■■
■■m_MainText1■■= TextWidget.Cast( m_Root.FindAnyWidget( "SGInfoT1" ) );
■■m_MainText2■■= TextWidget.Cast( m_Root.FindAnyWidget( "SGInfoT2" ) );
■■m_MainText3■■= TextWidget.Cast( m_Root.FindAnyWidget( "SGInfoT3" ) );
■■m_SecText1■■= TextWidget.Cast( m_Root.FindAnyWidget( "SGInfoC1" ) );
■■m_SecText2■■= TextWidget.Cast( m_Root.FindAnyWidget( "SGInfoC2" ) );
■■m_SecText3■■= TextWidget.Cast( m_Root.FindAnyWidget( "SGInfoC3" ) );
■■
■■ShowNewsfeed();
■■
■■m_Root.SetHandler( this );
■}
■
■void ShowNewsfeed()
■{
■■//m_Root.Show( true );
■■m_MainText1.SetText( "#layout_mainmenu_newsfeed_sgz_title_1" );
■■m_MainText1.Update();
■■m_MainText2.SetText( "#layout_mainmenu_newsfeed_sgz_title_2" );
■■m_MainText2.Update();
■■m_MainText3.SetText( "#layout_mainmenu_newsfeed_sgz_title_3" );
■■m_MainText3.Update();
■■m_SecText1.SetText( "#layout_mainmenu_newsfeed_sgz_text_1" );
■■m_SecText1.Update();
■■m_SecText2.SetText( "#layout_mainmenu_newsfeed_sgz_text_2" );
■■m_SecText2.Update();
■■m_SecText3.SetText( "#layout_mainmenu_newsfeed_sgz_text_3" );
■■m_SecText3.Update();
■}
■
```

```
// ---------------------------------------------
// ------ MainMenuStats.c - START --------------------
// ---------------------------------------------


class MainMenuStats extends ScriptedWidgetEventHandler
{
■protected Widget m_Root;
■
■protected Widget m_TimeSurvived;
■protected TextWidget m_TimeSurvivedValue;
■
■protected Widget m_PlayersKilled;
■protected TextWidget m_PlayersKilledValue;
■
■protected Widget m_InfectedKilled;
■protected TextWidget m_InfectedKilledValue;
■
■protected Widget m_DistanceTraveled;
■protected TextWidget m_DistanceTraveledValue;
■
■protected Widget m_LongRangeShot;
■protected TextWidget m_LongRangeShotValue;
■
■void MainMenuStats( Widget root )
■{
■■m_Root = root;
■■
■■m_TimeSurvived = m_Root.FindAnyWidget( "TimeSurvived" );
■■m_TimeSurvivedValue = TextWidget.Cast( m_Root.FindAnyWidget( "TimeSurvivedValue" ) );
■■
■■m_PlayersKilled = m_Root.FindAnyWidget( "PlayersKilled" );
■■m_PlayersKilledValue = TextWidget.Cast( m_Root.FindAnyWidget( "PlayersKilledValue" ) );
■■
■■m_InfectedKilled = m_Root.FindAnyWidget( "InfectedKilled" );
■■m_InfectedKilledValue = TextWidget.Cast( m_Root.FindAnyWidget( "InfectedKilledValue" ) );
■■
■■m_DistanceTraveled = m_Root.FindAnyWidget( "DistanceTraveled" );
■■m_DistanceTraveledValue = TextWidget.Cast( m_Root.FindAnyWidget( "DistanceTraveledValue" ) );
■■
■■m_LongRangeShot = m_Root.FindAnyWidget( "LongRangeShot" );
■■m_LongRangeShotValue = TextWidget.Cast( m_Root.FindAnyWidget( "LongRangeShotValue" ) );
■}
■■
■void ShowStats()
■{
■■m_Root.Show( true );
■■UpdateStats();
■}
■
■void HideStats()
■{
■■m_Root.Show( false );
■}
■
■void UpdateStats()
■{
■■PlayerBase player;
■■MissionMainMenu mission_main_menu = MissionMainMenu.Cast( GetGame().GetMission() );
■■
■■#ifdef PLATFORM_WINDOWS
■■■player = mission_main_menu.GetIntroScenePC().GetIntroCharacter().GetCharacterObj();
■■#endif
```

```
// ---------------------------------------------
// ------ MainMenuVideo.c - START --------------------
// ---------------------------------------------


class MainMenuVideo extends UIScriptedMenu
{
■protected string ■■■■m_BackButtonTextID;
■
■protected VideoWidget■■■m_Video;
■protected ref Timer■■■■m_VideoPlayTimer;
■protected ref WidgetFadeTimer■m_VideoFadeTimer;
■bool■■■■■■■m_IsPaused;
■
■override Widget Init()
■{
■■layoutRoot ■■■■= GetGame().GetWorkspace().CreateWidgets("gui/layouts/xbox/video_menu.layout"
■■m_Video■■■■■= VideoWidget.Cast(layoutRoot.FindAnyWidget("video"));
■■
■■m_VideoPlayTimer■■= new Timer();
■■m_VideoFadeTimer■■= new WidgetFadeTimer();
■■
■■#ifdef PLATFORM_PS4
■■■m_Video.LoadVideo("/app0/video/DayZ_onboarding_MASTER.mp4", 0);
■■#else
■■■m_Video.LoadVideo("G:\\video\\DayZ_onboarding_MASTER.mp4", 0);
■■#endif
■■m_Video.Play(VideoCommand.REWIND);
■■m_Video.Play(VideoCommand.PLAY);
■■m_VideoFadeTimer.FadeIn(m_Video, 1.5);
■■m_VideoPlayTimer.Run(0.005, this, "PlayVideoLoop", null, true);
■■
■■GetGame().GetMission().GetOnInputDeviceChanged().Insert(OnInputDeviceChanged);
■■
■■return layoutRoot;
■}
■
■void ~MainMenuVideo()
■{
■}
■
■//after show
■override void OnShow()
■{
■■super.OnShow();
■■GetGame().GetUIManager().ShowUICursor(false);
■■GetGame().GetSoundScene().SetSoundVolume(0.0,0.0);
■■
■■UpdateControlsElements();
■}

■//after hide
■override void OnHide()
■{
■■super.OnHide();
■■GetGame().GetUIManager().ShowUICursor(true);
■■GetGame().GetSoundScene().SetSoundVolume(1.0,1.0);
■}
■
■protected void OnInputDeviceChanged(EInputDeviceType pInputDeviceType)
■{
■■UpdateControlsElements();
■}
```

```
// ---------------------------------------------
// ------ Makarov.c - START --------------------
// ---------------------------------------------


class MakarovIJ70_Base : Pistol_Base
{
■override RecoilBase SpawnRecoilObject()
■{
■■return new MakarovRecoil(this);
■}
■
■//Debug menu Spawn Ground Special
■override void OnDebugSpawn()
■{
■■GameInventory inventory = GetInventory();
■■inventory.CreateInInventory( "PistolSuppressor" );
■■
■■SpawnAttachedMagazine("Mag_IJ70_8Rnd");
■}
};


// ---------------------------------------------
// ------ Makarov.c - END ----------------------
// ---------------------------------------------


// ---------------------------------------------
// ------ MakarovPBSuppressor.c - START --------------------
// ---------------------------------------------


class MakarovPBSuppressor extends ItemSuppressor
{
■override bool CanPutAsAttachment( EntityAI parent )
■{
■■if(!super.CanPutAsAttachment(parent)) {return false;}
■■if ( parent.FindAttachmentBySlotName("suppressorImpro") == NULL && parent.FindAttachmentBySlot
■■{
■■■return true;
■■}
■■return false;
■}
}


// ---------------------------------------------
// ------ MakarovPBSuppressor.c - END ----------------------
// ---------------------------------------------


// ---------------------------------------------
// ------ MakarovRecoil.c - START --------------------
// ---------------------------------------------


class MakarovRecoil: RecoilBase
{
■override void Init()
■{
■■vector point_1;
■■vector point_2;
```

```
// ---------------------------------------------
// ------ Man.c - START --------------------
// ---------------------------------------------


//-----------------------------------------------------------------------------
class LightAI extends ManSuper
{
};



// ---------------------------------------------
// ------ Man.c - END ----------------------
// ---------------------------------------------


// ---------------------------------------------
// ------ ManBase.c - START --------------------
// ---------------------------------------------


class ManBase extends DayZPlayerImplement // PSOVIS originaly extends Man
{■
■void ManBase()
■{
■■//Print("PSOVIS: this is ManBase");
■}

■// ------------------------------------------------------------------------
■// ! On Client, 'true' if this instance of a character is controlled by the player(ie. not a remote player)
■bool IsControlledPlayer()
■{
■■return( GetGame().GetPlayer() == this );
■}

■// ------------------------------------------------------------------------
■override void OnVariablesSynchronized()
■{
■■super.OnVariablesSynchronized();
■}
}


// ---------------------------------------------
// ------ ManBase.c - END ----------------------
// ---------------------------------------------


// ---------------------------------------------
// ------ ManSuit_ColorBase.c - START --------------------
// ---------------------------------------------


class ManSuit_ColorBase extends Clothing
{
■override void SetActions()
■{
■■super.SetActions();
■■AddAction(ActionWringClothes);
■}
};
class ManSuit_Beige extends ManSuit_ColorBase {};
```

```
// ---------------------------------------------
// ------ ManTrigger.c - START --------------------
// ---------------------------------------------


//! Trigger only accepting Object which IsMan()
class ManTrigger : Trigger
{
■override void EOnInit(IEntity other, int extra)
■{
■■SetExtents("-2 -2 -2", "2 2 2");
■}
■
■override protected bool CanAddObjectAsInsider(Object object)
■{
■■return object.IsMan();
■}
■
■
};


// ---------------------------------------------
// ------ ManTrigger.c - END ----------------------
// ---------------------------------------------


// ---------------------------------------------
// ------ MapHandler.c - START --------------------
// ---------------------------------------------


class MapHandler : ScriptedWidgetEventHandler
{
■protected Widget m_Root;
■
■void MapHandler(Widget w)
■{
■■m_Root = w;
■■m_Root.SetHandler(this);
■}
■
■override bool OnKeyDown(Widget w, int x, int y, int key)
■{
■■if (!super.OnKeyDown(w, x, y, key))
■■■return false;
■■
■■vector screen_to_map = MapWidget.Cast(w).ScreenToMap(Vector(x,y,0));
■■Print(key);
■■Print(screen_to_map);
■■return true;
■}
■
■override bool OnDoubleClick(Widget w, int x, int y, int button)
■{
■■super.OnDoubleClick(w, x, y, button);
■■
■■vector screen_to_map = MapWidget.Cast(w).ScreenToMap(Vector(x,y,0));
■■MapMenu m = MapMenu.Cast(g_Game.GetUIManager().FindMenu(MENU_MAP));
■■int rand = Math.RandomInt(0,eMapMarkerTypes.MARKERTYPE_MAX);
■■//m.AddMarker(screen_to_map,ARGB(255,255,0,0),rand);
■■
■■//m.AddUserMark(screen_to_map, "marker", ARGB(255,0,0,255), "\\dz\\gear\\navigation\\data\\map_tre
// ---------------------------------------------
// ------ MapMarkersInfo.c - START --------------------
// ---------------------------------------------


class MapMarkerTypes
{
■protected static ref map<int,string> m_MarkerTypes;
■
■static void Init()
■{
■■m_MarkerTypes = new ref map<int,string>;
■■
■■RegisterMarkerType(eMapMarkerTypes.MARKERTYPE_MAP_BORDER_CROSS,"\\DZ\\gear\\navigat
■■RegisterMarkerType(eMapMarkerTypes.MARKERTYPE_MAP_BROADLEAF,"\\DZ\\gear\\navigation\\d
■■RegisterMarkerType(eMapMarkerTypes.MARKERTYPE_MAP_CAMP,"\\DZ\\gear\\navigation\\data\\ma
■■RegisterMarkerType(eMapMarkerTypes.MARKERTYPE_MAP_FACTORY,"\\DZ\\gear\\navigation\\data
■■RegisterMarkerType(eMapMarkerTypes.MARKERTYPE_MAP_FIR,"\\DZ\\gear\\navigation\\data\\map_
■■RegisterMarkerType(eMapMarkerTypes.MARKERTYPE_MAP_FIREDEP,"\\DZ\\gear\\navigation\\data\
■■RegisterMarkerType(eMapMarkerTypes.MARKERTYPE_MAP_GOVOFFICE,"\\DZ\\gear\\navigation\\d
■■RegisterMarkerType(eMapMarkerTypes.MARKERTYPE_MAP_HILL,"\\DZ\\gear\\navigation\\data\\map
■■RegisterMarkerType(eMapMarkerTypes.MARKERTYPE_MAP_MONUMENT,"\\DZ\\gear\\navigation\\d
■■RegisterMarkerType(eMapMarkerTypes.MARKERTYPE_MAP_PALM,"\\DZ\\gear\\navigation\\data\\ma
■■RegisterMarkerType(eMapMarkerTypes.MARKERTYPE_MAP_POLICE,"\\DZ\\gear\\navigation\\data\\m
■■RegisterMarkerType(eMapMarkerTypes.MARKERTYPE_MAP_STATION,"\\DZ\\gear\\navigation\\data\
■■RegisterMarkerType(eMapMarkerTypes.MARKERTYPE_MAP_STORE,"\\DZ\\gear\\navigation\\data\\m
■■RegisterMarkerType(eMapMarkerTypes.MARKERTYPE_MAP_TOURISM,"\\DZ\\gear\\navigation\\data
■■RegisterMarkerType(eMapMarkerTypes.MARKERTYPE_MAP_TRANSMITTER,"\\DZ\\gear\\navigation
■■RegisterMarkerType(eMapMarkerTypes.MARKERTYPE_MAP_TSHELTER,"\\DZ\\gear\\navigation\\dat
■■RegisterMarkerType(eMapMarkerTypes.MARKERTYPE_MAP_TSIGN,"\\DZ\\gear\\navigation\\data\\m
■■RegisterMarkerType(eMapMarkerTypes.MARKERTYPE_MAP_VIEWPOINT,"\\DZ\\gear\\navigation\\da
■■RegisterMarkerType(eMapMarkerTypes.MARKERTYPE_MAP_VINEYARD,"\\DZ\\gear\\navigation\\dat
■■RegisterMarkerType(eMapMarkerTypes.MARKERTYPE_MAP_WATERPUMP,"\\DZ\\gear\\navigation\\
■}
■
■static void RegisterMarkerType(int id, string path)
■{
■■m_MarkerTypes.Set(id,path);
■}
■
■static string GetMarkerTypeFromID(int id)
■{
■■return m_MarkerTypes.Get(id);
■}
}

enum eMapMarkerTypes
{
■MARKERTYPE_MAP_BORDER_CROSS = 0,
■MARKERTYPE_MAP_BROADLEAF,
■MARKERTYPE_MAP_CAMP,
■MARKERTYPE_MAP_FACTORY,
■MARKERTYPE_MAP_FIR,
■MARKERTYPE_MAP_FIREDEP,
■MARKERTYPE_MAP_GOVOFFICE,
■MARKERTYPE_MAP_HILL,
■MARKERTYPE_MAP_MONUMENT,
■MARKERTYPE_MAP_PALM,
■MARKERTYPE_MAP_POLICE,
■MARKERTYPE_MAP_STATION,
■MARKERTYPE_MAP_STORE,
■MARKERTYPE_MAP_TOURISM,
■MARKERTYPE_MAP_TRANSMITTER
```

```
// ---------------------------------------------
// ------ MapMenu.c - START --------------------
// ---------------------------------------------


class MapMenu extends UIScriptedMenu
{
■protected const string COORD_FORMAT ■■■= "%1.%2%3";
■protected const int SCALE_RULER_LINE_WIDTH ■■= 8;
■protected const int SCALE_RULER_NUM_SEGMENTS ■= 10;

■protected bool ■■■■■■■m_WasChanged;
■protected bool ■■■■■■■m_HasCompass
■protected bool ■■■■■■■m_HasGPS
■protected bool ■■■■■■■m_IsOpenning;
■
■protected float■■■■■■■m_ToolScaleCellSizeCanvasWidth;

■protected ref IngameHud■■■■■m_Hud ;
■protected ref MapHandler ■■■■m_MapMenuHandler;
■protected ref MapWidget ■■■■m_MapWidgetInstance;
■protected ref SizeToChild■■■■m_LegendResizer;

■protected ImageWidget ■■■■■m_Images;
■protected Widget■■■■■■m_GPSMarker;
■protected ImageWidget■■■■■m_GPSMarkerArrow;
■protected Widget■■■■■■m_UpperLegendContainer;
■protected Widget■■■■■■m_ToolsCompassBase;
■protected ImageWidget■■■■■m_ToolsCompassArrow;
■protected TextWidget■■■■■m_ToolsCompassAzimuth;
■protected TextWidget■■■■■m_ToolsScaleContourText;
■protected TextWidget■■■■■m_ToolsGPSElevationText;
■protected TextWidget■■■■■m_ToolsGPSXText;
■protected TextWidget■■■■■m_ToolsGPSYText;
■protected TextWidget■■■■■m_ToolsScaleCellSizeText;
■protected CanvasWidget■■■■■m_ToolsScaleCellSizeCanvas;
■protected ItemMap ■■■■■■m_Map;
■//int ■■■■■■■■■m_MarkerCount;
■
■protected ref MapNavigationBehaviour■m_MapNavigationBehaviour;
■
■override Widget Init()
■{
■■layoutRoot ■■■= GetGame().GetWorkspace().CreateWidgets("gui/layouts/day_z_map.layout");
■■m_Hud ■■■■= IngameHud.Cast(GetGame().GetMission().GetHud());
■■m_IsOpenning■■= true;
■■
■■/*MapWidget m = MapWidget.Cast(layoutRoot.FindAnyWidget("Map"));
■■if (m)
■■{
■■■m.AddUserMark("2681 4.7 1751", "Lalal", ARGB(255,255,0,0), "\\dz\\gear\\navigation\\data\\map_tree
■■■m.AddUserMark("2683 4.7 1851", "Lala2", ARGB(255,0,255,0), "\\dz\\gear\\navigation\\data\\map_bu
■■■m.AddUserMark("2670 4.7 1651", "Lala3", ARGB(255,0,0,255), "\\dz\\gear\\navigation\\data\\map_bu
■■}*/
■■
■■Widget mapToolsContainer = layoutRoot.FindAnyWidget("Map_Tools_Container");
■■mapToolsContainer.GetScript(m_LegendResizer);

■■m_MapWidgetInstance■■■= MapWidget.Cast(layoutRoot.FindAnyWidget("Map"));
■■m_GPSMarker■■■■■= layoutRoot.FindAnyWidget("GPSMarkerCircle");
■■m_GPSMarkerArrow■■■= ImageWidget.Cast(layoutRoot.FindAnyWidget("GPSMarkerArrow"));
■■m_UpperLegendContainer■■= layoutRoot.FindAnyWidget("Tools_Extra");
■■layoutRoot.Update();
```

```
// ---------------------------------------------
// ------ MapNavigationBehaviour.c - START --------------------
// ---------------------------------------------


enum EMapNavigationType
{
■BASIC ■= 0,
■COMPASS■= 1,
■GPS ■= 2,
■ALL■■= 4
}

class MapNavigationBehaviour
{
■const int RANDOM_DEVIATION_MIN ■■■■■■= 4;
■const int RANDOM_DEVIATION_MAX ■■■■■■= 15;
■
■static const int DISPLAY_GRID_POS_MAX_CHARS_COUNT ■= 3;
■static const int DISPLAY_ALT_MAX_CHARS_COUNT ■■= 4;
■
■protected static const string GRID_SIZE_CFG_PATH■= "CfgWorlds %1 Grid Zoom1 stepX";

■protected int■■■■■m_RandomPositionDeviationX;
■protected int■■■■■m_RandomPositionDeviationZ;
■protected EMapNavigationType■m_NavigationType;
■protected PlayerBase■■■m_Player;
■
■protected ref array<EntityAI>■m_GPSInPossessionArr;
■protected ref array<EntityAI>■m_CompassInPossessionArr;
■
■void MapNavigationBehaviour(PlayerBase pPlayer, EMapNavigationType pNavigationType = EMapNavi
■{
■■m_Player ■■■■■= pPlayer;
■■m_NavigationType■■■= pNavigationType;
■■
■■m_GPSInPossessionArr ■■= new array<EntityAI>();
■■m_CompassInPossessionArr■= new array<EntityAI>();
■}
■
■protected void SetNavigationType(EMapNavigationType pType)
■{
■■m_NavigationType = m_NavigationType | pType;
■}
■
■protected void UnsetNavigationType(EMapNavigationType pType)
■{
■■m_NavigationType = m_NavigationType & ~pType;
■}

■EMapNavigationType GetNavigationType()
■{
■■return m_NavigationType;
■}
■
■void OnItemInPlayerPossession(EntityAI item)
■{
■■if (item.IsInherited(ItemGPS))
■■{
■■■if (m_GPSInPossessionArr.Find(item) == INDEX_NOT_FOUND)
■■■{
■■■■m_GPSInPossessionArr.Insert(item);
■■■■SetNavigationType(EMapNavigationType.GPS);
```

```
// ---------------------------------------------
// ------ Marmalade.c - START --------------------
// ---------------------------------------------


class Marmalade: Edible_Base
{
■override void SetActions()
■{
■■super.SetActions();
■■
■■AddAction(ActionForceFeed);
■■AddAction(ActionEatBig);
■}
};



// ---------------------------------------------
// ------ Marmalade.c - END ----------------------
// ---------------------------------------------


// ---------------------------------------------
// ------ Mask.c - START --------------------
// ---------------------------------------------


class MaskMdfr: ModifierBase
{
■const float IN_AREA_CONSUME_FILTER_QUANTITY_PER_SEC = 0.3;
■const float OUT_AREA_CONSUME_FILTER_QUANTITY_PER_SEC = 0.03;
■
■const float STAMINA_RECOVERY_MODIFIER = 0.5;
■const float STAMINA_DEPLETION_MODIFIER = 1.25;
■
■const float LOW_FILTER_SOUND_EVENT_MIN = 3;
■const float LOW_FILTER_SOUND_EVENT_MAX = 9;
■
■const float LOW_FILTER_SOUND_THRESHOLD = 0.2;//[0..1] , what's the filter quantity between 0 and
■
■float m_SoundTimeAccu1;
■float m_NextSoundEventTime;
■
■override void Init()
■{
■■m_TrackActivatedTime = false;
■■m_ID ■■■■■= eModifiers.MDF_MASK;
■■m_TickIntervalInactive ■= DEFAULT_TICK_TIME_INACTIVE;
■■m_TickIntervalActive ■= DEFAULT_TICK_TIME_ACTIVE_SHORT;
■}■

■override bool ActivateCondition(PlayerBase player)
■{
■■return MaskBase.Cast(player.GetInventory().FindAttachment(InventorySlots.MASK)) != null;
■}

■override bool DeactivateCondition(PlayerBase player)
■{
■■return !ActivateCondition( player);
■}

■override void OnTick(PlayerBase player, float deltaT)
```

```
// ---------------------------------------------
// ------ MaskBase.c - START --------------------
// ---------------------------------------------


class MaskBase extends Clothing
{
■float m_LowFilterEventTime;
■
■override bool IsGasMask()
■{
■■return true;
■}
■
■override array<int> GetEffectWidgetTypes()
■{
■■return {EffectWidgetsTypes.MASK_OCCLUDER, EffectWidgetsTypes.MASK_BREATH};
■}
■
■override bool AllowFoodConsumption()
■{
■■return false;
■}
■
■override void OnDebugSpawn()
■{
■■GetInventory().CreateInInventory("GasMask_Filter");
■}
■
■override void EEItemAttached(EntityAI item, string slot_name)
■{
■■super.EEItemAttached(item, slot_name);
■}
■
■
■EntityAI GetExternalFilter()
■{
■■return FindAttachmentBySlotName("GasMaskFilter");
■}
■
■//! has either external or integrated non-empty non-ruined filter ?
■bool HasValidFilter()
■{
■■ItemBase filter = ItemBase.Cast(GetExternalFilter());
■■if (filter && !filter.IsRuined() && filter.GetQuantity() > 0)
■■■return true;
■■else if (GetQuantity() > 0 && !IsRuined())
■■■return true;

■■return false;
■}
■
■float GetFilterQuantityMax()
■{
■■ItemBase filter = ItemBase.Cast(GetExternalFilter());
■■if (filter) //mask can have a filter, and filter is attached
■■■return filter.GetQuantityMax();
■■else if (HasQuantity())//filter was not attached, checking if mask has internal filter
■■■return GetQuantityMax();
■■
■■//mask does not have a filter attached nor does it have internal filter, max quantity can't be obtained
■■return 0;
■}
```

```
// ---------------------------------------------
// ------ Matchbox.c - START --------------------
// ---------------------------------------------


class Matchbox extends ItemBase
{
■override void InitItemVariables()
■{
■■super.InitItemVariables();
■■can_this_be_combined = true;
■}
■
■override bool CanIgniteItem( EntityAI ignite_target = NULL )
■{
■■if ( GetQuantity() > 0 && GetWet() < GameConstants.STATE_DAMP )
■■■return true;
■■else
■■■return false;
■}
■
■override void OnIgnitedTarget( EntityAI ignited_item )
■{
■■if ( GetGame().IsServer() )
■■{
■■■AddQuantity( -1 );
■■}
■}
■
■override void OnIgnitedTargetFailed( EntityAI target_item )
■{
■■if ( GetGame().IsServer() )
■■{
■■■AddQuantity( -1 );
■■}
■}
■
■override void SetActions()
■{
■■super.SetActions();
■■
■■AddAction(ActionLightItemOnFire);
■}
■
■/*
■override bool IsTargetIgnitionSuccessful( EntityAI item_target = NULL )
■{
■}
■*/
}


// ---------------------------------------------
// ------ Matchbox.c - END ----------------------
// ---------------------------------------------


// ---------------------------------------------
// ------ MeatTenderizer.c - START --------------------
// ---------------------------------------------


class MeatTenderizer: Inventory_Base
```

```
// ---------------------------------------------
// ------ MedicalScrubsHat_ColorBase.c - START --------------------
// ---------------------------------------------


class MedicalScrubsHat_ColorBase extends Clothing
{
■override void SetActions()
■{
■■super.SetActions();
■■AddAction(ActionWringClothes);
■}
};
class MedicalScrubsHat_Blue extends MedicalScrubsHat_ColorBase {};
class MedicalScrubsHat_White extends MedicalScrubsHat_ColorBase {};
class MedicalScrubsHat_Green extends MedicalScrubsHat_ColorBase {};


// ---------------------------------------------
// ------ MedicalScrubsHat_ColorBase.c - END ----------------------
// ---------------------------------------------


// ---------------------------------------------
// ------ MedicalScrubsPants_ColorBase.c - START --------------------
// ---------------------------------------------


class MedicalScrubsPants_ColorBase extends Clothing
{
■override void SetActions()
■{
■■super.SetActions();
■■AddAction(ActionWringClothes);
■}
};
class MedicalScrubsPants_Blue extends MedicalScrubsPants_ColorBase {};
class MedicalScrubsPants_Green extends MedicalScrubsPants_ColorBase {};
class MedicalScrubsPants_White extends MedicalScrubsPants_ColorBase {};


// ---------------------------------------------
// ------ MedicalScrubsPants_ColorBase.c - END ----------------------
// ---------------------------------------------


// ---------------------------------------------
// ------ MedicalScrubsShirt_ColorBase.c - START --------------------
// ---------------------------------------------


class MedicalScrubsShirt_ColorBase extends Clothing
{
■override void SetActions()
■{
■■super.SetActions();
■■AddAction(ActionWringClothes);
■}
};
class MedicalScrubsShirt_Blue extends MedicalScrubsShirt_ColorBase {};
class MedicalScrubsShirt_Green extends MedicalScrubsShirt_ColorBase {};
class MedicalScrubsShirt_White extends MedicalScrubsShirt_ColorBase {};
```

```
// ---------------------------------------------
// ------ MediumGasCannister.c - START --------------------
// ---------------------------------------------


class MediumGasCannister extends ItemBase
{
■override bool CanExplodeInFire()
■{
■■return true;
■}
}


// ---------------------------------------------
// ------ MediumGasCannister.c - END ----------------------
// ---------------------------------------------


// ---------------------------------------------
// ------ MediumTent.c - START --------------------
// ---------------------------------------------


class MediumTent extends TentBase
{
■void MediumTent()
■{■■
■■m_ToggleAnimations.Insert( new ToggleAnimations("EntranceO", "EntranceC", OPENING_0), 0 );
■■■
■■m_ShowAnimationsWhenPitched.Insert( "Body" );
■■//m_ShowAnimationsWhenPitched.Insert( "EntranceO" );
■■m_ShowAnimationsWhenPitched.Insert( "Pack" );

■■m_ShowAnimationsWhenPacked.Insert( "Inventory" );
■■
■■m_HalfExtents = Vector(0.8, 0.15, 1.3);
■}
■■
■override void EEInit()
■{■■
■■super.EEInit();
■}
■
■override void OnItemLocationChanged(EntityAI old_owner, EntityAI new_owner)
■{■■
■■super.OnItemLocationChanged(old_owner, new_owner);
■}
■
■override string GetSoundOpen()
■{
■■return "MediumTent_Door_Open_SoundSet";
■}
■
■override string GetSoundClose()
■{
■■return "MediumTent_Door_Close_SoundSet";
■}
■
■override bool HasClutterCutter()
■{
■■return true;
■}
```

```
// ---------------------------------------------
// ------ Megaphone.c - START --------------------
// ---------------------------------------------


class Megaphone extends ItemMegaphone
{
■override bool IsTransmitter()
■{
■■return true;
■}■

■//--- POWER EVENTS■
■override void OnSwitchOn()
■{
■■if ( !GetCompEM().CanWork() )
■■{
■■■GetCompEM().SwitchOff();
■■}
■}■
■
■override void OnWorkStart()
■{
■■//turn device on
■■SwitchOn ( true ); // Note: This is not Energy Manager function. This is engine function.
■}
■
■override void OnWorkStop()
■{
■■//auto switch off (EM)
■■GetCompEM().SwitchOff();
■■
■■//turn off device
■■SwitchOn ( false ); // Note: This is not Energy Manager function. This is engine function.
■}
■
■//TODO add turn on/off actions
■//proto native bool CanSpeak();
■//proto native void SetCanSpeak(bool onOff);
■
■override void SetActions()
■{
■■super.SetActions();
■■
■■AddAction(ActionTurnOnTransmitter);
■■AddAction(ActionTurnOffTransmitter);
■■AddAction(ActionRaiseMegaphone);
■}■
■
}



// ---------------------------------------------
// ------ Megaphone.c - END ---------------------
// ---------------------------------------------
// ---------------------------------------------
// ------ MeleeAttack.c - START --------------------
// ---------------------------------------------


class MeleeAttackSoundEvents extends PlayerSoundEventBase
{
■void MeleeAttackSoundEvents()
■{
■■m_Type = EPlayerSoundEventType.MELEE;
■■m_HasPriorityOverTypes = -1;
■}
■
■override bool CanPlay(PlayerBase player)
■{
■■return true;
■}
■
■override bool HasPriorityOverCurrent(PlayerBase player, EPlayerSoundEventID other_state_id, EPlayer
■{
■■if( type_other == EPlayerSoundEventType.DAMAGE )
■■{
■■■return false;
■■}
■■return true;
■}
■
■override void OnEnd()
■{
■■super.OnEnd();
■■if(m_Player)
■■■StaminaSoundHandlerClient.Cast(m_Player.m_PlayerSoundManagerClient.GetHandler(eSoundHand
■}
}

class MeleeAttackLightEvent extends MeleeAttackSoundEvents
{
■void MeleeAttackLightEvent()
■{
■■m_ID = EPlayerSoundEventID.MELEE_ATTACK_LIGHT;
■■m_SoundVoiceAnimEventClassID = 16;
■}
}

class MeleeAttackHeavyEvent extends MeleeAttackSoundEvents
{
■void MeleeAttackHeavyEvent()
■{
■■m_ID = EPlayerSoundEventID.MELEE_ATTACK_HEAVY;
■■m_SoundVoiceAnimEventClassID = 17;
■}
■
}


// ---------------------------------------------
// ------ MeleeAttack.c - END ---------------------
// ---------------------------------------------
// ---------------------------------------------
// ------ MeleeTargeting.c - START --------------------
// ---------------------------------------------


class MeleeTargetData
{■
■Object Obj;
■vector HitPos;
■int HitComponent;
■
■void MeleeTargetData(Object o, vector p, int c)
■{
■■Obj = o;
■■HitPos = p;
■■HitComponent = c;
■}
}

class MeleeTargetSettings
{
■vector ConeOrigin;
■float ConeLength;
■float ConeHalfAngle;
■float ConeHalfAngleRad;
■float ConeMinHeight;
■float ConeMaxHeight;
■
■vector ConeLeftPoint;
■vector ConeRightPoint;
■
■vector RayStart;
■vector RayEnd;
■vector Dir;
■vector XZDir;
■float MaxDist;
■
■EntityAI Attacker;
■ref array<typename> TargetableObjects
■
■void MeleeTargetSettings(vector coneOrigin, float coneLength, float coneHalfAngle, float coneMinHeight
■{
■■ConeOrigin ■■■= coneOrigin;
■■ConeLength ■■■= coneLength;
■■ConeHalfAngle ■■= coneHalfAngle;
■■ConeHalfAngleRad ■= Math.DEG2RAD * coneHalfAngle;
■■ConeMinHeight ■■= coneMinHeight;
■■ConeMaxHeight ■■= coneMaxHeight;
■■
■■RayStart ■■■= rayStart;
■■RayEnd ■■■■= rayStart + Math.SqrFloat(coneLength) * dir;
■■
■■Dir ■■■■= dir;
■■
■■XZDir ■■■■= dir;
■■XZDir[1] ■■■= 0;
■■XZDir.Normalize();
■■
■■MaxDist ■■■= maxDist;
■■
■■Attacker ■■■= pToIgnore;
■■TargetableObjects ■= targetableObjects;
■■
```

```
// ---------------------------------------------
// ------ MenuCarEngineSmoke.c - START --------------------
// ---------------------------------------------


class MenuCarEngineSmoke : EffectParticle
{
■void MenuCarEngineSmoke()
■{
■■SetParticleID(ParticleList.SMOKING_CAR_ENGINE);
■}
}


// ---------------------------------------------
// ------ MenuCarEngineSmoke.c - END ----------------------
// ---------------------------------------------


// ---------------------------------------------
// ------ MenuEvaporation.c - START --------------------
// ---------------------------------------------


class MenuEvaporation : EffectParticle
{
■void MenuEvaporation()
■{
■■SetParticleID(ParticleList.EVAPORATION);
■}
}


// ---------------------------------------------
// ------ MenuEvaporation.c - END ----------------------
// ---------------------------------------------


// ---------------------------------------------
// ------ menuManager.c - START --------------------
// ---------------------------------------------


#ifdef GAME_TEMPLATE

enum DialogPriority
{
■INFORMATIVE,
■WARNING,
■CRITICAL■
};

enum DialogResult
{
■PENDING,
■OK,
■YES,
■NO,
■CANCEL,
};

enum ScriptMenuPresetEnum
{
```

```
// ----------------------------------------------
// ------ MessageReceiverBase.c - START --------------------
// ----------------------------------------------


class MessageReceiverBase
{
■// message system
■PluginMessageManager m_ModuleMessageManager;
■string m_System;

■void OnReceive(int channel) {}
■void OnReceiveInt(int channel, int value) {}
■void OnReceiveFloat(int channel, float value) {}
■void OnReceiveString(int channel, string value) {}
■//! This method is called when an object inheriting from this class subscribes to a channel, and a messag
■void OnReceiveParam(int channel, Param params) {}
■
■void MessageReceiverBase()
■{
■■m_ModuleMessageManager = PluginMessageManager.Cast(GetPlugin(PluginMessageManager));
■}

■void Subscribe(int channel)
■{
■■m_ModuleMessageManager.Subscribe(this,channel);
■}

■void Unsubscribe(int channel)
■{
■■if( IsPluginManagerExists() ) m_ModuleMessageManager.Unsubscribe(this,channel);
■}

■void UnsubscribeAll()
■{
■■if( IsPluginManagerExists() ) m_ModuleMessageManager.UnsubscribeAll(this);
■}

■void Broadcast(int channel)
■{
■■m_ModuleMessageManager.Broadcast(channel);
■}

■void BroadcastInt(int channel, int value)
■{
■■m_ModuleMessageManager.BroadcastInt(channel, value);
■}

■void BroadcastFloat(int channel, float value)
■{
■■m_ModuleMessageManager.BroadcastFloat(channel, value);
■}

■void BroadcastString(int channel, string value)
■{
■■m_ModuleMessageManager.BroadcastString(channel, value);
■}

■void BroadcastParam(int channel,Param params)
■{
■■m_ModuleMessageManager.BroadcastParam(channel, params);
■}
```

```
// ---------------------------------------------
// ------ MessTin.c - START --------------------
// ---------------------------------------------


class MessTin: Inventory_Base {};



// ---------------------------------------------
// ------ MessTin.c - END ----------------------
// ---------------------------------------------


// ---------------------------------------------
// ------ MetalPlate.c - START --------------------
// ---------------------------------------------


class MetalPlate extends ItemBase
{
■override void SetActions()
■{
■■super.SetActions();
■■
■■AddAction(ActionAttachToConstruction);
■■AddAction(ActionAttachOnSelection);
■}
}


// ---------------------------------------------
// ------ MetalPlate.c - END ----------------------
// ---------------------------------------------


// ---------------------------------------------
// ------ MetalWire.c - START --------------------
// ---------------------------------------------


class MetalWire extends ItemBase
{
■static string SEL_WIRE_ROLLED■= "rolled";
■static string SEL_WIRE_PREFIX■= "Att_";
■static string SEL_WIRE_SUFIX■= "_plugged";
■static string SEL_PLUG_SUFIX■= "_plug";
■
■void MetalWire()
■{
■■
■}
■
■override bool CanPutAsAttachment( EntityAI parent )
■{
■■if (!super.CanPutAsAttachment(parent))
■■■return false;
■■
■■if ( parent.IsInherited(VehicleBattery) )
■■{
■■■EntityAI battery_owner = EntityAI.Cast( parent.GetHierarchyParent() );
■■■
■■■// Check for Not player as if parent is not player, battery is already attached and should not receive ne
```

```
// ---------------------------------------------
// ------ Mich2001Helmet.c - START --------------------
// ---------------------------------------------


class Mich2001Helmet extends HelmetBase
{
■override bool CanPutAsAttachment( EntityAI parent )
■{
■■if(!super.CanPutAsAttachment(parent)) {return false;}
■■
■■Clothing eyewear = Clothing.Cast(parent.FindAttachmentBySlotName("Eyewear"));
■■
■■if ( eyewear && eyewear.ConfigGetBool("isStrap") )
■■{
■■■return false;
■■}
■■return true;
■}
■
■override void SetActions()
■{
■■super.SetActions();
■■
■■AddAction(ActionTurnOnHelmetFlashlight); //use default light actions instead?
■■AddAction(ActionTurnOffHelmetFlashlight);
■■AddAction(ActionToggleNVG);
■}
■
■//Debug menu Spawn Ground Special
■override void OnDebugSpawn()
■{
■■EntityAI entity;
■■if ( Class.CastTo(entity, this) )
■■{
■■■entity.GetInventory().CreateInInventory( "NVGoggles" );
■■■entity.GetInventory().CreateInInventory( "UniversalLight" );
■■■entity.GetInventory().CreateInInventory( "Battery9V" );
■■■entity.GetInventory().CreateInInventory( "Battery9V" );
■■}
■}
};


// ---------------------------------------------
// ------ Mich2001Helmet.c - END ----------------------
// ---------------------------------------------


// ---------------------------------------------
// ------ MilitaryBelt.c - START --------------------
// ---------------------------------------------


class MilitaryBelt: Clothing
{
■override bool CanPutInCargo( EntityAI parent )
■{
■■if( !super.CanPutInCargo( parent ) )
■■{
■■■return false;
■■}
■■
```

```
// ---------------------------------------------
// ------ MilitaryBeret_ColorBase.c - START --------------------
// ---------------------------------------------


class MilitaryBeret_ColorBase extends Clothing {};
class MilitaryBeret_Red extends MilitaryBeret_ColorBase {};
class MilitaryBeret_UN extends MilitaryBeret_ColorBase {};
class MilitaryBeret_NZ extends MilitaryBeret_ColorBase {};
class MilitaryBeret_ChDKZ extends MilitaryBeret_ColorBase {};
class MilitaryBeret_CDF extends MilitaryBeret_ColorBase {};


// ---------------------------------------------
// ------ MilitaryBeret_ColorBase.c - END ----------------------
// ---------------------------------------------


// ---------------------------------------------
// ------ MilitaryBoots_ColorBase.c - START --------------------
// ---------------------------------------------


class MilitaryBoots_ColorBase extends ClothingBase
{
■override bool CanPutAsAttachment( EntityAI parent )
■{
■■if(!super.CanPutAsAttachment(parent)) {return false;}
■■if ( parent != this )
■■{
■■■return true;
■■}
■■return false;
■}■
};

class MilitaryBoots_Beige extends MilitaryBoots_ColorBase {};
class MilitaryBoots_Black extends MilitaryBoots_ColorBase {};
class MilitaryBoots_Bluerock extends MilitaryBoots_ColorBase {};
class MilitaryBoots_Brown extends MilitaryBoots_ColorBase {};
class MilitaryBoots_Redpunk extends MilitaryBoots_ColorBase {};


// ---------------------------------------------
// ------ MilitaryBoots_ColorBase.c - END ----------------------
// ---------------------------------------------


// ---------------------------------------------
// ------ Mil_Barracks1.c - START --------------------
// ---------------------------------------------


class Land_Mil_Barracks1 extends BuildingWithFireplace
{
}


// ---------------------------------------------
// ------ Mil_Barracks1.c - END ----------------------
// ---------------------------------------------
// ---------------------------------------------
// ------ Mil_Barracks3.c - START --------------------
// ---------------------------------------------


class Land_Mil_Barracks3 extends BuildingWithFireplace
{
}


// ---------------------------------------------
// ------ Mil_Barracks3.c - END ----------------------
// ---------------------------------------------


// ---------------------------------------------
// ------ Mil_Barracks4.c - START --------------------
// ---------------------------------------------


class Land_Mil_Barracks4 extends BuildingWithFireplace
{
}


// ---------------------------------------------
// ------ Mil_Barracks4.c - END ----------------------
// ---------------------------------------------


// ---------------------------------------------
// ------ MimeMask.c - START --------------------
// ---------------------------------------------


class MimeMask_Male : Clothing
{
■override bool CanPutAsAttachment(EntityAI parent)
■{
■■if (!super.CanPutAsAttachment(parent))
■■■return false;
■■
■■Clothing headgear = Clothing.Cast(parent.FindAttachmentBySlotName("Headgear"));
■■if (headgear && (headgear.ConfigGetBool("noMask") && !PumpkinHelmet.Cast(headgear)))
■■{
■■■return false;
■■}
■■
■■Clothing eyewear = Clothing.Cast(parent.FindAttachmentBySlotName("Eyewear"));
■■if (eyewear && SportGlasses_ColorBase.Cast(eyewear)) //eyewear.ConfigGetBool("noMask")
■■{
■■■return false;
■■}
■■
■■return true;
■}
};

class MimeMask_Female : MimeMask_Male
{
};
```

```
// ---------------------------------------------
// ------ MindStates.c - START --------------------
// ---------------------------------------------


class MindStateSoundEventBase extends InfectedSoundEventBase
{
■override bool CanPlay()
■{
■■if( !super.CanPlay() )
■■{
■■■return false;
■■}
■■return true;
■}
}

class CalmIdleSoundEvent extends MindStateSoundEventBase
{
■void CalmIdleSoundEvent()
■{
■■m_Type = EInfectedSoundEventType.GENERAL;
■■m_ID = EInfectedSoundEventID.MINDSTATE_CALM_IDLE;
■■m_SoundSetNameRoot = "CalmIdle";
■}
}

class CalmMoveSoundEvent extends MindStateSoundEventBase
{
■void CalmMoveSoundEvent()
■{
■■m_Type = EInfectedSoundEventType.GENERAL;
■■m_ID = EInfectedSoundEventID.MINDSTATE_CALM_MOVE;
■■m_SoundSetNameRoot = "CalmMove";
■}
}

class DisturbedIdleSoundEvent extends MindStateSoundEventBase
{
■void DisturbedIdleSoundEvent()
■{
■■m_Type = EInfectedSoundEventType.GENERAL;
■■m_ID = EInfectedSoundEventID.MINDSTATE_DISTURBED_IDLE;
■■m_SoundSetNameRoot = "DisturbedIdle";
■}
}

class ChaseMoveSoundEvent extends MindStateSoundEventBase
{
■void ChaseMoveSoundEvent()
■{
■■m_Type = EInfectedSoundEventType.GENERAL;
■■m_ID = EInfectedSoundEventID.MINDSTATE_CHASE_MOVE;
■■m_SoundSetNameRoot = "ChaseMove";
■}
}

class AlertedIdleSoundEvent extends MindStateSoundEventBase
{
■void AlertedIdleSoundEvent()
■{
■■m_Type = EInfectedSoundEventType.GENERAL;
■■m_ID = EInfectedSoundEventID.MINDSTATE_ALERTED_IDLE;
```

```
// ---------------------------------------------
// ------ MiniDress_ColorBase.c - START --------------------
// ---------------------------------------------


class MiniDress_ColorBase extends Clothing
{
■override void SetActions()
■{
■■super.SetActions();
■■AddAction(ActionWringClothes);
■}
};
class MiniDress_Pink extends MiniDress_ColorBase {};
class MiniDress_PinkChecker extends MiniDress_ColorBase {};
class MiniDress_RedChecker extends MiniDress_ColorBase {};
class MiniDress_WhiteChecker extends MiniDress_ColorBase {};
class MiniDress_GreenChecker extends MiniDress_ColorBase {};
class MiniDress_BrownChecker extends MiniDress_ColorBase {};
class MiniDress_BlueChecker extends MiniDress_ColorBase {};
class MiniDress_BlueWithDots extends MiniDress_ColorBase {};


// ---------------------------------------------
// ------ MiniDress_ColorBase.c - END ----------------------
// ---------------------------------------------


// ---------------------------------------------
// ------ MiscEvents.c - START --------------------
// ---------------------------------------------


class PickupHeavySoundEvent extends PlayerSoundEventBase
{
■void PickupHeavySoundEvent()
■{
■■m_HasPriorityOverTypes = -1;//-1 for all
■■m_Type = EPlayerSoundEventType.STAMINA;
■■m_ID = EPlayerSoundEventID.PICKUP_HEAVY;
■■m_SoundVoiceAnimEventClassID = 19;
■}
}


// ---------------------------------------------
// ------ MiscEvents.c - END ----------------------
// ---------------------------------------------


// ---------------------------------------------
// ------ MiscGameplayFunctions.c - START --------------------
// ---------------------------------------------


class TurnItemIntoItemLambda extends ReplaceItemWithNewLambda
{
■bool m_TransferAgents;
■bool m_TransferVariables;
■bool m_TransferHealth;
■bool m_ExcludeQuantity;
■float m_quantity_override;
■
```

```
// ---------------------------------------------
// ------ missionBase.c - START --------------------
// ---------------------------------------------


class MissionBase extends MissionBaseWorld
{
■PluginDeveloper ■■m_ModuleDeveloper;
■PluginKeyBinding■■m_ModuleKeyBinding
■PluginAdditionalInfo■m_ModuleServerInfo;
■
■ref WidgetEventHandler ■m_WidgetEventHandler;
■ref WorldData■■■m_WorldData;
■ref WorldLighting■■m_WorldLighting;
■
■ref array<PlayerBase> m_DummyPlayers = new array<PlayerBase>;

■void MissionBase()
■{
■■SetDispatcher(new DispatcherCaller);
■■
■■PluginManagerInit();

■■m_WidgetEventHandler = new WidgetEventHandler;
■■
■■//Debug.DestroyAllShapes();

■■//TODO clea up after Gamescom
■■m_ModuleServerInfo = PluginAdditionalInfo.Cast( GetPlugin(PluginAdditionalInfo) );
■■//
■■SoundSetMap.Init();

■■if (GetGame().IsServer())
■■{
■■■InitialiseWorldData();
■■}
■■else
■■{
■■■GetDayZGame().GetAnalyticsClient().RegisterEvents();
■■■m_WorldLighting■= new WorldLighting;
■■}
■■
■■GetOnInputDeviceConnected().Insert(UpdateInputDevicesAvailability);
■■GetOnInputDeviceDisconnected().Insert(UpdateInputDevicesAvailability);
■■
■■// There is a possibility different maps/servers may be using different effects
■■SEffectManager.Cleanup();
■}

■void ~MissionBase()
■{
■■PluginManagerDelete();
■■
■■if ( GetGame().IsClient() )
■■{
■■■GetDayZGame().GetAnalyticsClient().UnregisterEvents();■
■■}
■■TriggerEffectManager.DestroyInstance();
■}
■
■void InitialiseWorldData()
■{
■■string worldName = "empty";
```

```
// -------------------------------------------
// ------ MissionBaseWorld.c - START -------------------
// -------------------------------------------


class MissionBaseWorld extends Mission
{

}


// -------------------------------------------
// ------ MissionBaseWorld.c - END ---------------------
// -------------------------------------------


// -------------------------------------------
// ------ missionGameplay.c - START -------------------
// -------------------------------------------


class MissionGameplay extends MissionBase
{
■int■■■■■■■■m_LifeState;
■bool■■■■■■■m_Initialized;
■
■protected UIManager■■■■m_UIManager;
■
■Widget■■■■■■■m_HudRootWidget;
■ImageWidget ■■■■■m_MicrophoneIcon;
■
■ref InventoryMenu■■■■m_InventoryMenu;
■ref Chat ■■■■■■m_Chat;
■ref ActionMenu■■■■■m_ActionMenu;
■ref IngameHud■■■■■m_Hud;
■ref HudDebug■■■■■m_HudDebug;
■ref LogoutMenu■■■■■m_Logout;
■ref DebugMonitor■■■■m_DebugMonitor;
■
■protected ref ScriptInvoker ■■m_OnConnectivityChanged;
■
■protected ref GameplayEffectWidgets■■m_EffectWidgets;
■
■ref Timer■■■■■■m_ChatChannelHideTimer;
■ref WidgetFadeTimer■■■■m_ChatChannelFadeTimer;
■ref WidgetFadeTimer■■■■m_MicFadeTimer;
■
■Widget■■■■■■■m_ChatChannelArea;
■TextWidget■■■■■■m_ChatChannelText;
■NoteMenu ■■■■■■m_Note;
■
■protected ref Timer■■■■m_ToggleHudTimer;
■protected const int ■■■HOLD_LIMIT_TIME■= 300; //ms
■protected int■■■■■m_ActionDownTime;
■protected int■■■■■m_ActionUpTime;
■protected bool ■■■■■m_InitOnce;
■protected bool ■■■■■m_ControlDisabled; //DEPRECATED; disabled mode stored below
■protected int ■■■■■m_ControlDisabledMode;
■protected ref array<string> ■m_ActiveInputExcludeGroups; //exclude groups defined in 'specific.xml' file
■protected ref array<int> ■■m_ActiveInputRestrictions; //additional scripted restrictions
■protected bool■■■■■m_ProcessInputExcludes;
■protected bool ■■■■■m_QuickbarHold;
■protected bool ■■■■■m_PlayerRespawning;
```

```
// ---------------------------------------------
// ------ MissionLoader.c - START --------------------
// ---------------------------------------------


class MissionLoader extends UIScriptedWindow
{
■string■■■■m_PathToMissions;
■TextListboxWidget■m_WgtLstMsnList;
■ButtonWidget■■m_WgtBtnMsnPlay;
■ButtonWidget■■m_WgtBtnMsnClose;
■
■ref TStringArray■■m_ListMissionsNames;
■
■void MissionLoader( int id )
■{
■■m_Id = id;
■}
■■
■override Widget Init()
■{
■■m_ListMissionsNames = GetMissionList();
■■
■■m_WgtRoot■■■= GetGame().GetWorkspace().CreateWidgets("gui/layouts/day_z_mission_loader.layo
■■
■■m_WgtLstMsnList = TextListboxWidget.Cast( m_WgtRoot.FindAnyWidget("wgt_lst_missions") );
■■m_WgtBtnMsnPlay = ButtonWidget.Cast( m_WgtRoot.FindAnyWidget("wgt_btn_mission_play") );
■■m_WgtBtnMsnClose = ButtonWidget.Cast( m_WgtRoot.FindAnyWidget("wgt_btn_mission_close") );
■■
■■for ( int i = 0; i < m_ListMissionsNames.Count(); ++i )
■■{
■■■string mission_name = m_ListMissionsNames.Get(i);
■■■m_WgtLstMsnList.AddItem(mission_name, NULL, 0);
■■}

■■return m_WgtRoot;
■}

■private TStringArray GetMissionList()
■{
■■string■file_name;
■■int ■file_attr;
■■int■■flags;
■■TStringArray list = new TStringArray;
■■
■■m_PathToMissions = "$saves:\\missions";
■■
■■string path_find_pattern = m_PathToMissions+"\\*"; //*/
■■FindFileHandle file_handler = FindFile(path_find_pattern, file_name, file_attr, flags);
■■
■■bool found = true;
■■while ( found )
■■{■■■■
■■■if ( file_attr & FileAttr.DIRECTORY )
■■■{
■■■■list.Insert(file_name);
■■■}
■■■
■■■found = FindNextFile(file_handler, file_name, file_attr);;
■■}
■■
■■return list;
■}
```

```
// ----------------------------------------------
// ------ missionMainMenu.c - START --------------------
// ----------------------------------------------


class MissionMainMenu extends MissionBase
{
■private UIScriptedMenu m_mainmenu;
■private CreditsMenu m_CreditsMenu;
■private ref DayZIntroScenePC m_IntroScenePC;
■private ref DayZIntroSceneXbox m_IntroSceneXbox;
■private AbstractWave m_MenuMusic;
■bool m_NoCutscene;

■override void OnInit()
■{
■■if (!m_NoCutscene)
■■{
■■■CreateIntroScene();
■■}
■■
■■if (!m_mainmenu)
■■{
■■■#ifdef PLATFORM_CONSOLE
■■■if ( g_Game.GetGameState() != DayZGameState.PARTY )
■■■{
■■■■m_mainmenu = UIScriptedMenu.Cast( g_Game.GetUIManager().EnterScriptedMenu( MENU_TITL
■■■}
■■■#else
■■■■m_mainmenu = UIScriptedMenu.Cast( g_Game.GetUIManager().EnterScriptedMenu( MENU_MAIN
■■■#endif
■■}
■■
■■GetOnInputDeviceChanged().Insert(OnInputDeviceChanged);
■}
■
■override void Reset()
■{
■■#ifdef PLATFORM_CONSOLE
■■■delete m_IntroSceneXbox;
■■#else
■■■delete m_IntroScenePC;
■■#endif
■■
■■CreateIntroScene();
■}
■
■DayZIntroScenePC GetIntroScenePC()
■{
■■#ifdef PLATFORM_CONSOLE
■■■Error("missionMainMenu->GetIntroScenePC on PLATFORM_CONSOLE is not implemented!");
■■■return null;
■■#else
■■■return m_IntroScenePC;
■■#endif
■}
■
■DayZIntroSceneXbox GetIntroSceneXbox()
■{
■■#ifdef PLATFORM_CONSOLE
■■■return m_IntroSceneXbox;
■■#else
■■■Error("missionMainMenu->GetIntroScenePC on PLATFORM_PC is not implemented!");
```

```
// ---------------------------------------------
// ------ missionServer.c - START --------------------
// ---------------------------------------------


//! int  time of the logout end
//: string  uid of the player
typedef Param2<int, string> LogoutInfo;

class MissionServer extends MissionBase
{
■ref array<Man> m_Players;
■ref array<ref CorpseData> m_DeadPlayersArray;
■ref map<PlayerBase, ref LogoutInfo> m_LogoutPlayers;
■ref map<PlayerBase, ref LogoutInfo> m_NewLogoutPlayers;
■const int SCHEDULER_PLAYERS_PER_TICK = 5;
■int m_currentPlayer;
■int m_RespawnMode;
■
■// -----------------------
■// ARTILLERY SOUNDS SETUP
■// -----------------------
■private float ■■■■m_ArtyBarrageTimer = 0;■■■// This is not to be edited in Init.c this is just to incremen
■
■// Variables to be modified in Init.c
■protected bool ■■■■m_PlayArty = false; ■■■// Toggle if Off map artillery sounds are played
■protected float ■■■m_ArtyDelay = 0; ■■■■// Set how much time there is between two barrages (in sec
■protected int ■■■■m_MinSimultaneousStrikes = 0;■// The MIN of simultaneous shots on the map (Will
■protected int ■■■■m_MaxSimultaneousStrikes = 0;■// The MAX of simultaneous shots on the map (Wil
■protected ref array<vector> m_FiringPos; ■■■■■// Where we should fire from. On Init set the relevant d
■
■//All Chernarus firing coordinates
■protected const ref array<vector> CHERNARUS_STRIKE_POS =
■{
■■"-500.00 165.00 5231.69",
■■"-500.00 300.00 9934.41",
■■"10406.86 192.00 15860.00",
■■"4811.75 370.00 15860.00",
■■"-500.00 453.00 15860.00"
■};
■
■//All livonia firing coordinates
■protected const ref  array<vector> LIVONIA_STRIKE_POS =
■{
■■"7440.00 417.00 -500.00",
■■"-500.00 276.00 5473.00",
■■"-500.00 265.00 9852.00",
■■"4953.00 240.00 13300.00",
■■"9620.00 188.00 13300.00",
■■"13300.00 204.00 10322.00",
■■"13300.00 288.00 6204.00",
■■"13300.00 296.00 -500.00"
■};
■// -----------------------
■// END OF ARTILLERY SETUP
■// -----------------------
■
■PlayerBase m_player;
■MissionBase m_mission;
■PluginAdditionalInfo m_moduleDefaultCharacter;
■
■void MissionServer()
■{
```

```
// ---------------------------------------------
// ------ MKII.c - START --------------------
// ---------------------------------------------


class MKII_Base : Pistol_Base
{
■override RecoilBase SpawnRecoilObject()
■{
■■return new MkiiRecoil(this);
■}
■
■//Debug menu Spawn Ground Special
■override void OnDebugSpawn()
■{
■■super.OnDebugSpawn();
■}
};



// ---------------------------------------------
// ------ MKII.c - END ----------------------
// ---------------------------------------------


// ---------------------------------------------
// ------ MkiiRecoil.c - START --------------------
// ---------------------------------------------


class MkiiRecoil: RecoilBase
{
■override void Init()
■{
■■vector point_1;
■■vector point_2;
■■vector point_3;
■■vector point_4;
■■point_1[0] = m_Player.GetRandomGeneratorSyncManager().GetRandomInRange(RandomGeneratorS
■■point_2[0] = m_Player.GetRandomGeneratorSyncManager().GetRandomInRange(RandomGeneratorS
■■point_3[0] = m_Player.GetRandomGeneratorSyncManager().GetRandomInRange(RandomGeneratorS
■■point_4[0] = m_Player.GetRandomGeneratorSyncManager().GetRandomInRange(RandomGeneratorS
■■m_HandsCurvePoints.Insert(point_1);//forms a 2 dimensional spline(z is ignored)
■■m_HandsCurvePoints.Insert(point_2);
■■m_HandsCurvePoints.Insert(point_3);
■■m_HandsCurvePoints.Insert(point_4);
■■m_HandsCurvePoints.Insert("0 0 0");
■■m_HandsOffsetRelativeTime = 1;
■■
■■m_MouseOffsetRangeMin = 45;//in degrees min
■■m_MouseOffsetRangeMax = 80;//in degrees max
■■m_MouseOffsetDistance = 0.65;//how far should the mouse travel
■■m_MouseOffsetRelativeTime = 0.5;//[0..1] a time it takes to move the mouse the required distance rela
■
■■m_CamOffsetDistance = 0.02;
■■m_CamOffsetRelativeTime = 1;
■}
}


// ---------------------------------------------
// ------ MkiiRecoil.c - END ----------------------
// ---------------------------------------------
// ------ ModifierBase.c - START --------------------
// ---------------------------------------------


enum eModifiersTickType//bitmask
{
■TICK ■■■■■= 1,
■ACTIVATE_CHECK ■■■= 2,
■DEACTIVATE_CHECK ■■= 4,
}


class ModifierBase
{
■int■■■■■m_ID = 0;
■ModifiersManager ■m_Manager;//the manager instance
■string ■■■■m_System = "Modifiers";
■float ■■■■m_ActivatedTime;//overall time this modifier was active
■bool■■■■m_TrackActivatedTime;//should this modifier track overall time it was active ?
■bool■■■■m_IsPersistent;//is this modifier saved to the DB ?
■PlayerBase■■■m_Player;
■float■■■■m_TickIntervalInactive = 5;
■float■■■■m_TickIntervalActive = 3;
■bool■■■■m_IsActive;
■bool■■■■m_ShouldBeActive;
■float■■■■m_AccumulatedTimeActive;
■float■■■■m_AccumulatedTimeInactive;
■float■■■■m_LastTickedActive;
■int ■■■■m_TickType = (eModifiersTickType.TICK | eModifiersTickType.ACTIVATE_CHECK | eModifier
■float■■■■m_LastTickedInactive;
■bool■■■■m_IsLocked = false;
■EActivationType■■m_ActivationType;
■eModifierSyncIDs■m_SyncID;//max 32 synced modifiers supported, 0 == no sync
■PluginPlayerStatus ■■m_ModulePlayerStatus;

■void ModifierBase()
■{
■■Class.CastTo(m_ModulePlayerStatus, GetPlugin(PluginPlayerStatus));
■}
■
■void InitBase(PlayerBase player, ModifiersManager manager)
■{
■■m_Manager ■■■■= manager;
■■m_Player ■■■■= player;
■■Init();
■}
■
■void Init(){}


■PlayerBase GetPlayer()
■{
■■return m_Player;
■}
■
■bool IsPersistent()
■{
■■return m_IsPersistent;
■}

■void MakeParamObjectPersistent(Param object)
■{
```

```
// ----------------------------------------------
// ------ ModifiersManager.c - START --------------------
// ----------------------------------------------


//max 32 synced modifiers supported
enum eModifierSyncIDs
{

■MODIFIER_SYNC_WOUND_INFECT_1 ■= 0x00000001,
■MODIFIER_SYNC_WOUND_INFECT_2 ■= 0x00000002,
■MODIFIER_SYNC_CONTAMINATION ■= 0x00000004,//stage1
■MODIFIER_SYNC_CONTAMINATION2 ■= 0x00000008,//stage2 and stage3 share the same sync id
■MODIFIER_SYNC_ZONE_EXPOSURE ■= 0x00000010,
■MODIFIER_SYNC_DROWNING ■■■= 0x00000020,
■//0x00000040,
■//0x00000080,
■//0x00000100,
■//0x00000200,
■//0x00000400,
■//0x00000800,
■//0x00001000,
■LAST_INDEX,
};




enum EActivationType {

■TRIGGER_EVENT_OFF,
■TRIGGER_EVENT_ON_ACTIVATION,
■TRIGGER_EVENT_ON_CONNECT
};

const int DEFAULT_TICK_TIME_ACTIVE = 3;
const int DEFAULT_TICK_TIME_ACTIVE_SHORT = 1;
const int DEFAULT_TICK_TIME_INACTIVE = 3;
const int DEFAULT_TICK_TIME_INACTIVE_LONG = 10;

class ModifierDebugObj
{
■string ■m_Name;
■int ■m_ID;
■bool ■m_IsActive;
■bool■m_IsLocked;
■
■void ModifierDebugObj(int id, string name, bool active, bool locked)
■{
■■m_Name = name;
■■m_ID = id;
■■m_IsActive = active;
■■m_IsLocked = locked;
■}
■
■string GetName()
■{
■■return m_Name;
■}
■
■int GetID()
■{
■■return m_ID;
```

```
// ---------------------------------------------
// ------ ModInfo.c - START --------------------
// ---------------------------------------------


class ModInfo
{
■proto owned string GetName();
■proto owned string GetPicture();
■proto owned string GetLogo();
■proto owned string GetLogoSmall();
■proto owned string GetLogoOver();
■proto owned string GetTooltip();
■proto owned string GetOverview();
■proto owned string GetAction();
■proto owned string GetAuthor();
■proto owned string GetVersion();
■proto bool GetDefault();
■proto bool GetIsDLC();
■proto bool GetIsOwned();
■proto void GoToStore();
}


// ---------------------------------------------
// ------ ModInfo.c - END ----------------------
// ---------------------------------------------


// ---------------------------------------------
// ------ ModLoader.c - START --------------------
// ---------------------------------------------


class ModLoader
{
■protected static bool■■■■■■■m_Loaded;
■protected static ref array<ref ModStructure>■m_Mods;
■
■static array<ref ModStructure> GetMods()
■{
■■//if( !m_Loaded )
■■■LoadMods();
■■return m_Mods;
■}
■
■static void LoadMods()
■{
■■m_Mods = new array<ref ModStructure>;
■■
■■int mod_count = GetGame().ConfigGetChildrenCount( "CfgMods" );
■■
■■for( int i = 2; i < mod_count; i++ )
■■{
■■■string mod_name;
■■■GetGame().ConfigGetChildName( "CfgMods", i, mod_name );
■■■m_Mods.Insert( new ModStructure( i, "CfgMods " + mod_name ) );
■■}
■}
}


// ---------------------------------------------
// ---------------------------------------------
// ------ ModsMenuDetailed.c - START --------------------
// ---------------------------------------------


class ModsMenuDetailed extends ScriptedWidgetEventHandler
{
■protected Widget■■■■■■■■■■■m_Root;
■protected Widget■■■■■■■■■■■m_Content;
■protected Widget■■■■■■■■■■■m_CloseButton;
■protected ScrollWidget■■■■■■■■■■m_Scroll;
■protected ref map<ref ModInfo, ref ModsMenuDetailedEntry>■m_Data;
■
■protected ModInfo■■■■■■■■■■■m_Highlighted;
■
■//protected MainMenu■■■■■■■■■■■m_Menu;
■protected UIScriptedMenu■■■■■■■■■■■m_Menu;
■protected ModsMenuTooltip■■■■■■■■■m_Tooltip;
■protected ref Timer■■■■■■■■■■■m_TooltipTimer;
■protected ModInfo■■■■■■■■■■■m_TooltipMod;

■void ModsMenuDetailed(array<ref ModInfo> data, Widget parent, ModsMenuTooltip tooltip, UIScriptedM
■{
■■m_Root = GetGame().GetWorkspace().CreateWidgets("gui/layouts/new_ui/mods_menu/mods_menu_d
■■m_Content = m_Root.FindAnyWidget("ModsDetailedContent");
■■m_Scroll = ScrollWidget.Cast(m_Root.FindAnyWidget("ModsDetailedScroller"));
■■m_CloseButton = m_Root.FindAnyWidget("ModsDetailedHeaderButton");
■■
■■m_Menu = menu_parent;
■■m_Data = new map<ref ModInfo, ref ModsMenuDetailedEntry>;
■■m_Tooltip = tooltip;
■■
■■m_Root.SetHandler(this);

■■LoadEntries(data);
■}
■
■void ~ModsMenuDetailed()
■{
■■delete m_Root;
■}
■
■void Open()
■{
■■if( !m_Root.IsVisible() )
■■■m_Scroll.VScrollToPos( 0 );
■■m_Root.Show( true );
■}
■
■void Close()
■{
■■Highlight( null );
■■m_Root.Show( false );
■}
■
■bool IsOpen()
■{
■■return m_Root.IsVisible();
■}
■
■ModInfo GetHighlighted()
■{
■■return m_Highlighted;
```

```
// ---------------------------------------------
// ------ ModsMenuDetailedEntry.c - START --------------------
// ---------------------------------------------


class ModsMenuDetailedEntry extends ScriptedWidgetEventHandler
{
■protected Widget■■■m_Root;
■protected Widget■■■m_Detail;
■
■//Header
■protected ImageWidget■■m_IconSmall;
■protected ImageWidget■■m_IconCollapse;
■protected TextWidget■■m_Name;
■
■//Left Side Panel
■protected ImageWidget■■m_IconBig;
■protected MultilineTextWidget■■m_Author;
■protected TextWidget■■m_Version;
■protected RichTextWidget■m_ActionWebsite;
■protected RichTextWidget■m_ActionPurchase;
■
■//Description Panel
■protected RichTextWidget■m_Description;

■protected ModInfo■■■m_Data;
■protected ModsMenuDetailed■m_ParentMenu;
■protected bool■■■■m_IsOpen;

■void ModsMenuDetailedEntry(ModInfo data, Widget parent, ModsMenuDetailed parent_menu)
■{
■■m_Root = GetGame().GetWorkspace().CreateWidgets("gui/layouts/new_ui/mods_menu/mods_menu_c
■■m_Detail = m_Root.FindAnyWidget("DetailContainer");
■■
■■m_IconSmall = ImageWidget.Cast(m_Root.FindAnyWidget("IconSmall"));
■■m_IconCollapse = ImageWidget.Cast(m_Root.FindAnyWidget("collapse_button"));
■■m_IconCollapse.LoadImageFile( 1, "set:dayz_gui image:icon_open" );
■■m_Name = TextWidget.Cast(m_Root.FindAnyWidget("Name"));
■■
■■m_IconBig = ImageWidget.Cast(m_Root.FindAnyWidget("IconBig"));
■■m_Author = MultilineTextWidget.Cast(m_Root.FindAnyWidget("Author"));
■■m_Author.SetLineBreakingOverride(LinebreakOverrideMode.LINEBREAK_WESTERN);
■■
■■m_Version = TextWidget.Cast(m_Root.FindAnyWidget("Version"));
■■m_ActionWebsite = RichTextWidget.Cast(m_Root.FindAnyWidget("Link"));
■■m_ActionPurchase = RichTextWidget.Cast(m_Root.FindAnyWidget("Purchase"));
■■
■■m_Description = RichTextWidget.Cast(m_Root.FindAnyWidget("Description"));
■■
■■m_Data = data;
■■m_ParentMenu = parent_menu;
■■
■■m_Root.SetHandler(this);

■■LoadData();
■}
■
■void ~ModsMenuDetailedEntry()
■{
■■delete m_Root;
■}
■
■Widget GetWidget()
```

```
// ---------------------------------------------
// ------ ModsMenuSimple.c - START --------------------
// ---------------------------------------------


class ModsMenuSimple extends ScriptedWidgetEventHandler
{
■protected Widget■■■■■■■■■■m_Root;
■protected Widget■■■■■■■■■■m_MoreButton;
■protected ref map<ModInfo, ref ModsMenuSimpleEntry>■■m_Data;
■protected ModsMenuDetailed■■■■■■■■m_DetailMenu;
■
■void ModsMenuSimple(array<ref ModInfo> data, Widget parent, ModsMenuDetailed detail_menu)
■{
■■m_Root = GetGame().GetWorkspace().CreateWidgets("gui/layouts/new_ui/mods_menu/mods_menu_s
■■m_MoreButton = m_Root.FindAnyWidget("ModMore");
■■m_Data = new map<ModInfo, ref ModsMenuSimpleEntry>;
■■m_DetailMenu = detail_menu;
■■
■■m_Root.SetHandler(this);
■■LoadEntries(data);
■}
■
■void ~ModsMenuSimple()
■{
■■delete m_Root;
■}

■void LoadEntries(array<ref ModInfo> data)
■{
■■int count;
■■if( data.Count() > 13 )
■■{
■■■count = 13;
■■■m_Root.FindAnyWidget("ModMore").Show(true);
■■}
■■else
■■{
■■■count = data.Count();
■■■m_Root.FindAnyWidget("ModMore").Show(false);
■■}
■■
■■bool has_mods = false;
■■for (int i = count - 1; i >= 0; i--)
■■{
■■■ref ModsMenuSimpleEntry entry = new ModsMenuSimpleEntry(data.Get(i), i, m_Root, this);
■■■m_Data.Insert(data.Get(i), entry);
■■■has_mods = ( has_mods || !data.Get(i).GetIsDLC() );
■■}
■■
■■m_Root.FindAnyWidget("mods_separator_panel").Show( has_mods );
■}
■
■void Select( ModInfo mod )
■{
■■m_DetailMenu.Open();
■■m_DetailMenu.Highlight( mod );
■}
■
■override bool OnMouseButtonUp(Widget w, int x, int y, int button)
■{
■■if( w == m_MoreButton )
■■{
```

```
// ----------------------------------------------
// ------ ModsMenuSimpleEntry.c - START --------------------
// ----------------------------------------------


class ModsMenuSimpleEntry extends ScriptedWidgetEventHandler
{
■protected ButtonWidget■■m_ModButton;
■protected ImageWidget■■m_Icon;
■protected Widget■■■m_Hover;

■protected bool■■■■m_HasLogoOver;
■protected ModInfo■■■m_Data;
■protected ModsMenuSimple■m_ParentMenu;

■void ModsMenuSimpleEntry(ModInfo data, int index, Widget parent, ModsMenuSimple parent_menu)
■{
■■m_ModButton■■= ButtonWidget.Cast(GetGame().GetWorkspace().CreateWidgets("gui/layouts/new_u
■■m_Icon■■■= ImageWidget.Cast(m_ModButton.FindAnyWidget("Icon"));
■■m_Hover■■■= m_ModButton.FindAnyWidget("Overlay");
■■m_Data■■■= data;
■■m_ParentMenu■= parent_menu;
■■
■■if ( data.GetIsDLC() )
■■{
■■■bool isOwned = data.GetIsOwned();
■■■m_ModButton.SetSort( index );
■■■m_ModButton.FindAnyWidget("ModOwnership").Show( true );
■■■m_ModButton.FindAnyWidget("Owned").Show( isOwned );
■■■m_ModButton.FindAnyWidget("Unowned").Show( !isOwned );
■■}
■■else
■■{
■■■m_ModButton.SetSort( index + 5 );
■■}
■■
■■
■■m_ModButton.SetHandler(this);

■■LoadData();
■}
■
■void ~ModsMenuSimpleEntry()
■{
■■delete m_ModButton;
■}

■void LoadData()
■{
■■string logo = m_Data.GetLogo();
■■string logo_over = m_Data.GetLogoOver();
■■
■■if (logo != "")
■■{
■■■m_Icon.LoadImageFile(0, logo);
■■}
■■else
■■{
■■■//Load default image
■■}
■■
■■if (logo_over != "")
■■{
```

```
// ---------------------------------------------
// ------ ModsMenuTooltip.c - START --------------------
// ---------------------------------------------


class ModsMenuTooltip extends ScriptedWidgetEventHandler
{
■protected Widget■■■m_Root;
■protected RichTextWidget■m_Text;

■void ModsMenuTooltip(Widget parent)
■{
■■m_Root = GetGame().GetWorkspace().CreateWidgets("gui/layouts/new_ui/mods_menu/mods_tooltip.la
■■m_Text = RichTextWidget.Cast(m_Root.FindAnyWidget("TooltipText"));
■}
■
■void ~ModsMenuTooltip()
■{
■■delete m_Root;
■}
■
■void ShowTooltip(ModInfo mod_ref)
■{
■■m_Root.Show(true);
■■m_Text.SetText(mod_ref.GetTooltip());
■■
■■int x, y;
■■GetMousePos(x,y);
■■m_Root.SetPos(x, y);
■■
■■m_Text.Update();
■■m_Root.Update();
■}

■void HideTooltip()
■{
■■m_Root.Show(false);
■■m_Root.SetPos(0, 0);
■■m_Text.SetText("");
■}
}


// ---------------------------------------------
// ------ ModsMenuTooltip.c - END ---------------------
// ---------------------------------------------


// ---------------------------------------------
// ------ ModStructure.c - START --------------------
// ---------------------------------------------


class ModStructure
{
■protected int■■m_ModIndex;
■protected string■m_ModPath;
■protected string■m_ModName;
■protected string■m_ModLogo;
■protected string■m_ModLogoSmall;
■protected string■m_ModLogoOver;
■protected string■m_ModActionURL;
■protected string■m_ModTooltip;
```

```
// ---------------------------------------------
// ------ Morphine.c - START --------------------
// ---------------------------------------------


class Morphine: Inventory_Base
{
■override void SetActions()
■{
■■super.SetActions();
■■
■■AddAction(ActionInjectMorphineTarget);
■■AddAction(ActionInjectMorphineSelf);
■}
■
■override void OnApply(PlayerBase player)
■{
■■if (!player)
■■■return;
■■if( player.GetModifiersManager().IsModifierActive(eModifiers.MDF_MORPHINE ) )//effectively resets th
■■{
■■■player.GetModifiersManager().DeactivateModifier( eModifiers.MDF_MORPHINE );
■■}
■■player.GetModifiersManager().ActivateModifier( eModifiers.MDF_MORPHINE );
■}
};



// ---------------------------------------------
// ------ Morphine.c - END ----------------------
// ---------------------------------------------


// ---------------------------------------------
// ------ MorphineMdfr.c - START --------------------
// ---------------------------------------------


class MorphineMdfr: ModifierBase
{
■const int LIFETIME = 60;
■
■override void Init()
■{
■■m_TrackActivatedTime = true;
■■m_IsPersistent = true;
■■m_ID ■■■■■= eModifiers.MDF_MORPHINE;
■■m_TickIntervalInactive ■= DEFAULT_TICK_TIME_INACTIVE;
■■m_TickIntervalActive ■= 1;
■■DisableActivateCheck();
■}

■override bool ActivateCondition(PlayerBase player)
■{
■■return false;
■}
■
■override void OnReconnect(PlayerBase player)
■{
■■OnActivate(player);
■}
■
```

```
// ---------------------------------------------
// ------ Mosin9130.c - START --------------------
// ---------------------------------------------


/**@class■■Mosin9130
 * @brief■■base for Mosin
 * @NOTE■■name copies config base class
 **/
class Mosin9130_Base extends BoltActionRifle_InnerMagazine_Base
{
■void Mosin9130_Base ()
■{
■}
■
■override bool CanEnterIronsights()
■{
■■ItemOptics optic = GetAttachedOptics();
■■if (optic && PUScopeOptic.Cast(optic))
■■■return true;
■■return super.CanEnterIronsights();
■}
};


class Mosin9130 extends Mosin9130_Base
{
■override RecoilBase SpawnRecoilObject()
■{
■■return new MosinRecoil(this);
■}
■
■■■■
■//Debug menu Spawn Ground Special
■override void OnDebugSpawn()
■{
■■super.OnDebugSpawn();
■■GameInventory inventory = GetInventory();

■■inventory.CreateInInventory( "PUScopeOptic" );
■■inventory.CreateInInventory( "Mosin_Compensator" );■
■■
■■EntityAI entity;
■■if ( Class.CastTo(entity, this) )
■■{
■■■entity.SpawnEntityOnGroundPos("Ammo_762x54", entity.GetPosition());
■■}
■}
};

class SawedoffMosin9130_Base extends Mosin9130_Base
{
■override RecoilBase SpawnRecoilObject()
■{
■■return new MosinSawedOffRecoil(this);
■}
};



// ---------------------------------------------
// ------ Mosin9130.c - END ----------------------
// ---------------------------------------------
// ---------------------------------------------
// ------ MosinRecoil.c - START --------------------
// ---------------------------------------------


class MosinRecoil: RecoilBase
{
■override void Init()
■{
■■vector point_1;
■■vector point_2;
■■vector point_3;
■■vector point_4;
■■point_1[0] = m_Player.GetRandomGeneratorSyncManager().GetRandomInRange(RandomGeneratorS
■■point_2[0] = m_Player.GetRandomGeneratorSyncManager().GetRandomInRange(RandomGeneratorS
■■point_3[0] = m_Player.GetRandomGeneratorSyncManager().GetRandomInRange(RandomGeneratorS
■■point_4[0] = m_Player.GetRandomGeneratorSyncManager().GetRandomInRange(RandomGeneratorS
■■m_HandsCurvePoints.Insert(point_1);//forms a 2 dimensional spline(z is ignored)
■■m_HandsCurvePoints.Insert(point_2);
■■m_HandsCurvePoints.Insert(point_3);
■■m_HandsCurvePoints.Insert(point_4);
■■m_HandsCurvePoints.Insert("0 0 0");
■■m_HandsOffsetRelativeTime = 0.125;■
■■
■■m_MouseOffsetRangeMin = 80;//in degrees min
■■m_MouseOffsetRangeMax = 100;//in degrees max
■■m_MouseOffsetDistance = 1.8;//how far should the mouse travel
■■m_MouseOffsetRelativeTime = 0.0625;//[0..1] a time it takes to move the mouse the required distance
■■■
■■m_CamOffsetDistance = 0.02;
■■m_CamOffsetRelativeTime = 0.125;
■}
}


// ---------------------------------------------
// ------ MosinRecoil.c - END ----------------------
// ---------------------------------------------


// ---------------------------------------------
// ------ MosinSawedOffRecoil.c - START --------------------
// ---------------------------------------------


class MosinSawedOffRecoil: RecoilBase
{
■override void Init()
■{
■■vector point_1;
■■vector point_2;
■■vector point_3;
■■vector point_4;
■■point_1[0] = m_Player.GetRandomGeneratorSyncManager().GetRandomInRange(RandomGeneratorS
■■point_2[0] = m_Player.GetRandomGeneratorSyncManager().GetRandomInRange(RandomGeneratorS
■■point_3[0] = m_Player.GetRandomGeneratorSyncManager().GetRandomInRange(RandomGeneratorS
■■point_4[0] = m_Player.GetRandomGeneratorSyncManager().GetRandomInRange(RandomGeneratorS
■■m_HandsCurvePoints.Insert(point_1);//forms a 2 dimensional spline(z is ignored)
■■m_HandsCurvePoints.Insert(point_2);
■■m_HandsCurvePoints.Insert(point_3);
■■m_HandsCurvePoints.Insert(point_4);
■■m_HandsCurvePoints.Insert("0 0 0");
■■m_HandsOffsetRelativeTime = 0.125;■
```

```
// ---------------------------------------------
// ------ Mosin_Bayonet.c - START --------------------
// ---------------------------------------------


class Mosin_Bayonet extends Inventory_Base
{
■override bool IsMeleeFinisher()
■{
■■return true;
■}
■
■override array<int> GetValidFinishers()
■{
■■return {EMeleeHitType.FINISHER_LIVERSTAB,EMeleeHitType.FINISHER_NECKSTAB};
■}
■
■override bool CanPutAsAttachment( EntityAI parent )
■{
■■if(!super.CanPutAsAttachment(parent)) {return false;}
■■if ( parent.FindAttachmentBySlotName("suppressorImpro") == null && parent.FindAttachmentBySlotNa
■■{
■■■return true;
■■}
■■return false;
■}

■override void OnWasAttached(EntityAI parent, int slot_id)
■{
■■super.OnWasAttached(parent, slot_id);
■■
■■if( parent.IsWeapon() )
■■{
■■■parent.SetBayonetAttached(true,slot_id);
■■}■■
■}
■
■override void OnWasDetached(EntityAI parent, int slot_id)
■{
■■super.OnWasDetached(parent, slot_id);
■■
■■if( parent.IsWeapon() )
■■{
■■■parent.SetBayonetAttached(false);
■■}
■}
■
■override void SetActions()
■{
■■super.SetActions();
■■
■■AddAction(ActionBurnSewTarget);
■■AddAction(ActionUnrestrainTarget);
■■AddAction(ActionBurnSewSelf);
■■AddAction(ActionDigWorms);
■}
}


// ---------------------------------------------
// ------ Mosin_Bayonet.c - END ---------------------
// ---------------------------------------------
// ---------------------------------------------
// ------ Mosin_Compensator.c - START --------------------
// ---------------------------------------------


class Mosin_Compensator extends ItemSuppressor
{
■override bool CanPutAsAttachment( EntityAI parent )
■{
■■if(!super.CanPutAsAttachment(parent)) {return false;}
■■if ( parent.FindAttachmentBySlotName("suppressorImpro") == NULL && parent.FindAttachmentBySlot
■■{
■■■return true;
■■}
■■return false;
■}
}


// ---------------------------------------------
// ------ Mosin_Compensator.c - END ----------------------
// ---------------------------------------------


// ---------------------------------------------
// ------ MotoHelmet_ColorBase.c - START --------------------
// ---------------------------------------------


class MotoHelmet_ColorBase extends HelmetBase
{
■override array<int> GetEffectWidgetTypes()
■{
■■return {EffectWidgetsTypes.MOTO_OCCLUDER/*,EffectWidgetsTypes.MOTO_BREATH*/};
■}
■
■override bool CanPutAsAttachment( EntityAI parent )
■{
■■if(!super.CanPutAsAttachment(parent)) {return false;}
■■
■■Clothing eyewear = Clothing.Cast(parent.FindAttachmentBySlotName("Eyewear"));
■■if ( eyewear && eyewear.ConfigGetBool("isStrap") )
■■{
■■■return false;
■■}
■■
■■Clothing mask = Clothing.Cast(parent.FindAttachmentBySlotName("Mask"));
■■if ( mask && mask.ConfigGetBool("noHelmet") ) //TODO
■■{
■■■return false;
■■}
■■
■■return true;
■}

■override bool IsObstructingVoice()
■{
■■return true;
■}
■
■override int GetVoiceEffect()
■{
■■return VoiceEffectObstruction;
```

```
// ---------------------------------------------
// ------ MouflonSteakMeat.c - START --------------------
// ---------------------------------------------


class MouflonSteakMeat extends Edible_Base
{
■override bool CanBeCooked()
■{
■■return true;
■}■■
■
■override bool CanBeCookedOnStick()
■{
■■return true;
■}
■
■override bool IsMeat()
■{
■■return true;
■}
■
■override bool CanDecay()
■{
■■return true;
■}
■
■override void SetActions()
■{
■■super.SetActions();
■■
■■AddAction(ActionForceFeed);
■■AddAction(ActionEatMeat);
■■
■■AddAction(ActionCreateIndoorFireplace);
■■AddAction(ActionCreateIndoorOven);
■}
}



// ---------------------------------------------
// ------ MouflonSteakMeat.c - END ----------------------
// ---------------------------------------------


// ---------------------------------------------
// ------ MountainBag_ColorBase.c - START --------------------
// ---------------------------------------------


class MountainBag_ColorBase extends Clothing {};
class MountainBag_Red extends MountainBag_ColorBase {};
class MountainBag_Blue extends MountainBag_ColorBase {};
class MountainBag_Orange extends MountainBag_ColorBase {};
class MountainBag_Green extends MountainBag_ColorBase {};


// ---------------------------------------------
// ------ MountainBag_ColorBase.c - END ----------------------
// ---------------------------------------------
// ----------------------------------------------
// ------ MouseBinding.c - START --------------------
// ----------------------------------------------


class MouseBinding
{
■private int■■m_ActiveUIMenuID;
■private int■■m_MouseButton;
■private int■■m_MouseEvent;
■private string■m_CallbackTarget;
■private string■m_CallbackFunction;
■private string■m_InfoKeys;
■private string■m_InfoDescription
■
■void MouseBinding( int ui_id, int mouse_button, int mouse_event, string call_target, string call_function, s
■{
■■m_ActiveUIMenuID■= ui_id;
■■m_MouseButton■■= mouse_button;
■■m_MouseEvent■■= mouse_event;
■■m_CallbackTarget■= call_target;
■■m_CallbackFunction■= call_function;
■■m_InfoKeys■■■= info_bind;
■■m_InfoDescription■= info_description;
■}
■
■int GetUIMenuID()
■{
■■return m_ActiveUIMenuID;
■}
■
■int GetButtonID()
■{
■■return m_MouseButton;
■}
■
■int GetMouseEvent()
■{
■■return m_MouseEvent;
■}
■
■string GetCallbackTarget()
■{
■■return m_CallbackTarget;
■}
■
■string GetCallbackFunction()
■{
■■return m_CallbackFunction;
■}
■
■string GetInfoBind()
■{
■■return m_InfoKeys;
■}
■
■string GetInfoDescription()
■{
■■return m_InfoDescription;
■}
}
```

```
// ---------------------------------------------
// ------ MouseButtonInfo.c - START --------------------
// ---------------------------------------------


class MouseButtonInfo
{
■private int■■m_ButtonID;
■private int■■m_TimeLastPress;
■private int■■m_TimeLastRelease;
■
■void MouseButtonInfo(int button)
■{
■■m_ButtonID■■■= button;
■■m_TimeLastPress■■= -1;
■■m_TimeLastRelease■= -1;
■}
■
■int GetButtonID()
■{
■■return m_ButtonID;
■}
■
■int GetTimeLastPress()
■{
■■return m_TimeLastPress;
■}
■
■int GetTimeLastRelease()
■{
■■return m_TimeLastRelease;
■}
■
■void Press()
■{
■■m_TimeLastPress = GetGame().GetTime();
■}
■
■void Release()
■{
■■m_TimeLastRelease = GetGame().GetTime();
■}
■
■bool IsButtonDown()
■{
■■if ( m_TimeLastRelease < m_TimeLastPress )
■■{
■■■return true;
■■}
■■
■■return false;
■}
}


// ---------------------------------------------
// ------ MouseButtonInfo.c - END ----------------------
// ---------------------------------------------
// ---------------------------------------------
// ------ MouthRag.c - START --------------------
// ---------------------------------------------


class MouthRag extends Clothing
{
■bool m_IncomingLambdaChange;
■
■void MouthRag()
■{
■■m_IncomingLambdaChange = false;
■}

■override bool CanDetachAttachment(EntityAI parent)
■{
■■return m_IncomingLambdaChange;
■}

■override bool IsObstructingVoice()
■{
■■return true;
■}
■
■override int GetVoiceEffect()
■{
■■return VoiceEffectObstruction;
■}
■
■void SetIncomingLambaBool(bool state)
■{
■■m_IncomingLambdaChange = state;
■}
■
■bool GetIncomingLambdaBool()
■{
■■return m_IncomingLambdaChange;
■}
};


// ---------------------------------------------
// ------ MouthRag.c - END ----------------------
// ---------------------------------------------


// ---------------------------------------------
// ------ MP133.c - START --------------------
// ---------------------------------------------


enum PumpShotgunAnimState
{
■DEFAULT ■■■= 0, ■///< default weapon state, closed and discharged
};

enum PumpShotgunStableStateID
{
■UNKNOWN■■=  0,
■Empty■■=  1,
■Fireout■■=  2,
■Loaded■■=  3,
■Jammed■■=  4
```

```
// ---------------------------------------------
// ------ Mp133Recoil.c - START --------------------
// ---------------------------------------------


class Mp133Recoil: RecoilBase
{
■override void Init()
■{
■■vector point_1;
■■vector point_2;
■■vector point_3;
■■vector point_4;
■■point_1[0] = m_Player.GetRandomGeneratorSyncManager().GetRandomInRange(RandomGeneratorS
■■point_2[0] = m_Player.GetRandomGeneratorSyncManager().GetRandomInRange(RandomGeneratorS
■■point_3[0] = m_Player.GetRandomGeneratorSyncManager().GetRandomInRange(RandomGeneratorS
■■point_4[0] = m_Player.GetRandomGeneratorSyncManager().GetRandomInRange(RandomGeneratorS
■■m_HandsCurvePoints.Insert(point_1);//forms a 2 dimensional spline(z is ignored)
■■m_HandsCurvePoints.Insert(point_2);
■■m_HandsCurvePoints.Insert(point_3);
■■m_HandsCurvePoints.Insert(point_4);
■■m_HandsCurvePoints.Insert("0 0 0");
■■m_HandsOffsetRelativeTime = 0.125;
■■
■■m_MouseOffsetRangeMin = 70;//in degrees min
■■m_MouseOffsetRangeMax = 110;//in degrees max
■■m_MouseOffsetDistance = 3;//how far should the mouse travel
■■m_MouseOffsetRelativeTime = 0.0625; //0.03;//[0..1] a time it takes to move the mouse the required dis
■■■■
■■m_CamOffsetDistance = 0.05;
■■m_CamOffsetRelativeTime = 0.125;
■}
}


// ---------------------------------------------
// ------ Mp133Recoil.c - END ----------------------
// ---------------------------------------------


// ---------------------------------------------
// ------ MP5.c - START --------------------
// ---------------------------------------------



/**@class■MP5K_Base
 * @brief■basic mp5 submachine gun
 **/
class MP5K_Base : RifleBoltFree_Base
{
■void MP5K_Base ()
■{
■}
■
■override RecoilBase SpawnRecoilObject()
■{
■■return new Mp5kRecoil(this);
■}
■
■
■override int GetWeaponSpecificCommand(int weaponAction ,int subCommand)
■{
```

```
// ---------------------------------------------
// ------ Mp5kRecoil.c - START --------------------
// ---------------------------------------------


class Mp5kRecoil: RecoilBase
{
■override void Init()
■{■■
■■vector point_1;
■■vector point_2;
■■vector point_3;
■■vector point_4;
■■point_1[0] = m_Player.GetRandomGeneratorSyncManager().GetRandomInRange(RandomGeneratorS
■■point_2[0] = m_Player.GetRandomGeneratorSyncManager().GetRandomInRange(RandomGeneratorS
■■point_3[0] = m_Player.GetRandomGeneratorSyncManager().GetRandomInRange(RandomGeneratorS
■■point_4[0] = m_Player.GetRandomGeneratorSyncManager().GetRandomInRange(RandomGeneratorS
■■m_HandsCurvePoints.Insert(point_1);//forms a 2 dimensional spline(z is ignored)
■■m_HandsCurvePoints.Insert(point_2);
■■m_HandsCurvePoints.Insert(point_3);
■■m_HandsCurvePoints.Insert(point_4);
■■m_HandsCurvePoints.Insert("0 0 0");
■■m_HandsOffsetRelativeTime = 0.25;
■■
■■m_MouseOffsetRangeMin = 60;//in degrees min
■■m_MouseOffsetRangeMax = 120;//in degrees max
■■m_MouseOffsetDistance = 1.25;//how far should the mouse travel
■■m_MouseOffsetRelativeTime = 0.5;//[0..1] a time it takes to move the mouse the required distance rela
■
■■m_CamOffsetDistance = 0.0075;
■■m_CamOffsetRelativeTime = 1;
■}
}


// ---------------------------------------------
// ------ Mp5kRecoil.c - END ----------------------
// ---------------------------------------------


// ---------------------------------------------
// ------ MP5_Compensator.c - START --------------------
// ---------------------------------------------


class MP5_Compensator extends ItemSuppressor
{
■override bool CanPutAsAttachment( EntityAI parent )
■{
■■if(!super.CanPutAsAttachment(parent)) {return false;}
■■if ( !parent.FindAttachmentBySlotName("pistolMuzzle") )
■■{
■■■return true;
■■}
■■return false;
■}
}


// ---------------------------------------------
// ------ MP5_Compensator.c - END ----------------------
// ---------------------------------------------
// ---------------------------------------------
// ------ MP5_PlasticHndgrd.c - START --------------------
// ---------------------------------------------


class MP5_PlasticHndgrd extends Inventory_Base
{
/*
■override bool CanDetachAttachment( EntityAI attachment )
■{
■■return false;
■}
*/
}


// ---------------------------------------------
// ------ MP5_PlasticHndgrd.c - END ----------------------
// ---------------------------------------------


// ---------------------------------------------
// ------ MP5_RailHndgrd.c - START --------------------
// ---------------------------------------------


class MP5_RailHndgrd extends Inventory_Base
{
■override bool CanDetachAttachment (EntityAI parent)
■{
■■if ( Weapon_Base.Cast(parent) && parent.FindAttachmentBySlotName("weaponFlashlight")/*.IsKindOf
■■{
■■■return false;
■■}
■■return true;
■}
}


// ---------------------------------------------
// ------ MP5_RailHndgrd.c - END ----------------------
// ---------------------------------------------


// ---------------------------------------------
// ------ MushroomBase.c - START --------------------
// ---------------------------------------------


class MushroomBase : Edible_Base
{
■override bool CanBeCookedOnStick()
■{
■■return true;
■}

■override bool CanBeCooked()
■{
■■return true;
■}■
■
■override bool IsMushroom()
■{
```

```
// ---------------------------------------------
// ------ MuzzleFlashLight.c - START --------------------
// ---------------------------------------------


class MuzzleFlashLight extends PointLightBase
{
■static float m_Lifetime = 0.04;
■
■void MuzzleFlashLight()
■{
■■SetLifetime(m_Lifetime);
■■FadeIn(m_Lifetime * 0.5);
■■SetFadeOutTime(m_Lifetime * 0.5);
■■SetVisibleDuringDaylight(true);
■■SetRadiusTo( 15 );
■■SetBrightnessTo(1);
■■SetCastShadow(false);
■■SetAmbientColor(0.76, 0.68, 0.31);
■■SetDiffuseColor(0.76, 0.68, 0.31);
■■SetFlareVisible(false);
■}
}

class MuzzleFlashLight_1 extends PointLightBase
{
■static float m_Lifetime = 0.08;
■
■void MuzzleFlashLight_1()
■{
■■SetLifetime( m_Lifetime );
■■FadeIn( m_Lifetime * 0.2 );
■■SetFadeOutTime( m_Lifetime * 0.2 );
■■SetVisibleDuringDaylight( true );
■■SetRadiusTo( 17 );
■■SetBrightnessTo( 0.5 );
■■SetCastShadow( false );
■■SetAmbientColor( 0.56, 0.42, 0.3 );
■■SetDiffuseColor( 0.56, 0.42, 0.3 );
■■SetFlareVisible( false );
■}
}

class MuzzleFlashLight_2 extends PointLightBase
{
■static float m_Lifetime = 0.01;
■
■void MuzzleFlashLight_2()
■{
■■SetLifetime( m_Lifetime );
■■FadeIn( m_Lifetime * 0.1 );
■■SetFadeOutTime( m_Lifetime * 0.1 );
■■SetVisibleDuringDaylight( true );
■■SetRadiusTo( 12 );
■■SetBrightnessTo( 0.2 );
■■SetCastShadow( false );
■■SetAmbientColor( 0.36, 0.32, 0.13 );
■■SetDiffuseColor( 0.36, 0.32, 0.13 );
■■SetFlareVisible( false );
■}
}
```

```
// ----------------------------------------------
// ------ Nail.c - START --------------------
// ----------------------------------------------


class Nails extends Inventory_Base
{
■override bool CanPutAsAttachment( EntityAI parent )
■{
■■if(!super.CanPutAsAttachment(parent)) {return false;}
■■const int SLOTS_ARRAY = 8;
■■bool is_barrel = false;
■■bool is_opened_barrel = false;
■■bool slot_test = true;
■■string slot_names[SLOTS_ARRAY] = { "BerryR", "BerryB", "Plant", "OakBark", "BirchBark", "Lime", "Dis

■■
■■// is barrel
■■if ( parent.IsKindOf("Barrel_ColorBase") )
■■{
■■■is_barrel = true;
■■}

■■// is opened barrel■■■■
■■if ( is_barrel && parent.GetAnimationPhase("Lid") == 1 )
■■{
■■■is_opened_barrel = true;
■■}

■■// all of the barrel slots are empty
■■for ( int i = 0; i < SLOTS_ARRAY ; i++ )
■■{
■■■if ( parent.FindAttachmentBySlotName(slot_names[i]) != NULL )
■■■{
■■■■slot_test = false;
■■■■break;
■■■}
■■}
■■
■■if ( ( is_opened_barrel && slot_test ) || !is_barrel )
■■{
■■■return true;
■■}
■■return false;
■}

■override bool CanDetachAttachment( EntityAI parent )
■{
■■
■■bool is_barrel = false;
■■bool is_opened_barrel = false;
■■
■■// is barrel
■■if ( parent.IsKindOf("Barrel_ColorBase") )
■■{
■■■is_barrel = true;
■■}

■■// is opened barrel■■■■
■■if ( is_barrel && parent.GetAnimationPhase("Lid") == 1 )
■■{
■■■is_opened_barrel = true;
■■}
```

```
// ---------------------------------------------
// ------ NailedBaseballBat.c - START --------------------
// ---------------------------------------------


class NailedBaseballBat extends Inventory_Base
{
}


// ---------------------------------------------
// ------ NailedBaseballBat.c - END ----------------------
// ---------------------------------------------


// ---------------------------------------------
// ------ NBCBootsBase.c - START --------------------
// ---------------------------------------------


class NBCBootsBase extends Clothing {};
class NBCBootsGray extends NBCBootsBase {};


// ---------------------------------------------
// ------ NBCBootsBase.c - END ----------------------
// ---------------------------------------------


// ---------------------------------------------
// ------ NBCGloves_ColorBase.c - START --------------------
// ---------------------------------------------


class NBCGloves_ColorBase extends Clothing {};
class NBCGlovesGray extends NBCGloves_ColorBase {};


// ---------------------------------------------
// ------ NBCGloves_ColorBase.c - END ----------------------
// ---------------------------------------------


// ---------------------------------------------
// ------ NBCHood.c - START --------------------
// ---------------------------------------------


class NBCHood extends Clothing
{
}



// ---------------------------------------------
// ------ NBCHood.c - END ----------------------
// ---------------------------------------------
// ---------------------------------------------
// ------ NBCJacketBase.c - START --------------------
// ---------------------------------------------


class NBCJacketBase extends Clothing
{
■override void SetActions()
■{
■■super.SetActions();
■■AddAction(ActionWringClothes);
■}
};
class NBCJacketGray extends NBCJacketBase {};


// ---------------------------------------------
// ------ NBCJacketBase.c - END ----------------------
// ---------------------------------------------


// ---------------------------------------------
// ------ NBCPantsBase.c - START --------------------
// ---------------------------------------------


class NBCPantsBase extends Clothing
{
■override void SetActions()
■{
■■super.SetActions();
■■AddAction(ActionWringClothes);
■}
};
class NBCPantsGray extends NBCPantsBase {};


// ---------------------------------------------
// ------ NBCPantsBase.c - END ----------------------
// ---------------------------------------------


// ---------------------------------------------
// ------ NerveAgent.c - START --------------------
// ---------------------------------------------


class NerveAgent extends AgentBase
{
■override void Init()
■{
■■m_Type ■■■■■= eAgents.NERVE_AGENT;
■■m_Invasibility ■■■= 0;
■■m_TransferabilityIn■■= 0.1;
■■m_TransferabilityOut■= 1;
■■m_TransferabilityAirOut■= 0;
■■m_AntibioticsResistance = 0.5;//override in a func. GetAntiboticsResistance()
■■m_MaxCount ■■■■= 500;
■■m_Potency ■■■■= EStatLevels.CRITICAL;
■■m_DieOffSpeed ■■■= 0.1;
■}
■
```

```
// ---------------------------------------------
// ------ NioshFaceMask.c - START --------------------
// ---------------------------------------------


class NioshFaceMask: Clothing
{
■override bool CanPutAsAttachment( EntityAI parent )
■{
■■if (!super.CanPutAsAttachment(parent)) {return false;}
■■
■■Clothing headgear = Clothing.Cast(parent.FindAttachmentBySlotName("Headgear"));
■■if ( headgear && headgear.ConfigGetBool("noMask") )
■■{
■■■return false;
■■}
■■
■■return true;
■}
}



// ---------------------------------------------
// ------ NioshFaceMask.c - END ----------------------
// ---------------------------------------------


// ---------------------------------------------
// ------ Noise.c - START --------------------
// ---------------------------------------------


class NoiseSystem
{
■private void NoiseSystem() {}
■private void ~NoiseSystem() {}
■
■proto void AddNoise(EntityAI source_entity, NoiseParams noise_params, float external_strenght_multipl
■proto void AddNoisePos(EntityAI source_entity, vector pos, NoiseParams noise_params, float external_s
■
■//! Will make a noise at that position which the AI will "see" for the duration of 'lifetime'
■proto void AddNoiseTarget(vector pos, float lifetime, NoiseParams noise_params, float external_strength
}
■
class NoiseParams
{■
■void NoiseParams()
■{
■■
■}
■
■proto native void Load(string noise_name);
■
■proto native void LoadFromPath(string noise_path);
}


// ---------------------------------------------
// ------ Noise.c - END ----------------------
// ---------------------------------------------
// ---------------------------------------------
// ------ NoteMenu.c - START --------------------
// ---------------------------------------------


class NoteMenu extends UIScriptedMenu
{
■protected MultilineEditBoxWidget m_edit;
■protected HtmlWidget m_html;
■protected ButtonWidget m_confirm_button;
■protected ItemBase m_Paper;
■protected EntityAI m_Pen;
■protected bool m_IsWriting;
■//protected int m_Handwriting;
■protected int m_SymbolCount;
■protected string m_PenColor; //color in hex-code format, transferred as string. Could be transferred as in

■void NoteMenu()
■{
■■MissionGameplay mission = MissionGameplay.Cast(GetGame().GetMission());
■■if (mission)
■■{
■■■IngameHud hud = IngameHud.Cast(mission.GetHud());
■■■if (hud)
■■■{
■■■■hud.ShowHudUI(false);
■■■}
■■}
■}

■void ~NoteMenu()
■{
■■MissionGameplay mission = MissionGameplay.Cast(GetGame().GetMission());
■■if (mission)
■■{
■■■IngameHud hud = IngameHud.Cast(mission.GetHud());
■■■if (hud)
■■■{
■■■■hud.ShowHudUI(true);
■■■}
■■}
■}

■override void InitNoteRead(string text = "")
■{
■■m_IsWriting = false;
■■
■■if (m_edit)
■■{
■■■m_edit.Show(false);
■■}
■■
■■if (m_html)
■■{
■■■//TODO add text formating here. Just text string for now
■■■if (text)
■■■{
■■■■m_html.Show(true);
■■■■m_html.SetText(text);
■■■■m_SymbolCount = text.Length(); //string.LengthUtf8() ?
■■■■//m_html.SetText("<html><body><p>" + text + "</p></body></html>");
■■■}
■■}
```

```
// ---------------------------------------------
// ------ NoticeSpacer.c - START --------------------
// ---------------------------------------------


// ---------------------------------------------------------
class NoticeSpacer : AutoHeightSpacer
{
■override bool OnEvent(EventType eventType, Widget target, int parameter0, int parameter1)
■{
■■Update();
■■
■■return super.OnEvent(eventType, target, parameter0, parameter1);
■}
};



// ---------------------------------------------
// ------ NoticeSpacer.c - END ----------------------
// ---------------------------------------------


// ---------------------------------------------
// ------ NotificationData.c - START --------------------
// ---------------------------------------------


class NotificationData
{
■string m_Icon;
■string m_TitleText;
■string m_DescriptionText;
■
■void NotificationData( string icon, string title_text, string desc_text = "" )
■{
■■m_Icon = icon;
■■m_TitleText = title_text;
■■if( desc_text != "" )
■■■m_DescriptionText = desc_text;
■}
}


// ---------------------------------------------
// ------ NotificationData.c - END ----------------------
// ---------------------------------------------


// ---------------------------------------------
// ------ NotificationSystem.c - START --------------------
// ---------------------------------------------


const static float NOTIFICATION_FADE_TIME = 3.0;

enum NotificationType
{
■FRIEND_CONNECTED,
■INVITE_FAIL_SAME_SERVER,
■JOIN_FAIL_GET_SESSION,
■CONNECT_FAIL_GENERIC,
■//Please add types before this item
```

```
// ----------------------------------------------
// ------ NotificationUI.c - START --------------------
// ----------------------------------------------


class NotificationUI
{
■protected ref Widget■m_Root;
■protected ref Widget■m_Spacer;
■protected ref Widget■m_VoiceContent;
■protected ref Widget■m_NotificationContent;
■
■protected ref map<NotificationRuntimeData, Widget>■m_Notifications;
■protected ref map<string, Widget>■■■■■m_VoiceNotifications;
■
■protected float■■■■■■■■■■■m_Width;
■protected float■■■■■■■■■■■m_CurrentHeight;
■protected float■■■■■■■■■■■m_TargetHeight;
■protected ref map<string, Widget>■■■■■m_WidgetTimers;
■
■void NotificationUI()
■{
■■m_Root■■■■■= GetGame().GetWorkspace().CreateWidgets("gui/layouts/new_ui/notifications/notifica
■■m_Spacer■■■■= m_Root.FindAnyWidget( "NotificationSpacer" );
■■m_VoiceContent■■■= m_Root.FindAnyWidget( "VoiceContent" );
■■m_NotificationContent■= m_Root.FindAnyWidget( "NotificationContent" );
■■m_Notifications■■■= new map<NotificationRuntimeData, Widget>;
■■m_VoiceNotifications■= new map<string, Widget>;
■■m_WidgetTimers■■■= new map<string, Widget>;
■■
■■NotificationSystem ntfSys = NotificationSystem.GetInstance();
■■if (ntfSys)
■■{
■■■ntfSys.m_OnNotificationAdded.Insert( AddNotification );
■■■ntfSys.m_OnNotificationRemoved.Insert( RemoveNotification );
■■}
■}
■
■void ~NotificationUI()
■{
■■NotificationSystem ntfSys = NotificationSystem.GetInstance();
■■if (ntfSys)
■■{
■■■ntfSys.m_OnNotificationAdded.Remove( AddNotification );
■■■ntfSys.m_OnNotificationRemoved.Remove( RemoveNotification );
■■}
■}
■
■void AddNotification( NotificationRuntimeData data )
■{
■■Widget notification■■■= GetGame().GetWorkspace().CreateWidgets("gui/layouts/new_ui/notifications/
■■
■■ImageWidget icon■■■= ImageWidget.Cast( notification.FindAnyWidget( "Image" ) );
■■RichTextWidget title■■= RichTextWidget.Cast( notification.FindAnyWidget( "Title" ) );
■■
■■if ( data.GetIcon() != "" )
■■■icon.LoadImageFile( 0, data.GetIcon() );
■■title.SetText( data.GetTitleText() );
■■
■■if ( data.GetDetailText() != "" )
■■{
■■■Widget bottom_spacer■= notification.FindAnyWidget( "BottomSpacer" );
■■■RichTextWidget detail■= RichTextWidget.Cast( notification.FindAnyWidget( "Detail" ) );
```

```
// ---------------------------------------------
// ------ NotifierBase.c - START --------------------
// ---------------------------------------------


class NotifierBase
{
■
■float■■■■m_DeltaT; // time in seconds since the last tick
■ref Timer ■■■m_Timer1; // timer which can be used for whatever
■PlayerBase■■■m_Player; //the player this Notifier belongs to
■int■■■■■m_Type;
■NotifiersManager■m_Manager;
■int ■■■■m_TendencyBufferSize = 3;//for best results, this should be somewhat aligned with modifier fre
■const int ■■■TENDENCY_BUFFER_SIZE = 30;//this needs to be bigger or same size as buffer size of a
■bool■■■■m_ShowTendency;
■bool■■■■m_Active;
■//private float■■m_SecsToSound; //internal counter
■//protected float■■m_MinPauseBetweenSounds; //minimal amount of seconds that needs to pass till sou
■//protected float■■m_MaxPauseBetweenSounds; //maximal amount of seconds that can pass till sound
■//private float■■m_SecsSinceLastAnimation; //internal counter
■//private float■■m_SecsToAnimation; //internal counter
■//protected float■■m_MinPauseBetweenAnimations; //minimal amount of seconds that needs to pass till
■//protected float■■m_MaxPauseBetweenAnimations; //maximal amount of seconds that can pass till ani
■int■■■■■m_TickInterval;
■int■■■■■m_TickIntervalLastTick;
■float■■■■m_TendencyBuffer[TENDENCY_BUFFER_SIZE];
■int■■■■■m_TendencyBufferWriteIterator;
■float ■■■■m_LastTendency;
■float■■■■m_LastMA;
■bool■■■■m_FirstPass = true;
■//int■■■■■m_TendencyID;
■
■PluginPlayerStatus ■m_ModulePlayerStatus;
■■
■void NotifierBase(NotifiersManager manager)
■{

■■//m_Timer1 = new Timer();
■■m_ModulePlayerStatus = PluginPlayerStatus.Cast(GetPlugin(PluginPlayerStatus));
■■m_Active = true;
■■m_Manager = manager;
■■m_Player = manager.GetPlayer();
■■m_TickInterval = 1000;
■■manager.RegisterItself(GetNotifierType(), this);
■}

■bool IsTimeToTick(int current_time)
■{
■■return ( current_time > m_TickIntervalLastTick + m_TickInterval );
■}
■
■
■VirtualHud GetVirtualHud()
■{
■■return m_Player.GetVirtualHud();
■}

■int GetNotifierType()
■{
■■return m_Type;
■}
```

```
// ---------------------------------------------
// ------ NotifiersManager.c - START --------------------
// ---------------------------------------------


enum eNotifiers
{
■NTF_HEALTHY,
■NTF_BLEEDISH,
■NTF_HUNGRY,
■NTF_THIRSTY,
■NTF_STUFFED,
■NTF_SICK,
■NTF_WETNESS,
■NTF_WARMTH,
■NTF_FEVERISH,
■NTF_BLOOD,
■NTF_LIVES,
■NTF_STAMINA,
■//NTF_AGENT_INFECTION,
■NTF_PILLS,
■NTF_HEARTBEAT,
■NTF_FRACTURE,
■//----------------------------
■NTF_COUNT,// !!! LAST ITEM !!!

}


class NotifiersManager
{
■static const int ■■■■MAX_COUNT = 64;
■ref array<ref NotifierBase> ■m_Notifiers;
■ref NotifierBase ■■■■m_NotifiersStatic[MAX_COUNT];//introduced as a seperate array to allow for fast
■PlayerBase■■■■■■m_Player;
■ref VirtualHud■■■■■m_VirtualHud;
■int■■■■■■■■m_MinTickTime;
■string ■■■■■■■m_System = "Notifiers";
■
■void NotifiersManager(PlayerBase player)
■{
■■m_Player = player;

■■m_Notifiers = new array<ref NotifierBase>;

■■m_MinTickTime = MIN_TICK_NOTIFIERS;
■■Init();
■}

■void Init()
■{
■■m_Notifiers.Insert(new HungerNotfr(this));
■■m_Notifiers.Insert(new ThirstNotfr(this));
■■m_Notifiers.Insert(new WarmthNotfr(this));
■■m_Notifiers.Insert(new WetnessNotfr(this));
■■m_Notifiers.Insert(new HealthNotfr(this));
■■m_Notifiers.Insert(new FeverNotfr(this));
■■m_Notifiers.Insert(new SickNotfr(this));
■■m_Notifiers.Insert(new StuffedNotfr(this));
■■m_Notifiers.Insert(new BloodNotfr(this));
■■m_Notifiers.Insert(new PillsNotfr(this));
■■m_Notifiers.Insert(new HeartbeatNotfr(this));
■■m_Notifiers.Insert(new FracturedLegNotfr(this)); //New addition
```

```
// ---------------------------------------------
// ------ NurseDress_ColorBase.c - START --------------------
// ---------------------------------------------


class NurseDress_ColorBase extends Clothing
{
■override void SetActions()
■{
■■super.SetActions();
■■AddAction(ActionWringClothes);
■}
};
class NurseDress_White extends NurseDress_ColorBase {};
class NurseDress_Blue extends NurseDress_ColorBase {};


// ---------------------------------------------
// ------ NurseDress_ColorBase.c - END ----------------------
// ---------------------------------------------


// ---------------------------------------------
// ------ NutritionalProfile.c - START --------------------
// ---------------------------------------------


class NutritionalProfile
{
■float m_Energy;
■float m_WaterContent;
■float m_NutritionalIndex;
■float m_FullnessIndex;
■float m_Toxicity;
■int m_LiquidType;
■int m_Agents;
■string m_LiquidClassname;
■float m_Digestibility;
■
■void NutritionalProfile(float energy, float water_content, float nutritional_index, float fullness_index, float t
■{
■■m_Energy = energy;
■■m_WaterContent = water_content;
■■m_NutritionalIndex = nutritional_index;
■■m_FullnessIndex = fullness_index;
■■m_Toxicity = toxicity;
■■m_Agents = agents;
■■m_Digestibility = digestibility;
■}
■
■void MarkAsLiquid(int liquid_type, string classname)
■{
■■m_LiquidType = liquid_type;
■■m_LiquidClassname = classname;
■}
■
■int GetAgents()
■{
■■return m_Agents;
■}
■
■int GetLiquidType()
■{
```

```
// ---------------------------------------------
// ------ NVGHeadstrap.c - START --------------------
// ---------------------------------------------


class NVGHeadstrap extends Clothing
{
■override bool CanPutAsAttachment(EntityAI parent)
■{
■■if (!super.CanPutAsAttachment(parent))
■■■return false;
■■
■■Clothing helmet = Clothing.Cast(parent.FindAttachmentBySlotName( "Headgear" ));
■■if (helmet && helmet.ConfigGetBool("noNVStrap"))
■■{
■■■return false;
■■}
■■
■■Clothing mask = Clothing.Cast(parent.FindAttachmentBySlotName("Mask"));
■■if (mask && mask.ConfigGetBool("noEyewear"))
■■{
■■■return false;
■■}
■■
■■return true;
■}
■
■override void SetActions()
■{
■■super.SetActions();
■■
■■AddAction(ActionToggleNVG);
■}
};


// ---------------------------------------------
// ------ NVGHeadstrap.c - END ----------------------
// ---------------------------------------------


// ---------------------------------------------
// ------ NVGoggles.c - START --------------------
// ---------------------------------------------


class NVGoggles extends PoweredOptic_Base
{
■bool ■■m_IsLowered;
■Clothing ■m_Strap;
■ref Timer ■m_WorkCheckTimer;
■
■void NVGoggles()
■{
■■RotateGoggles(true);
■■m_WorkCheckTimer = new Timer;
■}
■
■override void SetActions()
■{
■■super.SetActions();
■■
■■RemoveAction(ActionViewOptics);
```

```
// ----------------------------------------------
// ------ Object.c - START --------------------
// ----------------------------------------------


enum ProcessDirectDamageFlags
{
■NO_ATTACHMENT_TRANSFER, //!< Do not transfer damage to attachments
■NO_GLOBAL_TRANSFER, ■//!< Do not transfer damage to global
■NO_TRANSFER, ■■■//!< NO_ATTACHMENT_TRANSFER | NO_GLOBAL_TRANSFER
}

class Object extends IEntity
{
■private void ~Object();
■private void Object();
■
■bool CanBeSkinned()
■{
■■return false;
■}
■
■/**@brief Delete this object in next frame
■ * @return \p void
■ *
■ * @code
■ *■■■ItemBase item = GetGame().GetPlayer().CreateInInventory("GrenadeRGD5");
■ *■■■item.Delete();
■ * @endcode
■**/
■void Delete()
■{
■■GetGame().GetCallQueue(CALL_CATEGORY_SYSTEM).Call(GetGame().ObjectDelete, this);
■}
■
■proto native void AddProxyPhysics(string proxySelectionName);■
■
■proto native void RemoveProxyPhysics(string proxySelectionName);■
■
■//! Object entered trigger
■void OnEnterTrigger(ScriptedEntity trigger) {}
■
■//! Object left trigger
■void OnLeaveTrigger(ScriptedEntity trigger) {}

■//! Retrieve all LODS■
■proto native bool GetLODS(notnull out array<LOD> lods);
■
■//! Retrieve LOD name
■proto native owned string GetLODName(LOD lod);
■
■//! Retrieve LOD by given name
■LOD GetLODByName( string name )
■{
■■array<LOD> lods = new array<LOD>;
■■GetLODS( lods );
■■
■■for ( int i = 0; i < lods.Count(); ++i )
■■{
■■■string lod_name = GetLODName( lods.Get( i ) );
■■■lod_name.ToLower();
■■■name.ToLower();
■■■if ( lod_name == name )
```

```
// ---------------------------------------------
// ------ ObjectFollower.c - START --------------------
// ---------------------------------------------


class ObjectFollower extends ScriptedWidgetEventHandler
{
■reference int■■■■■■m_PivotYOffset;

■protected Widget■■■■■m_Root;
■protected Object■■■■■m_TargetObject;
■protected bool ■■■■■■m_Visible;

■void ObjectFollower()
■{
■■m_Visible = false;
■■//GetGame().GetUpdateQueue(CALL_CATEGORY_GUI).Insert(Update);
■}
■
■void ~ObjectFollower()
■{
■■//GetGame().GetUpdateQueue(CALL_CATEGORY_GUI).Remove(Update);
■}

■protected void OnWidgetScriptInit(Widget w)
■{
■■m_Root = w;
■■m_Root.SetHandler(this);
■}

■void SetTarget(Object obj)
■{
■■m_TargetObject = obj;
■}

■override bool OnUpdate(Widget w)
■{
■■if ( m_Root == w )
■■{
■■■Update();
■■■return true;
■■}
■■
■■return false;
■}
■
■void Update()
■{
■■float x, y;
■■
■■if(m_Visible)
■■{
■■■if(m_TargetObject)
■■■{
■■■■GetOnScreenPosition(x, y);
■■■}
■■■
■■■m_Root.Show(true);
■■}
■■else
■■{
■■■m_Root.Show(false);■■
■■}
```

```
// ---------------------------------------------
// ------ ObjectSpawner.c - START --------------------
// ---------------------------------------------


class ObjectSpawnerHandler
{
■//------------------------------------------------------------------------------------
■static void SpawnObjects()
■{
■■if( CfgGameplayHandler.GetObjectSpawnersArr() && CfgGameplayHandler.GetObjectSpawnersArr().C
■■{
■■■TStringArray arr = CfgGameplayHandler.GetObjectSpawnersArr();
■■■foreach(string spawner_json: arr)
■■■{
■■■■string path = "$mission:"+spawner_json;
■■■■
■■■■if ( FileExist( path ) )
■■■■{
■■■■■ObjectSpawnerJson spawner;
■■■■■JsonFileLoader<ObjectSpawnerJson>.JsonLoadFile( path, spawner );

■■■■■foreach (ITEM_SpawnerObject o: spawner.Objects)
■■■■■{
■■■■■■SpawnObject(o);
■■■■■}
■■■■}
■■■}
■■}
■}
■//------------------------------------------------------------------------------------
■static void SpawnObject(ITEM_SpawnerObject item)
■{
■■Object object = GetGame().CreateObjectEx(item.name, vector.ArrayToVec(item.pos), ECE_SETUP | E
■■if ( object )
■■{
■■■object.SetOrientation( vector.ArrayToVec(item.ypr));
■■■if (item.scale != 0)
■■■■object.SetScale(item.scale);
■■}
■}
■//------------------------------------------------------------------------------------
■static void OnGameplayDataHandlerLoad()
■{
■■SpawnObjects();
■}
■//------------------------------------------------------------------------------------
};

class ObjectSpawnerJson
{
■ref array<ref ITEM_SpawnerObject> Objects;
};

class ITEM_SpawnerObject
{
■string name;
■float pos[3];
■float ypr[3];
■float scale;
};
```

```
// ---------------------------------------------
// ------ ObjectTyped.c - START --------------------
// ---------------------------------------------


class ObjectTyped extends Object
{
■
};



// ---------------------------------------------
// ------ ObjectTyped.c - END ----------------------
// ---------------------------------------------


// ---------------------------------------------
// ------ ObsoleteFishingRod.c - START --------------------
// ---------------------------------------------


class ObsoleteFishingRod: FishingRod_Base_New {};



// ---------------------------------------------
// ------ ObsoleteFishingRod.c - END ----------------------
// ---------------------------------------------


// ---------------------------------------------
// ------ OfficerHat.c - START --------------------
// ---------------------------------------------


class OfficerHat extends Clothing
{
}



// ---------------------------------------------
// ------ OfficerHat.c - END ----------------------
// ---------------------------------------------


// ---------------------------------------------
// ------ OffroadHatchback.c - START --------------------
// ---------------------------------------------


class OffroadHatchback extends CarScript
{
■protected ref UniversalTemperatureSource m_UTSource;
■protected ref UniversalTemperatureSourceSettings m_UTSSettings;
■protected ref UniversalTemperatureSourceLambdaEngine m_UTSLEngine;

■void OffroadHatchback()
■{
■■//m_dmgContactCoef ■■= 0.075;
■
■■m_EngineStartOK ■■= "offroad_engine_start_SoundSet";
```

```
// ---------------------------------------------
// ------ OffroadHatchbackFrontLight.c - START --------------------
// ---------------------------------------------


class OffroadHatchbackFrontLight extends CarLightBase
{
■void OffroadHatchbackFrontLight()
■{
■■m_SegregatedBrightness = 5;
■■m_SegregatedRadius = 55;
■■m_SegregatedAngle = 110;
■■m_SegregatedColorRGB = Vector(1.0, 0.8, 0.6);
■■
■■m_AggregatedBrightness = 10;
■■m_AggregatedRadius = 75;
■■m_AggregatedAngle = 120;
■■m_AggregatedColorRGB = Vector(1.0, 0.8, 0.6);
■■
■■FadeIn(0.3);
■■SetFadeOutTime(0.25);
■■
■■SegregateLight();
■}
}


// ---------------------------------------------
// ------ OffroadHatchbackFrontLight.c - END ----------------------
// ---------------------------------------------


// ---------------------------------------------
// ------ OffroadHatchbackRearLight.c - START --------------------
// ---------------------------------------------


class OffroadHatchbackRearLight extends CarRearLightBase
{
■void OffroadHatchbackRearLight()
■{
■■// Brake light only
■■m_SegregatedBrakeBrightness = 1;
■■m_SegregatedBrakeRadius = 6;
■■m_SegregatedBrakeAngle = 180;
■■m_SegregatedBrakeColorRGB = Vector(1, 0.05, 0.05);
■■
■■// Reverse light only
■■m_SegregatedBrightness = 2;
■■m_SegregatedRadius = 13;
■■m_SegregatedAngle = 180;
■■m_SegregatedColorRGB = Vector(1.0, 1.0, 1.0);
■■
■■// Brake & Revese lights combined
■■m_AggregatedBrightness = 2.5;
■■m_AggregatedRadius = 15;
■■m_AggregatedAngle = 180;
■■m_AggregatedColorRGB = Vector(1.0, 0.5, 0.5);
■■
■■FadeIn(0.1);
■■SetFadeOutTime(0.1);
■■SetVisibleDuringDaylight(false);
■■SetCastShadow(false);
```

```
// ---------------------------------------------
// ------ Offroad_02.c - START --------------------
// ---------------------------------------------


class Offroad_02 extends CarScript
{
■protected ref UniversalTemperatureSource m_UTSource;
■protected ref UniversalTemperatureSourceSettings m_UTSSettings;
■protected ref UniversalTemperatureSourceLambdaEngine m_UTSLEngine;

■void Offroad_02()
■{
■■//m_dmgContactCoef■■= 0.0365;

■■m_EngineStartOK■■■= "Offroad_02_engine_start_SoundSet";
■■m_EngineStartBattery■= "Offroad_02_engine_failed_start_battery_SoundSet";
■■m_EngineStartPlug■■= "Offroad_02_engine_failed_start_sparkplugs_SoundSet";
■■m_EngineStartFuel■■= "Offroad_02_engine_failed_start_fuel_SoundSet";
■■m_EngineStopFuel■■= "offroad_engine_stop_fuel_SoundSet";

■■m_CarDoorOpenSound■■= "offroad_02_door_open_SoundSet";
■■m_CarDoorCloseSound■■= "offroad_02_door_close_SoundSet";
■■
■■m_CarHornShortSoundName = "Offroad_02_Horn_Short_SoundSet";
■■m_CarHornLongSoundName■= "Offroad_02_Horn_SoundSet";
■■
■■SetEnginePos("0 0.7 1.7");
■}
■
■override void EEInit()
■{■■
■■super.EEInit();
■■
■■if (GetGame().IsServer() || !GetGame().IsMultiplayer())
■■{
 ■■■m_UTSSettings ■■■■■= new UniversalTemperatureSourceSettings();
■■■m_UTSSettings.m_ManualUpdate ■= true;
■■■m_UTSSettings.m_TemperatureMin■= 0;
■■■m_UTSSettings.m_TemperatureMax■= 30;
■■■m_UTSSettings.m_RangeFull■■= 0.5;
■■■m_UTSSettings.m_RangeMax■■= 2;
■■■m_UTSSettings.m_TemperatureCap■= 25;
■■■
■■■m_UTSLEngine■■■■■= new UniversalTemperatureSourceLambdaEngine();
■■■m_UTSource■■■■■■= new UniversalTemperatureSource(this, m_UTSSettings, m_UTSLEngine);
■■}■■
■}
■
■override void OnEngineStart()
■{
■■super.OnEngineStart();

■■if (GetGame().IsServer() || !GetGame().IsMultiplayer())
■■{
■■■m_UTSource.SetDefferedActive(true, 20.0);
■■}
■}
■
■override void OnEngineStop()
■{
■■super.OnEngineStop();
```

```
// ---------------------------------------------
// ------ Offroad_02FrontLight.c - START --------------------
// ---------------------------------------------


class Offroad_02FrontLight extends CarLightBase
{
■void Offroad_02FrontLight()
■{
■■m_SegregatedBrightness = 5;
■■m_SegregatedRadius = 75;
■■m_SegregatedAngle = 90;
■■m_SegregatedColorRGB = Vector(0.85, 0.85, 0.58);
■■
■■m_AggregatedBrightness = 10;
■■m_AggregatedRadius = 100;
■■m_AggregatedAngle = 120;
■■m_AggregatedColorRGB = Vector(0.85, 0.85, 0.58);
■■
■■FadeIn(0.1);
■■SetFadeOutTime(0.05);
■■
■■SegregateLight();
■}
};


// ---------------------------------------------
// ------ Offroad_02FrontLight.c - END ----------------------
// ---------------------------------------------


// ---------------------------------------------
// ------ Offroad_02RearLight.c - START --------------------
// ---------------------------------------------


class Offroad_02RearLight extends CarRearLightBase
{
■void Offroad_02RearLight()
■{
■■// Brake light only
■■m_SegregatedBrakeBrightness = 6;
■■m_SegregatedBrakeRadius = 6;
■■m_SegregatedBrakeAngle = 220;
■■m_SegregatedBrakeColorRGB = Vector(0.95, 0.15, 0.13);
■■
■■// Reverse light only
■■m_SegregatedBrightness = 2;
■■m_SegregatedRadius = 15;
■■m_SegregatedAngle = 160;
■■m_SegregatedColorRGB = Vector(0.64, 0.60, 0.62);
■■
■■// Brake & Revese lights combined
■■m_AggregatedBrightness = 7;
■■m_AggregatedRadius = 18;
■■m_AggregatedAngle = 170;
■■m_AggregatedColorRGB = Vector(1.0, 0.4, 0.5);
■■
■■FadeIn(0.1);
■■SetFadeOutTime(0.05);

■■SetVisibleDuringDaylight(false);
```

```
// ---------------------------------------------
// ------ OFSMBase.c - START --------------------
// ---------------------------------------------


/**@class■■OFSMBase
 * @brief■■base class for Orthogonal Finite State Machine
 *
 * stores current states (m_states) and transition table with possible transitions from each state
 * to another state via event
 **/
class OFSMBase<Class FSMStateBase, Class FSMEventBase, Class FSMActionBase, Class FSMGuard
{
■protected ref array<ref FSMStateBase> m_States; /// current fsm state
■protected ref array<ref FSMStateBase> m_InitialStates; /// configurable initial state of the machine
■protected ref array<ref FSMTransition<FSMStateBase, FSMEventBase, FSMActionBase, FSMGuardBas

■void OFSMBase ()
■{
■■m_States = new array<ref FSMStateBase>;
■■m_InitialStates = new array<ref FSMStateBase>;
■■m_Transitions = new array<ref FSMTransition<FSMStateBase, FSMEventBase, FSMActionBase, FSM
■}

■/**@fn■■■GetCurrentStates
■ * @brief■■returns currently active state
■ * @return■■current state the FSM is in (or NULL)
■ **/
■array<ref FSMStateBase> GetCurrentState ()
■{
■■return m_States;
■}

■/**@fn■■■SetInitialState
■ * @brief■■sets the initial_state for starting the machine
■ **/
■void SetInitialStates (array<ref FSMStateBase> initial_states)
■{
■■m_InitialStates = initial_states;

■■for (int s = 0; s < initial_states.Count(); ++s)
■■■m_States.Insert(initial_states[s]);
■}

■/**@fn■■■Start
■ * @brief■■starts the state machine by entering the initial_state (using intial_event as argument to initial
■ * @param[in]■e \p■optional event for starting the machind
■ **/
■void Start (array<ref FSMEventBase> initial_events = null)
■{
■■fsmbDebugPrint("[ofsm] " + this.ToString() + "::Start(" + initial_events.ToString() + "), init_state=" + m_I

■■for (int s = 0; s < m_States.Count(); ++s)
■■{
■■■m_States[s] = m_InitialStates[s];

■■■if (initial_events)
■■■■m_States[s].OnEntry(initial_events[s]);
■■■else
■■■■m_States[s].OnEntry(null);
■■}
■}
```

```
// ---------------------------------------------
// ------ OMNOGloves_ColorBase.c - START --------------------
// ---------------------------------------------


class OMNOGloves_ColorBase extends Clothing {};
class OMNOGloves_Gray extends OMNOGloves_ColorBase {};
class OMNOGloves_Brown extends OMNOGloves_ColorBase {};


// ---------------------------------------------
// ------ OMNOGloves_ColorBase.c - END ----------------------
// ---------------------------------------------


// ---------------------------------------------
// ------ OnlineServices.c - START --------------------
// ---------------------------------------------


class OnlineServices
{
■static ref ScriptInvoker■■■■■■■■■■■■m_FriendsAsyncInvoker■■= new ScriptInvoker();
■static ref ScriptInvoker■■■■■■■■■■■■m_PermissionsAsyncInvoker■= new ScriptInvoker();
■static ref ScriptInvoker■■■■■■■■■■■■m_ServersAsyncInvoker■■= new ScriptInvoker();
■static ref ScriptInvoker■■■■■■■■■■■■m_ServerAsyncInvoker■■= new ScriptInvoker();
■static ref ScriptInvoker■■■■■■■■■■■■m_MuteUpdateAsyncInvoker■= new ScriptInvoker(); // DEPR
■static ref ScriptInvoker■■■■■■■■■■■■m_ServerModLoadAsyncInvoker■= new ScriptInvoker();
■
■static ref BiosClientServices■■■■■■■■■■■m_ClientServices;
■static ref TrialService■■■■■■■■■■■■■m_TrialService;
■
■protected static string■■■■■■■■■■■■■m_InviteServerIP;
■protected static int■■■■■■■■■■■■■m_InviteServerPort;
■protected static string■■■■■■■■■■■■■m_CurrentServerIP;
■protected static int■■■■■■■■■■■■■m_CurrentServerPort;
■protected static ref GetServersResultRow■■■■■■■■m_CurrentServerInfo;
■
■
■protected static ref map<string, ref BiosFriendInfo>■■■■■m_FriendsList;
■protected static ref map<string, bool>■■■■■■■■■m_MuteList;
■protected static ref map<string, ref BiosPrivacyPermissionResultArray>■m_PermissionsList;
■
■protected static bool■■■■■■■■■■■■■m_FirstFriendsLoad■■■= true;
■protected static bool■■■■■■■■■■■■■m_MultiplayState■■■= false;
■protected static ref array<string>■■■■■■■■■■m_PendingInvites;

■protected static ref BiosUser■■■■■■■■■■■m_BiosUser;
■
■static void Init()
■{
■■#ifdef PLATFORM_CONSOLE
■■■#ifndef PLATFORM_WINDOWS // if app is not on Windows with -XBOX parameter
■■■if ( !m_TrialService )
■■■■m_TrialService = new TrialService;
■■■if ( !m_FriendsList )
■■■■m_FriendsList = new map<string, ref BiosFriendInfo>;
■■■if ( !m_MuteList )
■■■■m_MuteList = new map<string, bool>;
■■■if ( !m_PermissionsList )
■■■■m_PermissionsList = new map<string, ref BiosPrivacyPermissionResultArray>;
■■■
■■■m_FriendsList.Clear();
```

```
// ---------------------------------------------
// ------ OpenableBehaviour.c - START --------------------
// ---------------------------------------------


class OpenableBehaviour
{
■protected bool m_IsOpened;
■
■void OpenableBehaviour(bool bState = true)
■{
■■m_IsOpened = bState; //! initial value
■}

■void Open()
■{
■■SetState(true);
■}

■void Close()
■{
■■SetState(false);
■}
■
■bool IsOpened()
■{
■■return GetState();
■}
■
■protected void SetState(bool bState)
■{
■■m_IsOpened = bState;
■}
■
■protected bool GetState()
■{
■■return m_IsOpened;
■}
}


// ---------------------------------------------
// ------ OpenableBehaviour.c - END ----------------------
// ---------------------------------------------


// ---------------------------------------------
// ------ OpenCan.c - START --------------------
// ---------------------------------------------


class OpenCan extends RecipeBase
{■
■override void Init()
■{
■■m_Name = "#STR_OpenCan0";
■■m_IsInstaRecipe = false;//should this recipe be performed instantly without animation
■■m_AnimationLength = 1;//animation length in relative time units
■■m_Specialty = 0.01;// value > 0 for roughness, value < 0 for precision
■■
■■m_AnywhereInInventory = false;//is this recipe valid even when neither of the items is in hands
■■//conditions
■■m_MinDamageIngredient[0] = -1;//-1 = disable check
```

```
// ---------------------------------------------
// ------ OpenDir.c - START --------------------
// ---------------------------------------------


[WorkbenchPluginAttribute("Open Dir", "Just for testing", "ctrl+-", "", {"ScriptEditor"})]
class OpenDirPlugin: WorkbenchPlugin
{
■override void Run()
■{
■■ScriptEditor mod = Workbench.GetModule("ScriptEditor");
■■if (mod)
■■{
■■■string file;
■■■string absPath;
■■■if (mod.GetCurrentFile(file) && Workbench.GetAbsolutePath(file, absPath))
■■■{
■■■■if (absPath.Length() < 2) return;
■■■■
■■■■absPath.Replace("\\", "/");
■■■■
■■■■if (absPath[1] != ":")
■■■■{
■■■■■string cwd;
■■■■■Workbench.GetCwd(cwd);
■■■■■absPath = cwd + "/" + absPath;
■■■■}
■■■■
■■■■int index = absPath.IndexOf("/");
■■■■int last_index = index;
■■■■
■■■■while(index != -1)
■■■■{
■■■■■last_index = index;
■■■■■index = absPath.IndexOfFrom(last_index + 1, "/");
■■■■}
■■■■
■■■■if (last_index == -1) return;
■■■■
■■■■string path = absPath.Substring(0, last_index);
■■■■string command;
■■■■command.Replace("$path", path);
■■■■//Print(path);
■■■■//Workbench.RunCmd(command);
■■■■Workbench.RunCmd("cmd /c \"start " + path +"\"");
■■■}
■■}
■}
■
■override void Configure()
■{
■■Workbench.ScriptDialog("Configure OpenDir", "Usage: \n$path - will be replaced with file name", this);
■}
■
■[ButtonAttribute("OK")]
■void OkButton() {}
}




// ---------------------------------------------
// ------ OpenDir.c - END ----------------------
// ---------------------------------------------
// ------ OpenItem.c - START --------------------
// ---------------------------------------------


class OpenItem
{
■//! WIll open the 'item_target' by spawning a new entity and transferring item variables to the new one
■static void OpenAndSwitch(ItemBase item_tool, ItemBase item_target, PlayerBase player, float specialty
■{
■■array<int> spill_range = new array<int>;
■■
■■if( item_tool.ConfigIsExisting("OpenItemSpillRange") )
■■{
■■■item_tool.ConfigGetIntArray("OpenItemSpillRange", spill_range );
■■}
■■else
■■{
■■■Debug.LogError("OpenItemSpillRange config parameter missing, default values used ! ");
■■■Error("OpenItemSpillRange config parameter missing, default values used !");
■■■spill_range.Insert(0);
■■■spill_range.Insert(100);
■■}
■■float spill_modificator = Math.RandomIntInclusive( spill_range.Get(0),spill_range.Get(1) ) / 100;
■■
■■OpenItem.SwitchItems(item_target, player, spill_modificator, specialty_weight);
■}

■//! Will switch the 'item' for a new game entity, the new entity's classname will be formed by adding the 's
■static void SwitchItems (EntityAI old_item, PlayerBase player, float spill_modificator, float specialty_weig
■{
■■string old_name = old_item.GetType();
■■string new_name = old_name + "_Opened";
■■OpenAndSwitchLambda l = new OpenAndSwitchLambda(old_item, new_name, player, spill_modificato
■■l.SetTransferParams(true, true, true, true);
■■MiscGameplayFunctions.TurnItemIntoItemEx(player, l);
■}
};


class OpenAndSwitchLambda : TurnItemIntoItemLambda
{
■float m_SpillModifier;
■float m_SpecialtyWeight;
■void OpenAndSwitchLambda (EntityAI old_item, string new_item_type, PlayerBase player, float spill_mo

■override void CopyOldPropertiesToNew (notnull EntityAI old_item, EntityAI new_item)
■{
■■super.CopyOldPropertiesToNew(old_item, new_item);
■}
■
■override void OnSuccess (EntityAI new_item)
■{
■■super.OnSuccess(new_item);

■■ItemBase ib = ItemBase.Cast(new_item);
■■if ( new_item )
■■{■
■■■float quantity_old = ib.GetQuantity();
■■■float spill_amount = quantity_old * m_SpillModifier;
■■■spill_amount = m_Player.GetSoftSkillsManager().SubtractSpecialtyBonus(spill_amount, m_Specialty
■■■float quantity_new = quantity_old - spill_amount;
■
```

```
// ----------------------------------------------
// ------ Optics.c - START --------------------
// ----------------------------------------------


class M68Optic : ItemOptics {};
class M4_T3NRDSOptic : ItemOptics {};
class FNP45_MRDSOptic : ItemOptics {};
class Crossbow_RedpointOptic : ItemOptics {};
class ReflexOptic : ItemOptics {};
class ACOGOptic : ItemOptics {};
class KashtanOptic : ItemOptics {};
class LongrangeOptic : ItemOptics {};
class PistolOptic : ItemOptics {};
class GrozaOptic : ItemOptics {};
class KobraOptic : ItemOptics {};
class ACOGOptic_6x : ItemOptics {};

class KazuarOptic: ItemOptics
{
■override void InitOpticMode() //TODO - decide whether to randomize on spawn and how to determine it (
■{
■■super.InitOpticMode();
■■
■■SetCurrentOpticMode(GameConstants.OPTICS_STATE_NIGHTVISION);
■}
■
■override int GetCurrentNVType()
■{
■■if (IsWorking())
■■{
■■■switch (m_CurrentOpticMode)
■■■{
■■■■case GameConstants.OPTICS_STATE_DAY:
■■■■■return NVTypes.NV_OPTICS_KAZUAR_DAY;
■■■■
■■■■case GameConstants.OPTICS_STATE_NIGHTVISION:
■■■■■return NVTypes.NV_OPTICS_KAZUAR_NIGHT;
■■■}
■■■Error("Undefined optic mode of " + this);
■■■return NVTypes.NONE;
■■}
■■else
■■{
■■■return NVTypes.NV_OPTICS_OFF;
■■}
■}
■
■override void OnOpticModeChange()
■{
■■super.OnOpticModeChange();
■■
■■UpdateSelectionVisibility();
■}
■
■override void OnOpticEnter()
■{
■■super.OnOpticEnter();
■■
■■HideSelection("hide_cover_pilot");
■}
■
■override void UpdateSelectionVisibility()
```

```
// ---------------------------------------------
// ------ OptionSelector.c - START --------------------
// ---------------------------------------------


class OptionSelector extends OptionSelectorBase
{
■protected Widget■■■■■■m_PreviousOption;
■protected Widget■■■■■■m_NextOption;
■protected TextWidget■■■■■m_SelectedOption;
■protected int■■■■■■■m_SelectedOptionIndex;
■protected ref array<string>■■■■m_Options;
■
■void OptionSelector(Widget parent, int current_index, ScriptedWidgetEventHandler parent_c, bool disabl
■{
■■m_Options■■■■= { "#server_browser_disabled", "#server_browser_show", "#server_browser_hide" };
■■m_ParentClass■■■= parent_c;
■■m_SelectorType■■■= 2;
■■if (current_index < 0 || current_index >= m_Options.Count())
■■{
■■■m_SelectedOptionIndex = 0;
■■}
■■else
■■{
■■■m_SelectedOptionIndex = current_index;
■■}
■■
■■m_Root■■■■■■= GetGame().GetWorkspace().CreateWidgets("gui/layouts/new_ui/option_selector.la
■■#ifdef PLATFORM_CONSOLE
■■■m_Parent■■■■= parent.GetParent().GetParent();
■■#else
■■#ifdef PLATFORM_WINDOWS
■■■m_Parent■■■■= parent.GetParent();
■■#endif
■■#endif
■■
■■m_SelectedOption■■■= TextWidget.Cast(m_Root.FindAnyWidget("option_label"));
■■m_PreviousOption■■■= m_Root.FindAnyWidget("prev_option");
■■m_NextOption■■■■= m_Root.FindAnyWidget("next_option");
■■
■■#ifdef PLATFORM_CONSOLE
■■■m_NextOption.Show(false);
■■■m_PreviousOption.Show(false);
■■#endif
■■
■■m_SelectedOption.SetText(m_Options.Get(m_SelectedOptionIndex));
■■
■■m_Enabled = !disabled;
■■if (m_Enabled)
■■{
■■■Enable();
■■}
■■else
■■{
■■■Disable();
■■}
■■
■■m_Parent.SetHandler(this);
■}
■
■void ~OptionSelector()
■{
■■delete m_Root;
```

```
// ---------------------------------------------
// ------ OptionSelectorBase.c - START --------------------
// ---------------------------------------------


class OptionSelectorBase extends ScriptedWidgetEventHandler
{
	protected int							m_SelectorType = 0;
	protected Widget						m_Parent;
	protected Widget						m_Root;
	
	protected bool							m_Enabled;
	
	protected ScriptedWidgetEventHandler	m_ParentClass;
	
	ref ScriptInvoker						m_OptionFocused = new ScriptInvoker;
	ref ScriptInvoker						m_OptionUnfocused = new ScriptInvoker;
	ref ScriptInvoker						m_AttemptOptionChange = new ScriptInvoker;
	ref ScriptInvoker						m_OptionChanged = new ScriptInvoker;
	
	void ~OptionSelectorBase()
	{
		delete m_Root;
	}
	
	Widget GetParent()
	{
		return m_Parent;
	}
	
	bool IsFocusable(Widget w)
	{
		if (w)
		{
			return w == m_Parent;
		}
		return false;
	}
	
	override bool OnMouseEnter(Widget w, int x, int y)
	{
		if (!IsFocusable(w))
			return true;
		
		if (m_ParentClass)
		{
			m_ParentClass.OnFocus(m_Root.GetParent(), -1, m_SelectorType);
			#ifndef PLATFORM_CONSOLE
			m_ParentClass.OnMouseEnter(m_Root.GetParent().GetParent(), x, y);
			#endif
		}
		
		UIScriptedMenu menu = GetGame().GetUIManager().GetMenu();
		
		if (menu && menu.IsInherited(CharacterCreationMenu))
		{
			menu.OnFocus(m_Root.GetParent(), -1, m_SelectorType);
			menu.OnMouseEnter(m_Root.GetParent().GetParent(), x, y);
		}
		
		#ifndef PLATFORM_CONSOLE
		ColorHighlight(w);
		#else
```

```
// ---------------------------------------------
// ------ OptionSelectorEditbox.c - START --------------------
// ---------------------------------------------


class OptionSelectorEditbox extends OptionSelectorBase
{
■protected EditBoxWidget■■■m_EditBox;
■
■void OptionSelectorEditbox(Widget parent, string value, ScriptedWidgetEventHandler parent_menu, boo
■{
■■m_Root■■■■■= GetGame().GetWorkspace().CreateWidgets("gui/layouts/new_ui/option_editbox.layo
■■#ifdef PLATFORM_CONSOLE
■■■m_Parent■■■■= parent.GetParent().GetParent();
■■#else
■■#ifdef PLATFORM_WINDOWS
■■■m_Parent■■■■= parent.GetParent();
■■#endif
■■#endif
■■
■■m_SelectorType■■■■= 1;
■■m_ParentClass■■■■= parent_menu;
■■m_EditBox■■■■■= EditBoxWidget.Cast(m_Root.FindAnyWidget("option_value"));
■■
■■SetValue(value);
■■Enable();
■■
■■m_Parent.SetHandler(this);
■}
■
■void ~OptionSelectorEditbox()
■{
■■delete m_Root;
■}
■
■override void Enable()
■{
■■super.Enable();
■■
■■m_EditBox.ClearFlags(WidgetFlags.IGNOREPOINTER);
■}
■
■override void Disable()
■{
■■super.Disable();
■■
■■m_EditBox.SetFlags(WidgetFlags.IGNOREPOINTER);
■}
■
■override bool OnMouseEnter(Widget w, int x, int y)
■{
■■if (!IsFocusable(w))
■■■return true;
■■
■■if (m_ParentClass)
■■{
■■■m_ParentClass.OnFocus(m_Root.GetParent(), -1, m_SelectorType);
■■■m_ParentClass.OnMouseEnter(m_Root.GetParent().GetParent(), x, y);
■■}
■■
■■UIScriptedMenu menu = GetGame().GetUIManager().GetMenu();
■■
■■if (menu && menu.IsInherited(CharacterCreationMenu))
```

```
// ---------------------------------------------
// ------ OptionSelectorLevelMarker.c - START --------------------
// ---------------------------------------------


/**
\brief This Option Selector handles a Slider Marker, which basically has 2 sliders
■   One slider is for selecting the value, and the other slider is for displaying some other value/ information
@param m_Slider2 other slider value, which can be used to convery some other info to the player
*/
class OptionSelectorLevelMarker extends OptionSelectorSliderSetup
{
■protected SliderWidget■■m_Slider2;
■
■void OptionSelectorLevelMarker(Widget parent, float value, ScriptedWidgetEventHandler parentMenu, b
■{
■■m_Root■■■■■= GetGame().GetWorkspace().CreateWidgets("gui/layouts/new_ui/option_level_marke
■■#ifdef PLATFORM_CONSOLE
■■■m_Parent■■■■= parent.GetParent().GetParent();
■■#else
■■#ifdef PLATFORM_WINDOWS
■■■m_Parent■■■■= parent.GetParent();
■■#endif
■■#endif
■■
■■m_SelectorType■■■= 1;
■■m_ParentClass■■■= parentMenu;
■■m_Slider■■■■= SliderWidget.Cast(m_Root.FindAnyWidget("marker_value"));
■■m_Slider2■■■■= SliderWidget.Cast(m_Root.FindAnyWidget("other_slider_value"));
■■
■■m_MinValue■■■■■= min;
■■m_MaxValue■■■■■= max;
■■
■■SetValue(value, false);
■■SetSlider2Value(0);
■■Enable();

■■m_Parent.SetHandler(this);
■}
■
■override void Enable()
■{
■■super.Enable();
■■m_Slider2.ClearFlags(WidgetFlags.IGNOREPOINTER);
■}
■
■override void Disable()
■{
■■super.Disable();
■■m_Slider2.SetFlags(WidgetFlags.IGNOREPOINTER);
■}
■
■void SetSlider2Value(float value)
■{
■■m_Slider2.SetCurrent(NormalizeInput(value));
■}
■
■override bool OnUpdate(Widget w)
■{
■■return false;
■}
}
```

```
// ---------------------------------------------
// ------ OptionSelectorMultistate.c - START --------------------
// ---------------------------------------------


class OptionSelectorMultistate extends OptionSelector
{
■protected bool ■m_CanSwitch;
■
■void OptionSelectorMultistate(Widget parent, int current_index, ScriptedWidgetEventHandler parent_c, b
■{
■■m_CanSwitch = true;
■■
■■m_SelectorType■■■■= 2;
■■m_Options■■■■■= options;
■■if (options.Count() == 0)
■■{
■■■Error("Invalid OptionSelectorMultistate options");
■■}
■■
■■if (current_index < 0 || current_index >= m_Options.Count())
■■{
■■■m_SelectedOptionIndex = 0;
■■}
■■else
■■{
■■■m_SelectedOptionIndex = current_index;
■■}
■■
■■m_SelectedOption.SetText(m_Options.Get(m_SelectedOptionIndex));
■}
■
■void LoadNewValues(notnull array<string> options, int current_index)
■{
■■m_Options = options;
■■SetValue(current_index);
■}
■
■override void SetNextOption()
■{
■■int idx = m_SelectedOptionIndex;
■■idx++;
■■if (idx >= m_Options.Count())
■■{
■■■idx = 0;
■■}
■■
■■m_AttemptOptionChange.Invoke(idx);
■■if (m_CanSwitch)
■■{
■■■PerformSetOption(idx);
■■}
■}
■
■override void SetPrevOption()
■{
■■int idx = m_SelectedOptionIndex;
■■idx--;
■■if (idx < 0)
■■{
■■■idx = m_Options.Count() - 1;
■■}
■■
```

```
// ---------------------------------------------
// ------ OptionSelectorSlider.c - START --------------------
// ---------------------------------------------


class OptionSelectorSlider extends OptionSelectorSliderSetup
{
■void OptionSelectorSlider(Widget parent, float value, ScriptedWidgetEventHandler parent_menu, bool di
■{
■■m_Root■■■■■= GetGame().GetWorkspace().CreateWidgets("gui/layouts/new_ui/option_slider.layout
■■#ifdef PLATFORM_CONSOLE
■■■m_Parent■■■■= parent.GetParent().GetParent();
■■#else
■■#ifdef PLATFORM_WINDOWS
■■■m_Parent■■■■= parent.GetParent();
■■#endif
■■#endif
■■
■■m_SelectorType■■■■= 1;
■■m_ParentClass■■■■= parent_menu;
■■m_Slider■■■■■= SliderWidget.Cast(m_Root.FindAnyWidget("option_value"));
■■m_Slider.SetCurrent(value);
■■
■■m_MinValue■■■■■= min;
■■m_MaxValue■■■■■= max;
■■
■■SetValue(value);
■■Enable();
■■
■■m_Parent.SetHandler(this);
■}
}


// ---------------------------------------------
// ------ OptionSelectorSlider.c - END ----------------------
// ---------------------------------------------


// ---------------------------------------------
// ------ OptionSelectorSliderSetup.c - START --------------------
// ---------------------------------------------


class OptionSelectorSliderSetup extends OptionSelectorBase
{
■protected SliderWidget■■■m_Slider;
■protected float■■■■■m_MinValue;
■protected float■■■■■m_MaxValue;
■
■void ~OptionSelectorSliderSetup()
■{
■■delete m_Root;
■}
■
■override void Enable()
■{
■■super.Enable();
■■
■■m_Slider.ClearFlags(WidgetFlags.IGNOREPOINTER);
■}
■
■override void Disable()
```

```
// ---------------------------------------------
// ------ OptionsMenu.c - START --------------------
// ---------------------------------------------


class OptionsMenu extends UIScriptedMenu
{
■const int MODAL_ID_DEFAULT = 100;
■const int DIALOG_TAB_OFFSET = 1400;

■protected TabberUI■■■■■m_Tabber;
■protected ref OptionsMenuGame■■m_GameTab;
■protected ref OptionsMenuSounds■■m_SoundsTab;
■protected ref OptionsMenuVideo■■m_VideoTab;
■protected ref OptionsMenuControls■m_ControlsTab;
■
■protected ref GameOptions■■■m_Options;
■
■protected ButtonWidget■■■■m_Apply;
■protected ButtonWidget■■■■m_Back;
■protected ButtonWidget■■■■m_Reset; //undo
■protected ButtonWidget■■■■m_Defaults; //defaults
■
■protected Widget■■■■■m_Details;
■protected TextWidget■■■■m_Version;
■
■protected int ■■■■■■m_ActiveTabIdx = 0;
■protected bool ■■■■■■m_ModalLock;
■protected bool ■■■■■■m_CanApplyOrReset;
■protected bool ■■■■■■m_CanToggle;
■
■void OptionsMenu()
■{
■■
■}
■
■override Widget Init()
■{
■■m_Options■■= new GameOptions();
■■
■■#ifdef PLATFORM_XBOX
■■layoutRoot■■= GetGame().GetWorkspace().CreateWidgets("gui/layouts/new_ui/options/xbox/options_
■■#else
■■#ifdef PLATFORM_PS4
■■layoutRoot■■= GetGame().GetWorkspace().CreateWidgets("gui/layouts/new_ui/options/ps/options_m
■■#else
■■#ifdef PLATFORM_WINDOWS
■■layoutRoot■■= GetGame().GetWorkspace().CreateWidgets("gui/layouts/new_ui/options/pc/options_m
■■#endif
■■#endif
■■#endif
■■
■■layoutRoot.FindAnyWidget("Tabber").GetScript(m_Tabber);
■■
■■m_Details■■= layoutRoot.FindAnyWidget("settings_details");
■■m_Version■■= TextWidget.Cast(layoutRoot.FindAnyWidget("version"));
■■
■■m_GameTab■■= new OptionsMenuGame(layoutRoot.FindAnyWidget("Tab_0"), m_Details, m_Option
■■m_SoundsTab■■= new OptionsMenuSounds(layoutRoot.FindAnyWidget("Tab_1"), m_Details, m_Opt
■■
■■#ifdef PLATFORM_XBOX
■■m_ControlsTab■= new OptionsMenuControls(layoutRoot.FindAnyWidget("Tab_2"), m_Details, m_Opti
■■#else
```

```
// ---------------------------------------------
// ------ OptionsMenuControls.c - START --------------------
// ---------------------------------------------


class OptionsMenuControls extends ScriptedWidgetEventHandler
{
■protected Widget■■■■■■m_Root;
■
■protected Widget■■■■■■m_SettingsRoot;
■protected Widget■■■■■■m_DetailsRoot;
■protected Widget■■■■■■m_DetailsBodyDefault;
■protected Widget■■■■■■m_DetailsBodyConnectivity;
#ifdef PLATFORM_CONSOLE
■protected bool ■■■■■■■m_MaKOptionAvailable;
■protected Widget■■■■■■m_ConsoleControllerSensitivityWidget;
■protected Widget■■■■■■m_ConsoleMouseSensitivityWidget;
#endif
■protected TextWidget■■■■■m_DetailsLabel;
■protected RichTextWidget■■■■m_DetailsText;
■protected GridSpacerWidget■■■■m_Keybindings;
■
■protected GameOptions■■■■■m_Options;
■protected OptionsMenu■■■■■m_Menu;
■
■// console options accessors
■protected ref SwitchOptionsAccess■■m_KeyboardOption;
■protected ref SwitchOptionsAccess■■m_AimHelperOption;
■
■// console options selectors
■protected ref OptionSelectorMultistate■m_KeyboardSelector;
■protected ref OptionSelectorMultistate■m_AimHelperSelector;

■// mouse accessors
■protected ref SwitchOptionsAccess■■m_Mouse_InvertOption;
■protected ref NumericOptionsAccess■■m_Mouse_VSensitivityOption;
■protected ref NumericOptionsAccess■■m_Mouse_HSensitivityOption;
■protected ref NumericOptionsAccess■■m_Mouse_AimMod_VSensitivityOption;
■protected ref NumericOptionsAccess■■m_Mouse_AimMod_HSensitivityOption;
■
■// mouse selectors
■protected ref OptionSelectorMultistate■m_Mouse_InvertSelector;
■protected ref OptionSelectorSlider■■m_Mouse_VSensitivitySelector;
■protected ref OptionSelectorSlider■■m_Mouse_HSensitivitySelector;
■protected ref OptionSelectorSlider■■m_Mouse_AimMod_VSensitivitySelector;
■protected ref OptionSelectorSlider■■m_Mouse_AimMod_HSensitivitySelector;

■// gamepad/controller accessors
■protected ref NumericOptionsAccess■■m_ControllerLS_VSensitivityOption;
■protected ref NumericOptionsAccess■■m_ControllerLS_HSensitivityOption;
■protected ref NumericOptionsAccess■■m_ControllerLS_VehicleMod_HSensitivityOption;
■protected ref SwitchOptionsAccess■■m_ControllerRS_InvertOption;
■protected ref NumericOptionsAccess■■m_ControllerRS_VSensitivityOption;
■protected ref NumericOptionsAccess■■m_ControllerRS_HSensitivityOption;
■protected ref NumericOptionsAccess■■m_ControllerRS_CurvatureOption;
■protected ref NumericOptionsAccess■■m_ControllerRS_AimMod_VSensitivityOption;
■protected ref NumericOptionsAccess■■m_ControllerRS_AimMod_HSensitivityOption;
■protected ref NumericOptionsAccess■■m_ControllerRS_AimMod_CurvatureOption;
■
■// gamepad/controller selectors
■protected ref OptionSelectorSlider■■m_ControllerLS_VSensitivitySelector;
■protected ref OptionSelectorSlider■■m_ControllerLS_HSensitivitySelector;
■protected ref OptionSelectorSlider■■m_ControllerLS_VehicleMod_HSensitivitySelector;
```

```
// ---------------------------------------------
// ------ OptionsMenuGame.c - START --------------------
// ---------------------------------------------


class OptionsMenuGame extends ScriptedWidgetEventHandler
{
■protected Widget■■■■■■m_Root;
■
■protected Widget■■■■■■m_SettingsRoot;
■protected Widget■■■■■■m_DetailsRoot;
■protected Widget■■■■■■m_DetailsBodyDefault;
■protected Widget■■■■■■m_DetailsBodyConnectivity;
■protected TextWidget■■■■■m_DetailsLabel;
■protected RichTextWidget■■■■m_DetailsText;
■protected ButtonWidget■■■■■m_QuickbarButton;
■
■protected ref NumericOptionsAccess■■m_FOVOption;
■protected ref ListOptionsAccess■■■m_LanguageOption;
■protected ref ListOptionsAccess■■■m_PauseOption;
■#ifdef PLATFORM_CONSOLE
■protected ref NumericOptionsAccess■■m_BrightnessOption;
■protected ref OptionSelectorSlider■■m_BrightnessSelector;
■#endif
■
■protected ref OptionSelectorMultistate■m_LanguageSelector;
■protected ref OptionSelectorSlider■■m_FOVSelector;
■protected ref OptionSelectorMultistate■m_ShowHUDSelector;
■protected ref OptionSelectorMultistate■m_ShowCrosshairSelector;
■protected ref OptionSelectorMultistate■m_ShowQuickbarSelector;
■protected ref OptionSelectorMultistate■m_ShowGameSelector;
■protected ref OptionSelectorMultistate■m_ShowAdminSelector;
■protected ref OptionSelectorMultistate■m_ShowPlayerSelector;
■protected ref OptionSelectorMultistate ■m_ShowServerInfoSelector;
■protected ref OptionSelectorMultistate ■m_BleedingIndication;
■protected ref OptionSelectorMultistate ■m_ConnectivityInfo;
■protected ref OptionSelectorMultistate■m_PauseSelector;
■
■protected GameOptions■■■■■m_Options;
■protected OptionsMenu■■■■■m_Menu;
■
■protected ref map<int, ref Param2<string, string>> m_TextMap;
■
■void OptionsMenuGame(Widget parent, Widget details_root, GameOptions options, OptionsMenu menu)
■{
■■m_Root■■■■■■= GetGame().GetWorkspace().CreateWidgets(GetLayoutName(), parent);
■■
■■m_DetailsRoot■■■■= details_root;
■■m_DetailsBodyDefault ■■= m_DetailsRoot.FindAnyWidget("settings_details_body");
■■m_DetailsBodyConnectivity ■= m_DetailsRoot.FindAnyWidget("settings_details_body_connectivity");
■■m_DetailsLabel■■■■= TextWidget.Cast(m_DetailsRoot.FindAnyWidget("details_label"));
■■m_DetailsText■■■■= RichTextWidget.Cast(m_DetailsRoot.FindAnyWidget("details_content"));
■■
■■m_Options■■■■■= options;
■■m_Menu■■■■■■= menu;
■■
■■m_FOVOption■■■■■= NumericOptionsAccess.Cast(m_Options.GetOptionByType(OptionAccessTyp
■■m_LanguageOption■■■= ListOptionsAccess.Cast(m_Options.GetOptionByType(OptionAccessType.A
■■m_PauseOption■■■■= ListOptionsAccess.Cast(m_Options.GetOptionByType(OptionAccessType.AT
■■
■■m_Root.FindAnyWidget("fov_setting_option").SetUserID(OptionAccessType.AT_OPTIONS_FIELD_OF
■■m_Root.FindAnyWidget("hud_setting_option").SetUserID(OptionIDsScript.OPTION_HUD);
■■m_Root.FindAnyWidget("crosshair_setting_option").SetUserID(OptionIDsScript.OPTION_CROSSHAIR
// ---------------------------------------------
// ------ OptionsMenuSounds.c - START --------------------
// ---------------------------------------------


class OptionsMenuSounds extends ScriptedWidgetEventHandler
{
■protected Widget■■■■■■m_Root;
■
■protected Widget■■■■■■m_SettingsRoot;
■protected Widget■■■■■■m_DetailsRoot;
■protected Widget■■■■■■m_DetailsBodyDefault;
■protected Widget■■■■■■m_DetailsBodyConnectivity;
■protected TextWidget■■■■■m_DetailsLabel;
■protected RichTextWidget■■■■m_DetailsText;
■
■protected ref NumericOptionsAccess■■m_MasterOption;
■protected ref NumericOptionsAccess■■m_EffectsOption;
■protected ref NumericOptionsAccess■■m_VOIPOption;
■protected ref NumericOptionsAccess  ■m_VOIPThresholdOption;
■protected ref NumericOptionsAccess■■m_MusicOption;
■protected ref ListOptionsAccess■■■m_InputModeOption;
■
■protected ref OptionSelectorSlider■■m_MasterSelector;
■protected ref OptionSelectorSlider■■m_EffectsSelector;
■protected ref OptionSelectorSlider■■m_VOIPSelector;
■protected ref OptionSelectorLevelMarker■m_VOIPThresholdSelector;
■protected ref OptionSelectorSlider■■m_MusicSelector;
■protected ref OptionSelectorMultistate■m_InputModeSelector;
■
■protected ref Timer ■■■■■m_AudioLevelTimer;
■protected GameOptions■■■■■m_Options;
■protected OptionsMenu■■■■■m_Menu;
■protected MissionGameplay■■■■m_MissionGameplay;
■protected VONManagerBase■■■■m_VonManager;
■
■protected ref map<int, ref Param2<string, string>> m_TextMap;
■
■private bool ■■■■■■■m_WasMicCapturing;
■
■void OptionsMenuSounds(Widget parent, Widget details_root, GameOptions options, OptionsMenu men
■{
■■m_Root■■■■■= GetGame().GetWorkspace().CreateWidgets(GetLayoutName(), parent);
■■m_DetailsRoot■■■= details_root;
■■m_DetailsBodyDefault ■■= m_DetailsRoot.FindAnyWidget("settings_details_body");
■■m_DetailsBodyConnectivity ■= m_DetailsRoot.FindAnyWidget("settings_details_body_connectivity");
■■m_DetailsLabel■■■= TextWidget.Cast( m_DetailsRoot.FindAnyWidget("details_label"));
■■m_DetailsText■■■= RichTextWidget.Cast( m_DetailsRoot.FindAnyWidget("details_content"));■
■■m_Options■■■■= options;
■■m_Menu■■■■■= menu;
■■
■■m_MasterOption■■■= NumericOptionsAccess.Cast(m_Options.GetOptionByType(OptionAccessType
■■m_EffectsOption■■■= NumericOptionsAccess.Cast(m_Options.GetOptionByType(OptionAccessType
■■m_MusicOption■■■= NumericOptionsAccess.Cast(m_Options.GetOptionByType(OptionAccessType.
■■m_VOIPOption■■■= NumericOptionsAccess.Cast(m_Options.GetOptionByType(OptionAccessType.A
■■m_VOIPThresholdOption■= NumericOptionsAccess.Cast(m_Options.GetOptionByType(OptionAccess
■■m_InputModeOption■■= ListOptionsAccess.Cast(m_Options.GetOptionByType(OptionAccessType.A
■■
■■m_MissionGameplay■■= MissionGameplay.Cast(GetGame().GetMission());
■■m_VonManager ■■■= VONManager.GetInstance();
■■m_AudioLevelTimer = new Timer();
■■m_AudioLevelTimer.Run(0.1, this, "UpdateAudioLevel", NULL, true);
■■
```

```
// ---------------------------------------------
// ------ OptionsMenuVideo.c - START --------------------
// ---------------------------------------------


class OptionsMenuVideo extends ScriptedWidgetEventHandler
{
■protected Widget■■■■■■m_Root;
■
■protected Widget■■■■■■m_SettingsRoot;
■protected Widget■■■■■■m_DetailsRoot;
■protected Widget■■■■■■m_DetailsBodyDefault;
■protected Widget■■■■■■m_DetailsBodyConnectivity;
■protected TextWidget■■■■■m_DetailsLabel;
■protected RichTextWidget■■■■m_DetailsText;
■
■protected GameOptions■■■■■m_Options;
■protected OptionsMenu■■■■■m_Menu;
■
■#ifdef PLATFORM_CONSOLE
■■#ifdef PLATFORM_PS4
■■■protected ref OptionSelectorMultistate■m_FlipModeSelector;
■■#endif
■#else
■■//Overall
■■protected ref OptionSelectorMultistate■m_OverallQualitySelector;
■■
■■//Screen
■■protected ref OptionSelectorMultistate■m_DisplayModeSelector;
■■protected ref OptionSelectorMultistate■m_ResolutionSelector;
■■protected ref OptionSelectorSlider■■m_BrightnessSelector;
■■protected ref OptionSelectorMultistate■m_VSyncSelector;
■■//protected ref OptionSelectorMultistate■m_ColorDepthSelector;
■■
■■//Scene
■■protected ref OptionSelectorMultistate■m_ObjectDetailSelector;
■■protected ref OptionSelectorMultistate■m_TerrainDetailSelector;
■■protected ref OptionSelectorMultistate■m_TextureDetailSelector;
■■protected ref OptionSelectorMultistate■m_ShadowDetailSelector;
■■
■■//Rendering
■■protected ref OptionSelectorMultistate■m_TextureFilteringSelector;
■■protected ref OptionSelectorMultistate■m_TerrainSurfaceDetailSelector;
■■protected ref OptionSelectorMultistate■m_PPAASelector;
■■protected ref OptionSelectorMultistate■m_HWAASelector;
■■protected ref OptionSelectorMultistate■m_ATOCSelector;
■■protected ref OptionSelectorMultistate■m_AOSelector;
■■protected ref OptionSelectorMultistate■m_PPQualitySelector;
■■protected ref OptionSelectorMultistate■m_SSRQualitySelector;
■#endif
■
■#ifdef PLATFORM_CONSOLE
■■#ifdef PLATFORM_PS4
■■■protected ref ListOptionsAccess■■■m_FlipModeOption;
■■#endif
■#else
■■//Overall
■■protected ref ListOptionsAccess■■■m_OverallQualityOption;
■■
■■//Screen
■■protected ref ListOptionsAccess■■■m_DisplayModeOption;
■■protected ref ListOptionsAccess■■■m_ResolutionOption;
■■protected ref NumericOptionsAccess■■m_BrightnessOption;
```

```
// ---------------------------------------------
// ------ Orange.c - START --------------------
// ---------------------------------------------


class Orange : Edible_Base
{
■override bool CanBeCooked()
■{
■■return false;
■}■■
■
■override bool CanBeCookedOnStick()
■{
■■return false;
■}
■
■override bool IsFruit()
■{
■■return true;
■}
■
■override bool CanDecay()
■{
■■return true;
■}
■
■override void SetActions()
■{
■■super.SetActions();
■■
■■AddAction(ActionForceFeed);
■■AddAction(ActionEatFruit);
■■AddAction(ActionCreateIndoorFireplace);
■■AddAction(ActionCreateIndoorOven);
■}
}



// ---------------------------------------------
// ------ Orange.c - END ----------------------
// ---------------------------------------------


// ---------------------------------------------
// ------ OrientalMachete.c - START --------------------
// ---------------------------------------------


class OrientalMachete extends ToolBase
{
■void Machete()
■{
■}

■override bool IsMeleeFinisher()
■{
■■return true;
■}
■
■override array<int> GetValidFinishers()
■{
```

```
// ---------------------------------------------
// ------ OrienteeringCompass.c - START --------------------
// ---------------------------------------------


class OrienteeringCompass: ItemCompass {};



// ---------------------------------------------
// ------ OrienteeringCompass.c - END ----------------------
// ---------------------------------------------


// ---------------------------------------------
// ------ OTS14.c - START --------------------
// ---------------------------------------------


class OTS14 : Rifle_Base
{
};


// ---------------------------------------------
// ------ OTS14.c - END ----------------------
// ---------------------------------------------


// ---------------------------------------------
// ------ OvenIndoor.c - START --------------------
// ---------------------------------------------


class OvenIndoor extends FireplaceBase
{
■protected float ■■■■m_SmokePosX;
■protected float ■■■■m_SmokePosY;
■protected float ■■■■m_SmokePosZ;
■protected int■■■■■m_FirePointIndex = 1;■//limited to 1 decimal place (1-9)
■
■static const string OVENPOINT_ACTION_SELECTION■= "oven_action";
■static const string OVENPOINT_FIRE_POSITION ■= "oven_point";
■static const string OVENPOINT_PLACE_ROT ■■= "oven_rot";
■static const string OVENPOINT_SMOKE_POSITION ■= "oven_smoke";
■
■void OvenIndoor()
■{
■■//Particles - default for FireplaceBase
■■PARTICLE_FIRE_START ■= ParticleList.OVEN_FIRE_START;
■■PARTICLE_SMALL_FIRE ■= ParticleList.OVEN_SMALL_FIRE;
■■PARTICLE_NORMAL_FIRE■= ParticleList.OVEN_NORMAL_FIRE;
■■PARTICLE_SMALL_SMOKE ■= ParticleList.HOUSE_SMALL_SMOKE;
■■PARTICLE_NORMAL_SMOKE■= ParticleList.HOUSE_NORMAL_SMOKE;
■■PARTICLE_FIRE_END ■■= ParticleList.OVEN_FIRE_END;
■■PARTICLE_STEAM_END■■= ParticleList.BARREL_FIRE_STEAM_2END;
■■
■■//register sync variables
■■RegisterNetSyncVariableFloat( "m_SmokePosX", 0, 0, 2 );
■■RegisterNetSyncVariableFloat( "m_SmokePosY", 0, 0, 2 );
■■RegisterNetSyncVariableFloat( "m_SmokePosZ", 0, 0, 2 );
■■RegisterNetSyncVariableInt( "m_FirePointIndex", 0, 9 );
■■
```

```
// ---------------------------------------------
// ------ P1.c - START --------------------
// ---------------------------------------------


class P1_Base : Pistol_Base
{
■override RecoilBase SpawnRecoilObject()
■{
■■return new P1Recoil(this);
■}
■
};

class P1 : P1_Base {};


// ---------------------------------------------
// ------ P1.c - END ----------------------
// ---------------------------------------------


// ---------------------------------------------
// ------ P1Recoil.c - START --------------------
// ---------------------------------------------


class P1Recoil: RecoilBase
{
■override void Init()
■{
■■vector point_1;
■■vector point_2;
■■vector point_3;
■■vector point_4;
■■point_1[0] = m_Player.GetRandomGeneratorSyncManager().GetRandomInRange(RandomGeneratorS
■■point_2[0] = m_Player.GetRandomGeneratorSyncManager().GetRandomInRange(RandomGeneratorS
■■point_3[0] = m_Player.GetRandomGeneratorSyncManager().GetRandomInRange(RandomGeneratorS
■■point_4[0] = m_Player.GetRandomGeneratorSyncManager().GetRandomInRange(RandomGeneratorS
■■m_HandsCurvePoints.Insert(point_1);//forms a 2 dimensional spline(z is ignored)
■■m_HandsCurvePoints.Insert(point_2);
■■m_HandsCurvePoints.Insert(point_3);
■■m_HandsCurvePoints.Insert(point_4);
■■m_HandsCurvePoints.Insert("0 0 0");
■■m_HandsOffsetRelativeTime = 0.5;
■■
■■m_MouseOffsetRangeMin = 75;//in degrees min
■■m_MouseOffsetRangeMax = 105;//in degrees max
■■m_MouseOffsetDistance = 1.2;//how far should the mouse travel
■■m_MouseOffsetRelativeTime = 0.65;//[0..1] a time it takes to move the mouse the required distance rel
■
■■m_CamOffsetDistance = 0.02;
■■m_CamOffsetRelativeTime = 1;
■}
}


// ---------------------------------------------
// ------ P1Recoil.c - END ----------------------
// ---------------------------------------------
// --------------------------------------------
// ------ PackagedFood.c - START --------------------
// --------------------------------------------


class Honey: Edible_Base
{
■override void SetActions()
■{
■■super.SetActions();
■■
■■AddAction(ActionForceFeedSmall);
■■AddAction(ActionEatSmall);
■}
};

class Zagorky_ColorBase: Edible_Base
{
■override void SetActions()
■{
■■super.SetActions();
■■
■■AddAction(ActionForceFeedSmall);
■■AddAction(ActionEatSmall);
■}
};

class Snack_ColorBase: Edible_Base
{
■override void SetActions()
■{
■■super.SetActions();
■■
■■AddAction(ActionForceFeedSmall);
■■AddAction(ActionEatSmall);
■}
};



// --------------------------------------------
// ------ PackagedFood.c - END ---------------------
// --------------------------------------------


// --------------------------------------------
// ------ Paddle.c - START --------------------
// --------------------------------------------


class Paddle: Inventory_Base
{
■override void SetActions()
■{
■■super.SetActions();
■■
■■AddAction(ActionClapBearTrapWithThisItem);
■}
};



// --------------------------------------------
// ---------------------------------------------
// ------ PainHeavyState.c - START --------------------
// ---------------------------------------------


class PainHeavySymptom extends SymptomBase
{
■//this is just for the Symptom parameters set-up and is called even if the Symptom doesn't execute, don't
■override void OnInit()
■{
■■m_SymptomType = SymptomTypes.PRIMARY;
■■m_Priority = 1;
■■m_ID = SymptomIDs.SYMPTOM_PAIN_HEAVY;
■■m_SyncToClient = true;
■■m_MaxCount = 2;
■■m_Duration = 1;
■}
■
■//!gets called every frame
■override void OnUpdateServer(PlayerBase player, float deltatime)
■{

■}

■override void OnUpdateClient(PlayerBase player, float deltatime)
■{
■}
■
■override void OnAnimationPlayFailed()
■{
■■
■}
■
■override bool CanActivate()
■{
■■return true;
■}
■
■//!gets called once on an Symptom which is being activated
■override void OnGetActivatedServer(PlayerBase player)
■{
■■if (LogManager.IsSymptomLogEnable())
■■■Debug.SymptomLog("n/a", this.ToString(), "n/a", "OnGetActivated", m_Player.ToString());
■■
■■PlaySound(EPlayerSoundEventID.TAKING_DMG_HEAVY);
■}

■//!gets called once on a Symptom which is being activated
■override void OnGetActivatedClient(PlayerBase player)
■{
■■//player.SpawnShockEffect(0.5);
■■player.SpawnDamageDealtEffect2();
■■if (LogManager.IsSymptomLogEnable()) Debug.SymptomLog("n/a", this.ToString(), "n/a", "OnGetActiva
■}

■//!only gets called once on an active Symptom that is being deactivated
■override void OnGetDeactivatedServer(PlayerBase player)
■{
■■if (LogManager.IsSymptomLogEnable()) Debug.SymptomLog("n/a", this.ToString(), "n/a", "OnGetDeac
■}

■//!only gets called once on an active Symptom that is being deactivated
■override void OnGetDeactivatedClient(PlayerBase player)
```

```
// ---------------------------------------------
// ------ PainKillersMdfr.c - START --------------------
// ---------------------------------------------


class PainKillersMdfr: ModifierBase
{
■const int LIFETIME = 240;
■const int ACTIVATION_DELAY = 15;
■override void Init()
■{
■■m_TrackActivatedTime = true;
■■m_IsPersistent = true;
■■m_ID ■■■■■= eModifiers.MDF_PAINKILLERS;
■■m_TickIntervalInactive ■= DEFAULT_TICK_TIME_INACTIVE;
■■m_TickIntervalActive ■= 1;
■■DisableActivateCheck();
■}

■override bool ActivateCondition(PlayerBase player)
■{
■■return false;
■}
■
■override void OnReconnect(PlayerBase player)
■{
■■OnActivate(player);
■}
■
■override string GetDebugText()
■{
■■return (LIFETIME - GetAttachedTime()).ToString();
■}

■
■override void OnActivate(PlayerBase player)
■{
■■if (player.GetBrokenLegs() != eBrokenLegs.NO_BROKEN_LEGS)
■■■player.m_ShockHandler.SetMultiplier(0.5);//was 0.75 //Switch the shock multiplier NEED A CONST
■■player.IncreaseHealingsCount();
■■/*
■■if( player.GetNotifiersManager() )
■■■player.GetNotifiersManager().ActivateByType(eNotifiers.NTF_PILLS);
■■*/
■■m_Player.m_InjuryHandler.m_ForceInjuryAnimMask = m_Player.m_InjuryHandler.m_ForceInjuryAnim
■}
■
■override void OnDeactivate(PlayerBase player)
■{
■■if (player.GetBrokenLegs() != eBrokenLegs.NO_BROKEN_LEGS)
■■■player.m_ShockHandler.SetMultiplier(1); //Reset the shock multiplier when modifier stops
■■player.DecreaseHealingsCount();
■■/*
■■if( player.GetNotifiersManager() )
■■■player.GetNotifiersManager().DeactivateByType(eNotifiers.NTF_PILLS);
■■*/
■■m_Player.m_InjuryHandler.m_ForceInjuryAnimMask = m_Player.m_InjuryHandler.m_ForceInjuryAnim
■■m_Player.m_InjuryHandler.m_ForceInjuryAnimMask = m_Player.m_InjuryHandler.m_ForceInjuryAnim
■}
■
■override bool DeactivateCondition(PlayerBase player)
■{
■■float attached_time = GetAttachedTime();
```

```
// ---------------------------------------------
// ------ PainkillerTablets.c - START --------------------
// ---------------------------------------------


class PainkillerTablets extends Edible_Base
{
■override void SetActions()
■{
■■super.SetActions();
■■
■■AddAction(ActionForceConsumeSingle);
■■AddAction(ActionConsumeSingle);
■}
■
■override void OnConsume(float amount, PlayerBase consumer)
■{
■■if (consumer.GetModifiersManager().IsModifierActive(eModifiers.MDF_PAINKILLERS)) // effectively res
■■{
■■■consumer.GetModifiersManager().DeactivateModifier(eModifiers.MDF_PAINKILLERS, false);
■■}

■■consumer.GetModifiersManager().ActivateModifier(eModifiers.MDF_PAINKILLERS);
■}
}


// ---------------------------------------------
// ------ PainkillerTablets.c - END ----------------------
// ---------------------------------------------


// ---------------------------------------------
// ------ PainLightState.c - START --------------------
// ---------------------------------------------


class PainLightSymptom extends SymptomBase
{
■//this is just for the Symptom parameters set-up and is called even if the Symptom doesn't execute, don't
■override void OnInit()
■{
■■m_SymptomType = SymptomTypes.PRIMARY;
■■m_Priority = 1;
■■m_ID = SymptomIDs.SYMPTOM_PAIN_LIGHT;
■■m_SyncToClient = true;
■■m_MaxCount = 2;
■■m_Duration = 1;
■}
■
■//!gets called every frame
■override void OnUpdateServer(PlayerBase player, float deltatime)
■{

■}

■override void OnUpdateClient(PlayerBase player, float deltatime)
■{
■}
■
■override void OnAnimationPlayFailed()
■{
■■
```

```
// ---------------------------------------------
// ------ PaintAK101.c - START --------------------
// ---------------------------------------------


class PaintAK101 extends RecipeBase
{■
■override void Init()
■{
■■m_Name = "#STR_paint0";
■■m_IsInstaRecipe = false;//should this recipe be performed instantly without animation
■■m_AnimationLength = 1.5;//animation length in relative time units
■■m_Specialty = 0.01;// value > 0 for roughness, value < 0 for precision
■■
■■
■■//conditions
■■m_MinDamageIngredient[0] = -1;//-1 = disable check
■■m_MaxDamageIngredient[0] = 3;//-1 = disable check
■■
■■m_MinQuantityIngredient[0] = 25;//-1 = disable check
■■m_MaxQuantityIngredient[0] = -1;//-1 = disable check
■■
■■m_MinDamageIngredient[1] = -1;//-1 = disable check
■■m_MaxDamageIngredient[1] = 3;//-1 = disable check
■■
■■m_MinQuantityIngredient[1] = -1;//-1 = disable check
■■m_MaxQuantityIngredient[1] = -1;//-1 = disable check
■■//----------------------------------------------------------------------------------------------------------------------
■■
■■//INGREDIENTS
■■//ingredient 1
■■InsertIngredient(0,"Spraycan_ColorBase");//you can insert multiple ingredients this way
■■
■■m_IngredientAddHealth[0] = 0;// 0 = do nothing
■■m_IngredientSetHealth[0] = -1; // -1 = do nothing
■■m_IngredientAddQuantity[0] = -25;// 0 = do nothing
■■m_IngredientDestroy[0] = false;//true = destroy, false = do nothing
■■m_IngredientUseSoftSkills[0] = true;// set 'true' to allow modification of the values by softskills on this in
■■
■■//ingredient 2
■■InsertIngredient(1,"AK101");//you can insert multiple ingredients this way
■■
■■m_IngredientAddHealth[1] = 0;// 0 = do nothing
■■m_IngredientSetHealth[1] = -1; // -1 = do nothing
■■m_IngredientAddQuantity[1] = 0;// 0 = do nothing
■■m_IngredientDestroy[1] = false;// false = do nothing
■■m_IngredientUseSoftSkills[1] = false;// set 'true' to allow modification of the values by softskills on this i
■■//----------------------------------------------------------------------------------------------------------------------
■■
■■//result1
■■//AddResult("AK101_");//add results here
/*
■■m_ResultSetFullQuantity[0] = false;//true = set full quantity, false = do nothing
■■m_ResultSetQuantity[0] = -1;//-1 = do nothing
■■m_ResultSetHealth[0] = -1;//-1 = do nothing
■■m_ResultInheritsHealth[0] = 1;// (value) == -1 means do nothing; a (value) >= 0 means this result will in
■■m_ResultInheritsColor[0] = 0;// (value) == -1 means do nothing; a (value) >= 0 means this result classn
■■m_ResultToInventory[0] = 1;//(value) == -2 spawn result on the ground;(value) == -1 place anywhere in
■■m_ResultUseSoftSkills[0] = false;// set 'true' to allow modification of the values by softskills on this resu
■■m_ResultReplacesIngredient[0] = 1;// value == -1 means do nothing; a value >= 0 means this result wil
■■*/
■}
```

```
// ---------------------------------------------
// ------ PaintAK10130Mag.c - START --------------------
// ---------------------------------------------


class PaintAK10130Mag extends RecipeBase
{■
■override void Init()
■{
■■m_Name = "#STR_paint0";
■■m_IsInstaRecipe = false;//should this recipe be performed instantly without animation
■■m_AnimationLength = 1.5;//animation length in relative time units
■■m_Specialty = 0.01;// value > 0 for roughness, value < 0 for precision
■■
■■
■■//conditions
■■m_MinDamageIngredient[0] = -1;//-1 = disable check
■■m_MaxDamageIngredient[0] = 3;//-1 = disable check
■■
■■m_MinQuantityIngredient[0] = 25;//-1 = disable check
■■m_MaxQuantityIngredient[0] = -1;//-1 = disable check
■■
■■m_MinDamageIngredient[1] = -1;//-1 = disable check
■■m_MaxDamageIngredient[1] = 3;//-1 = disable check
■■
■■m_MinQuantityIngredient[1] = -1;//-1 = disable check
■■m_MaxQuantityIngredient[1] = -1;//-1 = disable check
■■//----------------------------------------------------------------------------------------------------------------------
■■
■■//INGREDIENTS
■■//ingredient 1
■■InsertIngredient(0,"Spraycan_ColorBase");//you can insert multiple ingredients this way
■■
■■m_IngredientAddHealth[0] = 0;// 0 = do nothing
■■m_IngredientSetHealth[0] = -1; // -1 = do nothing
■■m_IngredientAddQuantity[0] = -25;// 0 = do nothing
■■m_IngredientDestroy[0] = false;//true = destroy, false = do nothing
■■m_IngredientUseSoftSkills[0] = true;// set 'true' to allow modification of the values by softskills on this in
■■
■■//ingredient 2
■■InsertIngredient(1,"Mag_AK101_30Rnd");//you can insert multiple ingredients this way
■■
■■m_IngredientAddHealth[1] = 0;// 0 = do nothing
■■m_IngredientSetHealth[1] = -1; // -1 = do nothing
■■m_IngredientAddQuantity[1] = 0;// 0 = do nothing
■■m_IngredientDestroy[1] = false;// false = do nothing
■■m_IngredientUseSoftSkills[1] = false;// set 'true' to allow modification of the values by softskills on this i
■■//----------------------------------------------------------------------------------------------------------------------
■■
■■//result1
■■/*
■■AddResult("Mag_AK101_30Rnd_");//add results here

■■m_ResultSetFullQuantity[0] = false;//true = set full quantity, false = do nothing
■■m_ResultSetQuantity[0] = -1;//-1 = do nothing
■■m_ResultSetHealth[0] = -1;//-1 = do nothing
■■m_ResultInheritsHealth[0] = 1;// (value) == -1 means do nothing; a (value) >= 0 means this result will in
■■m_ResultInheritsColor[0] = 0;// (value) == -1 means do nothing; a (value) >= 0 means this result classn
■■m_ResultToInventory[0] = 1;//(value) == -2 spawn result on the ground;(value) == -1 place anywhere in
■■m_ResultUseSoftSkills[0] = false;// set 'true' to allow modification of the values by softskills on this resu
■■m_ResultReplacesIngredient[0] = 1;// value == -1 means do nothing; a value >= 0 means this result wil
■■*/
■}
```

```
// ----------------------------------------------
// ------ PaintAK74.c - START --------------------
// ----------------------------------------------


class PaintAK74 extends RecipeBase
{■
■override void Init()
■{
■■m_Name = "#STR_paint0";
■■m_IsInstaRecipe = false;//should this recipe be performed instantly without animation
■■m_AnimationLength = 1.5;//animation length in relative time units
■■m_Specialty = 0.01;// value > 0 for roughness, value < 0 for precision
■■
■■
■■//conditions
■■m_MinDamageIngredient[0] = -1;//-1 = disable check
■■m_MaxDamageIngredient[0] = 3;//-1 = disable check
■■
■■m_MinQuantityIngredient[0] = 25;//-1 = disable check
■■m_MaxQuantityIngredient[0] = -1;//-1 = disable check
■■
■■m_MinDamageIngredient[1] = -1;//-1 = disable check
■■m_MaxDamageIngredient[1] = 3;//-1 = disable check
■■
■■m_MinQuantityIngredient[1] = -1;//-1 = disable check
■■m_MaxQuantityIngredient[1] = -1;//-1 = disable check
■■//----------------------------------------------------------------------------------------------------------------------
■■
■■//INGREDIENTS
■■//ingredient 1
■■InsertIngredient(0,"Spraycan_ColorBase");//you can insert multiple ingredients this way
■■
■■m_IngredientAddHealth[0] = 0;// 0 = do nothing
■■m_IngredientSetHealth[0] = -1; // -1 = do nothing
■■m_IngredientAddQuantity[0] = -25;// 0 = do nothing
■■m_IngredientDestroy[0] = false;//true = destroy, false = do nothing
■■m_IngredientUseSoftSkills[0] = true;// set 'true' to allow modification of the values by softskills on this in
■■
■■//ingredient 2
■■InsertIngredient(1,"AK74");//you can insert multiple ingredients this way
■■
■■m_IngredientAddHealth[1] = 0;// 0 = do nothing
■■m_IngredientSetHealth[1] = -1; // -1 = do nothing
■■m_IngredientAddQuantity[1] = 0;// 0 = do nothing
■■m_IngredientDestroy[1] = false;// false = do nothing
■■m_IngredientUseSoftSkills[1] = false;// set 'true' to allow modification of the values by softskills on this i
■■//----------------------------------------------------------------------------------------------------------------------
■■
■■//result1
■■AddResult("AK74_");//add results here

■■m_ResultSetFullQuantity[0] = false;//true = set full quantity, false = do nothing
■■m_ResultSetQuantity[0] = -1;//-1 = do nothing
■■m_ResultSetHealth[0] = -1;//-1 = do nothing
■■m_ResultInheritsHealth[0] = 1;// (value) == -1 means do nothing; a (value) >= 0 means this result will in
■■m_ResultInheritsColor[0] = 0;// (value) == -1 means do nothing; a (value) >= 0 means this result classn
■■m_ResultToInventory[0] = 1;//(value) == -2 spawn result on the ground;(value) == -1 place anywhere in
■■m_ResultUseSoftSkills[0] = false;// set 'true' to allow modification of the values by softskills on this resu
■■m_ResultReplacesIngredient[0] = 1;// value == -1 means do nothing; a value >= 0 means this result wil
■}

■override bool CanDo(ItemBase ingredients[], PlayerBase player)
```

```
// ---------------------------------------------
// ------ PaintAK7430Mag.c - START --------------------
// ---------------------------------------------


class PaintAK7430Mag extends RecipeBase
{■
■override void Init()
■{
■■m_Name = "#STR_paint0";
■■m_IsInstaRecipe = false;//should this recipe be performed instantly without animation
■■m_AnimationLength = 1.5;//animation length in relative time units
■■m_Specialty = 0.01;// value > 0 for roughness, value < 0 for precision
■■
■■
■■//conditions
■■m_MinDamageIngredient[0] = -1;//-1 = disable check
■■m_MaxDamageIngredient[0] = 3;//-1 = disable check
■■
■■m_MinQuantityIngredient[0] = 25;//-1 = disable check
■■m_MaxQuantityIngredient[0] = -1;//-1 = disable check
■■
■■m_MinDamageIngredient[1] = -1;//-1 = disable check
■■m_MaxDamageIngredient[1] = 3;//-1 = disable check
■■
■■m_MinQuantityIngredient[1] = -1;//-1 = disable check
■■m_MaxQuantityIngredient[1] = -1;//-1 = disable check
■■■//----------------------------------------------------------------------------------------------------------------------
■■
■■//INGREDIENTS
■■//ingredient 1
■■InsertIngredient(0,"Spraycan_ColorBase");//you can insert multiple ingredients this way
■■
■■m_IngredientAddHealth[0] = 0;// 0 = do nothing
■■m_IngredientSetHealth[0] = -1; // -1 = do nothing
■■m_IngredientAddQuantity[0] = -25;// 0 = do nothing
■■m_IngredientDestroy[0] = false;//true = destroy, false = do nothing
■■m_IngredientUseSoftSkills[0] = true;// set 'true' to allow modification of the values by softskills on this in
■■
■■//ingredient 2
■■InsertIngredient(1,"Mag_AK74_30Rnd");//you can insert multiple ingredients this way
■■
■■m_IngredientAddHealth[1] = 0;// 0 = do nothing
■■m_IngredientSetHealth[1] = -1; // -1 = do nothing
■■m_IngredientAddQuantity[1] = 0;// 0 = do nothing
■■m_IngredientDestroy[1] = false;// false = do nothing
■■m_IngredientUseSoftSkills[1] = false;// set 'true' to allow modification of the values by softskills on this i
■■■//----------------------------------------------------------------------------------------------------------------------
■■
■■//result1
■■■/*
■■AddResult("Mag_AK74_30Rnd_");//add results here

■■m_ResultSetFullQuantity[0] = false;//true = set full quantity, false = do nothing
■■m_ResultSetQuantity[0] = 1;//-1 = do nothing
■■m_ResultSetHealth[0] = -1;//-1 = do nothing
■■m_ResultInheritsHealth[0] = 1;// (value) == -1 means do nothing; a (value) >= 0 means this result will in
■■m_ResultInheritsColor[0] = 0;// (value) == -1 means do nothing; a (value) >= 0 means this result classn
■■m_ResultToInventory[0] = 1;//(value) == -2 spawn result on the ground;(value) == -1 place anywhere in
■■m_ResultUseSoftSkills[0] = false;// set 'true' to allow modification of the values by softskills on this resu
■■m_ResultReplacesIngredient[0] = 1;// value == -1 means do nothing; a value >= 0 means this result wil
■■■*/
■}
```

```
// ---------------------------------------------
// ------ PaintAK74HndgrdBlack.c - START --------------------
// ---------------------------------------------


class PaintAK74HndgrdBlack extends RecipeBase
{■
■override void Init()
■{
■■m_Name = "#STR_paint0";
■■m_IsInstaRecipe = false;//should this recipe be performed instantly without animation
■■m_AnimationLength = 1.5;//animation length in relative time units
■■m_Specialty = 0.01;// value > 0 for roughness, value < 0 for precision
■■
■■
■■//conditions
■■m_MinDamageIngredient[0] = -1;//-1 = disable check
■■m_MaxDamageIngredient[0] = 3;//-1 = disable check
■■
■■m_MinQuantityIngredient[0] = 25;//-1 = disable check
■■m_MaxQuantityIngredient[0] = -1;//-1 = disable check
■■
■■m_MinDamageIngredient[1] = -1;//-1 = disable check
■■m_MaxDamageIngredient[1] = 3;//-1 = disable check
■■
■■m_MinQuantityIngredient[1] = -1;//-1 = disable check
■■m_MaxQuantityIngredient[1] = -1;//-1 = disable check
■■//----------------------------------------------------------------------------------------------------------------------
■■
■■//INGREDIENTS
■■//ingredient 1
■■InsertIngredient(0,"Spraycan_Black");//you can insert multiple ingredients this way
■■
■■m_IngredientAddHealth[0] = 0;// 0 = do nothing
■■m_IngredientSetHealth[0] = -1; // -1 = do nothing
■■m_IngredientAddQuantity[0] = -25;// 0 = do nothing
■■m_IngredientDestroy[0] = false;//true = destroy, false = do nothing
■■m_IngredientUseSoftSkills[0] = true;// set 'true' to allow modification of the values by softskills on this in
■■
■■//ingredient 2
■■InsertIngredient(1,"AK74_Hndgrd");//you can insert multiple ingredients this way
■■
■■m_IngredientAddHealth[1] = 0;// 0 = do nothing
■■m_IngredientSetHealth[1] = -1; // -1 = do nothing
■■m_IngredientAddQuantity[1] = 0;// 0 = do nothing
■■m_IngredientDestroy[1] = false;// false = do nothing
■■m_IngredientUseSoftSkills[1] = false;// set 'true' to allow modification of the values by softskills on this i
■■//----------------------------------------------------------------------------------------------------------------------
■■
■■//result1
■■/*
■■AddResult("AK74_Hndgrd_Black");//add results here

■■m_ResultSetFullQuantity[0] = false;//true = set full quantity, false = do nothing
■■m_ResultSetQuantity[0] = -1;//-1 = do nothing
■■m_ResultSetHealth[0] = -1;//-1 = do nothing
■■m_ResultInheritsHealth[0] = 1;// (value) == -1 means do nothing; a (value) >= 0 means this result will in
■■m_ResultInheritsColor[0] = -1;// (value) == -1 means do nothing; a (value) >= 0 means this result classn
■■m_ResultToInventory[0] = 1;//(value) == -2 spawn result on the ground;(value) == -1 place anywhere in
■■m_ResultUseSoftSkills[0] = false;// set 'true' to allow modification of the values by softskills on this resu
■■m_ResultReplacesIngredient[0] = -1;// value == -1 means do nothing; a value >= 0 means this result wi
■■*/
■}
```

```
// ---------------------------------------------
// ------ PaintAK74HndgrdCamo.c - START --------------------
// ---------------------------------------------


class PaintAK74HndgrdCamo extends RecipeBase
{■
■override void Init()
■{
■■m_Name = "#STR_paint0";
■■m_IsInstaRecipe = false;//should this recipe be performed instantly without animation
■■m_AnimationLength = 1.5;//animation length in relative time units
■■m_Specialty = 0.01;// value > 0 for roughness, value < 0 for precision
■■
■■
■■//conditions
■■m_MinDamageIngredient[0] = -1;//-1 = disable check
■■m_MaxDamageIngredient[0] = 3;//-1 = disable check
■■
■■m_MinQuantityIngredient[0] = 25;//-1 = disable check
■■m_MaxQuantityIngredient[0] = -1;//-1 = disable check
■■
■■m_MinDamageIngredient[1] = -1;//-1 = disable check
■■m_MaxDamageIngredient[1] = 3;//-1 = disable check
■■
■■m_MinQuantityIngredient[1] = -1;//-1 = disable check
■■m_MaxQuantityIngredient[1] = -1;//-1 = disable check
■■//----------------------------------------------------------------------------------------------------------------------
■■
■■//INGREDIENTS
■■//ingredient 1
■■InsertIngredient(0,"Spraycan_Green");//you can insert multiple ingredients this way
■■
■■m_IngredientAddHealth[0] = 0;// 0 = do nothing
■■m_IngredientSetHealth[0] = -1; // -1 = do nothing
■■m_IngredientAddQuantity[0] = -25;// 0 = do nothing
■■m_IngredientDestroy[0] = false;//true = destroy, false = do nothing
■■m_IngredientUseSoftSkills[0] = true;// set 'true' to allow modification of the values by softskills on this in
■■
■■//ingredient 2
■■InsertIngredient(1,"AK74_Hndgrd");//you can insert multiple ingredients this way
■■
■■m_IngredientAddHealth[1] = 0;// 0 = do nothing
■■m_IngredientSetHealth[1] = -1; // -1 = do nothing
■■m_IngredientAddQuantity[1] = 0;// 0 = do nothing
■■m_IngredientDestroy[1] = false;// false = do nothing
■■m_IngredientUseSoftSkills[1] = false;// set 'true' to allow modification of the values by softskills on this i
■■//----------------------------------------------------------------------------------------------------------------------
■■
■■//result1
■■/*
■■AddResult("AK74_Hndgrd_Camo");//add results here

■■m_ResultSetFullQuantity[0] = false;//true = set full quantity, false = do nothing
■■m_ResultSetQuantity[0] = -1;//-1 = do nothing
■■m_ResultSetHealth[0] = -1;//-1 = do nothing
■■m_ResultInheritsHealth[0] = 1;// (value) == -1 means do nothing; a (value) >= 0 means this result will in
■■m_ResultInheritsColor[0] = -1;// (value) == -1 means do nothing; a (value) >= 0 means this result classn
■■m_ResultToInventory[0] = 1;//(value) == -2 spawn result on the ground;(value) == -1 place anywhere in
■■m_ResultUseSoftSkills[0] = false;// set 'true' to allow modification of the values by softskills on this resu
■■m_ResultReplacesIngredient[0] = -1;// value == -1 means do nothing; a value >= 0 means this result wi
■■*/
■}
```

```
// ---------------------------------------------
// ------ PaintAK74WoodBttstckBlack.c - START --------------------
// ---------------------------------------------


class PaintAK74WoodBttstckBlack extends RecipeBase
{■
■override void Init()
■{
■■m_Name = "#STR_paint0";
■■m_IsInstaRecipe = false;//should this recipe be performed instantly without animation
■■m_AnimationLength = 1.5;//animation length in relative time units
■■m_Specialty = 0.01;// value > 0 for roughness, value < 0 for precision
■■
■■
■■//conditions
■■m_MinDamageIngredient[0] = -1;//-1 = disable check
■■m_MaxDamageIngredient[0] = 3;//-1 = disable check
■■
■■m_MinQuantityIngredient[0] = 25;//-1 = disable check
■■m_MaxQuantityIngredient[0] = -1;//-1 = disable check
■■
■■m_MinDamageIngredient[1] = -1;//-1 = disable check
■■m_MaxDamageIngredient[1] = 3;//-1 = disable check
■■
■■m_MinQuantityIngredient[1] = -1;//-1 = disable check
■■m_MaxQuantityIngredient[1] = -1;//-1 = disable check
■■//----------------------------------------------------------------------------------------------------------------------
■■
■■//INGREDIENTS
■■//ingredient 1
■■InsertIngredient(0,"Spraycan_Black");//you can insert multiple ingredients this way
■■
■■m_IngredientAddHealth[0] = 0;// 0 = do nothing
■■m_IngredientSetHealth[0] = -1; // -1 = do nothing
■■m_IngredientAddQuantity[0] = -25;// 0 = do nothing
■■m_IngredientDestroy[0] = false;//true = destroy, false = do nothing
■■m_IngredientUseSoftSkills[0] = true;// set 'true' to allow modification of the values by softskills on this in
■■
■■//ingredient 2
■■InsertIngredient(1,"AK74_WoodBttstck");//you can insert multiple ingredients this way
■■
■■m_IngredientAddHealth[1] = 0;// 0 = do nothing
■■m_IngredientSetHealth[1] = -1; // -1 = do nothing
■■m_IngredientAddQuantity[1] = 0;// 0 = do nothing
■■m_IngredientDestroy[1] = false;// false = do nothing
■■m_IngredientUseSoftSkills[1] = false;// set 'true' to allow modification of the values by softskills on this i
■■//----------------------------------------------------------------------------------------------------------------------
■■
■■//result1
■■/*
■■AddResult("AK74_WoodBttstck_Black");//add results here

■■m_ResultSetFullQuantity[0] = false;//true = set full quantity, false = do nothing
■■m_ResultSetQuantity[0] = -1;//-1 = do nothing
■■m_ResultSetHealth[0] = -1;//-1 = do nothing
■■m_ResultInheritsHealth[0] = 1;// (value) == -1 means do nothing; a (value) >= 0 means this result will in
■■m_ResultInheritsColor[0] = -1;// (value) == -1 means do nothing; a (value) >= 0 means this result classn
■■m_ResultToInventory[0] = 1;//(value) == -2 spawn result on the ground;(value) == -1 place anywhere in
■■m_ResultUseSoftSkills[0] = false;// set 'true' to allow modification of the values by softskills on this resu
■■m_ResultReplacesIngredient[0] = -1;// value == -1 means do nothing; a value >= 0 means this result wi
■■*/
■}
```

```
// ---------------------------------------------
// ------ PaintAK74WoodBttstckCamo.c - START --------------------
// ---------------------------------------------


class PaintAK74WoodBttstckCamo extends RecipeBase
{■
■override void Init()
■{
■■m_Name = "#STR_paint0";
■■m_IsInstaRecipe = false;//should this recipe be performed instantly without animation
■■m_AnimationLength = 1.5;//animation length in relative time units
■■m_Specialty = 0.01;// value > 0 for roughness, value < 0 for precision
■■
■■
■■//conditions
■■m_MinDamageIngredient[0] = -1;//-1 = disable check
■■m_MaxDamageIngredient[0] = 3;//-1 = disable check
■■
■■m_MinQuantityIngredient[0] = 25;//-1 = disable check
■■m_MaxQuantityIngredient[0] = -1;//-1 = disable check
■■
■■m_MinDamageIngredient[1] = -1;//-1 = disable check
■■m_MaxDamageIngredient[1] = 3;//-1 = disable check
■■
■■m_MinQuantityIngredient[1] = -1;//-1 = disable check
■■m_MaxQuantityIngredient[1] = -1;//-1 = disable check
■■//----------------------------------------------------------------------------------------------------------------------
■■
■■//INGREDIENTS
■■//ingredient 1
■■InsertIngredient(0,"Spraycan_Green");//you can insert multiple ingredients this way
■■
■■m_IngredientAddHealth[0] = 0;// 0 = do nothing
■■m_IngredientSetHealth[0] = -1; // -1 = do nothing
■■m_IngredientAddQuantity[0] = -25;// 0 = do nothing
■■m_IngredientDestroy[0] = false;//true = destroy, false = do nothing
■■m_IngredientUseSoftSkills[0] = true;// set 'true' to allow modification of the values by softskills on this in
■■
■■//ingredient 2
■■InsertIngredient(1,"AK74_WoodBttstck");//you can insert multiple ingredients this way
■■
■■m_IngredientAddHealth[1] = 0;// 0 = do nothing
■■m_IngredientSetHealth[1] = -1; // -1 = do nothing
■■m_IngredientAddQuantity[1] = 0;// 0 = do nothing
■■m_IngredientDestroy[1] = false;// false = do nothing
■■m_IngredientUseSoftSkills[1] = false;// set 'true' to allow modification of the values by softskills on this i
■■//----------------------------------------------------------------------------------------------------------------------
■■
■■//result1
■■/*
■■AddResult("AK74_WoodBttstck_Camo");//add results here

■■m_ResultSetFullQuantity[0] = false;//true = set full quantity, false = do nothing
■■m_ResultSetQuantity[0] = -1;//-1 = do nothing
■■m_ResultSetHealth[0] = -1;//-1 = do nothing
■■m_ResultInheritsHealth[0] = 1;// (value) == -1 means do nothing; a (value) >= 0 means this result will in
■■m_ResultInheritsColor[0] = -1;// (value) == -1 means do nothing; a (value) >= 0 means this result classn
■■m_ResultToInventory[0] = 1;//(value) == -2 spawn result on the ground;(value) == -1 place anywhere in
■■m_ResultUseSoftSkills[0] = false;// set 'true' to allow modification of the values by softskills on this resu
■■m_ResultReplacesIngredient[0] = -1;// value == -1 means do nothing; a value >= 0 means this result wi
■■*/
■}
```

```
// ---------------------------------------------
// ------ PaintAKFoldingBttstck.c - START --------------------
// ---------------------------------------------


class PaintAKFoldingBttstck extends RecipeBase
{■
■override void Init()
■{
■■m_Name = "#STR_paint0";
■■m_IsInstaRecipe = false;//should this recipe be performed instantly without animation
■■m_AnimationLength = 1.5;//animation length in relative time units
■■m_Specialty = 0.01;// value > 0 for roughness, value < 0 for precision
■■
■■
■■//conditions
■■m_MinDamageIngredient[0] = -1;//-1 = disable check
■■m_MaxDamageIngredient[0] = 3;//-1 = disable check
■■
■■m_MinQuantityIngredient[0] = 25;//-1 = disable check
■■m_MaxQuantityIngredient[0] = -1;//-1 = disable check
■■
■■m_MinDamageIngredient[1] = -1;//-1 = disable check
■■m_MaxDamageIngredient[1] = 3;//-1 = disable check
■■
■■m_MinQuantityIngredient[1] = -1;//-1 = disable check
■■m_MaxQuantityIngredient[1] = -1;//-1 = disable check
■■//----------------------------------------------------------------------------------------------------------------------
■■
■■//INGREDIENTS
■■//ingredient 1
■■InsertIngredient(0,"Spraycan_ColorBase");//you can insert multiple ingredients this way
■■
■■m_IngredientAddHealth[0] = 0;// 0 = do nothing
■■m_IngredientSetHealth[0] = -1; // -1 = do nothing
■■m_IngredientAddQuantity[0] = -25;// 0 = do nothing
■■m_IngredientDestroy[0] = false;//true = destroy, false = do nothing
■■m_IngredientUseSoftSkills[0] = true;// set 'true' to allow modification of the values by softskills on this in
■■
■■//ingredient 2
■■InsertIngredient(1,"AK_FoldingBttstck");//you can insert multiple ingredients this way
■■
■■m_IngredientAddHealth[1] = 0;// 0 = do nothing
■■m_IngredientSetHealth[1] = -1; // -1 = do nothing
■■m_IngredientAddQuantity[1] = 0;// 0 = do nothing
■■m_IngredientDestroy[1] = false;// false = do nothing
■■m_IngredientUseSoftSkills[1] = false;// set 'true' to allow modification of the values by softskills on this i
■■//----------------------------------------------------------------------------------------------------------------------
■■
■■//result1
■■/*
■■AddResult("AK_FoldingBttstck_");//add results here

■■m_ResultSetFullQuantity[0] = false;//true = set full quantity, false = do nothing
■■m_ResultSetQuantity[0] = -1;//-1 = do nothing
■■m_ResultSetHealth[0] = -1;//-1 = do nothing
■■m_ResultInheritsHealth[0] = 1;// (value) == -1 means do nothing; a (value) >= 0 means this result will in
■■m_ResultInheritsColor[0] = 0;// (value) == -1 means do nothing; a (value) >= 0 means this result classn
■■m_ResultToInventory[0] = 1;//(value) == -2 spawn result on the ground;(value) == -1 place anywhere in
■■m_ResultUseSoftSkills[0] = false;// set 'true' to allow modification of the values by softskills on this resu
■■m_ResultReplacesIngredient[0] = -1;// value == -1 means do nothing; a value >= 0 means this result wi
■■*/
■}
```

```
// ---------------------------------------------
// ------ PaintAKMDrumMag.c - START --------------------
// ---------------------------------------------


class PaintAKMDrumMag extends RecipeBase
{■
■override void Init()
■{
■■m_Name = "#STR_paint0";
■■m_IsInstaRecipe = false;//should this recipe be performed instantly without animation
■■m_AnimationLength = 1.5;//animation length in relative time units
■■m_Specialty = 0.01;// value > 0 for roughness, value < 0 for precision
■■
■■
■■//conditions
■■m_MinDamageIngredient[0] = -1;//-1 = disable check
■■m_MaxDamageIngredient[0] = 3;//-1 = disable check
■■
■■m_MinQuantityIngredient[0] = 25;//-1 = disable check
■■m_MaxQuantityIngredient[0] = -1;//-1 = disable check
■■
■■m_MinDamageIngredient[1] = -1;//-1 = disable check
■■m_MaxDamageIngredient[1] = 3;//-1 = disable check
■■
■■m_MinQuantityIngredient[1] = -1;//-1 = disable check
■■m_MaxQuantityIngredient[1] = -1;//-1 = disable check
■■//----------------------------------------------------------------------------------------------------------------------
■■
■■//INGREDIENTS
■■//ingredient 1
■■InsertIngredient(0,"Spraycan_ColorBase");//you can insert multiple ingredients this way
■■
■■m_IngredientAddHealth[0] = 0;// 0 = do nothing
■■m_IngredientSetHealth[0] = -1; // -1 = do nothing
■■m_IngredientAddQuantity[0] = -25;// 0 = do nothing
■■m_IngredientDestroy[0] = false;//true = destroy, false = do nothing
■■m_IngredientUseSoftSkills[0] = true;// set 'true' to allow modification of the values by softskills on this in
■■
■■//ingredient 2
■■InsertIngredient(1,"Mag_AKM_Drum75Rnd");//you can insert multiple ingredients this way
■■
■■m_IngredientAddHealth[1] = 0;// 0 = do nothing
■■m_IngredientSetHealth[1] = -1; // -1 = do nothing
■■m_IngredientAddQuantity[1] = 0;// 0 = do nothing
■■m_IngredientDestroy[1] = false;// false = do nothing
■■m_IngredientUseSoftSkills[1] = false;// set 'true' to allow modification of the values by softskills on this i
■■//----------------------------------------------------------------------------------------------------------------------
■■
■■//result1
■■/*
■■AddResult("Mag_AKM_Drum75Rnd_");//add results here

■■m_ResultSetFullQuantity[0] = false;//true = set full quantity, false = do nothing
■■m_ResultSetQuantity[0] = 1;//-1 = do nothing
■■m_ResultSetHealth[0] = -1;//-1 = do nothing
■■m_ResultInheritsHealth[0] = 1;// (value) == -1 means do nothing; a (value) >= 0 means this result will in
■■m_ResultInheritsColor[0] = 0;// (value) == -1 means do nothing; a (value) >= 0 means this result classn
■■m_ResultToInventory[0] = 1;//(value) == -2 spawn result on the ground;(value) == -1 place anywhere in
■■m_ResultUseSoftSkills[0] = false;// set 'true' to allow modification of the values by softskills on this resu
■■m_ResultReplacesIngredient[0] = 1;// value == -1 means do nothing; a value >= 0 means this result wil
■■*/
■}
```

```
// ---------------------------------------------
// ------ PaintAKMPalm30Mag.c - START --------------------
// ---------------------------------------------


class PaintAKMPalm30Mag extends RecipeBase
{■
■override void Init()
■{
■■m_Name = "#STR_paint0";
■■m_IsInstaRecipe = false;//should this recipe be performed instantly without animation
■■m_AnimationLength = 1.5;//animation length in relative time units
■■m_Specialty = 0.01;// value > 0 for roughness, value < 0 for precision
■■
■■
■■//conditions
■■m_MinDamageIngredient[0] = -1;//-1 = disable check
■■m_MaxDamageIngredient[0] = 3;//-1 = disable check
■■
■■m_MinQuantityIngredient[0] = 25;//-1 = disable check
■■m_MaxQuantityIngredient[0] = -1;//-1 = disable check
■■
■■m_MinDamageIngredient[1] = -1;//-1 = disable check
■■m_MaxDamageIngredient[1] = 3;//-1 = disable check
■■
■■m_MinQuantityIngredient[1] = -1;//-1 = disable check
■■m_MaxQuantityIngredient[1] = -1;//-1 = disable check
■■//----------------------------------------------------------------------------------------------------------------------
■■
■■//INGREDIENTS
■■//ingredient 1
■■InsertIngredient(0,"Spraycan_ColorBase");//you can insert multiple ingredients this way
■■
■■m_IngredientAddHealth[0] = 0;// 0 = do nothing
■■m_IngredientSetHealth[0] = -1; // -1 = do nothing
■■m_IngredientAddQuantity[0] = -25;// 0 = do nothing
■■m_IngredientDestroy[0] = false;//true = destroy, false = do nothing
■■m_IngredientUseSoftSkills[0] = true;// set 'true' to allow modification of the values by softskills on this in
■■
■■//ingredient 2
■■InsertIngredient(1,"Mag_AKM_Palm30Rnd");//you can insert multiple ingredients this way
■■
■■m_IngredientAddHealth[1] = 0;// 0 = do nothing
■■m_IngredientSetHealth[1] = -1; // -1 = do nothing
■■m_IngredientAddQuantity[1] = 0;// 0 = do nothing
■■m_IngredientDestroy[1] = false;// false = do nothing
■■m_IngredientUseSoftSkills[1] = false;// set 'true' to allow modification of the values by softskills on this i
■■//----------------------------------------------------------------------------------------------------------------------
■■
■■//result1
■■/*
■■AddResult("Mag_AKM_Palm30Rnd_");//add results here

■■m_ResultSetFullQuantity[0] = false;//true = set full quantity, false = do nothing
■■m_ResultSetQuantity[0] = -1;//-1 = do nothing
■■m_ResultSetHealth[0] = -1;//-1 = do nothing
■■m_ResultInheritsHealth[0] = 1;// (value) == -1 means do nothing; a (value) >= 0 means this result will in
■■m_ResultInheritsColor[0] = 0;// (value) == -1 means do nothing; a (value) >= 0 means this result classn
■■m_ResultToInventory[0] = 1;//(value) == -2 spawn result on the ground;(value) == -1 place anywhere in
■■m_ResultUseSoftSkills[0] = false;// set 'true' to allow modification of the values by softskills on this resu
■■m_ResultReplacesIngredient[0] = 1;// value == -1 means do nothing; a value >= 0 means this result wil
■■*/
■}
```

```
// ---------------------------------------------
// ------ PaintAKPlasticBttstck.c - START --------------------
// ---------------------------------------------


class PaintAKPlasticBttstck extends RecipeBase
{■
■override void Init()
■{
■■m_Name = "#STR_paint0";
■■m_IsInstaRecipe = false;//should this recipe be performed instantly without animation
■■m_AnimationLength = 1.5;//animation length in relative time units
■■m_Specialty = 0.01;// value > 0 for roughness, value < 0 for precision
■■
■■
■■//conditions
■■m_MinDamageIngredient[0] = -1;//-1 = disable check
■■m_MaxDamageIngredient[0] = 3;//-1 = disable check
■■
■■m_MinQuantityIngredient[0] = 25;//-1 = disable check
■■m_MaxQuantityIngredient[0] = -1;//-1 = disable check
■■
■■m_MinDamageIngredient[1] = -1;//-1 = disable check
■■m_MaxDamageIngredient[1] = 3;//-1 = disable check
■■
■■m_MinQuantityIngredient[1] = -1;//-1 = disable check
■■m_MaxQuantityIngredient[1] = -1;//-1 = disable check
■■//----------------------------------------------------------------------------------------------------------------------
■■
■■//INGREDIENTS
■■//ingredient 1
■■InsertIngredient(0,"Spraycan_ColorBase");//you can insert multiple ingredients this way
■■
■■m_IngredientAddHealth[0] = 0;// 0 = do nothing
■■m_IngredientSetHealth[0] = -1; // -1 = do nothing
■■m_IngredientAddQuantity[0] = -25;// 0 = do nothing
■■m_IngredientDestroy[0] = false;//true = destroy, false = do nothing
■■m_IngredientUseSoftSkills[0] = true;// set 'true' to allow modification of the values by softskills on this in
■■
■■//ingredient 2
■■InsertIngredient(1,"AK_PlasticBttstck");//you can insert multiple ingredients this way
■■
■■m_IngredientAddHealth[1] = 0;// 0 = do nothing
■■m_IngredientSetHealth[1] = -1; // -1 = do nothing
■■m_IngredientAddQuantity[1] = 0;// 0 = do nothing
■■m_IngredientDestroy[1] = false;// false = do nothing
■■m_IngredientUseSoftSkills[1] = false;// set 'true' to allow modification of the values by softskills on this i
■■//----------------------------------------------------------------------------------------------------------------------
■■
■■//result1
■■/*
■■AddResult("AK_PlasticBttstck_");//add results here

■■m_ResultSetFullQuantity[0] = false;//true = set full quantity, false = do nothing
■■m_ResultSetQuantity[0] = -1;//-1 = do nothing
■■m_ResultSetHealth[0] = -1;//-1 = do nothing
■■m_ResultInheritsHealth[0] = 1;// (value) == -1 means do nothing; a (value) >= 0 means this result will in
■■m_ResultInheritsColor[0] = 0;// (value) == -1 means do nothing; a (value) >= 0 means this result classn
■■m_ResultToInventory[0] = 1;//(value) == -2 spawn result on the ground;(value) == -1 place anywhere in
■■m_ResultUseSoftSkills[0] = false;// set 'true' to allow modification of the values by softskills on this resu
■■m_ResultReplacesIngredient[0] = -1;// value == -1 means do nothing; a value >= 0 means this result wi
■■*/
■}
```

```
// ---------------------------------------------
// ------ PaintAKRailHndgrd.c - START --------------------
// ---------------------------------------------


class PaintAKRailHndgrd extends RecipeBase
{■
■override void Init()
■{
■■m_Name = "#STR_paint0";
■■m_IsInstaRecipe = false;//should this recipe be performed instantly without animation
■■m_AnimationLength = 1.5;//animation length in relative time units
■■m_Specialty = 0.01;// value > 0 for roughness, value < 0 for precision
■■
■■
■■//conditions
■■m_MinDamageIngredient[0] = -1;//-1 = disable check
■■m_MaxDamageIngredient[0] = 3;//-1 = disable check
■■
■■m_MinQuantityIngredient[0] = 25;//-1 = disable check
■■m_MaxQuantityIngredient[0] = -1;//-1 = disable check
■■
■■m_MinDamageIngredient[1] = -1;//-1 = disable check
■■m_MaxDamageIngredient[1] = 3;//-1 = disable check
■■
■■m_MinQuantityIngredient[1] = -1;//-1 = disable check
■■m_MaxQuantityIngredient[1] = -1;//-1 = disable check
■■//----------------------------------------------------------------------------------------------------------------------
■■
■■//INGREDIENTS
■■//ingredient 1
■■InsertIngredient(0,"Spraycan_ColorBase");//you can insert multiple ingredients this way
■■
■■m_IngredientAddHealth[0] = 0;// 0 = do nothing
■■m_IngredientSetHealth[0] = -1; // -1 = do nothing
■■m_IngredientAddQuantity[0] = -25;// 0 = do nothing
■■m_IngredientDestroy[0] = false;//true = destroy, false = do nothing
■■m_IngredientUseSoftSkills[0] = true;// set 'true' to allow modification of the values by softskills on this in
■■
■■//ingredient 2
■■InsertIngredient(1,"AK_RailHndgrd");//you can insert multiple ingredients this way
■■
■■m_IngredientAddHealth[1] = 0;// 0 = do nothing
■■m_IngredientSetHealth[1] = -1; // -1 = do nothing
■■m_IngredientAddQuantity[1] = 0;// 0 = do nothing
■■m_IngredientDestroy[1] = false;// false = do nothing
■■m_IngredientUseSoftSkills[1] = false;// set 'true' to allow modification of the values by softskills on this i
■■//----------------------------------------------------------------------------------------------------------------------
■■
■■//result1
■■/*
■■AddResult("AK_RailHndgrd_");//add results here

■■m_ResultSetFullQuantity[0] = false;//true = set full quantity, false = do nothing
■■m_ResultSetQuantity[0] = -1;//-1 = do nothing
■■m_ResultSetHealth[0] = -1;//-1 = do nothing
■■m_ResultInheritsHealth[0] = 1;// (value) == -1 means do nothing; a (value) >= 0 means this result will in
■■m_ResultInheritsColor[0] = 0;// (value) == -1 means do nothing; a (value) >= 0 means this result classn
■■m_ResultToInventory[0] = 1;//(value) == -2 spawn result on the ground;(value) == -1 place anywhere in
■■m_ResultUseSoftSkills[0] = false;// set 'true' to allow modification of the values by softskills on this resu
■■m_ResultReplacesIngredient[0] = -1;// value == -1 means do nothing; a value >= 0 means this result wi
■■*/
■}
```

```
// ---------------------------------------------
// ------ PaintAKS74U.c - START --------------------
// ---------------------------------------------


class PaintAKS74U extends RecipeBase
{■
■override void Init()
■{
■■m_Name = "#STR_paint0";
■■m_IsInstaRecipe = false;//should this recipe be performed instantly without animation
■■m_AnimationLength = 1.5;//animation length in relative time units
■■m_Specialty = 0.01;// value > 0 for roughness, value < 0 for precision
■■
■■
■■//conditions
■■m_MinDamageIngredient[0] = -1;//-1 = disable check
■■m_MaxDamageIngredient[0] = 3;//-1 = disable check
■■
■■m_MinQuantityIngredient[0] = 25;//-1 = disable check
■■m_MaxQuantityIngredient[0] = -1;//-1 = disable check
■■
■■m_MinDamageIngredient[1] = -1;//-1 = disable check
■■m_MaxDamageIngredient[1] = 3;//-1 = disable check
■■
■■m_MinQuantityIngredient[1] = -1;//-1 = disable check
■■m_MaxQuantityIngredient[1] = -1;//-1 = disable check
■■//----------------------------------------------------------------------------------------------------------------------
■■
■■//INGREDIENTS
■■//ingredient 1
■■InsertIngredient(0,"Spraycan_ColorBase");//you can insert multiple ingredients this way
■■
■■m_IngredientAddHealth[0] = 0;// 0 = do nothing
■■m_IngredientSetHealth[0] = -1; // -1 = do nothing
■■m_IngredientAddQuantity[0] = -25;// 0 = do nothing
■■m_IngredientDestroy[0] = false;//true = destroy, false = do nothing
■■m_IngredientUseSoftSkills[0] = true;// set 'true' to allow modification of the values by softskills on this in
■■
■■//ingredient 2
■■InsertIngredient(1,"AKS74U");//you can insert multiple ingredients this way
■■
■■m_IngredientAddHealth[1] = 0;// 0 = do nothing
■■m_IngredientSetHealth[1] = -1; // -1 = do nothing
■■m_IngredientAddQuantity[1] = 0;// 0 = do nothing
■■m_IngredientDestroy[1] = false;// false = do nothing
■■m_IngredientUseSoftSkills[1] = false;// set 'true' to allow modification of the values by softskills on this i
■■//----------------------------------------------------------------------------------------------------------------------
■■
■■//result1
■■/*
■■AddResult("AKS74U_");//add results here

■■m_ResultSetFullQuantity[0] = false;//true = set full quantity, false = do nothing
■■m_ResultSetQuantity[0] = -1;//-1 = do nothing
■■m_ResultSetHealth[0] = -1;//-1 = do nothing
■■m_ResultInheritsHealth[0] = 1;// (value) == -1 means do nothing; a (value) >= 0 means this result will in
■■m_ResultInheritsColor[0] = 0;// (value) == -1 means do nothing; a (value) >= 0 means this result classn
■■m_ResultToInventory[0] = 1;//(value) == -2 spawn result on the ground;(value) == -1 place anywhere in
■■m_ResultUseSoftSkills[0] = false;// set 'true' to allow modification of the values by softskills on this resu
■■m_ResultReplacesIngredient[0] = 1;// value == -1 means do nothing; a value >= 0 means this result wil
■■*/
■}
```

```
// ---------------------------------------------
// ------ PaintAKS74UBttstck.c - START --------------------
// ---------------------------------------------


class PaintAKS74UBttstck extends RecipeBase
{■
■override void Init()
■{
■■m_Name = "#STR_paint0";
■■m_IsInstaRecipe = false;//should this recipe be performed instantly without animation
■■m_AnimationLength = 1.5;//animation length in relative time units
■■m_Specialty = 0.01;// value > 0 for roughness, value < 0 for precision
■■
■■
■■//conditions
■■m_MinDamageIngredient[0] = -1;//-1 = disable check
■■m_MaxDamageIngredient[0] = 3;//-1 = disable check
■■
■■m_MinQuantityIngredient[0] = 25;//-1 = disable check
■■m_MaxQuantityIngredient[0] = -1;//-1 = disable check
■■
■■m_MinDamageIngredient[1] = -1;//-1 = disable check
■■m_MaxDamageIngredient[1] = 3;//-1 = disable check
■■
■■m_MinQuantityIngredient[1] = -1;//-1 = disable check
■■m_MaxQuantityIngredient[1] = -1;//-1 = disable check
■■//----------------------------------------------------------------------------------------------------------------------
■■
■■//INGREDIENTS
■■//ingredient 1
■■InsertIngredient(0,"Spraycan_ColorBase");//you can insert multiple ingredients this way
■■
■■m_IngredientAddHealth[0] = 0;// 0 = do nothing
■■m_IngredientSetHealth[0] = -1; // -1 = do nothing
■■m_IngredientAddQuantity[0] = -25;// 0 = do nothing
■■m_IngredientDestroy[0] = false;//true = destroy, false = do nothing
■■m_IngredientUseSoftSkills[0] = true;// set 'true' to allow modification of the values by softskills on this in
■■
■■//ingredient 2
■■InsertIngredient(1,"AKS74U_Bttstck");//you can insert multiple ingredients this way
■■
■■m_IngredientAddHealth[1] = 0;// 0 = do nothing
■■m_IngredientSetHealth[1] = -1; // -1 = do nothing
■■m_IngredientAddQuantity[1] = 0;// 0 = do nothing
■■m_IngredientDestroy[1] = false;// false = do nothing
■■m_IngredientUseSoftSkills[1] = false;// set 'true' to allow modification of the values by softskills on this i
■■//----------------------------------------------------------------------------------------------------------------------
■■
■■//result1
■■/*
■■AddResult("AKS74U_Bttstck_");//add results here

■■m_ResultSetFullQuantity[0] = false;//true = set full quantity, false = do nothing
■■m_ResultSetQuantity[0] = -1;//-1 = do nothing
■■m_ResultSetHealth[0] = -1;//-1 = do nothing
■■m_ResultInheritsHealth[0] = 1;// (value) == -1 means do nothing; a (value) >= 0 means this result will in
■■m_ResultInheritsColor[0] = 0;// (value) == -1 means do nothing; a (value) >= 0 means this result classn
■■m_ResultToInventory[0] = 1;//(value) == -2 spawn result on the ground;(value) == -1 place anywhere in
■■m_ResultUseSoftSkills[0] = false;// set 'true' to allow modification of the values by softskills on this resu
■■m_ResultReplacesIngredient[0] = -1;// value == -1 means do nothing; a value >= 0 means this result wi
■■*/
■}
```

```
// -------------------------------------------
// ------ PaintAKWoodBttstckBlack.c - START -------------------
// -------------------------------------------


class PaintAKWoodBttstckBlack extends RecipeBase
{■
■override void Init()
■{
■■m_Name = "#STR_paint0";
■■m_IsInstaRecipe = false;//should this recipe be performed instantly without animation
■■m_AnimationLength = 1.5;//animation length in relative time units
■■m_Specialty = 0.01;// value > 0 for roughness, value < 0 for precision
■■
■■
■■//conditions
■■m_MinDamageIngredient[0] = -1;//-1 = disable check
■■m_MaxDamageIngredient[0] = 3;//-1 = disable check
■■
■■m_MinQuantityIngredient[0] = 25;//-1 = disable check
■■m_MaxQuantityIngredient[0] = -1;//-1 = disable check
■■
■■m_MinDamageIngredient[1] = -1;//-1 = disable check
■■m_MaxDamageIngredient[1] = 3;//-1 = disable check
■■
■■m_MinQuantityIngredient[1] = -1;//-1 = disable check
■■m_MaxQuantityIngredient[1] = -1;//-1 = disable check
■■//---------------------------------------------------------------------------------------------------------------
■■
■■//INGREDIENTS
■■//ingredient 1
■■InsertIngredient(0,"Spraycan_Black");//you can insert multiple ingredients this way
■■
■■m_IngredientAddHealth[0] = 0;// 0 = do nothing
■■m_IngredientSetHealth[0] = -1; // -1 = do nothing
■■m_IngredientAddQuantity[0] = -25;// 0 = do nothing
■■m_IngredientDestroy[0] = false;//true = destroy, false = do nothing
■■m_IngredientUseSoftSkills[0] = true;// set 'true' to allow modification of the values by softskills on this in
■■
■■//ingredient 2
■■InsertIngredient(1,"AK_WoodBttstck");//you can insert multiple ingredients this way
■■
■■m_IngredientAddHealth[1] = 0;// 0 = do nothing
■■m_IngredientSetHealth[1] = -1; // -1 = do nothing
■■m_IngredientAddQuantity[1] = 0;// 0 = do nothing
■■m_IngredientDestroy[1] = false;// false = do nothing
■■m_IngredientUseSoftSkills[1] = false;// set 'true' to allow modification of the values by softskills on this i
■■//---------------------------------------------------------------------------------------------------------------
■■
■■//result1
■■/*
■■AddResult("AK_WoodBttstck_Black");//add results here

■■m_ResultSetFullQuantity[0] = false;//true = set full quantity, false = do nothing
■■m_ResultSetQuantity[0] = -1;//-1 = do nothing
■■m_ResultSetHealth[0] = -1;//-1 = do nothing
■■m_ResultInheritsHealth[0] = 1;// (value) == -1 means do nothing; a (value) >= 0 means this result will in
■■m_ResultInheritsColor[0] = -1;// (value) == -1 means do nothing; a (value) >= 0 means this result classn
■■m_ResultToInventory[0] = 1;//(value) == -2 spawn result on the ground;(value) == -1 place anywhere in
■■m_ResultUseSoftSkills[0] = false;// set 'true' to allow modification of the values by softskills on this resu
■■m_ResultReplacesIngredient[0] = -1;// value == -1 means do nothing; a value >= 0 means this result wi
■■*/
■}
```

```
// ---------------------------------------------
// ------ PaintAKWoodBttstckCamo.c - START --------------------
// ---------------------------------------------


class PaintAKWoodBttstckCamo extends RecipeBase
{■
■override void Init()
■{
■■m_Name = "#STR_paint0";
■■m_IsInstaRecipe = false;//should this recipe be performed instantly without animation
■■m_AnimationLength = 1.5;//animation length in relative time units
■■m_Specialty = 0.01;// value > 0 for roughness, value < 0 for precision
■■
■■
■■//conditions
■■m_MinDamageIngredient[0] = -1;//-1 = disable check
■■m_MaxDamageIngredient[0] = 3;//-1 = disable check
■■
■■m_MinQuantityIngredient[0] = 25;//-1 = disable check
■■m_MaxQuantityIngredient[0] = -1;//-1 = disable check
■■
■■m_MinDamageIngredient[1] = -1;//-1 = disable check
■■m_MaxDamageIngredient[1] = 3;//-1 = disable check
■■
■■m_MinQuantityIngredient[1] = -1;//-1 = disable check
■■m_MaxQuantityIngredient[1] = -1;//-1 = disable check
■■//----------------------------------------------------------------------------------------------------------------------
■■
■■//INGREDIENTS
■■//ingredient 1
■■InsertIngredient(0,"Spraycan_Green");//you can insert multiple ingredients this way
■■
■■m_IngredientAddHealth[0] = 0;// 0 = do nothing
■■m_IngredientSetHealth[0] = -1; // -1 = do nothing
■■m_IngredientAddQuantity[0] = -25;// 0 = do nothing
■■m_IngredientDestroy[0] = false;//true = destroy, false = do nothing
■■m_IngredientUseSoftSkills[0] = true;// set 'true' to allow modification of the values by softskills on this in
■■
■■//ingredient 2
■■InsertIngredient(1,"AK_WoodBttstck");//you can insert multiple ingredients this way
■■
■■m_IngredientAddHealth[1] = 0;// 0 = do nothing
■■m_IngredientSetHealth[1] = -1; // -1 = do nothing
■■m_IngredientAddQuantity[1] = 0;// 0 = do nothing
■■m_IngredientDestroy[1] = false;// false = do nothing
■■m_IngredientUseSoftSkills[1] = false;// set 'true' to allow modification of the values by softskills on this i
■■//----------------------------------------------------------------------------------------------------------------------
■■
■■//result1
■■/*
■■AddResult("AK_WoodBttstck_Camo");//add results here

■■m_ResultSetFullQuantity[0] = false;//true = set full quantity, false = do nothing
■■m_ResultSetQuantity[0] = -1;//-1 = do nothing
■■m_ResultSetHealth[0] = -1;//-1 = do nothing
■■m_ResultInheritsHealth[0] = 1;// (value) == -1 means do nothing; a (value) >= 0 means this result will in
■■m_ResultInheritsColor[0] = -1;// (value) == -1 means do nothing; a (value) >= 0 means this result classn
■■m_ResultToInventory[0] = 1;//(value) == -2 spawn result on the ground;(value) == -1 place anywhere in
■■m_ResultUseSoftSkills[0] = false;// set 'true' to allow modification of the values by softskills on this resu
■■m_ResultReplacesIngredient[0] = -1;// value == -1 means do nothing; a value >= 0 means this result wi
■■*/
■}
```

```
// ---------------------------------------------
// ------ PaintB95.c - START --------------------
// ---------------------------------------------


class PaintB95 extends RecipeBase
{■
■override void Init()
■{
■■m_Name = "#STR_paint0";
■■m_IsInstaRecipe = false;//should this recipe be performed instantly without animation
■■m_AnimationLength = 1.5;//animation length in relative time units
■■m_Specialty = 0.01;// value > 0 for roughness, value < 0 for precision
■■
■■
■■//conditions
■■m_MinDamageIngredient[0] = -1;//-1 = disable check
■■m_MaxDamageIngredient[0] = 3;//-1 = disable check
■■
■■m_MinQuantityIngredient[0] = 25;//-1 = disable check
■■m_MaxQuantityIngredient[0] = -1;//-1 = disable check
■■
■■m_MinDamageIngredient[1] = -1;//-1 = disable check
■■m_MaxDamageIngredient[1] = 3;//-1 = disable check
■■
■■m_MinQuantityIngredient[1] = -1;//-1 = disable check
■■m_MaxQuantityIngredient[1] = -1;//-1 = disable check
■■//----------------------------------------------------------------------------------------------------------------------
■■
■■//INGREDIENTS
■■//ingredient 1
■■InsertIngredient(0,"Spraycan_ColorBase");//you can insert multiple ingredients this way
■■
■■m_IngredientAddHealth[0] = 0;// 0 = do nothing
■■m_IngredientSetHealth[0] = -1; // -1 = do nothing
■■m_IngredientAddQuantity[0] = -25;// 0 = do nothing
■■m_IngredientDestroy[0] = false;//true = destroy, false = do nothing
■■m_IngredientUseSoftSkills[0] = true;// set 'true' to allow modification of the values by softskills on this in
■■
■■//ingredient 2
■■InsertIngredient(1,"B95");//you can insert multiple ingredients this way
■■
■■m_IngredientAddHealth[1] = 0;// 0 = do nothing
■■m_IngredientSetHealth[1] = -1; // -1 = do nothing
■■m_IngredientAddQuantity[1] = 0;// 0 = do nothing
■■m_IngredientDestroy[1] = false;// false = do nothing
■■m_IngredientUseSoftSkills[1] = false;// set 'true' to allow modification of the values by softskills on this i
■■//----------------------------------------------------------------------------------------------------------------------
■■
■■//result1
■■/*
■■AddResult("B95_");//add results here

■■m_ResultSetFullQuantity[0] = false;//true = set full quantity, false = do nothing
■■m_ResultSetQuantity[0] = -1;//-1 = do nothing
■■m_ResultSetHealth[0] = -1;//-1 = do nothing
■■m_ResultInheritsHealth[0] = 1;// (value) == -1 means do nothing; a (value) >= 0 means this result will in
■■m_ResultInheritsColor[0] = 0;// (value) == -1 means do nothing; a (value) >= 0 means this result classn
■■m_ResultToInventory[0] = 1;//(value) == -2 spawn result on the ground;(value) == -1 place anywhere in
■■m_ResultUseSoftSkills[0] = false;// set 'true' to allow modification of the values by softskills on this resu
■■m_ResultReplacesIngredient[0] = 1;// value == -1 means do nothing; a value >= 0 means this result wil
■■*/
■}
```

```
// ---------------------------------------------
// ------ PaintBallisticHelmet.c - START --------------------
// ---------------------------------------------


class PaintBallisticHelmet extends RecipeBase
{■
■override void Init()
■{
■■m_Name = "#STR_paint0";
■■m_IsInstaRecipe = false;//should this recipe be performed instantly without animation
■■m_AnimationLength = 1.5;//animation length in relative time units
■■m_Specialty = 0.01;// value > 0 for roughness, value < 0 for precision
■■
■■
■■//conditions
■■m_MinDamageIngredient[0] = -1;//-1 = disable check
■■m_MaxDamageIngredient[0] = 3;//-1 = disable check
■■
■■m_MinQuantityIngredient[0] = 25;//-1 = disable check
■■m_MaxQuantityIngredient[0] = -1;//-1 = disable check
■■
■■m_MinDamageIngredient[1] = -1;//-1 = disable check
■■m_MaxDamageIngredient[1] = 3;//-1 = disable check
■■
■■m_MinQuantityIngredient[1] = -1;//-1 = disable check
■■m_MaxQuantityIngredient[1] = -1;//-1 = disable check
■■//----------------------------------------------------------------------------------------------------------------------
■■
■■//INGREDIENTS
■■//ingredient 1
■■InsertIngredient(0,"Spraycan_ColorBase");//you can insert multiple ingredients this way
■■
■■m_IngredientAddHealth[0] = 0;// 0 = do nothing
■■m_IngredientSetHealth[0] = -1; // -1 = do nothing
■■m_IngredientAddQuantity[0] = -25;// 0 = do nothing
■■m_IngredientDestroy[0] = false;//true = destroy, false = do nothing
■■m_IngredientUseSoftSkills[0] = true;// set 'true' to allow modification of the values by softskills on this in
■■
■■//ingredient 2
■■InsertIngredient(1,"BallisticHelmet_ColorBase");//you can insert multiple ingredients this way
■■
■■m_IngredientAddHealth[1] = 0;// 0 = do nothing
■■m_IngredientSetHealth[1] = -1; // -1 = do nothing
■■m_IngredientAddQuantity[1] = 0;// 0 = do nothing
■■m_IngredientDestroy[1] = false;// false = do nothing
■■m_IngredientUseSoftSkills[1] = false;// set 'true' to allow modification of the values by softskills on this i
■■//----------------------------------------------------------------------------------------------------------------------
■■
■■//result1
■■/*
■■AddResult("BallisticHelmet_");//add results here

■■m_ResultSetFullQuantity[0] = false;//true = set full quantity, false = do nothing
■■m_ResultSetQuantity[0] = -1;//-1 = do nothing
■■m_ResultSetHealth[0] = -1;//-1 = do nothing
■■m_ResultInheritsHealth[0] = 1;// (value) == -1 means do nothing; a (value) >= 0 means this result will in
■■m_ResultInheritsColor[0] = 0;// (value) == -1 means do nothing; a (value) >= 0 means this result classn
■■m_ResultToInventory[0] = 1;//(value) == -2 spawn result on the ground;(value) == -1 place anywhere in
■■m_ResultUseSoftSkills[0] = false;// set 'true' to allow modification of the values by softskills on this resu
■■m_ResultReplacesIngredient[0] = -1;// value == -1 means do nothing; a value >= 0 means this result wi
■■*/
■}
```

```
// ---------------------------------------------
// ------ PaintCMAG10.c - START --------------------
// ---------------------------------------------


class PaintCMAG10 extends RecipeBase
{■
■override void Init()
■{
■■m_Name = "#STR_paint0";
■■m_IsInstaRecipe = false;//should this recipe be performed instantly without animation
■■m_AnimationLength = 1.5;//animation length in relative time units
■■m_Specialty = 0.01;// value > 0 for roughness, value < 0 for precision
■■
■■
■■//conditions
■■m_MinDamageIngredient[0] = -1;//-1 = disable check
■■m_MaxDamageIngredient[0] = 3;//-1 = disable check
■■
■■m_MinQuantityIngredient[0] = 25;//-1 = disable check
■■m_MaxQuantityIngredient[0] = -1;//-1 = disable check
■■
■■m_MinDamageIngredient[1] = -1;//-1 = disable check
■■m_MaxDamageIngredient[1] = 3;//-1 = disable check
■■
■■m_MinQuantityIngredient[1] = -1;//-1 = disable check
■■m_MaxQuantityIngredient[1] = -1;//-1 = disable check
■■//----------------------------------------------------------------------------------------------------------------------
■■
■■//INGREDIENTS
■■//ingredient 1
■■InsertIngredient(0,"Spraycan_ColorBase");//you can insert multiple ingredients this way
■■
■■m_IngredientAddHealth[0] = 0;// 0 = do nothing
■■m_IngredientSetHealth[0] = -1; // -1 = do nothing
■■m_IngredientAddQuantity[0] = -25;// 0 = do nothing
■■m_IngredientDestroy[0] = false;//true = destroy, false = do nothing
■■m_IngredientUseSoftSkills[0] = true;// set 'true' to allow modification of the values by softskills on this in
■■
■■//ingredient 2
■■InsertIngredient(1,"Mag_CMAG_10Rnd");//you can insert multiple ingredients this way
■■
■■m_IngredientAddHealth[1] = 0;// 0 = do nothing
■■m_IngredientSetHealth[1] = -1; // -1 = do nothing
■■m_IngredientAddQuantity[1] = 0;// 0 = do nothing
■■m_IngredientDestroy[1] = false;// false = do nothing
■■m_IngredientUseSoftSkills[1] = false;// set 'true' to allow modification of the values by softskills on this i
■■//----------------------------------------------------------------------------------------------------------------------
■■
■■//result1
■■/*
■■AddResult("Mag_CMAG_10Rnd_");//add results here

■■m_ResultSetFullQuantity[0] = false;//true = set full quantity, false = do nothing
■■m_ResultSetQuantity[0] = -1;//-1 = do nothing
■■m_ResultSetHealth[0] = -1;//-1 = do nothing
■■m_ResultInheritsHealth[0] = 1;// (value) == -1 means do nothing; a (value) >= 0 means this result will in
■■m_ResultInheritsColor[0] = 0;// (value) == -1 means do nothing; a (value) >= 0 means this result classn
■■m_ResultToInventory[0] = 1;//(value) == -2 spawn result on the ground;(value) == -1 place anywhere in
■■m_ResultUseSoftSkills[0] = false;// set 'true' to allow modification of the values by softskills on this resu
■■m_ResultReplacesIngredient[0] = 1;// value == -1 means do nothing; a value >= 0 means this result wil
■■*/
■}
```

```
// ---------------------------------------------
// ------ PaintCMAG20.c - START --------------------
// ---------------------------------------------


class PaintCMAG20 extends RecipeBase
{■
■override void Init()
■{
■■m_Name = "#STR_paint0";
■■m_IsInstaRecipe = false;//should this recipe be performed instantly without animation
■■m_AnimationLength = 1.5;//animation length in relative time units
■■m_Specialty = 0.01;// value > 0 for roughness, value < 0 for precision
■■
■■
■■//conditions
■■m_MinDamageIngredient[0] = -1;//-1 = disable check
■■m_MaxDamageIngredient[0] = 3;//-1 = disable check
■■
■■m_MinQuantityIngredient[0] = 25;//-1 = disable check
■■m_MaxQuantityIngredient[0] = -1;//-1 = disable check
■■
■■m_MinDamageIngredient[1] = -1;//-1 = disable check
■■m_MaxDamageIngredient[1] = 3;//-1 = disable check
■■
■■m_MinQuantityIngredient[1] = -1;//-1 = disable check
■■m_MaxQuantityIngredient[1] = -1;//-1 = disable check
■■//----------------------------------------------------------------------------------------------------------------------
■■
■■//INGREDIENTS
■■//ingredient 1
■■InsertIngredient(0,"Spraycan_ColorBase");//you can insert multiple ingredients this way
■■
■■m_IngredientAddHealth[0] = 0;// 0 = do nothing
■■m_IngredientSetHealth[0] = -1; // -1 = do nothing
■■m_IngredientAddQuantity[0] = -25;// 0 = do nothing
■■m_IngredientDestroy[0] = false;//true = destroy, false = do nothing
■■m_IngredientUseSoftSkills[0] = true;// set 'true' to allow modification of the values by softskills on this in
■■
■■//ingredient 2
■■InsertIngredient(1,"Mag_CMAG_20Rnd");//you can insert multiple ingredients this way
■■
■■m_IngredientAddHealth[1] = 0;// 0 = do nothing
■■m_IngredientSetHealth[1] = -1; // -1 = do nothing
■■m_IngredientAddQuantity[1] = 0;// 0 = do nothing
■■m_IngredientDestroy[1] = false;// false = do nothing
■■m_IngredientUseSoftSkills[1] = false;// set 'true' to allow modification of the values by softskills on this i
■■//----------------------------------------------------------------------------------------------------------------------
■■
■■//result1
■■/*
■■AddResult("Mag_CMAG_20Rnd_");//add results here

■■m_ResultSetFullQuantity[0] = false;//true = set full quantity, false = do nothing
■■m_ResultSetQuantity[0] = -1;//-1 = do nothing
■■m_ResultSetHealth[0] = -1;//-1 = do nothing
■■m_ResultInheritsHealth[0] = 1;// (value) == -1 means do nothing; a (value) >= 0 means this result will in
■■m_ResultInheritsColor[0] = 0;// (value) == -1 means do nothing; a (value) >= 0 means this result classn
■■m_ResultToInventory[0] = 1;//(value) == -2 spawn result on the ground;(value) == -1 place anywhere in
■■m_ResultUseSoftSkills[0] = false;// set 'true' to allow modification of the values by softskills on this resu
■■m_ResultReplacesIngredient[0] = 1;// value == -1 means do nothing; a value >= 0 means this result wil
■■*/
■}
```

```
// ---------------------------------------------
// ------ PaintCMAG30.c - START --------------------
// ---------------------------------------------


class PaintCMAG30 extends RecipeBase
{■
■override void Init()
■{
■■m_Name = "#STR_paint0";
■■m_IsInstaRecipe = false;//should this recipe be performed instantly without animation
■■m_AnimationLength = 1.5;//animation length in relative time units
■■m_Specialty = 0.01;// value > 0 for roughness, value < 0 for precision
■■
■■
■■//conditions
■■m_MinDamageIngredient[0] = -1;//-1 = disable check
■■m_MaxDamageIngredient[0] = 3;//-1 = disable check
■■
■■m_MinQuantityIngredient[0] = 25;//-1 = disable check
■■m_MaxQuantityIngredient[0] = -1;//-1 = disable check
■■
■■m_MinDamageIngredient[1] = -1;//-1 = disable check
■■m_MaxDamageIngredient[1] = 3;//-1 = disable check
■■
■■m_MinQuantityIngredient[1] = -1;//-1 = disable check
■■m_MaxQuantityIngredient[1] = -1;//-1 = disable check
■■//----------------------------------------------------------------------------------------------------------------------
■■
■■//INGREDIENTS
■■//ingredient 1
■■InsertIngredient(0,"Spraycan_ColorBase");//you can insert multiple ingredients this way
■■
■■m_IngredientAddHealth[0] = 0;// 0 = do nothing
■■m_IngredientSetHealth[0] = -1; // -1 = do nothing
■■m_IngredientAddQuantity[0] = -25;// 0 = do nothing
■■m_IngredientDestroy[0] = false;//true = destroy, false = do nothing
■■m_IngredientUseSoftSkills[0] = true;// set 'true' to allow modification of the values by softskills on this in
■■
■■//ingredient 2
■■InsertIngredient(1,"Mag_CMAG_30Rnd");//you can insert multiple ingredients this way
■■
■■m_IngredientAddHealth[1] = 0;// 0 = do nothing
■■m_IngredientSetHealth[1] = -1; // -1 = do nothing
■■m_IngredientAddQuantity[1] = 0;// 0 = do nothing
■■m_IngredientDestroy[1] = false;// false = do nothing
■■m_IngredientUseSoftSkills[1] = false;// set 'true' to allow modification of the values by softskills on this i
■■//----------------------------------------------------------------------------------------------------------------------
■■
■■//result1
■■/*
■■AddResult("Mag_CMAG_30Rnd_");//add results here

■■m_ResultSetFullQuantity[0] = false;//true = set full quantity, false = do nothing
■■m_ResultSetQuantity[0] = -1;//-1 = do nothing
■■m_ResultSetHealth[0] = -1;//-1 = do nothing
■■m_ResultInheritsHealth[0] = 1;// (value) == -1 means do nothing; a (value) >= 0 means this result will in
■■m_ResultInheritsColor[0] = 0;// (value) == -1 means do nothing; a (value) >= 0 means this result classn
■■m_ResultToInventory[0] = 1;//(value) == -2 spawn result on the ground;(value) == -1 place anywhere in
■■m_ResultUseSoftSkills[0] = false;// set 'true' to allow modification of the values by softskills on this resu
■■m_ResultReplacesIngredient[0] = 1;// value == -1 means do nothing; a value >= 0 means this result wil
■■*/
■}
```

```
// ---------------------------------------------
// ------ PaintCMAG40.c - START --------------------
// ---------------------------------------------


class PaintCMAG40 extends RecipeBase
{■
■override void Init()
■{
■■m_Name = "#STR_paint0";
■■m_IsInstaRecipe = false;//should this recipe be performed instantly without animation
■■m_AnimationLength = 1.5;//animation length in relative time units
■■m_Specialty = 0.01;// value > 0 for roughness, value < 0 for precision
■■
■■
■■//conditions
■■m_MinDamageIngredient[0] = -1;//-1 = disable check
■■m_MaxDamageIngredient[0] = 3;//-1 = disable check
■■
■■m_MinQuantityIngredient[0] = 25;//-1 = disable check
■■m_MaxQuantityIngredient[0] = -1;//-1 = disable check
■■
■■m_MinDamageIngredient[1] = -1;//-1 = disable check
■■m_MaxDamageIngredient[1] = 3;//-1 = disable check
■■
■■m_MinQuantityIngredient[1] = -1;//-1 = disable check
■■m_MaxQuantityIngredient[1] = -1;//-1 = disable check
■■//----------------------------------------------------------------------------------------------------------------------
■■
■■//INGREDIENTS
■■//ingredient 1
■■InsertIngredient(0,"Spraycan_ColorBase");//you can insert multiple ingredients this way
■■
■■m_IngredientAddHealth[0] = 0;// 0 = do nothing
■■m_IngredientSetHealth[0] = -1; // -1 = do nothing
■■m_IngredientAddQuantity[0] = -25;// 0 = do nothing
■■m_IngredientDestroy[0] = false;//true = destroy, false = do nothing
■■m_IngredientUseSoftSkills[0] = true;// set 'true' to allow modification of the values by softskills on this in
■■
■■//ingredient 2
■■InsertIngredient(1,"Mag_CMAG_40Rnd");//you can insert multiple ingredients this way
■■
■■m_IngredientAddHealth[1] = 0;// 0 = do nothing
■■m_IngredientSetHealth[1] = -1; // -1 = do nothing
■■m_IngredientAddQuantity[1] = 0;// 0 = do nothing
■■m_IngredientDestroy[1] = false;// false = do nothing
■■m_IngredientUseSoftSkills[1] = false;// set 'true' to allow modification of the values by softskills on this i
■■//----------------------------------------------------------------------------------------------------------------------
■■
■■//result1
■■/*
■■AddResult("Mag_CMAG_40Rnd_");//add results here

■■m_ResultSetFullQuantity[0] = false;//true = set full quantity, false = do nothing
■■m_ResultSetQuantity[0] = -1;//-1 = do nothing
■■m_ResultSetHealth[0] = -1;//-1 = do nothing
■■m_ResultInheritsHealth[0] = 1;// (value) == -1 means do nothing; a (value) >= 0 means this result will in
■■m_ResultInheritsColor[0] = 0;// (value) == -1 means do nothing; a (value) >= 0 means this result classn
■■m_ResultToInventory[0] = 1;//(value) == -2 spawn result on the ground;(value) == -1 place anywhere in
■■m_ResultUseSoftSkills[0] = false;// set 'true' to allow modification of the values by softskills on this resu
■■m_ResultReplacesIngredient[0] = 1;// value == -1 means do nothing; a value >= 0 means this result wil
■■*/
■}
```

```
// ---------------------------------------------
// ------ PaintCz527.c - START --------------------
// ---------------------------------------------


class PaintCz527 extends RecipeBase
{■
■override void Init()
■{
■■m_Name = "#STR_paint0";
■■m_IsInstaRecipe = false;//should this recipe be performed instantly without animation
■■m_AnimationLength = 1.5;//animation length in relative time units
■■m_Specialty = 0.01;// value > 0 for roughness, value < 0 for precision
■■
■■
■■//conditions
■■m_MinDamageIngredient[0] = -1;//-1 = disable check
■■m_MaxDamageIngredient[0] = 3;//-1 = disable check
■■
■■m_MinQuantityIngredient[0] = 25;//-1 = disable check
■■m_MaxQuantityIngredient[0] = -1;//-1 = disable check
■■
■■m_MinDamageIngredient[1] = -1;//-1 = disable check
■■m_MaxDamageIngredient[1] = 3;//-1 = disable check
■■
■■m_MinQuantityIngredient[1] = -1;//-1 = disable check
■■m_MaxQuantityIngredient[1] = -1;//-1 = disable check
■■//----------------------------------------------------------------------------------------------------------------------
■■
■■//INGREDIENTS
■■//ingredient 1
■■InsertIngredient(0,"Spraycan_ColorBase");//you can insert multiple ingredients this way
■■
■■m_IngredientAddHealth[0] = 0;// 0 = do nothing
■■m_IngredientSetHealth[0] = -1; // -1 = do nothing
■■m_IngredientAddQuantity[0] = -25;// 0 = do nothing
■■m_IngredientDestroy[0] = false;//true = destroy, false = do nothing
■■m_IngredientUseSoftSkills[0] = true;// set 'true' to allow modification of the values by softskills on this in
■■
■■//ingredient 2
■■InsertIngredient(1,"CZ527");//you can insert multiple ingredients this way
■■
■■m_IngredientAddHealth[1] = 0;// 0 = do nothing
■■m_IngredientSetHealth[1] = -1; // -1 = do nothing
■■m_IngredientAddQuantity[1] = 0;// 0 = do nothing
■■m_IngredientDestroy[1] = false;// false = do nothing
■■m_IngredientUseSoftSkills[1] = false;// set 'true' to allow modification of the values by softskills on this i
■■//----------------------------------------------------------------------------------------------------------------------
■■
■■//result1
■■/*
■■AddResult("CZ527_");//add results here

■■m_ResultSetFullQuantity[0] = false;//true = set full quantity, false = do nothing
■■m_ResultSetQuantity[0] = -1;//-1 = do nothing
■■m_ResultSetHealth[0] = -1;//-1 = do nothing
■■m_ResultInheritsHealth[0] = 1;// (value) == -1 means do nothing; a (value) >= 0 means this result will in
■■m_ResultInheritsColor[0] = 0;// (value) == -1 means do nothing; a (value) >= 0 means this result classn
■■m_ResultToInventory[0] = 1;//(value) == -2 spawn result on the ground;(value) == -1 place anywhere in
■■m_ResultUseSoftSkills[0] = false;// set 'true' to allow modification of the values by softskills on this resu
■■m_ResultReplacesIngredient[0] = 1;// value == -1 means do nothing; a value >= 0 means this result wil
■■*/
■}
```

```
// ---------------------------------------------
// ------ PaintCz527CamoBlack.c - START --------------------
// ---------------------------------------------


class PaintCz527CamoBlack extends RecipeBase
{■
■override void Init()
■{
■■m_Name = "#STR_paint0";
■■m_IsInstaRecipe = false;//should this recipe be performed instantly without animation
■■m_AnimationLength = 1.5;//animation length in relative time units
■■m_Specialty = 0.01;// value > 0 for roughness, value < 0 for precision
■■
■■
■■//conditions
■■m_MinDamageIngredient[0] = -1;//-1 = disable check
■■m_MaxDamageIngredient[0] = 3;//-1 = disable check
■■
■■m_MinQuantityIngredient[0] = 25;//-1 = disable check
■■m_MaxQuantityIngredient[0] = -1;//-1 = disable check
■■
■■m_MinDamageIngredient[1] = -1;//-1 = disable check
■■m_MaxDamageIngredient[1] = 3;//-1 = disable check
■■
■■m_MinQuantityIngredient[1] = -1;//-1 = disable check
■■m_MaxQuantityIngredient[1] = -1;//-1 = disable check
■■//----------------------------------------------------------------------------------------------------------------------
■■
■■//INGREDIENTS
■■//ingredient 1
■■InsertIngredient(0,"Spraycan_Black");//you can insert multiple ingredients this way
■■
■■m_IngredientAddHealth[0] = 0;// 0 = do nothing
■■m_IngredientSetHealth[0] = -1; // -1 = do nothing
■■m_IngredientAddQuantity[0] = -25;// 0 = do nothing
■■m_IngredientDestroy[0] = false;//true = destroy, false = do nothing
■■m_IngredientUseSoftSkills[0] = true;// set 'true' to allow modification of the values by softskills on this in
■■
■■//ingredient 2
■■InsertIngredient(1,"CZ527_Green");//you can insert multiple ingredients this way
■■
■■m_IngredientAddHealth[1] = 0;// 0 = do nothing
■■m_IngredientSetHealth[1] = -1; // -1 = do nothing
■■m_IngredientAddQuantity[1] = 0;// 0 = do nothing
■■m_IngredientDestroy[1] = false;// false = do nothing
■■m_IngredientUseSoftSkills[1] = false;// set 'true' to allow modification of the values by softskills on this i
■■//----------------------------------------------------------------------------------------------------------------------
■■
■■//result1
■■/*
■■AddResult("CZ527_Camo");//add results here

■■m_ResultSetFullQuantity[0] = false;//true = set full quantity, false = do nothing
■■m_ResultSetQuantity[0] = -1;//-1 = do nothing
■■m_ResultSetHealth[0] = -1;//-1 = do nothing
■■m_ResultInheritsHealth[0] = 1;// (value) == -1 means do nothing; a (value) >= 0 means this result will in
■■m_ResultInheritsColor[0] = -1;// (value) == -1 means do nothing; a (value) >= 0 means this result classn
■■m_ResultToInventory[0] = 1;//(value) == -2 spawn result on the ground;(value) == -1 place anywhere in
■■m_ResultUseSoftSkills[0] = false;// set 'true' to allow modification of the values by softskills on this resu
■■m_ResultReplacesIngredient[0] = 1;// value == -1 means do nothing; a value >= 0 means this result wil
■■*/
■}
```

```
// ---------------------------------------------
// ------ PaintCz527CamoGreen.c - START --------------------
// ---------------------------------------------


class PaintCz527CamoGreen extends RecipeBase
{■
■override void Init()
■{
■■m_Name = "#STR_paint0";
■■m_IsInstaRecipe = false;//should this recipe be performed instantly without animation
■■m_AnimationLength = 1.5;//animation length in relative time units
■■m_Specialty = 0.01;// value > 0 for roughness, value < 0 for precision
■■
■■
■■//conditions
■■m_MinDamageIngredient[0] = -1;//-1 = disable check
■■m_MaxDamageIngredient[0] = 3;//-1 = disable check
■■
■■m_MinQuantityIngredient[0] = 25;//-1 = disable check
■■m_MaxQuantityIngredient[0] = -1;//-1 = disable check
■■
■■m_MinDamageIngredient[1] = -1;//-1 = disable check
■■m_MaxDamageIngredient[1] = 3;//-1 = disable check
■■
■■m_MinQuantityIngredient[1] = -1;//-1 = disable check
■■m_MaxQuantityIngredient[1] = -1;//-1 = disable check
■■//----------------------------------------------------------------------------------------------------------------------
■■
■■//INGREDIENTS
■■//ingredient 1
■■InsertIngredient(0,"Spraycan_Green");//you can insert multiple ingredients this way
■■
■■m_IngredientAddHealth[0] = 0;// 0 = do nothing
■■m_IngredientSetHealth[0] = -1; // -1 = do nothing
■■m_IngredientAddQuantity[0] = -25;// 0 = do nothing
■■m_IngredientDestroy[0] = false;//true = destroy, false = do nothing
■■m_IngredientUseSoftSkills[0] = true;// set 'true' to allow modification of the values by softskills on this in
■■
■■//ingredient 2
■■InsertIngredient(1,"CZ527_Black");//you can insert multiple ingredients this way
■■
■■m_IngredientAddHealth[1] = 0;// 0 = do nothing
■■m_IngredientSetHealth[1] = -1; // -1 = do nothing
■■m_IngredientAddQuantity[1] = 0;// 0 = do nothing
■■m_IngredientDestroy[1] = false;// false = do nothing
■■m_IngredientUseSoftSkills[1] = false;// set 'true' to allow modification of the values by softskills on this i
■■//----------------------------------------------------------------------------------------------------------------------
■■
■■//result1
■■/*
■■AddResult("CZ527_Camo");//add results here

■■m_ResultSetFullQuantity[0] = false;//true = set full quantity, false = do nothing
■■m_ResultSetQuantity[0] = -1;//-1 = do nothing
■■m_ResultSetHealth[0] = -1;//-1 = do nothing
■■m_ResultInheritsHealth[0] = 1;// (value) == -1 means do nothing; a (value) >= 0 means this result will in
■■m_ResultInheritsColor[0] = -1;// (value) == -1 means do nothing; a (value) >= 0 means this result class
■■m_ResultToInventory[0] = 1;//(value) == -2 spawn result on the ground;(value) == -1 place anywhere in
■■m_ResultUseSoftSkills[0] = false;// set 'true' to allow modification of the values by softskills on this resu
■■m_ResultReplacesIngredient[0] = 1;// value == -1 means do nothing; a value >= 0 means this result wil
■■*/
■}
```

```
// ---------------------------------------------
// ------ PaintDarkMotohelmet.c - START --------------------
// ---------------------------------------------


class PaintDarkMotohelmet extends RecipeBase
{■
■override void Init()
■{
■■m_Name = "#STR_paint0";
■■m_IsInstaRecipe = false;//should this recipe be performed instantly without animation
■■m_AnimationLength = 1.5;//animation length in relative time units
■■m_Specialty = 0.01;// value > 0 for roughness, value < 0 for precision
■■
■■
■■//conditions
■■m_MinDamageIngredient[0] = -1;//-1 = disable check
■■m_MaxDamageIngredient[0] = 3;//-1 = disable check
■■
■■m_MinQuantityIngredient[0] = 25;//-1 = disable check
■■m_MaxQuantityIngredient[0] = -1;//-1 = disable check
■■
■■m_MinDamageIngredient[1] = -1;//-1 = disable check
■■m_MaxDamageIngredient[1] = 3;//-1 = disable check
■■
■■m_MinQuantityIngredient[1] = -1;//-1 = disable check
■■m_MaxQuantityIngredient[1] = -1;//-1 = disable check
■■//----------------------------------------------------------------------------------------------------------------------
■■
■■//INGREDIENTS
■■//ingredient 1
■■InsertIngredient(0,"Spraycan_ColorBase");//you can insert multiple ingredients this way
■■
■■m_IngredientAddHealth[0] = 0;// 0 = do nothing
■■m_IngredientSetHealth[0] = -1; // -1 = do nothing
■■m_IngredientAddQuantity[0] = -25;// 0 = do nothing
■■m_IngredientDestroy[0] = false;//true = destroy, false = do nothing
■■m_IngredientUseSoftSkills[0] = true;// set 'true' to allow modification of the values by softskills on this in
■■
■■//ingredient 2
■■InsertIngredient(1,"DarkMotoHelmet_ColorBase");//you can insert multiple ingredients this way
■■
■■m_IngredientAddHealth[1] = 0;// 0 = do nothing
■■m_IngredientSetHealth[1] = -1; // -1 = do nothing
■■m_IngredientAddQuantity[1] = 0;// 0 = do nothing
■■m_IngredientDestroy[1] = false;// false = do nothing
■■m_IngredientUseSoftSkills[1] = false;// set 'true' to allow modification of the values by softskills on this i
■■//----------------------------------------------------------------------------------------------------------------------
■■
■■//result1
■■/*
■■AddResult("DarkMotoHelmet_");//add results here

■■m_ResultSetFullQuantity[0] = false;//true = set full quantity, false = do nothing
■■m_ResultSetQuantity[0] = -1;//-1 = do nothing
■■m_ResultSetHealth[0] = -1;//-1 = do nothing
■■m_ResultInheritsHealth[0] = 1;// (value) == -1 means do nothing; a (value) >= 0 means this result will in
■■m_ResultInheritsColor[0] = 0;// (value) == -1 means do nothing; a (value) >= 0 means this result classn
■■m_ResultToInventory[0] = 1;//(value) == -2 spawn result on the ground;(value) == -1 place anywhere in
■■m_ResultUseSoftSkills[0] = false;// set 'true' to allow modification of the values by softskills on this resu
■■m_ResultReplacesIngredient[0] = -1;// value == -1 means do nothing; a value >= 0 means this result wi
■■*/
■}
```

```
// ---------------------------------------------
// ------ PaintFirefighterAxe.c - START --------------------
// ---------------------------------------------


class PaintFirefighterAxe extends RecipeBase
{■
■override void Init()
■{
■■m_Name = "#STR_paint0";
■■m_IsInstaRecipe = false;//should this recipe be performed instantly without animation
■■m_AnimationLength = 1.5;//animation length in relative time units
■■m_Specialty = 0.01;// value > 0 for roughness, value < 0 for precision
■■
■■
■■//conditions
■■m_MinDamageIngredient[0] = -1;//-1 = disable check
■■m_MaxDamageIngredient[0] = 3;//-1 = disable check
■■
■■m_MinQuantityIngredient[0] = 25;//-1 = disable check
■■m_MaxQuantityIngredient[0] = -1;//-1 = disable check
■■
■■m_MinDamageIngredient[1] = -1;//-1 = disable check
■■m_MaxDamageIngredient[1] = 3;//-1 = disable check
■■
■■m_MinQuantityIngredient[1] = -1;//-1 = disable check
■■m_MaxQuantityIngredient[1] = -1;//-1 = disable check
■■//----------------------------------------------------------------------------------------------------------------------
■■
■■//INGREDIENTS
■■//ingredient 1
■■InsertIngredient(0,"Spraycan_ColorBase");//you can insert multiple ingredients this way
■■
■■m_IngredientAddHealth[0] = 0;// 0 = do nothing
■■m_IngredientSetHealth[0] = -1; // -1 = do nothing
■■m_IngredientAddQuantity[0] = -25;// 0 = do nothing
■■m_IngredientDestroy[0] = false;//true = destroy, false = do nothing
■■m_IngredientUseSoftSkills[0] = true;// set 'true' to allow modification of the values by softskills on this in
■■
■■//ingredient 2
■■InsertIngredient(1,"FirefighterAxe");//you can insert multiple ingredients this way
■■
■■m_IngredientAddHealth[1] = 0;// 0 = do nothing
■■m_IngredientSetHealth[1] = -1; // -1 = do nothing
■■m_IngredientAddQuantity[1] = 0;// 0 = do nothing
■■m_IngredientDestroy[1] = false;// false = do nothing
■■m_IngredientUseSoftSkills[1] = false;// set 'true' to allow modification of the values by softskills on this i
■■//----------------------------------------------------------------------------------------------------------------------
■■
■■//result1
■■/*
■■AddResult("FirefighterAxe_");//add results here

■■m_ResultSetFullQuantity[0] = false;//true = set full quantity, false = do nothing
■■m_ResultSetQuantity[0] = -1;//-1 = do nothing
■■m_ResultSetHealth[0] = -1;//-1 = do nothing
■■m_ResultInheritsHealth[0] = 1;// (value) == -1 means do nothing; a (value) >= 0 means this result will in
■■m_ResultInheritsColor[0] = 0;// (value) == -1 means do nothing; a (value) >= 0 means this result classn
■■m_ResultToInventory[0] = 1;//(value) == -2 spawn result on the ground;(value) == -1 place anywhere in
■■m_ResultUseSoftSkills[0] = false;// set 'true' to allow modification of the values by softskills on this resu
■■m_ResultReplacesIngredient[0] = -1;// value == -1 means do nothing; a value >= 0 means this result wi
■■*/
■}
```

```
// ---------------------------------------------
// ------ PaintGhillieAttBlack.c - START --------------------
// ---------------------------------------------


class PaintGhillieAttBlack extends RecipeBase
{
■override void Init()
■{
■■m_Name = "#STR_paint0";
■■m_IsInstaRecipe = false;//should this recipe be performed instantly without animation
■■m_AnimationLength = 1.5;//animation length in relative time units
■■m_Specialty = 0.01;// value > 0 for roughness, value < 0 for precision
■■
■■
■■//conditions
■■m_MinDamageIngredient[0] = -1;//-1 = disable check
■■m_MaxDamageIngredient[0] = 3;//-1 = disable check
■■
■■m_MinQuantityIngredient[0] = 40;//-1 = disable check
■■m_MaxQuantityIngredient[0] = -1;//-1 = disable check
■■
■■m_MinDamageIngredient[1] = -1;//-1 = disable check
■■m_MaxDamageIngredient[1] = 3;//-1 = disable check
■■
■■m_MinQuantityIngredient[1] = -1;//-1 = disable check
■■m_MaxQuantityIngredient[1] = -1;//-1 = disable check
■■//----------------------------------------------------------------------------------------------------------------------
■■
■■//INGREDIENTS
■■//ingredient 1
■■InsertIngredient(0,"Spraycan_Black");//you can insert multiple ingredients this way
■■
■■m_IngredientAddHealth[0] = 0;// 0 = do nothing
■■m_IngredientSetHealth[0] = -1; // -1 = do nothing
■■m_IngredientAddQuantity[0] = -40;// 0 = do nothing
■■m_IngredientDestroy[0] = false;//true = destroy, false = do nothing
■■m_IngredientUseSoftSkills[0] = true;// set 'true' to allow modification of the values by softskills on this in
■■
■■//ingredient 2
■■InsertIngredient(1,"GhillieAtt_Woodland");//you can insert multiple ingredients this way
■■
■■m_IngredientAddHealth[1] = 0;// 0 = do nothing
■■m_IngredientSetHealth[1] = -1; // -1 = do nothing
■■m_IngredientAddQuantity[1] = 0;// 0 = do nothing
■■m_IngredientDestroy[1] = false;// false = do nothing
■■m_IngredientUseSoftSkills[1] = false;// set 'true' to allow modification of the values by softskills on this i
■■//----------------------------------------------------------------------------------------------------------------------
■■
■■//result1
■■/*
■■AddResult("GhillieAtt_Mossy");//add results here

■■m_ResultSetFullQuantity[0] = false;//true = set full quantity, false = do nothing
■■m_ResultSetQuantity[0] = -1;//-1 = do nothing
■■m_ResultSetHealth[0] = -1;//-1 = do nothing
■■m_ResultInheritsHealth[0] = 1;// (value) == -1 means do nothing; a (value) >= 0 means this result will in
■■m_ResultInheritsColor[0] = -1;// (value) == -1 means do nothing; a (value) >= 0 means this result class
■■m_ResultToInventory[0] = 1;//(value) == -2 spawn result on the ground;(value) == -1 place anywhere in
■■m_ResultUseSoftSkills[0] = false;// set 'true' to allow modification of the values by softskills on this resu
■■m_ResultReplacesIngredient[0] = -1;// value == -1 means do nothing; a value >= 0 means this result wi
■■*/
■}
```

```
// ---------------------------------------------
// ------ PaintGhillieAttGreen.c - START --------------------
// ---------------------------------------------


class PaintGhillieAttGreen extends RecipeBase
{
■override void Init()
■{
■■m_Name = "#STR_paint0";
■■m_IsInstaRecipe = false;//should this recipe be performed instantly without animation
■■m_AnimationLength = 1.5;//animation length in relative time units
■■m_Specialty = 0.01;// value > 0 for roughness, value < 0 for precision
■■
■■
■■//conditions
■■m_MinDamageIngredient[0] = -1;//-1 = disable check
■■m_MaxDamageIngredient[0] = 3;//-1 = disable check
■■
■■m_MinQuantityIngredient[0] = 40;//-1 = disable check
■■m_MaxQuantityIngredient[0] = -1;//-1 = disable check
■■
■■m_MinDamageIngredient[1] = -1;//-1 = disable check
■■m_MaxDamageIngredient[1] = 3;//-1 = disable check
■■
■■m_MinQuantityIngredient[1] = -1;//-1 = disable check
■■m_MaxQuantityIngredient[1] = -1;//-1 = disable check
■■//----------------------------------------------------------------------------------------------------------------------
■■
■■//INGREDIENTS
■■//ingredient 1
■■InsertIngredient(0,"Spraycan_Green");//you can insert multiple ingredients this way
■■
■■m_IngredientAddHealth[0] = 0;// 0 = do nothing
■■m_IngredientSetHealth[0] = -1; // -1 = do nothing
■■m_IngredientAddQuantity[0] = -40;// 0 = do nothing
■■m_IngredientDestroy[0] = false;//true = destroy, false = do nothing
■■m_IngredientUseSoftSkills[0] = true;// set 'true' to allow modification of the values by softskills on this in
■■
■■//ingredient 2
■■InsertIngredient(1,"GhillieAtt_Tan");//you can insert multiple ingredients this way
■■
■■m_IngredientAddHealth[1] = 0;// 0 = do nothing
■■m_IngredientSetHealth[1] = -1; // -1 = do nothing
■■m_IngredientAddQuantity[1] = 0;// 0 = do nothing
■■m_IngredientDestroy[1] = false;// false = do nothing
■■m_IngredientUseSoftSkills[1] = false;// set 'true' to allow modification of the values by softskills on this i
■■//----------------------------------------------------------------------------------------------------------------------
■■
■■//result1
■■AddResult("GhillieAtt_Woodland");//add results here

■■m_ResultSetFullQuantity[0] = false;//true = set full quantity, false = do nothing
■■m_ResultSetQuantity[0] = -1;//-1 = do nothing
■■m_ResultSetHealth[0] = -1;//-1 = do nothing
■■m_ResultInheritsHealth[0] = 1;// (value) == -1 means do nothing; a (value) >= 0 means this result will in
■■m_ResultInheritsColor[0] = -1;// (value) == -1 means do nothing; a (value) >= 0 means this result classn
■■m_ResultToInventory[0] = 1;//(value) == -2 spawn result on the ground;(value) == -1 place anywhere in
■■m_ResultUseSoftSkills[0] = false;// set 'true' to allow modification of the values by softskills on this resu
■■m_ResultReplacesIngredient[0] = -1;// value == -1 means do nothing; a value >= 0 means this result wi
■}

■override bool CanDo(ItemBase ingredients[], PlayerBase player)//final check for recipe's validity
```

```
// ---------------------------------------------
// ------ PaintGhillieAttGreenMossy.c - START --------------------
// ---------------------------------------------


class PaintGhillieAttGreenMossy extends RecipeBase
{
■override void Init()
■{
■■m_Name = "#STR_paint0";
■■m_IsInstaRecipe = false;//should this recipe be performed instantly without animation
■■m_AnimationLength = 1.5;//animation length in relative time units
■■m_Specialty = 0.01;// value > 0 for roughness, value < 0 for precision
■■
■■
■■//conditions
■■m_MinDamageIngredient[0] = -1;//-1 = disable check
■■m_MaxDamageIngredient[0] = 3;//-1 = disable check
■■
■■m_MinQuantityIngredient[0] = 40;//-1 = disable check
■■m_MaxQuantityIngredient[0] = -1;//-1 = disable check
■■
■■m_MinDamageIngredient[1] = -1;//-1 = disable check
■■m_MaxDamageIngredient[1] = 3;//-1 = disable check
■■
■■m_MinQuantityIngredient[1] = -1;//-1 = disable check
■■m_MaxQuantityIngredient[1] = -1;//-1 = disable check
■■//----------------------------------------------------------------------------------------------------------------------
■■
■■//INGREDIENTS
■■//ingredient 1
■■InsertIngredient(0,"Spraycan_Green");//you can insert multiple ingredients this way
■■
■■m_IngredientAddHealth[0] = 0;// 0 = do nothing
■■m_IngredientSetHealth[0] = -1; // -1 = do nothing
■■m_IngredientAddQuantity[0] = -40;// 0 = do nothing
■■m_IngredientDestroy[0] = false;//true = destroy, false = do nothing
■■m_IngredientUseSoftSkills[0] = true;// set 'true' to allow modification of the values by softskills on this in
■■
■■//ingredient 2
■■InsertIngredient(1,"GhillieAtt_Mossy");//you can insert multiple ingredients this way
■■
■■m_IngredientAddHealth[1] = 0;// 0 = do nothing
■■m_IngredientSetHealth[1] = -1; // -1 = do nothing
■■m_IngredientAddQuantity[1] = 0;// 0 = do nothing
■■m_IngredientDestroy[1] = false;// false = do nothing
■■m_IngredientUseSoftSkills[1] = false;// set 'true' to allow modification of the values by softskills on this i
■■//----------------------------------------------------------------------------------------------------------------------
■■
■■//result1
■■/*
■■AddResult("GhillieAtt_Woodland");//add results here

■■m_ResultSetFullQuantity[0] = false;//true = set full quantity, false = do nothing
■■m_ResultSetQuantity[0] = -1;//-1 = do nothing
■■m_ResultSetHealth[0] = -1;//-1 = do nothing
■■m_ResultInheritsHealth[0] = 1;// (value) == -1 means do nothing; a (value) >= 0 means this result will in
■■m_ResultInheritsColor[0] = -1;// (value) == -1 means do nothing; a (value) >= 0 means this result class
■■m_ResultToInventory[0] = 1;//(value) == -2 spawn result on the ground;(value) == -1 place anywhere in
■■m_ResultUseSoftSkills[0] = false;// set 'true' to allow modification of the values by softskills on this resu
■■m_ResultReplacesIngredient[0] = -1;// value == -1 means do nothing; a value >= 0 means this result wi
■■*/
■}
```

```
// ---------------------------------------------
// ------ PaintGhillieBushragBlack.c - START --------------------
// ---------------------------------------------


class PaintGhillieBushragBlack extends RecipeBase
{
■override void Init()
■{
■■m_Name = "#STR_paint0";
■■m_IsInstaRecipe = false;//should this recipe be performed instantly without animation
■■m_AnimationLength = 1.5;//animation length in relative time units
■■m_Specialty = 0.01;// value > 0 for roughness, value < 0 for precision
■■
■■
■■//conditions
■■m_MinDamageIngredient[0] = -1;//-1 = disable check
■■m_MaxDamageIngredient[0] = 3;//-1 = disable check
■■
■■m_MinQuantityIngredient[0] = 50;//-1 = disable check
■■m_MaxQuantityIngredient[0] = -1;//-1 = disable check
■■
■■m_MinDamageIngredient[1] = -1;//-1 = disable check
■■m_MaxDamageIngredient[1] = 3;//-1 = disable check
■■
■■m_MinQuantityIngredient[1] = -1;//-1 = disable check
■■m_MaxQuantityIngredient[1] = -1;//-1 = disable check
■■//----------------------------------------------------------------------------------------------------------------------
■■
■■//INGREDIENTS
■■//ingredient 1
■■InsertIngredient(0,"Spraycan_Black");//you can insert multiple ingredients this way
■■
■■m_IngredientAddHealth[0] = 0;// 0 = do nothing
■■m_IngredientSetHealth[0] = -1; // -1 = do nothing
■■m_IngredientAddQuantity[0] = -50;// 0 = do nothing
■■m_IngredientDestroy[0] = false;//true = destroy, false = do nothing
■■m_IngredientUseSoftSkills[0] = true;// set 'true' to allow modification of the values by softskills on this in
■■
■■//ingredient 2
■■InsertIngredient(1,"GhillieBushrag_Woodland");//you can insert multiple ingredients this way
■■
■■m_IngredientAddHealth[1] = 0;// 0 = do nothing
■■m_IngredientSetHealth[1] = -1; // -1 = do nothing
■■m_IngredientAddQuantity[1] = 0;// 0 = do nothing
■■m_IngredientDestroy[1] = false;// false = do nothing
■■m_IngredientUseSoftSkills[1] = false;// set 'true' to allow modification of the values by softskills on this i
■■//----------------------------------------------------------------------------------------------------------------------
■■
■■//result1
■■/*
■■AddResult("GhillieBushrag_Mossy");//add results here

■■m_ResultSetFullQuantity[0] = false;//true = set full quantity, false = do nothing
■■m_ResultSetQuantity[0] = -1;//-1 = do nothing
■■m_ResultSetHealth[0] = -1;//-1 = do nothing
■■m_ResultInheritsHealth[0] = 1;// (value) == -1 means do nothing; a (value) >= 0 means this result will in
■■m_ResultInheritsColor[0] = -1;// (value) == -1 means do nothing; a (value) >= 0 means this result classn
■■m_ResultToInventory[0] = 1;//(value) == -2 spawn result on the ground;(value) == -1 place anywhere in
■■m_ResultUseSoftSkills[0] = false;// set 'true' to allow modification of the values by softskills on this resu
■■m_ResultReplacesIngredient[0] = -1;// value == -1 means do nothing; a value >= 0 means this result wi
■■*/
■}
```

```
// ---------------------------------------------
// ------ PaintGhillieBushragGreen.c - START --------------------
// ---------------------------------------------


class PaintGhillieBushragGreen extends RecipeBase
{
■override void Init()
■{
■■m_Name = "#STR_paint0";
■■m_IsInstaRecipe = false;//should this recipe be performed instantly without animation
■■m_AnimationLength = 1.5;//animation length in relative time units
■■m_Specialty = 0.01;// value > 0 for roughness, value < 0 for precision
■■
■■
■■//conditions
■■m_MinDamageIngredient[0] = -1;//-1 = disable check
■■m_MaxDamageIngredient[0] = 3;//-1 = disable check
■■
■■m_MinQuantityIngredient[0] = 50;//-1 = disable check
■■m_MaxQuantityIngredient[0] = -1;//-1 = disable check
■■
■■m_MinDamageIngredient[1] = -1;//-1 = disable check
■■m_MaxDamageIngredient[1] = 3;//-1 = disable check
■■
■■m_MinQuantityIngredient[1] = -1;//-1 = disable check
■■m_MaxQuantityIngredient[1] = -1;//-1 = disable check
■■//----------------------------------------------------------------------------------------------------------------------
■■
■■//INGREDIENTS
■■//ingredient 1
■■InsertIngredient(0,"Spraycan_Green");//you can insert multiple ingredients this way
■■
■■m_IngredientAddHealth[0] = 0;// 0 = do nothing
■■m_IngredientSetHealth[0] = -1; // -1 = do nothing
■■m_IngredientAddQuantity[0] = -50;// 0 = do nothing
■■m_IngredientDestroy[0] = false;//true = destroy, false = do nothing
■■m_IngredientUseSoftSkills[0] = true;// set 'true' to allow modification of the values by softskills on this in
■■
■■//ingredient 2
■■InsertIngredient(1,"GhillieBushrag_Tan");//you can insert multiple ingredients this way
■■
■■m_IngredientAddHealth[1] = 0;// 0 = do nothing
■■m_IngredientSetHealth[1] = -1; // -1 = do nothing
■■m_IngredientAddQuantity[1] = 0;// 0 = do nothing
■■m_IngredientDestroy[1] = false;// false = do nothing
■■m_IngredientUseSoftSkills[1] = false;// set 'true' to allow modification of the values by softskills on this i
■■//----------------------------------------------------------------------------------------------------------------------
■■
■■//result1
■■AddResult("GhillieBushrag_Woodland");//add results here

■■m_ResultSetFullQuantity[0] = false;//true = set full quantity, false = do nothing
■■m_ResultSetQuantity[0] = -1;//-1 = do nothing
■■m_ResultSetHealth[0] = 0;//-1 = do nothing
■■m_ResultInheritsHealth[0] = 1;// (value) == -1 means do nothing; a (value) >= 0 means this result will in
■■m_ResultInheritsColor[0] = -1;// (value) == -1 means do nothing; a (value) >= 0 means this result class
■■m_ResultToInventory[0] = 1;//(value) == -2 spawn result on the ground;(value) == -1 place anywhere in
■■m_ResultUseSoftSkills[0] = false;// set 'true' to allow modification of the values by softskills on this resu
■■m_ResultReplacesIngredient[0] = -1;// value == -1 means do nothing; a value >= 0 means this result wi
■}

■override bool CanDo(ItemBase ingredients[], PlayerBase player)//final check for recipe's validity
```

```
// ---------------------------------------------
// ------ PaintGhillieBushragGreenMossy.c - START --------------------
// ---------------------------------------------


class PaintGhillieBushragGreenMossy extends RecipeBase
{
■override void Init()
■{
■■m_Name = "#STR_paint0";
■■m_IsInstaRecipe = false;//should this recipe be performed instantly without animation
■■m_AnimationLength = 1.5;//animation length in relative time units
■■m_Specialty = 0.01;// value > 0 for roughness, value < 0 for precision
■■
■■
■■//conditions
■■m_MinDamageIngredient[0] = -1;//-1 = disable check
■■m_MaxDamageIngredient[0] = 3;//-1 = disable check
■■
■■m_MinQuantityIngredient[0] = 50;//-1 = disable check
■■m_MaxQuantityIngredient[0] = -1;//-1 = disable check
■■
■■m_MinDamageIngredient[1] = -1;//-1 = disable check
■■m_MaxDamageIngredient[1] = 3;//-1 = disable check
■■
■■m_MinQuantityIngredient[1] = -1;//-1 = disable check
■■m_MaxQuantityIngredient[1] = -1;//-1 = disable check
■■//----------------------------------------------------------------------------------------------------------------------
■■
■■//INGREDIENTS
■■//ingredient 1
■■InsertIngredient(0,"Spraycan_Green");//you can insert multiple ingredients this way
■■
■■m_IngredientAddHealth[0] = 0;// 0 = do nothing
■■m_IngredientSetHealth[0] = -1; // -1 = do nothing
■■m_IngredientAddQuantity[0] = -50;// 0 = do nothing
■■m_IngredientDestroy[0] = false;//true = destroy, false = do nothing
■■m_IngredientUseSoftSkills[0] = true;// set 'true' to allow modification of the values by softskills on this in
■■
■■//ingredient 2
■■InsertIngredient(1,"GhillieBushrag_Mossy");//you can insert multiple ingredients this way
■■
■■m_IngredientAddHealth[1] = 0;// 0 = do nothing
■■m_IngredientSetHealth[1] = -1; // -1 = do nothing
■■m_IngredientAddQuantity[1] = 0;// 0 = do nothing
■■m_IngredientDestroy[1] = false;// false = do nothing
■■m_IngredientUseSoftSkills[1] = false;// set 'true' to allow modification of the values by softskills on this i
■■//----------------------------------------------------------------------------------------------------------------------
■■
■■//result1
■■/*
■■AddResult("GhillieBushrag_Woodland");//add results here

■■m_ResultSetFullQuantity[0] = false;//true = set full quantity, false = do nothing
■■m_ResultSetQuantity[0] = -1;//-1 = do nothing
■■m_ResultSetHealth[0] = -1;//-1 = do nothing
■■m_ResultInheritsHealth[0] = 1;// (value) == -1 means do nothing; a (value) >= 0 means this result will in
■■m_ResultInheritsColor[0] = -1;// (value) == -1 means do nothing; a (value) >= 0 means this result classn
■■m_ResultToInventory[0] = 1;//(value) == -2 spawn result on the ground;(value) == -1 place anywhere in
■■m_ResultUseSoftSkills[0] = false;// set 'true' to allow modification of the values by softskills on this resu
■■m_ResultReplacesIngredient[0] = -1;// value == -1 means do nothing; a value >= 0 means this result wi
■■*/
■}
```

```
// ---------------------------------------------
// ------ PaintGhillieHoodBlack.c - START --------------------
// ---------------------------------------------


class PaintGhillieHoodBlack extends RecipeBase
{
■override void Init()
■{
■■m_Name = "#STR_paint0";
■■m_IsInstaRecipe = false;//should this recipe be performed instantly without animation
■■m_AnimationLength = 1.5;//animation length in relative time units
■■m_Specialty = 0.01;// value > 0 for roughness, value < 0 for precision
■■
■■
■■//conditions
■■m_MinDamageIngredient[0] = -1;//-1 = disable check
■■m_MaxDamageIngredient[0] = 3;//-1 = disable check
■■
■■m_MinQuantityIngredient[0] = 40;//-1 = disable check
■■m_MaxQuantityIngredient[0] = -1;//-1 = disable check
■■
■■m_MinDamageIngredient[1] = -1;//-1 = disable check
■■m_MaxDamageIngredient[1] = 3;//-1 = disable check
■■
■■m_MinQuantityIngredient[1] = -1;//-1 = disable check
■■m_MaxQuantityIngredient[1] = -1;//-1 = disable check
■■//----------------------------------------------------------------------------------------------------------------------
■■
■■//INGREDIENTS
■■//ingredient 1
■■InsertIngredient(0,"Spraycan_Black");//you can insert multiple ingredients this way
■■
■■m_IngredientAddHealth[0] = 0;// 0 = do nothing
■■m_IngredientSetHealth[0] = -1; // -1 = do nothing
■■m_IngredientAddQuantity[0] = -40;// 0 = do nothing
■■m_IngredientDestroy[0] = false;//true = destroy, false = do nothing
■■m_IngredientUseSoftSkills[0] = true;// set 'true' to allow modification of the values by softskills on this in
■■
■■//ingredient 2
■■InsertIngredient(1,"GhillieHood_Woodland");//you can insert multiple ingredients this way
■■
■■m_IngredientAddHealth[1] = 0;// 0 = do nothing
■■m_IngredientSetHealth[1] = -1; // -1 = do nothing
■■m_IngredientAddQuantity[1] = 0;// 0 = do nothing
■■m_IngredientDestroy[1] = false;// false = do nothing
■■m_IngredientUseSoftSkills[1] = false;// set 'true' to allow modification of the values by softskills on this i
■■//----------------------------------------------------------------------------------------------------------------------
■■
■■//result1
■■AddResult("GhillieHood_Mossy");//add results here

■■m_ResultSetFullQuantity[0] = false;//true = set full quantity, false = do nothing
■■m_ResultSetQuantity[0] = -1;//-1 = do nothing
■■m_ResultSetHealth[0] = -1;//-1 = do nothing
■■m_ResultInheritsHealth[0] = 1;// (value) == -1 means do nothing; a (value) >= 0 means this result will in
■■m_ResultInheritsColor[0] = -1;// (value) == -1 means do nothing; a (value) >= 0 means this result classn
■■m_ResultToInventory[0] = 1;//(value) == -2 spawn result on the ground;(value) == -1 place anywhere in
■■m_ResultUseSoftSkills[0] = false;// set 'true' to allow modification of the values by softskills on this resu
■■m_ResultReplacesIngredient[0] = -1;// value == -1 means do nothing; a value >= 0 means this result wi
■}

■override bool CanDo(ItemBase ingredients[], PlayerBase player)//final check for recipe's validity
```

```
// ---------------------------------------------
// ------ PaintGhillieHoodGreen.c - START --------------------
// ---------------------------------------------


class PaintGhillieHoodGreen extends RecipeBase
{
■override void Init()
■{
■■m_Name = "#STR_paint0";
■■m_IsInstaRecipe = false;//should this recipe be performed instantly without animation
■■m_AnimationLength = 1.5;//animation length in relative time units
■■m_Specialty = 0.01;// value > 0 for roughness, value < 0 for precision
■■
■■
■■//conditions
■■m_MinDamageIngredient[0] = -1;//-1 = disable check
■■m_MaxDamageIngredient[0] = 3;//-1 = disable check
■■
■■m_MinQuantityIngredient[0] = 40;//-1 = disable check
■■m_MaxQuantityIngredient[0] = -1;//-1 = disable check
■■
■■m_MinDamageIngredient[1] = -1;//-1 = disable check
■■m_MaxDamageIngredient[1] = 3;//-1 = disable check
■■
■■m_MinQuantityIngredient[1] = -1;//-1 = disable check
■■m_MaxQuantityIngredient[1] = -1;//-1 = disable check
■■//----------------------------------------------------------------------------------------------------------------------
■■
■■//INGREDIENTS
■■//ingredient 1
■■InsertIngredient(0,"Spraycan_Green");//you can insert multiple ingredients this way
■■
■■m_IngredientAddHealth[0] = 0;// 0 = do nothing
■■m_IngredientSetHealth[0] = -1; // -1 = do nothing
■■m_IngredientAddQuantity[0] = -40;// 0 = do nothing
■■m_IngredientDestroy[0] = false;//true = destroy, false = do nothing
■■m_IngredientUseSoftSkills[0] = true;// set 'true' to allow modification of the values by softskills on this in
■■
■■//ingredient 2
■■InsertIngredient(1,"GhillieHood_Tan");//you can insert multiple ingredients this way
■■
■■m_IngredientAddHealth[1] = 0;// 0 = do nothing
■■m_IngredientSetHealth[1] = -1; // -1 = do nothing
■■m_IngredientAddQuantity[1] = 0;// 0 = do nothing
■■m_IngredientDestroy[1] = false;// false = do nothing
■■m_IngredientUseSoftSkills[1] = false;// set 'true' to allow modification of the values by softskills on this i
■■//----------------------------------------------------------------------------------------------------------------------
■■
■■//result1
■■/*
■■AddResult("GhillieHood_Woodland");//add results here

■■m_ResultSetFullQuantity[0] = false;//true = set full quantity, false = do nothing
■■m_ResultSetQuantity[0] = -1;//-1 = do nothing
■■m_ResultSetHealth[0] = -1;//-1 = do nothing
■■m_ResultInheritsHealth[0] = 1;// (value) == -1 means do nothing; a (value) >= 0 means this result will in
■■m_ResultInheritsColor[0] = -1;// (value) == -1 means do nothing; a (value) >= 0 means this result classn
■■m_ResultToInventory[0] = 1;//(value) == -2 spawn result on the ground;(value) == -1 place anywhere in
■■m_ResultUseSoftSkills[0] = false;// set 'true' to allow modification of the values by softskills on this resu
■■m_ResultReplacesIngredient[0] = -1;// value == -1 means do nothing; a value >= 0 means this result wi
■■*/
■}
```

```
// ----------------------------------------------
// ------ PaintGhillieHoodGreenMossy.c - START --------------------
// ----------------------------------------------


class PaintGhillieHoodGreenMossy extends RecipeBase
{
■override void Init()
■{
■■m_Name = "#STR_paint0";
■■m_IsInstaRecipe = false;//should this recipe be performed instantly without animation
■■m_AnimationLength = 1.5;//animation length in relative time units
■■m_Specialty = 0.01;// value > 0 for roughness, value < 0 for precision
■■
■■
■■//conditions
■■m_MinDamageIngredient[0] = -1;//-1 = disable check
■■m_MaxDamageIngredient[0] = 3;//-1 = disable check
■■
■■m_MinQuantityIngredient[0] = 40;//-1 = disable check
■■m_MaxQuantityIngredient[0] = -1;//-1 = disable check
■■
■■m_MinDamageIngredient[1] = -1;//-1 = disable check
■■m_MaxDamageIngredient[1] = 3;//-1 = disable check
■■
■■m_MinQuantityIngredient[1] = -1;//-1 = disable check
■■m_MaxQuantityIngredient[1] = -1;//-1 = disable check
■■//----------------------------------------------------------------------------------------------------------------------
■■
■■//INGREDIENTS
■■//ingredient 1
■■InsertIngredient(0,"Spraycan_Green");//you can insert multiple ingredients this way
■■
■■m_IngredientAddHealth[0] = 0;// 0 = do nothing
■■m_IngredientSetHealth[0] = -1; // -1 = do nothing
■■m_IngredientAddQuantity[0] = -40;// 0 = do nothing
■■m_IngredientDestroy[0] = false;//true = destroy, false = do nothing
■■m_IngredientUseSoftSkills[0] = true;// set 'true' to allow modification of the values by softskills on this in
■■
■■//ingredient 2
■■InsertIngredient(1,"GhillieHood_Mossy");//you can insert multiple ingredients this way
■■
■■m_IngredientAddHealth[1] = 0;// 0 = do nothing
■■m_IngredientSetHealth[1] = -1; // -1 = do nothing
■■m_IngredientAddQuantity[1] = 0;// 0 = do nothing
■■m_IngredientDestroy[1] = false;// false = do nothing
■■m_IngredientUseSoftSkills[1] = false;// set 'true' to allow modification of the values by softskills on this i
■■//----------------------------------------------------------------------------------------------------------------------
■■
■■//result1
■■/*
■■AddResult("GhillieHood_Woodland");//add results here

■■m_ResultSetFullQuantity[0] = false;//true = set full quantity, false = do nothing
■■m_ResultSetQuantity[0] = -1;//-1 = do nothing
■■m_ResultSetHealth[0] = -1;//-1 = do nothing
■■m_ResultInheritsHealth[0] = 1;// (value) == -1 means do nothing; a (value) >= 0 means this result will in
■■m_ResultInheritsColor[0] = -1;// (value) == -1 means do nothing; a (value) >= 0 means this result classn
■■m_ResultToInventory[0] = 1;//(value) == -2 spawn result on the ground;(value) == -1 place anywhere in
■■m_ResultUseSoftSkills[0] = false;// set 'true' to allow modification of the values by softskills on this resu
■■m_ResultReplacesIngredient[0] = -1;// value == -1 means do nothing; a value >= 0 means this result wi
■■*/
■}
```

```
// ---------------------------------------------
// ------ PaintGhillieSuitBlack.c - START --------------------
// ---------------------------------------------


class PaintGhillieSuitBlack extends RecipeBase
{
■override void Init()
■{
■■m_Name = "#STR_paint0";
■■m_IsInstaRecipe = false;//should this recipe be performed instantly without animation
■■m_AnimationLength = 1.5;//animation length in relative time units
■■m_Specialty = 0.01;// value > 0 for roughness, value < 0 for precision
■■
■■
■■//conditions
■■m_MinDamageIngredient[0] = -1;//-1 = disable check
■■m_MaxDamageIngredient[0] = 3;//-1 = disable check
■■
■■m_MinQuantityIngredient[0] = 90;//-1 = disable check
■■m_MaxQuantityIngredient[0] = -1;//-1 = disable check
■■
■■m_MinDamageIngredient[1] = -1;//-1 = disable check
■■m_MaxDamageIngredient[1] = 3;//-1 = disable check
■■
■■m_MinQuantityIngredient[1] = -1;//-1 = disable check
■■m_MaxQuantityIngredient[1] = -1;//-1 = disable check
■■//----------------------------------------------------------------------------------------------------------------------
■■
■■//INGREDIENTS
■■//ingredient 1
■■InsertIngredient(0,"Spraycan_Black");//you can insert multiple ingredients this way
■■
■■m_IngredientAddHealth[0] = 0;// 0 = do nothing
■■m_IngredientSetHealth[0] = -1; // -1 = do nothing
■■m_IngredientAddQuantity[0] = -90;// 0 = do nothing
■■m_IngredientDestroy[0] = false;//true = destroy, false = do nothing
■■m_IngredientUseSoftSkills[0] = true;// set 'true' to allow modification of the values by softskills on this in
■■
■■//ingredient 2
■■InsertIngredient(1,"GhillieSuit_Woodland");//you can insert multiple ingredients this way
■■
■■m_IngredientAddHealth[1] = 0;// 0 = do nothing
■■m_IngredientSetHealth[1] = -1; // -1 = do nothing
■■m_IngredientAddQuantity[1] = 0;// 0 = do nothing
■■m_IngredientDestroy[1] = false;// false = do nothing
■■m_IngredientUseSoftSkills[1] = false;// set 'true' to allow modification of the values by softskills on this i
■■//----------------------------------------------------------------------------------------------------------------------
■■
■■//result1
■■AddResult("GhillieSuit_Mossy");//add results here

■■m_ResultSetFullQuantity[0] = false;//true = set full quantity, false = do nothing
■■m_ResultSetQuantity[0] = -1;//-1 = do nothing
■■m_ResultSetHealth[0] = -1;//-1 = do nothing
■■m_ResultInheritsHealth[0] = 1;// (value) == -1 means do nothing; a (value) >= 0 means this result will in
■■m_ResultInheritsColor[0] = -1;// (value) == -1 means do nothing; a (value) >= 0 means this result classn
■■m_ResultToInventory[0] = 1;//(value) == -2 spawn result on the ground;(value) == -1 place anywhere in
■■m_ResultUseSoftSkills[0] = false;// set 'true' to allow modification of the values by softskills on this resu
■■m_ResultReplacesIngredient[0] = -1;// value == -1 means do nothing; a value >= 0 means this result wi
■}

■override bool CanDo(ItemBase ingredients[], PlayerBase player)//final check for recipe's validity
```

```
// ---------------------------------------------
// ------ PaintGhillieSuitGreen.c - START --------------------
// ---------------------------------------------


class PaintGhillieSuitGreen extends RecipeBase
{
■override void Init()
■{
■■m_Name = "#STR_paint0";
■■m_IsInstaRecipe = false;//should this recipe be performed instantly without animation
■■m_AnimationLength = 1.5;//animation length in relative time units
■■m_Specialty = 0.01;// value > 0 for roughness, value < 0 for precision
■■
■■
■■//conditions
■■m_MinDamageIngredient[0] = -1;//-1 = disable check
■■m_MaxDamageIngredient[0] = 3;//-1 = disable check
■■
■■m_MinQuantityIngredient[0] = 90;//-1 = disable check
■■m_MaxQuantityIngredient[0] = -1;//-1 = disable check
■■
■■m_MinDamageIngredient[1] = -1;//-1 = disable check
■■m_MaxDamageIngredient[1] = 3;//-1 = disable check
■■
■■m_MinQuantityIngredient[1] = -1;//-1 = disable check
■■m_MaxQuantityIngredient[1] = -1;//-1 = disable check
■■■//----------------------------------------------------------------------------------------------------------------------
■■
■■//INGREDIENTS
■■//ingredient 1
■■InsertIngredient(0,"Spraycan_Green");//you can insert multiple ingredients this way
■■
■■m_IngredientAddHealth[0] = 0;// 0 = do nothing
■■m_IngredientSetHealth[0] = -1; // -1 = do nothing
■■m_IngredientAddQuantity[0] = -90;// 0 = do nothing
■■m_IngredientDestroy[0] = false;//true = destroy, false = do nothing
■■m_IngredientUseSoftSkills[0] = true;// set 'true' to allow modification of the values by softskills on this in
■■
■■//ingredient 2
■■InsertIngredient(1,"GhillieSuit_Tan");//you can insert multiple ingredients this way
■■
■■m_IngredientAddHealth[1] = 0;// 0 = do nothing
■■m_IngredientSetHealth[1] = -1; // -1 = do nothing
■■m_IngredientAddQuantity[1] = 0;// 0 = do nothing
■■m_IngredientDestroy[1] = false;// false = do nothing
■■m_IngredientUseSoftSkills[1] = false;// set 'true' to allow modification of the values by softskills on this i
■■■//----------------------------------------------------------------------------------------------------------------------
■■
■■//result1
■■■/*
■■AddResult("GhillieSuit_Woodland");//add results here

■■m_ResultSetFullQuantity[0] = false;//true = set full quantity, false = do nothing
■■m_ResultSetQuantity[0] = -1;//-1 = do nothing
■■m_ResultSetHealth[0] = -1;//-1 = do nothing
■■m_ResultInheritsHealth[0] = 1;// (value) == -1 means do nothing; a (value) >= 0 means this result will in
■■m_ResultInheritsColor[0] = -1;// (value) == -1 means do nothing; a (value) >= 0 means this result classn
■■m_ResultToInventory[0] = 1;//(value) == -2 spawn result on the ground;(value) == -1 place anywhere in
■■m_ResultUseSoftSkills[0] = false;// set 'true' to allow modification of the values by softskills on this resu
■■m_ResultReplacesIngredient[0] = -1;// value == -1 means do nothing; a value >= 0 means this result wi
■■■*/
■}
```

```
// ---------------------------------------------
// ------ PaintGhillieSuitGreenMossy.c - START --------------------
// ---------------------------------------------


class PaintGhillieSuitGreenMossy extends RecipeBase
{
■override void Init()
■{
■■m_Name = "#STR_paint0";
■■m_IsInstaRecipe = false;//should this recipe be performed instantly without animation
■■m_AnimationLength = 1.5;//animation length in relative time units
■■m_Specialty = 0.01;// value > 0 for roughness, value < 0 for precision
■■
■■
■■//conditions
■■m_MinDamageIngredient[0] = -1;//-1 = disable check
■■m_MaxDamageIngredient[0] = 3;//-1 = disable check
■■
■■m_MinQuantityIngredient[0] = 90;//-1 = disable check
■■m_MaxQuantityIngredient[0] = -1;//-1 = disable check
■■
■■m_MinDamageIngredient[1] = -1;//-1 = disable check
■■m_MaxDamageIngredient[1] = 3;//-1 = disable check
■■
■■m_MinQuantityIngredient[1] = -1;//-1 = disable check
■■m_MaxQuantityIngredient[1] = -1;//-1 = disable check
■■//----------------------------------------------------------------------------------------------------------------------
■■
■■//INGREDIENTS
■■//ingredient 1
■■InsertIngredient(0,"Spraycan_Green");//you can insert multiple ingredients this way
■■
■■m_IngredientAddHealth[0] = 0;// 0 = do nothing
■■m_IngredientSetHealth[0] = -1; // -1 = do nothing
■■m_IngredientAddQuantity[0] = -90;// 0 = do nothing
■■m_IngredientDestroy[0] = false;//true = destroy, false = do nothing
■■m_IngredientUseSoftSkills[0] = true;// set 'true' to allow modification of the values by softskills on this in
■■
■■//ingredient 2
■■InsertIngredient(1,"GhillieSuit_Mossy");//you can insert multiple ingredients this way
■■
■■m_IngredientAddHealth[1] = 0;// 0 = do nothing
■■m_IngredientSetHealth[1] = -1; // -1 = do nothing
■■m_IngredientAddQuantity[1] = 0;// 0 = do nothing
■■m_IngredientDestroy[1] = false;// false = do nothing
■■m_IngredientUseSoftSkills[1] = false;// set 'true' to allow modification of the values by softskills on this i
■■//----------------------------------------------------------------------------------------------------------------------
■■
■■//result1
■■/*
■■AddResult("GhillieSuit_Woodland");//add results here

■■m_ResultSetFullQuantity[0] = false;//true = set full quantity, false = do nothing
■■m_ResultSetQuantity[0] = -1;//-1 = do nothing
■■m_ResultSetHealth[0] = -1;//-1 = do nothing
■■m_ResultInheritsHealth[0] = 1;// (value) == -1 means do nothing; a (value) >= 0 means this result will in
■■m_ResultInheritsColor[0] = -1;// (value) == -1 means do nothing; a (value) >= 0 means this result classn
■■m_ResultToInventory[0] = 1;//(value) == -2 spawn result on the ground;(value) == -1 place anywhere in
■■m_ResultUseSoftSkills[0] = false;// set 'true' to allow modification of the values by softskills on this resu
■■m_ResultReplacesIngredient[0] = -1;// value == -1 means do nothing; a value >= 0 means this result wi
■■*/
■}
```

```
// ---------------------------------------------
// ------ PaintGhillieTopBlack.c - START --------------------
// ---------------------------------------------


class PaintGhillieTopBlack extends RecipeBase
{
■override void Init()
■{
■■m_Name = "#STR_paint0";
■■m_IsInstaRecipe = false;//should this recipe be performed instantly without animation
■■m_AnimationLength = 1.5;//animation length in relative time units
■■m_Specialty = 0.01;// value > 0 for roughness, value < 0 for precision
■■
■■
■■//conditions
■■m_MinDamageIngredient[0] = -1;//-1 = disable check
■■m_MaxDamageIngredient[0] = 3;//-1 = disable check
■■
■■m_MinQuantityIngredient[0] = 75;//-1 = disable check
■■m_MaxQuantityIngredient[0] = -1;//-1 = disable check
■■
■■m_MinDamageIngredient[1] = -1;//-1 = disable check
■■m_MaxDamageIngredient[1] = 3;//-1 = disable check
■■
■■m_MinQuantityIngredient[1] = -1;//-1 = disable check
■■m_MaxQuantityIngredient[1] = -1;//-1 = disable check
■■//----------------------------------------------------------------------------------------------------------------------
■■
■■//INGREDIENTS
■■//ingredient 1
■■InsertIngredient(0,"Spraycan_Black");//you can insert multiple ingredients this way
■■
■■m_IngredientAddHealth[0] = 0;// 0 = do nothing
■■m_IngredientSetHealth[0] = -1; // -1 = do nothing
■■m_IngredientAddQuantity[0] = -75;// 0 = do nothing
■■m_IngredientDestroy[0] = false;//true = destroy, false = do nothing
■■m_IngredientUseSoftSkills[0] = true;// set 'true' to allow modification of the values by softskills on this in
■■
■■//ingredient 2
■■InsertIngredient(1,"GhillieTop_Woodland");//you can insert multiple ingredients this way
■■
■■m_IngredientAddHealth[1] = 0;// 0 = do nothing
■■m_IngredientSetHealth[1] = -1; // -1 = do nothing
■■m_IngredientAddQuantity[1] = 0;// 0 = do nothing
■■m_IngredientDestroy[1] = false;// false = do nothing
■■m_IngredientUseSoftSkills[1] = false;// set 'true' to allow modification of the values by softskills on this i
■■//----------------------------------------------------------------------------------------------------------------------
■■
■■//result1
■■/*
■■AddResult("GhillieTop_Mossy");//add results here

■■m_ResultSetFullQuantity[0] = false;//true = set full quantity, false = do nothing
■■m_ResultSetQuantity[0] = -1;//-1 = do nothing
■■m_ResultSetHealth[0] = -1;//-1 = do nothing
■■m_ResultInheritsHealth[0] = 1;// (value) == -1 means do nothing; a (value) >= 0 means this result will in
■■m_ResultInheritsColor[0] = -1;// (value) == -1 means do nothing; a (value) >= 0 means this result classn
■■m_ResultToInventory[0] = 1;//(value) == -2 spawn result on the ground;(value) == -1 place anywhere in
■■m_ResultUseSoftSkills[0] = false;// set 'true' to allow modification of the values by softskills on this resu
■■m_ResultReplacesIngredient[0] = -1;// value == -1 means do nothing; a value >= 0 means this result wi
■■*/
■}
```

```
// ---------------------------------------------
// ------ PaintGhillieTopGreen.c - START --------------------
// ---------------------------------------------


class PaintGhillieTopGreen extends RecipeBase
{
■override void Init()
■{
■■m_Name = "#STR_paint0";
■■m_IsInstaRecipe = false;//should this recipe be performed instantly without animation
■■m_AnimationLength = 1.5;//animation length in relative time units
■■m_Specialty = 0.01;// value > 0 for roughness, value < 0 for precision
■■
■■
■■//conditions
■■m_MinDamageIngredient[0] = -1;//-1 = disable check
■■m_MaxDamageIngredient[0] = 3;//-1 = disable check
■■
■■m_MinQuantityIngredient[0] = 75;//-1 = disable check
■■m_MaxQuantityIngredient[0] = -1;//-1 = disable check
■■
■■m_MinDamageIngredient[1] = -1;//-1 = disable check
■■m_MaxDamageIngredient[1] = 3;//-1 = disable check
■■
■■m_MinQuantityIngredient[1] = -1;//-1 = disable check
■■m_MaxQuantityIngredient[1] = -1;//-1 = disable check
■■//----------------------------------------------------------------------------------------------------------------------
■■
■■//INGREDIENTS
■■//ingredient 1
■■InsertIngredient(0,"Spraycan_Green");//you can insert multiple ingredients this way
■■
■■m_IngredientAddHealth[0] = 0;// 0 = do nothing
■■m_IngredientSetHealth[0] = -1; // -1 = do nothing
■■m_IngredientAddQuantity[0] = -75;// 0 = do nothing
■■m_IngredientDestroy[0] = false;//true = destroy, false = do nothing
■■m_IngredientUseSoftSkills[0] = true;// set 'true' to allow modification of the values by softskills on this in
■■
■■//ingredient 2
■■InsertIngredient(1,"GhillieTop_Tan");//you can insert multiple ingredients this way
■■
■■m_IngredientAddHealth[1] = 0;// 0 = do nothing
■■m_IngredientSetHealth[1] = -1; // -1 = do nothing
■■m_IngredientAddQuantity[1] = 0;// 0 = do nothing
■■m_IngredientDestroy[1] = false;// false = do nothing
■■m_IngredientUseSoftSkills[1] = false;// set 'true' to allow modification of the values by softskills on this i
■■//----------------------------------------------------------------------------------------------------------------------
■■
■■//result1
■■/*
■■AddResult("GhillieTop_Woodland");//add results here

■■m_ResultSetFullQuantity[0] = false;//true = set full quantity, false = do nothing
■■m_ResultSetQuantity[0] = -1;//-1 = do nothing
■■m_ResultSetHealth[0] = -1;//-1 = do nothing
■■m_ResultInheritsHealth[0] = 1;// (value) == -1 means do nothing; a (value) >= 0 means this result will in
■■m_ResultInheritsColor[0] = -1;// (value) == -1 means do nothing; a (value) >= 0 means this result classn
■■m_ResultToInventory[0] = 1;//(value) == -2 spawn result on the ground;(value) == -1 place anywhere in
■■m_ResultUseSoftSkills[0] = false;// set 'true' to allow modification of the values by softskills on this resu
■■m_ResultReplacesIngredient[0] = -1;// value == -1 means do nothing; a value >= 0 means this result wi
■■*/
■}
```

```
// ---------------------------------------------
// ------ PaintGhillieTopGreenMossy.c - START --------------------
// ---------------------------------------------


class PaintGhillieTopGreenMossy extends RecipeBase
{
■override void Init()
■{
■■m_Name = "#STR_paint0";
■■m_IsInstaRecipe = false;//should this recipe be performed instantly without animation
■■m_AnimationLength = 1.5;//animation length in relative time units
■■m_Specialty = 0.01;// value > 0 for roughness, value < 0 for precision
■■
■■
■■//conditions
■■m_MinDamageIngredient[0] = -1;//-1 = disable check
■■m_MaxDamageIngredient[0] = 3;//-1 = disable check
■■
■■m_MinQuantityIngredient[0] = 75;//-1 = disable check
■■m_MaxQuantityIngredient[0] = -1;//-1 = disable check
■■
■■m_MinDamageIngredient[1] = -1;//-1 = disable check
■■m_MaxDamageIngredient[1] = 3;//-1 = disable check
■■
■■m_MinQuantityIngredient[1] = -1;//-1 = disable check
■■m_MaxQuantityIngredient[1] = -1;//-1 = disable check
■■//----------------------------------------------------------------------------------------------------------------------
■■
■■//INGREDIENTS
■■//ingredient 1
■■InsertIngredient(0,"Spraycan_Green");//you can insert multiple ingredients this way
■■
■■m_IngredientAddHealth[0] = 0;// 0 = do nothing
■■m_IngredientSetHealth[0] = -1; // -1 = do nothing
■■m_IngredientAddQuantity[0] = -75;// 0 = do nothing
■■m_IngredientDestroy[0] = false;//true = destroy, false = do nothing
■■m_IngredientUseSoftSkills[0] = true;// set 'true' to allow modification of the values by softskills on this in
■■
■■//ingredient 2
■■InsertIngredient(1,"GhillieTop_Mossy");//you can insert multiple ingredients this way
■■
■■m_IngredientAddHealth[1] = 0;// 0 = do nothing
■■m_IngredientSetHealth[1] = -1; // -1 = do nothing
■■m_IngredientAddQuantity[1] = 0;// 0 = do nothing
■■m_IngredientDestroy[1] = false;// false = do nothing
■■m_IngredientUseSoftSkills[1] = false;// set 'true' to allow modification of the values by softskills on this i
■■//----------------------------------------------------------------------------------------------------------------------
■■
■■//result1
■■/*
■■AddResult("GhillieTop_Woodland");//add results here

■■m_ResultSetFullQuantity[0] = false;//true = set full quantity, false = do nothing
■■m_ResultSetQuantity[0] = -1;//-1 = do nothing
■■m_ResultSetHealth[0] = -1;//-1 = do nothing
■■m_ResultInheritsHealth[0] = 1;// (value) == -1 means do nothing; a (value) >= 0 means this result will in
■■m_ResultInheritsColor[0] = -1;// (value) == -1 means do nothing; a (value) >= 0 means this result classn
■■m_ResultToInventory[0] = 1;//(value) == -2 spawn result on the ground;(value) == -1 place anywhere in
■■m_ResultUseSoftSkills[0] = false;// set 'true' to allow modification of the values by softskills on this resu
■■m_ResultReplacesIngredient[0] = -1;// value == -1 means do nothing; a value >= 0 means this result wi
■■*/
■}
```

```
// ---------------------------------------------
// ------ PaintGorkaHelmet.c - START --------------------
// ---------------------------------------------


class PaintGorkaHelmet extends RecipeBase
{■
■override void Init()
■{
■■m_Name = "#STR_paint0";
■■m_IsInstaRecipe = false;//should this recipe be performed instantly without animation
■■m_AnimationLength = 1.5;//animation length in relative time units
■■m_Specialty = 0.01;// value > 0 for roughness, value < 0 for precision
■■
■■
■■//conditions
■■m_MinDamageIngredient[0] = -1;//-1 = disable check
■■m_MaxDamageIngredient[0] = 3;//-1 = disable check
■■
■■m_MinQuantityIngredient[0] = 25;//-1 = disable check
■■m_MaxQuantityIngredient[0] = -1;//-1 = disable check
■■
■■m_MinDamageIngredient[1] = -1;//-1 = disable check
■■m_MaxDamageIngredient[1] = 3;//-1 = disable check
■■
■■m_MinQuantityIngredient[1] = -1;//-1 = disable check
■■m_MaxQuantityIngredient[1] = -1;//-1 = disable check
■■//----------------------------------------------------------------------------------------------------------------------
■■
■■//INGREDIENTS
■■//ingredient 1
■■InsertIngredient(0,"Spraycan_ColorBase");//you can insert multiple ingredients this way
■■
■■m_IngredientAddHealth[0] = 0;// 0 = do nothing
■■m_IngredientSetHealth[0] = -1; // -1 = do nothing
■■m_IngredientAddQuantity[0] = -25;// 0 = do nothing
■■m_IngredientDestroy[0] = false;//true = destroy, false = do nothing
■■m_IngredientUseSoftSkills[0] = true;// set 'true' to allow modification of the values by softskills on this in
■■
■■//ingredient 2
■■InsertIngredient(1,"GorkaHelmet");//you can insert multiple ingredients this way
■■
■■m_IngredientAddHealth[1] = 0;// 0 = do nothing
■■m_IngredientSetHealth[1] = -1; // -1 = do nothing
■■m_IngredientAddQuantity[1] = 0;// 0 = do nothing
■■m_IngredientDestroy[1] = false;// false = do nothing
■■m_IngredientUseSoftSkills[1] = false;// set 'true' to allow modification of the values by softskills on this i
■■//----------------------------------------------------------------------------------------------------------------------
■■
■■//result1
■■/*
■■AddResult("GorkaHelmet_");//add results here

■■m_ResultSetFullQuantity[0] = false;//true = set full quantity, false = do nothing
■■m_ResultSetQuantity[0] = -1;//-1 = do nothing
■■m_ResultSetHealth[0] = -1;//-1 = do nothing
■■m_ResultInheritsHealth[0] = 1;// (value) == -1 means do nothing; a (value) >= 0 means this result will in
■■m_ResultInheritsColor[0] = 0;// (value) == -1 means do nothing; a (value) >= 0 means this result classn
■■m_ResultToInventory[0] = 1;//(value) == -2 spawn result on the ground;(value) == -1 place anywhere in
■■m_ResultUseSoftSkills[0] = false;// set 'true' to allow modification of the values by softskills on this resu
■■m_ResultReplacesIngredient[0] = -1;// value == -1 means do nothing; a value >= 0 means this result wi
■■*/
■}
```

```
// ---------------------------------------------
// ------ PaintGorkaHelmetComplete.c - START --------------------
// ---------------------------------------------


class PaintGorkaHelmetComplete extends RecipeBase
{■
■override void Init()
■{
■■m_Name = "#STR_paint0";
■■m_IsInstaRecipe = false;//should this recipe be performed instantly without animation
■■m_AnimationLength = 1.5;//animation length in relative time units
■■m_Specialty = 0.01;// value > 0 for roughness, value < 0 for precision
■■
■■
■■//conditions
■■m_MinDamageIngredient[0] = -1;//-1 = disable check
■■m_MaxDamageIngredient[0] = 3;//-1 = disable check
■■
■■m_MinQuantityIngredient[0] = 25;//-1 = disable check
■■m_MaxQuantityIngredient[0] = -1;//-1 = disable check
■■
■■m_MinDamageIngredient[1] = -1;//-1 = disable check
■■m_MaxDamageIngredient[1] = 3;//-1 = disable check
■■
■■m_MinQuantityIngredient[1] = -1;//-1 = disable check
■■m_MaxQuantityIngredient[1] = -1;//-1 = disable check
■■//----------------------------------------------------------------------------------------------------------------------
■■
■■//INGREDIENTS
■■//ingredient 1
■■InsertIngredient(0,"Spraycan_ColorBase");//you can insert multiple ingredients this way
■■
■■m_IngredientAddHealth[0] = 0;// 0 = do nothing
■■m_IngredientSetHealth[0] = -1; // -1 = do nothing
■■m_IngredientAddQuantity[0] = -25;// 0 = do nothing
■■m_IngredientDestroy[0] = false;//true = destroy, false = do nothing
■■m_IngredientUseSoftSkills[0] = true;// set 'true' to allow modification of the values by softskills on this in
■■
■■//ingredient 2
■■InsertIngredient(1,"GorkaHelmetComplete");//you can insert multiple ingredients this way
■■
■■m_IngredientAddHealth[1] = 0;// 0 = do nothing
■■m_IngredientSetHealth[1] = -1; // -1 = do nothing
■■m_IngredientAddQuantity[1] = 0;// 0 = do nothing
■■m_IngredientDestroy[1] = false;// false = do nothing
■■m_IngredientUseSoftSkills[1] = false;// set 'true' to allow modification of the values by softskills on this i
■■//----------------------------------------------------------------------------------------------------------------------
■■
■■//result1
■■/*
■■AddResult("GorkaHelmetComplete_");//add results here

■■m_ResultSetFullQuantity[0] = false;//true = set full quantity, false = do nothing
■■m_ResultSetQuantity[0] = -1;//-1 = do nothing
■■m_ResultSetHealth[0] = -1;//-1 = do nothing
■■m_ResultInheritsHealth[0] = 1;// (value) == -1 means do nothing; a (value) >= 0 means this result will in
■■m_ResultInheritsColor[0] = 0;// (value) == -1 means do nothing; a (value) >= 0 means this result classn
■■m_ResultToInventory[0] = 1;//(value) == -2 spawn result on the ground;(value) == -1 place anywhere in
■■m_ResultUseSoftSkills[0] = false;// set 'true' to allow modification of the values by softskills on this resu
■■m_ResultReplacesIngredient[0] = 1;// value == -1 means do nothing; a value >= 0 means this result wil
■■*/
■}
```

```
// ---------------------------------------------
// ------ PaintItem.c - START --------------------
// ---------------------------------------------


class PaintItem
{
■//! WIll open the 'item_target' by spawning a new entity and transferring item variables to the new one
■static void Paint(ItemBase item_tool, ItemBase item_target, string base_name, PlayerBase player, float s
■{
■■string spray_color = item_tool.ConfigGetString("color");
■■string item_color = item_target.ConfigGetString("color");

■■string new_class_name = base_name + "_" + spray_color;
■■PaintItem.SwitchItems(item_target, new_class_name, player);
■}

■static bool CanPaintItem(ItemBase item_tool, ItemBase item_target)
■{
■■string spray_color = item_tool.ConfigGetString("color");
■■string item_color = item_target.ConfigGetString("color");
■■
■■if( spray_color != item_color )
■■{
■■■return true;
■■}
■■else
■■{
■■■return false;
■■}
■}
■
■//! Will switch the 'item' for a new game entity, the new entity's classname will be formed by adding the 's
■static void SwitchItems (EntityAI old_item, string new_item, PlayerBase player)
■{
■■MiscGameplayFunctions.TurnItemIntoItemEx(player, new PaintItemLambda(old_item, new_item, playe
■}
};


class PaintItemLambda : TurnItemIntoItemLambda
{
■void PaintItemLambda (EntityAI old_item, string new_item_type, PlayerBase player) { }
};




// ---------------------------------------------
// ------ PaintItem.c - END ----------------------
// ---------------------------------------------


// ---------------------------------------------
// ------ PaintM4A1.c - START --------------------
// ---------------------------------------------


class PaintM4A1 extends RecipeBase
{■
■override void Init()
■{
■■m_Name = "#STR_paint0";
■■m_IsInstaRecipe = false;//should this recipe be performed instantly without animation
```

```
// ---------------------------------------------
// ------ PaintM4CQBBttstck.c - START --------------------
// ---------------------------------------------


class PaintM4CQBBttstck extends RecipeBase
{■
■override void Init()
■{
■■m_Name = "#STR_paint0";
■■m_IsInstaRecipe = false;//should this recipe be performed instantly without animation
■■m_AnimationLength = 1.5;//animation length in relative time units
■■m_Specialty = 0.01;// value > 0 for roughness, value < 0 for precision
■■
■■
■■//conditions
■■m_MinDamageIngredient[0] = -1;//-1 = disable check
■■m_MaxDamageIngredient[0] = 3;//-1 = disable check
■■
■■m_MinQuantityIngredient[0] = 25;//-1 = disable check
■■m_MaxQuantityIngredient[0] = -1;//-1 = disable check
■■
■■m_MinDamageIngredient[1] = -1;//-1 = disable check
■■m_MaxDamageIngredient[1] = 3;//-1 = disable check
■■
■■m_MinQuantityIngredient[1] = -1;//-1 = disable check
■■m_MaxQuantityIngredient[1] = -1;//-1 = disable check
■■//----------------------------------------------------------------------------------------------------------------------
■■
■■//INGREDIENTS
■■//ingredient 1
■■InsertIngredient(0,"Spraycan_ColorBase");//you can insert multiple ingredients this way
■■
■■m_IngredientAddHealth[0] = 0;// 0 = do nothing
■■m_IngredientSetHealth[0] = -1; // -1 = do nothing
■■m_IngredientAddQuantity[0] = -25;// 0 = do nothing
■■m_IngredientDestroy[0] = false;//true = destroy, false = do nothing
■■m_IngredientUseSoftSkills[0] = true;// set 'true' to allow modification of the values by softskills on this in
■■
■■//ingredient 2
■■InsertIngredient(1,"M4_CQBBttstck");//you can insert multiple ingredients this way
■■
■■m_IngredientAddHealth[1] = 0;// 0 = do nothing
■■m_IngredientSetHealth[1] = -1; // -1 = do nothing
■■m_IngredientAddQuantity[1] = 0;// 0 = do nothing
■■m_IngredientDestroy[1] = false;// false = do nothing
■■m_IngredientUseSoftSkills[1] = false;// set 'true' to allow modification of the values by softskills on this i
■■//----------------------------------------------------------------------------------------------------------------------
■■
■■//result1
■■/*
■■AddResult("M4_CQBBttstck_");//add results here

■■m_ResultSetFullQuantity[0] = false;//true = set full quantity, false = do nothing
■■m_ResultSetQuantity[0] = -1;//-1 = do nothing
■■m_ResultSetHealth[0] = -1;//-1 = do nothing
■■m_ResultInheritsHealth[0] = 1;// (value) == -1 means do nothing; a (value) >= 0 means this result will in
■■m_ResultInheritsColor[0] = 0;// (value) == -1 means do nothing; a (value) >= 0 means this result classn
■■m_ResultToInventory[0] = -1;//(value) == -2 spawn result on the ground;(value) == -1 place anywhere in
■■m_ResultUseSoftSkills[0] = false;// set 'true' to allow modification of the values by softskills on this resu
■■m_ResultReplacesIngredient[0] = -1;// value == -1 means do nothing; a value >= 0 means this result wi
■■*/
■}
```

```
// ---------------------------------------------
// ------ PaintM4MPBttstck.c - START --------------------
// ---------------------------------------------


class PaintM4MPBttstck extends RecipeBase
{■
■override void Init()
■{
■■m_Name = "#STR_paint0";
■■m_IsInstaRecipe = false;//should this recipe be performed instantly without animation
■■m_AnimationLength = 1.5;//animation length in relative time units
■■m_Specialty = 0.01;// value > 0 for roughness, value < 0 for precision
■■
■■
■■//conditions
■■m_MinDamageIngredient[0] = -1;//-1 = disable check
■■m_MaxDamageIngredient[0] = 3;//-1 = disable check
■■
■■m_MinQuantityIngredient[0] = 25;//-1 = disable check
■■m_MaxQuantityIngredient[0] = -1;//-1 = disable check
■■
■■m_MinDamageIngredient[1] = -1;//-1 = disable check
■■m_MaxDamageIngredient[1] = 3;//-1 = disable check
■■
■■m_MinQuantityIngredient[1] = -1;//-1 = disable check
■■m_MaxQuantityIngredient[1] = -1;//-1 = disable check
■■//----------------------------------------------------------------------------------------------------------------------
■■
■■//INGREDIENTS
■■//ingredient 1
■■InsertIngredient(0,"Spraycan_ColorBase");//you can insert multiple ingredients this way
■■
■■m_IngredientAddHealth[0] = 0;// 0 = do nothing
■■m_IngredientSetHealth[0] = -1; // -1 = do nothing
■■m_IngredientAddQuantity[0] = -25;// 0 = do nothing
■■m_IngredientDestroy[0] = false;//true = destroy, false = do nothing
■■m_IngredientUseSoftSkills[0] = true;// set 'true' to allow modification of the values by softskills on this in
■■
■■//ingredient 2
■■InsertIngredient(1,"M4_MPBttstck");//you can insert multiple ingredients this way
■■
■■m_IngredientAddHealth[1] = 0;// 0 = do nothing
■■m_IngredientSetHealth[1] = -1; // -1 = do nothing
■■m_IngredientAddQuantity[1] = 0;// 0 = do nothing
■■m_IngredientDestroy[1] = false;// false = do nothing
■■m_IngredientUseSoftSkills[1] = false;// set 'true' to allow modification of the values by softskills on this i
■■//----------------------------------------------------------------------------------------------------------------------
■■
■■//result1
■■/*
■■AddResult("M4_MPBttstck_");//add results here

■■m_ResultSetFullQuantity[0] = false;//true = set full quantity, false = do nothing
■■m_ResultSetQuantity[0] = -1;//-1 = do nothing
■■m_ResultSetHealth[0] = -1;//-1 = do nothing
■■m_ResultInheritsHealth[0] = 1;// (value) == -1 means do nothing; a (value) >= 0 means this result will in
■■m_ResultInheritsColor[0] = 0;// (value) == -1 means do nothing; a (value) >= 0 means this result classn
■■m_ResultToInventory[0] = -1;//(value) == -2 spawn result on the ground;(value) == -1 place anywhere in
■■m_ResultUseSoftSkills[0] = false;// set 'true' to allow modification of the values by softskills on this resu
■■m_ResultReplacesIngredient[0] = -1;// value == -1 means do nothing; a value >= 0 means this result wi
■■*/
■}
```

```
// ---------------------------------------------
// ------ PaintM4MPHndgrd.c - START --------------------
// ---------------------------------------------


class PaintM4MPHndgrd extends RecipeBase
{■
■override void Init()
■{
■■m_Name = "#STR_paint0";
■■m_IsInstaRecipe = false;//should this recipe be performed instantly without animation
■■m_AnimationLength = 1.5;//animation length in relative time units
■■m_Specialty = 0.01;// value > 0 for roughness, value < 0 for precision
■■
■■
■■//conditions
■■m_MinDamageIngredient[0] = -1;//-1 = disable check
■■m_MaxDamageIngredient[0] = 3;//-1 = disable check
■■
■■m_MinQuantityIngredient[0] = 25;//-1 = disable check
■■m_MaxQuantityIngredient[0] = -1;//-1 = disable check
■■
■■m_MinDamageIngredient[1] = -1;//-1 = disable check
■■m_MaxDamageIngredient[1] = 3;//-1 = disable check
■■
■■m_MinQuantityIngredient[1] = -1;//-1 = disable check
■■m_MaxQuantityIngredient[1] = -1;//-1 = disable check
■■//----------------------------------------------------------------------------------------------------------------------
■■
■■//INGREDIENTS
■■//ingredient 1
■■InsertIngredient(0,"Spraycan_ColorBase");//you can insert multiple ingredients this way
■■
■■m_IngredientAddHealth[0] = 0;// 0 = do nothing
■■m_IngredientSetHealth[0] = -1; // -1 = do nothing
■■m_IngredientAddQuantity[0] = -25;// 0 = do nothing
■■m_IngredientDestroy[0] = false;//true = destroy, false = do nothing
■■m_IngredientUseSoftSkills[0] = true;// set 'true' to allow modification of the values by softskills on this in
■■
■■//ingredient 2
■■InsertIngredient(1,"M4_MPHndgrd");//you can insert multiple ingredients this way
■■
■■m_IngredientAddHealth[1] = 0;// 0 = do nothing
■■m_IngredientSetHealth[1] = -1; // -1 = do nothing
■■m_IngredientAddQuantity[1] = 0;// 0 = do nothing
■■m_IngredientDestroy[1] = false;// false = do nothing
■■m_IngredientUseSoftSkills[1] = false;// set 'true' to allow modification of the values by softskills on this i
■■//----------------------------------------------------------------------------------------------------------------------
■■
■■//result1
■■/*
■■AddResult("M4_MPHndgrd_");//add results here

■■m_ResultSetFullQuantity[0] = false;//true = set full quantity, false = do nothing
■■m_ResultSetQuantity[0] = -1;//-1 = do nothing
■■m_ResultSetHealth[0] = -1;//-1 = do nothing
■■m_ResultInheritsHealth[0] = 1;// (value) == -1 means do nothing; a (value) >= 0 means this result will in
■■m_ResultInheritsColor[0] = 0;// (value) == -1 means do nothing; a (value) >= 0 means this result classn
■■m_ResultToInventory[0] = -1;//(value) == -2 spawn result on the ground;(value) == -1 place anywhere in
■■m_ResultUseSoftSkills[0] = false;// set 'true' to allow modification of the values by softskills on this resu
■■m_ResultReplacesIngredient[0] = -1;// value == -1 means do nothing; a value >= 0 means this result wi
■■*/
■}
```

```
// ---------------------------------------------
// ------ PaintM4OEBttstck.c - START --------------------
// ---------------------------------------------


class PaintM4OEBttstck extends RecipeBase
{■
■override void Init()
■{
■■m_Name = "#STR_paint0";
■■m_IsInstaRecipe = false;//should this recipe be performed instantly without animation
■■m_AnimationLength = 1.5;//animation length in relative time units
■■m_Specialty = 0.01;// value > 0 for roughness, value < 0 for precision
■■
■■
■■//conditions
■■m_MinDamageIngredient[0] = -1;//-1 = disable check
■■m_MaxDamageIngredient[0] = 3;//-1 = disable check
■■
■■m_MinQuantityIngredient[0] = 25;//-1 = disable check
■■m_MaxQuantityIngredient[0] = -1;//-1 = disable check
■■
■■m_MinDamageIngredient[1] = -1;//-1 = disable check
■■m_MaxDamageIngredient[1] = 3;//-1 = disable check
■■
■■m_MinQuantityIngredient[1] = -1;//-1 = disable check
■■m_MaxQuantityIngredient[1] = -1;//-1 = disable check
■■//----------------------------------------------------------------------------------------------------------------------
■■
■■//INGREDIENTS
■■//ingredient 1
■■InsertIngredient(0,"Spraycan_ColorBase");//you can insert multiple ingredients this way
■■
■■m_IngredientAddHealth[0] = 0;// 0 = do nothing
■■m_IngredientSetHealth[0] = -1; // -1 = do nothing
■■m_IngredientAddQuantity[0] = -25;// 0 = do nothing
■■m_IngredientDestroy[0] = false;//true = destroy, false = do nothing
■■m_IngredientUseSoftSkills[0] = true;// set 'true' to allow modification of the values by softskills on this in
■■
■■//ingredient 2
■■InsertIngredient(1,"M4_OEBttstck");//you can insert multiple ingredients this way
■■
■■m_IngredientAddHealth[1] = 0;// 0 = do nothing
■■m_IngredientSetHealth[1] = -1; // -1 = do nothing
■■m_IngredientAddQuantity[1] = 0;// 0 = do nothing
■■m_IngredientDestroy[1] = false;// false = do nothing
■■m_IngredientUseSoftSkills[1] = false;// set 'true' to allow modification of the values by softskills on this i
■■//----------------------------------------------------------------------------------------------------------------------
■■
■■//result1
■■/*
■■AddResult("M4_OEBttstck_");//add results here

■■m_ResultSetFullQuantity[0] = false;//true = set full quantity, false = do nothing
■■m_ResultSetQuantity[0] = -1;//-1 = do nothing
■■m_ResultSetHealth[0] = -1;//-1 = do nothing
■■m_ResultInheritsHealth[0] = 1;// (value) == -1 means do nothing; a (value) >= 0 means this result will in
■■m_ResultInheritsColor[0] = 0;// (value) == -1 means do nothing; a (value) >= 0 means this result classn
■■m_ResultToInventory[0] = -1;//(value) == -2 spawn result on the ground;(value) == -1 place anywhere in
■■m_ResultUseSoftSkills[0] = false;// set 'true' to allow modification of the values by softskills on this resu
■■m_ResultReplacesIngredient[0] = -1;// value == -1 means do nothing; a value >= 0 means this result wi
■■*/
■}
```

```
// ---------------------------------------------
// ------ PaintM4PlasticHndgrd.c - START --------------------
// ---------------------------------------------


class PaintM4PlasticHndgrd extends RecipeBase
{■
■override void Init()
■{
■■m_Name = "#STR_paint0";
■■m_IsInstaRecipe = false;//should this recipe be performed instantly without animation
■■m_AnimationLength = 1.5;//animation length in relative time units
■■m_Specialty = 0.01;// value > 0 for roughness, value < 0 for precision
■■
■■
■■//conditions
■■m_MinDamageIngredient[0] = -1;//-1 = disable check
■■m_MaxDamageIngredient[0] = 3;//-1 = disable check
■■
■■m_MinQuantityIngredient[0] = 25;//-1 = disable check
■■m_MaxQuantityIngredient[0] = -1;//-1 = disable check
■■
■■m_MinDamageIngredient[1] = -1;//-1 = disable check
■■m_MaxDamageIngredient[1] = 3;//-1 = disable check
■■
■■m_MinQuantityIngredient[1] = -1;//-1 = disable check
■■m_MaxQuantityIngredient[1] = -1;//-1 = disable check
■■//----------------------------------------------------------------------------------------------------------------------
■■
■■//INGREDIENTS
■■//ingredient 1
■■InsertIngredient(0,"Spraycan_ColorBase");//you can insert multiple ingredients this way
■■
■■m_IngredientAddHealth[0] = 0;// 0 = do nothing
■■m_IngredientSetHealth[0] = -1; // -1 = do nothing
■■m_IngredientAddQuantity[0] = -25;// 0 = do nothing
■■m_IngredientDestroy[0] = false;//true = destroy, false = do nothing
■■m_IngredientUseSoftSkills[0] = true;// set 'true' to allow modification of the values by softskills on this in
■■
■■//ingredient 2
■■InsertIngredient(1,"M4_PlasticHndgrd");//you can insert multiple ingredients this way
■■
■■m_IngredientAddHealth[1] = 0;// 0 = do nothing
■■m_IngredientSetHealth[1] = -1; // -1 = do nothing
■■m_IngredientAddQuantity[1] = 0;// 0 = do nothing
■■m_IngredientDestroy[1] = false;// false = do nothing
■■m_IngredientUseSoftSkills[1] = false;// set 'true' to allow modification of the values by softskills on this i
■■//----------------------------------------------------------------------------------------------------------------------
■■
■■//result1
■■/*
■■AddResult("M4_PlasticHndgrd_");//add results here

■■m_ResultSetFullQuantity[0] = false;//true = set full quantity, false = do nothing
■■m_ResultSetQuantity[0] = -1;//-1 = do nothing
■■m_ResultSetHealth[0] = -1;//-1 = do nothing
■■m_ResultInheritsHealth[0] = 1;// (value) == -1 means do nothing; a (value) >= 0 means this result will in
■■m_ResultInheritsColor[0] = 0;// (value) == -1 means do nothing; a (value) >= 0 means this result classn
■■m_ResultToInventory[0] = -1;//(value) == -2 spawn result on the ground;(value) == -1 place anywhere in
■■m_ResultUseSoftSkills[0] = false;// set 'true' to allow modification of the values by softskills on this resu
■■m_ResultReplacesIngredient[0] = -1;// value == -1 means do nothing; a value >= 0 means this result wi
■■*/
■}
```

```
// ---------------------------------------------
// ------ PaintM4RISHndgrd.c - START --------------------
// ---------------------------------------------


class PaintM4RISHndgrd extends RecipeBase
{■
■override void Init()
■{
■■m_Name = "#STR_paint0";
■■m_IsInstaRecipe = false;//should this recipe be performed instantly without animation
■■m_AnimationLength = 1.5;//animation length in relative time units
■■m_Specialty = 0.01;// value > 0 for roughness, value < 0 for precision
■■
■■
■■//conditions
■■m_MinDamageIngredient[0] = -1;//-1 = disable check
■■m_MaxDamageIngredient[0] = 3;//-1 = disable check
■■
■■m_MinQuantityIngredient[0] = 25;//-1 = disable check
■■m_MaxQuantityIngredient[0] = -1;//-1 = disable check
■■
■■m_MinDamageIngredient[1] = -1;//-1 = disable check
■■m_MaxDamageIngredient[1] = 3;//-1 = disable check
■■
■■m_MinQuantityIngredient[1] = -1;//-1 = disable check
■■m_MaxQuantityIngredient[1] = -1;//-1 = disable check
■■//----------------------------------------------------------------------------------------------------------------------
■■
■■//INGREDIENTS
■■//ingredient 1
■■InsertIngredient(0,"Spraycan_ColorBase");//you can insert multiple ingredients this way
■■
■■m_IngredientAddHealth[0] = 0;// 0 = do nothing
■■m_IngredientSetHealth[0] = -1; // -1 = do nothing
■■m_IngredientAddQuantity[0] = -25;// 0 = do nothing
■■m_IngredientDestroy[0] = false;//true = destroy, false = do nothing
■■m_IngredientUseSoftSkills[0] = true;// set 'true' to allow modification of the values by softskills on this in
■■
■■//ingredient 2
■■InsertIngredient(1,"M4_RISHndgrd");//you can insert multiple ingredients this way
■■
■■m_IngredientAddHealth[1] = 0;// 0 = do nothing
■■m_IngredientSetHealth[1] = -1; // -1 = do nothing
■■m_IngredientAddQuantity[1] = 0;// 0 = do nothing
■■m_IngredientDestroy[1] = false;// false = do nothing
■■m_IngredientUseSoftSkills[1] = false;// set 'true' to allow modification of the values by softskills on this i
■■//----------------------------------------------------------------------------------------------------------------------
■■
■■//result1
■■/*
■■AddResult("M4_RISHndgrd_");//add results here

■■m_ResultSetFullQuantity[0] = false;//true = set full quantity, false = do nothing
■■m_ResultSetQuantity[0] = -1;//-1 = do nothing
■■m_ResultSetHealth[0] = -1;//-1 = do nothing
■■m_ResultInheritsHealth[0] = 1;// (value) == -1 means do nothing; a (value) >= 0 means this result will in
■■m_ResultInheritsColor[0] = 0;// (value) == -1 means do nothing; a (value) >= 0 means this result classn
■■m_ResultToInventory[0] = -1;//(value) == -2 spawn result on the ground;(value) == -1 place anywhere i
■■m_ResultUseSoftSkills[0] = false;// set 'true' to allow modification of the values by softskills on this resu
■■m_ResultReplacesIngredient[0] = -1;// value == -1 means do nothing; a value >= 0 means this result wi
■■*/
■}
```

```
// ---------------------------------------------
// ------ PaintMosin.c - START --------------------
// ---------------------------------------------


class PaintMosin extends RecipeBase
{■
■override void Init()
■{
■■m_Name = "#STR_paint0";
■■m_IsInstaRecipe = false;//should this recipe be performed instantly without animation
■■m_AnimationLength = 1.5;//animation length in relative time units
■■m_Specialty = 0.01;// value > 0 for roughness, value < 0 for precision
■■
■■
■■//conditions
■■m_MinDamageIngredient[0] = -1;//-1 = disable check
■■m_MaxDamageIngredient[0] = 3;//-1 = disable check
■■
■■m_MinQuantityIngredient[0] = 25;//-1 = disable check
■■m_MaxQuantityIngredient[0] = -1;//-1 = disable check
■■
■■m_MinDamageIngredient[1] = -1;//-1 = disable check
■■m_MaxDamageIngredient[1] = 3;//-1 = disable check
■■
■■m_MinQuantityIngredient[1] = -1;//-1 = disable check
■■m_MaxQuantityIngredient[1] = -1;//-1 = disable check
■■//----------------------------------------------------------------------------------------------------------------------
■■
■■//INGREDIENTS
■■//ingredient 1
■■InsertIngredient(0,"Spraycan_ColorBase");//you can insert multiple ingredients this way
■■
■■m_IngredientAddHealth[0] = 0;// 0 = do nothing
■■m_IngredientSetHealth[0] = -1; // -1 = do nothing
■■m_IngredientAddQuantity[0] = -25;// 0 = do nothing
■■m_IngredientDestroy[0] = false;//true = destroy, false = do nothing
■■m_IngredientUseSoftSkills[0] = true;// set 'true' to allow modification of the values by softskills on this in
■■
■■//ingredient 2
■■InsertIngredient(1,"Mosin9130");//you can insert multiple ingredients this way
■■InsertIngredient(1,"Mosin9130_Black");//you can insert multiple ingredients this way
■■InsertIngredient(1,"Mosin9130_Green");//you can insert multiple ingredients this way
■■InsertIngredient(1,"Mosin9130_Camo");//you can insert multiple ingredients this way
■■
■■m_IngredientAddHealth[1] = 0;// 0 = do nothing
■■m_IngredientSetHealth[1] = -1; // -1 = do nothing
■■m_IngredientAddQuantity[1] = 0;// 0 = do nothing
■■m_IngredientDestroy[1] = false;// false = do nothing
■■m_IngredientUseSoftSkills[1] = false;// set 'true' to allow modification of the values by softskills on this i
■■//----------------------------------------------------------------------------------------------------------------------
■■
■■//result1
■■/*
■■AddResult("Mosin9130_");//add results here

■■m_ResultSetFullQuantity[0] = false;//true = set full quantity, false = do nothing
■■m_ResultSetQuantity[0] = -1;//-1 = do nothing
■■m_ResultSetHealth[0] = -1;//-1 = do nothing
■■m_ResultInheritsHealth[0] = 1;// (value) == -1 means do nothing; a (value) >= 0 means this result will in
■■m_ResultInheritsColor[0] = 0;// (value) == -1 means do nothing; a (value) >= 0 means this result classn
■■m_ResultToInventory[0] = -1;//(value) == -2 spawn result on the ground;(value) == -1 place anywhere in
■■m_ResultUseSoftSkills[0] = false;// set 'true' to allow modification of the values by softskills on this resu
```

```
// ---------------------------------------------
// ------ PaintMosinCamoBlack.c - START --------------------
// ---------------------------------------------


class PaintMosinCamoBlack extends RecipeBase
{■
■override void Init()
■{
■■m_Name = "#STR_paint0";
■■m_IsInstaRecipe = false;//should this recipe be performed instantly without animation
■■m_AnimationLength = 1.5;//animation length in relative time units
■■m_Specialty = 0.01;// value > 0 for roughness, value < 0 for precision
■■
■■
■■//conditions
■■m_MinDamageIngredient[0] = -1;//-1 = disable check
■■m_MaxDamageIngredient[0] = 3;//-1 = disable check
■■
■■m_MinQuantityIngredient[0] = 25;//-1 = disable check
■■m_MaxQuantityIngredient[0] = -1;//-1 = disable check
■■
■■m_MinDamageIngredient[1] = -1;//-1 = disable check
■■m_MaxDamageIngredient[1] = 3;//-1 = disable check
■■
■■m_MinQuantityIngredient[1] = -1;//-1 = disable check
■■m_MaxQuantityIngredient[1] = -1;//-1 = disable check
■■//----------------------------------------------------------------------------------------------------------------------
■■
■■//INGREDIENTS
■■//ingredient 1
■■InsertIngredient(0,"Spraycan_Black");//you can insert multiple ingredients this way
■■
■■m_IngredientAddHealth[0] = 0;// 0 = do nothing
■■m_IngredientSetHealth[0] = -1; // -1 = do nothing
■■m_IngredientAddQuantity[0] = -25;// 0 = do nothing
■■m_IngredientDestroy[0] = false;//true = destroy, false = do nothing
■■m_IngredientUseSoftSkills[0] = true;// set 'true' to allow modification of the values by softskills on this in
■■
■■//ingredient 2
■■InsertIngredient(1,"Mosin9130_Green");//you can insert multiple ingredients this way
■■
■■m_IngredientAddHealth[1] = 0;// 0 = do nothing
■■m_IngredientSetHealth[1] = -1; // -1 = do nothing
■■m_IngredientAddQuantity[1] = 0;// 0 = do nothing
■■m_IngredientDestroy[1] = false;// false = do nothing
■■m_IngredientUseSoftSkills[1] = false;// set 'true' to allow modification of the values by softskills on this i
■■//----------------------------------------------------------------------------------------------------------------------
■■
■■//result1
■■/*
■■AddResult("Mosin9130_Camo");//add results here

■■m_ResultSetFullQuantity[0] = false;//true = set full quantity, false = do nothing
■■m_ResultSetQuantity[0] = -1;//-1 = do nothing
■■m_ResultSetHealth[0] = -1;//-1 = do nothing
■■m_ResultInheritsHealth[0] = 1;// (value) == -1 means do nothing; a (value) >= 0 means this result will in
■■m_ResultInheritsColor[0] = -1;// (value) == -1 means do nothing; a (value) >= 0 means this result classn
■■m_ResultToInventory[0] = -1;//(value) == -2 spawn result on the ground;(value) == -1 place anywhere in
■■m_ResultUseSoftSkills[0] = false;// set 'true' to allow modification of the values by softskills on this resu
■■m_ResultReplacesIngredient[0] = 1;// value == -1 means do nothing; a value >= 0 means this result wil
■■*/
■}
```

```
// ---------------------------------------------
// ------ PaintMosinCamoGreen.c - START --------------------
// ---------------------------------------------


class PaintMosinCamoGreen extends RecipeBase
{■
■override void Init()
■{
■■m_Name = "#STR_paint0";
■■m_IsInstaRecipe = false;//should this recipe be performed instantly without animation
■■m_AnimationLength = 1.5;//animation length in relative time units
■■m_Specialty = 0.01;// value > 0 for roughness, value < 0 for precision
■■
■■
■■//conditions
■■m_MinDamageIngredient[0] = -1;//-1 = disable check
■■m_MaxDamageIngredient[0] = 3;//-1 = disable check
■■
■■m_MinQuantityIngredient[0] = 25;//-1 = disable check
■■m_MaxQuantityIngredient[0] = -1;//-1 = disable check
■■
■■m_MinDamageIngredient[1] = -1;//-1 = disable check
■■m_MaxDamageIngredient[1] = 3;//-1 = disable check
■■
■■m_MinQuantityIngredient[1] = -1;//-1 = disable check
■■m_MaxQuantityIngredient[1] = -1;//-1 = disable check
■■//----------------------------------------------------------------------------------------------------------------------
■■
■■//INGREDIENTS
■■//ingredient 1
■■InsertIngredient(0,"Spraycan_Green");//you can insert multiple ingredients this way
■■
■■m_IngredientAddHealth[0] = 0;// 0 = do nothing
■■m_IngredientSetHealth[0] = -1; // -1 = do nothing
■■m_IngredientAddQuantity[0] = -25;// 0 = do nothing
■■m_IngredientDestroy[0] = false;//true = destroy, false = do nothing
■■m_IngredientUseSoftSkills[0] = true;// set 'true' to allow modification of the values by softskills on this in
■■
■■//ingredient 2
■■InsertIngredient(1,"Mosin9130_Black");//you can insert multiple ingredients this way
■■
■■m_IngredientAddHealth[1] = 0;// 0 = do nothing
■■m_IngredientSetHealth[1] = -1; // -1 = do nothing
■■m_IngredientAddQuantity[1] = 0;// 0 = do nothing
■■m_IngredientDestroy[1] = false;// false = do nothing
■■m_IngredientUseSoftSkills[1] = false;// set 'true' to allow modification of the values by softskills on this i
■■//----------------------------------------------------------------------------------------------------------------------
■■
■■//result1
■■/*
■■AddResult("Mosin9130_Camo");//add results here

■■m_ResultSetFullQuantity[0] = false;//true = set full quantity, false = do nothing
■■m_ResultSetQuantity[0] = -1;//-1 = do nothing
■■m_ResultSetHealth[0] = -1;//-1 = do nothing
■■m_ResultInheritsHealth[0] = 1;// (value) == -1 means do nothing; a (value) >= 0 means this result will in
■■m_ResultInheritsColor[0] = -1;// (value) == -1 means do nothing; a (value) >= 0 means this result classn
■■m_ResultToInventory[0] = -1;//(value) == -2 spawn result on the ground;(value) == -1 place anywhere in
■■m_ResultUseSoftSkills[0] = false;// set 'true' to allow modification of the values by softskills on this resu
■■m_ResultReplacesIngredient[0] = 1;// value == -1 means do nothing; a value >= 0 means this result wil
■■*/
■}
```

```
// ---------------------------------------------
// ------ PaintMotohelmet.c - START --------------------
// ---------------------------------------------


class PaintMotohelmet extends RecipeBase
{■
■override void Init()
■{
■■m_Name = "#STR_paint0";
■■m_IsInstaRecipe = false;//should this recipe be performed instantly without animation
■■m_AnimationLength = 1.5;//animation length in relative time units
■■m_Specialty = 0.01;// value > 0 for roughness, value < 0 for precision
■■
■■
■■//conditions
■■m_MinDamageIngredient[0] = -1;//-1 = disable check
■■m_MaxDamageIngredient[0] = 3;//-1 = disable check
■■
■■m_MinQuantityIngredient[0] = 25;//-1 = disable check
■■m_MaxQuantityIngredient[0] = -1;//-1 = disable check
■■
■■m_MinDamageIngredient[1] = -1;//-1 = disable check
■■m_MaxDamageIngredient[1] = 3;//-1 = disable check
■■
■■m_MinQuantityIngredient[1] = -1;//-1 = disable check
■■m_MaxQuantityIngredient[1] = -1;//-1 = disable check
■■//----------------------------------------------------------------------------------------------------------------------
■■
■■//INGREDIENTS
■■//ingredient 1
■■InsertIngredient(0,"Spraycan_ColorBase");//you can insert multiple ingredients this way
■■
■■m_IngredientAddHealth[0] = 0;// 0 = do nothing
■■m_IngredientSetHealth[0] = -1; // -1 = do nothing
■■m_IngredientAddQuantity[0] = -25;// 0 = do nothing
■■m_IngredientDestroy[0] = false;//true = destroy, false = do nothing
■■m_IngredientUseSoftSkills[0] = true;// set 'true' to allow modification of the values by softskills on this in
■■
■■//ingredient 2
■■InsertIngredient(1,"MotoHelmet_ColorBase");//you can insert multiple ingredients this way
■■
■■m_IngredientAddHealth[1] = 0;// 0 = do nothing
■■m_IngredientSetHealth[1] = -1; // -1 = do nothing
■■m_IngredientAddQuantity[1] = 0;// 0 = do nothing
■■m_IngredientDestroy[1] = false;// false = do nothing
■■m_IngredientUseSoftSkills[1] = false;// set 'true' to allow modification of the values by softskills on this i
■■//----------------------------------------------------------------------------------------------------------------------
■■
■■//result1
■■/*
■■AddResult("MotoHelmet_");//add results here

■■m_ResultSetFullQuantity[0] = false;//true = set full quantity, false = do nothing
■■m_ResultSetQuantity[0] = -1;//-1 = do nothing
■■m_ResultSetHealth[0] = -1;//-1 = do nothing
■■m_ResultInheritsHealth[0] = 1;// (value) == -1 means do nothing; a (value) >= 0 means this result will in
■■m_ResultInheritsColor[0] = 0;// (value) == -1 means do nothing; a (value) >= 0 means this result classn
■■m_ResultToInventory[0] = -1;//(value) == -2 spawn result on the ground;(value) == -1 place anywhere in
■■m_ResultUseSoftSkills[0] = false;// set 'true' to allow modification of the values by softskills on this resu
■■m_ResultReplacesIngredient[0] = 1;// value == -1 means do nothing; a value >= 0 means this result wil
■■*/
■}
```

```
// ---------------------------------------------
// ------ PaintRuger1022.c - START --------------------
// ---------------------------------------------


class PaintRuger1022 extends RecipeBase
{■
■override void Init()
■{
■■m_Name = "#STR_paint0";
■■m_IsInstaRecipe = false;//should this recipe be performed instantly without animation
■■m_AnimationLength = 1.5;//animation length in relative time units
■■m_Specialty = 0.01;// value > 0 for roughness, value < 0 for precision
■■
■■
■■//conditions
■■m_MinDamageIngredient[0] = -1;//-1 = disable check
■■m_MaxDamageIngredient[0] = 3;//-1 = disable check
■■
■■m_MinQuantityIngredient[0] = 25;//-1 = disable check
■■m_MaxQuantityIngredient[0] = -1;//-1 = disable check
■■
■■m_MinDamageIngredient[1] = -1;//-1 = disable check
■■m_MaxDamageIngredient[1] = 3;//-1 = disable check
■■
■■m_MinQuantityIngredient[1] = -1;//-1 = disable check
■■m_MaxQuantityIngredient[1] = -1;//-1 = disable check
■■//----------------------------------------------------------------------------------------------------------------------
■■
■■//INGREDIENTS
■■//ingredient 1
■■InsertIngredient(0,"Spraycan_ColorBase");//you can insert multiple ingredients this way
■■
■■m_IngredientAddHealth[0] = 0;// 0 = do nothing
■■m_IngredientSetHealth[0] = -1; // -1 = do nothing
■■m_IngredientAddQuantity[0] = -25;// 0 = do nothing
■■m_IngredientDestroy[0] = false;//true = destroy, false = do nothing
■■m_IngredientUseSoftSkills[0] = true;// set 'true' to allow modification of the values by softskills on this in
■■
■■//ingredient 2
■■InsertIngredient(1,"Ruger1022");//you can insert multiple ingredients this way
■■
■■m_IngredientAddHealth[1] = 0;// 0 = do nothing
■■m_IngredientSetHealth[1] = -1; // -1 = do nothing
■■m_IngredientAddQuantity[1] = 0;// 0 = do nothing
■■m_IngredientDestroy[1] = false;// false = do nothing
■■m_IngredientUseSoftSkills[1] = false;// set 'true' to allow modification of the values by softskills on this i
■■//----------------------------------------------------------------------------------------------------------------------
■■
■■//result1
■■/*
■■AddResult("Ruger1022_");//add results here

■■m_ResultSetFullQuantity[0] = false;//true = set full quantity, false = do nothing
■■m_ResultSetQuantity[0] = -1;//-1 = do nothing
■■m_ResultSetHealth[0] = -1;//-1 = do nothing
■■m_ResultInheritsHealth[0] = 1;// (value) == -1 means do nothing; a (value) >= 0 means this result will in
■■m_ResultInheritsColor[0] = 0;// (value) == -1 means do nothing; a (value) >= 0 means this result classn
■■m_ResultToInventory[0] = -1;//(value) == -2 spawn result on the ground;(value) == -1 place anywhere in
■■m_ResultUseSoftSkills[0] = false;// set 'true' to allow modification of the values by softskills on this resu
■■m_ResultReplacesIngredient[0] = 1;// value == -1 means do nothing; a value >= 0 means this result wil
■■*/
■}
```

```
// ---------------------------------------------
// ------ PaintSawedoffMosin.c - START --------------------
// ---------------------------------------------


class PaintSawedoffMosin extends RecipeBase
{■
■override void Init()
■{
■■m_Name = "#STR_paint0";
■■m_IsInstaRecipe = false;//should this recipe be performed instantly without animation
■■m_AnimationLength = 1.5;//animation length in relative time units
■■m_Specialty = 0.01;// value > 0 for roughness, value < 0 for precision
■■
■■
■■//conditions
■■m_MinDamageIngredient[0] = -1;//-1 = disable check
■■m_MaxDamageIngredient[0] = 3;//-1 = disable check
■■
■■m_MinQuantityIngredient[0] = 25;//-1 = disable check
■■m_MaxQuantityIngredient[0] = -1;//-1 = disable check
■■
■■m_MinDamageIngredient[1] = -1;//-1 = disable check
■■m_MaxDamageIngredient[1] = 3;//-1 = disable check
■■
■■m_MinQuantityIngredient[1] = -1;//-1 = disable check
■■m_MaxQuantityIngredient[1] = -1;//-1 = disable check
■■//----------------------------------------------------------------------------------------------------------------------
■■
■■//INGREDIENTS
■■//ingredient 1
■■InsertIngredient(0,"Spraycan_ColorBase");//you can insert multiple ingredients this way
■■
■■m_IngredientAddHealth[0] = 0;// 0 = do nothing
■■m_IngredientSetHealth[0] = -1; // -1 = do nothing
■■m_IngredientAddQuantity[0] = -25;// 0 = do nothing
■■m_IngredientDestroy[0] = false;//true = destroy, false = do nothing
■■m_IngredientUseSoftSkills[0] = true;// set 'true' to allow modification of the values by softskills on this in
■■
■■//ingredient 2
■■InsertIngredient(1,"SawedoffMosin9130");//you can insert multiple ingredients this way
■■
■■m_IngredientAddHealth[1] = 0;// 0 = do nothing
■■m_IngredientSetHealth[1] = -1; // -1 = do nothing
■■m_IngredientAddQuantity[1] = 0;// 0 = do nothing
■■m_IngredientDestroy[1] = false;// false = do nothing
■■m_IngredientUseSoftSkills[1] = false;// set 'true' to allow modification of the values by softskills on this i
■■//----------------------------------------------------------------------------------------------------------------------
■■
■■//result1
■■/*
■■AddResult("SawedoffMosin9130_");//add results here

■■m_ResultSetFullQuantity[0] = false;//true = set full quantity, false = do nothing
■■m_ResultSetQuantity[0] = -1;//-1 = do nothing
■■m_ResultSetHealth[0] = -1;//-1 = do nothing
■■m_ResultInheritsHealth[0] = 1;// (value) == -1 means do nothing; a (value) >= 0 means this result will in
■■m_ResultInheritsColor[0] = 0;// (value) == -1 means do nothing; a (value) >= 0 means this result classn
■■m_ResultToInventory[0] = -1;//(value) == -2 spawn result on the ground;(value) == -1 place anywhere in
■■m_ResultUseSoftSkills[0] = false;// set 'true' to allow modification of the values by softskills on this resu
■■m_ResultReplacesIngredient[0] = 1;// value == -1 means do nothing; a value >= 0 means this result wil
■■*/
■}
```

```
// ---------------------------------------------
// ------ PaintSawedoffMosinCamoBlack.c - START --------------------
// ---------------------------------------------


class PaintSawedoffMosinCamoBlack extends RecipeBase
{■
■override void Init()
■{
■■m_Name = "#STR_paint0";
■■m_IsInstaRecipe = false;//should this recipe be performed instantly without animation
■■m_AnimationLength = 1.5;//animation length in relative time units
■■m_Specialty = 0.01;// value > 0 for roughness, value < 0 for precision
■■
■■
■■//conditions
■■m_MinDamageIngredient[0] = -1;//-1 = disable check
■■m_MaxDamageIngredient[0] = 3;//-1 = disable check
■■
■■m_MinQuantityIngredient[0] = 25;//-1 = disable check
■■m_MaxQuantityIngredient[0] = -1;//-1 = disable check
■■
■■m_MinDamageIngredient[1] = -1;//-1 = disable check
■■m_MaxDamageIngredient[1] = 3;//-1 = disable check
■■
■■m_MinQuantityIngredient[1] = -1;//-1 = disable check
■■m_MaxQuantityIngredient[1] = -1;//-1 = disable check
■■//----------------------------------------------------------------------------------------------------------------------
■■
■■//INGREDIENTS
■■//ingredient 1
■■InsertIngredient(0,"Spraycan_Black");//you can insert multiple ingredients this way
■■
■■m_IngredientAddHealth[0] = 0;// 0 = do nothing
■■m_IngredientSetHealth[0] = -1; // -1 = do nothing
■■m_IngredientAddQuantity[0] = -25;// 0 = do nothing
■■m_IngredientDestroy[0] = false;//true = destroy, false = do nothing
■■m_IngredientUseSoftSkills[0] = true;// set 'true' to allow modification of the values by softskills on this in
■■
■■//ingredient 2
■■InsertIngredient(1,"SawedoffMosin9130_Green");//you can insert multiple ingredients this way
■■
■■m_IngredientAddHealth[1] = 0;// 0 = do nothing
■■m_IngredientSetHealth[1] = -1; // -1 = do nothing
■■m_IngredientAddQuantity[1] = 0;// 0 = do nothing
■■m_IngredientDestroy[1] = false;// false = do nothing
■■m_IngredientUseSoftSkills[1] = false;// set 'true' to allow modification of the values by softskills on this i
■■//----------------------------------------------------------------------------------------------------------------------
■■
■■//result1
■■/*
■■AddResult("SawedoffMosin9130_Camo");//add results here

■■m_ResultSetFullQuantity[0] = false;//true = set full quantity, false = do nothing
■■m_ResultSetQuantity[0] = -1;//-1 = do nothing
■■m_ResultSetHealth[0] = -1;//-1 = do nothing
■■m_ResultInheritsHealth[0] = 1;// (value) == -1 means do nothing; a (value) >= 0 means this result will in
■■m_ResultInheritsColor[0] = -1;// (value) == -1 means do nothing; a (value) >= 0 means this result classn
■■m_ResultToInventory[0] = -1;//(value) == -2 spawn result on the ground;(value) == -1 place anywhere in
■■m_ResultUseSoftSkills[0] = false;// set 'true' to allow modification of the values by softskills on this resu
■■m_ResultReplacesIngredient[0] = 1;// value == -1 means do nothing; a value >= 0 means this result wil
■■*/
■}
```

```
// ---------------------------------------------
// ------ PaintSawedoffMosinCamoGreen.c - START --------------------
// ---------------------------------------------


class PaintSawedoffMosinCamoGreen extends RecipeBase
{■
■override void Init()
■{
■■m_Name = "#STR_paint0";
■■m_IsInstaRecipe = false;//should this recipe be performed instantly without animation
■■m_AnimationLength = 1.5;//animation length in relative time units
■■m_Specialty = 0.01;// value > 0 for roughness, value < 0 for precision
■■
■■
■■//conditions
■■m_MinDamageIngredient[0] = -1;//-1 = disable check
■■m_MaxDamageIngredient[0] = 3;//-1 = disable check
■■
■■m_MinQuantityIngredient[0] = 25;//-1 = disable check
■■m_MaxQuantityIngredient[0] = -1;//-1 = disable check
■■
■■m_MinDamageIngredient[1] = -1;//-1 = disable check
■■m_MaxDamageIngredient[1] = 3;//-1 = disable check
■■
■■m_MinQuantityIngredient[1] = -1;//-1 = disable check
■■m_MaxQuantityIngredient[1] = -1;//-1 = disable check
■■//----------------------------------------------------------------------------------------------------------------------
■■
■■//INGREDIENTS
■■//ingredient 1
■■InsertIngredient(0,"Spraycan_Green");//you can insert multiple ingredients this way
■■
■■m_IngredientAddHealth[0] = 0;// 0 = do nothing
■■m_IngredientSetHealth[0] = -1; // -1 = do nothing
■■m_IngredientAddQuantity[0] = -25;// 0 = do nothing
■■m_IngredientDestroy[0] = false;//true = destroy, false = do nothing
■■m_IngredientUseSoftSkills[0] = true;// set 'true' to allow modification of the values by softskills on this in
■■
■■//ingredient 2
■■InsertIngredient(1,"SawedoffMosin9130_Black");//you can insert multiple ingredients this way
■■
■■m_IngredientAddHealth[1] = 0;// 0 = do nothing
■■m_IngredientSetHealth[1] = -1; // -1 = do nothing
■■m_IngredientAddQuantity[1] = 0;// 0 = do nothing
■■m_IngredientDestroy[1] = false;// false = do nothing
■■m_IngredientUseSoftSkills[1] = false;// set 'true' to allow modification of the values by softskills on this i
■■//----------------------------------------------------------------------------------------------------------------------
■■
■■//result1
■■/*
■■AddResult("SawedoffMosin9130_Camo");//add results here

■■m_ResultSetFullQuantity[0] = false;//true = set full quantity, false = do nothing
■■m_ResultSetQuantity[0] = -1;//-1 = do nothing
■■m_ResultSetHealth[0] = -1;//-1 = do nothing
■■m_ResultInheritsHealth[0] = 1;// (value) == -1 means do nothing; a (value) >= 0 means this result will in
■■m_ResultInheritsColor[0] = -1;// (value) == -1 means do nothing; a (value) >= 0 means this result classn
■■m_ResultToInventory[0] = -1;//(value) == -2 spawn result on the ground;(value) == -1 place anywhere in
■■m_ResultUseSoftSkills[0] = false;// set 'true' to allow modification of the values by softskills on this resu
■■m_ResultReplacesIngredient[0] = 1;// value == -1 means do nothing; a value >= 0 means this result wil
■■*/
■}
```

```
// ---------------------------------------------
// ------ PaintSKS.c - START --------------------
// ---------------------------------------------


class PaintSKS extends RecipeBase
{■
■override void Init()
■{
■■m_Name = "#STR_paint0";
■■m_IsInstaRecipe = false;//should this recipe be performed instantly without animation
■■m_AnimationLength = 1.5;//animation length in relative time units
■■m_Specialty = 0.01;// value > 0 for roughness, value < 0 for precision
■■
■■
■■//conditions
■■m_MinDamageIngredient[0] = -1;//-1 = disable check
■■m_MaxDamageIngredient[0] = 3;//-1 = disable check
■■
■■m_MinQuantityIngredient[0] = 25;//-1 = disable check
■■m_MaxQuantityIngredient[0] = -1;//-1 = disable check
■■
■■m_MinDamageIngredient[1] = -1;//-1 = disable check
■■m_MaxDamageIngredient[1] = 3;//-1 = disable check
■■
■■m_MinQuantityIngredient[1] = -1;//-1 = disable check
■■m_MaxQuantityIngredient[1] = -1;//-1 = disable check
■■//----------------------------------------------------------------------------------------------------------------------
■■
■■//INGREDIENTS
■■//ingredient 1
■■InsertIngredient(0,"Spraycan_ColorBase");//you can insert multiple ingredients this way
■■
■■m_IngredientAddHealth[0] = 0;// 0 = do nothing
■■m_IngredientSetHealth[0] = -1; // -1 = do nothing
■■m_IngredientAddQuantity[0] = -25;// 0 = do nothing
■■m_IngredientDestroy[0] = false;//true = destroy, false = do nothing
■■m_IngredientUseSoftSkills[0] = true;// set 'true' to allow modification of the values by softskills on this in
■■
■■//ingredient 2
■■InsertIngredient(1,"SKS");//you can insert multiple ingredients this way
■■
■■m_IngredientAddHealth[1] = 0;// 0 = do nothing
■■m_IngredientSetHealth[1] = -1; // -1 = do nothing
■■m_IngredientAddQuantity[1] = 0;// 0 = do nothing
■■m_IngredientDestroy[1] = false;// false = do nothing
■■m_IngredientUseSoftSkills[1] = false;// set 'true' to allow modification of the values by softskills on this i
■■//----------------------------------------------------------------------------------------------------------------------
■■
■■//result1
■■/*
■■AddResult("SKS_");//add results here

■■m_ResultSetFullQuantity[0] = false;//true = set full quantity, false = do nothing
■■m_ResultSetQuantity[0] = -1;//-1 = do nothing
■■m_ResultSetHealth[0] = -1;//-1 = do nothing
■■m_ResultInheritsHealth[0] = 1;// (value) == -1 means do nothing; a (value) >= 0 means this result will in
■■m_ResultInheritsColor[0] = 0;// (value) == -1 means do nothing; a (value) >= 0 means this result classn
■■m_ResultToInventory[0] = -1;//(value) == -2 spawn result on the ground;(value) == -1 place anywhere in
■■m_ResultUseSoftSkills[0] = false;// set 'true' to allow modification of the values by softskills on this resu
■■m_ResultReplacesIngredient[0] = 1;// value == -1 means do nothing; a value >= 0 means this result wil
■■*/
■}
```

```
// ---------------------------------------------
// ------ PaintZSh3PilotHelmet.c - START --------------------
// ---------------------------------------------


class PaintZSh3PilotHelmet extends RecipeBase
{■
■override void Init()
■{
■■m_Name = "#STR_paint0";
■■m_IsInstaRecipe = false;//should this recipe be performed instantly without animation
■■m_AnimationLength = 1.5;//animation length in relative time units
■■m_Specialty = 0.01;// value > 0 for roughness, value < 0 for precision
■■
■■
■■//conditions
■■m_MinDamageIngredient[0] = -1;//-1 = disable check
■■m_MaxDamageIngredient[0] = 3;//-1 = disable check
■■
■■m_MinQuantityIngredient[0] = 25;//-1 = disable check
■■m_MaxQuantityIngredient[0] = -1;//-1 = disable check
■■
■■m_MinDamageIngredient[1] = -1;//-1 = disable check
■■m_MaxDamageIngredient[1] = 3;//-1 = disable check
■■
■■m_MinQuantityIngredient[1] = -1;//-1 = disable check
■■m_MaxQuantityIngredient[1] = -1;//-1 = disable check
■■//----------------------------------------------------------------------------------------------------------------------
■■
■■//INGREDIENTS
■■//ingredient 1
■■InsertIngredient(0,"Spraycan_ColorBase");//you can insert multiple ingredients this way
■■
■■m_IngredientAddHealth[0] = 0;// 0 = do nothing
■■m_IngredientSetHealth[0] = -1; // -1 = do nothing
■■m_IngredientAddQuantity[0] = -25;// 0 = do nothing
■■m_IngredientDestroy[0] = false;//true = destroy, false = do nothing
■■m_IngredientUseSoftSkills[0] = true;// set 'true' to allow modification of the values by softskills on this in
■■
■■//ingredient 2
■■InsertIngredient(1,"ZSh3PilotHelmet");//you can insert multiple ingredients this way
■■
■■m_IngredientAddHealth[1] = 0;// 0 = do nothing
■■m_IngredientSetHealth[1] = -1; // -1 = do nothing
■■m_IngredientAddQuantity[1] = 0;// 0 = do nothing
■■m_IngredientDestroy[1] = false;// false = do nothing
■■m_IngredientUseSoftSkills[1] = false;// set 'true' to allow modification of the values by softskills on this i
■■//----------------------------------------------------------------------------------------------------------------------
■■
■■//result1
■■/*
■■AddResult("ZSh3PilotHelmet_");//add results here

■■m_ResultSetFullQuantity[0] = false;//true = set full quantity, false = do nothing
■■m_ResultSetQuantity[0] = -1;//-1 = do nothing
■■m_ResultSetHealth[0] = -1;//-1 = do nothing
■■m_ResultInheritsHealth[0] = 1;// (value) == -1 means do nothing; a (value) >= 0 means this result will in
■■m_ResultInheritsColor[0] = 0;// (value) == -1 means do nothing; a (value) >= 0 means this result classn
■■m_ResultToInventory[0] = -1;//(value) == -2 spawn result on the ground;(value) == -1 place anywhere in
■■m_ResultUseSoftSkills[0] = false;// set 'true' to allow modification of the values by softskills on this resu
■■m_ResultReplacesIngredient[0] = 1;// value == -1 means do nothing; a value >= 0 means this result wil
■■*/
■}
```

```
// ---------------------------------------------
// ------ Paper.c - START --------------------
// ---------------------------------------------


class Paper extends ItemBase
{
■protected ref WrittenNoteData m_NoteContents;
■
■void Paper()
■{
■■m_NoteContents = new WrittenNoteData(this);
■}
■
■void ~Paper() {}
■
■override bool OnStoreLoad(ParamsReadContext ctx, int version)
■{
■■if (!super.OnStoreLoad(ctx, version))
■■■return false;
■■
■■if (version >= 108 && !ctx.Read(m_NoteContents))
■■■return false;
■■■■
■■return true;
■}
■
■override void OnStoreSave(ParamsWriteContext ctx)
■{
■■super.OnStoreSave(ctx);
■■
■■ctx.Write(m_NoteContents);■■
■}
■
■override WrittenNoteData GetWrittenNoteData()
■{
■■return m_NoteContents;
■}
■
■//=================================================================
■// IGNITION ACTION
■//=================================================================
■override bool HasFlammableMaterial()
■{
■■return true;
■}
■
■override bool CanBeIgnitedBy(EntityAI igniter = null)
■{
■■return !GetHierarchyParent();
■}
■
■override bool CanIgniteItem(EntityAI ignite_target = null)
■{
■■return false;
■}
■
■override void OnIgnitedTarget(EntityAI ignited_item) {}
■
■override void OnIgnitedThis(EntityAI fire_source)
■{
■■Fireplace.IgniteEntityAsFireplace(this, fire_source);
■}
```

```
// ---------------------------------------------
// ------ param.c - START --------------------
// ---------------------------------------------


/**
 * \defgroup Tools Tools
 * \desc Helpful functions & classes
 * @{
 */

//---------------------------------------------------------------------------
/**
\brief Base Param Class with no parameters. Used as general purpose parameter overloaded with Param
 */
class Param: Managed
{
■bool Serialize(Serializer ctx)
■{
■■return false;
■}
■■
■bool Deserializer(Serializer ctx)
■{
■■return false;
■}
};

/**
 \brief Param Class Template with one parameter.
 \n usage:
 @code
 Param paramA = new Param1<int>(55);
 Param paramB = new Param1<string>("Hello");
 @endcode
 */
class Param1<Class T1> extends Param
{
■T1 param1;

■void Param1(T1 p1)
■{
■■param1 = p1;
■}
■■
■override bool Serialize(Serializer ctx)
■{
■■return ctx.Write(param1);
■}
■■
■override bool Deserializer(Serializer ctx)
■{
■■return ctx.Read(param1);
■}
};

/**
 \brief Param Class Template with two parameters.
 \n usage:
 @code
 Param param = new Param2<float, string>(3.14, "Pi");
 @endcode
 */
```

```
// ---------------------------------------------
// ------ ParamedicJacket_ColorBase.c - START --------------------
// ---------------------------------------------


class ParamedicJacket_ColorBase extends Clothing
{
■override void SetActions()
■{
■■super.SetActions();
■■AddAction(ActionWringClothes);
■}
};
class ParamedicJacket_Blue extends ParamedicJacket_ColorBase {};
class ParamedicJacket_Crimson extends ParamedicJacket_ColorBase {};
class ParamedicJacket_Green extends ParamedicJacket_ColorBase {};


// ---------------------------------------------
// ------ ParamedicJacket_ColorBase.c - END ----------------------
// ---------------------------------------------


// ---------------------------------------------
// ------ ParamedicPants_ColorBase.c - START --------------------
// ---------------------------------------------


class ParamedicPants_ColorBase extends Clothing
{
■override void SetActions()
■{
■■super.SetActions();
■■AddAction(ActionWringClothes);
■}
};
class ParamedicPants_Blue extends ParamedicPants_ColorBase {};
class ParamedicPants_Crimson extends ParamedicPants_ColorBase {};
class ParamedicPants_Green extends ParamedicPants_ColorBase {};


// ---------------------------------------------
// ------ ParamedicPants_ColorBase.c - END ----------------------
// ---------------------------------------------


// ---------------------------------------------
// ------ Particle.c - START --------------------
// ---------------------------------------------


/**
\brief Legacy way of using particles in the game
*■\note They work okay when just needing to play a particle once every once in a while
*■■  But are extremely wasteful when it comes to playing multiple Particles at the same time
*/
class Particle : ParticleBase
{
■/** \name Generic data
■■Generic data for the Particle
■*/
■//@{
■//ID from ParticleList if assigned
```

```
// ---------------------------------------------
// ------ ParticleBase.c - START --------------------
// ---------------------------------------------


/**
\brief Invokers for ParticleBase events, called from events
■@code
■void EnableOnEndPrint(ParticleBase psrc)
■{
■■psrc.GetEvents().Event_OnParticleEnd.Insert(PrintParticleEnded);
■}
■
■void PrintParticleEnded(ParticleBase psrc)
■{
■■Print(string.Format("%1 ended.", psrc.GetDebugNameNative());
■}
■@endcode
*/
class ParticleEvents
{
■/**
■\brief Called when particle starts playing
■■\note Particle: Called when any Play method has been called
■■\note ParticleSource: Called when PlayParticleNative is successful
■*/
■ref ScriptInvoker Event_OnParticleStart = new ScriptInvoker();
■/**
■\brief Called when particle stops playing
■■\note Particle: Called when any Stop method has been called or when lifetime is over
■■\note ParticleSource: Called when StopParticleNative is successful, when the particle ends and when t
■*/
■ref ScriptInvoker Event_OnParticleStop = new ScriptInvoker();
■/**
■\brief Called when particle is reset
■■\note Particle: Not present, there is no reset functionality
■■\note ParticleSource: Called when ResetParticleNative is successful
■*/
■ref ScriptInvoker Event_OnParticleReset = new ScriptInvoker();
■/**
■\brief Called when particle ends
■■\note Particle: Called when lifetime is over and there are no emitors active anymore
■■\note ParticleSource: Called when particle ends naturally or after the particle is stopped and there are n
■■\warning Looped never ends naturally and need to be stopped
■*/
■ref ScriptInvoker Event_OnParticleEnd = new ScriptInvoker();
■/**
■\brief Called when particle receives a parent
■■\note Particle: Not present
■*/
■ref ScriptInvoker Event_OnParticleParented = new ScriptInvoker();
■/**
■\brief Called when particle is orphaned
■■\note Particle: Not present
■*/
■ref ScriptInvoker Event_OnParticleUnParented = new ScriptInvoker();
}

/**
\brief Engine base class with internal functionality
■\note Is NOT intended for direct creation
■■* As this does not have an actual particle
■■* Use either Particle or ParticleSource
```

```
// ---------------------------------------------
// ------ ParticleList.c - START --------------------
// ---------------------------------------------


// Register all particles below!

// Example how to register particles from a mod
/*
modded class ParticleList
{
■static const int MODDED_PARTICLE = RegisterParticle( "mod_folder/" , "my_modded_particle");
}
*/

class ParticleList
{
■ref static map<int, string> m_ParticlePaths; // Contains full paths to all particles. WARNING: Paths are w
■ref static map<string, int> m_ParticleNames; // Contains file NAME (without suffix) to id.
■
■static int m_lastID = 0;
■
■// REGISTER ALL PARTICLES BELOW:
■
■static const int INVALID■■■■■■= -1;
■static const int NONE■■■■■■■= 0; // 0 does not exist either, valid particle starts from 1
■static const int PARTICLE_TEST■■■■■= RegisterParticle("_test_orientation");
■static const int DEBUG_DOT■■■■■■= RegisterParticle("debug_dot");
■
■// FIREPLACE■
■// Normal fireplace
■static const int CAMP_FIRE_START■■■■= RegisterParticle("fire_small_camp_01_start");
■static const int CAMP_SMALL_FIRE ■■■■= RegisterParticle("fire_small_camp_01");
■static const int CAMP_NORMAL_FIRE■■■■= RegisterParticle("fire_medium_camp_01");
■static const int CAMP_SMALL_SMOKE ■■■■= RegisterParticle("smoke_small_camp_01");
■static const int CAMP_NORMAL_SMOKE■■■■= RegisterParticle("smoke_medium_camp_01");
■static const int CAMP_FIRE_END■■■■■= RegisterParticle("fire_small_camp_01_end");
■static const int CAMP_STEAM_2END■■■■= RegisterParticle("steam_medium_camp_2end");
■static const int CAMP_STEAM_EXTINGUISH_START■= RegisterParticle("default_01");
■static const int CAMP_STOVE_FIRE■■■■= RegisterParticle("fire_small_stove_01");
■static const int CAMP_STOVE_FIRE_START■■■= RegisterParticle("fire_small_stove_01_start");
■static const int CAMP_STOVE_FIRE_END■■■= RegisterParticle("fire_small_stove_01_end");
■static const int CAMP_NO_IGNITE_WIND■■■= RegisterParticle("fire_extinguish_wind");
■// Fireplace indoor
■static const int HOUSE_FIRE_START■■■■= RegisterParticle("fire_small_house_01_start");
■static const int HOUSE_SMALL_FIRE ■■■■= RegisterParticle("fire_small_house_01");
■static const int HOUSE_SMALL_SMOKE ■■■■= RegisterParticle("smoke_small_house_01");
■static const int HOUSE_NORMAL_FIRE■■■■= RegisterParticle("fire_medium_house_01");
■static const int HOUSE_NORMAL_SMOKE■■■■= RegisterParticle("smoke_medium_house_01");
■static const int HOUSE_FIRE_END■■■■■= RegisterParticle("fire_small_house_01_end");
■static const int HOUSE_FIRE_STEAM_2END■■■= RegisterParticle("steam_medium_house_2end");
■// Fireplace in barrel with holes
■static const int BARREL_FIRE_START■■■■= RegisterParticle("fire_small_barrel_01_start");
■static const int BARREL_SMALL_FIRE ■■■■= RegisterParticle("fire_small_barrel_01");
■static const int BARREL_SMALL_SMOKE ■■■= RegisterParticle("smoke_small_barrel_01");
■static const int BARREL_NORMAL_FIRE■■■■= RegisterParticle("fire_medium_barrel_01");
■static const int BARREL_NORMAL_SMOKE■■■= RegisterParticle("smoke_medium_barrel_01");
■static const int BARREL_FIRE_END■■■■= RegisterParticle("fire_small_barrel_01_end");
■static const int BARREL_FIRE_STEAM_2END■■■= RegisterParticle("steam_medium_camp_2end");
■// Fireplace in indoor oven
■static const int OVEN_FIRE_START■■■■= RegisterParticle("fire_small_oven_01_start");
■static const int OVEN_SMALL_FIRE ■■■■= RegisterParticle("fire_small_oven_01");
■static const int OVEN_NORMAL_FIRE■■■■= RegisterParticle("fire_medium_oven_01");
```

```
// ---------------------------------------------
// ------ ParticleManager.c - START --------------------
// ---------------------------------------------


//! Flags for ParticleManagerSettings
enum ParticleManagerSettingsFlags
{
■NONE,
■//! Particles will be locked to the index and not reused
■FIXED_INDEX,
■//! Allocation blocks the game until it is done
■BLOCKING,
■//! Disable the creation of virtual particles when the pool is still allocating
■DISABLE_VIRTUAL,
■//! Reuse stopped particles even if they are owned by something
■REUSE_OWNED,
};

//! Class simply to have easily modded constants
class ParticleManagerConstants
{
■/** \name Global ParticleManager settings
 ■■Settings applied to the global ParticleManager
■*/
■//@{
■static const int POOL_SIZE = 10000;
■static const int FLAGS = ParticleManagerSettingsFlags.NONE;
■//@}
}

//! Settings given to ParticleManager on creation (in ctor)
class ParticleManagerSettings
{
■/**
■\brief Constructor (ctor)
■■\param poolSize \p int Size of pool (amount of created and reserved particles)
■■\param flags \p int ParticleManagerSettingsFlags
■*/
■void ParticleManagerSettings(int poolSize, int flags = ParticleManagerSettingsFlags.NONE)
■{
■}
■
■//! dtor
■void ~ParticleManagerSettings()
■{
■}
}

//! Invokers for ParticleManager events
class ParticleManagerEvents
{
■ref ScriptInvoker Event_OnAllocation = new ScriptInvoker();
■ref ScriptInvoker Event_OnAllocationEnd = new ScriptInvoker();
}

//! Has a fixed pool of precreated and reserved particles
class ParticleManager : Managed
{
■//! Static ParticleManager
■private static ref ParticleManager g_ParticleManager;

■//! Access to the static ParticleManager
```

```
// ----------------------------------------------
// ------ ParticleSource.c - START --------------------
// ----------------------------------------------


//! Flags to pass to ParticleSource.SetParticleAutoDestroyFlags
enum ParticleAutoDestroyFlags
{
■//! No automatic destroying
■NONE,
■//! Destroy when the Particle ends (looping particle never ends)
■ON_END,
■//! Destroy when particle stops
■ON_STOP,
■//! ON_END | ON_STOP
■ALL,
}

//! Flags to pass to ParticleSource.PlayParticleEx
enum PlayParticleFlags
{
■//! No flags
■NONE,■■
■// Is just a placeholder for now
}

//! Flags to pass to ParticleSource.StopParticle
enum StopParticleFlags
{
■//! No flags
■NONE,
■//! Reset state after stopping
■RESET,
■/**
■\brief Flag will make the particle stop immediately, taking it out of simulation and clearing VISIBLE flag
■*■\note By default the particle will gradually fade
■*■\note Note that when using IMMEDIATE, it is possible to call PlayParticle and it will resume from the m
■*■■  Which is not possible by default, as the gradual fade is accomplished by nulling the lifetime
■*/
■IMMEDIATE,
■//! Is default behaviour, but can be used in conjuction with IMMEDIATE which hides it when this flag is no
■VISIBLE,
■//! (SPF_IMMEDIATE | SPF_VISIBLE) "Freezes" the particle while keeping it visible
■PAUSE,
}

//! Mode for GetParticle
enum EGetParticleMode
{
■//! Full path with ext ■■("graphics/particles/smoking_barrel_small.ptc")
■FULL,
■//! Full path without ext ■("graphics/particles/smoking_barrel_small")
■NO_EXT,
■//! Filename only ■■■("smoking_barrel_small")
■FILE,
}

enum ParticlePropertiesFlags
{
■NONE,
■//! Makes the particle start playing immediately after being created
■PLAY_ON_CREATION,
■//! Only applicable when there is a parent, this will force the localOri to be in world space instead of local
```

```
// ---------------------------------------------
// ------ ParticleTest.c - START --------------------
// ---------------------------------------------


// Particle test for Maxman

class ParticleTest extends ItemBase
{
■protected int■PARTICLE_PATH;
■protected Particle ■m_Particle;
■
■// Constructor
■void ParticleTest()
■{
■■if ( !GetGame().IsServer()  ||  !GetGame().IsMultiplayer() ) // Client side
■■{
■■■string path = ParticleList.GetPathToParticles();
■■■
■■■
■■■
■■■
■■■
■■■// Enter particle ID to play when ParticleTest spawns
■■■PARTICLE_PATH = ParticleList.DEBUG_DOT;
■■■
■■■// Alternatively, uncomment the second line and enter particle filename without *.ptc suffix instead. Ex
■■■string particle_filename = "menu_engine_fire";
■■■//PARTICLE_PATH = ParticleList.GetParticleID( path + particle_filename );
■■■
■■■
■■■
■■■
■■■
■■■m_Particle = ParticleManager.GetInstance().PlayOnObject( PARTICLE_PATH, this, GetPosition());
■■}
■}

■// Destructor
■override void EEDelete(EntityAI parent)
■{
■■super.EEDelete(parent);
■■
■■if (m_Particle  &&  GetGame()) // GetGame() is null when the game is being shut down
■■{
■■■m_Particle.Stop();
■■■GetGame().ObjectDelete(m_Particle);
■■}
■}
}


// ---------------------------------------------
// ------ ParticleTest.c - END ----------------------
// ---------------------------------------------


// ---------------------------------------------
// ------ PartyLight.c - START --------------------
// ---------------------------------------------


class PartyLight extends PointLightBase
```

```
// ---------------------------------------------
// ------ PartyTent.c - START --------------------
// ---------------------------------------------


class PartyTent extends TentBase
{
■void PartyTent()
■{
■■m_ToggleAnimations.Insert( new ToggleAnimations("Door1o", "Door1c", OPENING_0), 0 );
■■m_ToggleAnimations.Insert( new ToggleAnimations("Door2o", "Door2c", OPENING_1), 0 );
■■m_ToggleAnimations.Insert( new ToggleAnimations("Door3o", "Door3c", OPENING_2), 0 );
■■m_ToggleAnimations.Insert( new ToggleAnimations("Door4o", "Door4c", OPENING_3), 0 );
■■m_ToggleAnimations.Insert( new ToggleAnimations("Door5o", "Door5c", OPENING_4), 0 );
■■m_ToggleAnimations.Insert( new ToggleAnimations("Door6o", "Door6c", OPENING_5), 0 );
■■
■■m_ShowAnimationsWhenPitched.Insert( "Body" );
■■m_ShowAnimationsWhenPitched.Insert( "Pack" );
■■/*m_ShowAnimationsWhenPitched.Insert( "Door1o" );
■■m_ShowAnimationsWhenPitched.Insert( "Door2o" );
■■m_ShowAnimationsWhenPitched.Insert( "Door3o" );
■■m_ShowAnimationsWhenPitched.Insert( "Door4o" );
■■m_ShowAnimationsWhenPitched.Insert( "Door5o" );
■■m_ShowAnimationsWhenPitched.Insert( "Door6o" );*/
■■
■■m_ShowAnimationsWhenPacked.Insert( "Inventory" );
■■
■■m_HalfExtents = Vector(1.3, 0.35, 2.7);
■}
■
■override string GetSoundOpen()
■{
■■return "LargeTent_Door_Open_SoundSet";
■}
■
■override string GetSoundClose()
■{
■■return "LargeTent_Door_Close_SoundSet";
■}
■
■override string GetSoundOpenWindow()
■{
■■return "LargeTent_Window_Open_SoundSet";
■}
■
■override string GetSoundCloseWindow()
■{
■■return "LargeTent_Window_Close_SoundSet";
■}
■■
■override bool HasClutterCutter()
■{
■■return true;
■}
■
■override string GetClutterCutter()
■{
■■return "LargeTentClutterCutter"; //TODO add custom clutter cutter?
■}
■
■//================================================================
■// ADVANCED PLACEMENT
■//================================================================
// ----------------------------------------------
// ------ PatchItem.c - START --------------------
// ----------------------------------------------


class PatchItem extends RecipeBase
{■
■override void Init()
■{
■■m_Name = "#STR_PatchItem0";
■■m_IsInstaRecipe = false;//should this recipe be performed instantly without animation
■■m_AnimationLength = 1;//animation length in relative time units
■■m_Specialty = -0.02;// value > 0 for roughness, value < 0 for precision
■■
■■
■■//conditions
■■m_MinDamageIngredient[0] = -1;//-1 = disable check
■■m_MaxDamageIngredient[0] = 3;//-1 = disable check
■■
■■m_MinQuantityIngredient[0] = -1;//-1 = disable check
■■m_MaxQuantityIngredient[0] = -1;//-1 = disable check
■■
■■m_MinDamageIngredient[1] = 1;//-1 = disable check
■■m_MaxDamageIngredient[1] = 3;//-1 = disable check
■■
■■m_MinQuantityIngredient[1] = -1;//-1 = disable check
■■m_MaxQuantityIngredient[1] = -1;//-1 = disable check
■■//----------------------------------------------------------------------------------------------------------------------
■■
■■//INGREDIENTS
■■//ingredient 1
■■InsertIngredient(0,"LeatherSewingKit");//you can insert multiple ingredients this way
■■InsertIngredient(0,"SewingKit");//you can insert multiple ingredients this way
■■InsertIngredient(0,"TireRepairKit");//you can insert multiple ingredients this way
■■
■■m_IngredientAddHealth[0] = 0;// 0 = do nothing
■■m_IngredientSetHealth[0] = -1; // -1 = do nothing
■■m_IngredientAddQuantity[0] = 0;// 0 = do nothing
■■m_IngredientDestroy[0] = false;//true = destroy, false = do nothing
■■m_IngredientUseSoftSkills[0] = true;// set 'true' to allow modification of the values by softskills on this in
■■
■■//ingredient 2
■■InsertIngredient(1,"Inventory_Base");//you can insert multiple ingredients this way
■■
■■m_IngredientAddHealth[1] = 0;// 0 = do nothing
■■m_IngredientSetHealth[1] = -1; // -1 = do nothing
■■m_IngredientAddQuantity[1] = 0;// 0 = do nothing
■■m_IngredientDestroy[1] = false;// false = do nothing
■■m_IngredientUseSoftSkills[1] = false;// set 'true' to allow modification of the values by softskills on this i
■■
■■//----------------------------------------------------------------------------------------------------------------------
■■
■■//result1
■■//AddResult("");//add results here

■■m_ResultSetFullQuantity[0] = false;//true = set full quantity, false = do nothing
■■m_ResultSetQuantity[0] = -1;//-1 = do nothing
■■m_ResultSetHealth[0] = -1;//-1 = do nothing
■■m_ResultInheritsHealth[0] = -1;// (value) == -1 means do nothing; a (value) >= 0 means this result will i
■■m_ResultInheritsColor[0] = -1;// (value) == -1 means do nothing; a (value) >= 0 means this result classn
■■m_ResultToInventory[0] = -2;//(value) == -2 spawn result on the ground;(value) == -1 place anywhere in
■■m_ResultUseSoftSkills[0] = false;// set 'true' to allow modification of the values by softskills on this resu
■■m_ResultReplacesIngredient[0] = -1;// value == -1 means do nothing; a value >= 0 means this result wi
```

```
// ---------------------------------------------
// ------ PBOAPI.c - START --------------------
// ---------------------------------------------


class PBOAPI
{
■proto native owned string GetPBOVersion(string openName);
}

static proto native PBOAPI GetPBOAPI();


// ---------------------------------------------
// ------ PBOAPI.c - END ----------------------
// ---------------------------------------------


// ---------------------------------------------
// ------ PeachesCan.c - START --------------------
// ---------------------------------------------


class PeachesCan : Edible_Base
{
■override void Open()
■{
■■//super.Open();
■■ReplaceEdibleWithNew("PeachesCan_Opened");
■}
■
■override bool IsOpen()
■{
■■return false;
■}
}



// ---------------------------------------------
// ------ PeachesCan.c - END ----------------------
// ---------------------------------------------


// ---------------------------------------------
// ------ PeachesCan_Opened.c - START --------------------
// ---------------------------------------------


class PeachesCan_Opened: Edible_Base
{
■override bool CanDecay()
■{
■■return true;
■}
■
■override bool CanProcessDecay()
■{
■■return !(GetAgents() & eAgents.FOOD_POISON);
■}
■
■override void SetActions()
■{
```

```
// ---------------------------------------------
// ------ Pear.c - START --------------------
// ---------------------------------------------


class Pear : Edible_Base
{
■override bool CanBeCooked()
■{
■■return true;
■}■■
■
■override bool CanBeCookedOnStick()
■{
■■return true;
■}■
■
■override bool IsFruit()
■{
■■return true;
■}
■
■override void SetActions()
■{
■■super.SetActions();
■■
■■AddAction(ActionForceFeed);
■■AddAction(ActionEatFruit);
■■AddAction(ActionCreateIndoorFireplace);
■■AddAction(ActionCreateIndoorOven);
■}
■
■override bool CanDecay()
■{
■■return true;
■}
■
■override void EEOnCECreate()
■{
■■int rand = Math.RandomInt(0,10);
■■
■■if ( rand > 6 )
■■{
■■■ChangeFoodStage( FoodStageType.ROTTEN );
■■■SetHealth( "", "", GetMaxHealth()*0.1 );
■■}
■■else if ( rand > 2 )
■■{
■■■ChangeFoodStage( FoodStageType.DRIED );
■■■SetHealth( "", "", GetMaxHealth()*0.4 );
■■}
■}
}



// ---------------------------------------------
// ------ Pear.c - END ---------------------
// ---------------------------------------------
// ---------------------------------------------
// ------ PeelPotato.c - START --------------------
// ---------------------------------------------


class PeelPotato extends RecipeBase
{■
■override void Init()
■{
■■m_Name = "#PeelPotato";
■■m_IsInstaRecipe = false;//should this recipe be performed instantly without animation
■■m_AnimationLength = 1;//animation length in relative time units
■■m_Specialty = 0.01;// value > 0 for roughness, value < 0 for precision
■■
■■m_AnywhereInInventory = false;//is this recipe valid even when neither of the items is in hands
■■//conditions
■■m_MinDamageIngredient[0] = -1;//-1 = disable check
■■m_MaxDamageIngredient[0] = 3;//-1 = disable check
■■
■■m_MinQuantityIngredient[0] = -1;//-1 = disable check
■■m_MaxQuantityIngredient[0] = -1;//-1 = disable check
■■
■■m_MinDamageIngredient[1] = -1;//-1 = disable check
■■m_MaxDamageIngredient[1] = 3;//-1 = disable check
■■
■■m_MinQuantityIngredient[1] = -1;//-1 = disable check
■■m_MaxQuantityIngredient[1] = -1;//-1 = disable check
■■//----------------------------------------------------------------------------------------------------------------------
■■
■■//INGREDIENTS
■■//ingredient 1
■■InsertIngredient(0,"PotatoSeed");//you can insert multiple ingredients this way
■■
■■m_IngredientAddHealth[0] = 0;// 0 = do nothing
■■m_IngredientSetHealth[0] = -1; // -1 = do nothing
■■m_IngredientAddQuantity[0] = 0;// 0 = do nothing
■■m_IngredientDestroy[0] = true;//true = destroy, false = do nothing
■■m_IngredientUseSoftSkills[0] = false;// set 'true' to allow modification of the values by softskills on this i
■■
■■//ingredient 2
■■InsertIngredient(1,"Sickle");
■■InsertIngredient(1,"KukriKnife");
■■InsertIngredient(1,"FangeKnife");
■■InsertIngredient(1,"Hacksaw");
■■InsertIngredient(1,"KitchenKnife");
■■InsertIngredient(1,"SteakKnife");
■■InsertIngredient(1,"StoneKnife");
■■InsertIngredient(1,"Cleaver");
■■InsertIngredient(1,"CombatKnife");
■■InsertIngredient(1,"HuntingKnife");
■■InsertIngredient(1,"Machete");
■■InsertIngredient(1,"CrudeMachete");
■■InsertIngredient(1,"OrientalMachete");
■■InsertIngredient(1,"AK_Bayonet");
■■InsertIngredient(1,"M9A1_Bayonet");
■■InsertIngredient(1,"Mosin_Bayonet");
■■InsertIngredient(1,"SKS_Bayonet");
■■InsertIngredient(1,"Wrench");
■■InsertIngredient(1,"Screwdriver");
■■InsertIngredient(1,"Hatchet");
■■InsertIngredient(1,"HandSaw");
■■InsertIngredient(1,"BoneKnife");
■■
```

```
// ---------------------------------------------
// ------ Pelt_Base.c - START --------------------
// ---------------------------------------------


class Pelt_Base extends ItemBase
{■
■override bool IsPeltBase()
■{
■■return true;
■}
■
■override void SetActions()
■{
■■super.SetActions();
■■
■■AddAction(ActionAttach);
■■AddAction(ActionDetach);
■}
}

class WildboarPelt: Pelt_Base {  };
class RabbitPelt: Pelt_Base {  };
class CowPelt: Pelt_Base {  };
class PigPelt: Pelt_Base {  };
class DeerPelt: Pelt_Base {  };
class GoatPelt: Pelt_Base {  };
class BearPelt: Pelt_Base {  };
class WolfPelt: Pelt_Base {  };
class SheepPelt: Pelt_Base {  };
class MouflonPelt: Pelt_Base {  };
class FoxPelt: Pelt_Base {  };


// ---------------------------------------------
// ------ Pelt_Base.c - END ----------------------
// ---------------------------------------------


// ---------------------------------------------
// ------ Pen_Black.c - START --------------------
// ---------------------------------------------





// ---------------------------------------------
// ------ Pen_Black.c - END ----------------------
// ---------------------------------------------


// ---------------------------------------------
// ------ Pen_Blue.c - START --------------------
// ---------------------------------------------





// ---------------------------------------------
// ------ Pen_Blue.c - END ----------------------
// ---------------------------------------------
// ---------------------------------------------
// ------ Pen_ColorBase.c - START --------------------
// ---------------------------------------------


class Pen_ColorBase: Inventory_Base
{
■override void SetActions()
■{
■■super.SetActions();
■■
■■AddAction(ActionWritePaper);
■}
};
class Pen_Black: Pen_ColorBase {};
class Pen_Red: Pen_ColorBase {};
class Pen_Green: Pen_ColorBase {};
class Pen_Blue: Pen_ColorBase {};


// ---------------------------------------------
// ------ Pen_ColorBase.c - END ----------------------
// ---------------------------------------------


// ---------------------------------------------
// ------ Pen_Green.c - START --------------------
// ---------------------------------------------





// ---------------------------------------------
// ------ Pen_Green.c - END ----------------------
// ---------------------------------------------


// ---------------------------------------------
// ------ Pen_Red.c - START --------------------
// ---------------------------------------------





// ---------------------------------------------
// ------ Pen_Red.c - END ----------------------
// ---------------------------------------------


// ---------------------------------------------
// ------ PepperSeedsPack.c - START --------------------
// ---------------------------------------------


class PepperSeedsPack extends SeedPackBase
{■
}


// ---------------------------------------------
// ------ PepperSeedsPack.c - END ----------------------
// ---------------------------------------------
// ------ PersistentFlag.c - START --------------------
// ---------------------------------------------


// These are flags which are persistent for the player entity, ie. are saved when the player is disconnected a
// Normal persistence events OnStoreSave/OnStoreLoad are used to save/load these flags within a single
// !!! Every value needs to be a power of 2 !!!

enum PersistentFlag
{
■AREA_PRESENCE = 1;
■//ADDITIONAL_FLAG_1 = 2
■//ADDITIONAL_FLAG_2 = 4
■//ADDITIONAL_FLAG_3 = 8
■//ADDITIONAL_FLAG_4 = 16
}


// ---------------------------------------------
// ------ PersistentFlag.c - END ----------------------
// ---------------------------------------------


// ---------------------------------------------
// ------ PersonalLight.c - START --------------------
// ---------------------------------------------


class PersonalLight extends PointLightBase
{■
■void PersonalLight()
■{
■■SetVisibleDuringDaylight(false);
■■SetRadiusTo( 3 );
■■SetBrightnessTo(0.38);
■■SetDiffuseColor(0.2, 0.23, 0.25);
■■SetCastShadow(false);
■■EnableSpecular(false);
■■EnableLinear(true);
■■SetFlareVisible(false);
■}
■
■override void OnFrameLightSource(IEntity other, float timeSlice)
■{
■■if ( GetGame()  &&  IsEnabled() )
■■{
■■■vector pos = GetGame().GetCurrentCameraPosition();
■■■pos += Vector( 0, -0.4, 0 );
■■■
■■■SetPosition( pos );
■■}
■}
■
■// Experiment with dynamic range of Personal Light based on distance between camera and player's cha
■/*override void OnFrameLightSource(IEntity other, float timeSlice)
■{
■■if ( GetGame()  &&  IsEnabled() )
■■{
■■■vector pos = GetGame().GetCurrentCameraPosition();
■■■vector dir = GetGame().GetCurrentCameraDirection();
■■■
■■■SetPosition(pos);
```

```
// ---------------------------------------------
// ------ PersonalRadio.c - START --------------------
// ---------------------------------------------


class PersonalRadio extends TransmitterBase
{
■void PersonalRadio()
■{
■■SOUND_RADIO_TURNED_ON = "personalradio_staticnoise_SoundSet";
■}
■
■override void OnDebugSpawn()
■{
■■GetInventory().CreateInInventory("Battery9V");
■}
}



// ---------------------------------------------
// ------ PersonalRadio.c - END ----------------------
// ---------------------------------------------


// ---------------------------------------------
// ------ PetrolLighter.c - START --------------------
// ---------------------------------------------


class PetrolLighter extends ItemBase
{
■override bool CanIgniteItem( EntityAI ignite_target = NULL )
■{
■■if ( GetQuantity() > 0 )
■■■return true;
■■else
■■■return false;
■}
■
■override void OnIgnitedTarget( EntityAI ignited_item )
■{
■■if ( GetGame().IsServer() )
■■{
■■■AddQuantity( -0.5 );
■■}
■}
■
■override void OnIgnitedTargetFailed( EntityAI target_item )
■{
■■if ( GetGame().IsServer() )
■■{
■■■AddQuantity( -0.5 );
■■}
■}

■/*
■override bool IsTargetIgnitionSuccessful( EntityAI item_target = NULL )
■{
■}
■*/
■
■override void SetActions()
```

```
// ---------------------------------------------
// ------ PickAxe.c - START --------------------
// ---------------------------------------------


class Pickaxe extends ItemBase
{
■override bool CanMakeGardenplot()
■{
■■return true;
■}
■
■override void SetActions()
■{
■■super.SetActions();

■■AddAction(ActionTogglePlaceObject);
■■AddAction(ActionDigGardenPlot);
■■AddAction(ActionDismantleGardenPlot);
■■AddAction(ActionUnrestrainTarget);
■■AddAction(ActionMineRock);
■■AddAction(ActionDismantlePart);
■■AddAction(ActionBuildPart);
■■AddAction(ActionBuryBody);
■■AddAction(ActionBuryAshes);
■■AddAction(ActionDigWorms);
■■AddAction(ActionSkinning);
■■AddAction(ActionDigOutStash);
■■AddAction(ActionDigInStash);
■■AddAction(ActionCreateGreenhouseGardenPlot);
■}
}■


// ---------------------------------------------
// ------ PickAxe.c - END ----------------------
// ---------------------------------------------


// ---------------------------------------------
// ------ PigSteakMeat.c - START --------------------
// ---------------------------------------------


class PigSteakMeat extends Edible_Base
{
■override bool CanBeCooked()
■{
■■return true;
■}■■
■
■override bool CanBeCookedOnStick()
■{
■■return true;
■}
■
■override bool IsMeat()
■{
■■return true;
■}

■override bool CanDecay()
■{
```

```
// ---------------------------------------------
// ------ PileOfWoodenPlanks.c - START --------------------
// ---------------------------------------------


class PileOfWoodenPlanks extends ItemBase
{■
■void PileOfWoodenPlanks()
■{
■■if ( GetGame().IsServer() )
■■{
■■■SetAllowDamage(false);
■■}
■}
■
■// Shows/Hides selections. Call this in init or after every quantity change.
■void UpdateSelections()
■{
■■RemoveProxyPhysics( "stage_1" );
■■RemoveProxyPhysics( "stage_2" );
■■RemoveProxyPhysics( "stage_3" );
■■
■■// Show/Hide selections according to the current quantity
■■if ( this )
■■{
■■■float quantity = GetQuantity();
■■■float quantity_max = GetQuantityMax();
■■■
■■■if ( quantity > GetQuantityMax() *0.66 )
■■■{
■■■■// Show 3/3 amount of planks
■■■■ShowSelection ( "stage_3" );
■■■■HideSelection ( "stage_2" );
■■■■HideSelection ( "stage_1" );
■■■■
■■■■AddProxyPhysics( "stage_3" );
■■■}
■■■
■■■if ( quantity > quantity_max *0.33  &&  quantity <= quantity_max *0.66 )
■■■{
■■■■// Show 2/3 amount of planks
■■■■HideSelection ( "stage_3" );
■■■■ShowSelection ( "stage_2" );
■■■■HideSelection ( "stage_1" );
■■■■
■■■■AddProxyPhysics( "stage_2" );
■■■}
■■■
■■■if ( quantity > 0  &&  quantity <= quantity_max *0.33 )
■■■{
■■■■// Show 1/3 amount of planks
■■■■HideSelection ( "stage_3" );
■■■■HideSelection ( "stage_2" );
■■■■ShowSelection ( "stage_1" );
■■■■
■■■■AddProxyPhysics( "stage_1" );
■■■}
■■■
■■■if ( quantity == 0 )
■■■{
■■■■// Show 0 planks. Object should be deleted now.
■■■■HideSelection ( "stage_3" );
■■■■HideSelection ( "stage_2" );
```

```
// ---------------------------------------------
// ------ PillsNotfr.c - START --------------------
// ---------------------------------------------


class PillsNotfr: NotifierBase
{
■void PillsNotfr(NotifiersManager manager)
■{
■■m_Active = false;
■}

■override int GetNotifierType()
■{
■■return eNotifiers.NTF_PILLS;
■}

■override void DisplayBadge()
■{
■■
■■DisplayElementBadge dis_elm = DisplayElementBadge.Cast(GetVirtualHud().GetElement(eDisplayEle
■■
■■if( dis_elm )
■■{
■■■dis_elm.SetLevel(eBadgeLevel.FIRST);
■■}
■}

■override void HideBadge()
■{
■■
■■DisplayElementBadge dis_elm = DisplayElementBadge.Cast(GetVirtualHud().GetElement(eDisplayEle
■■
■■if( dis_elm )
■■{
■■■dis_elm.SetLevel(eBadgeLevel.NONE);
■■}
■}
};


// ---------------------------------------------
// ------ PillsNotfr.c - END ----------------------
// ---------------------------------------------


// ---------------------------------------------
// ------ PilotkaCap.c - START --------------------
// ---------------------------------------------


class PilotkaCap extends Clothing
{
■override void SetActions()
■{
■■super.SetActions();
■■AddAction(ActionWringClothes);
■}
}



// ---------------------------------------------
// ---------------------------------------------
// ------ Pipe.c - START --------------------
// ---------------------------------------------


class Pipe extends Inventory_Base
{
}


// ---------------------------------------------
// ------ Pipe.c - END ----------------------
// ---------------------------------------------


// ---------------------------------------------
// ------ PipeWrench.c - START --------------------
// ---------------------------------------------


class PipeWrench extends Inventory_Base
{
■override void SetActions()
■{
■■super.SetActions();
■■//AddAction(ActionDismantlePart);
■■//AddAction(ActionRepairCarEngine);
■■AddAction(ActionMineRock);
■}
}


// ---------------------------------------------
// ------ PipeWrench.c - END ----------------------
// ---------------------------------------------


// ---------------------------------------------
// ------ PistolAlt_Base.c - START --------------------
// ---------------------------------------------


class PistolAlt_Base extends Pistol_Base
{}


// ---------------------------------------------
// ------ PistolAlt_Base.c - END ----------------------
// ---------------------------------------------


// ---------------------------------------------
// ------ PistollightLight.c - START --------------------
// ---------------------------------------------


class PistollightLight extends SpotLightBase
{
■private static float m_DefaultBrightness = 5.25;
■private static float m_DefaultRadius = 18;
■private static float m_DefaultAngle = 100;
■
■
```

```
// ---------------------------------------------
// ------ PistolSuppressor.c - START --------------------
// ---------------------------------------------


class PistolSuppressor extends ItemSuppressor
{
■override bool CanPutAsAttachment( EntityAI parent )
■{
■■if(!super.CanPutAsAttachment(parent)) {return false;}
■■if ( parent.FindAttachmentBySlotName("suppressorImpro") == NULL && parent.FindAttachmentBySlotN
■■{
■■■return true;
■■}
■■return false;
■}
}


// ---------------------------------------------
// ------ PistolSuppressor.c - END ----------------------
// ---------------------------------------------


// ---------------------------------------------
// ------ Pistol_Base.c - START --------------------
// ---------------------------------------------



enum PistolAnimState
{
■DEFAULT ■■■= 0, ■///< default weapon state, closed and discharged
■OPENED_DISCHARGED ■= 1, ■///< opened and discharged
■CLOSED_CHARGED■■= 2,■///< closed and charged
■JAMMED■■■■= 3■■///< jammed weapon
};

// naming convention of the states respect following order:
// closed/opened | charged/discharged | bullet/nobullet | magazine/nomag
enum PistolStableStateID
{
■UNKNOWN■■■■=  0,
■CLO_DIS_BU0_MA0■■=  1,
■CLO_CHG_BU0_MA0■■=  2,
■CLO_CHG_BU1_MA0■■=  3,
■CLO_JAM_BU1_MA0■■=  4,
■OPE_DIS_BU0_MA0■■=  5,
■CLO_CHG_BU0_MA1■■=  6,
■CLO_DIS_BU0_MA1■■=  7,
■CLO_JAM_BU1_MA1■■=  8,
■CLO_CHG_BU1_MA1■■=  9,
■OPE_DIS_BU0_MA1■■= 10,
}

class Pistol_CLO_DIS_BU0_MA0 extends WeaponStableState
{
■override void OnEntry (WeaponEventBase e) { if (LogManager.IsWeaponLogEnable()) { wpnPrint("[wpns
■override void OnExit (WeaponEventBase e) { super.OnExit(e); if (LogManager.IsWeaponLogEnable()) { w
■override int GetCurrentStateID () { return PistolStableStateID.CLO_DIS_BU0_MA0; }
■override bool HasBullet () { return false; }
■override bool HasMagazine () { return false; }
■override bool IsJammed () { return false; }
```

```
// ---------------------------------------------
// ------ Pitchfork.c - START --------------------
// ---------------------------------------------


class Pitchfork extends ItemBase
{
■void Pitchfork()
■{
■}

■override void SetActions()
■{
■■super.SetActions();
■■AddAction(ActionClapBearTrapWithThisItem);
■}
}


// ---------------------------------------------
// ------ Pitchfork.c - END ----------------------
// ---------------------------------------------


// ---------------------------------------------
// ------ Plant.c - START --------------------
// ---------------------------------------------


typedef WoodBase PlantSuper;


// ---------------------------------------------
// ------ Plant.c - END ----------------------
// ---------------------------------------------


// ---------------------------------------------
// ------ PlantBase.c - START --------------------
// ---------------------------------------------


class PlantBase extends ItemBase
{
■// Plant states
■static const int STATE_DRY ■■■■■= 0;
■static const int STATE_GROWING ■■■■= 1;
■static const int STATE_MATURE ■■■■= 2;
■static const int STATE_SPOILED ■■■■= 3;

■private float ■m_SprayUsage; // How much spray is needed to stop infestation of plant
■
■private float ■m_InfestationChance;
■
■private int ■m_GrowthStagesCount;
■private int ■m_CropsCount;
■private bool ■m_HasCrops;
■private string ■m_CropsType;
■private float ■m_PlantMaterialMultiplier;
■
■private int ■m_PlantState;
■private int ■m_PlantStateIndex;
■private float ■m_CurrentPlantMaterialQuantity;
```

```
// ---------------------------------------------
// ------ PlantMaterial.c - START --------------------
// ---------------------------------------------


class PlantMaterialHealth
{
■string m_InfestedTex;
■string m_InfestedMat;
■string m_HealthyTex;
■string m_HealthyMat;
■
■void PlantMaterialHealth()
■{
■■m_InfestedTex = "";
■■m_InfestedMat = "";
■■m_HealthyTex = "";
■■m_HealthyMat = "";
■}
}



// ---------------------------------------------
// ------ PlantMaterial.c - END ----------------------
// ---------------------------------------------


// ---------------------------------------------
// ------ Plastic_Explosive.c - START --------------------
// ---------------------------------------------


class Plastic_Explosive : ExplosivesBase
{
■protected const string SLOT_TRIGGER ■■■■■= "TriggerRemoteDetonator_Receiver";
■protected const string ANIM_PHASE_TRIGGER_REMOTE ■■= "TriggerRemote";
■
■protected bool m_UsedAsCharge;
■
■protected ref RemotelyActivatedItemBehaviour m_RAIB;

■void Plastic_Explosive()
■{
■■m_RAIB = new RemotelyActivatedItemBehaviour(this);
■■
■■SetAmmoType("Plastic_Explosive_Ammo");
■■SetParticleExplosion(ParticleList.PLASTIC_EXPLOSION);

■■RegisterNetSyncVariableInt("m_RAIB.m_PairDeviceNetIdLow");
■■RegisterNetSyncVariableInt("m_RAIB.m_PairDeviceNetIdHigh");
■}
■
■override void EOnInit(IEntity other, int extra)
■{
■■LockTriggerSlots();
■}

■//! special behaviour - do not call super from ExplosivesBase
■override void EEKilled(Object killer)■
■{
■■//analytics (behaviour from EntityAI)
■■GetGame().GetAnalyticsServer().OnEntityKilled(killer, this);
```

```
// ---------------------------------------------
// ------ PlateCarrierHolster.c - START --------------------
// ---------------------------------------------


class PlateCarrierHolster extends Clothing {};


// ---------------------------------------------
// ------ PlateCarrierHolster.c - END ----------------------
// ---------------------------------------------


// ---------------------------------------------
// ------ PlateCarrierVest.c - START --------------------
// ---------------------------------------------


class PlateCarrierVest: Clothing
{
■override void OnWasAttached( EntityAI parent, int slot_id )
■{
■■super.OnWasAttached( parent, slot_id );
■■
■■if ( GetGame().IsServer() && parent.IsInherited( DayZInfected ) )
■■{
■■■float coef = Math.RandomFloatInclusive( 0.2, 0.4 );
■■■SetHealth01( "", "", coef);
■■}
■}
■
■//Debug menu Spawn Ground Special
■override void OnDebugSpawn()
■{
■■EntityAI entity;
■■if ( Class.CastTo(entity, this) )
■■{
■■■entity.GetInventory().CreateInInventory( "PlateCarrierPouches" );
■■■entity.GetInventory().CreateInInventory( "PlateCarrierHolster" );
■■}
■}
};


// ---------------------------------------------
// ------ PlateCarrierVest.c - END ----------------------
// ---------------------------------------------


// ---------------------------------------------
// ------ PlayerAgentPool.c - START --------------------
// ---------------------------------------------


class PlayerAgentPool
{
■ref map<int,float> m_VirusPool = new map<int,float>;
■ref array<int> m_VirusPoolArray = new array<int>;
■float m_LastTicked = 0;
■float m_TotalAgentCount;
■PlayerBase m_Player;
■int m_AgentMask;
■■
```

```
// ---------------------------------------------
// ------ PlayerBase.c - START --------------------
// ---------------------------------------------


class PlayerBase extends ManBase
{
■const int ■■■■■■SIMPLIFIED_SHOCK_CAP = 63;
■const int ■■■■■■SHAKE_LEVEL_MAX = 7;■■■■■
■private int■■■■■■m_LifeSpanState;
■private int■■■■■■m_LastShavedSeconds;
■private int■■■■■■m_BloodType;
■private bool■■■■■m_HasBloodTypeVisible;
■private bool■■■■■m_LiquidTendencyDrain; //client-side only - Obsolete
■private bool ■■■■■m_FlagRaisingTendency;
■private int■■■■■■m_HasBloodyHandsVisible;
■private int■■■■■■m_FaceCoveredForShaveLayers = 0;
■protected bool■■■■■m_HasHeatBuffer;
■protected bool ■■■■■m_PlayerLoaded;
■protected bool ■■■■■m_PlayerDisconnectProcessed;
■protected bool ■■■■■m_ProcessUIWarning;
■protected int ■■■■■m_LocalRefreshAnimStateIdx;
■protected int ■■■■■m_RefreshAnimStateIdx;
■private int ■■■■■m_StoreLoadVersion;
■const int ■■■■■■ACT_STORE_SAVE_VERSION = 4;
■protected int ■■■■■m_LifespanLevelLocal; //control variable for change calls
■protected int ■■■■■m_AnimCommandStarting; //signals the command that is about to start the next fra
■EUndergroundPresence■■■m_UndergroundPresence;
■
■private PluginPlayerStatus■■m_ModulePlayerStatus;
■PluginConfigEmotesProfile ■■m_ConfigEmotesProfile;
■private PluginLifespan ■■■m_ModuleLifespan;
■protected PluginPresenceNotifier  m_PresenceNotifier;
■
■protected ref UndergroundHandlerClient■m_UndergroundHandler;
■ref PlayerStats ■■■■m_PlayerStats;
■PluginRecipesManager ■■■m_ModuleRecipesManager;
■ref BleedingSourcesManagerServer ■m_BleedingManagerServer;
■ref BleedingSourcesManagerRemote ■m_BleedingManagerRemote;
■ref ModifiersManager■■■m_ModifiersManager;
■ref NotifiersManager ■■■m_NotifiersManager;
■ref protected ActionManagerBase■m_ActionManager;
■//ref PlayerLightManager ■■■m_PlayerLightManager;
■ref PlayerAgentPool■■■■m_AgentPool;
■int■■■■■■■■m_Agents;
■ref Environment■■■■■m_Environment;
■ref EmoteManager ■■■■m_EmoteManager;
//■ref VehicleManager ■■■■m_VehicleManager;
■ref SymptomManager■■■■m_SymptomManager;
■ref VirtualHud ■■■■■m_VirtualHud;
■ref StaminaHandler■■■■m_StaminaHandler;
■ref InjuryAnimationHandler■■m_InjuryHandler;
■ref ShockHandler■■■■m_ShockHandler; //New shock handler
■ref SoftSkillsManager■■■m_SoftSkillsManager;
■ref StanceIndicator■■■■m_StanceIndicator;
■ref TransferValues■■■■m_TrasferValues;
■ref DebugMonitorValues■■■m_DebugMonitorValues;
■const int ■■■■■■OVERLOAD_LIMIT = 30000;
■float ■■■■■■■m_CargoLoad;
■float■■■■■■■m_VisibilityCoef;
■float ■■■■■■■m_OriginalSlidePoseAngle;
■int■■■■■■■■m_SoundEvent;
■int■■■■■■■■m_SoundEventParam;
```

```
// ---------------------------------------------
// ------ PlayerBaseClient.c - START --------------------
// ---------------------------------------------


class PlayerBaseClient extends PlayerBase
{■
■static ScriptedLightBase ■m_PersonalLight;
■static bool ■■■■m_PersonalLightEnabledOnCurrentServer = false; // "disablePersonalLight" in server.c
■static bool ■■■■m_PersonalLightDisabledByDebug = false;
■static bool ■■■■m_PersonalLightIsSwitchedOn = true;
■
■//! Creates PL if it doesn't exist already.
■static void CreatePersonalLight()
■{
■■if (!m_PersonalLight && ( !GetGame().IsServer() || !GetGame().IsMultiplayer() ))
■■{
■■■m_PersonalLight = ScriptedLightBase.CreateLight(PersonalLight, "0 0 0");
■■}
■}
■
■/*
■override void OnRPC(PlayerIdentity sender, int rpc_type, ParamsReadContext ctx)
■{
■■super.OnRPC(sender, rpc_type, ctx);
■■
■■switch( rpc_type )
■■{
■■■case ERPCs.RPC_TOGGLE_PERSONAL_LIGHT:
■■■{
■■■■Param1<bool> is_enabled = new Param1<bool>(false);
■■■■
■■■■if (ctx.Read(is_enabled))
■■■■{
■■■■■m_PersonalLightEnabledOnCurrentServer = is_enabled.param1;
■■■■■UpdatePersonalLight();
■■■■}
■■■■
■■■■break;
■■■}
■■}
■}*/
■
■override void OnGameplayDataHandlerSync()
■{
■■super.OnGameplayDataHandlerSync();
■■m_PersonalLightEnabledOnCurrentServer = !CfgGameplayHandler.GetDisablePersonalLight();
■■UpdatePersonalLight();
■■UpdateHitDirectionValues();
■}
■
■
■//! Controls the ON/OFF switch of the Personal Light. PL will still shine only if the server allows it.
■static void SwitchPersonalLight(bool state)
■{
■■if ( !GetGame().IsServer() || !GetGame().IsMultiplayer() )
■■{
■■■m_PersonalLightIsSwitchedOn = state;
■■■UpdatePersonalLight();
■■}
■}
■
■//! Updates state of PL
```

```
// ---------------------------------------------
// ------ PlayerConstants.c - START --------------------
// ---------------------------------------------


class PlayerConstants
{
■static const float WEAPON_RAISE_BLEND_DELAY = 0.136;
■static const float MELEE2_MOVEMENT_BLEND_DELAY = 0.35;
■static const float HEAVY_HIT_THRESHOLD = 0.5; //defines what is considered a 'heavy hit' from the ind
■//-------------------------------------------------------
■//■■■■SHOES DAMAGE/FEET BLEEDING
■//-------------------------------------------------------
■static const float BAREFOOT_MOVEMENT_BLEED_MODIFIER = 0.1;
■static const float SHOES_MOVEMENT_DAMAGE_PER_STEP = 0.035;
■static const int CHECK_EVERY_N_STEP = 10;//will process every n-th step(for performance reasons)
■//-------------------------------------------------------
■//■■■■■STAT LEVELS START
■//-------------------------------------------------------
■static const float SL_HEALTH_CRITICAL = 15;
■static const float SL_HEALTH_LOW = 30;
■static const float SL_HEALTH_NORMAL = 50;
■static const float SL_HEALTH_HIGH = 80;■
■
■static const float SL_TOXICITY_CRITICAL = 20;
■static const float SL_TOXICITY_LOW = 40;
■static const float SL_TOXICITY_NORMAL = 60;
■static const float SL_TOXICITY_HIGH = 80;

■static const float SL_BLOOD_CRITICAL = 3000;
■static const float SL_BLOOD_LOW = 3500;
■static const float SL_BLOOD_NORMAL = 4000;
■static const float SL_BLOOD_HIGH = 4500;
■
■static const float SL_ENERGY_CRITICAL = 0;
■static const float SL_ENERGY_LOW = 300;
■static const float SL_ENERGY_NORMAL = 800;
■static const float SL_ENERGY_HIGH = 3500;
■static const float SL_ENERGY_MAX = 5000;
■
■static const float SL_WATER_CRITICAL = 0;
■static const float SL_WATER_LOW = 300;
■static const float SL_WATER_NORMAL = 800;
■static const float SL_WATER_HIGH = 3500;
■static const float SL_WATER_MAX = 5000;
■//-------------------------------------------------------
■//■■■■■■STAT LEVELS END
■//-------------------------------------------------------

■//----------------------------------------------------
■static const float NORMAL_TEMPERATURE_L = 36.0;
■static const float NORMAL_TEMPERATURE_H = 36.5;
■static const float HIGH_TEMPERATURE_L ■= 38.5;
■static const float HIGH_TEMPERATURE_H■= 39.0;
■//----------------------------------------------------
■static const float DIGESTION_SPEED■■■■■■■= 1.7;■//quantity processed in the stomach per second
■static const float CONSUMPTION_MULTIPLIER_BASE■■■■= 1;■//must not be 0 or less
■static const float STOMACH_ENERGY_TRANSFERED_PER_SEC ■■= 3;■//amount of kcal transfered
■static const float STOMACH_WATER_TRANSFERED_PER_SEC ■■= 6;■//amount of ml transfered to w
■static const float STOMACH_SOLID_EMPTIED_PER_SEC ■■■= 7;■//amount of g/ml emptied from sto

■static const float LOW_WATER_THRESHOLD ■■■■■■= SL_WATER_LOW;■■//threshold from which
```

```
// ---------------------------------------------
// ------ PlayerContainer.c - START --------------------
// ---------------------------------------------


class PlayerContainer: CollapsibleContainer
{
■protected ref AttachmentsGroupContainer■■m_PlayerAttachmentsContainer;
■protected ref map<int, SlotsIcon>■■■m_InventorySlots;
■protected ref map<EntityAI, ref Container>■m_ShowedItems = new ref map<EntityAI, ref Container>;
■protected ref map<int, ref Container>■■m_ShowedItemsIDs = new ref map<int, ref Container>;
■protected PlayerBase■■■■■■m_Player;
■
■protected const■int■■■■■■■HEADER_INDEX_OFFSET = 2;
■
■override void UpdateRadialIcon()
■{
■■if ( m_SlotIcon )
■■{
■■■bool show_radial_icon;
■■■show_radial_icon = IsHidden();
■■■Widget rip = m_SlotIcon.GetRadialIconPanel();
■■■rip.Show( !m_Player.GetInventory().IsInventoryLockedForLockType( HIDE_INV_FROM_SCRIPT ) &
■■■SetOpenForSlotIcon(show_radial_icon);
■■}
■}
■
■bool HasEntityContainerVisible( EntityAI entity )
■{
■■ClosableContainer cont = ClosableContainer.Cast( m_ShowedItems.Get( entity ) );
■■return ( cont && cont.IsOpened() );
■}
■
■SlotsIcon GetSlotsIcon( int row, int column )
■{
■■return m_PlayerAttachmentsContainer.GetSlotsIcon(row, column);
■}

■void PlayerContainer( LayoutHolder parent, int sort = -1 )
■{
■■m_InventorySlots = new ref map<int, SlotsIcon>;
■■m_PlayerAttachmentsContainer = new AttachmentsGroupContainer(this);
■■
■■m_PlayerAttachmentsContainer.SetHeader(GetHeader());
■■m_CollapsibleHeader.SetName( "#container_inventory" );
■■SetHeader(null);
■■m_Body.Insert( m_PlayerAttachmentsContainer );
■■m_MainWidget = m_RootWidget.FindAnyWidget( "body" );
■■m_PlayerAttachmentsContainer.GetRootWidget().SetColor(166 << 24 | 80 << 16 | 80 << 8 | 80);
■■WidgetEventHandler.GetInstance().RegisterOnChildAdd( m_MainWidget, this, "OnChildAdd" );
■■WidgetEventHandler.GetInstance().RegisterOnChildRemove( m_MainWidget, this, "OnChildRemove" )

■■//START - InitGhostSlots
■■string config_path_ghosts_slots = "CfgVehicles SurvivorBase InventoryEquipment playerSlots";
■■ref array<string> player_ghosts_slots = new array<string>;
■■GetGame().ConfigGetTextArray( config_path_ghosts_slots, player_ghosts_slots );

■■for ( int i = 0; i < player_ghosts_slots.Count(); i++ )
■■{
■■■string slot_name = player_ghosts_slots.Get ( i );
■■■string path = "CfgSlots" + " " + slot_name;

■■■if ( GetGame().ConfigIsExisting( path ) )
```

```
// ---------------------------------------------
// ------ PlayerLightManager.c - START --------------------
// ---------------------------------------------


class ActionTargetLighSource : ActionTarget
{
■bool ■m_Remove;
}

//WIP
class PlayerLightManager
{
■int ■■■■■■■■■■m_SelectedLightSource;
■ref ActionTargetLighSource ■■■■m_LightItemTarget;
■ref array<ref ActionTarget> ■■■m_ValidLightItems;
■PlayerBase ■■■■■■■■m_Player;
■
■void PlayerLightManager(PlayerBase player)
■{
■■m_LightItemTarget = null;
■■m_ValidLightItems = new array<ref ActionTarget>;
■■m_Player = player;
■}
■
■// can be anything, as long as it has appropriate actions for handling lights, see Mich2001Helmet
■void AddLightSource(Object object)
■{
■■m_LightItemTarget = new ActionTargetLighSource(object, null, -1, vector.Zero, -1);
■■UpdateLightSourceList();
■}
■
■void RemoveLightSource(Object object)
■{
■■m_LightItemTarget = new ActionTargetLighSource(object, null, -1, vector.Zero, -1);
■■m_LightItemTarget.m_Remove = true;
■■UpdateLightSourceList();
■}
■
■void UpdateLightSourceList()
■{
■■if ( m_LightItemTarget )
■■{
■■■if ( !m_LightItemTarget.m_Remove )
■■■{
■■■■m_ValidLightItems.Insert(m_LightItemTarget);
■■■}
■■■else
■■■{
■■■■m_ValidLightItems.RemoveItem(m_LightItemTarget);
■■■}
■■}
■}
■
■ref array<ref ActionTarget> GetLightSourceList()
■{
■■return m_ValidLightItems;
■}
■
■void SetSelectedLightSourceIdx(int value)
■{
■■m_SelectedLightSource = value;
■}
```

```
// ---------------------------------------------
// ------ PlayerListEntryScriptedWidget.c - START --------------------
// ---------------------------------------------


class PlayerListEntryScriptedWidget extends ScriptedWidgetEventHandler
{
■protected string■■■■■m_Name;
■protected string■■■■■m_UID;
■protected bool■■■■■■m_Mute;
■protected bool■■■■■■m_GlobalMute;
■
■protected Widget■■■■■m_Root;
■protected TextWidget■■■■m_PlayerName;
■protected ImageWidget■■■■m_PlayerAvatar;
■protected ImageWidget■■■■m_MicrophoneIcon;
■protected ImageWidget■■■■m_MuteIcon;
■protected ButtonWidget■■■■m_PlayerButton;
■
■protected PlayerListScriptedWidget■m_Tab;
■protected bool■■■■■■m_Selected;
■
■void PlayerListEntryScriptedWidget( Widget parent, string name, string uid, bool show_permissions, Play
■{
■■m_Name■■■■= name;
■■m_UID■■■■= uid;
■■m_Tab■■■■= tab;
■■
■■m_Root■■■■= GetGame().GetWorkspace().CreateWidgets( "gui/layouts/xbox/ingamemenu_xbox/pla
■■m_PlayerName■■= TextWidget.Cast( m_Root.FindAnyWidget( "Name" ) );
■■m_PlayerAvatar■■= ImageWidget.Cast( m_Root.FindAnyWidget( "Avatar" ) );
■■m_MicrophoneIcon■= ImageWidget.Cast( m_Root.FindAnyWidget( "Microphone" ) );
■■m_MuteIcon■■■= ImageWidget.Cast( m_Root.FindAnyWidget( "Muted" ) );
■■m_PlayerButton■■= ButtonWidget.Cast( m_Root.FindAnyWidget( "Button" ) );
■■
■■m_MicrophoneIcon.Show( show_permissions && !IsLocalPlayer() );
■■
■■m_PlayerName.SetText( name );
■■m_Root.SetHandler( this );
■}
■
■void ~PlayerListEntryScriptedWidget()
■{
■■delete m_Root;
■}
■
■Widget GetButtonWidget()
■{
■■return m_PlayerButton;
■}
■
■void LoadPermissions( BiosPrivacyPermissionResultArray results )
■{
■■foreach ( BiosPrivacyPermissionResult result : results )
■■{
■■■if ( result.m_Permission == EBiosPrivacyPermission.COMMUNICATE_VOICE )
■■■{
■■■■m_GlobalMute = !result.m_IsAllowed;
■■■■if ( result.m_IsAllowed && !m_Mute )
■■■■{
■■■■■m_MuteIcon.Show( false );
■■■■}
■■■■else if ( !result.m_IsAllowed || m_Mute )
```

```
// ---------------------------------------------
// ------ PlayerListScriptedWidget.c - START --------------------
// ---------------------------------------------


class PlayerListScriptedWidget extends ScriptedWidgetEventHandler
{
■protected Widget■■■■■■■■■■■■m_Root;
■protected ScrollWidget■■■■■■■■■■■m_ScrollContainer;
■protected Widget■■■■■■■■■■■■m_Content;
■protected ref map<string, ref PlayerListEntryScriptedWidget>■m_Entries;
■
■protected int■■■■■■■■■■■■■m_TotalEntries;
■protected PlayerListEntryScriptedWidget■■■■■■■m_SelectedEntry;
■
■void PlayerListScriptedWidget( Widget parent, string header_text = "" )
■{
■■m_Root■■■■= GetGame().GetWorkspace().CreateWidgets( "gui/layouts/xbox/ingamemenu_xbox/pla
■■m_ScrollContainer■= ScrollWidget.Cast( m_Root.FindAnyWidget( "ScrollFrame" ) );
■■m_Content■■■= m_Root.FindAnyWidget( "Content" );
■■
■■m_Entries■■■= new map<string, ref PlayerListEntryScriptedWidget>;

■■m_ScrollContainer.VScrollToPos01( 0 );
■}
■
■void ~PlayerListScriptedWidget()
■{
■■delete m_Root;
■}
■
■void FocusFirst()
■{
■■if ( m_Content && m_Content.GetChildren() )
■■■SetFocus( m_Content.GetChildren().FindAnyWidget( "Button" ) );
■■m_ScrollContainer.VScrollToPos01( 0 );
■}
■
■void Reload( SyncPlayerList player_list )
■{
■■if ( player_list && player_list.m_PlayerList && m_Entries )
■■{
■■■foreach ( string UID, PlayerListEntryScriptedWidget widget : m_Entries )
■■■{
■■■■SyncPlayer player_found;
■■■■foreach ( SyncPlayer player : player_list.m_PlayerList )
■■■■{
■■■■■if ( player && player.m_UID == UID )
■■■■■{
■■■■■■player_found = player;
■■■■■■break;
■■■■■}
■■■■}
■■■■if ( !player_found )
■■■■{
■■■■■RemovePlayer( UID );
■■■■}
■■■}
■■■
■■■for ( int i = 0; i < player_list.m_PlayerList.Count(); i++ )
■■■{
■■■■SyncPlayer player2 = player_list.m_PlayerList.Get( i );
■■■■PlayerListEntryScriptedWidget player_widget;
```

```
// ---------------------------------------------
// ------ PlayerPreview.c - START --------------------
// ---------------------------------------------


class PlayerPreview: LayoutHolder
{
■protected ref PlayerPreviewWidget m_CharacterPanelWidget;

■protected int m_CharacterRotationX;
■protected int m_CharacterRotationY;
■protected int m_CharacterScaleDelta;
■protected vector m_CharacterOrientation;
■protected bool m_IsHolding;
■
■void PlayerPreview( LayoutHolder parent )
■{
■■m_CharacterPanelWidget = PlayerPreviewWidget.Cast( m_Parent.GetMainWidget().FindAnyWidget( "

■■WidgetEventHandler.GetInstance().RegisterOnMouseButtonDown( m_Parent.GetMainWidget().FindAn
■■WidgetEventHandler.GetInstance().RegisterOnMouseWheel( m_Parent.GetMainWidget().FindAnyWid
■■
■■m_CharacterScaleDelta = 1;
■■m_CharacterPanelWidget.SetPlayer( GetGame().GetPlayer() );
■■m_CharacterPanelWidget.SetModelPosition( "0 0 0.605" );
■■m_CharacterPanelWidget.SetSize( 1.34, 1.34 );  // default scale
■■UpdateScale();
■}
■
■void RefreshPlayerPreview()
■{
■■m_CharacterPanelWidget.Refresh();
■}
■
■void UpdateRotation( int mouse_x, int mouse_y, bool is_dragging )
■{
■■if ( m_CharacterPanelWidget )
■■{
■■■vector orientation = m_CharacterOrientation;
■■■orientation[1] = orientation[1] - ( m_CharacterRotationX - mouse_x );
■■■
■■■m_CharacterPanelWidget.SetModelOrientation( orientation );
■
■■■if ( !is_dragging )
■■■{■
■■■■m_CharacterOrientation = orientation;
■■■}
■■}
■}

■void UpdateScale()
■{
■■if ( m_CharacterPanelWidget )
■■{
■■■float w, h;
■■■m_CharacterPanelWidget.GetSize( w, h );
■■■w = w + ( m_CharacterScaleDelta / 25 );
■■■h = h + ( m_CharacterScaleDelta / 25 );
■■■if ( w > 0.62 && w < 3 )
■■■{
■■■■m_CharacterPanelWidget.SetSize( w, h );
■■■}
■■■else if ( w < 0.62 )
```

```
// ---------------------------------------------
// ------ PlayerSoundEventBase.c - START --------------------
// ---------------------------------------------


enum EPlayerSoundEventType
{
■GENERAL ■= 0x00000001,
■MELEE ■■= 0x00000002,
■STAMINA ■= 0x00000004,
■DAMAGE ■■= 0x00000008,
■DUMMY ■■= 0x00000010,
■INJURY ■■= 0x00000020,
■DROWNING ■= 0x00000040,
■//HEAT_COMFORT■= 0x00000080,
}

enum EPlayerSoundEventParam
{
■SKIP_CONTROLLED_PLAYER ■= 0x00000001,
■HIGHEST_PRIORITY■■= 0x00000002,
■/*
■STAMINA ■= 0x00000004,
■DAMAGE ■■= 0x00000008,
■DUMMY ■■= 0x00000010,
■INJURY ■■= 0x00000020,
■HEAT_COMFORT■= 0x00000040,
■*/
■
■// ONLY COUNT BELLOW
■ENUM_COUNT,

}

class PlayerSoundEventBase extends SoundEventBase
{
■PlayerBase ■m_Player;
■float■■m_DummySoundLength;
■float ■■m_DummyStartTime;
■bool■■m_IsDummyType;
■bool■■m_ProcessPlaybackEvent;
■float ■■m_PlayTime;

■ref HumanMovementState m_Hms = new HumanMovementState();
■EPlayerSoundEventType m_HasPriorityOverTypes;
■
■bool IsDummy()
■{
■■return m_IsDummyType;
■}
■■
■EPlayerSoundEventType GetPriorityOverTypes()
■{
■■return m_HasPriorityOverTypes;
■}
■
■// !can this event play during hold breath
■bool HasHoldBreathException()
■{
■■return false;
■}
■
■void PlayerSoundEventBase()
```

```
// ---------------------------------------------
// ------ PlayerSoundEventHandler.c - START --------------------
// ---------------------------------------------


enum EPlayerSoundEventID
{
■HOLD_BREATH = 1,
■EXHAUSTED_BREATH,
■RELEASE_BREATH,
■STAMINA_DOWN_LIGHT,
■STAMINA_DOWN_HEAVY,
■STAMINA_UP_LIGHT,
■STAMINA_UP_HEAVY,
■STAMINA_UP_END,
■STAMINA_NORMAL_DUMMY,
■TAKING_DMG_LIGHT,
■TAKING_DMG_HEAVY,
■SYMPTOM_COUGH,
■SYMPTOM_LAUGHTER,
■SYMPTOM_SNEEZE,
■JUMP,
■MELEE_ATTACK_LIGHT,
■MELEE_ATTACK_HEAVY
■INJURED_LIGHT,
■INJURED_MEDIUM,
■INJURED_HIGH,
■FREEZING,
■HOT,
■SYMPTOM_FATIGUE,
■STAMINA_LOW_FILTER_UPPER,
■STAMINA_LOW_FILTER_MID,
■STAMINA_LOW_FILTER_LOWER,
■DROWNING_BREATH,
■DROWNING_PAIN,
■PICKUP_HEAVY,
■//--------------
■// Count bellow, put enums above
■//--------------
■ENUM_COUNT,
}

class PlayerSoundEventHandler extends SoundEventHandler
{
■PlayerBase m_Player;
■const int SOUND_EVENTS_MAX = EPlayerSoundEventID.ENUM_COUNT;
■static ref PlayerSoundEventBase m_AvailableStates[SOUND_EVENTS_MAX];
■static ref map<int,int> m_ConfigIDToScriptIDmapping = new ref map<int,int> ;
■ref PlayerSoundEventBase m_CurrentState;
■ref Timer m_UpdateTimer;
■
■
■void PlayerSoundEventHandler(PlayerBase player)
■{
■■m_Player = player;
■■
■■if(!m_UpdateTimer && !m_Player.IsControlledPlayer())
■■{
■■■m_UpdateTimer = new Timer();
■■■m_UpdateTimer.Run(1, this, "OnTick", null, true);//ticking for remotes, the controlled player is ticking c
■■}
■■
■■RegisterState(new HoldBreathSoundEvent());
```

```
// ----------------------------------------------
// ------ PlayerSoundManager.c - START --------------------
// ----------------------------------------------



const float SOUNDS_HEARING_DISTANCE = 50;

enum eSoundHandlers
{
■STAMINA,
■HUNGER,
■INJURY,
■//FREEZING,
■
■//------
■// add stuff above !
■//------
■COUNT
}



class PlayerSoundManagerBase
{
■PlayerBase m_Player;
■
■const int MAX_HANDLERS_COUNT = eSoundHandlers.COUNT;
■ref SoundHandlerBase m_Handlers[MAX_HANDLERS_COUNT];
■
■void PlayerSoundManagerBase(PlayerBase player)
■{
■■m_Player = player;
■■Init();
■}
■
■void RegisterHandler(SoundHandlerBase handler)
■{
■■int index = handler.GetID();
■■m_Handlers[index] = handler;
■}
■
■SoundHandlerBase GetHandler(eSoundHandlers id)
■{
■■return m_Handlers[id];
■}
■
■void Init()
■{
■
■}
■
■void Update()
■{
■
■}
■

}

class PlayerSoundManagerServer extends PlayerSoundManagerBase
{
■//------------------------
```

```
// ----------------------------------------------
// ------ PlayerStatBase.c - START --------------------
// ----------------------------------------------


class PlayerStatBase
{
■int■m_Type;
■void ■OnStoreSave( ParamsWriteContext ctx );
■bool ■OnStoreLoad( ParamsReadContext ctx);
■void ■OnRPC(ParamsReadContext ctx);
■float ■Get();
■string ■GetLabel();
■void ■SetByFloat(float value);
■void ■SetByFloatEx(float value, string system = "" );
■bool■IsSynced();
■array<PlayerStatRecord> GetRecords();
■void ■Init(int id/*, PlayerStats manager*/);
■void ■SerializeValue(array<ref StatDebugObject> objects, int flags);
■float ■GetNormalized();
■float ■GetMax();
■float ■GetMin();
■int■■GetType()
■{
■■return m_Type;
■}
};

class PlayerStat<Class T> extends PlayerStatBase
{
■protected T ■■m_MinValue;
■protected T ■■m_MaxValue;
■protected T ■■m_Value;
■protected string ■m_ValueLabel;
■protected int■■m_Flags;
■
■ref array<PlayerStatRecord> m_Records;
■PlayerStats ■■m_Manager;
■
■
■void PlayerStat(T min, T max,T init, string label, int flags)
■{
■■m_MinValue ■■= min;
■■m_MaxValue ■■= max;
■■m_Value■■■= init;
■■m_ValueLabel ■= label;
■■m_Flags■■■= flags;
■■
■■m_Records = new array<PlayerStatRecord>;
■}
■■
■override void Init(int id/*, PlayerStats manager*/)
■{
■■m_Type = id;
■■//m_Manager = manager;■
■}
■
■override void SerializeValue(array<ref StatDebugObject> objects, int flags)
■{
■■objects.Insert( new StatDebugObject(GetLabel(), Get(), eRemoteStatType.PLAYER_STATS) );
■}
■
■PlayerStats GetManager()
```

```
// ---------------------------------------------
// ------ PlayerStatRecord.c - START --------------------
// ---------------------------------------------


class PlayerStatRecord
{
■float ■■m_Value;
■float ■■m_Time;
■string■■m_System;
■
■void PlayerStatRecord(float value, float time, string system)
■{
■■m_Value = value;
■■m_Time = time;
■■m_System = system;
■}
■
■string GetStringOutput()
■{
■■return m_Time.ToString()+", " +m_Value.ToString() + ", " +m_System;
■■
■}

};



// ---------------------------------------------
// ------ PlayerStatRecord.c - END ----------------------
// ---------------------------------------------


// ---------------------------------------------
// ------ PlayerStats.c - START --------------------
// ---------------------------------------------


enum EPSstatsFlags
{
■EMPTY,
};

class PlayerStats
{
■ref array<ref PlayerStatBase> m_PlayerStats;
■ref array<ref StatDebugObject> m_PlayerStatsDebug;
■
■//ref PlayerStatsPCO_current m_PCO = new PlayerStatsPCO_current();
■ref PCOHandlerStats m_PCOhandler = new PCOHandlerStats();
■
■ref Timer m_SyncTimer;
■Man m_Player;
■bool m_AllowLogs;
■string m_System = "Stats"; //debuging tag
■
■//int m_SystemVersion = 101;
■
■void PlayerStats(Man player)
■{
■■Init(player);
■}
```

```
// ---------------------------------------------
// ------ PlayerStatsPCO.c - START --------------------
// ---------------------------------------------




class PCOHandlerStats
{
■ref map<int, ref PlayerStatsPCO_Base> m_PCOs = new map<int, ref PlayerStatsPCO_Base>;
■int m_HighestVersion;
■
■void PCOHandlerStats()
■{
■■RegisterPCO(new PlayerStatsPCO_v100);
■■RegisterPCO(new PlayerStatsPCO_v101);
■■RegisterPCO(new PlayerStatsPCO_current);
■}
■
■void RegisterPCO(PlayerStatsPCO_Base pco)
■{
■■int version = pco.GetVersion();
■■
■■if( version > m_HighestVersion )
■■{
■■■m_HighestVersion = version;
■■}
■■
■■m_PCOs.Insert(version, pco);
■}
■
■PlayerStatsPCO_Base GetPCO(int version = -1)
■{
■■//PlayerStatsPCO_Base pco;

■■if(version == -1)//no version set - fetch the highest version
■■{
■■■return m_PCOs.Get(m_HighestVersion);
■■}
■■else if( !m_PCOs.Contains(version))//version set - version not present, fetch the closest lower version
■■{
■■■for(int i = version; i > 100; i--)
■■■{
■■■■if(m_PCOs.Contains(i))
■■■■{
■■■■■//Print("fetching PCO version:"+ i);
■■■■■return m_PCOs.Get(i);■
■■■■}
■■■}
■■■return null;
■■}
■■else//version set - version present, fetch it
■■{
■■■//Print("fetching PCO version:"+ version);
■■■return m_PCOs.Get(version);
■■}
■}
}


class PlayerStatsPCO_Base
{
■void PlayerStatsPCO_Base()
```

```
// ----------------------------------------------
// ------ PlayerStomach.c - START --------------------
// ----------------------------------------------


class StomachItem
{
■ref NutritionalProfile m_Profile;
■float m_Amount;
■int m_FoodStage;
■//bool m_IsLiquid;
■string m_ClassName;
■int m_Agents;
■
■void StomachItem(string class_name, float amount, NutritionalProfile profile,int foodstage, int agents)
■{
■■m_Amount = amount;
■■m_Profile = profile;
■■//m_IsLiquid = is_liquid;
■■m_FoodStage = foodstage;
■■m_ClassName = class_name;
■■m_Agents = agents;
■}
■
■string GetClassName()
■{
■■return m_ClassName;
■}
■
■/*
■void SetLiquid(bool is_liquid)
■{
■■m_IsLiquid = is_liquid;
■}
■
■bool IsLiquid()
■{
■■return m_IsLiquid;
■}*/
■

■
■int GetFoodStage()
■{
■■return m_FoodStage;
■}
■
■void SetFoodStage(int food_stage)
■{
■■m_FoodStage = food_stage;
■}
■
■// returns grams or ml's of this item/liquid
■float GetAmount()
■{
■■return m_Amount;
■}
■
■// adds grams or ml's of this item/liquid
■void AddAmount(float amount)
■{
■■m_Amount += amount;
■}
```

```
// ---------------------------------------------
// ------ PleurotusMushroom.c - START --------------------
// ---------------------------------------------


class PleurotusMushroom : MushroomBase
{
■override void SetActions()
■{
■■super.SetActions();
■■
■■AddAction(ActionCreateIndoorFireplace);
■■AddAction(ActionCreateIndoorOven);
■}
}



// ---------------------------------------------
// ------ PleurotusMushroom.c - END ----------------------
// ---------------------------------------------


// ---------------------------------------------
// ------ Pliers.c - START --------------------
// ---------------------------------------------


class Pliers extends ToolBase
{
■void Pliers()
■{
■■m_MineDisarmRate = 100;
■}

■override void SetActions()
■{
■■super.SetActions();
■■
■■AddAction(ActionDismantlePart);
■■AddAction(ActionBuildPart);
■■AddAction(ActionUnrestrainTarget);
■■AddAction(ActionMountBarbedWire);
■■AddAction(ActionUnmountBarbedWire);
■■AddAction(ActionLockAttachment);
■■AddAction(ActionDisarmExplosive);
■}
}


// ---------------------------------------------
// ------ Pliers.c - END ----------------------
// ---------------------------------------------


// ---------------------------------------------
// ------ PluginAdditionalInfo.c - START --------------------
// ---------------------------------------------


class PluginAdditionalInfo extends PluginLocalProfile
{
■override string GetFileName()
```

```
// ---------------------------------------------
// ------ PluginAdminLog.c - START --------------------
// ---------------------------------------------


class PluginAdminLog extends PluginBase■■■// Class for admin log messages handled by script
{
■string ■■■■■■■m_PlayerName;
■string■■■■■■■m_Pid;
■vector ■■■■■■■m_Position;
■string■■■■■■■m_PlayerPrefix;
■string ■■■■■■■m_PlayerPrefix2;
■string■■■■■■■m_Message;
■string■■■■■■■m_DisplayName;
■string ■■■■■■■m_HitMessage;
■float■■■■■■■m_Distance;
■PlayerBase■■■■■■m_Player;
■PlayerBase■■■■■■m_Source;
■string■■■■■■■m_ItemInHands;
■string ■■■■■■■m_PosArray[3];
■int ■■■■■■■m_DotIndex;
■PlayerStat<float>■■■■m_StatWater;
■PlayerStat<float>■■■■m_StatEnergy;
■BleedingSourcesManagerServer■m_BleedMgr;
■// filters
■protected int ■■■■■m_HitFilter;
■protected int ■■■■■m_PlacementFilter;
■protected int ■■■■■m_ActionsFilter;
■protected int ■■■■■m_PlayerListFilter;
■
■ref Timer■■■■■■m_Timer;
■autoptr array<Man>■■■■m_PlayerArray; ■
■const int ■■■■■■TIMER_PLAYERLIST = GetPlayerListTimer();
■■
■static int GetPlayerListTimer()
■{
■■return 300; // seconds
■}
■
■/*
■■EXE side ADM log messages (not removable):
■■Connect / Disconnect
■■Chat
■■Player->Admin report (ingame chat: #toadmin <text>)■
■*/
■
■void PluginAdminLog()
■{
■■m_HitFilter = GetGame().ServerConfigGetInt("adminLogPlayerHitsOnly");  //  1 - log player hits only / 0
■■m_PlacementFilter = GetGame().ServerConfigGetInt("adminLogPlacement"); //  1 - log placement ( tra
■■m_ActionsFilter = GetGame().ServerConfigGetInt("adminLogBuildActions"); //  1 - log basebuilding acti
■■m_PlayerListFilter = GetGame().ServerConfigGetInt("adminLogPlayerList"); // 1 - log periodic player lis
■■
■■m_PlayerArray = new array<Man>;
■■
■■if ( m_PlayerListFilter == 1 )
■■{
■■■m_Timer = new Timer();
■■■m_Timer.Run( TIMER_PLAYERLIST , this, "PlayerList", NULL, true );
■■}
■}
■
■void ~PluginAdminLog()
```

```
// ---------------------------------------------
// ------ PluginBase.c - START --------------------
// ---------------------------------------------


class PluginBase
{
■void PluginBase();■
■void ~PluginBase();
■
■void OnInit();■
■void OnUpdate(float delta_time);■
■void OnDestroy();
■
■void Log(string msg, string label)
■{
■■Debug.Log(msg, GetModuleName(), "n/a", label, "n/a");
■}
■
■string GetModuleName()
■{
■■return ClassName();
■}

■typename GetModuleType()
■{
■■return GetModuleName().ToType();
■}
}
```

```
// ---------------------------------------------
// ------ PluginBase.c - END ----------------------
// ---------------------------------------------


// ---------------------------------------------
// ------ PluginCameraTools.c - START --------------------
// ---------------------------------------------


class PluginCameraTools extends PluginBase
{
■protected static PluginCameraTools m_Instance;
 ■protected static bool m_IsOpen;
■
■override void OnInit()
■{
■■m_Instance = this;
■}
■
```

```
// ---------------------------------------------
// ------ PluginCharPlacement.c - START --------------------
// ---------------------------------------------


class PluginCharPlacement extends PluginBase
{
■PluginDeveloper ■m_ModuleDeveloper;
■PlayerBase■■ ■m_Character;
■ref ■■■■Timer m_Timer;
■bool ■■■■m_Initialized;
■vector ■■■■m_cam_pos;
■vector ■■■■m_character_pos;
■vector ■■■■m_camera_dir;
■vector ■■■■m_cameraTrans[4];
■vector ■■■■m_demoPos;
■vector ■■■■m_demoRot;
■vector ■■■■m_camera_orientation;
■
■const float ■■FOV = 0.5236;
■
■void CheckInit()
■{
■■if ( m_Initialized )
■■{
■■■Print("logging...");
■■■Log(FOV.ToString(),"FOV");
■■■Log(m_camera_dir.ToString(),"camera dir");
■■■Log(m_cam_pos.ToString(),"camera pos");
■■■Log(m_demoPos.ToString(),"character pos");
■■■
■■■return;
■■}
■■
■■if( FreeDebugCamera.GetInstance().IsActive() )
■■■Init();
■}
■
■void Init()
■{
■■if(m_Character)
■■■m_Character.Delete();
■■
■■m_Initialized = true;
■■m_Character = PlayerBase.Cast( GetGame().CreateObject("SurvivorF_Judy", FreeDebugCamera.GetI
■■
■■FreeDebugCamera.GetInstance().SetFOV(FOV); //default scene FOV
■■m_demoRot = "0 0 0";
■}
■
■override void OnUpdate(float delta_time)
■{
■■if(!m_Initialized) return;
■■if(!FreeDebugCamera.GetInstance().IsActive()) return;
■■if(!m_Character) return;
■■m_camera_orientation = FreeDebugCamera.GetInstance().GetOrientation();
■■m_camera_orientation[1] = 1;
■■FreeDebugCamera.GetInstance().SetOrientation(m_camera_orientation); //remove to unlock vertical ax
■■m_cam_pos = FreeDebugCamera.GetInstance().GetPosition();
■■m_character_pos = m_Character.GetPosition();
■■m_camera_dir = FreeDebugCamera.GetInstance().GetDirection();
■■
■■m_camera_dir.Normalize();
```

```
// ---------------------------------------------
// ------ PluginConfigDebugProfile.c - START --------------------
// ---------------------------------------------


typedef Param3<string, bool, vector> LocationParams;//  param1 - name, param2 - isCustom?, param3 - p
class PluginConfigDebugProfile extends PluginConfigHandler
{■
■protected const string SCENE_DRAW_SELECTION■■■■= "scene_editor_draw_selection";
■protected const string SCENE_LOAD_PLAYER_POS■■■= "scene_editor_load_player_pos";
■protected const string SCENE_ROTATION_ANGLE■■■■= "scene_editor_rotation_angle";
■protected const string ITEM_TAB_SELECTED■■■■= "console_TabSelected";
■protected const string PRESET_LIST■■■■■■= "console_presets";
■protected const string PRESET_DEFAULT■■■■■= "console_preset_default";
■protected const string ITEM_SEARCH■■■■■■= "console_ItemSearch";
■protected const string SPAWN_DISTANCE■■■■■= "console_SpawnDistance";
■protected const string CHAR_STATS_VIS■■■■■= "console_character_stats_visible";
■protected const string CHAR_LEVELS_VIS■■■■■= "console_character_levels_visible";
■protected const string CHAR_MODIFIERS_VIS■■■■= "console_character_modifiers_visible";
■protected const string CHAR_AGENTS_VIS■■■■■= "console_character_agents_visible";
■protected const string CHAR_DEBUG_VIS■■■■■= "console_character_debug_visible";
■protected const string CHAR_STOMACH_VIS■■■■■= "console_character_stomach_visible";
■protected const string FREE_CAMERA_CROSSHAIR■■■= "console_free_camera_crosshair_visible";
■protected const string VERSION_VIS■■■■■■= "console_version_visible";
■protected const string MERGE_TYPE■■■■■■= "category_merge_type";
■protected const string TEMP_VIS■■■■■■■= "console_temperature_visible";
■protected const string SUB_PARAM_ITEM■■■■■= "item";
■protected const string SUB_PARAM_ITEM_NAME■■■■= "name";
■protected const string SUB_PARAM_ITEM_HEALTH■■■= "health";
■protected const string SUB_PARAM_ITEM_QUANTITY■■■= "quantity";
■protected const string LOGS_ENABLED■■■■■■= "logs_enabled";
■protected const string CONFIG_CLASSES_FLAG■■■■= "toggled_config_classes_flag";
■protected const string ITEM_CATEGORY_FLAG■■■■= "toggled_item_categories_flag";
■protected const string ITEM_PREVIEW■■■■■■= "show_item_preview";
■protected const string BATCH_RECT■■■■■■= "batch_spawn_rect";
■protected const string BATCH_QUANT■■■■■■= "batch_spawn_quantity";
■protected const string SOUNDFILTER■■■■■■= "soundset_editbox";
■protected const string ITEMDEBUG■■■■■■= "item_debug";
■protected const string SPAWN_LOC_INDEX■■■■■= "spawn_loc_index";
■protected const string FILTER_REVERSED■■■■■= "filter_order_reversed";

■protected ref map<string, ref CfgParam>■■■■m_DefaultValues;
■protected ref TStringArray ■■■■■■■m_PresetList;■
■protected const string POSITION_NAME_ROOT■■■= "console_positions_";
■
■//========================================
■// GetInstance
■//========================================
■static PluginConfigDebugProfile GetInstance()
■{
■■return PluginConfigDebugProfile.Cast( GetPlugin(PluginConfigDebugProfile) );
■}■
■
■//========================================
■// GetCfgParamArray
■//========================================
■protected CfgParamArray GetNewCfgParamArray( array<ref CfgParam> params )
■{
■■CfgParamArray param = new CfgParamArray( "" );
■■param.SetParams( params );
■■return param;
■}
■
```

```
// --------------------------------------------
// ------ PluginConfigDebugProfileFixed.c - START --------------------
// --------------------------------------------


class PluginConfigDebugProfileFixed extends PluginConfigDebugProfile
{
■const string POSITIONS_LIST■■■= "console_positions";
■const string POSITIONS_LIST_ENOCH■= "console_positions_enoch";
■const string SUB_PARAM_POS_NAME■■= "name";
■const string SUB_PARAM_POS_VEC■■= "pos";
■
■//========================================
■// ModuleLocalProfileUIFixed
■//========================================
■void ModuleLocalProfileUIFixed()
■{
■■m_ReadOnly = true;
■}
■
■//========================================
■// GetFileName
■//========================================
■override string GetFileName()
■{
■■return CFG_FILE_FIXED_PROFILE;
■}
}


// --------------------------------------------
// ------ PluginConfigDebugProfileFixed.c - END ---------------------
// --------------------------------------------


// --------------------------------------------
// ------ PluginConfigEmotesProfile.c - START --------------------
// --------------------------------------------


class PluginConfigEmotesProfile extends PluginConfigHandler
{■
■/*protected const string EMOTE_1■■■■= "emote_slot_1";
■protected const string EMOTE_2■■■■= "emote_slot_2";
■protected const string EMOTE_3■■■■= "emote_slot_3";
■protected const string EMOTE_4■■■■= "emote_slot_4";
■protected const string EMOTE_5■■■■= "emote_slot_5";
■protected const string EMOTE_6■■■■= "emote_slot_6";
■protected const string EMOTE_7■■■■= "emote_slot_7";
■protected const string EMOTE_8■■■■= "emote_slot_8";
■protected const string EMOTE_9■■■■= "emote_slot_9";
■protected const string EMOTE_10■■■■= "emote_slot_10";
■protected const string EMOTE_11■■■■= "emote_slot_11";
■protected const string EMOTE_12■■■■= "emote_slot_12";*/

■protected ref map<string, ref CfgParam>■■m_DefaultValues;
■protected ref TStringArray ■■■■■m_PresetList;■
■
■//========================================
■// GetInstance
■//========================================
■static PluginConfigEmotesProfile GetInstance()
■{
```

```
// ---------------------------------------------
// ------ PluginConfigHandler.c - START --------------------
// ---------------------------------------------


class PluginConfigHandler extends PluginFileHandler
{■
■ref array<ref CfgParam> m_CfgParams;
■
■void PluginConfigHandler()
■{
■■m_CfgParams = new array<ref CfgParam>;
■■
■■LoadConfigFile();
■}

■override void OnInit()
■{
■■super.OnInit();
■}
■
■override string GetFileName()
■{
■■return CFG_FILE_USER_PROFILE;
■}
■
■void LoadConfigFile()
■{
■■LoadFile();
■■
■■m_CfgParams.Clear();
■■
■■for ( int j = 0; j < m_FileContent.Count(); j++ )
■■{
■■■m_CfgParams.Insert( ParseText(m_FileContent.Get(j)) );
■■}■
■}
■
■void SaveConfigToFile()
■{
■■ClearFileNoSave();
■■
■■for ( int i = 0; i < m_CfgParams.Count(); ++i )
■■{
■■■string s = ParseParam(m_CfgParams.Get(i));

■■■m_FileContent.Insert(s);
■■}
■■
■■SaveFile();
■}
■
■CfgParam GetParamByName(string name, int cfg_type)
■{
■■CfgParam param;
■■
■■for ( int i = 0; i < m_CfgParams.Count(); ++i )
■■{
■■■param = m_CfgParams.Get(i);
■■■
■■■if ( param.GetName() == name )
■■■{
■■■■return param;
```

```
// ---------------------------------------------
// ------ PluginConfigScene.c - START --------------------
// ---------------------------------------------


class PluginConfigScene extends PluginConfigHandler
{
■protected string FILE_ROOT■■■■■= "$saves:";
■protected const string FILE_ROOT_SCENES■■= "scenes";
■
■protected const string PARAM_MISSION■■= "Mission";
■protected const string PARAM_TIME■■■= "InitTime";
■protected const string PARAM_YEAR■■■= "InitYear";
■protected const string PARAM_MONTH■■■= "InitMonth";
■protected const string PARAM_DAY■■■= "InitDay";
■protected const string PARAM_HOUR■■■= "InitHour";
■protected const string PARAM_MINUTE■■■= "InitMinute";
■protected const string PARAM_OVERCAST■■= "WeatherInitOvercast";
■protected const string PARAM_RAIN■■■= "WeatherInitRain";
■protected const string PARAM_FOG■■■= "WeatherInitFog";
■protected const string PARAM_WIND_F■■■= "WeatherInitWindForce";
■protected const string PARAM_PLAYER■■■= "Player";
■protected const string PARAM_OBJ_COUNT■■= "SceneObjectsCount";
■protected const string PARAM_OBJ_NAME■■= "SceneObject";
■protected const string PARAM_OBJ_ATT■■= "Att";
■protected const string PARAM_OBJ_LNK■■= "Lnk";
■protected const string PARAM_OBJ_TYPE■■= "type";
■protected const string PARAM_OBJ_POS■■= "position";
■protected const string PARAM_OBJ_ROT■■= "rotation";
■protected const string PARAM_OBJ_DMG■■= "damage";
■protected const string PARAM_OBJ_HLT■■= "health";
■protected const string PARAM_OBJ_ISCR■■= "init_script";
■
■TStringArray m_CfgTemplate;
■
■protected string m_FileSceneName;

■//========================================
■// OnInit
■//========================================
■override void OnInit()
■{
■■//super.OnInit();
■■■
■■FILE_ROOT = g_Game.GetMissionFolderPath();
■}
■
■//========================================
■// GetPathScenes
■//========================================
■string GetPathScenes()
■{■
■■return FILE_ROOT+"\\"+FILE_ROOT_SCENES;
■}

■//========================================
■// GetFileName
■//========================================
■override string GetFileName()
■{
■■return GetPathScenes()+"\\"+m_FileSceneName;
■}
```

```
// ---------------------------------------------
// ------ PluginConfigViewer.c - START --------------------
// ---------------------------------------------


class PluginConfigViewer extends PluginBase
{
■void PluginConfigViewer()
■{
■}
■
■string MakeTabs( int count, bool inheritance = false )
■{
■■if ( count == 0 )
■■{
■■■return "";
■■}

■■string tabs = "|--";
■■if ( inheritance )
■■{
■■■tabs = "|<<";
■■}
■■
■■for ( int i = 0; i < count - 1; i++ )
■■{
■■■tabs = "|  " + tabs;
■■}

■■return tabs;
■}

■string GetOnlyChildPath( string config_class_path, string class_path )
■{
■■int config_class_path_len = config_class_path.Length();
■■int class_path_len = class_path.Length();
■■
■■if ( class_path_len > config_class_path_len )
■■{
■■■int start = config_class_path_len;
■■■int count = class_path_len - start;
■■■
■■■return class_path.Substring(start, count).Trim();
■■}
■■else
■■{
■■■return "";
■■}
■}

■string GetBaseClassPath( string config_class_path, string class_path, string config_class )
■{
■■if ( class_path == "" )
■■{
■■■return "";
■■}
■■
■■int start = config_class_path.Length();
■■int count = class_path.Length() - start;
■■string class_path_without_config_class = GetOnlyChildPath( config_class_path, class_path );
■■
■■ref TStringArray full_path = new TStringArray;
■■GetGame().ConfigGetFullPath( config_class_path, full_path );
```

```
// ---------------------------------------------
// ------ PluginDayZCreatureAIDebug.c - START --------------------
// ---------------------------------------------


typedef Param4<float, string, int, string> ■DayZCreatureAnimScriptDebugAnimEventData;
typedef Param1<string> ■■■■■■DayZCreatureAnimScriptDebugAnimPredictionData;
typedef Param1<string> ■■■■■■DayZCreatureAnimScriptDebugAnimTagData;

class DayZCreatureAnimScriptDebug
{■■
■proto native void ■SetCreature(DayZCreature creature);

■proto native int ■GetVariableCount();
■proto native owned string GetVariableName(int index);
■proto native int ■GetVariableType(int index);
■proto native int ■GetVariableInt(int index);
■proto native float ■GetVariableFloat(int index);
■proto native bool ■GetVariableBool(int index);
■
■proto native int ■SetVariableInt(int index, int value);
■proto native float ■SetVariableFloat(int index, float value);
■proto native bool ■SetVariableBool(int index, bool value);
■■
■proto native int ■GetCommandCount();
■proto native owned string GetCommandName(int index);
■proto native int ■GetCommandID(int index);

■proto native void ■ActivateCommand(int index, int userInt, float userFloat);
■
■const int ■■■■■■■■■■■■■■m_iMaxAnimEventsCount = 50;
■ref array<string> ■■■■■■■■■■■■m_EventsFilter = new array<string>;
■
■ref array<ref DayZCreatureAnimScriptDebugAnimEventData>■■■m_AnimEvents = new array<ref DayZ
■ref array<ref DayZCreatureAnimScriptDebugAnimPredictionData>■m_AnimPredictions = new array<ref
■ref array<ref DayZCreatureAnimScriptDebugAnimTagData>■■■m_AnimTags = new array<ref DayZCre
■
■void OnAnimationEventsStart()
■{
■■m_AnimPredictions.Clear();
■■m_AnimTags.Clear();
■}

■void OnAnimationEvent(string evType, int userInt, string userString)
■{
■■if (m_EventsFilter.Find(evType) != -1)
■■{
■■■return;
■■}
■■
■■m_AnimEvents.InsertAt(new DayZCreatureAnimScriptDebugAnimEventData(GetWorldTime(), evType,
■■
■■if (m_AnimEvents.Count() > m_iMaxAnimEventsCount)
■■{
■■■m_AnimEvents.Remove(m_AnimEvents.Count() - 1);
■■}
■}
■
■void OnAnimationPrediction(string predName)
■{
■■m_AnimPredictions.Insert(new DayZCreatureAnimScriptDebugAnimPredictionData(predName));
■}
■
```

```
// ---------------------------------------------
// ------ PluginDayZInfectedDebug.c - START --------------------
// ---------------------------------------------


// *****************************************************************************
// ! PluginDayZInfectedDebugUIHandler
// *****************************************************************************
class PluginDayZInfectedDebugUIHandler extends ScriptedWidgetEventHandler
{
■override bool OnClick(Widget w, int x, int y, int button)
■{
■■super.OnClick(w, x, y, button);
■■return m_pPluginInfectedDebug.OnClick(w, x, y, button);
■}

■override bool OnChange(Widget w, int x, int y, bool finished)
■{
■■super.OnChange(w, x, y, finished);
■■return m_pPluginInfectedDebug.OnChange(w, x, y, finished);
■}

■PluginDayZInfectedDebug■■m_pPluginInfectedDebug;
}

// *****************************************************************************
// ! PluginDayZInfectedDebugAttackDesc
// *****************************************************************************
class PluginDayZInfectedDebugAttackDesc
{
■void PluginDayZInfectedDebugAttackDesc(string pName, int pValue)
■{
■■name = pName;
■■animValue = pValue;
■}
■
■string name;
■int animValue;
}

// *****************************************************************************
// ! PluginDayZInfectedDebug
// *****************************************************************************
class PluginDayZInfectedDebug extends PluginBase
{
■int m_CurrentMode = 0;
■bool m_HasFocus = false;
■bool m_IsActive = false;

■//! main controls■
■Widget ■■■■■■■■m_MainWnd;
■ButtonWidget■■■■■■m_SpawnEntityButton;
■ButtonWidget■■■■■■m_CardMovementButton;
■ButtonWidget■■■■■■m_CardFightButton;
■Widget■■■■■■■■m_CardMovementFrame;
■Widget■■■■■■■■m_CardFightFrame;
■TextWidget■■■■■■■m_StatusText;
■
■//! movement card
■XComboBoxWidget ■■■■■m_StanceCB;
■XComboBoxWidget ■■■■■m_MindStateCB;
■EditBoxWidget■■■■■■m_MovementSpeedEB;
■
```

```
// ---------------------------------------------
// ------ PluginDayzPlayerDebug.c - START --------------------
// ---------------------------------------------


// *****************************************************************************************
// ! PluginDayzPlayerDebugUserData
// *****************************************************************************************
class PluginDayzPlayerDebugUserData
{
■void PluginDayzPlayerDebugUserData(int pUserData, bool pFullBody, int pStanceMask = -1)
■{
■■m_iUserData ■= pUserData;
■■m_bIsFullBody■= pFullBody;
■■m_iStanceMask■= pStanceMask;
■}
■■

■int■■GetInt()
■{
■■return m_iUserData;
■}

■bool■IsFullBody()
■{
■■■return m_bIsFullBody;
■}
■
■int ■GetStanceMask()
■{
■■■return m_iStanceMask;
■}

■protected int ■m_iUserData;
■protected bool■m_bIsFullBody;
■protected int ■m_iStanceMask;
}

// *****************************************************************************************
// ! PluginDayzPlayerDebugUI
// *****************************************************************************************
class PluginDayzPlayerDebugUIHandler extends ScriptedWidgetEventHandler
{
■override bool OnClick(Widget w, int x, int y, int button)
■{
■■super.OnClick(w, x, y, button);
■■return m_pPluginPlayerDebug.OnClick(w, x, y, button);
■}
■
■override bool OnChange(Widget w, int x, int y, bool finished)
■{
■■super.OnChange(w, x, y, finished);
■■return m_pPluginPlayerDebug.OnChange(w, x, y, finished);
■}


■PluginDayzPlayerDebug■■m_pPluginPlayerDebug;
}


// *****************************************************************************************
// ! PluginDayzPlayerDebugUI
// *****************************************************************************************
// ---------------------------------------------
// ------ PluginDayzPlayerDebug_Ctrl.c - START --------------------
// ---------------------------------------------




// *************************************************************************************
// ! PluginDayzPlayerDebug_Ctrl
// *************************************************************************************
class PluginDayzPlayerDebug_Ctrl
{■
■Widget ■■■■■■■■■m_MainWnd;

■XComboBoxWidget ■■■■■■m_ForceStance;
■EditBoxWidget■■■■■■■m_MovSpeedEB;
■EditBoxWidget■■■■■■■m_MovDirEB;
■CheckBoxWidget■■■■■■■m_MovOverrideCheckbox;
■//

■int ■■■■■■■■■m_ApplyStanceChange;
■bool■■■■■■■■■m_OverrideMovementChange = false;


■//--------------------------------------------------
  // gui stuff

■void PluginDayzPlayerDebug_Ctrl(Widget pMainWnd)
■{
■■m_MainWnd ■■■= pMainWnd;
■■m_ApplyStanceChange = -2;

■■CreateModuleWidgets();
■}
■
■
■void ~PluginDayzPlayerDebug_Ctrl()
■{
■■DestroyModuleWidgets();
■}
■
■
■void CreateModuleWidgets()
■{
■■m_ForceStance = XComboBoxWidget.Cast( m_MainWnd.FindAnyWidget("StanceV") );
■■m_MovSpeedEB = EditBoxWidget.Cast( m_MainWnd.FindAnyWidget("MovSpeedEB") );
■■m_MovDirEB = EditBoxWidget.Cast( m_MainWnd.FindAnyWidget("MovDirEB") );
■■m_MovOverrideCheckbox = CheckBoxWidget.Cast( m_MainWnd.FindAnyWidget("OverrideMovCheck
■}
■
■void DestroyModuleWidgets()
■{
■}

■

■//--------------------------------------------------
    // main update

■
■/* void Tick()
■{
■■DayZPlayer player = DayZPlayer.Cast( GetGame().GetPlayer() );
```

```
// ---------------------------------------------
// ------ PluginDayzPlayerDebug_OtherCmds.c - START --------------------
// ---------------------------------------------


// *************************************************************************************
// ! PluginDayzPlayerDebug_OtherCmds
// *************************************************************************************

class PluginDayzPlayerDebug_OtherCmds
{■
■// widgets
■Widget■■■■■■■■m_MainWnd;

■XComboBoxWidget ■■■■■m_DeathTypeCB;
■EditBoxWidget ■■■■■■m_DeathDirectionEdit;
■ButtonWidget■■■■■■m_DeathStartButton;
■
■XComboBoxWidget ■■■■■m_HitTypeCB;
■ButtonWidget■■■■■■m_HitStartButton;

■XComboBoxWidget ■■■■■m_UnconTypeCB;
■ButtonWidget■■■■■■m_UnconStartButton;
■ButtonWidget■■■■■■m_UnconEndButton;
■
■// command handler properties
■bool■■■■■■■■m_CH_DeathStart = false;
■bool■■■■■■■■m_CH_HitStart = false;
■bool■■■■■■■■m_CH_UnconStart = false;
■bool■■■■■■■■m_CH_UnconEnd■= false;

■
■//--------------------------------------------------
■// gui stuff

■void PluginDayzPlayerDebug_OtherCmds(Widget pMainWnd)
■{
■■m_MainWnd = pMainWnd;
■■CreateModuleWidgets();
■}
■
■
■void ~PluginDayzPlayerDebug_OtherCmds()
■{
■■DestroyModuleWidgets();
■}
■
■
■void CreateModuleWidgets()
■{
■■m_DeathTypeCB■■■= XComboBoxWidget.Cast( m_MainWnd.FindAnyWidget("DeathTypeCB") );
■■m_DeathDirectionEdit■= EditBoxWidget.Cast( m_MainWnd.FindAnyWidget("DeathDirectionEdit") );
■■m_DeathStartButton■■= ButtonWidget.Cast( m_MainWnd.FindAnyWidget("DeathStartButton") );
■■
■■m_HitTypeCB■■■■= XComboBoxWidget.Cast( m_MainWnd.FindAnyWidget("HitTypeCB") );
■■m_HitStartButton■■= ButtonWidget.Cast( m_MainWnd.FindAnyWidget("HitStartButton") );

■■m_UnconTypeCB■■■= XComboBoxWidget.Cast( m_MainWnd.FindAnyWidget("UnconTypeCB") );
■■m_UnconStartButton■■= ButtonWidget.Cast( m_MainWnd.FindAnyWidget("UnconStartButton") );
■■m_UnconEndButton■■= ButtonWidget.Cast( m_MainWnd.FindAnyWidget("UnconEndButton") );
■}
■
■void DestroyModuleWidgets()
```

```
// ---------------------------------------------
// ------ PluginDayzPlayerDebug_Weapons.c - START --------------------
// ---------------------------------------------



// *************************************************************************************
// ! PluginDayzPlayerWeaponsDebugUserData
// *************************************************************************************
class PluginDayzPlayerWeaponsDebugUserData
{
■void PluginDayzPlayerWeaponsDebugUserData(int pAction, int pActionType)
■{
■■m_iAction = pAction;
■■m_iActionT = pActionType;
■}
■■

■int■■GetAction()
■{
■■return m_iAction;
■}

■int■■GetActionType()
■{
■■return m_iActionT;
■}


■protected int ■m_iAction;
■protected int ■m_iActionT;
}


// *************************************************************************************
// ! PluginDayzPlayerDebug_AbilityConfig
// *************************************************************************************
class PluginDayzPlayerDebug_AbilityConfig
{
■void PluginDayzPlayerDebug_AbilityConfig(string pName, int pAction, int pActionT)
■{
■■m_Name = pName;
■■m_iAction = pAction;
■■m_iActionT = pActionT;
■}

■string■■m_Name;
■int■■■■m_iAction;
■int■■■■m_iActionT;
};



// *************************************************************************************
// ! PluginDayzPlayerDebug
// *************************************************************************************
class PluginDayzPlayerDebug_Weapons
{■
■Widget ■■■■■■■■■m_MainWnd;

■TextListboxWidget■■■■m_Selector;
■ButtonWidget■■■■■■m_ButtonStart;
■TextWidget■■■■■■■m_WidgetActionRunning;■
```

```
// ---------------------------------------------
// ------ PluginDeveloper.c - START --------------------
// ---------------------------------------------


class PluginDeveloper extends PluginBase
{
■protected bool■■■■■■■■■m_IsWinHolding;■
■protected int■■■■■■■■■m_FeaturesMask;
■UIScriptedMenu■■■■■■■■■m_ScriptConsole;
■
■#ifdef DEVELOPER
■ref PresetSpawnBase ■■■■■■■■m_PresetObj;
■#endif
■
■
■static PluginDeveloper GetInstance()
■{
■■return PluginDeveloper.Cast( GetPlugin( PluginDeveloper ) );
■}
■
■//! Set Player position at his cursor position in the world
■void TeleportAtCursor()
■{
■■DeveloperTeleport.TeleportAtCursorEx();
■}
■
■//! Teleport player at position
■void Teleport(PlayerBase player, vector position)
■{
■■DeveloperTeleport.SetPlayerPosition(player, position);
■}
■
■//! Enable / Disable Free camera (Fly mod)
■void ToggleFreeCameraBackPos()
■{
■■DeveloperFreeCamera.FreeCameraToggle( PlayerBase.Cast( GetGame().GetPlayer() ), false );
■}
■
■//! Enable / Disable Free camera (Fly mod) - disable of camera will teleport player at current free camera
■void ToggleFreeCamera()
■{
■■DeveloperFreeCamera.FreeCameraToggle( PlayerBase.Cast( GetGame().GetPlayer() ), true );
■}
■■
■bool IsEnabledFreeCamera()
■{
■■return DeveloperFreeCamera.IsFreeCameraEnabled();
■}■
■
■// System Public Events
■void PluginDeveloper()
■{
■■
■}

■override void OnInit()
■{
■■super.OnInit();
■■
■■DeveloperFreeCamera.OnInit();
■}
■
```

```
// ---------------------------------------------
// ------ PluginDeveloperSync.c - START --------------------
// ---------------------------------------------


enum PDS_SYSTEMS
{
■STATS ■■■■= 1,
■LEVELS ■■■■= 2,
■MODS ■■■■= 4,
■AGENTS ■■■■= 8,
■STOMACH ■■■= 16,
■MODS_DETAILED ■■= 32,
■TEMPERATURE ■■= 64,
}


class RegisterPlayerData
{
■int m_Bitmask;
■int m_DetailedModifierIndex;
■
■void RegisterPlayerData()
■{
■■m_Bitmask = 0;
■■m_DetailedModifierIndex = 0;
■}
}

class PluginDeveloperSync extends PluginBase
{
■int m_DetailedInfoRequested = 0;
■ref Timer m_UpdateTimer;
■ref map<PlayerBase, ref RegisterPlayerData> m_RegisteredPlayers;
■
■ref array<ref SyncedValue> m_PlayerStatsSynced;
■ref array<ref Param> m_PlayerStomachSynced;
■ref array<ref SyncedValueLevel> m_PlayerLevelsSynced;
■ref array<ref SyncedValueModifier> m_PlayerModsSynced;
■string ■m_PlayerModsDetailedSynced;
■string ■m_EnvironmentDebugMessage;
■ref array<ref SyncedValueAgent> m_PlayerAgentsSynced;
■
■bool m_StatsUpdateStatus;
■bool m_LevelsUpdateStatus;
■bool m_ModsUpdateStatus;
■bool m_AgentsUpdateStatus;
■bool m_StomachUpdateStatus;
■
■void PluginDeveloperSync()
■{
■■m_RegisteredPlayers ■= new map<PlayerBase, ref RegisterPlayerData>;
■■m_PlayerStatsSynced ■= new array<ref SyncedValue>;
■■m_PlayerLevelsSynced ■= new array<ref SyncedValueLevel>;
■■m_PlayerModsSynced ■■= new array<ref SyncedValueModifier>;
■■m_PlayerAgentsSynced ■= new array<ref SyncedValueAgent>;
■■m_PlayerStomachSynced ■= new array<ref Param>;
■■
■■m_StatsUpdateStatus ■= false;
■■m_LevelsUpdateStatus ■= false;
■■m_ModsUpdateStatus ■■= false;
■■m_AgentsUpdateStatus ■= false;
■■m_StomachUpdateStatus ■= false;
```

```
// ---------------------------------------------
// ------ PluginDiagMenu.c - START --------------------
// ---------------------------------------------


// For modding, see PluginDiagMenuModding.c
// !!! MODDING DISCLAIMER: These are debug functionality files, if you are thinking about modding the va
//                        These files will not be maintained with the thought of "what if a modder modded this" (Exc
//                        Which is why the modding functionality was developed with the thought of the modded on

enum DebugActionType
{
■GENERIC_ACTIONS = 1,
■PLAYER_AGENTS = 2,
■UNLIMITED_AMMO = 4,
};

#ifdef DIAG_DEVELOPER
enum ESubscriberSystems
{
■TRIGGERS = 0x00000001,
■//SYSTEM2 = 0x00000002,
■//SYSTEM3 = 0x00000004,
■//SYSTEM4 = 0x00000008,
}

typedef Param1<int> SendDiagRPCSelfBasicParam;
typedef Param2<int,Param> SendDiagRPCSelfParamParam;

typedef Param2<int,bool> SendDiagRPCBasicParam;
typedef Param3<int,bool,Param> SendDiagRPCParamParam;
#endif

class PluginDiagMenu : PluginBase
{
#ifdef DIAG_DEVELOPER
■
■int m_ModdedDiagID = DiagMenuIDs.MODDED_MENU;
■
■// LEVEL 3 - Script > Misc > Hair Hiding
■ref map<int,bool> m_HairHidingStateMap;
■ref TStringArray m_HairSelectionArray;
■int m_TotalHairLevelsAdjusted;
■
■override void OnInit()
■{■■■■
■■//----------------------
■■m_HairHidingStateMap = new map<int,bool>;
■■m_HairSelectionArray = new TStringArray;
■■
■■g_Game.ConfigGetTextArray("cfgVehicles Head_Default simpleHiddenSelections", m_HairSelectionAr
■■m_TotalHairLevelsAdjusted = m_HairSelectionArray.Count() - 1;
■■for (int i = 0; i < m_HairSelectionArray.Count(); ++i)
■■{
■■■m_HairHidingStateMap.Insert(i, 1); //all considered "shown" on init
■■}
■■
■■//----------------------
■■
■■DiagMenu.InitScriptDiags();
■■
■■RegisterDiags();
■}
```

```
// ---------------------------------------------
// ------ PluginDiagMenuClient.c - START --------------------
// ---------------------------------------------


// For modding, see PluginDiagMenuModding.c
// !!! MODDING DISCLAIMER: These are debug functionality files, if you are thinking about modding the va
//                         These files will not be maintained with the thought of "what if a modder modded this" (Exc
//                         Which is why the modding functionality was developed with the thought of the modded or

class PluginDiagMenuClient : PluginDiagMenu
{
#ifdef DIAG_DEVELOPER
■static int m_SystemsMask;
■
■// Cheats
■bool m_ModifiersDisabled;
■int m_IsInvincible;■
■bool m_StaminaDisabled;
■
■// Misc
■float m_Playtime;
■bool m_LogPlayerStats;■
■Shape m_VehicleFreeAreaBox;■
■ref EnvDebugData m_EnvDebugData;
■
■override void OnInit()
■{
■■super.OnInit();
■■
■■BindCallbacks();
■}
■
■protected void BindCallbacks()
■{■
■■//------------------------------------------------------------
■■// LEVEL 2 - Script > Crafting
■■//------------------------------------------------------------
■■DiagMenu.BindCallback(DiagMenuIDs.CRAFTING_GENERATE, CBCraftingGenerate);
■■DiagMenu.BindCallback(DiagMenuIDs.CRAFTING_INSTANT, CBCraftingInstant);
■■DiagMenu.BindCallback(DiagMenuIDs.CRAFTING_DUMP, CBCraftingDump);
■■
■■//------------------------------------------------------------
■■// LEVEL 2 - Script > Vehicles
■■//------------------------------------------------------------■
■■DiagMenu.BindCallback(DiagMenuIDs.VEHICLE_DEBUG_OUTPUT, CBVehicleDebugOutput);
■■DiagMenu.BindCallback(DiagMenuIDs.VEHICLE_DUMP_CRASH_DATA, CBDumpCrashData);
■■
■■//------------------------------------------------------------
■■// LEVEL 2 - Script > Cheats
■■//------------------------------------------------------------
■■DiagMenu.BindCallback(DiagMenuIDs.CHEATS_MODIFIERS, CBCheatsModifiers);
■■DiagMenu.BindCallback(DiagMenuIDs.CHEATS_KILL_PLAYER, CBCheatsKillPlayer);
■■DiagMenu.BindCallback(DiagMenuIDs.CHEATS_INVINCIBILITY, CBCheatsInvincibility);
■■DiagMenu.BindCallback(DiagMenuIDs.CHEATS_DISABLE_STAMINA, CBCheatsStaminaDisable);
■■DiagMenu.BindCallback(DiagMenuIDs.CHEATS_RESET_PLAYER, CBCheatsResetPlayer);
■■DiagMenu.BindCallback(DiagMenuIDs.CHEATS_RESET_PLAYER_MAX, CBCheatsResetPlayerMax)
■■DiagMenu.BindCallback(DiagMenuIDs.CHEATS_INVENTORY_ACCESS, CBCheatsInventoryAccess)
■■DiagMenu.BindCallback(DiagMenuIDs.CHEATS_FIX_ITEMS, CBCheatsFixItems);
■■
■■//------------------------------------------------------------
■■// LEVEL 2 - Script > Player Agents
■■//------------------------------------------------------------
```

```
// ---------------------------------------------
// ------ PluginDiagMenuModding.c - START --------------------
// ---------------------------------------------


/**
 * \defgroup Modding Modding DiagMenu
 *■\warning Only available on developer and diag builds
 *■\note This file is extensive documentation about modding DiagMenu, please read EVERYTHING before
 * @{
 */

#ifdef DOXYGEN
#ifdef MODDING_TEST
#ifdef DIAG_DEVELOPER

/**
\brief Example of adding DiagMenu entries by modders
■\note PluginDiagMenu is the base of ..Client and ...Server, which is why we are registering the IDs on thi
■\note Keep in mind that in SinglePlayer missions, both Client and Server PluginDiagMenu are created
■\note Keep in mind that PluginDiagMenu... is destroyed and recreated every time the world changes
*/
#ifdef DOXYGEN
class PluginDiagMenuModded // Just to not have it show up for the regular entry... Doxygen doesn't know
#else
modded class PluginDiagMenu
#endif
{■
■/**
■\brief The name of the root menu where all debugs of this mod will be placed in
■■\note Try to think of a name which will make it easy to identify as originating from your mod
■■\note The 'maximum' length of the name of an entry is 24 characters (excess will be cut off when rende
■■\note The hardcap length of the name of an entry is 64 characters
■*/
■protected string m_ModdedDiagsExampleRootMenu = "BI - DiagsModdingExample";
■
■/**
■\brief The name of an example sub menu
■*/
■protected string m_ModdedDiagsExampleSubMenu = "Example Sub Menu";
■
■/** \name Modded Diag IDs
■*■To prevent mod conflicts, a system has been set up to create unique IDs for the modded Diags
■*■These are then best saved to a variable so that they can be used by the other DiagMenu functions
■*■!!! Remember to give them as unique as possible name, as something with a generic name could caus
■*/
■//@{
■protected int m_ModdedDiagsExampleRootMenuID;
■■protected int m_ModdedDiagsExampleBoolID;
■■protected int m_ModdedDiagsExampleSubMenuID;
■■■protected int m_ModdedDiagsExampleRangeID;
■//@}
■
■//---------------------------------------------
■/**
■\brief Obtain unique IDs and store them in variables
■■\warning Please only call GetModdedDiagID when necessary, as every time it is called it will increment
■■\note Don't forget to call super!
■*/
■override protected void RegisterModdedDiagsIDs()
■{
■■super.RegisterModdedDiagsIDs();
■■
```

```
// ---------------------------------------------
// ------ PluginDiagMenuServer.c - START --------------------
// ---------------------------------------------


// For modding, see PluginDiagMenuModding.c
// !!! MODDING DISCLAIMER: These are debug functionality files, if you are thinking about modding the va
//                         These files will not be maintained with the thought of "what if a modder modded this" (Exc
//                         Which is why the modding functionality was developed with the thought of the modded or

class PluginDiagMenuServer : PluginDiagMenu
{
#ifdef DIAG_DEVELOPER
■static ref map<Man, int> m_Subscribers = new map<Man, int>;
■
■// A bit of a hack, because SP creates both Client and Server Plugins
■override private void RegisterDiags()
■{
■■if (GetGame().IsMultiplayer())
■■{
■■■super.RegisterDiags();
■■}
■}
■
■//---------------------------------------------
■override void OnRPC(PlayerBase player, int rpc_type, ParamsReadContext ctx)
■{
■■super.OnRPC(player, rpc_type, ctx);
■■
■■switch (rpc_type)
■■{■■■
■■■case ERPCs.DEV_DIAGMENU_SUBSCRIBE:
■■■{
■■■■if (ctx.Read(CachedObjectsParams.PARAM1_INT))
■■■■{
■■■■■int newMask = CachedObjectsParams.PARAM1_INT.param1;
■■■■■int currentMask = m_Subscribers.Get(player);
■■■■
■■■■■if (newMask != currentMask)
■■■■■{
■■■■■■if (newMask == 0)
■■■■■■{
■■■■■■■m_Subscribers.Remove(player);
■■■■■■}
■■■■■■else
■■■■■■{
■■■■■■■m_Subscribers.Set(player, newMask);
■■■■■■}
■■■■■}
■■■■}
■■■■break;
■■■}
■■■case ERPCs.DIAG_VEHICLE_DEBUG_OUTPUT:
■■■{
■■■■if (ctx.Read(CachedObjectsParams.PARAM1_INT))
■■■■■CarScript.DEBUG_OUTPUT_TYPE = CachedObjectsParams.PARAM1_INT.param1;
■■■■break;
■■■}
■■■
■■■case ERPCs.DIAG_VEHICLES_DUMP_CRASH_DATA_REQUEST:
■■■{
■■■■if (ctx.Read(CachedObjectsParams.PARAM1_BOOL))
■■■■■CrashDebugData.SendData(player);
```

```
// ---------------------------------------------
// ------ PluginDoorRuler.c - START --------------------
// ---------------------------------------------


class PluginDoorRuler extends PluginBase
{
■PluginDeveloper m_ModuleDeveloper;
■ItemBase m_Ruler;
■ref Timer m_Timer;
■bool m_Initialized;
■
■void CheckInit()
■{
■■if( FreeDebugCamera.GetInstance().IsActive() )
■■■Init();
■}
■
■void Init()
■{
■■if(m_Ruler)
■■■m_Ruler.Delete();
■■
■■m_Initialized = true;
■■m_Ruler = ItemBase.Cast( GetGame().CreateObject("DoorTestCamera", FreeDebugCamera.GetInstan
■}
■
■override void OnUpdate(float delta_time)
■{
■■if(!m_Initialized) return;
■■if(!FreeDebugCamera.GetInstance().IsActive()) return;
■■if(!m_Ruler) return;
■■vector cam_pos = FreeDebugCamera.GetInstance().GetPosition();
■■vector ruler_pos = m_Ruler.GetPosition();
■■vector camera_dir = FreeDebugCamera.GetInstance().GetDirection();
■■camera_dir.Normalize();
■■m_Ruler.SetPosition(FreeDebugCamera.GetInstance().GetPosition() + ( camera_dir * 2) );
■■m_Ruler.SetAngles(FreeDebugCamera.GetInstance().GetAngles());
■}
}



// ---------------------------------------------
// ------ PluginDoorRuler.c - END ----------------------
// ---------------------------------------------


// ---------------------------------------------
// ------ PluginDrawCheckerboard.c - START --------------------
// ---------------------------------------------


// quick and dirty way for displaying of checker overlay on screen
// - userd for camera settings primarily
class PluginDrawCheckerboard extends PluginBase
{
■private■ref Widget■m_MainWindow;
■private bool■■m_IsActive;
■
■
■void PluginDrawCheckerboard()
■{
```

```
// ----------------------------------------------
// ------ PluginFileHandler.c - START --------------------
// ----------------------------------------------


class PluginFileHandler extends PluginBase
{
■static bool FileDuplicate(string source_name, string dest_name)
■{
■■return CopyFile(source_name, dest_name);
■}

■static bool FileDelete(string file)
■{
■■return DeleteFile(file);
■}

■static void FileRename(string source_name, string dest_name)
■{
■■FileDuplicate(source_name, dest_name);
■■FileDelete(source_name);
■}■
■
■bool■m_ReadOnly;
■int■■m_LineLimit;
■
■ref TStringArray■m_FileContent;
■
■void PluginFileHandler()
■{
■■m_FileContent = new TStringArray;
■■m_LineLimit = -1;
■■
■■LoadFile();
■}

■void ~PluginFileHandler()
■{
■■ClearFileNoSave();
■}
■
■override void OnInit()
■{
■■super.OnInit();
■}
■
■string GetFileName()
■{
■■return string.Empty;
■}
■
■void ClearFileNoSave()
■{
■■m_FileContent.Clear();■
■}

■bool LoadFile()
■{
■■ClearFileNoSave();

■■FileHandle file_index = OpenFile(GetFileName(), FileMode.READ);
■■
■■if ( file_index == 0 )
```

```
// ---------------------------------------------
// ------ PluginHorticulture.c - START --------------------
// ---------------------------------------------


class PluginHorticulture extends PluginBase
{
■ref map<string, ref PlantMaterialHealth> m_PlantMaterials;
■
■void PluginHorticulture()
■{
■■m_PlantMaterials = new map<string, ref PlantMaterialHealth>;
■■
■■LoadFromCfg();
■}

■void LoadFromCfg()
■{
■■string cfg_access = "CfgHorticulture"; // it is stored in dz\gear\cultivation\cfgHorticulture.hpp
■■int cfg_horticulture_count = GetGame().ConfigGetChildrenCount( cfg_access );

■■for ( int i = 0; i < cfg_horticulture_count; i++ )
■■{
■■■string cfg_class_name = "";
■■■GetGame().ConfigGetChildName( cfg_access, i, cfg_class_name );
■■■string cfg_class_access = cfg_access + " " + cfg_class_name;

■■■int cfg_class_count = GetGame().ConfigGetChildrenCount( cfg_class_access );
■■
■■■for ( int j = 0; j < cfg_class_count; j++ )
■■■{
■■■■string cfg_subclass_name = "";
■■■■GetGame().ConfigGetChildName( cfg_class_access, j, cfg_subclass_name );
■■■■string cfg_subclass_access = cfg_class_access + " " + cfg_subclass_name;
■■
■■■■int cfg_subclass_count = GetGame().ConfigGetChildrenCount( cfg_subclass_access );
■■■■
■■■■PlantMaterialHealth plantMaterialHealth = NULL;
■■■■
■■■■if ( cfg_class_name == "Plants" )
■■■■{
■■■■■plantMaterialHealth = new PlantMaterialHealth;
■■■■■m_PlantMaterials.Set( cfg_subclass_name, plantMaterialHealth );
■■■■}
■■■■
■■■■for ( int k = 0; k < cfg_subclass_count; k++ )
■■■■{
■■■■■string cfg_variable_name = "";
■■■■■GetGame().ConfigGetChildName( cfg_subclass_access, k, cfg_variable_name );
■■■■■string cfg_variable_access = cfg_subclass_access + " " + cfg_variable_name;
■■■■■
■■■■■if ( cfg_class_name == "Plants" )
■■■■■{
■■■■■■string string_param = "";
■■■■■■GetGame().ConfigGetText( cfg_variable_access, string_param );
■■■■■■
■■■■■■if ( cfg_variable_name == "infestedTex" )
■■■■■■{
■■■■■■■plantMaterialHealth.m_InfestedTex = string_param;
■■■■■■}
■■■■■■else if ( cfg_variable_name == "infestedMat" )
■■■■■■{
■■■■■■■plantMaterialHealth.m_InfestedMat = string_param;
```

```
// ---------------------------------------------
// ------ PluginItemDiagnostic.c - START --------------------
// ---------------------------------------------


class PluginItemDiagnosticEventHandler extends ScriptedWidgetEventHandler
{
■PluginItemDiagnostic m_Owner;
■override bool OnMouseEnter(Widget w, int x, int y)
■{
■■if( ButtonWidget.Cast(w))
■■{
■■■GetGame().GetMission().AddActiveInputExcludes({"menu"});
■■}
■■return true;

■}
■
■override bool OnMouseLeave(Widget w, Widget enterW, int x, int y)
■{
■■if( ButtonWidget.Cast(w))
■■{
■■■GetGame().GetMission().RemoveActiveInputExcludes({"menu"},true);
■■}
■■
■■return true;
■}
■
■override bool OnClick( Widget w, int x, int y, int button )
■{
■■return m_Owner.OnClick( w, x, y, button );
■}
■
■override bool OnMouseButtonDown(Widget w, int x, int y, int button)
■{
■■return m_Owner.OnMouseButtonDown( w, x, y, button );
■}
■override bool OnMouseButtonUp(Widget w, int x, int y, int button)
■{
■■return m_Owner.OnMouseButtonUp( w, x, y, button );
■}
}



class PluginItemDiagnostic extends PluginDeveloper
{
■Object m_Entity;
■
■
■ref Timer ■■■■■■myTimer1;
■ref map<PlayerBase,Object> ■m_ObserversMap = new map<PlayerBase,Object>;
■ref array<string> ■■■■m_Agents = new array<string>;
■ref map<string,float> ■■■m_Floats = new map<string,float>;
■ref map<string,float> ■■■m_VarsNumbersFinalServer = new map<string,float>;
■ref map<string,float> ■■■m_VarsFinalClient = new map<string,float>;
■ref array<ref Param>■■m_Properties = new array<ref Param>;
■bool ■■■■■■■■■m_IsActive■= false;
■bool■■■■■■■■■m_ScriptMenuOpened;
■string■■■■■■■■■m_NoteClient;
■ref PluginItemDiagnosticEventHandler ■m_EventHandler;
■bool■■■■■■■■■m_IsDragging;
■PluginConfigDebugProfile m_ConfigDebugProfile;
```

```
// ---------------------------------------------
// ------ PluginKeyBinding.c - START --------------------
// ---------------------------------------------


/**
 * \defgroup MouseEvents Mouse Events
 * \desc constants for mouse events in PluginKeyBinding
 * @{
 */

const int MB_EVENT_PRESS■■■= 0;
const int MB_EVENT_CLICK■■■= 1;
const int MB_EVENT_RELEASE■■■= 2;
const int MB_EVENT_DOUBLECLICK■■= 3;
const int MB_EVENT_DRAG■■■■= 4;
/** @}*/

class PluginKeyBinding extends PluginBase
{
■static PluginKeyBinding instance;
■
■// System Public Events
■void PluginKeyBinding()
■{
■■if ( instance == null )
■■{
■■■instance = this;
■■}
■}

■//==============================================
■// OnInit
■//==============================================
■override void OnInit()
■{
■■super.OnInit();
■■
■■m_KeyBindings■= new array<ref KeyBinding>;
■■m_MouseBindings■= new array<ref MouseBinding>;
■■m_MouseButtons■= new array<ref MouseButtonInfo>;
■■m_MouseButtons.Insert ( new MouseButtonInfo( MouseState.LEFT ) );
■■m_MouseButtons.Insert ( new MouseButtonInfo( MouseState.RIGHT ) );
■■m_MouseButtons.Insert ( new MouseButtonInfo( MouseState.MIDDLE ) );
■■GetGame().GetUpdateQueue(CALL_CATEGORY_SYSTEM).Insert(this.OnFrame);
■■
■■/////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
■■//Keyboard Binds■|UI_ID■■■■■■■|Key1■■■■■|Key2■■■■|Callback Plugin■■|Callback Function
■■//-----------------------------------------------------------------------------------------------------------------------------------
■■//■■■■■|constants.h■■■■■|constants.h■■■|constants.h■■|only plugin name■■|only function■■■
■■//■■■■■|MENU_***■■■■■■|KeyCode.KC_***■■■|KeyCode.KC_***■■|■■■■■■|in plugin■■■■■
■■//-----------------------------------------------------------------------------------------------------------------------------------
■■#ifdef DEVELOPER
■■RegisterKeyBind(■ MENU_ANY■■■■■■,KeyCode.KC_LCONTROL■,KeyCode.KC_F1■■,"PluginDe
■■RegisterKeyBind(■ MENU_ANY■■■■■■,KeyCode.KC_LCONTROL■,KeyCode.KC_F2■■,"PluginDe
■■RegisterKeyBind(■ MENU_NONE|MENU_SCRIPTCONSOLE■,KeyCode.KC_LCONTROL■,KeyCode
■■RegisterKeyBind(■ MENU_NONE|MENU_SCRIPTCONSOLE■,KeyCode.KC_RCONTROL■,KeyCode
■■RegisterKeyBind(■ MENU_NONE|MENU_SCENE_EDITOR■,KeyCode.KC_LCONTROL■,KeyCode.K
■■RegisterKeyBind(■ MENU_NONE|MENU_SCENE_EDITOR■,KeyCode.KC_RCONTROL■,KeyCode.
■■RegisterKeyBind(■ MENU_NONE■■■■■■,KeyCode.KC_LCONTROL■,KeyCode.KC_V■■,"PluginD
■■#endif
■■
■■#ifdef DIAG_DEVELOPER
```

```
// ---------------------------------------------
// ------ PluginLifespan.c - START --------------------
// ---------------------------------------------


//-----------------------------
// LIFESPAN plugin
//-----------------------------
/*
Lifespan plugin handles player facial hair, bloody hands, blood type in HUD
*/

enum eBloodyHandsTypes
{
■CLEAN = 0,//clean needs to be 0
■SALMONELA,
■JUST_BLOOD,
■//--- ONLY LAST_INDEX BELLOW !!!
■LAST_INDEX,
}

enum LifeSpanState
{
■BEARD_NONE■■= 0,
■BEARD_MEDIUM■= 1,
■BEARD_LARGE■■= 2,■
■BEARD_EXTRA■■= 3,■
■COUNT■■■= 4,
}

class PluginLifespan extends PluginBase
{
■protected static const int LIFESPAN_MIN = 0;
■protected static const int LIFESPAN_MAX = 240; // value in minutes when player achieved maximum ag
■protected int m_FakePlaytime;
■
■protected ref map<PlayerBase, ref LifespanLevel> m_PlayerCurrentLevel;
■protected ref map<string, ref array< ref LifespanLevel>> m_LifespanLevels;
■protected ref map<string, ref BloodyHands> m_BloodyHands;
■protected ref map<PlayerBase, int> m_BloodType;

■//=======================================
■// GetInstance
■//=======================================
■static PluginLifespan GetInstance()
■{
■■return PluginLifespan.Cast( GetPlugin(PluginLifespan) );
■}
■
■void PluginLifespan()
■{
■■LoadFromCfg();
■■m_PlayerCurrentLevel = new map<PlayerBase, ref LifespanLevel>;
■■m_BloodType = new map<PlayerBase, int>;
■■m_FakePlaytime = 0;
■}

//-----------------------------
// Player load from config
//-----------------------------
■void LoadFromCfg()
■{
■■m_LifespanLevels = new map<string, ref array< ref LifespanLevel>>;
```

```
// ---------------------------------------------
// ------ PluginLocalEnscriptHistory.c - START --------------------
// ---------------------------------------------


class PluginLocalEnscriptHistory extends PluginLocalHistoryBase
{■
■void PluginLocalEnscriptHistory()
■{
■}
■■
■void ~PluginLocalEnscriptHistory()
■{
■}

■override string GetFileName()
■{
■■return CFG_FILE_ENS_HISTORY;
■}
}

class PluginLocalEnscriptHistoryServer extends PluginLocalHistoryBase
{■
■void PluginLocalEnscriptHistoryServer()
■{
■}
■■
■void ~PluginLocalEnscriptHistoryServer()
■{
■}

■override string GetFileName()
■{
■■return CFG_FILE_ENS_HISTORY_SERVER;
■}
}




// ---------------------------------------------
// ------ PluginLocalEnscriptHistory.c - END ---------------------
// ---------------------------------------------


// ---------------------------------------------
// ------ PluginLocalHistoryBase.c - START --------------------
// ---------------------------------------------


class PluginLocalHistoryBase extends PluginFileHandler
{■
■void PluginLocalHistoryBase()
■{
■■m_ReadOnly = false;
■}
■■
■void ~PluginLocalHistoryBase()
■{
■}
■
■override void OnInit()
■{
```

```
// ---------------------------------------------
// ------ PluginLocalProfile.c - START --------------------
// ---------------------------------------------


class PluginLocalProfile extends PluginFileHandler
{
■ref map<string, string>■■■■■■m_ConfigParams;
■ref map<string, ref TStringArray>■■■■m_ConfigParamsArray;
■ref map<string, ref map<string, string>>■■m_ConfigParamsInArray;
■ref map<string, ref array<ref map<string, string>>>■m_ConfigParamsArrayInArray;
■
■void PluginLocalProfile()
■{
■■m_ConfigParams■■■■= new map<string, string>;
■■m_ConfigParamsArray■■■= new map<string, ref TStringArray>;
■■m_ConfigParamsInArray■■= new map<string, ref map<string, string>>;
■■m_ConfigParamsArrayInArray■= new map<string, ref array<ref map<string, string>>>;
■}

■override string GetFileName()
■{
■■return CFG_FILE_USER_PROFILE;
■}

■override void OnInit()
■{
■■super.OnInit();
■■
■■LoadConfigFile();
■}
■
■bool LoadConfigFile()
■{
■■TStringArray file_content = GetFileContent();
■■
■■string line_content = "";
■■string param_name = "";
■■string param_value = "";
■■TStringArray param_content;
■■int pos_sep = -1;
■■
■■for ( int i = 0; i < file_content.Count(); ++i )
■■{
■■■line_content = file_content.Get(i);
■■■
■■■param_content = ParseTextToParameter(line_content);
■■
■■■if ( param_content.Count() == 0 )
■■■{
■■■■PrintString(GetFileName()+"("+i.ToString()+"): Error in config -> Maybe missing '=' !");
■■■■return false;■■■
■■■}
■■■
■■■param_name■= param_content.Get(0);
■■■param_value■= param_content.Get(1);
■■■
■■■//{{name = apple, damage = 0.5, quantity = 2},{name = banana, damage = 0.5, quantity = 2}}
■■■
■■■TStringArray array_items = new TStringArray;
■■■ParseTextToArray(param_value, array_items);
■■■//{name = apple, damage = 0.5, quantity = 2}
■■■//{name = banana, damage = 0.5, quantity = 2}
```

```
// --------------------------------------------
// ------ PluginLocalProfileScene.c - START --------------------
// --------------------------------------------


/*
//Mission = "ChernarusPlus"
typedef CfgParamString DefCfgParamString

//InitTime = 220
typedef CfgParamInt DefCfgParamInt

//InitWeatherRain = 0.5
typedef CfgParamFloat DefCfgParamFloat

//InitWeatherFog = 0.3
typedef CfgParamFloat DefCfgParamFloat

//SceneObjectsCount = 1
typedef CfgParamInt DefCfgParamInt
*/

class PluginLocalProfileScene extends PluginLocalProfile
{
■private const string FILE_ROOT■■■= "$saves:";
■private const string FILE_ROOT_SCENES■= "Scenes";
■private const string PARAM_MISSION■■= "Mission";
■private const string PARAM_TIME■■■= "InitTime";
■private const string PARAM_RAIN■■■= "InitWeatherRain";
■private const string PARAM_FOG■■■= "InitWeatherFog";
■private const string PARAM_OBJ_COUNT■= "SceneObjectsCount";
■private const string PARAM_OBJ_NAME■■= "SceneObject";
■
■private string m_FileSceneName;
■
■//=========================================
■// GetPathScenes
■//=========================================
■string GetPathScenes()
■{
■■return FILE_ROOT+"\\"+FILE_ROOT_SCENES;
■}
■
■//=========================================
■// GetFileName
■//=========================================
■override string GetFileName()
■{
■■string file_name = GetPathScenes()+"\\"+m_FileSceneName;
■■return file_name;
■}
■
■//=========================================
■// OnInit
■//=========================================
■override void OnInit()
■{
■■super.OnInit();
■}
■
■//=========================================
■// GetSceneList
■//=========================================■
```

```
// ---------------------------------------------
// ------ PluginManager.c - START --------------------
// ---------------------------------------------


class PluginManager
{■
■private ref array<typename>■■■■■m_PluginRegister;■// list of modules for creation
■private ref map<typename, ref PluginBase>■m_PluginsPtrs;■■// plugin, plugin pointer
■
■//=================================
■// Constructor
■//=================================
■void PluginManager()
■{
■■m_PluginRegister■= new array<typename>;
■■m_PluginsPtrs■■= new map<typename, ref PluginBase>;
■}
■
■//=================================
■// Destructor
■//=================================
■void ~PluginManager()
■{
■■int i;
■■PluginBase plugin;
■■
■■for ( i = 0; i < m_PluginsPtrs.Count(); ++i)
■■{
■■■plugin = m_PluginsPtrs.GetElement(i);
■■■if ( plugin != NULL )
■■■{
■■■■plugin.OnDestroy();
■■■■delete plugin;
■■■}
■■}
■■
■■GetGame().GetUpdateQueue(CALL_CATEGORY_GAMEPLAY).Remove(this.MainOnUpdate);
■}
■
■//=================================
■// Init
■//=================================
■void Init()
■{■
■■//------------------------------------------------------------------
■■// Register modules
■■//------------------------------------------------------------------
■■//■■■■Module Class Name ■■■■■■■■Client■Server
■■//------------------------------------------------------------------
■■RegisterPlugin( "PluginHorticulture",■■■■■■■true, ■true );
■■RegisterPlugin( "PluginRepairing",■■■■■■■■true, ■true );
■■RegisterPlugin( "PluginPlayerStatus",■■■■■■■true, ■true );
■■RegisterPlugin( "PluginMessageManager",■■■■■■■true, ■true );
■■RegisterPlugin( "PluginLifespan",■■■■■■■■true, ■true );
■■RegisterPlugin( "PluginVariables",■■■■■■■■true, ■true );
■■RegisterPlugin( "PluginObjectsInteractionManager",■■■■false, ■true );
■■RegisterPlugin( "PluginRecipesManager",■■■■■■■true, ■true );
■■RegisterPlugin( "PluginTransmissionAgents",■■■■■■true, ■true );
■■RegisterPlugin( "PluginAdditionalInfo",■■■■■■■true,■true ); //TODO clean up after Gamescom
■■RegisterPlugin( "PluginConfigEmotesProfile",■■■■■true, ■true );
■■RegisterPlugin( "PluginPresenceNotifier",■■■■■■true,■false );
■■RegisterPlugin( "PluginAdminLog",■■■■■■■■false, ■true );
```

```
// ---------------------------------------------
// ------ PluginMessageManager.c - START --------------------
// ---------------------------------------------


/*! Class PluginMessageManager provides some basic Message Distribution mechanics, if you get instanc
you can use its methods to broadcast messages.
*/
class PluginMessageManager extends PluginBase
{
■int channelsUsed = 0;
■ref array<ref MessageReceiverBase>■channelList[NUM_OF_CHANNELS];
■
■void PluginMessageManager()
■{
■■for (int i = 0; i < NUM_OF_CHANNELS; i++)
■■{
■■■channelList[i] = new array<ref MessageReceiverBase>;
■■}
■}

■//! Broadcasts an empty message on a channel provided in the 'index' parameter
■void Broadcast(int index)
■{
■■//Debug.Log("Broadcasting message on a channel: "+ ToString(index), "Messaging System");
■■array<ref MessageReceiverBase> x = channelList[index];
■■//int test = channelList[index].Count();
■■for(int i = 0; i < x.Count(); i++)
■■{
■■■MessageReceiverBase mrb = x.Get(i);
■■■
■■■if(■mrb != NULL )
■■■{
■■■■//string s = "Broadcasting message to: "+ToString(mrb);
■■■■//Log(s, LogTemplates.TEMPLATE_BROADCAST);
■■■■mrb.OnReceive(index);
■■■}
■■}
■}

■//! Broadcasts a message on a channel provided in the 'index' parameter, passes the Int variable
■void BroadcastInt(int index, int value)
■{
■■//Debug.Log("Broadcasting message on a channel: "+ ToString(index), "Messaging System");
■■array<ref MessageReceiverBase> x = channelList[index];
■■//int test = channelList[index].Count();
■■for(int i = 0; i < x.Count(); i++)
■■{
■■■MessageReceiverBase mrb = x.Get(i);
■■■
■■■if(■mrb )
■■■{
■■■■//string s = "Broadcasting message to: "+ToString(mrb);
■■■■//Log(s, LogTemplates.TEMPLATE_BROADCAST);
■■■■mrb.OnReceiveInt(index, value);
■■■}
■■■
■■}
■}

■void BroadcastFloat(int index, float value)
■{
■■//Debug.Log("Broadcasting message on a channel: "+ ToString(index), "Messaging System");
```

```
// --------------------------------------------
// ------ PluginMissionConfig.c - START --------------------
// --------------------------------------------


class PluginMissionConfig extends PluginConfigHandler
{■
■protected const string FILE_NAME■■■■= "scene_editor.cfg";
■protected const string SCENE_EDITOR_SCENE■■= "scene_load";
■
■//========================================
■// GetInstance
■//========================================
■static PluginMissionConfig GetInstance()
■{
■■return PluginMissionConfig.Cast( GetPlugin(PluginMissionConfig) );
■}
■
■//========================================
■// OnInit
■//========================================
■override void OnInit()
■{
■■super.OnInit();
■■
■■CfgParamString cfg_scene_name = CfgParamString.Cast( GetParamByName( SCENE_EDITOR_SCE
■■
■■if ( cfg_scene_name.GetValue() == STRING_EMPTY )
■■{
■■■cfg_scene_name.SetValue( PluginSceneManager.SCENE_DEFAULT_NAME );
■■■SaveConfigToFile();
■■}
■}
■
■//========================================
■// GetFileName
■//========================================
■override string GetFileName()
■{
■■return g_Game.GetMissionFolderPath() +"\\"+ FILE_NAME;
■}

■//========================================
■// GetSceneEditorName
■//========================================
■string GetSceneEditorName()
■{
■■CfgParamString cfg_scene_name = CfgParamString.Cast( GetParamByName( SCENE_EDITOR_SCE
■■
■■if ( cfg_scene_name.GetValue() == STRING_EMPTY )
■■{
■■■cfg_scene_name.SetValue( PluginSceneManager.SCENE_DEFAULT_NAME );
■■■SaveConfigToFile();
■■}
■■
■■return cfg_scene_name.GetValue();
■}

■//========================================
■// SetSceneEditorName
■//========================================
■void SetSceneEditorName( string value )
■{
```

```
// ---------------------------------------------
// ------ PluginNutritionDumper.c - START --------------------
// ---------------------------------------------



class PluginNutritionDumper extends PluginBase
{
■/*
■ref TStringArray m_AllPaths = new TStringArray;
■ref TStringArray m_AllLines = new TStringArray;
■ref map<string, int> m_ParamPool = new map<string, int>;
■
■string config_path;
■string child_name;
■int scope;
■string path;
■PlayerBase m_Player;
■override void OnInit()
■{
■■m_AllPaths.Insert("CfgVehicles");
■■m_AllPaths.Insert("cfgLiquidDefinitions");
■■m_Player = PlayerBase.Cast(GetGame().GetPlayer());
■}
■
■void CheckInit()
■{
■■m_AllLines.Clear();
■■string line = "Classname(stage),energy,water,toxicity,fullnessIndex,nutritionalIndex";
■■m_AllLines.Insert(line);
■■
■■for(int i = 0; i < m_AllPaths.Count(); i++)
■■{
■■■config_path = m_AllPaths.Get(i);
■■■int children_count = GetGame().ConfigGetChildrenCount(config_path);
■■■
■■■for(int x = 0; x < children_count; x++)
■■■{
■■■■GetGame().ConfigGetChildName(config_path, x, child_name);
■■■■path = config_path + " " + child_name;
■■■■scope = GetGame().ConfigGetInt( config_path + " " + child_name + " scope" );
■■■■bool should_check = 1;
■■■■if( config_path == "CfgVehicles" && scope == 0)
■■■■{
■■■■■should_check = 0;
■■■■}
■■■■
■■■■if ( should_check )
■■■■{
■■■■■bool has_nutrition = GetGame().ConfigIsExisting(path + " Nutrition");
■■■■■bool has_stages = GetGame().ConfigIsExisting(path + " Food");
■■■■■if(has_nutrition || has_stages)
■■■■■{
■■■■■■EntityAI item = PlayerBase.Cast(GetGame().GetPlayer()).SpawnEntityOnGroundOnCursorDir(c
■■■■■■Edible_Base edible = Edible_Base.Cast(item);
■■■■■■if(edible)
■■■■■■{
■■■■■■■//Print("spawning " + child_name);
■■■■■■■line = "";
■■■■■■■NutritionalProfile profile;
■■■■■■■if(!has_stages)
■■■■■■■{
■■■■■■■■profile = edible.GetNutritionalProfile();
```

```
// ---------------------------------------------
// ------ PluginObjectsInteractionManager.c - START --------------------
// ---------------------------------------------


class PluginObjectsInteractionManager extends PluginBase
{
■private ref ■■array<Object> m_LockedObjects;
■private ref ■■array<float> m_LockedObjectsDecay;
■private const float■■TIME_TO_FORCED_UNLOCK = 60;
■private const float ■TICK_RATE = 10;
■private ref Timer■m_DecayTimer;
■■
■void PluginObjectsInteractionManager()
■{
■■m_LockedObjects = new array<Object>;
■■m_LockedObjectsDecay = new array<float>;
■■//TIMERDEPRECATED - timer for decaying objects
■■m_DecayTimer = new Timer();
■■m_DecayTimer.Run(TICK_RATE, this, "Decay", NULL,true);
■}

■bool IsFree(Object target)
■{
■■if ( target && m_LockedObjects.Count() > 0 )
■■{
■■■for ( int i = 0; i < m_LockedObjects.Count(); i++ )
■■■{■
■■■■if ( m_LockedObjects.Get(i) == target )
■■■■{
■■■■■return false;
■■■■}
■■■}
■■}
■■return true;
■}

■void Lock(Object target)
■{
■■if ( target && !IsFree(target) )
■■{
■■■m_LockedObjects.Insert(target);
■■■m_LockedObjectsDecay.Insert(0);
■■}
■}

■void Release(Object target)
■{
■■if ( target && m_LockedObjects.Count() > 0 )
■■{
■■■for ( int i = 0; i < m_LockedObjects.Count(); i++ )
■■■{■
■■■■if ( m_LockedObjects.Get(i) == target )
■■■■{
■■■■■m_LockedObjects.Remove(i);
■■■■■m_LockedObjectsDecay.Remove(i);
■■■■■break;
■■■■}
■■■}
■■}
■}
■
■//FAILSAFE - checks periodically locked objects and releases them automaticaly if given time has passe
```

```
// ---------------------------------------------
// ------ PluginPermanentCrossHair.c - START --------------------
// ---------------------------------------------


class PluginPermanentCrossHair extends PluginBase
{
■private Hud m_Hud;

■void SwitchPermanentCrossHair(bool state)
■{
■■if (!m_Hud)
■■■m_Hud = GetGame().GetMission().GetHud();
■■m_Hud.SetPermanentCrossHair(state);
■}
}


// ---------------------------------------------
// ------ PluginPermanentCrossHair.c - END ----------------------
// ---------------------------------------------


// ---------------------------------------------
// ------ PluginPlayerStatus.c - START --------------------
// ---------------------------------------------


class PluginPlayerStatus extends PluginBase
{
■ref multiMap<int, string> ■m_NotifiersLabel;
■ref multiMap<int, int>■■m_NotifiersIndexColor;

■private ref multiMap<int, string>■m_NotifiersIcons;

■void PluginPlayerStatus()
■{
■■m_NotifiersLabel = new multiMap<int, string>; // [key] label
■■m_NotifiersIndexColor = new multiMap<int, int>; // [key] index, color

■■m_NotifiersIcons = new multiMap<int, string>; // [key] iconName
■■m_NotifiersIcons.Insert( NTFKEY_HUNGRY, "iconHunger" );
■■m_NotifiersIcons.Insert( NTFKEY_THIRSTY, "iconThirsty" );
■■m_NotifiersIcons.Insert( NTFKEY_SICK, "iconHealth" );
■■m_NotifiersIcons.Insert( NTFKEY_BACTERIA, "iconBacteria" );
■■m_NotifiersIcons.Insert( NTFKEY_BLEEDISH, "iconBlood" );
■■m_NotifiersIcons.Insert( NTFKEY_FEVERISH, "iconTemperature" );
■■m_NotifiersIcons.Insert( NTFKEY_FRACTURE, "iconFracture" );
■}
■
■void SetNotifier( int key, int index = 9, string label = "", int color =  0xFFFFFFFF )
■{
■■if ( key )
■■{
■■■if ( m_NotifiersLabel.HasKey( key ) )
■■■{
■■■■m_NotifiersLabel.Remove( key );
■■■■m_NotifiersIndexColor.Remove( key );
■■■}

■■■if ( label != "" )
■■■{
■■■■m_NotifiersLabel.Insert( key, label );
```

```
// ----------------------------------------------
// ------ PluginPresenceNotifier.c - START --------------------
// ----------------------------------------------


enum EPresenceNotifierNoiseEventType
{
■LAND_LIGHT,
■LAND_HEAVY,
}

class PresenceNotifierNoiseEvent
{
■protected float m_TimerLength;
■protected int m_Value;
■
■void PresenceNotifierNoiseEvent(float pValue, float pLength)
■{
■■m_Value = pValue;
■■m_TimerLength = pLength;
■}

■float GetTimerLength()
■{
■■return m_TimerLength;
■}
■
■int GetValue()
■{
■■return m_Value;
■}
}

class PresenceNotifierNoiseEvents
{
■protected int m_Value;

■protected static ref map<EPresenceNotifierNoiseEventType, ref PresenceNotifierNoiseEvent> ■m_Prese
■protected ref Timer■m_CooldownTimer;
■
■void PresenceNotifierNoiseEvents()
■{
■■m_Value = 0;
■■m_CooldownTimer = new Timer(CALL_CATEGORY_SYSTEM);
■■m_PresenceNotifierNotifierEvents = new ref map<EPresenceNotifierNoiseEventType, ref PresenceNot
■}

■void RegisterEvent(EPresenceNotifierNoiseEventType pEventType, int pValue, float pLength)
■{
■■PresenceNotifierNoiseEvent pnne = new PresenceNotifierNoiseEvent(pValue, pLength);
■■m_PresenceNotifierNotifierEvents.Insert(pEventType, pnne);
■}
■
■void ProcessEvent(EPresenceNotifierNoiseEventType pEventType)
■{
■■PresenceNotifierNoiseEvent pnne;
■■
■■pnne = m_PresenceNotifierNotifierEvents.Get(pEventType);
■■
■■if (m_CooldownTimer.IsRunning())
■■{
■■■m_CooldownTimer.Stop();
■■}
```

```
// ---------------------------------------------
// ------ PluginRecipesManager.c - START --------------------
// ---------------------------------------------


enum ERecipeSanityCheck
{
■IS_IN_PLAYER_INVENTORY = 1,
■NOT_OWNED_BY_ANOTHER_LIVE_PLAYER = 2,
■CLOSE_ENOUGH = 4,
}

const float ACCEPTABLE_DISTANCE = 5;

const int SANITY_CHECK_ACCEPTABLE_RESULT = ERecipeSanityCheck.NOT_OWNED_BY_ANOTHE

class PluginRecipesManager extends PluginRecipesManagerBase
{
■static ref map<string,ref CacheObject > m_RecipeCache = new map<string,ref CacheObject >;
■static ref map<typename, bool> m_RecipesInitializedItem = new ref map<typename, bool>;
■
■
■ref Timer m_TestTimer;
■const int MAX_NUMBER_OF_RECIPES = GetMaxNumberOfRecipes();
■const int MAX_CONCURENT_RECIPES = 128;
■const int MAX_INGREDIENTS = 5;
■
■int m_RegRecipeIndex;
■int m_ResolvedRecipes[MAX_CONCURENT_RECIPES];
■
■bool m_EnableDebugCrafting = false;
■ItemBase m_Ingredients[MAX_INGREDIENTS];
■int m_IngredientBitMask[MAX_INGREDIENTS];
■int m_IngredientBitMaskSize[MAX_INGREDIENTS];
■
■int m_BitsResults[MAX_INGREDIENTS];
■
■ItemBase m_ingredient1[MAX_CONCURENT_RECIPES];
■ItemBase m_ingredient2[MAX_CONCURENT_RECIPES];
■ItemBase m_ingredient3[MAX_CONCURENT_RECIPES];
■
■ItemBase m_sortedIngredients[MAX_NUMBER_OF_INGREDIENTS];
■
■ref array<int> m_RecipesMatched = new array<int>;
■ref array<string> m_CachedItems = new array<string>;

■ref array<ref RecipeBase> m_RecipeList = new array<ref RecipeBase>;//all recipes
■static ref map<string, int> m_RecipeNamesList = new map<string, int>;//all recipes
■
■ref Timer myTimer1;
■
■static int GetMaxNumberOfRecipes()
■{
■■return 2048;
■}
■
■void PluginRecipesManager()
■{

■■CreateAllRecipes();
■■GenerateRecipeCache();
■■
■■myTimer1 = new Timer();
```

```
// ---------------------------------------------
// ------ PluginRecipesManagerBase.c - START --------------------
// ---------------------------------------------


enum Ingredient
{
■FIRST = 1,
■SECOND = 2,
■BOTH = 3;
}

class PluginRecipesManagerBase extends PluginBase
{■
■protected void RegisterRecipe(RecipeBase recipe);
■protected void UnregisterRecipe(string clasname);
■
■//! Please do not delete commented recipes, they are usually commented out for a reason
■void RegisterRecipies()
■{
■■RegisterRecipe(new CraftTorch);
■■RegisterRecipe(new CraftFireplace);
■■RegisterRecipe(new CraftLongTorch);
■■RegisterRecipe(new SharpenBroom);
■■RegisterRecipe(new SharpenLongStick);
■■RegisterRecipe(new PokeHolesBarrel);
■■RegisterRecipe(new CraftBaseBallBatNailed);
■■RegisterRecipe(new CraftBaseBallBatBarbed);
■■RegisterRecipe(new CraftGhillieHood);
■■RegisterRecipe(new DeCraftGhillieHood);
■■RegisterRecipe(new CraftGhillieAttachment);
■■RegisterRecipe(new DeCraftGhillieAttachment);
■■RegisterRecipe(new CraftGhillieBushrag);
■■RegisterRecipe(new DeCraftGhillieBushrag);
■■RegisterRecipe(new CraftGhillieTop);
■■RegisterRecipe(new DeCraftGhillieTop);
■■RegisterRecipe(new CraftGhillieSuit);
■■RegisterRecipe(new DeCraftGhillieSuit);
■■RegisterRecipe(new CraftStoneKnife);
■■RegisterRecipe(new CraftBait);
■■RegisterRecipe(new CraftRagRope);
■■RegisterRecipe(new CraftSuppressor);
■■RegisterRecipe(new CleanWeapon);
■■RegisterRecipe(new RepairWithTape);
■■RegisterRecipe(new RepairWithRags);
■■RegisterRecipe(new RepairEyePatch);
■■RegisterRecipe(new CraftBoneKnife);
■■//RegisterRecipe(new CraftArrow);
■■//RegisterRecipe(new CraftArrowBone);
■■RegisterRecipe(new CraftBoneHook);
■■RegisterRecipe(new CraftBurlapStrips);
■■RegisterRecipe(new CraftLeatherCourierBag);
■■RegisterRecipe(new CraftCourierBag);
■■RegisterRecipe(new CraftImprovisedBag);
■■RegisterRecipe(new CraftImprovisedLeatherBag);
■■RegisterRecipe(new CraftImprovisedExplosive);
■■RegisterRecipe(new CraftLeatherSack);
■■RegisterRecipe(new DeCraftLeatherCourierBag);
■■RegisterRecipe(new DeCraftCourierBag);
■■RegisterRecipe(new DeCraftImprovisedBag);
■■RegisterRecipe(new DeCraftImprovisedLeatherBag);
■■RegisterRecipe(new DeCraftLeatherSack);
■■RegisterRecipe(new SplitLongWoodenStick);
```

```
// ---------------------------------------------
// ------ PluginRemotePlayerDebugClient.c - START --------------------
// ---------------------------------------------


enum eRemoteDebugType
{
■NONE,
■DAMAGE_ONLY,
■ALL,
}

class PluginRemotePlayerDebugClient extends PluginBase
{
■const int MAX_SIMULTANIOUS_PLAYERS = 10;
■ref array<ref RemotePlayerStatDebug> m_PlayerDebugStats = new array<ref RemotePlayerStatDebug>
■ref map<PlayerBase, ref RemotePlayerDamageDebug> m_PlayerDebugDamage = new map<PlayerBas
■
■ref Widget ■m_RootWidget[MAX_SIMULTANIOUS_PLAYERS];
■ref Widget ■m_RootWidgetDamage[MAX_SIMULTANIOUS_PLAYERS];
■ref TextListboxWidget m_StatListWidgets[MAX_SIMULTANIOUS_PLAYERS];
■ref TextListboxWidget m_DamageListWidgets[MAX_SIMULTANIOUS_PLAYERS];
■ref TextWidget m_DistanceWidget[MAX_SIMULTANIOUS_PLAYERS];
■eRemoteDebugType m_DebugType;
■
■override void OnInit()
■{
■■#ifndef NO_GUI
■■InitWidgets();
■■#endif
■}
■
■override void OnUpdate(float delta_time)
■{
■■#ifndef NO_GUI
■■if ( m_DebugType !=0 ) UpdateWidgetsStats();
■■#endif
■}
■
■void InitWidgets()
■{
■■for (int i = 0; i < MAX_SIMULTANIOUS_PLAYERS; ++i)
■■{
■■■m_RootWidget[i] = GetGame().GetWorkspace().CreateWidgets("gui/layouts/debug/day_z_debug_rem
■■■m_RootWidgetDamage[i] = GetGame().GetWorkspace().CreateWidgets("gui/layouts/debug/day_z_de
■■■m_DamageListWidgets[i] = TextListboxWidget.Cast(m_RootWidgetDamage[i].FindAnyWidget("TextL
■■■m_StatListWidgets[i] = TextListboxWidget.Cast(m_RootWidget[i].FindAnyWidget("TextListboxWidget
■■■m_DistanceWidget[i] = TextWidget.Cast(m_RootWidget[i].FindAnyWidget("TextWidget0"));
■■}
■}■■
■
■void EnableWidgets(bool enable)
■{
■■for (int i = 0; i < MAX_SIMULTANIOUS_PLAYERS; ++i)
■■{
■■■m_RootWidget[i].Show(enable);
■■■m_RootWidgetDamage[i].Show(enable);
■■}
■}
■
■void UpdateWidgetsStats()
■{
■■int i = 0;
```

```
// ---------------------------------------------
// ------ PluginRemotePlayerDebugServer.c - START --------------------
// ---------------------------------------------


class PluginRemotePlayerDebugServer extends PluginBase
{■
■ref set<PlayerBase> m_ClientList = new set<PlayerBase>;
■
■ref array<ref RemotePlayerStatDebug> m_PlayerDebugStats = new array<ref RemotePlayerStatDebug>
■ref map<PlayerBase, ref RemotePlayerDamageDebug> m_PlayerDebugDamage = new map<PlayerBas
■float m_AccuTime;
■const int INTERVAL = 1;
■
■eRemoteDebugType m_DebugType;
■bool m_Watching;
■
■override void OnUpdate(float delta_time)
■{
■■#ifdef SERVER
■■if ( m_ClientList.Count() != 0 )
■■{
■■■m_AccuTime += delta_time;
■■■
■■■if ( m_AccuTime > INTERVAL )
■■■{
■■■■m_AccuTime = 0;
■■■■SendDebug();
■■■}
■■}
■■#endif
■}
■
■void SetWatching(bool enable)
■{
■■m_Watching = enable;
■}
■
■bool GetWatching()
■{
■■return m_Watching;
■}
■
■void GatherPlayerInfo()
■{
■■array<Man> players = new array<Man>;
■■GetGame().GetPlayers(players);
■■m_PlayerDebugStats.Clear();
■■
■■foreach (Man playerMan : players)
■■{
■■■PlayerBase player = PlayerBase.Cast(playerMan);
■■■RemotePlayerStatDebug rpd = new RemotePlayerStatDebug(player);
■■■m_PlayerDebugStats.Insert(rpd);
■■}■■
■}
■
■void SendDebug()
■{
#ifdef DIAG_DEVELOPER
■■GatherPlayerInfo();
■■array<ref RemotePlayerDamageDebug> player_damage = new array<ref RemotePlayerDamageDebug
■■
```

```
// ---------------------------------------------
// ------ PluginRepairing.c - START --------------------
// ---------------------------------------------


class PluginRepairing extends PluginBase
{
■bool Repair(PlayerBase player, ItemBase repair_kit, Object item, float specialty_weight, string damage_z
■{
■■switch (item.GetHealthLevel(damage_zone))
■■{
■■■case GameConstants.STATE_PRISTINE:
■■■■break;
■■■case GameConstants.STATE_RUINED:
■■■■#ifdef DEVELOPER
■■■■Debug.Log("repairing from GameConstants.STATE_RUINED");
■■■■#endif
■■■■CalculateHealth( player, repair_kit, item, specialty_weight, damage_zone, use_kit_qty );
■■■■break;
■■■case GameConstants.STATE_WORN:
■■■■if (CanRepairToPristine(player) || CanBeRepairedToPristine(item))
■■■■{
■■■■■CalculateHealth( player, repair_kit, item, specialty_weight,/* GameConstants.DAMAGE_PRISTIN
■■■■}
■■■■break;
■■■default:
■■■■CalculateHealth( player, repair_kit, item, specialty_weight, damage_zone, use_kit_qty );
■■■■break;
■■}

■■return true;
■}

■void CalculateHealth( PlayerBase player, ItemBase kit, Object item, float specialty_weight, string damag
■{
■■EntityAI entity;
■■Class.CastTo(entity,item);
■■
■■if (entity != null)
■■{
■■■entity.SetAllowDamage(true);
■■}
■■
■■bool kit_has_quantity = kit.HasQuantity();
■■int health_levels_count = item.GetNumberOfHealthLevels(damage_zone);
■■float cur_kit_quantity = kit.GetQuantity();
■■float kit_repair_cost_per_level = GetKitRepairCost( kit, item );
■■float kit_repair_cost_adjusted; //used with specialty_weight, disconnected
■■float new_quantity;
■■
■■int target_level = Math.Clamp(item.GetHealthLevel(damage_zone) - 1, 0, health_levels_count - 1);
■■float health_coef;
■■if ( !CanRepairToPristine( player ) && !CanBeRepairedToPristine( item ) )
■■{
■■■target_level = Math.Clamp(target_level, GameConstants.STATE_WORN, health_levels_count - 1);
■■}
■■health_coef = item.GetHealthLevelValue(target_level,damage_zone);
■■
■■//handles kit depletion; TODO: move to separate method.
■■if ( cur_kit_quantity > kit_repair_cost_per_level )
■■{
■■■kit_repair_cost_adjusted = kit_repair_cost_per_level; //TODO: removed speciality weight for now, it s
■■■//kit_repair_cost_adjusted = player.GetSoftSkillsManager().SubtractSpecialtyBonus( kit_repair_cost_
// ---------------------------------------------
// ------ PluginSceneManager.c - START --------------------
// ---------------------------------------------


class PluginSceneManager extends PluginBase
{
■static string■■SCENE_DEFAULT_NAME■■= "default";
■static string■■SCENE_SUFIX■■■■= "scene";
■static PlayerBase■PLAYER;
■static PluginSceneManager instance;
■
■bool■■■m_RulerActivated;
■ref array<vector>■m_RulerPoints;
■ref array<Shape>■m_RulerLines;
■ref array<Shape>■m_RulerSpheres;
■
■static const int SCENE_EDITOR_CMD_REFRESH■= 0;
■static const int SCENE_EDITOR_CMD_POPUP■■= 1;
■static const int SCENE_EDITOR_CMD_SAVE■■= 2;
■
■//----------------------------------------------------------------------------------
■// >> Public scope

■// System Events
■void PluginSceneManager()
■{
■■if ( instance == NULL )
■■{
■■■instance = this;
■■}
■}
■
■// System Events
■void ~PluginSceneManager()
■{
■■if ( instance == NULL )
■■{
■■■instance = this;
■■}
■}
■
■static PluginSceneManager GetInstance()
■{
■■return PluginSceneManager.Cast( GetPlugin( PluginSceneManager ) );
■}

■//==========================================
■// OnInit (System Event)
■//==========================================
■override void OnInit()
■{
■■m_ModuleDeveloper■■■= PluginDeveloper.Cast( GetPlugin(PluginDeveloper) );
■■m_ModuleConfigScene■■■= PluginConfigScene.Cast( GetPlugin(PluginConfigScene) );
■■m_ModuleConfigDebugProfile■= PluginConfigDebugProfile.Cast( GetPlugin(PluginConfigDebugProfile
■■
■■//Ruler
■■m_RulerActivated = false;
■■m_RulerPoints■= new array<vector>;
■■m_RulerLines■= new array<Shape>;
■■m_RulerSpheres■= new array<Shape>;
■}
```

```
// ---------------------------------------------
// ------ PluginSoundDebug.c - START --------------------
// ---------------------------------------------


class PluginSoundDebug extends PluginBase
{
■override void OnInit()
■{
■■m_TickTimer = new Timer();
■■m_TickTimer.Run(0.1, this, "OnGUITimer", NULL, true);
■}
■
■override void OnUpdate(float delta_time)
■{
■■;
■}
■
■override void OnDestroy()
■{
■■m_TickTimer = NULL;
■}
■
■void Show()
■{
■■m_TickTimer = new Timer();
■■m_TickTimer.Run(0.1, this, "OnGUITimer", NULL, true);
■}
■
■void Hide()
■{
■■m_TickTimer = NULL;
■}
■
■void OnGUITimer()
■{
■■DbgUI.BeginCleanupScope();
■■DbgUI.Begin("Sound debug", 10, 10);
■■
■■DbgUI.PushID_Str("SoundParams");
■■DbgUI.Text("SoundParams: ");
■■DbgUI.SameLine();
■■string soundsetName = "BearGrowl_SoundSet";
■■DbgUI.InputText("", soundsetName, 200);
■■DbgUI.PopID();
■■
■■DbgUI.PushID_Str("Offset");
■■DbgUI.Text("Offset pos: ");
■■vector posOffset = "0 0 0";
■■float posVal;
■■
■■DbgUI.SameLine();
■■DbgUI.PushID_Int(1);
■■■DbgUI.InputFloat("", posVal, 80);
■■DbgUI.PopID();
■■posOffset[0] = posVal;
■■
■■DbgUI.SameLine();
■■DbgUI.PushID_Int(2);
■■■DbgUI.InputFloat("", posVal, 80);
■■DbgUI.PopID();
■■posOffset[1] = posVal;
■■
```

```
// ---------------------------------------------
// ------ PluginTransmissionAgents.c - START --------------------
// ---------------------------------------------


enum InjectTypes
{
    PLAYER_TO_ITEM,
■ITEM_TO_PLAYER,
■PLAYER_AIR_PLAYER,
};

class PluginTransmissionAgents extends PluginBase
{
■static ref map<int, ref AgentBase> m_AgentList =  new map<int, ref AgentBase>;
■ref map<int, string> m_SimpleAgentList = new map<int, string>;
■bool m_IsConstructed = false;
■
■void PluginTransmissionAgents()
■{
■■//add new agents here
■■RegisterAgent(new InfluenzaAgent);
■■RegisterAgent(new CholeraAgent);
■■RegisterAgent(new SalmonellaAgent);
■■RegisterAgent(new BrainAgent);
■■RegisterAgent(new FoodPoisonAgent);
■■RegisterAgent(new ChemicalAgent);
■■RegisterAgent(new WoundAgent);
■■RegisterAgent(new NerveAgent);
■}
■
■void RegisterAgent(AgentBase agent)
■{
■■m_AgentList.Insert(agent.GetAgentType(), agent);
■}

■void ConstructSimpleAgentList()
■{
■■string agent_name;
■■int agent_type;
■■
■■for(int i = 0; i < m_AgentList.Count();i++)
■■{
■■■AgentBase agent = m_AgentList.GetElement(i);
■■■agent_name = agent.GetName();
■■■agent_type = agent.GetAgentType();
■■■m_SimpleAgentList.Insert(agent_type, agent_name);
■■}
■}

■map<int, ref AgentBase> GetAgentList()
■{
■■return■m_AgentList;
■}
■// this is a list which is easy to work with for displaying all agents and interacting with them, it doesn't serv
■map<int, string> GetSimpleAgentList()
■{
■■if( !m_IsConstructed )
■■{
■■■ConstructSimpleAgentList();
■■■m_IsConstructed■= true;
■■}
■■return m_SimpleAgentList;
```

```
// ---------------------------------------------
// ------ PluginUniversalTemperatureSourceClient.c - START --------------------
// ---------------------------------------------


class PluginUniversalTemperatureSourceClient extends PluginBase
{
■const int MAX_SIMULTANEOUS_UTS = 10;

■protected float m_UTSAverageTemperature;

■protected ref array<ref UTemperatureSourceDebug> m_UTemperatureSourceDebugs;
■
■protected ref Widget m_RootWidget[MAX_SIMULTANEOUS_UTS];
■protected TextListboxWidget m_StatListWidgets[MAX_SIMULTANEOUS_UTS];
■protected TextWidget m_HeaderWidget[MAX_SIMULTANEOUS_UTS];
■
■protected PlayerBase m_Player;
■
■void PluginUniversalTemperatureSourceClient()
■{
■■m_UTemperatureSourceDebugs = new array<ref UTemperatureSourceDebug>();
■}
■
■override void OnInit()
■{
■■#ifndef NO_GUI
■■InitWidgets();
■■#endif
■}
■
■override void OnUpdate(float delta_time)
■{
■■#ifndef NO_GUI
■■if (!m_Player)
■■{
■■■return;
■■}

■■UpdateStatWidgets();
■■DrawDebugs();
■■#endif
■}
■
■void InitWidgets()
■{
■■for (int i = 0; i < MAX_SIMULTANEOUS_UTS; i++)
■■{
■■■m_RootWidget[i] = GetGame().GetWorkspace().CreateWidgets("gui/layouts/debug/day_z_debug_rem
■■■m_StatListWidgets[i] = TextListboxWidget.Cast(m_RootWidget[i].FindAnyWidget("TextListboxWidget
■■■m_HeaderWidget[i] = TextWidget.Cast(m_RootWidget[i].FindAnyWidget("TextWidget0"));
■■}
■}
■
■void DrawDebugs()
■{
■■foreach (UTemperatureSourceDebug utsd : m_UTemperatureSourceDebugs)
■■{
■■■float fullRange = utsd.GetValue(1).ToFloat();
■■■float maxRange ■= utsd.GetValue(2).ToFloat();
■■■float temp■■= utsd.GetValue(3).ToFloat();
■■■vector sphPos ■= utsd.GetValue(0).ToVector();
```

```
// ---------------------------------------------
// ------ PluginUniversalTemperatureSourceServer.c - START --------------------
// ---------------------------------------------


class PluginUniversalTemperatureSourceServer extends PluginBase
{
■const int INTERVAL■■■= 1;■■//! [s]
■const float LOOKUP_RADIUS■= 20;■■//! [m]
■■
■protected float m_AccuTime;
■protected ref set<PlayerBase> m_ClientList;

■protected ref array<ref UTemperatureSourceDebug> m_UTemperatureSourceDebugs;

■void PluginUniversalTemperatureSourceServer()
■{
■■m_ClientList ■■■■= new set<PlayerBase>();
■■m_UTemperatureSourceDebugs■= new array<ref UTemperatureSourceDebug>();
■}
■
■override void OnUpdate(float delta_time)
■{
■■if (m_ClientList.Count() > 0)
■■{
■■■m_AccuTime += delta_time;
■■■
■■■if (m_AccuTime > INTERVAL)
■■■{
■■■■m_AccuTime = 0;
■■■■SendDebug();
■■■}
■■}
■}
■
■void GatherTemperatureSources(PlayerBase player)
■{
■■m_UTemperatureSourceDebugs.Clear();
■■
■■if (!player)
■■{
■■■return;
■■}

■■vector playerPos■■■■= player.GetPosition();
■■array<Object> nearestObjects■= new array<Object>;

■■GetGame().GetObjectsAtPosition(playerPos, LOOKUP_RADIUS, nearestObjects, null);

■■UTemperatureSource uts;
■■UTemperatureSourceDebug utsd;

■■for (int i = 0; i < nearestObjects.Count(); i++)
■■{
■■■EntityAI ent = EntityAI.Cast(nearestObjects.Get(i));
■■■if (ent && ent.IsUniversalTemperatureSource() && ent != player)
■■■{
■■■■uts = ent.GetUniversalTemperatureSource();
■■■■utsd = new UTemperatureSourceDebug();
■■■■utsd.AddHeader(ent.GetType());
■■■■utsd.Add("position", uts.GetPosition().ToString(false));
■■■■utsd.Add("fullrange", uts.GetFullRange().ToString());
■■■■utsd.Add("maxrange", uts.GetMaxRange().ToString());
```

```
// ---------------------------------------------
// ------ PluginVariables.c - START --------------------
// ---------------------------------------------


class PluginVariables extends PluginBase
{
■void PluginVariables()
■{
■■m_Id = 0;
■■m_Variables = new map<int, string>;
■■//■■■■("variable name")
■■RegisterVariable("varNote");
■■RegisterVariable("varColor");
■■
■}

■void ~PluginVariables()
■{
■}
■
■void RegisterVariable(string name)
■{
■■m_Id++;
■■m_Variables.Set(m_Id,name);//REWORK.V maybe have 2 maps, one with key
■■
■}

■int m_Id;
■ref map<int, string> m_Variables;

■string GetName(int id)
■{
■■return m_Variables.Get(id);
■}

■int GetID(string name)
■{
■■return m_Variables.GetKeyByValue(name);
■}
}



// ---------------------------------------------
// ------ PluginVariables.c - END ----------------------
// ---------------------------------------------


// ---------------------------------------------
// ------ Plum.c - START --------------------
// ---------------------------------------------


class Plum : Edible_Base
{
■override bool CanBeCooked()
■{
■■return true;
■}■■
■
■override bool CanBeCookedOnStick()
■{
```

```
// ---------------------------------------------
// ------ PM73Rak.c - START --------------------
// ---------------------------------------------


class PM73Rak_Base : Rifle_Base
{
};


// ---------------------------------------------
// ------ PM73Rak.c - END ----------------------
// ---------------------------------------------


// ---------------------------------------------
// ------ PMM.c - START --------------------
// ---------------------------------------------


class PMM : Pistol_Base
{
};


// ---------------------------------------------
// ------ PMM.c - END ----------------------
// ---------------------------------------------


// ---------------------------------------------
// ------ PMTCreationAndCleanup.c - START --------------------
// ---------------------------------------------


class OwnershipTestDummyClass
{
■// Simply exists
}

class PMTCreationAndCleanup : PMTF
{
■int m_EventTestManagerID;
■bool m_bTestEventsPassed = false;
■
■int m_OwnershipTestManagerID;
■
■//---------------------------------------------------------------------------
■// Ctor - Decides the tests to run
■//---------------------------------------------------------------------------
■void PMTCreationAndCleanup()
■{
■■//AddInitTest("TestInvalidSize");
■■AddInitTest("TestCCSB");
■■AddInitTest("TestEvents");
■■AddInitTest("TestOwnership");
■}
■
■//---------------------------------------------------------------------------
■// Tests
■//---------------------------------------------------------------------------
■// Test invalid size VMEs
■TFResult TestInvalidSize()
```

```
// ---------------------------------------------
// ------ PMTF.c - START --------------------
// ---------------------------------------------


class PMTF : TestFramework
{
■private static int PM_CREATED = 0;
■private ref map<int, ref ParticleManager> m_Managers = new map<int, ref ParticleManager>();
■
■//-----------------------------------------------------------------------
■// Manager management
■//-----------------------------------------------------------------------
■int InsertManager(ParticleManager pm)
■{
■■Assert(m_Managers.Insert(PM_CREATED, pm));
■■++PM_CREATED;
■■
■■return PM_CREATED - 1;
■}
■
■bool GetManager(int id, out ParticleManager pm)
■{
■■return m_Managers.Find(id, pm);
■}
■
■//-----------------------------------------------------------------------
■// Prints
■//-----------------------------------------------------------------------
■protected void PrintPMStats(notnull ParticleManager pm)
■{
■■Debug.ParticleLog(string.Format(
■■■"Poolsize: %1 | Allocated: %2 | Virtual: %3 | Playing: %4", pm.GetPoolSize(), pm.GetAllocatedCount(
■■this, "PrintPMStats", pm);
■}
■
■protected void PrintActiveStats()
■{
■■Debug.ParticleLog(string.Format(
■■■"Active ParticleManagers: %1 | Active ParticleSources: %2", ParticleManager.GetStaticActiveCount()
■■this, "PrintActiveStats");
■}
■
■//-----------------------------------------------------------------------
■// Helpers
■//-----------------------------------------------------------------------
■protected ParticleManager CreatePMFixedBlocking(int size)
■{
■■return new ParticleManager(new ParticleManagerSettings(size, ParticleManagerSettingsFlags.BLOCK
■}
}


// ---------------------------------------------
// ------ PMTF.c - END ----------------------
// ---------------------------------------------
// ---------------------------------------------
// ------ PMTPlayback.c - START --------------------
// ---------------------------------------------


class PMTPlayback : PMTF
{
■ref array<ParticleSource> m_ParticleSources = new array<ParticleSource>();
■
■// TestOnePlaying
■int m_OnePlayingManagerID;
■bool m_bOnePlayingTestSuccess = false;
■
■// TestOnePlayingStandAloneAutoDestroy
■int m_OnePlayingSAADPSID;
■bool m_bOnePlayingSAADEnded = false;
■
■// TestOnePlayingStandAlone
■int m_OnePlayingSAPSID;
■bool m_bOnePlayingSAEnded = false;
■
■// TestStop
■int m_StopPSID;
■float m_StopAccumulatedTime;
■static const float STOP_ACCUMULATED_TIME_STOP_CUTOFF = 2;
■static const float STOP_ACCUMULATED_TIME_PLAY_CUTOFF = 3;
■bool m_bStopWasStopped = false;
■bool m_bStopWasResumed = false;
■bool m_bStopEnded = false;
■
■//---------------------------------------------------------------------------
■// Ctor - Decides the tests to run
■//---------------------------------------------------------------------------
■void PMTPlayback()
■{
■■//AddInitTest("TestOnePlaying");
■■//AddInitTest("TestOnePlayingStandAloneAutoDestroy");
■■//AddInitTest("TestOnePlayingStandAlone");
■■//AddInitTest("TestWiggleStress");
■■AddInitTest("TestStopping");
■}
■
■//---------------------------------------------------------------------------
■// Tests
■//---------------------------------------------------------------------------
■// Test one particle playing
■TFResult TestOnePlaying()
■{
■■ParticleManager pm = CreatePMFixedBlocking(1);
■■m_OnePlayingManagerID = InsertManager(pm);

■■DayZPlayer player = GetGame().GetPlayer();
■■
■■ParticleSource p = pm.CreateParticleById(ParticleList.EXPLOSION_LANDMINE, new ParticlePropertie
■■p.GetEvents().Event_OnParticleEnd.Insert(PassOnePlaying);
■■
■■AddFrameTest("CheckOnePlaying");

■■return BTFR(p.IsParticlePlaying());
■}
■
■//---------------------------------------------------------------------------
■// Test one standalone particle playing which will auto destroy
```

```
// ---------------------------------------------
// ------ PointLightBase.c - START --------------------
// ---------------------------------------------


class PointLightBase extends ScriptedLightBase
{
■void PointLightBase()
■{
■■SetLightType(LightSourceType.PointLight); // This function must be called in constructor of the light!
■}
}


// ---------------------------------------------
// ------ PointLightBase.c - END ----------------------
// ---------------------------------------------


// ---------------------------------------------
// ------ Poisoning.c - START --------------------
// ---------------------------------------------


class PoisoningMdfr: ModifierBase
{
■static const int AGENT_THRESHOLD_ACTIVATE = 150;
■static const int AGENT_THRESHOLD_DEACTIVATE = 0;
■
■static const int VOMIT_OCCURRENCES_PER_HOUR_MIN = 60;
■static const int VOMIT_OCCURRENCES_PER_HOUR_MAX = 120;

■static const int WATER_DRAIN_FROM_VOMIT = 70;
■static const int ENERGY_DRAIN_FROM_VOMIT = 55;
■
■override void Init()
■{
■■m_TrackActivatedTime■= true;
■■m_IsPersistent■■■= true;
■■m_ID ■■■■■= eModifiers.MDF_POISONING;
■■m_TickIntervalInactive ■= DEFAULT_TICK_TIME_INACTIVE;
■■m_TickIntervalActive ■= DEFAULT_TICK_TIME_ACTIVE;
■}
■
■override string GetDebugText()
■{
■■return ("Activate threshold: "+AGENT_THRESHOLD_ACTIVATE + "| " +"Deativate threshold: "+AGEN
■}
■
■override bool ActivateCondition(PlayerBase player)
■{
■■if(player.GetSingleAgentCount(eAgents.FOOD_POISON) >= AGENT_THRESHOLD_ACTIVATE)
■■{
■■■return true;
■■}
■■else
■■{
■■■return false;
■■}
■}
■
■override bool DeactivateCondition(PlayerBase player)
■{
```

```
// ---------------------------------------------
// ------ PokeHolesBarrel.c - START --------------------
// ---------------------------------------------


class PokeHolesBarrel extends RecipeBase
{■
■override void Init()
■{
■■m_Name = "#STR_PokeHolesBarrel0";
■■m_IsInstaRecipe = false;//should this recipe be performed instantly without animation
■■m_AnimationLength = 2;//animation length in relative time units
■■m_Specialty = 0.01;// value > 0 for roughness, value < 0 for precision

■■//conditions
■■m_MinDamageIngredient[0] = -1;
■■m_MaxDamageIngredient[0] = 3;
■■m_MinQuantityIngredient[0] = -1;
■■m_MaxQuantityIngredient[0] = -1;
■■
■■m_MinDamageIngredient[1] = -1;
■■m_MaxDamageIngredient[1] = 3;
■■m_MinQuantityIngredient[1] = -1;
■■m_MaxQuantityIngredient[1] = -1;
■■
■■
■■//ingredient 1
■■InsertIngredient(0,"Barrel_ColorBase");//you can insert multiple ingredients this way

■■m_IngredientAddHealth[0] = 0;
■■m_IngredientAddQuantity[0] = 0;
■■m_IngredientSetHealth[0] = -1; // -1 = do nothing
■■m_IngredientDestroy[0] = 1;
■■m_IngredientUseSoftSkills[1] = false;// set 'true' to allow modification of the values by softskills on this i
■■
■■//ingredient 2
■■InsertIngredient(1,"Sickle");//you can insert multiple ingredients this way
■■InsertIngredient(1,"KukriKnife");
■■InsertIngredient(1,"FangeKnife");
■■InsertIngredient(1,"Hacksaw");
■■InsertIngredient(1,"KitchenKnife");
■■InsertIngredient(1,"SteakKnife");
■■InsertIngredient(1,"HayHook");
■■InsertIngredient(1,"StoneKnife");
■■InsertIngredient(1,"Cleaver");
■■InsertIngredient(1,"CombatKnife");
■■InsertIngredient(1,"HuntingKnife");
■■InsertIngredient(1,"Machete");
■■InsertIngredient(1,"CrudeMachete");
■■InsertIngredient(1,"OrientalMachete");
■■InsertIngredient(1,"Screwdriver");
■■InsertIngredient(1,"Crowbar");
■■InsertIngredient(1,"Pickaxe");
■■InsertIngredient(1,"WoodAxe");
■■InsertIngredient(1,"Hatchet");
■■InsertIngredient(1,"FirefighterAxe");
■■InsertIngredient(1,"Sword");
■■InsertIngredient(1,"AK_Bayonet");
■■InsertIngredient(1,"M9A1_Bayonet");
■■InsertIngredient(1,"Mosin_Bayonet");
■■InsertIngredient(1,"SKS_Bayonet");
■■
■■m_IngredientAddHealth[1] = -100;
```

```
// ---------------------------------------------
// ------ PoliceCap.c - START --------------------
// ---------------------------------------------


class PoliceCap extends Clothing
{
}



// ---------------------------------------------
// ------ PoliceCap.c - END ----------------------
// ---------------------------------------------


// ---------------------------------------------
// ------ PoliceJacket.c - START --------------------
// ---------------------------------------------


class PoliceJacket extends Clothing
{
■override void SetActions()
■{
■■super.SetActions();
■■AddAction(ActionWringClothes);
■}
};


// ---------------------------------------------
// ------ PoliceJacket.c - END ----------------------
// ---------------------------------------------


// ---------------------------------------------
// ------ PoliceJacketOrel.c - START --------------------
// ---------------------------------------------


class PoliceJacketOrel extends Clothing
{
■override void SetActions()
■{
■■super.SetActions();
■■AddAction(ActionWringClothes);
■}
};


// ---------------------------------------------
// ------ PoliceJacketOrel.c - END ----------------------
// ---------------------------------------------


// ---------------------------------------------
// ------ PolicePants.c - START --------------------
// ---------------------------------------------


class PolicePants extends Clothing
{
```

```
// ---------------------------------------------
// ------ PolicePantsOrel.c - START --------------------
// ---------------------------------------------


class PolicePantsOrel extends Clothing
{
■override void SetActions()
■{
■■super.SetActions();
■■AddAction(ActionWringClothes);
■}
};


// ---------------------------------------------
// ------ PolicePantsOrel.c - END ----------------------
// ---------------------------------------------


// ---------------------------------------------
// ------ PoliceVest.c - START --------------------
// ---------------------------------------------


class PoliceVest extends Clothing {};


// ---------------------------------------------
// ------ PoliceVest.c - END ----------------------
// ---------------------------------------------


// ---------------------------------------------
// ------ PortableGasLamp.c - START --------------------
// ---------------------------------------------


class PortableGasLamp extends ItemBase
{
■PortableGasLampLight■m_Light;
■
■private const string GAS_LIGHT_MATERIAL_ON ■■= "dz\\gear\\cooking\\data\\GasLightOn.rvmat";
■private const string GAS_LIGHT_MATERIAL_OFF ■= "dz\\data\\data\\default.rvmat";

■//sound
■const string SOUND_BURNING ■■= "portablegaslamp_burn_SoundSet";
■const string SOUND_TURN_ON■■= "portablegaslamp_turn_on_SoundSet";
■const string SOUND_TURN_OFF■■= "portablegaslamp_turn_off_SoundSet";
■
■protected EffectSound m_SoundBurningLoop;
■protected EffectSound m_SoundTurnOn;
■protected EffectSound m_SoundTurnOff;
■
■//--- POWER EVENTS
■override void OnSwitchOn()
■{
■■super.OnSwitchOn();
■■
■■//sound (client only)
■■SoundTurnOn();
■}
```

```
// ---------------------------------------------
// ------ PortableGasLampLight.c - START --------------------
// ---------------------------------------------


class PortableGasLampLight extends PointLightBase
{
■static float m_TorchRadius = 15;
■static float m_TorchBrightness = 5.0;
■
■void PortableGasLampLight()
■{
■■SetVisibleDuringDaylight(false);
■■SetRadiusTo( m_TorchRadius );
■■SetBrightnessTo(m_TorchBrightness);
■■SetCastShadow(true);
■■FadeIn(0.5);
■■SetFadeOutTime(0.1);
■■SetDiffuseColor(1.0, 0.7, 0.5);
■■SetAmbientColor(1.0, 0.7, 0.5);
■■SetFlickerAmplitude(0.3);
■■SetFlickerSpeed(0.75);
■■SetDancingShadowsMovementSpeed(0.005);
■■SetDancingShadowsAmplitude(0.003);
■}
■
■/*override void OnFrameLightSource(IEntity other, float timeSlice)
■{
■■
■}*/
}


// ---------------------------------------------
// ------ PortableGasLampLight.c - END ----------------------
// ---------------------------------------------


// ---------------------------------------------
// ------ PortableGasStove.c - START --------------------
// ---------------------------------------------


class PortableGasStove extends ItemBase
{
■StoveLight■m_Light;
■
■protected const string FLAME_BUTANE_ON ■■■= "dz\\gear\\cooking\\data\\flame_butane_ca.paa";
■protected const string FLAME_BUTANE_OFF ■■= "";
■typename ATTACHMENT_COOKING_POT ■■■■= Pot;
■typename ATTACHMENT_FRYING_PAN ■■■■■= FryingPan;
■typename ATTACHMENT_CAULDRON■ ■■■■= Cauldron;
■
■//cooking
■protected const float PARAM_COOKING_TEMP_THRESHOLD■■■= 100;■■//temperature threshold fo
■protected const float PARAM_COOKING_EQUIP_TEMP_INCREASE■■= 10;■■//how much will temper
■protected const float PARAM_COOKING_EQUIP_MAX_TEMP■■■= 250;■■//maximum temperature of
■protected const float PARAM_COOKING_TIME_INC_COEF■■■= 0.5;■■//cooking time increase coefic
■
■private ■■float m_TimeFactor;
■//
■ref Cooking m_CookingProcess;
■ItemBase m_CookingEquipment;
```

```
// ---------------------------------------------
// ------ Pot.c - START --------------------
// ---------------------------------------------


class Pot extends Bottle_Base
{
■void Pot()
■{

■}
■
■void ~Pot()
■{

■}
■
■override bool IsContainer()
■{
■■return true;
■}
■
■override string GetPouringSoundset()
■{
■■return "emptyVessle_Pot_SoundSet";
■}
■
■■override string GetEmptyingLoopSoundsetHard()
■{
■■return "pour_HardGround_Pot_SoundSet";
■}
■
■override string GetEmptyingLoopSoundsetSoft()
■{
■■return "pour_SoftGround_Pot_SoundSet";
■}
■
■override string GetEmptyingLoopSoundsetWater()
■{
■■return "pour_Water_Pot_SoundSet";
■}
■
■override string GetEmptyingEndSoundsetHard()
■{
■■return "pour_End_HardGround_Pot_SoundSet";
■}
■
■override string GetEmptyingEndSoundsetSoft()
■{
■■return "pour_End_SoftGround_Pot_SoundSet";
■}
■
■override string GetEmptyingEndSoundsetWater()
■{
■■return "pour_End_Water_Pot_SoundSet";
■}
■
■override bool CanPutInCargo( EntityAI parent )
■{
■■if ( !super.CanPutInCargo( parent ) )
■■■return false;
■■
■■if ( parent && IsCargoException4x3( parent ) )
```

```
// ----------------------------------------------
// ------ Potato.c - START --------------------
// ----------------------------------------------


class Potato : Edible_Base
{
■override bool CanBeCooked()
■{
■■return true;
■}■■
■
■override bool CanBeCookedOnStick()
■{
■■return true;
■}■
■
■override bool IsFruit()
■{
■■return true;
■}

■override bool CanDecay()
■{
■■return true;
■}
■
■override void SetActions()
■{
■■super.SetActions();
■■
■■AddAction(ActionForceFeed);
■■AddAction(ActionEatFruit);
■■AddAction(ActionCreateIndoorFireplace);
■■AddAction(ActionCreateIndoorOven);
■}
■
■override void EEOnCECreate()
■{
■■int rand = Math.RandomInt(0,10);

■■if ( rand > 6 )
■■{
■■■ChangeFoodStage( FoodStageType.ROTTEN );
■■■SetHealth( "", "", GetMaxHealth()*0.1 );
■■}
■■else if ( rand > 2 )
■■{
■■■ChangeFoodStage( FoodStageType.DRIED );
■■■SetHealth( "", "", GetMaxHealth()*0.4 );
■■}
■}
}



// ----------------------------------------------
// ------ Potato.c - END ---------------------
// ----------------------------------------------
// ---------------------------------------------
// ------ PourLiquid.c - START --------------------
// ---------------------------------------------


class PourLiquid extends RecipeBase
{■
■override void Init()
■{
■■m_Name = "#STR_PourLiquid0";
■■m_IsInstaRecipe = false;//should this recipe be performed instantly without animation
■■m_AnimationLength = 1.5;//animation length in relative time units
■■m_Specialty = -0.01;// value > 0 for roughness, value < 0 for precision
■■
■■
■■//conditions
■■m_MinDamageIngredient[0] = -1;//-1 = disable check
■■m_MaxDamageIngredient[0] = 3;//-1 = disable check
■■
■■m_MinQuantityIngredient[0] = 0;//-1 = disable check
■■m_MaxQuantityIngredient[0] = -1;//-1 = disable check
■■
■■m_MinDamageIngredient[1] = -1;//-1 = disable check
■■m_MaxDamageIngredient[1] = 3;//-1 = disable check
■■
■■m_MinQuantityIngredient[1] = 0;//-1 = disable check
■■m_MaxQuantityIngredient[1] = -1;//-1 = disable check
■■//----------------------------------------------------------------------------------------------------------------------
■■
■■//INGREDIENTS
■■//ingredient 1
■■InsertIngredient(0,"Pot");//you can insert multiple ingredients this way
■■InsertIngredient(0,"CanisterGasoline");//you can insert multiple ingredients this way
■■InsertIngredient(0,"Canteen");//you can insert multiple ingredients this way
■■InsertIngredient(0,"WaterBottle");//you can insert multiple ingredients this way
■■InsertIngredient(0,"Vodka");//you can insert multiple ingredients this way
■■InsertIngredient(0,"WaterPouch_ColorBase");//you can insert multiple ingredients this way
■■InsertIngredient(0,"Barrel_ColorBase");//you can insert multiple ingredients this way
■■
■■m_IngredientAddHealth[0] = 0;// 0 = do nothing
■■m_IngredientSetHealth[0] = -1; // -1 = do nothing
■■m_IngredientAddQuantity[0] = 0;// 0 = do nothing
■■m_IngredientDestroy[0] = false;//true = destroy, false = do nothing
■■m_IngredientUseSoftSkills[0] = false;// set 'true' to allow modification of the values by softskills on this i
■■
■■//ingredient 2
■■InsertIngredient(1,"Pot");//you can insert multiple ingredients this way
■■InsertIngredient(1,"CanisterGasoline");//you can insert multiple ingredients this way
■■InsertIngredient(1,"Canteen");//you can insert multiple ingredients this way
■■InsertIngredient(1,"WaterBottle");//you can insert multiple ingredients this way
■■InsertIngredient(1,"Vodka");//you can insert multiple ingredients this way
■■InsertIngredient(1,"WaterPouch_ColorBase");//you can insert multiple ingredients this way
■■InsertIngredient(1,"Barrel_ColorBase");//you can insert multiple ingredients this way
■■
■■m_IngredientAddHealth[1] = 0;// 0 = do nothing
■■m_IngredientSetHealth[1] = -1; // -1 = do nothing
■■m_IngredientAddQuantity[1] = 0;// 0 = do nothing
■■m_IngredientDestroy[1] = false;// false = do nothing
■■m_IngredientUseSoftSkills[1] = false;// set 'true' to allow modification of the values by softskills on this i
■■//----------------------------------------------------------------------------------------------------------------------
■■
■■//result1
■■//AddResult("");//add results here
```

```
// ---------------------------------------------
// ------ PowderedMilk.c - START --------------------
// ---------------------------------------------


class PowderedMilk: Edible_Base
{
■override void SetActions()
■{
■■super.SetActions();
■■
■■AddAction(ActionForceFeed);
■■AddAction(ActionEatBig);
■}
};



// ---------------------------------------------
// ------ PowderedMilk.c - END ----------------------
// ---------------------------------------------


// ---------------------------------------------
// ------ PoweredOptic_Base.c - START --------------------
// ---------------------------------------------


class PoweredOptic_Base extends ItemOptics
{
■protected PlayerBase ■■■■m_Player;
■
■void SetPlayer( PlayerBase player )
■{
■■m_Player = player;
■}
■
■PlayerBase GetPlayer()
■{
■■return m_Player;
■}
}



// ---------------------------------------------
// ------ PoweredOptic_Base.c - END ----------------------
// ---------------------------------------------


// ---------------------------------------------
// ------ PowerGenerator.c - START --------------------
// ---------------------------------------------


class PowerGenerator extends ItemBase
{
■float■■■■■■m_Fuel;
■private static float■■m_FuelTankCapacity; // Capacity in ml.
■private static float■■m_FuelToEnergyRatio; // Conversion ratio of 1 ml of fuel to X Energy
■private int■■■■■m_FuelPercentage;
■
■static const string■■■START_SOUND = "powerGeneratorTurnOn_SoundSet";
```

```
// ---------------------------------------------
// ------ PP19.c - START --------------------
// ---------------------------------------------


/**@class■PP19_Base
 * @brief■basic PP19 submachine gun
 **/
class PP19_Base : RifleBoltFree_Base
{
■void PP19_Base()
■{
■}
■
■override RecoilBase SpawnRecoilObject()
■{
■■return new PP19Recoil(this);
■}
■
■//some command is different for this weapon
■override int GetWeaponSpecificCommand(int weaponAction ,int subCommand)
■{
■■if ( weaponAction == WeaponActions.CHAMBERING)
■■{
■■■switch (subCommand)
■■■{
■■■■case WeaponActionChamberingTypes.CHAMBERING_ONEBULLET_UNIQUE_CLOSED:
■■■■■return WeaponActionChamberingTypes.CHAMBERING_ONEBULLET_OPENED;
■■■■
■■■■default:
■■■■■return subCommand;
■■■}
■■
■■}
■■return subCommand;
■}
■
■override bool CanEnterIronsights()
■{
■■ItemOptics optic = GetAttachedOptics();
■■if (optic && PSO1Optic.Cast(optic) || PSO11Optic.Cast(optic) || KazuarOptic.Cast(optic))
■■■return true;
■■return super.CanEnterIronsights();
■}
■
■//Debug menu Spawn Ground Special
■override void OnDebugSpawn()
■{
■■GameInventory inventory = GetInventory();
■■inventory.CreateInInventory( "PistolSuppressor" );
■■inventory.CreateInInventory( "PP19_Bttstck" );
■■inventory.CreateInInventory( "KobraOptic" );
■■inventory.CreateInInventory( "Battery9V" );
■■
■■SpawnAttachedMagazine("Mag_PP19_64Rnd");
■}
};


// ---------------------------------------------
// ------ PP19.c - END ----------------------
// ---------------------------------------------
// ---------------------------------------------
// ------ PP19Recoil.c - START --------------------
// ---------------------------------------------


class PP19Recoil: RecoilBase
{
■override void Init()
■{
■■vector point_1;
■■vector point_2;
■■vector point_3;
■■vector point_4;
■■point_1[0] = m_Player.GetRandomGeneratorSyncManager().GetRandomInRange(RandomGeneratorS
■■point_2[0] = m_Player.GetRandomGeneratorSyncManager().GetRandomInRange(RandomGeneratorS
■■point_3[0] = m_Player.GetRandomGeneratorSyncManager().GetRandomInRange(RandomGeneratorS
■■point_4[0] = m_Player.GetRandomGeneratorSyncManager().GetRandomInRange(RandomGeneratorS
■■m_HandsCurvePoints.Insert(point_1);//forms a 2 dimensional spline(z is ignored)
■■m_HandsCurvePoints.Insert(point_2);
■■m_HandsCurvePoints.Insert(point_3);
■■m_HandsCurvePoints.Insert(point_4);
■■m_HandsCurvePoints.Insert("0 0 0");
■■m_HandsOffsetRelativeTime = 1;
■■
■■m_MouseOffsetRangeMin = 60;//in degrees min
■■m_MouseOffsetRangeMax = 110;//in degrees max
■■m_MouseOffsetDistance = 0.6;//how far should the mouse travel
■■m_MouseOffsetRelativeTime = 0.5;//[0..1] a time it takes to move the mouse the required distance rela
■
■■m_CamOffsetDistance = 0.0075;
■■m_CamOffsetRelativeTime = 1;
■}
}


// ---------------------------------------------
// ------ PP19Recoil.c - END ----------------------
// ---------------------------------------------


// ---------------------------------------------
// ------ PPEChromAber.c - START --------------------
// ---------------------------------------------


//!ChromAber - PostProcessEffectType.ChromAber
class PPEChromAber: PPEClassBase
{
■static const int PARAM_POWERX = 0;
■static const int PARAM_POWERY = 1;
■
■static const int L_0_INTRO = 100;
■static const int L_1_INTRO = 100;
■
■override int GetPostProcessEffectID()
■{
■■return PostProcessEffectType.ChromAber;
■}
■
■override string GetDefaultMaterialPath()
■{
■■return "Graphics/Materials/postprocess/chromaber";
■}
```

```
// ---------------------------------------------
// ------ PPEColorGrading.c - START --------------------
// ---------------------------------------------


//!ColorGrading - PostProcessEffectType.ColorGrading
//TODO
class PPEColorGrading: PPEClassBase
{
■//static const int PARAM_COLORTABLE = 0;
■
■override int GetPostProcessEffectID()
■{
■■return PostProcessEffectType.ColorGrading;
■}
■
■override string GetDefaultMaterialPath()
■{
■■return "Graphics/Materials/postprocess/colorgrading";
■}
■
■override void RegisterMaterialParameters()
■{
■■//RegisterParameterTexture(PARAM_COLORTABLE,"ColorTable","Graphics/Textures/postprocess/col
■}
}


// ---------------------------------------------
// ------ PPEColorGrading.c - END ----------------------
// ---------------------------------------------


// ---------------------------------------------
// ------ PPEColors.c - START --------------------
// ---------------------------------------------


//!Colors - PostProcessEffectType.Colors
//TODO - 'ColorsEffect' type may be used differently in c++, no emat linked to it? Investigate.
class PPEColors: PPEClassBase
{
■static const int PARAM_BRIGHTNESS = 0;
■static const int PARAM_CONTRAST = 1;
■static const int PARAM_OFFSET = 2;
■static const int PARAM_OVERLAYFACTOR = 3;
■static const int PARAM_OVERLAYCOLOR = 4;
■static const int PARAM_SATURATION = 5;
■static const int PARAM_COLORIZATIONCOLOR = 6;
■static const int PARAM_DESATURATIONWEIGHTS = 7;
■
■override int GetPostProcessEffectID()
■{
■■return PostProcessEffectType.Colors;
■}
■
■override string GetDefaultMaterialPath()
■{
■■return "Graphics/Materials/postprocess/colors";
■}
■
■override void RegisterMaterialParameters()
■{
```

```
// ---------------------------------------------
// ------ PPEConstants.c - START --------------------
// ---------------------------------------------


//! PPE type priorities, C++ based. DO NOT CHANGE ORDER! Used only when calling 'SetCameraPostPr
enum PostProcessPrioritiesCamera
{
■PPP_SSAO,
■PPP_CLOUDS,
■PPP_DOF,
■PPP_ROTBLUR,
■PPP_GODRAYS,
■PPP_RAIN,
■PPP_RADIALBLUR,
■PPP_CHROMABER,
■PPP_WETDISTORT,
■PPP_DYNBLUR,
■PPP_UNDERWATER,
■PPP_DOF_BOKEH,
■PPP_COLORGRADE,
■PPP_GLOW,
■PPP_FILMGRAIN,
■PPP_FILMGRAIN_NV,
■PPP_FXAA,
■PPP_SMAA,
■PPP_GAUSS_FILTER,
■PPP_MEDIAN, //unused?
■//PPP_SSR
};

enum PPERequesterCategory
{
■NONE = 0;
■GAMEPLAY_EFFECTS = 2;
■MENU_EFFECTS = 4;
■MISC_EFFECTS = 8;
■ALL = 14; //GAMEPLAY_EFFECTS|MENU_EFFECTS|MISC_EFFECTS
};

/**
/brief IDs of custom PPE classes
/note Currently used for various native exceptions that used to be handled outside of script-side postproce
/note Can be used for custom functionality as well, C++ permitting.
*/
enum PPEExceptions
{
■NONE = -1,
■EXPOSURE = 50,
■DOF,
■EYEACCOM,
■NVLIGHTPARAMS,
};

//! PP operators, specify operation between subsequent layers
enum PPOperators
{
■LOWEST, //only lowest value gets used. Note - if first request, it is compared against default values!
■HIGHEST, //only highest value gets used. Note - if first request, it is compared against default values!
■ADD, //LINEAR addition
■ADD_RELATIVE, //LINEAR relative addition (relative to diff between current and max, where applicable.
■SUBSTRACT, //LINEAR substraction
■SUBSTRACT_RELATIVE, //LINEAR relative substraction
```

```
// ---------------------------------------------
// ------ PPEDepthOfField.c - START --------------------
// ---------------------------------------------


//!DepthOfField - PostProcessEffectType.DepthOfField
//TODO - may be just a dummy, since CGame.OverrideDOF function handles script overrides?
class PPEDepthOfField: PPEClassBase
{
■static const int PARAM_DOFLQ = 0;
■static const int PARAM_FOCALDISTANCE = 1;
■static const int PARAM_HYPERFOCAL = 2;
■static const int PARAM_FOCALOFFSET = 3;
■static const int PARAM_BLURFACTOR = 4;
■static const int PARAM_SIMPLEDOF = 5;
■static const int PARAM_SIMPLEHFNEAR = 6;
■static const int PARAM_SIMPLEDOFSIZE = 7;
■static const int PARAM_SIMPLEDOFGAUSS = 8;
■
■override int GetPostProcessEffectID()
■{
■■return PostProcessEffectType.DepthOfField;
■}
■
■override string GetDefaultMaterialPath()
■{
■■string ret = "Graphics/Materials/postprocess/depthoffieldTest2";
■■/*
■■GameOptions m_Options = new GameOptions();
■■ListOptionsAccess loa = ListOptionsAccess.Cast( m_Options.GetOptionByType( AT_POSTPROCESS
■■
■■switch (loa.GetIndex())
■■{
■■■case POSTPROCESS_OPTION_VALUE_LOW:
■■■■ret = "Graphics/Materials/postprocess/hbao_low";
■■■break;
■■■
■■■case POSTPROCESS_OPTION_VALUE_MEDIUM:
■■■■ret = "Graphics/Materials/postprocess/hbao_medium";
■■■break;
■■■
■■■case POSTPROCESS_OPTION_VALUE_HIGH:
■■■■ret = "Graphics/Materials/postprocess/hbao_high";
■■■break;
■■■
■■■case POSTPROCESS_OPTION_VALUE_HIGHEST:
■■■■ret = "Graphics/Materials/postprocess/hbao_highest";
■■■break;
■■}
■■*/
■■return ret;
■}
■
■override void RegisterMaterialParameters()
■{
■■RegisterParameterScalarBool(PARAM_DOFLQ,"DOFLowQuality",false);
■■RegisterParameterScalarFloat(PARAM_FOCALDISTANCE,"FocalDistance",0.1,0.0,1.0);
■■RegisterParameterScalarFloat(PARAM_HYPERFOCAL,"HyperFocal",0.85,0.1,100.0);
■■RegisterParameterScalarFloat(PARAM_FOCALOFFSET,"FocalOffset",0.0,0.0,1.0);
■■RegisterParameterScalarFloat(PARAM_BLURFACTOR,"BlurFactor",4.0,0.0,10.0);
■■RegisterParameterScalarBool(PARAM_SIMPLEDOF,"SimpleDOF",false);
■■RegisterParameterScalarFloat(PARAM_SIMPLEHFNEAR,"SimpleHyperFocalNear",0.7,0.1,100.0);
■■RegisterParameterScalarInt(PARAM_SIMPLEDOFSIZE,"SimpleDOFSize",2.0,1.0,4.0);
```

```
// ---------------------------------------------
// ------ PPEDOF.c - START --------------------
// ---------------------------------------------


//-------------------------------------------------------
//Native exceptions - legacy methods for direct access to specific postprocesses. Each one is evaluated ar

//!DOF postprocess, does not directly use materials
class PPEDOF: PPEClassBase
{
■//g_Game.OverrideDOF(bool enable, float focusDistance, float focusLength, float focusLengthNear, float
■static const int PARAM_ENABLE = 0;
■static const int PARAM_FOCUS_DIST = 1;
■static const int PARAM_FOCUS_LEN = 2;
■static const int PARAM_FOCUS_LEN_NEAR = 3;
■static const int PARAM_BLUR = 4;
■static const int PARAM_FOCUS_DEPTH_OFFSET = 5;
■
■static const int L_0_ADS = 100;
■static const int L_1_ADS = 100;
■static const int L_2_ADS = 100;
■static const int L_3_ADS = 100;
■static const int L_4_ADS = 100;
■static const int L_5_ADS = 100;
■
■override int GetPostProcessEffectID()
■{
■■return PPEExceptions.DOF;
■}
■
■override void RegisterMaterialParameters()
■{
■■ //no known max. 1000 used as max
■■RegisterParameterScalarBool(PARAM_ENABLE,"Enable",false);
■■
■■RegisterParameterScalarFloat(PARAM_FOCUS_DIST,"FocusDistance",2.0,0.0,1000.0); //2.0f used in
■■RegisterParameterScalarFloat(PARAM_FOCUS_LEN,"FocusLength",-1.0,-1.0,1000.0); //-1.0f used as
■■RegisterParameterScalarFloat(PARAM_FOCUS_LEN_NEAR,"FocusLengthNear",-1.0,-1.0,1000.0); //-
■■RegisterParameterScalarFloat(PARAM_BLUR,"Blur",1.0,0.0,1000.0); //1.0f used in code with null came
■■RegisterParameterScalarFloat(PARAM_FOCUS_DEPTH_OFFSET,"FocusDepthOffset",0.0,0.0,1000.0
■}
■
■override void ApplyValueChanges()
■{
■■if (m_UpdatedParameters.Count() > 0)
■■{
■■■SetFinalParameterValue(-1); //unique handling
■■}
■■
■■m_UpdatedParameters.Clear();
■}
■
■//! Overriden to handle the specific exception
■override void SetFinalParameterValue(int parameter_idx)
■{
■■Param enabled_par = GetParameterCommandData(PARAM_ENABLE).GetCurrentValues();
■■bool is_enabled = Param1<bool>.Cast(enabled_par).param1;
■■
■■if (is_enabled)
■■{
■■■array<float> array_values = {-1.0};
■■■
```

```
// ---------------------------------------------
// ------ PPEDynamicBlur.c - START --------------------
// ---------------------------------------------


//!DynamicBlur - PostProcessEffectType.DynamicBlur
class PPEDynamicBlur: PPEClassBase
{
■static const int PARAM_BLURRINESS = 0;
■
■override int GetPostProcessEffectID()
■{
■■return PostProcessEffectType.DynamicBlur;
■}
■
■override string GetDefaultMaterialPath()
■{
■■return "Graphics/Materials/postprocess/dynamicblur";
■}
■
■override void RegisterMaterialParameters()
■{
■■RegisterParameterScalarFloat(PARAM_BLURRINESS,"Blurriness",20,0,30);
■}
}


// ---------------------------------------------
// ------ PPEDynamicBlur.c - END ----------------------
// ---------------------------------------------


// ---------------------------------------------
// ------ PPEExposureNative.c - START --------------------
// ---------------------------------------------


//-------------------------------------------------------
//Native exceptions - legacy methods for direct access to specific postprocesses. Each one is evaluated ar

//!EV postprocess, does not directly use materials
class PPEExposureNative: PPEClassBase
{
■//g_Game.SetEVValue
■static const int PARAM_INTENSITY = 0;
■
■static const int L_0_NVG_OPTIC = 100;
■static const int L_0_NVG_GOGGLES = 200;
■static const int L_0_NVG_OFF = 300;
■static const int L_0_FLASHBANG = 400;
■static const int L_0_BURLAP = 500;
■static const int L_0_DEATH = 1000;
■
■override int GetPostProcessEffectID()
■{
■■return PPEExceptions.EXPOSURE;
■}
■
■override void RegisterMaterialParameters()
■{
■■RegisterParameterScalarFloat(PARAM_INTENSITY,"Intensity",0.0,-100.0,100.0); //no real range, it see
■}
■
```

```
// ---------------------------------------------
// ------ PPEEyeAccomodationNative.c - START --------------------
// ---------------------------------------------


//-------------------------------------------------------
//Native exceptions - legacy methods for direct access to specific postprocesses. Each one is evaluated ar

//!Eye Accomodation postprocess, does not directly use materials
class PPEEyeAccomodationNative: PPEClassBase
{
■//g_Game.GetWorld().SetEyeAccom
■static const int PARAM_INTENSITY = 0;
■
■static const int L_0_UNDERGROUND = 100;
■static const int L_0_NVG_GENERIC = 300;
■/*static const int L_0_NVG_OPTIC = 400;
■static const int L_0_NVG_GOGGLES = 401;
■static const int L_0_NVG_OFF = 402;*/
■static const int L_0_BURLAP = 500;
■
■override int GetPostProcessEffectID()
■{
■■return PPEExceptions.EYEACCOM;
■}
■
■override void RegisterMaterialParameters()
■{
■■RegisterParameterScalarFloat(PARAM_INTENSITY,"Intensity",1.0,0.0,1000.0); //no known max. 1000
■}
■
■//! Overriden to handle the specific exception
■override void SetFinalParameterValue(int parameter_idx)
■{
■■Param values = GetParameterCommandData(parameter_idx).GetCurrentValues();
■■float value_var_float = Param1<float>.Cast(values).param1;
■■
■■g_Game.GetWorld().SetEyeAccom(value_var_float);
■■//DbgPrnt("PPEDebug | SetFinalParameterValue | PPEEyeAccomodationNative | float val: " + value_va
■}
}



// ---------------------------------------------
// ------ PPEEyeAccomodationNative.c - END ---------------------
// ---------------------------------------------


// ---------------------------------------------
// ------ PPEffects.c - START --------------------
// ---------------------------------------------


//! Deprecated; 'PPEManager' used instead
class PPEffects
{
■// COLORIZE IDs
■static const int COLORIZE_NV = 100;
■
■//CONSTANTS
■static const float COLOR_SHOCK = 0.1;//shock color value (relative) //todo
■
```

```
// ---------------------------------------------
// ------ PPEFilmGrain.c - START --------------------
// ---------------------------------------------


//!FilmGrain - PostProcessEffectType.FilmGrain
/*!
■Be advised, for this to take any effect, 'g_Game.NightVissionLightParams' needs to have non-zero 'noise
■/note set by 'PPELightIntensityParamsNative.PARAM_NOISE_MULT' currently
*/
class PPEFilmGrain: PPEClassBase
{
■static const int PARAM_INTENSITY = 0;
■static const int PARAM_SHARPNESS = 1;
■static const int PARAM_GRAINSIZE = 2;
■static const int PARAM_INTENSITYX0 = 3;
■static const int PARAM_INTENSITYX1 = 4;
■static const int PARAM_MONOCHROMATIC = 5;
■static const int PARAM_SIMPLE = 6;
■static const int PARAM_DISTORT = 7;
■static const int PARAM_FREQUENCY = 8;
■//static const int PARAM_NOISEMAP = 9;
■
■static const int L_1_NVG = 100;
■static const int L_1_TOXIC_TINT = 200;
■static const int L_2_NVG = 100;
■static const int L_2_TOXIC_TINT = 200;
■
■override int GetPostProcessEffectID()
■{
■■return PostProcessEffectType.FilmGrain;
■}
■
■override string GetDefaultMaterialPath()
■{
■■return "Graphics/Materials/postprocess/filmgrainNV"; //TODO - differentiate between filmgrainNV?
■}
■
■override void RegisterMaterialParameters()
■{
■■RegisterParameterScalarFloat(PARAM_INTENSITY,"Intensity",0.0,0.0,1.0);
■■RegisterParameterScalarFloat(PARAM_SHARPNESS,"Sharpness",2.35,0.0,20.0);
■■RegisterParameterScalarFloat(PARAM_GRAINSIZE,"GrainSize",2.75,1.0,9.0);
■■RegisterParameterScalarFloat(PARAM_INTENSITYX0,"IntensityX0",0.0,0.0,1.0);
■■RegisterParameterScalarFloat(PARAM_INTENSITYX1,"IntensityX1",0.0,0.0,1.0);
■■RegisterParameterScalarBool(PARAM_MONOCHROMATIC,"Monochromatic",true);
■■RegisterParameterScalarBool(PARAM_SIMPLE,"Simple",false);
■■RegisterParameterScalarBool(PARAM_DISTORT,"Distort",true);
■■RegisterParameterScalarFloat(PARAM_FREQUENCY,"Frequency",20.0,1.0,1000.0);
■■//RegisterParameterTexture(PARAM_NOISEMAP,"NoiseMap","{0B1C7AEDC4645C8A}System/texture
■}
}


// ---------------------------------------------
// ------ PPEFilmGrain.c - END ---------------------
// ---------------------------------------------
// ---------------------------------------------
// ------ PPEFXAA.c - START --------------------
// ---------------------------------------------


//!FXAA - PostProcessEffectType.FXAA
class PPEFXAA: PPEClassBase
{
■static const int PARAM_PRESET = 0;
■
■override int GetPostProcessEffectID()
■{
■■return PostProcessEffectType.FXAA;
■}
■
■override string GetDefaultMaterialPath()
■{
■■return "Graphics/Materials/postprocess/fxaa";
■}
■
■override void RegisterMaterialParameters()
■{
■■RegisterParameterScalarInt(PARAM_PRESET,"Preset",0,0,5);
■}
}


// ---------------------------------------------
// ------ PPEFXAA.c - END ----------------------
// ---------------------------------------------


// ---------------------------------------------
// ------ PPEGaussFilter.c - START --------------------
// ---------------------------------------------


//!GaussFilter - PostProcessEffectType.GaussFilter
class PPEGaussFilter: PPEClassBase
{
■static const int PARAM_INTENSITY = 0;
■
■static const int L_0_ADS = 100;
■static const int L_0_SHOCK = 200;
■static const int L_0_FEVER = 300;
■static const int L_0_FLASHBANG = 400;
■static const int L_0_INV = 500;
■static const int L_0_MENU = 600;
■static const int L_0_CONTROLS = 700;
■static const int L_0_TUTORIALS = 800;
■static const int L_0_SERVER_BROWSER = 900;
■static const int L_0_DISCONNECT = 1000;
■
■override int GetPostProcessEffectID()
■{
■■return PostProcessEffectType.GaussFilter;
■}
■
■override string GetDefaultMaterialPath()
■{
■■return "Graphics/Materials/postprocess/gauss";
■}
■
```

```
// ---------------------------------------------
// ------ PPEGlow.c - START --------------------
// ---------------------------------------------


/*enum PPEGlow
{
■PARAM_VISIBLEPERCENT = 0,
}*/

//!Glow - PostProcessEffectType.Glow
class PPEGlow: PPEClassBase
{
■static const int PARAM_VISIBLEPERCENT = 0;
■static const int PARAM_TARGETBRIGHTNESS = 1;
■static const int PARAM_SPEEDDARKTOBRIGHT = 2;
■static const int PARAM_SPEEDBRIGHTTODARK = 3;
■static const int PARAM_TONEMAPPING = 4;
■static const int PARAM_HDR = 5;
■static const int PARAM_FILMICSHOULDERSTRENGTH = 6;
■static const int PARAM_FILMICLINEARSTRENGTH = 7;
■static const int PARAM_FILMICLINEARANGLE = 8;
■static const int PARAM_FILMICTOESTRENGTH = 9;
■static const int PARAM_FILMICTOENUMERATOR = 10;
■static const int PARAM_FILMICTOEDENUMERATOR = 11;
■static const int PARAM_FILMICEXPOSUREBIAS = 12;
■static const int PARAM_FILMICWHITEPOINT = 13;
■static const int PARAM_BLOOMTHRESHOLD = 14;
■static const int PARAM_BLOOMSTEEPNESS = 15;
■static const int PARAM_BLOOMINTENSITY = 16;
■static const int PARAM_BRIGHTNESS = 17;
■static const int PARAM_CONTRAST = 18;
■static const int PARAM_OFFSET = 19;
■static const int PARAM_OVERLAYFACTOR = 20;
■static const int PARAM_OVERLAYCOLOR = 21;
■static const int PARAM_SATURATION = 22;
■static const int PARAM_COLORIZATIONCOLOR = 23;
■static const int PARAM_DESATURATIONWEIGHTS = 24;
■static const int PARAM_VIGNETTE = 25;
■static const int PARAM_VIGNETTECOLOR = 26;
■static const int PARAM_LENSDISTORT = 27;
■static const int PARAM_MAXCHROMABBERATION = 28;
■static const int PARAM_LENSCENTERX = 29;
■static const int PARAM_LENSCENTERY = 30;
■
■//layer info
■static const int L_20_HIT = 100;
■static const int L_20_FLASHBANG = 300;
■static const int L_20_SHOCK = 500;
■
■static const int L_21_SHOCK = 100;
■static const int L_21_HIT = 300;
■static const int L_21_FLASHBANG = 500;
■
■static const int L_22_BLOODLOSS = 100;
■
■static const int L_23_GLASSES = 100;
■static const int L_23_TOXIC_TINT = 200;
■static const int L_23_NVG = 600;
■
■static const int L_25_MENU = 100;
■static const int L_25_TUNNEL = 300;
■static const int L_25_SHOCK = 500;
```

```
// ---------------------------------------------
// ------ PPEGodRays.c - START --------------------
// ---------------------------------------------


//!GodRays - PostProcessEffectType.GodRays
class PPEGodRays: PPEClassBase
{
■static const int PARAM_INTENSITY = 0;
■static const int PARAM_SUNVISIBLE = 1;
■static const int PARAM_OVERBURNINTENSITY = 2;
■static const int PARAM_OVERBURNSTART = 3;
■static const int PARAM_OVERBURNEND = 4;
■//static const int PARAM_SUNMASKMAT = 5; //Warning, not to be used until DECLARE_RESOURCE_N
■
■static const int L_0_GLASSES = 100;
■
■override int GetPostProcessEffectID()
■{
■■return PostProcessEffectType.GodRays;
■}
■
■override string GetDefaultMaterialPath()
■{
■■return "Graphics/Materials/postprocess/godrayssun";
■}
■
■override void RegisterMaterialParameters()
■{
■■RegisterParameterScalarFloat(PARAM_INTENSITY,"Intensity",0.8,0,1);
■■RegisterParameterScalarFloat(PARAM_SUNVISIBLE,"SunVisible",1.0,0,1);
■■RegisterParameterScalarFloat(PARAM_OVERBURNINTENSITY,"OverBurnIntensity",0.25,0,1);
■■RegisterParameterScalarFloat(PARAM_OVERBURNSTART,"OverBurnStart",0.025,0,1);
■■RegisterParameterScalarFloat(PARAM_OVERBURNEND,"OverBurnEnd",0.175,0,1);
■■
■■//TODO
■■//SunMaskMat - DECLARE_RESOURCE_NAME
■}
}


// ---------------------------------------------
// ------ PPEGodRays.c - END ----------------------
// ---------------------------------------------



// ---------------------------------------------
// ------ PPEHBAO.c - START --------------------
// ---------------------------------------------


//!HBAO - PostProcessEffectType.HBAO
//TODO - may be just a dummy, since SSAO already uses the HBAO materials?
class PPEHBAO: PPEClassBase
{
■static const int PARAM_RADIUSMETERS = 0;
■static const int PARAM_INTENSITY = 1;
■static const int PARAM_BLURSHARPNESS = 2;
■static const int PARAM_NDOTVBIAS = 3;
■static const int PARAM_SMALLSCALEAO = 4;
■static const int PARAM_LARGESCALEAO = 5;
■static const int PARAM_NUMDIRECTIONS = 6;
■static const int PARAM_NUMSAMPLES = 7;
```

```
// ---------------------------------------------
// ------ PPELightIntensityParamsNative.c - START --------------------
// ---------------------------------------------


//-------------------------------------------------------
//Native exceptions - legacy methods for direct access to specific postprocesses. Each one is evaluated ar

//!g_Game.NightVissionLightParams, does not directly use materials. Controls light multiplication and filmg
class PPELightIntensityParamsNative: PPEClassBase
{
■//g_Game.SetEVValue
■static const int PARAM_LIGHT_MULT = 0;
■static const int PARAM_NOISE_MULT = 1;
■
■static const int L_0_NVG = 100;
■static const int L_0_TOXIC_TINT = 200;
■static const int L_1_NVG = 100;
■static const int L_1_TOXIC_TINT = 200;
■
■override int GetPostProcessEffectID()
■{
■■return PPEExceptions.NVLIGHTPARAMS;
■}
■
■override void RegisterMaterialParameters()
■{
■■RegisterParameterScalarFloat(PARAM_LIGHT_MULT,"lightIntensityMul",1.0,0.0,50.0); //some reasona
■■RegisterParameterScalarFloat(PARAM_NOISE_MULT,"noiseIntensity",0.0,0.0,50.0); //some reasonab
■}
■
■override void ApplyValueChanges()
■{
■■if (m_UpdatedParameters.Count() > 0)
■■{
■■■SetFinalParameterValue(-1); //unique handling
■■}
■■
■■m_UpdatedParameters.Clear();
■}
■
■//! Overriden to handle the specific exception
■override void SetFinalParameterValue(int parameter_idx)
■{
■■array<float> array_values = new array<float>;
■■
■■for (int i = 0; i < PARAM_NOISE_MULT + 1; i++)
■■{
■■■Param values = GetParameterCommandData(i).GetCurrentValues();
■■■float value_var_float = Param1<float>.Cast(values).param1;
■■■array_values.Insert(value_var_float);
■■}
■■
■■g_Game.NightVissionLightParams(array_values.Get(PARAM_LIGHT_MULT),array_values.Get(PARA
■■
■■//DbgPrnt("PPEDebug | SetFinalParameterValue | PPELightIntensityParamsNative | float val: " + value
■}
}


// ---------------------------------------------
// ------ PPELightIntensityParamsNative.c - END ----------------------
// ---------------------------------------------
// ---------------------------------------------
// ------ PPEManager.c - START --------------------
// ---------------------------------------------


//! Static component of PPE manager, used to hold the instance.
class PPEManagerStatic
{
■static ref PPEManager m_Manager;
■
■static void CreateManagerStatic()
■{
■■if (m_Manager)
■■{
■■■Debug.Log("PPEManagerStatic | CreateManagerStatic - PPEManager already exists");
■■■return;
■■}
■■
■■m_Manager = new PPEManager;
■}
■
■static void DestroyManagerStatic()
■{
■■if (m_Manager)
■■{
■■■m_Manager.Cleanup();
■■■delete m_Manager;
■■}
■}
■
■//! Returns the manager instance singleton
■static PPEManager GetPPEManager()
■{
■■return m_Manager;
■}
}

/**
/brief Postprocess manager, responsible for updates, receiving, and re-distributing requester data to their r
■/par Basic post process flow outline:
■Getting a registered 'PPERequester' instance from the 'PPERequesterBank'
■
■/par
■Launching the requester, either through an overriden 'Start' method, or custom method with some setters
■
■/par On render update, PPEManager:
■Handles queued requester changes, re-distributes individual commands to material structure
■■
■■/par
■■Updates the material/parameter structure and calculates the blend values
■■
■■/par
■■Sets the final values via native functions (only called once per changed parameter - optimization stonks
■
■/note Requester serves as a centralized platform for specific effec/group of effects. Although technically t
■/note and clearer command hierarchy (no value setters without clear parentage).
*/
class PPEManager extends Managed
{
■const int CAMERA_ID = 0;
■
■protected bool ■■■■■■■■■m_ManagerInitialized;
■protected ref map<int, ref PPEClassBase> ■■m_PPEClassMap; //contains sorted postprocess classes
```

```
// ---------------------------------------------
// ------ PPEMatClassesBase.c - START --------------------
// ---------------------------------------------


//! Created once, on manager init. Script-side representation of C++ material class, separate handling.
class PPEClassBase
{
■protected PPEManager ■■■■■■■■■■■■m_Manager;
■protected string ■■■■■■■■■■■■■m_MaterialPath = "";
■protected Material ■■■■■■■■■■■■■m_Material;
■
■protected ref map<int, ref array<int>>■■■■■■■■m_ParameterUpdateQueueMap;
■protected ref array<int> ■■■■■■■■■■■m_ParameterRemovalQueue;
■protected ref array<int> ■■■■■■■■■■■m_UpdatedParameters;
■
■protected ref map<int,ref PPEMatClassParameterCommandData> ■■m_MaterialParamMapStructure; //
■
■void PPEClassBase(string mat_path_override = "")
■{
■■Init(mat_path_override);
■■CreateMaterial();
■■CreateDataStructure();
■■RegisterMaterialParameters();
■}
■
■protected void Init(string mat_path_override = "")
■{
■■if (mat_path_override != "")
■■{
■■■m_MaterialPath = mat_path_override;
■■}
■■else
■■{
■■■m_MaterialPath = GetDefaultMaterialPath();
■■}
■■m_Manager = PPEManagerStatic.GetPPEManager();
■}
■
■protected void CreateMaterial()
■{
■■if (m_MaterialPath != "")
■■■m_Material = GetGame().GetWorld().GetMaterial(m_MaterialPath);
■}
■
■Material GetMaterial()
■{
■■return m_Material;
■}
■
■//do not override!
■protected void CreateDataStructure()
■{
■■m_MaterialParamMapStructure = new map<int,ref PPEMatClassParameterCommandData>;
■■//m_ParameterUpdateQueue = new array<int>;
■■m_ParameterUpdateQueueMap = new map<int, ref array<int>>;
■■m_ParameterRemovalQueue = new array<int>;
■■m_UpdatedParameters = new array<int>;
■}
■
■//! inserted into associative array by parameter int value, parameter registration order does not matter (st
■protected void RegisterMaterialParameters();
■
```

```
// ----------------------------------------------
// ------ PPEMatClassParameterBool.c - START --------------------
// ----------------------------------------------


class PPEMatClassParameterBool extends PPEMatClassParameterCommandData
{
■protected ref map<int,ref array<bool,int>> m_LayerInfo; //<priority,<value,operator>>
■
■protected PPETemplateDefBool m_Bool;
■protected bool m_ValueDefault;
■
■void PPEMatClassParameterBool(int mat_idx, int parameter_idx, PPEClassBase parent)
■{
■■m_LayerInfo = new map<int,ref array<bool,int>>;
■}
■
■override void InitDefaults()
■{
■■Class.CastTo(m_Bool,m_Defaults);
■■m_ValueDefault = m_Bool.param2;
■}
■
■override void InitCuttent()
■{
■■m_CurrentValues = new Param1<bool>(m_ValueDefault);
■}
■
■override int GetParameterVarType()
■{
■■return PPEConstants.VAR_TYPE_BOOL;
■}
■
■override void Update(float timeslice, out Param p_total, out bool setting_defaults, int order)
■{
■■super.Update(timeslice,p_total,setting_defaults,order);
■■
■■protected int active_request_count = 0;
■■
■■protected PPERequestParamDataBool req_data;
■■
■■protected bool setting_value_zero = false;
■■
■■protected bool bool_value_temp = false;
■■protected bool bool_value_total = m_ValueDefault;
■■
■■if (p_total == null)
■■{
■■■p_total = new Param1<bool>(false);
■■}
■■
■■if (m_RequestMap.Count() > 0)
■■{
■■■m_LayerInfo.Clear();
■■}
■■else
■■{
■■■//DbgPrnt("m_RequestMap.Count() is zero! Using default values. | mat/par: " + m_MaterialIndex + "/"
■■■SetParameterValueDefault(p_total);
■■■m_Parent.ParamUpdateRemove(m_ParameterIndex);
■■■return;
■■}
■■
```

```
// ---------------------------------------------
// ------ PPEMatClassParameterColor.c - START --------------------
// ---------------------------------------------



class ColorValuesData : Managed
{
■protected ref array<float> m_Values;
■protected int m_Operator;
■
■void ColorValuesData(ref array<float> values, int operator)
■{
■■m_Values = new array<float>;
■■m_Values.Copy(values);
■■m_Operator = operator;
■}
■
■void SetValues(ref array<float> values)
■{
■■m_Values.Copy(values);
■}
■
■array<float> GetValues()
■{
■■return m_Values;
■}
■
■void SetOperator(int operator)
■{
■■m_Operator = operator;
■}
■
■int GetOperator()
■{
■■return m_Operator;
■}
}

class PPEMatClassParameterColor extends PPEMatClassParameterCommandData
{
■const int VALUE_RED = 0;
■const int VALUE_GREEN = 1;
■const int VALUE_BLUE = 2;
■const int VALUE_ALPHA = 3;
■const int VALUES_COUNT = 4; //rgba
■
■protected PPETemplateDefColor m_Color;
■protected ref map<int,ref ColorValuesData> m_LayerInfo; //<priority,<<values>,operator>>
■
■float m_ColorDefaults[4];
■
■void PPEMatClassParameterColor(int mat_idx, int parameter_idx, PPEClassBase parent)
■{
■■m_LayerInfo = new map<int,ref ColorValuesData>;
■}
■
■override void InitDefaults()
■{
■■m_Color = PPETemplateDefColor.Cast(m_Defaults);
■■m_ColorDefaults = {m_Color.param2,m_Color.param3,m_Color.param4,m_Color.param5};
■}
■
```

```
// ----------------------------------------------
// ------ PPEMatClassParameterCommandData.c - START --------------------
// ----------------------------------------------


typedef map<int,ref PPERequestParamDataBase> ActiveParameterRequestsMap; //<request_ID, data>

class PPEMatClassParameterCommandData
{
■//default layer constants, complex data like colors PPEMatClassParameterColor are handled separately
■const int LAYER_INFO_VALUE = 0;
■const int LAYER_INFO_OPERATOR = 1;
■
■ref array<int> ■■■■■■■■■m_CommandLayersArray; //for tracking active priorities and sorting them //
■protected int ■■■■■■■■■m_UpdatedCount;
■protected int ■■■■■■■■■m_MaterialIndex; //just a helper
■protected int ■■■■■■■■■m_ParameterIndex;
■protected ref ActiveParameterRequestsMap ■■m_RequestMap;//<request_ID, parameter data>
■protected PPEClassBase ■■■■■■■m_Parent;
■protected ref Param ■■■■■■■m_Defaults; // Careful, formating is such, that param1 is ALWAYS string
■protected ref Param ■■■■■■■m_CurrentValues; // Careful, only actual values, WITHOUT string
■
■protected ref map<int,ref array<int>> ■■■m_Dependencies;
■
■void PPEMatClassParameterCommandData(int mat_idx, int parameter_idx, PPEClassBase parent)
■{
■■m_MaterialIndex = mat_idx;
■■m_ParameterIndex = parameter_idx;
■■m_Parent = parent;
■■
■■m_CommandLayersArray = new array<int>;
■■m_UpdatedCount = 0;
■■m_RequestMap = new ActiveParameterRequestsMap;
■}
■
■int GetParameterVarType()
■{
■■return -1;
■}
■
■void SetMaterialIndex(int value)
■{
■■m_MaterialIndex = value;
■}
■
■void SetParameterIndex(int value)
■{
■■m_ParameterIndex = value;
■}
■
■void SetParent(PPEClassBase parent)
■{
■■m_Parent = parent;
■}
■
■void InsertRequestData(PPERequestParamDataBase request_data)
■{
■■m_RequestMap.Set(request_data.GetRequesterIDX(),request_data);//<request_ID, data>
■}
■
■void Update(float timeslice, out Param p_total, out bool setting_defaults, int order)
■{
■■//insert dependencies to another update round
```

```
// ---------------------------------------------
// ------ PPEMatClassParameterFloat.c - START --------------------
// ---------------------------------------------


class PPEMatClassParameterFloat extends PPEMatClassParameterCommandData
{
■protected ref map<int,ref array<float,int>> m_LayerInfo; //<priority,<value,operator>>
■
■protected PPETemplateDefFloat m_Float;
■protected float m_ValueDefault;
■protected float m_ValueMin;
■protected float m_ValueMax;
■
■void PPEMatClassParameterFloat(int mat_idx, int parameter_idx, PPEClassBase parent)
■{
■■m_LayerInfo = new map<int,ref array<float,int>>;
■}
■
■override void InitDefaults()
■{
■■Class.CastTo(m_Float,m_Defaults);
■■//m_Float = PPETemplateDefFloat.Cast(m_Defaults);
■■m_ValueDefault = m_Float.param2;
■■m_ValueMin = m_Float.param3;
■■m_ValueMax = m_Float.param4;
■}
■
■override void InitCuttent()
■{
■■m_CurrentValues = new Param1<float>(m_ValueDefault);
■}
■
■override int GetParameterVarType()
■{
■■return PPEConstants.VAR_TYPE_FLOAT;
■}
■
■override void Update(float timeslice, out Param p_total, out bool setting_defaults, int order)
■{
■■super.Update(timeslice,p_total,setting_defaults,order);
■■
■■protected int active_request_count = 0;
■■
■■protected PPERequestParamDataFloat req_data;
■■
■■protected bool setting_value_zero = false;
■■
■■protected float float_value_temp = 0.0;
■■protected float float_value_default = 0.0;
■■protected float float_value_total = 0.0;
■■
■■if (p_total == null)
■■{
■■■p_total = new Param1<float>(0.0);
■■}
■■
■■if (m_RequestMap.Count() > 0)
■■{
■■■m_LayerInfo.Clear();
■■}
■■else
■■{
```

```
// ---------------------------------------------
// ------ PPEMatClassParameterInt.c - START --------------------
// ---------------------------------------------


class PPEMatClassParameterInt extends PPEMatClassParameterCommandData
{
■protected ref map<int,ref array<int,int>> m_LayerInfo; //<priority,<value,operator>>
■
■protected PPETemplateDefInt m_Int;
■protected int m_ValueDefault;
■
■void PPEMatClassParameterInt(int mat_idx, int parameter_idx, PPEClassBase parent)
■{
■■m_LayerInfo = new map<int,ref array<int,int>>;
■}
■
■override void InitDefaults()
■{
■■Class.CastTo(m_Int,m_Defaults);
■■m_ValueDefault = m_Int.param2;
■}
■
■override void InitCuttent()
■{
■■m_CurrentValues = new Param1<int>(m_ValueDefault);
■}
■
■override int GetParameterVarType()
■{
■■return PPEConstants.VAR_TYPE_INT;
■}
■
■override void Update(float timeslice, out Param p_total, out bool setting_defaults, int order)
■{
■■super.Update(timeslice,p_total,setting_defaults,order);
■■
■■protected int active_request_count = 0;
■■
■■protected PPERequestParamDataInt req_data;
■■
■■protected bool setting_value_zero = false;
■■
■■protected bool int_value_temp = false;
■■protected bool int_value_total = m_ValueDefault;
■■
■■if (p_total == null)
■■{
■■■p_total = new Param1<int>(0);
■■}
■■
■■if (m_RequestMap.Count() > 0)
■■{
■■■m_LayerInfo.Clear();
■■}
■■else
■■{
■■■//DbgPrnt("m_RequestMap.Count() is zero! Using default values. | mat/par: " + m_MaterialIndex + "/"
■■■SetParameterValueDefault(p_total);
■■■m_Parent.ParamUpdateRemove(m_ParameterIndex);
■■■return;
■■}
■■
```

```
// ---------------------------------------------
// ------ PPEMatClassParameterResource.c - START --------------------
// ---------------------------------------------

class PPEMatClassParameterResource extends PPEMatClassParameterCommandData
{
■override int GetParameterVarType()
■{
■■return PPEConstants.VAR_TYPE_RESOURCE;
■}
}



// ---------------------------------------------
// ------ PPEMatClassParameterResource.c - END ----------------------
// ---------------------------------------------


// ---------------------------------------------
// ------ PPEMatClassParameterTexture.c - START --------------------
// ---------------------------------------------

class PPEMatClassParameterTexture extends PPEMatClassParameterCommandData
{
■override int GetParameterVarType()
■{
■■return PPEConstants.VAR_TYPE_TEXTURE;
■}
}



// ---------------------------------------------
// ------ PPEMatClassParameterTexture.c - END ----------------------
// ---------------------------------------------


// ---------------------------------------------
// ------ PPEMatClassParameterVector.c - START --------------------
// ---------------------------------------------

class PPEMatClassParameterVector extends PPEMatClassParameterCommandData
{
■override int GetParameterVarType()
■{
■■return PPEConstants.VAR_TYPE_VECTOR;
■}
}



// ---------------------------------------------
// ------ PPEMatClassParameterVector.c - END ----------------------
// ---------------------------------------------
// ---------------------------------------------
// ------ PPEMedian.c - START --------------------
// ---------------------------------------------


//!Median - PostProcessEffectType.Median
//unused/not modifiable from script?
class PPEMedian: PPEClassBase
{
■static const int PARAM_KERNEL = 0;
■static const int PARAM_METHOD = 1;
■
■override int GetPostProcessEffectID()
■{
■■return PostProcessEffectType.Median;
■}
■
■override string GetDefaultMaterialPath()
■{
■■return "Graphics/Materials/postprocess/median";
■}
■
■override void RegisterMaterialParameters()
■{
■■RegisterParameterScalarInt(PARAM_KERNEL,"Kernel",3,3,5);
■■RegisterParameterScalarInt(PARAM_METHOD,"Method",2,0,2);
■}
}


// ---------------------------------------------
// ------ PPEMedian.c - END ----------------------
// ---------------------------------------------


// ---------------------------------------------
// ------ PPENone.c - START --------------------
// ---------------------------------------------


//!Dummy class - PostProcessEffectType.None
class PPENone: PPEClassBase
{
■override int GetPostProcessEffectID()
■{
■■return PostProcessEffectType.None;
■}
■
■override string GetDefaultMaterialPath()
■{
■■return "";
■}
};


// ---------------------------------------------
// ------ PPENone.c - END ----------------------
// ---------------------------------------------
// ---------------------------------------------
// ------ PPERadialBlur.c - START --------------------
// ---------------------------------------------


//!RadialBlur - PostProcessEffectType.RadialBlur
class PPERadialBlur: PPEClassBase
{
■static const int PARAM_POWERX = 0;
■static const int PARAM_POWERY = 1;
■static const int PARAM_OFFSETX = 2;
■static const int PARAM_OFFSETY = 3;
■static const int PARAM_PIXELSCALE = 4;
■
■
■static const int L_0_PAIN_BLUR = 100;
■
■
■override int GetPostProcessEffectID()
■{
■■return PostProcessEffectType.RadialBlur;
■}
■
■override string GetDefaultMaterialPath()
■{
■■return "Graphics/Materials/postprocess/radialblur";
■}
■
■override void RegisterMaterialParameters()
■{
■■RegisterParameterScalarFloat(PARAM_POWERX,"PowerX",0.0,0.0,0.1);
■■RegisterParameterScalarFloat(PARAM_POWERY,"PowerY",0.0,0.0,0.1);
■■RegisterParameterScalarFloat(PARAM_OFFSETX,"OffsetX",0.05,0.0,0.5);
■■RegisterParameterScalarFloat(PARAM_OFFSETY,"OffsetY",0.05,0.0,0.5);
■■RegisterParameterScalarFloat(PARAM_PIXELSCALE,"PixelScale",0.5,0.125,1.0);
■}
}


// ---------------------------------------------
// ------ PPERadialBlur.c - END ----------------------
// ---------------------------------------------


// ---------------------------------------------
// ------ PPERain.c - START --------------------
// ---------------------------------------------


//!Rain - PostProcessEffectType.Rain
class PPERain: PPEClassBase
{
■static const int PARAM_DROPDISTANCE = 0;
■static const int PARAM_DROPSIZEX = 1;
■static const int PARAM_DROPSIZEY = 2;
■static const int PARAM_RAINFOGGINESS = 3;
■static const int PARAM_RAINDENSITY = 4;
■static const int PARAM_FOGCOLORMULT = 5;
■static const int PARAM_BCKGRNDCOLORMULT = 6;
■static const int PARAM_REFRACTIONSCALE = 7;
■static const int PARAM_SUNVISIBLE = 8;
■static const int PARAM_GODRAYINTNEAR = 9;
■static const int PARAM_GODRAYINTFAR = 10;
```

```
// ---------------------------------------------
// ------ PPERBloodLoss.c - START --------------------
// ---------------------------------------------


class PPERequester_BloodLoss extends PPERequester_GameplayBase
{
■void SetBloodLossLevel(float level)
■{
■■SetTargetValueFloat(PostProcessEffectType.Glow,PPEGlow.PARAM_SATURATION,true,level,PPEGl
■}
}


// ---------------------------------------------
// ------ PPERBloodLoss.c - END ----------------------
// ---------------------------------------------


// ---------------------------------------------
// ------ PPERBurlapSack.c - START --------------------
// ---------------------------------------------


class PPERequester_BurlapSackEffects extends PPERequester_GameplayBase
{
■override protected void OnStart(Param par = null)
■{
■■super.OnStart();
■■
■■SetTargetValueFloat(PPEExceptions.EXPOSURE,PPEExposureNative.PARAM_INTENSITY,false,-10
■■SetTargetValueFloat(PPEExceptions.EYEACCOM,PPEEyeAccomodationNative.PARAM_INTENSITY,
■■SetTargetValueFloat(PostProcessEffectType.Glow,PPEGlow.PARAM_VIGNETTE,false,2.0,PPEGlow.
■■SetTargetValueColor(PostProcessEffectType.Glow,PPEGlow.PARAM_VIGNETTECOLOR,{0.0,0.0,0.0
■■
■■if (GetGame() && GetGame().GetMission() && GetGame().GetMission().GetEffectWidgets())
■■{
■■■GetGame().GetMission().GetEffectWidgets().AddSuspendRequest(EffectWidgetSuspends.BURLAPS
■■}
■}
■
■override protected void OnStop(Param par = null)
■{
■■super.OnStop();
■■
■■if (GetGame() && GetGame().GetMission() && GetGame().GetMission().GetEffectWidgets())
■■{
■■■GetGame().GetMission().GetEffectWidgets().RemoveSuspendRequest(EffectWidgetSuspends.BURL
■■}
■}
}


// ---------------------------------------------
// ------ PPERBurlapSack.c - END ----------------------
// ---------------------------------------------
// ---------------------------------------------
// ------ PPERCameraADS_Opt.c - START --------------------
// ---------------------------------------------


class PPERequester_CameraADS extends PPERequester_GameplayBase
{
■void SetValuesOptics(out array<float> mask_array, out array<float> lens_array, float gauss = 0.0)
■{
■■//mask
■■GetGame().ResetPPMask();
■■if (mask_array.Count() != 4)
■■{
■■■mask_array = {0.0,0.0,0.0,0.0};
■■}
■■GetGame().AddPPMask(mask_array[0], mask_array[1], mask_array[2], mask_array[3]);
■■
■■//lens
■■if (lens_array.Count() != 4)
■■{
■■■lens_array = {0.0,0.0,0.0,0.0};
■■}
■■SetTargetValueFloat(PostProcessEffectType.Glow,PPEGlow.PARAM_LENSDISTORT,false,lens_array
■■SetTargetValueFloat(PostProcessEffectType.Glow,PPEGlow.PARAM_MAXCHROMABBERATION,fals
■■SetTargetValueFloat(PostProcessEffectType.Glow,PPEGlow.PARAM_LENSCENTERX,false,lens_arra
■■SetTargetValueFloat(PostProcessEffectType.Glow,PPEGlow.PARAM_LENSCENTERY,false,lens_arra
■■
■■//DOF - no DOF in optics
■■SetTargetValueBoolDefault(PPEExceptions.DOF,PPEDOF.PARAM_ENABLE);
■■
■■//blur
■■SetTargetValueFloat(PostProcessEffectType.GaussFilter,PPEGaussFilter.PARAM_INTENSITY,false,g
■}
■
■void SetValuesIronsights(out array<float> DOF_array)
■{
■■//mask - no mask
■■GetGame().ResetPPMask();
■■
■■//lens - no lens
■■SetTargetValueFloatDefault(PostProcessEffectType.Glow,PPEGlow.PARAM_LENSDISTORT);
■■SetTargetValueFloatDefault(PostProcessEffectType.Glow,PPEGlow.PARAM_MAXCHROMABBERATI
■■SetTargetValueFloatDefault(PostProcessEffectType.Glow,PPEGlow.PARAM_LENSCENTERX);
■■SetTargetValueFloatDefault(PostProcessEffectType.Glow,PPEGlow.PARAM_LENSCENTERY);
■■
■■//DOF
■■SetTargetValueBool(PPEExceptions.DOF,PPEDOF.PARAM_ENABLE,DOF_array[0],PPEDOF.L_0_A
■■SetTargetValueFloat(PPEExceptions.DOF,PPEDOF.PARAM_FOCUS_DIST,false,DOF_array[1],PPED
■■SetTargetValueFloat(PPEExceptions.DOF,PPEDOF.PARAM_FOCUS_LEN,false,DOF_array[2],PPED
■■SetTargetValueFloat(PPEExceptions.DOF,PPEDOF.PARAM_FOCUS_LEN_NEAR,false,DOF_array[3
■■SetTargetValueFloat(PPEExceptions.DOF,PPEDOF.PARAM_BLUR,false,DOF_array[4],PPEDOF.L_4
■■SetTargetValueFloat(PPEExceptions.DOF,PPEDOF.PARAM_FOCUS_DEPTH_OFFSET,false,DOF_a
■■
■■//blur - no gauss blur (DOF bokeh only)
■■SetTargetValueFloatDefault(PostProcessEffectType.GaussFilter,PPEGaussFilter.PARAM_INTENSITY
■}
■
■override protected void OnStop(Param par = null)
■{
■■super.OnStop(par);
■■
■■if ( !GetGame() )
■■■return;
```

```
// ---------------------------------------------
// ------ PPERCameraNV.c - START --------------------
// ---------------------------------------------


class PPERequester_CameraNV extends PPERequester_GameplayBase
{
■static const int NV_NO_BATTERY= 0; //darkness
■static const int NV_DEFAULT_OPTICS = 1;
■static const int NV_DEFAULT_GLASSES = 2;
■static const int NV_PUMPKIN = 3;
■static const int NV_TRANSITIVE = 4; //resets EV during camera transitions
■static const int NV_DAYTIME_OPTICS = 5;
■
■protected int m_CurrentMode;
■protected float m_UGExposureCoef = 1.0;
■
■protected void SetNVMode(int mode)
■{
■■SetTargetValueFloat(PPEExceptions.EYEACCOM,PPEEyeAccomodationNative.PARAM_INTENSITY,
■■
■■switch (mode)
■■{
■■■case NV_NO_BATTERY: //battery off
■■■■SetTargetValueColor(PostProcessEffectType.Glow,PPEGlow.PARAM_COLORIZATIONCOLOR,{0
■■■■SetTargetValueFloatDefault(PPEExceptions.EXPOSURE,PPEExposureNative.PARAM_INTENSIT
■■■■SetTargetValueFloatDefault(PostProcessEffectType.FilmGrain,PPEFilmGrain.PARAM_SHARPNES
■■■■SetTargetValueFloatDefault(PostProcessEffectType.FilmGrain,PPEFilmGrain.PARAM_GRAINSIZE
■■■■SetTargetValueFloat(PPEExceptions.NVLIGHTPARAMS,PPELightIntensityParamsNative.PARAM_
■■■■SetTargetValueFloat(PPEExceptions.NVLIGHTPARAMS,PPELightIntensityParamsNative.PARAM_
■■■break;
■■■
■■■case NV_DEFAULT_OPTICS: //NV optic on
■■■■SetTargetValueColor(PostProcessEffectType.Glow,PPEGlow.PARAM_COLORIZATIONCOLOR,{0
■■■■SetTargetValueFloat(PPEExceptions.EXPOSURE,PPEExposureNative.PARAM_INTENSITY,false,
■■■■SetTargetValueFloat(PostProcessEffectType.FilmGrain,PPEFilmGrain.PARAM_SHARPNESS,false
■■■■SetTargetValueFloat(PostProcessEffectType.FilmGrain,PPEFilmGrain.PARAM_GRAINSIZE,false,1
■■■■SetTargetValueFloat(PPEExceptions.NVLIGHTPARAMS,PPELightIntensityParamsNative.PARAM_
■■■■SetTargetValueFloat(PPEExceptions.NVLIGHTPARAMS,PPELightIntensityParamsNative.PARAM_
■■■break;
■■■
■■■case NV_DAYTIME_OPTICS: //NV optic in daytime mode
■■■■SetTargetValueColor(PostProcessEffectType.Glow,PPEGlow.PARAM_COLORIZATIONCOLOR,{0
■■■■SetTargetValueFloat(PPEExceptions.EXPOSURE,PPEExposureNative.PARAM_INTENSITY,false,
■■■■SetTargetValueFloat(PostProcessEffectType.FilmGrain,PPEFilmGrain.PARAM_SHARPNESS,false
■■■■SetTargetValueFloat(PostProcessEffectType.FilmGrain,PPEFilmGrain.PARAM_GRAINSIZE,false,1
■■■■SetTargetValueFloat(PPEExceptions.NVLIGHTPARAMS,PPELightIntensityParamsNative.PARAM_
■■■■SetTargetValueFloat(PPEExceptions.NVLIGHTPARAMS,PPELightIntensityParamsNative.PARAM_
■■■break;
■■■
■■■case NV_DEFAULT_GLASSES: //goggles on
■■■■SetTargetValueColor(PostProcessEffectType.Glow,PPEGlow.PARAM_COLORIZATIONCOLOR,{0
■■■■SetTargetValueFloat(PPEExceptions.EXPOSURE,PPEExposureNative.PARAM_INTENSITY,false,
■■■■SetTargetValueFloat(PostProcessEffectType.FilmGrain,PPEFilmGrain.PARAM_SHARPNESS,false
■■■■SetTargetValueFloat(PostProcessEffectType.FilmGrain,PPEFilmGrain.PARAM_GRAINSIZE,false,1
■■■■SetTargetValueFloat(PPEExceptions.NVLIGHTPARAMS,PPELightIntensityParamsNative.PARAM_
■■■■SetTargetValueFloat(PPEExceptions.NVLIGHTPARAMS,PPELightIntensityParamsNative.PARAM_
■■■break;
■■■
■■■case NV_PUMPKIN: //pumpkin-o-vision
■■■■SetTargetValueColor(PostProcessEffectType.Glow,PPEGlow.PARAM_COLORIZATIONCOLOR,{1
■■■■SetTargetValueFloat(PPEExceptions.EXPOSURE,PPEExposureNative.PARAM_INTENSITY,false,
■■■■SetTargetValueFloatDefault(PostProcessEffectType.FilmGrain,PPEFilmGrain.PARAM_SHARPNES
// ---------------------------------------------
// ------ PPERContaminated.c - START --------------------
// ---------------------------------------------


// PPE when player is in contaminated area trigger
class PPERequester_ContaminatedAreaTint extends PPERequester_GameplayBase
{
■protected vector m_StartRGB = vector.Zero;
■protected float m_AccumulatedTime = 0;
■protected bool m_FadeIn = false;
■protected bool m_FadeOut = false;
■
■const float FADE_TIME = 3;
■// the end result is 1 - the value set here
■const float R_TARGET = 0.142; // end value 0.858
■const float G_TARGET = 0.15; // end value 0.850
■const float B_TARGET = 0.44; // end value 0.560
■
■override protected void OnStart( Param par = null )
■{
■■super.OnStart( par );
■■
■■m_AccumulatedTime = 0;
■■
■■m_FadeIn = true;
■■m_FadeOut = false;
■■
■■SetTargetValueFloat(PostProcessEffectType.FilmGrain,PPEFilmGrain.PARAM_SHARPNESS,false,10
■■SetTargetValueFloat(PostProcessEffectType.FilmGrain,PPEFilmGrain.PARAM_GRAINSIZE,false,1.0,
■}
■
■override protected void OnUpdate( float delta )
■{
■■super.OnUpdate( delta );
■■
■■if ( m_FadeIn && m_AccumulatedTime <= FADE_TIME )
■■{
■■■m_AccumulatedTime += delta;
■■
■■■m_StartRGB[0] = 1 - FadeColourMult( 0, 1, m_AccumulatedTime / FADE_TIME ) * R_TARGET;
■■■m_StartRGB[1] = 1 - FadeColourMult( 0, 1, m_AccumulatedTime / FADE_TIME ) * G_TARGET;
■■■m_StartRGB[2] = 1 - FadeColourMult( 0, 1, m_AccumulatedTime / FADE_TIME ) * B_TARGET;
■■■
■■■SetTargetValueColor(PostProcessEffectType.Glow,PPEGlow.PARAM_COLORIZATIONCOLOR,{m_
■■}
■■
■■if ( m_FadeOut )
■■{
■■■if (m_AccumulatedTime <= FADE_TIME)
■■■{
■■■■m_AccumulatedTime += delta;
■■■
■■■■m_StartRGB[0] = ( 1 - R_TARGET ) + FadeColourMult( 0, R_TARGET, m_AccumulatedTime / FAD
■■■■m_StartRGB[1] = ( 1 - G_TARGET ) + FadeColourMult( 0, G_TARGET, m_AccumulatedTime / FAD
■■■■m_StartRGB[2] = ( 1 - B_TARGET ) + FadeColourMult( 0, B_TARGET, m_AccumulatedTime / FAD
■■■■
■■■■SetTargetValueColor(PostProcessEffectType.Glow,PPEGlow.PARAM_COLORIZATIONCOLOR,{m
■■■}
■■■else
■■■{
■■■■Stop(); //proper termination after a fadeout
■■■}
```

```
// ---------------------------------------------
// ------ PPERControllerDisconnectBlur.c - START --------------------
// ---------------------------------------------


class PPERequester_ControllerDisconnectBlur extends PPERequester_MenuBase
{
■override protected void OnStart(Param par = null)
■{
■■super.OnStart();
■■
■■SetTargetValueFloat(PostProcessEffectType.GaussFilter,PPEGaussFilter.PARAM_INTENSITY,false,1
■}
}


// ---------------------------------------------
// ------ PPERControllerDisconnectBlur.c - END ----------------------
// ---------------------------------------------


// ---------------------------------------------
// ------ PPERControlsBlur.c - START --------------------
// ---------------------------------------------


class PPERequester_ControlsBlur extends PPERequester_MenuBase
{
■protected float m_Gauss;
■
■override protected void OnStart(Param par = null)
■{
■■super.OnStart();
■■
■■m_Gauss = Param1<float>.Cast(par).param1;
■■SetTargetValueFloat(PostProcessEffectType.GaussFilter,PPEGaussFilter.PARAM_INTENSITY,true,m
■}
}


// ---------------------------------------------
// ------ PPERControlsBlur.c - END ----------------------
// ---------------------------------------------


// ---------------------------------------------
// ------ PPERDeathDarkening.c - START --------------------
// ---------------------------------------------


class PPERequester_DeathDarkening extends PPERequester_GameplayBase
{
■protected float m_Value;
■
■override protected void OnStart(Param par = null)
■{
■■super.OnStart();
■■
■■m_Value = Param1<float>.Cast(par).param1;
■■//Print("PPERequester_DeathDarkening | level: " + m_Value);
■■m_Value = Easing.EaseInCubic(m_Value);
■■m_Value = Math.Lerp(0,-10,m_Value);
■■//Print("PPERequester_DeathDarkening | value: " + value);
```

```
// ----------------------------------------------
// ------ PPERDrowningEffect.c - START --------------------
// ----------------------------------------------


class PPERequester_Drowning extends PPERequester_GameplayBase
{
	private float m_EffectPhase = 0;
	private float m_Magnitude = 0;
	private float m_MinMagnitude = 0.3;
	private float m_MaxMagnitude = 0.3;												// Actual Maximum is Min+Max
	private float m_Frequency = 5;
	private float m_Stamina01;
	
	override protected void OnStart(Param par = null)
	{
		super.OnStart(par);
		
		m_EffectPhase = 0;
		m_Magnitude = 0;
	}
	
	override protected void OnUpdate(float delta)
	{			
		super.OnUpdate(delta);
		
		if (!m_IsRunning)
			return;
		
		m_EffectPhase += delta * m_Frequency / (m_Stamina01 + 1);	// Start with lower frequency and increa
		
		float currentVal = (Math.Sin(m_EffectPhase) +1)/2; 							// Makes current val oscillate between 0
		currentVal = Easing.EaseInExpo(currentVal);									// Ease it to tweak the effect
		currentVal *= currentVal * m_MaxMagnitude;									// Scale the normalized value to make it pr
		currentVal += m_MinMagnitude;
		m_Magnitude = Math.Lerp(m_Magnitude, currentVal, delta * m_Frequency);		// Learp to smooth the c
		
		SetTargetValueFloat(PostProcessEffectType.Glow,PPEGlow.PARAM_VIGNETTE,true, m_Magnitude,
		SetTargetValueColor(PostProcessEffectType.Glow,PPEGlow.PARAM_VIGNETTECOLOR,{0.0,0.025,
	}
	
	void SetStamina01(float stamina01)
	{
		m_Stamina01 = stamina01;
	}
}


// ----------------------------------------------
// ------ PPERDrowningEffect.c - END ----------------------
// ----------------------------------------------



// ----------------------------------------------
// ------ PPERequestData.c - START --------------------
// ----------------------------------------------


//! Data for one material parameter, requester side
class PPERequestParamDataBase
{
	PPERequesterBase m_Requester;
	protected bool m_UpdatingDataValues; //new values are being sent periodically
```

```
// ---------------------------------------------
// ------ PPERequesterBank.c - START --------------------
// ---------------------------------------------


/**
■\brief Requester bank contains all registered type instances as singletons. Creating new instances outsi
*/
class PPERequesterBank extends Managed
{
■private static ref map<int,ref PPERequesterBase> m_Instances;
■private static bool m_Initialized = false;
■
■static int m_lastID = -1;
■
■static const int INVALID■■■■■= -1;
■static const int REQ_INVENTORYBLUR■■■= RegisterRequester(PPERequester_InventoryBlur);
■static const int REQ_CONTROLLERDISCONNECT■= RegisterRequester(PPERequester_ControllerDis
■static const int REQ_GLASSESSPORTBLACK■■= RegisterRequester(PPERequester_GlassesSportBla
■static const int REQ_GLASSESSPORTBLUE■■= RegisterRequester(PPERequester_GlassesSportBlue
■static const int REQ_GLASSESSPORTGREEN■■= RegisterRequester(PPERequester_GlassesSportG
■static const int REQ_GLASSESSPORTORANGE■■= RegisterRequester(PPERequester_GlassesSport
■static const int REQ_GLASSESAVIATOR■■■= RegisterRequester(PPERequester_GlassesAviator);
■static const int REQ_GLASSESDESIGNER■■= RegisterRequester(PPERequester_GlassesDesignerBla
■static const int REQ_GLASSESTACTICAL■■= RegisterRequester(PPERequester_TacticalGoggles);
■static const int REQ_MOTOHELMETBLACK■■= RegisterRequester(PPERequester_MotoHelmetBlack)
■static const int REQ_GLASSESWELDING■■■= RegisterRequester(PPERequester_WeldingMask);
■static const int REQ_CAMERANV■■■■= RegisterRequester(PPERequester_CameraNV);
■static const int REQ_CAMERAADS■■■■= RegisterRequester(PPERequester_CameraADS);
■static const int REQ_BLOODLOSS■■■■= RegisterRequester(PPERequester_BloodLoss);
■static const int REQ_DEATHEFFECTS■■■= RegisterRequester(PPERequester_DeathDarkening);
■static const int REQ_UNCONEFFECTS■■■= RegisterRequester(PPERequester_UnconEffects);
■static const int REQ_TUNELVISSION■■■= RegisterRequester(PPERequester_TunnelVisionEffects);
■static const int REQ_BURLAPSACK■■■■= RegisterRequester(PPERequester_BurlapSackEffects);
■static const int REQ_INTROCHROMABB■■■= RegisterRequester(PPERequester_IntroChromAbb);
■static const int REQ_FEVEREFFECTS■■■= RegisterRequester(PPERequester_FeverEffects);
■static const int REQ_FLASHBANGEFFECTS■■= RegisterRequester(PPERequester_FlashbangEffects)
■static const int REQ_SHOCKHITEFFECTS■■= RegisterRequester(PPERequester_ShockHitReaction);
■static const int REQ_HEALTHHITEFFECTS■■= RegisterRequester(PPERequester_HealthHitReaction)
■static const int REQ_MENUEFFECTS■■■= RegisterRequester(PPERequester_MenuEffects);
■static const int REQ_CONTROLLERBLUR■■■= RegisterRequester(PPERequester_ControlsBlur);
■static const int REQ_SERVERBROWSEREFFECTS■= RegisterRequester(PPERequester_ServerBrows
■static const int REQ_TUTORIALEFFECTS■■= RegisterRequester(PPERequester_TutorialMenu);
■static const int REQ_CONTAMINATEDAREA■■= RegisterRequester(PPERequester_ContaminatedAre
■static const int REQ_SPOOKYAREA■■■■= RegisterRequester(PPERequester_SpookyAreaTint);
■static const int REQ_PAINBLUR■■■■= RegisterRequester(PPERequester_PainBlur);
■static const int REQ_UNDERGROUND■■■= RegisterRequester(PPERUndergroundAcco);
■static const int REQ_DROWNING■■■■= RegisterRequester(PPERequester_Drowning);
■
■private static ref PPERequesterRegistrations ■m_Registrations; //more registrations to be placed here
■
■static void Init()
■{
■■m_Registrations = new PPERequesterRegistrations;
■■
■■if (!m_Instances)
■■■m_Instances = new map<int,ref PPERequesterBase>;
■■
■■m_Initialized = true;
■}
■
■static void Cleanup()
■{
```

```
// ---------------------------------------------
// ------ PPERequestPlatformsBase.c - START --------------------
// ---------------------------------------------



class PPERequesterBase
{
■protected bool m_IsRunning; //helps determine if the requester has been stopped or not
■protected bool m_ValuesSent;
■protected int m_IDX;
■
■protected bool m_Valid = false;
■
■protected ref map<int,ref map<int,ref PPERequestParamDataBase>> m_RequestDataStructure; // <mat
■
■void PPERequesterBase(int requester_IDX)
■{
■■m_Valid = PPERequesterBank.VerifyRequester(this);
■■
■■m_IDX = requester_IDX;
■■m_ValuesSent = true;
■■m_IsRunning = false;
■■m_RequestDataStructure = new map<int,ref map<int,ref PPERequestParamDataBase>>;
■}
■
■//! Has to be set for the requester to be handled
■void SetRequesterUpdating(bool state)
■{
■■if (!m_Valid)
■■{
■■■Debug.Log("" + this + " not valid!");
■■■return;
■■}
■■
■■PPEManagerStatic.GetPPEManager().SetRequestUpdating(this,state);
■■
■■//TODO - separate into its own function?
■■if (state)
■■■PPEManagerStatic.GetPPEManager().SetRequestActive(this,true);
■}
■
■void Start(Param par = null)
■{
■■OnStart(par);
■■m_IsRunning = true;
■}
■
■void Stop(Param par = null)
■{
■■if (!m_RequestDataStructure || m_RequestDataStructure.Count() <= 0)
■■■return;
■■
■■OnStop(par);
■■m_IsRunning = false;
■}
■
■bool IsRequesterRunning()
■{
■■return m_IsRunning;
■}
■
■//! automatically assigned by PPERequesterBank
```

```
// ---------------------------------------------
// ------ PPERFever.c - START --------------------
// ---------------------------------------------


class PPERequester_FeverEffects extends PPERequester_GameplayBase
{
■void SetFeverIntensity(float intensity)
■{
■■SetTargetValueFloat(PostProcessEffectType.GaussFilter,PPEGaussFilter.PARAM_INTENSITY,true,in
■}
}


// ---------------------------------------------
// ------ PPERFever.c - END ----------------------
// ---------------------------------------------


// ---------------------------------------------
// ------ PPERFlashbangEffects.c - START --------------------
// ---------------------------------------------


class PPERequester_FlashbangEffects extends PPERequester_GameplayBase
{
■const float VAL_FACTOR = 0.85;
■const float EXPOSURE_MAX = 50;
■protected float m_Exposure;
■protected float m_Intensity;
■
■override protected void OnStart(Param par = null)
■{
■■if (!m_IsRunning)
■■{
■■■GetGame().GetMission().GetEffectWidgets().AddActiveEffects({EffectWidgetsTypes.COVER_FLASH
■■}
■■
■■super.OnStart();
■}
■
■override protected void OnStop(Param par = null)
■{
■■super.OnStop();
■■
■■m_Intensity = 0.0;
■■m_Exposure = 0.0;
■■
■■GetGame().GetMission().GetEffectWidgets().RemoveActiveEffects({EffectWidgetsTypes.COVER_FLA
■}
■
■override protected void OnUpdate( float delta )
■{
■■super.OnUpdate( delta );
■■
■■if ( GetGame() && GetGame().GetMission() && GetGame().GetMission().GetEffectWidgets() )
■■{
■■■Param1<float> par = new Param1<float>(1 - m_Intensity);
■■■GetGame().GetMission().GetEffectWidgets().UpdateWidgets(EffectWidgetsTypes.COVER_FLASHBA
■■}
■}
■
■void SetFlashbangIntensity(float intensity, float daytime, toggle)
```

```
// ---------------------------------------------
// ------ PPERGlasses.c - START --------------------
// ---------------------------------------------


//glasses and helmets
//! base, not to be used directly, would lead to layering collisions!
class PPERequester_GenericBlackGlassesBase extends PPERequester_GameplayBase
{
■override protected void OnStart(Param par = null)
■{
■■super.OnStart();
■■
■■SetTargetValueFloat(PostProcessEffectType.GodRays,PPEGodRays.PARAM_INTENSITY,true,0,PPE
■■SetTargetValueColor(PostProcessEffectType.Glow,PPEGlow.PARAM_COLORIZATIONCOLOR,{0.4,0
■}
}

class PPERequester_GlassesSportBlack extends PPERequester_GenericBlackGlassesBase{}
class PPERequester_GlassesDesignerBlack extends PPERequester_GenericBlackGlassesBase{}
class PPERequester_MotoHelmetBlack extends PPERequester_GenericBlackGlassesBase{}

class PPERequester_GlassesSportBlue extends PPERequester_GameplayBase
{
■override protected void OnStart(Param par = null)
■{
■■super.OnStart();
■■
■■SetTargetValueFloat(PostProcessEffectType.GodRays,PPEGodRays.PARAM_INTENSITY,true,0,PPE
■■SetTargetValueColor(PostProcessEffectType.Glow,PPEGlow.PARAM_COLORIZATIONCOLOR,{0.3,0
■}
}

class PPERequester_GlassesSportGreen extends PPERequester_GameplayBase■
{
■override protected void OnStart(Param par = null)
■{
■■super.OnStart();
■■
■■SetTargetValueFloat(PostProcessEffectType.GodRays,PPEGodRays.PARAM_INTENSITY,true,0,PPE
■■SetTargetValueColor(PostProcessEffectType.Glow,PPEGlow.PARAM_COLORIZATIONCOLOR,{0.2,0
■}
}

class PPERequester_GlassesSportOrange extends PPERequester_GameplayBase
{
■override protected void OnStart(Param par = null)
■{
■■super.OnStart();
■■
■■SetTargetValueFloat(PostProcessEffectType.GodRays,PPEGodRays.PARAM_INTENSITY,true,0,PPE
■■SetTargetValueColor(PostProcessEffectType.Glow,PPEGlow.PARAM_COLORIZATIONCOLOR,{0.0,0
■}
}

class PPERequester_GlassesAviator extends PPERequester_GameplayBase
{
■override protected void OnStart(Param par = null)
■{
■■super.OnStart();
■■
■■SetTargetValueFloat(PostProcessEffectType.GodRays,PPEGodRays.PARAM_INTENSITY,true,0,PPE
■■SetTargetValueColor(PostProcessEffectType.Glow,PPEGlow.PARAM_COLORIZATIONCOLOR,{0.3,0
// ---------------------------------------------
// ------ PPERHealthHit.c - START --------------------
// ---------------------------------------------


class PPERequester_HealthHitReaction extends PPERequester_GameplayBase
{
■const float INTENSITY_MULT = 6.0;
■
■void SetHitIntensity(float intensity)
■{
■■//color overlay
■■SetTargetValueFloat(PostProcessEffectType.Glow,PPEGlow.PARAM_OVERLAYFACTOR,true,0.05,P
■■SetTargetValueColor(PostProcessEffectType.Glow,PPEGlow.PARAM_OVERLAYCOLOR,{intensity * I
■}
}


// ---------------------------------------------
// ------ PPERHealthHit.c - END ----------------------
// ---------------------------------------------


// ---------------------------------------------
// ------ PPERIntroChromAbb.c - START --------------------
// ---------------------------------------------


class PPERequester_IntroChromAbb extends PPERequester_MenuBase
{
■override protected void OnStart(Param par = null)
■{
■■super.OnStart();
■■
■■SetTargetValueFloat(PostProcessEffectType.ChromAber,PPEChromAber.PARAM_POWERX,false,0.0
■■SetTargetValueFloat(PostProcessEffectType.ChromAber,PPEChromAber.PARAM_POWERY,false,0.0
■}
}


// ---------------------------------------------
// ------ PPERIntroChromAbb.c - END ----------------------
// ---------------------------------------------


// ---------------------------------------------
// ------ PPERInventoryBlur.c - START --------------------
// ---------------------------------------------


class PPERequester_InventoryBlur extends PPERequester_GameplayBase
{
■override protected void OnStart(Param par = null)
■{
■■super.OnStart();
■■SetTargetValueFloat(PostProcessEffectType.GaussFilter,PPEGaussFilter.PARAM_INTENSITY,false,0
■}
}


// ---------------------------------------------
// ------ PPERInventoryBlur.c - END ----------------------
// ---------------------------------------------
// ---------------------------------------------
// ------ PPERMenuEffects.c - START --------------------
// ---------------------------------------------


class PPERequester_MenuEffects extends PPERequester_MenuBase
{
■void SetBlurIntensity(float gauss)
■{
■■SetTargetValueFloat(PostProcessEffectType.GaussFilter,PPEGaussFilter.PARAM_INTENSITY,true,ga
■}
■
■void SetVignetteIntensity(float vignette)
■{
■■SetTargetValueFloat(PostProcessEffectType.Glow,PPEGlow.PARAM_VIGNETTE,false,vignette,PPEG
■■SetTargetValueColor(PostProcessEffectType.Glow,PPEGlow.PARAM_VIGNETTECOLOR,{0.0,0.0,0.0
■}
}


// ---------------------------------------------
// ------ PPERMenuEffects.c - END ----------------------
// ---------------------------------------------


// ---------------------------------------------
// ------ PPERotBlur.c - START --------------------
// ---------------------------------------------


//!Rotation Blur
class PPERotBlur: PPEClassBase
{
■static const int PARAM_POWER = 0;
■static const int PARAM_MINANGLEPERSEC = 1;
■static const int PARAM_MAXANGLEPERSEC = 2;
■static const int PARAM_MINDEPTH = 3;
■static const int PARAM_MAXDEPTH = 4;
■
■override int GetPostProcessEffectID()
■{
■■return PostProcessEffectType.RotBlur;
■}
■
■override string GetDefaultMaterialPath()
■{
■■return "Graphics/Materials/postprocess/rotblur";
■}
■
■override void RegisterMaterialParameters()
■{
■■RegisterParameterScalarFloat(PARAM_POWER,"Power",0.0,0.0,0.09);
■■RegisterParameterScalarFloat(PARAM_MINANGLEPERSEC,"MinAnglePerSec",10.0,0.0,180.0);
■■RegisterParameterScalarFloat(PARAM_MAXANGLEPERSEC,"MaxAnglePerSec",100.0,0.0,180.0);
■■RegisterParameterScalarFloat(PARAM_MINDEPTH,"MinDepth",2.5,0.0,5.0);
■■RegisterParameterScalarFloat(PARAM_MAXDEPTH,"MaxDepth",4.5,0.0,50.0);
■}
}


// ---------------------------------------------
// ------ PPERotBlur.c - END ----------------------
// ---------------------------------------------
// ---------------------------------------------
// ------ PPERPain.c - START --------------------
// ---------------------------------------------


class PPERequester_PainBlur extends PPERequester_GameplayBase
{
■void SetRadialBlur(float power_x, float power_y, float offset_x, float offset_y, float pixel_scale = 0.5 )
■{
■■SetTargetValueFloat(PostProcessEffectType.RadialBlur,PPERadialBlur.PARAM_POWERX,false,powe
■■SetTargetValueFloat(PostProcessEffectType.RadialBlur,PPERadialBlur.PARAM_POWERY,false,powe
■■SetTargetValueFloat(PostProcessEffectType.RadialBlur,PPERadialBlur.PARAM_OFFSETX,false,offse
■■SetTargetValueFloat(PostProcessEffectType.RadialBlur,PPERadialBlur.PARAM_OFFSETY,false,offse
■■SetTargetValueFloat(PostProcessEffectType.RadialBlur,PPERadialBlur.PARAM_PIXELSCALE,false,p
■}
}



// ---------------------------------------------
// ------ PPERPain.c - END ----------------------
// ---------------------------------------------


// ---------------------------------------------
// ------ PPERServerBrowser.c - START --------------------
// ---------------------------------------------


class PPERequester_ServerBrowserBlur extends PPERequester_MenuBase
{
■protected float m_Gauss;
■
■override protected void OnStart(Param par = null)
■{
■■super.OnStart();
■■
■■m_Gauss = Param1<float>.Cast(par).param1;
■■
■■SetTargetValueFloat(PostProcessEffectType.GaussFilter,PPEGaussFilter.PARAM_INTENSITY,true,m
■}
}


// ---------------------------------------------
// ------ PPERServerBrowser.c - END ----------------------
// ---------------------------------------------


// ---------------------------------------------
// ------ PPERShockHit.c - START --------------------
// ---------------------------------------------


class PPERequester_ShockHitReaction extends PPERequester_GameplayBase
{
■protected float m_Gauss;
■protected float m_VignetteIntensity;
■protected float m_OverlayFactor;
■protected ref array<float> m_OverlayColor;
■
■override protected void OnStart(Param par = null)
■{
```

```
// ---------------------------------------------
// ------ PPERSpooky.c - START --------------------
// ---------------------------------------------


// PPE when player is in contaminated area trigger
class PPERequester_SpookyAreaTint extends PPERequester_GameplayBase
{
■/*
■protected vector m_StartRGB = vector.Zero;
■protected float m_AccumulatedTime = 0;
■protected bool m_FadeIn = false;
■protected bool m_FadeOut = false;
■
■const float FADE_TIME = 6;
■// the end result is 1 - the value set here
■const float R_TARGET = 0.0;
■const float G_TARGET = 0.3;
■const float B_TARGET = 0.4;
■
■override protected void OnStart( Param par = null )
■{
■■super.OnStart( par );
■■
■■m_AccumulatedTime = 0;
■■
■■m_FadeIn = true;
■■m_FadeOut = false;
■■
■■//().GetWorld().LoadNewLightingCfg("$mission:lighting_halloween.txt");
■}
■
■override protected void OnUpdate( float delta )
■{
■■super.OnUpdate( delta );
■■
■■if ( m_FadeIn && m_AccumulatedTime <= FADE_TIME )
■■{
■■■m_AccumulatedTime += delta;
■■
■■■m_StartRGB[0] = 1 - FadeColourMult( 0, 1, m_AccumulatedTime / FADE_TIME ) * R_TARGET;
■■■m_StartRGB[1] = 1 - FadeColourMult( 0, 1, m_AccumulatedTime / FADE_TIME ) * G_TARGET;
■■■m_StartRGB[2] = 1 - FadeColourMult( 0, 1, m_AccumulatedTime / FADE_TIME ) * B_TARGET;
■■■
■■■SetTargetValueColor(PostProcessEffectType.Glow,PPEGlow.PARAM_COLORIZATIONCOLOR,{m_
■■}
■■
■■if ( m_FadeOut && m_AccumulatedTime <= FADE_TIME )
■■{
■■■m_AccumulatedTime += delta;
■■
■■■m_StartRGB[0] = ( 1 - R_TARGET ) + FadeColourMult( 0, R_TARGET, m_AccumulatedTime / FADE
■■■m_StartRGB[1] = ( 1 - G_TARGET ) + FadeColourMult( 0, G_TARGET, m_AccumulatedTime / FADE
■■■m_StartRGB[2] = ( 1 - B_TARGET ) + FadeColourMult( 0, B_TARGET, m_AccumulatedTime / FADE
■■■
■■■SetTargetValueColor(PostProcessEffectType.Glow,PPEGlow.PARAM_COLORIZATIONCOLOR,{m_
■■}
■}
■
■override void OnStop(Param par = null)
■{
■■m_FadeIn = false;
■■m_FadeOut = false;
```

```
// ---------------------------------------------
// ------ PPERTunnel.c - START --------------------
// ---------------------------------------------


class PPERequester_TunnelVisionEffects extends PPERequester_GameplayBase
{
■protected float m_Intensity;
■
■override protected void OnStart(Param par = null)
■{
■■super.OnStart();
■■
■■m_Intensity = Param1<float>.Cast(par).param1;
■■
■■SetTargetValueFloat(PostProcessEffectType.Glow,PPEGlow.PARAM_VIGNETTE,false,m_Intensity,PI
■■SetTargetValueColor(PostProcessEffectType.Glow,PPEGlow.PARAM_VIGNETTECOLOR,{0.0,0.0,0.0
■}
}


// ---------------------------------------------
// ------ PPERTunnel.c - END ----------------------
// ---------------------------------------------


// ---------------------------------------------
// ------ PPERTutorial.c - START --------------------
// ---------------------------------------------


class PPERequester_TutorialMenu extends PPERequester_MenuBase
{
■protected float m_Gauss;
■
■override protected void OnStart(Param par = null)
■{
■■super.OnStart();
■■
■■m_Gauss = Param1<float>.Cast(par).param1;
■■
■■SetTargetValueFloat(PostProcessEffectType.GaussFilter,PPEGaussFilter.PARAM_INTENSITY,true,m
■}
}


// ---------------------------------------------
// ------ PPERTutorial.c - END ----------------------
// ---------------------------------------------


// ---------------------------------------------
// ------ PPERUnconEffects.c - START --------------------
// ---------------------------------------------


class PPERequester_UnconEffects extends PPERequester_GameplayBase
{
■protected float m_Intensity;
■
■override protected void OnStart(Param par = null)
■{
■■super.OnStart();
```

```
// ---------------------------------------------
// ------ PPERUndergroundAcco.c - START --------------------
// ---------------------------------------------


class PPERUndergroundAcco extends PPERequester_GameplayBase
{
■void SetEyeAccommodation(float value)
■{
■■SetTargetValueFloat(PPEExceptions.EYEACCOM,PPEEyeAccomodationNative.PARAM_INTENSITY,
■}
}


// ---------------------------------------------
// ------ PPERUndergroundAcco.c - END ----------------------
// ---------------------------------------------


// ---------------------------------------------
// ------ PPESMAA.c - START --------------------
// ---------------------------------------------


//!SMAA - PostProcessEffectType.SMAA
class PPESMAA: PPEClassBase
{
■static const int PARAM_PRESET = 0;
■
■override int GetPostProcessEffectID()
■{
■■return PostProcessEffectType.SMAA;
■}
■
■override string GetDefaultMaterialPath()
■{
■■return "Graphics/Materials/postprocess/smaa";
■}
■
■override void RegisterMaterialParameters()
■{
■■RegisterParameterScalarInt(PARAM_PRESET,"Preset",0,0,3);
■}
}


// ---------------------------------------------
// ------ PPESMAA.c - END ----------------------
// ---------------------------------------------


// ---------------------------------------------
// ------ PPESSAO.c - START --------------------
// ---------------------------------------------


//!SSAO - PostProcessEffectType.SSAO
class PPESSAO: PPEClassBase
{
■static const int POSTPROCESS_OPTION_VALUE_LOW = 0;
■static const int POSTPROCESS_OPTION_VALUE_MEDIUM = 1;
■static const int POSTPROCESS_OPTION_VALUE_HIGH = 2;
■static const int POSTPROCESS_OPTION_VALUE_HIGHEST = 3;
```

```
// ---------------------------------------------
// ------ PPESunMask.c - START --------------------
// ---------------------------------------------


//!SunMask - PostProcessEffectType.SunMask
//A dummy class in script; 'SunMaskEffect' is used as a part of Rain and GodRays
class PPESunMask: PPEClassBase
{
■/*
■static const int PARAM_INTENSITY = 0;
■static const int PARAM_SUNSIZE = 1;
■static const int PARAM_VSTREAKINT = 2;
■static const int PARAM_DSTREAKINT = 3;
■static const int PARAM_SUNMASK = 4;
■*/
■override int GetPostProcessEffectID()
■{
■■return PostProcessEffectType.SunMask;
■}
■
■override string GetDefaultMaterialPath()
■{
■■return "";
■}
■
■/*override void RegisterMaterialParameters()
■{
■■RegisterParameterScalarFloat(PARAM_INTENSITY,"Intensity",0.8,0.0,1.0);
■■RegisterParameterScalarFloat(PARAM_SUNSIZE,"SunSize",0.2,0.0,1.0);
■■RegisterParameterScalarFloat(PARAM_VSTREAKINT,"VerticalStreakIntensity",0.75,0.0,5.0);
■■RegisterParameterScalarFloat(PARAM_DSTREAKINT,"DiagonalStreakIntensity",0.6,0.0,5.0);
■■//RegisterParameterResource(PARAM_SUNMASK,"SunMask","");
■}*/
}


// ---------------------------------------------
// ------ PPESunMask.c - END ----------------------
// ---------------------------------------------


// ---------------------------------------------
// ------ PPEUnderWater.c - START --------------------
// ---------------------------------------------


//!UnderWater - PostProcessEffectType.UnderWater
class PPEUnderWater: PPEClassBase
{
■static const int PARAM_RSINTENSITY = 0;
■static const int PARAM_RSPLANEDIST = 1;
■static const int PARAM_RSDEPTHDECREASE = 2;
■static const int PARAM_RSUVSCALE = 3;
■static const int PARAM_WATERPLANELUM = 4;
■
■override int GetPostProcessEffectID()
■{
■■return PostProcessEffectType.UnderWater;
■}
■
■override string GetDefaultMaterialPath()
■{
```

```
// ---------------------------------------------
// ------ PPEWetDistort.c - START --------------------
// ---------------------------------------------


//!WetDistort - PostProcessEffectType.WetDistort
class PPEWetDistort: PPEClassBase
{
■static const int PARAM_BLURPOWER = 0;
■static const int PARAM_LOCALBLURPOWER = 1;
■static const int PARAM_EFFPOWERTOP = 2;
■static const int PARAM_EFFPOWERBOTTOM = 3;
■static const int PARAM_BLURDOWNSIZE = 4;
■static const int PARAM_BLURGAUSS = 5;
■static const int PARAM_WAVSPDX1 = 6;
■static const int PARAM_WAVSPDX2 = 7;
■static const int PARAM_WAVSPDY1 = 8;
■static const int PARAM_WAVSPDY2 = 9;
■static const int PARAM_WAVEAMPX1 = 10;
■static const int PARAM_WAVEAMPX2 = 11;
■static const int PARAM_WAVEAMPY1 = 12;
■static const int PARAM_WAVEAMPY2 = 13;
■static const int PARAM_PHASERANDX = 14;
■static const int PARAM_PHASERANDY = 15;
■static const int PARAM_PHASEPOSX = 16;
■static const int PARAM_PHASEPOSY = 17;
■
■override int GetPostProcessEffectID()
■{
■■return PostProcessEffectType.WetDistort;
■}
■
■override string GetDefaultMaterialPath()
■{
■■return "Graphics/Materials/postprocess/wetdistort";
■}
■
■override void RegisterMaterialParameters()
■{
■■RegisterParameterScalarFloat(PARAM_BLURPOWER,"BlurPower",0.5,0.0,1.0);
■■RegisterParameterScalarFloat(PARAM_LOCALBLURPOWER,"LocalBlurPower",0.4,0.0,1.0);
■■RegisterParameterScalarFloat(PARAM_EFFPOWERTOP,"EffectPowerTop",0.0,0.0,1.0);
■■RegisterParameterScalarFloat(PARAM_EFFPOWERBOTTOM,"EffectPowerBottom",0.0,0.0,1.0);
■■RegisterParameterScalarInt(PARAM_BLURDOWNSIZE,"BlurDownSize",2,0,3);
■■RegisterParameterScalarInt(PARAM_BLURGAUSS,"BlurGauss",1,1,4);
■■RegisterParameterScalarFloat(PARAM_WAVSPDX1,"WaveSpeedX1",4.1,0.5,10.0);
■■RegisterParameterScalarFloat(PARAM_WAVSPDX2,"WaveSpeedX2",3.7,0.5,10.0);
■■RegisterParameterScalarFloat(PARAM_WAVSPDY1,"WaveSpeedY1",2.5,0.5,10.0);
■■RegisterParameterScalarFloat(PARAM_WAVSPDY2,"WaveSpeedY2",1.85,0.5,10.0);
■■RegisterParameterScalarFloat(PARAM_WAVEAMPX1,"WaveAmpX1",0.005,0.001,0.02);
■■RegisterParameterScalarFloat(PARAM_WAVEAMPX2,"WaveAmpX2",0.004,0.001,0.02);
■■RegisterParameterScalarFloat(PARAM_WAVEAMPY1,"WaveAmpY1",0.009,0.001,0.02);
■■RegisterParameterScalarFloat(PARAM_WAVEAMPY2,"WaveAmpY2",0.007,0.001,0.02);
■■RegisterParameterScalarFloat(PARAM_PHASERANDX,"PhaseRandX",0.5,0.0,1.0);
■■RegisterParameterScalarFloat(PARAM_PHASERANDY,"PhaseRandY",0.3,0.0,1.0);
■■RegisterParameterScalarFloat(PARAM_PHASEPOSX,"PhasePosX",10.0,0.0,20.0);
■■RegisterParameterScalarFloat(PARAM_PHASEPOSY,"PhasePosY",6.0,0.0,20.0);
■}
}


// ---------------------------------------------
// ------ PPEWetDistort.c - END ----------------------
// ---------------------------------------------
// ------ PrepareCarp.c - START --------------------
// ---------------------------------------------


class PrepareCarp extends PrepareFish
{■
■override void Init()
■{
■■m_Name = "#STR_gutandprepare0";
■■m_IsInstaRecipe = false;//should this recipe be performed instantly without animation
■■m_AnimationLength = 1.5;//animation length in relative time units
■■m_Specialty = 0.02;// value > 0 for roughness, value < 0 for precision
■■
■■
■■//conditions
■■m_MinDamageIngredient[0] = -1;//-1 = disable check
■■m_MaxDamageIngredient[0] = 3;//-1 = disable check
■■
■■m_MinQuantityIngredient[0] = -1;//-1 = disable check
■■m_MaxQuantityIngredient[0] = -1;//-1 = disable check
■■
■■m_MinDamageIngredient[1] = -1;//-1 = disable check
■■m_MaxDamageIngredient[1] = 3;//-1 = disable check
■■
■■m_MinQuantityIngredient[1] = -1;//-1 = disable check
■■m_MaxQuantityIngredient[1] = -1;//-1 = disable check
■■//----------------------------------------------------------------------------------------------------------------------
■■
■■//INGREDIENTS
■■//ingredient 1
■■InsertIngredient(0,"Carp");//you can insert multiple ingredients this way

■■m_IngredientAddHealth[0] = 0;// 0 = do nothing
■■m_IngredientSetHealth[0] = -1; // -1 = do nothing
■■m_IngredientAddQuantity[0] = 0;// 0 = do nothing
■■m_IngredientDestroy[0] = true;//true = destroy, false = do nothing
■■m_IngredientUseSoftSkills[0] = false;// set 'true' to allow modification of the values by softskills on this i
■■
■■//ingredient 2
■■InsertIngredient(1,"Sickle");//you can insert multiple ingredients this way
■■InsertIngredient(1,"KukriKnife");
■■InsertIngredient(1,"FangeKnife");
■■InsertIngredient(1,"Hacksaw");
■■InsertIngredient(1,"HandSaw");
■■InsertIngredient(1,"KitchenKnife");
■■InsertIngredient(1,"SteakKnife");
■■InsertIngredient(1,"StoneKnife");
■■InsertIngredient(1,"Cleaver");
■■InsertIngredient(1,"CombatKnife");
■■InsertIngredient(1,"HuntingKnife");
■■InsertIngredient(1,"Machete");
■■InsertIngredient(1,"CrudeMachete");
■■InsertIngredient(1,"OrientalMachete");
■■InsertIngredient(1,"WoodAxe");
■■InsertIngredient(1,"Hatchet");
■■InsertIngredient(1,"FirefighterAxe");
■■InsertIngredient(1,"Sword");
■■InsertIngredient(1,"AK_Bayonet");
■■InsertIngredient(1,"M9A1_Bayonet");
■■InsertIngredient(1,"SKS_Bayonet");
■■InsertIngredient(1,"BoneKnife");
■■InsertIngredient(1,"Screwdriver");
```

```
// ---------------------------------------------
// ------ PrepareChicken.c - START --------------------
// ---------------------------------------------


class PrepareChicken extends RecipeBase
{■
■override void Init()
■{
■■m_Name = "#skin";
■■m_IsInstaRecipe = false;//should this recipe be performed instantly without animation
■■m_AnimationLength = 1.5;//animation length in relative time units
■■m_Specialty = 0.02;// value > 0 for roughness, value < 0 for precision
■■
■■
■■//conditions
■■m_MinDamageIngredient[0] = -1;//-1 = disable check
■■m_MaxDamageIngredient[0] = 3;//-1 = disable check
■■
■■m_MinQuantityIngredient[0] = -1;//-1 = disable check
■■m_MaxQuantityIngredient[0] = -1;//-1 = disable check
■■
■■m_MinDamageIngredient[1] = -1;//-1 = disable check
■■m_MaxDamageIngredient[1] = 3;//-1 = disable check
■■
■■m_MinQuantityIngredient[1] = -1;//-1 = disable check
■■m_MaxQuantityIngredient[1] = -1;//-1 = disable check
■■//----------------------------------------------------------------------------------------------------------------------
■■
■■//INGREDIENTS
■■//ingredient 1
■■InsertIngredient(0,"DeadChicken_ColorBase");//you can insert multiple ingredients this way

■■m_IngredientAddHealth[0] = 0;// 0 = do nothing
■■m_IngredientSetHealth[0] = -1; // -1 = do nothing
■■m_IngredientAddQuantity[0] = 0;// 0 = do nothing
■■m_IngredientDestroy[0] = true;//true = destroy, false = do nothing
■■m_IngredientUseSoftSkills[0] = false;// set 'true' to allow modification of the values by softskills on this i
■■
■■//ingredient 2
■■InsertIngredient(1,"Sickle");//you can insert multiple ingredients this way
■■InsertIngredient(1,"KukriKnife");
■■InsertIngredient(1,"FangeKnife");
■■InsertIngredient(1,"Hacksaw");
■■InsertIngredient(1,"HandSaw");
■■InsertIngredient(1,"KitchenKnife");
■■InsertIngredient(1,"SteakKnife");
■■InsertIngredient(1,"StoneKnife");
■■InsertIngredient(1,"Cleaver");
■■InsertIngredient(1,"CombatKnife");
■■InsertIngredient(1,"HuntingKnife");
■■InsertIngredient(1,"Machete");
■■InsertIngredient(1,"CrudeMachete");
■■InsertIngredient(1,"OrientalMachete");
■■InsertIngredient(1,"WoodAxe");
■■InsertIngredient(1,"Hatchet");
■■InsertIngredient(1,"FirefighterAxe");
■■InsertIngredient(1,"Sword");
■■InsertIngredient(1,"AK_Bayonet");
■■InsertIngredient(1,"M9A1_Bayonet");
■■InsertIngredient(1,"SKS_Bayonet");
■■InsertIngredient(1,"BoneKnife");
■■InsertIngredient(1,"Screwdriver");
```

```
// ---------------------------------------------
// ------ PrepareFish.c - START --------------------
// ---------------------------------------------


class PrepareFish extends RecipeBase
{■
■override void Init()
■{
■■
■}

■override bool CanDo(ItemBase ingredients[], PlayerBase player)//final check for recipe's validity
■{
■■return true;
■}

■override void Do(ItemBase ingredients[], PlayerBase player,array<ItemBase> results, float specialty_wei
■{
■■ItemBase ingredient = ingredients[0];
■■
■■for (int i=0; i < results.Count(); i++)
■■{
■■■ItemBase item_result;
■■■Class.CastTo(item_result, results.Get(i));
■■■
■■■//Trasnfer current food state
■■■MiscGameplayFunctions.TransferItemProperties(ingredient, item_result);
■■■item_result.SetQuantityNormalized(Math.RandomFloat(0.8,1));
■■}
■■
■■PluginLifespan lifespan = PluginLifespan.Cast( GetPlugin( PluginLifespan ) );
■■lifespan.UpdateBloodyHandsVisibility( player, true );
■}
};



// ---------------------------------------------
// ------ PrepareFish.c - END ----------------------
// ---------------------------------------------


// ---------------------------------------------
// ------ PrepareMackerel.c - START --------------------
// ---------------------------------------------


class PrepareMackerel extends PrepareFish
{■
■override void Init()
■{
■■m_Name = "#STR_gutandprepare0";
■■m_IsInstaRecipe = false;//should this recipe be performed instantly without animation
■■m_AnimationLength = 1.5;//animation length in relative time units
■■m_Specialty = 0.02;// value > 0 for roughness, value < 0 for precision
■■
■■
■■//conditions
■■m_MinDamageIngredient[0] = -1;//-1 = disable check
■■m_MaxDamageIngredient[0] = 3;//-1 = disable check
■■
■■m_MinQuantityIngredient[0] = -1;//-1 = disable check
```

```
// ---------------------------------------------
// ------ PrepareRabbit.c - START --------------------
// ---------------------------------------------


class PrepareRabbit extends RecipeBase
{■
■override void Init()
■{
■■m_Name = "#skin";
■■m_IsInstaRecipe = false;//should this recipe be performed instantly without animation
■■m_AnimationLength = 1.5;//animation length in relative time units
■■m_Specialty = 0.02;// value > 0 for roughness, value < 0 for precision
■■
■■
■■//conditions
■■m_MinDamageIngredient[0] = -1;//-1 = disable check
■■m_MaxDamageIngredient[0] = 3;//-1 = disable check
■■
■■m_MinQuantityIngredient[0] = -1;//-1 = disable check
■■m_MaxQuantityIngredient[0] = -1;//-1 = disable check
■■
■■m_MinDamageIngredient[1] = -1;//-1 = disable check
■■m_MaxDamageIngredient[1] = 3;//-1 = disable check
■■
■■m_MinQuantityIngredient[1] = -1;//-1 = disable check
■■m_MaxQuantityIngredient[1] = -1;//-1 = disable check
■■//----------------------------------------------------------------------------------------------------------------------
■■
■■//INGREDIENTS
■■//ingredient 1
■■InsertIngredient(0,"DeadRabbit");//you can insert multiple ingredients this way

■■m_IngredientAddHealth[0] = 0;// 0 = do nothing
■■m_IngredientSetHealth[0] = -1; // -1 = do nothing
■■m_IngredientAddQuantity[0] = 0;// 0 = do nothing
■■m_IngredientDestroy[0] = true;//true = destroy, false = do nothing
■■m_IngredientUseSoftSkills[0] = false;// set 'true' to allow modification of the values by softskills on this i
■■
■■//ingredient 2
■■InsertIngredient(1,"Sickle");//you can insert multiple ingredients this way
■■InsertIngredient(1,"KukriKnife");
■■InsertIngredient(1,"FangeKnife");
■■InsertIngredient(1,"Hacksaw");
■■InsertIngredient(1,"HandSaw");
■■InsertIngredient(1,"KitchenKnife");
■■InsertIngredient(1,"SteakKnife");
■■InsertIngredient(1,"StoneKnife");
■■InsertIngredient(1,"Cleaver");
■■InsertIngredient(1,"CombatKnife");
■■InsertIngredient(1,"HuntingKnife");
■■InsertIngredient(1,"Machete");
■■InsertIngredient(1,"CrudeMachete");
■■InsertIngredient(1,"OrientalMachete");
■■InsertIngredient(1,"WoodAxe");
■■InsertIngredient(1,"Hatchet");
■■InsertIngredient(1,"FirefighterAxe");
■■InsertIngredient(1,"Sword");
■■InsertIngredient(1,"AK_Bayonet");
■■InsertIngredient(1,"M9A1_Bayonet");
■■InsertIngredient(1,"SKS_Bayonet");
■■InsertIngredient(1,"BoneKnife");
■■InsertIngredient(1,"Screwdriver");
```

```
// ---------------------------------------------
// ------ PresetHandlers.c - START --------------------
// ---------------------------------------------


#ifdef DEVELOPER
//---------------------------------------------------------
//Class names need to match the preset name + _Preset suffix
//---------------------------------------------------------


//TEMPLATE:
/*

class Farmer_Preset extends PresetSpawnBase
{
■override void OnPresetSpawn(PlayerBase player)
■{
■■//m_ItemsInventory -> contains all items in player inventory
■}
}

*/

//--------------------------------------------

class FreshSpawn_Preset extends PresetSpawnBase
{
■override void OnPresetSpawn(PlayerBase player)
■{
■■FindAndTakeToHandsFromInventory("Plum");
■}
}
//--------------------------------------------

class Farmer_Preset extends PresetSpawnBase
{
■override void OnPresetSpawn(PlayerBase player)
■{

■}
}
//--------------------------------------------
class AntiHazard_Preset extends PresetSpawnBase
{
■override void OnPresetSpawn(PlayerBase player)
■{
■■EntityAI mask = FindItem("AirborneMask");
■■
■■if (mask)
■■{
■■■mask.OnDebugSpawn();
■■}
■■
■}
}
//--------------------------------------------

class Update117_Preset extends PresetSpawnBase
{■
■vector randompos;
■override void OnPresetSpawn(PlayerBase player)
■{
```

```
// ---------------------------------------------
// ------ PresetsMenu.c - START --------------------
// ---------------------------------------------


class PresetsMenu extends UIScriptedMenu
{
■void PresetsMenu()
■{
■}

■void ~PresetsMenu()
■{
■}

■override Widget Init()
■{
■■layoutRoot = GetGame().GetWorkspace().CreateWidgets("gui/layouts/day_z_options_controls_preset.
■■
■■m_schemes_list = TextListboxWidget.Cast( layoutRoot.FindAnyWidget("PresetListboxWidget") );
■■
■■int c = GetGame().GetInput().GetProfilesCount();
■■for (int i = 0; i < c; i++)
■■{
■■■string tmp;
■■■GetGame().GetInput().GetProfileName(i, tmp);
■■■m_schemes_list.AddItem(tmp, NULL, 0);
■■}

■■i = GetGame().GetInput().GetCurrentProfile();
■■if (i >= 0 && i < c)
■■{
■■■m_schemes_list.SelectRow(i);
■■}

■■return layoutRoot;
■}

■override bool OnClick(Widget w, int x, int y, int button)
■{
■■super.OnClick(w, x, y, button);
■■
■■switch (w.GetUserID())
■■{
■■case IDC_CANCEL:
■■■Close();

■■■return true;

■■case IDC_OK:
■■■int index = m_schemes_list.GetSelectedRow();
■■■if (index != -1)
■■■{
■■■■GetGame().GetInput().SetProfile(index);
■■■■GetGame().GetMission().GetOnInputPresetChanged().Invoke();
■■■■Close();
■■■}
■■■
■■■return true;
■■}

■■return false;
■}
```

```
// ---------------------------------------------
// ------ PressVest_ColorBase.c - START --------------------
// ---------------------------------------------


class PressVest_ColorBase extends Clothing {};
class PressVest_Blue extends PressVest_ColorBase {};
class PressVest_LightBlue extends PressVest_ColorBase {};


// ---------------------------------------------
// ------ PressVest_ColorBase.c - END ----------------------
// ---------------------------------------------


// ---------------------------------------------
// ------ PrisonerCap.c - START --------------------
// ---------------------------------------------


class PrisonerCap extends Clothing
{
■override void SetActions()
■{
■■super.SetActions();
■■AddAction(ActionWringClothes);
■}
};



// ---------------------------------------------
// ------ PrisonerCap.c - END ----------------------
// ---------------------------------------------


// ---------------------------------------------
// ------ PrisonUniformJacket.c - START --------------------
// ---------------------------------------------


class PrisonUniformJacket extends Clothing
{
■override void SetActions()
■{
■■super.SetActions();
■■AddAction(ActionWringClothes);
■}
};


// ---------------------------------------------
// ------ PrisonUniformJacket.c - END ----------------------
// ---------------------------------------------


// ---------------------------------------------
// ------ PrisonUniformPants.c - START --------------------
// ---------------------------------------------


class PrisonUniformPants extends Clothing
{
```

```
// ---------------------------------------------
// ------ ProfileOptionsUI.c - START --------------------
// ---------------------------------------------


class ProfileOptionsUI extends ScriptedWidgetEventHandler
{
	void ~ProfileOptionsUI()
	{
		Deattach();
	}

	void Attach(int profileOption, array<Widget> widget_array)
	{
		m_profileOption = profileOption;
		m_widget_array = new array<Widget>;
		m_widget_array = widget_array;
			
		for (int i = 0; i < m_widget_array.Count(); i++)
		{
			m_widget = m_widget_array.Get(i);
			if (m_widget)
			{
				m_widget.SetHandler(this);
				
				Init();
			}
		}
	}
	
	void Deattach()
	{
		//m_profileOption = NULL;
		if (m_widget_array && m_widget_array.Count() > 0)
		{
			for (int i = 0; i < m_widget_array.Count(); i++)
			{
				m_widget = m_widget_array.Get(i);
				if (m_widget)
				{
					m_widget.SetHandler(NULL);
					m_widget = NULL;
				}
			}
		}
	}
	
	void Init()
	{
		int state;
		ButtonWidget button;
		
		if ( Class.CastTo(button, m_widget) )
		{
			state = m_widget_array.Find(m_widget);
			button.SetState( g_Game.GetProfileOption(m_profileOption) == state ); //works for bool values only!!
		}
	}

	void SetValue()
	{
		int state;
		ButtonWidget button;
```

```
// ---------------------------------------------
// ------ ProgressAsync.c - START --------------------
// ---------------------------------------------


class ProgressAsync
{
■//proto native void SetUserData(Widget inst);
■proto static native void SetUserData(Widget inst);
■proto static native void SetProgressData(Widget inst);
■proto static native void DestroyAllPendingProgresses();
■proto static native void StartProgress();
}


// ---------------------------------------------
// ------ ProgressAsync.c - END ----------------------
// ---------------------------------------------


// ---------------------------------------------
// ------ ProjectedCrosshair.c - START --------------------
// ---------------------------------------------


class ProjectedCrosshair extends ScriptedWidgetEventHandler
{
■protected Widget■■m_Root;
■protected vector■■m_Position;
■protected bool ■■■m_Visible;
■protected bool■■■m_Debug;
■
■protected PlayerBase■m_Player;
■protected Weapon_Base■m_Weapon;■■

■void ProjectedCrosshair()
■{
■■m_Player = NULL;
■■m_Weapon = NULL;
■■m_Visible = false;
■■m_Debug = false;

■■GetGame().GetUpdateQueue(CALL_CATEGORY_GUI).Insert(this.Update);
■}

■void ~ProjectedCrosshair()
■{
■■GetGame().GetUpdateQueue(CALL_CATEGORY_GUI).Remove(this.Update);
■}
■
■void OnWidgetScriptInit(Widget w)
■{
■■m_Root = w;
■■m_Root.SetHandler(this);
■■m_Root.Update();
■}
■
■//! Update
■protected void Update()
■{
#ifdef DIAG_DEVELOPER
■■m_Debug = DiagMenu.GetBool( DiagMenuIDs.WEAPON_DEBUG );
#endif
```

```
// ---------------------------------------------
// ------ PropertyModifiers.c - START --------------------
// ---------------------------------------------


class PropertyModifiers
{
■vector m_RecoilModifiers;
■vector m_SwayModifiers;
■vector m_SightMisalignment;
■
■float m_BarrelLength;
■//ref Timer test_timer;
■ItemBase m_OwnerItem;
■float m_Weapon
■
■void PropertyModifiers(ItemBase owner)
■{
■■//test_timer = new Timer();
■■//test_timer.Run(5, this, "Refresh", NULL, true);
■■m_OwnerItem = owner;
■■UpdateModifiers();
■■m_BarrelLength = CalculateBarrelLength(owner);
■}
■
■//! Get Barrel Legth in mm
■float GetBarrelLength()
■{
■■//return m_BarrelLength * 1000:
■■return CalculateBarrelLength(m_OwnerItem) * 1000;
■}
■
■float CalculateBarrelLength(ItemBase owner)
■{
■■vector usti_hlavne_position = owner.GetSelectionPositionLS( "usti hlavne" );//usti hlavne
■■vector konec_hlavne_position = owner.GetSelectionPositionLS( "konec hlavne" );//konec hlavne
■■usti_hlavne_position = owner.ModelToWorld(usti_hlavne_position);
■■konec_hlavne_position = owner.ModelToWorld(konec_hlavne_position);
■■return vector.Distance(usti_hlavne_position, konec_hlavne_position);
■}


■void UpdateModifiers()
■{
■■m_RecoilModifiers = GetModifierRaw(m_OwnerItem, "recoilModifier");
■■m_SwayModifiers = GetModifierRaw(m_OwnerItem, "swayModifier");
■■m_SightMisalignment = GetModifierRaw(m_OwnerItem, "sightMisalignmentModifier");
■■
■■for (int i = 0; i < m_OwnerItem.GetInventory().AttachmentCount(); i++)
■■{
■■■ItemBase attachment = ItemBase.Cast(m_OwnerItem.GetInventory().GetAttachmentFromIndex(i));
■■■
■■■vector temp_value = PropertyModifiers.GetModifierRaw(attachment, "recoilModifier");
■■■m_RecoilModifiers[0] = m_RecoilModifiers[0] * temp_value[0];
■■■m_RecoilModifiers[1] = m_RecoilModifiers[1] * temp_value[1];
■■■m_RecoilModifiers[2] = m_RecoilModifiers[2] * temp_value[2];
■■■
■■■temp_value = PropertyModifiers.GetModifierRaw(attachment, "swayModifier");
■■■m_SwayModifiers[0] = m_SwayModifiers[0] * temp_value[0];
■■■m_SwayModifiers[1] = m_SwayModifiers[1] * temp_value[1];
■■■m_SwayModifiers[2] = m_SwayModifiers[2] * temp_value[2];
■■■
■■■temp_value = PropertyModifiers.GetModifierRaw(attachment, "sightMisalignmentModifier");
```

```
// ---------------------------------------------
// ------ proto.c - START --------------------
// ---------------------------------------------


/** @file */


/*
Function/method modifiers:
proto■- prototyping of internal function (C++ side)
native - native call convention of internal function (C++ side)
volatile - internal function that may call back to script (hint for
  compiler that context need to be saved on stack)
private - function may not be called from script
event - hint for tools that the function should be exposed as
 Entity script event.

Variable modifiers:
owned - modifier for returing internal functions. Tells to script-VM,
that returning variable (string or array) must not be released
out - modifier for function parameters. It tells that variable will
 be changed by function call (used mainly by internal functions)
inout - modifier for function parameters. It tells that variable will
 be used and then changed by function call (used mainly by internal functions)

const - constants. May not be modified.
reference - hint for tools (Material editor), that the variable may be used
 as parameter in material
*/


/*===================================================================*/
/*■■■■■■■Enforce engine API■■■■■■ */
/*===================================================================*/

//placeholder
class AnimEvent
{
■int type;
■int slot;
};

class SoundEvent
{
■int type;
■int handle;
};


typedef int[] vobject;

class vobject
{
■proto native IEntitySource ToEntitySource();
}

#ifdef ENF_DONE

//-----------------------------------------
// SOUND API
//-----------------------------------------
//TODO:
```

```
// ---------------------------------------------
// ------ PsilocybeMushroom.c - START --------------------
// ---------------------------------------------


class PsilocybeMushroom : MushroomBase
{
}



// ---------------------------------------------
// ------ PsilocybeMushroom.c - END ----------------------
// ---------------------------------------------


// ---------------------------------------------
// ------ PSO11Optic.c - START --------------------
// ---------------------------------------------


class PSO11Optic extends ItemOptics
{
}


// ---------------------------------------------
// ------ PSO11Optic.c - END ----------------------
// ---------------------------------------------


// ---------------------------------------------
// ------ PSO1Optic.c - START --------------------
// ---------------------------------------------


class PSO1Optic extends ItemOptics
{
}


// ---------------------------------------------
// ------ PSO1Optic.c - END ----------------------
// ---------------------------------------------


// ---------------------------------------------
// ------ Pumpkin.c - START --------------------
// ---------------------------------------------


class Pumpkin : Edible_Base
{
■override bool CanBeCooked()
■{
■■return false;
■}■■
■
■override bool CanBeCookedOnStick()
■{
■■return false;
■}
■
```

```
// ---------------------------------------------
// ------ PumpkinHelmet.c - START --------------------
// ---------------------------------------------


// Have a spooky Halloween everyone!

class PumpkinHelmet : Clothing
{
■void PumpkinHelmet()
■{
■■SetEventMask(EntityEvent.INIT); // Enable EOnInit event
■}
■
■override bool CanPutAsAttachment( EntityAI parent )
■{
■■if(!super.CanPutAsAttachment(parent)) {return false;}
■■
■■Clothing eyewear = Clothing.Cast(parent.FindAttachmentBySlotName("Eyewear"));
■■if ( eyewear && eyewear.ConfigGetBool("isStrap") )
■■{
■■■return false;
■■}
■■
■■Clothing mask = Clothing.Cast(parent.FindAttachmentBySlotName("Mask"));
■■if ( mask && (mask.ConfigGetBool("noHelmet") && !HockeyMask.Cast(mask) && !SantasBeard.Cast(m
■■{
■■■return false;
■■}
■■
■■return true;
■}
■
■override void OnMovedInsideCargo(EntityAI container)
■{
■■UpdateGlowState();
■}
■
■override void OnMovedWithinCargo(EntityAI container)
■{
■■UpdateGlowState();
■}
■
■override void OnRemovedFromCargo(EntityAI container)
■{
■■UpdateGlowState();
■}
■
■override void EOnInit( IEntity other, int extra)
■{
■■UpdateGlowState();
■}
■
■override void OnItemLocationChanged( EntityAI old_owner, EntityAI new_owner)
■{
■■super.OnItemLocationChanged( old_owner, new_owner);
■■
■■UpdateGlowState();
■}
■
■override void EEHealthLevelChanged(int oldLevel, int newLevel, string zone)
■{
■■super.EEHealthLevelChanged(oldLevel,newLevel,zone);
```

```
// ---------------------------------------------
// ------ PumpkinSeedsPack.c - START --------------------
// ---------------------------------------------


class PumpkinSeedsPack extends SeedPackBase
{■
}


// ---------------------------------------------
// ------ PumpkinSeedsPack.c - END ----------------------
// ---------------------------------------------


// ---------------------------------------------
// ------ PunchedCard.c - START --------------------
// ---------------------------------------------


class PunchedCard : Paper
{
■override void SetActions()
■{
■■super.SetActions();

■■AddAction(ActionUseUndergroundPanel);
■}
■
■void OnUse()
■{
■■AddHealthLevel(1);
■}
}



// ---------------------------------------------
// ------ PunchedCard.c - END ----------------------
// ---------------------------------------------


// ---------------------------------------------
// ------ PurificationTablets.c - START --------------------
// ---------------------------------------------


class PurificationTablets extends Edible_Base
{
/*
■override void SetActions()
■{
■■super.SetActions();
■■
■■AddAction(ActionForceConsumeSingle);
■■AddAction(ActionConsumeSingle);
■■//AddAction(ActionForceFeed);
■■//AddAction(ActionEatBig);
■}
*/
};
```

```
// ---------------------------------------------
// ------ PurifyWater.c - START --------------------
// ---------------------------------------------


class PurifyWater extends RecipeBase
{■
■override void Init()
■{
■■m_Name = "#STR_PurifyWater0";
■■m_IsInstaRecipe = false;//should this recipe be performed instantly without animation
■■m_AnimationLength = 1;//animation length in relative time units
■■m_Specialty = -0.02;// value > 0 for roughness, value < 0 for precision
■■
■■
■■//conditions
■■m_MinDamageIngredient[0] = -1;//-1 = disable check
■■m_MaxDamageIngredient[0] = 3;//-1 = disable check
■■
■■m_MinQuantityIngredient[0] = 1;//-1 = disable check
■■m_MaxQuantityIngredient[0] = -1;//-1 = disable check
■■
■■m_MinDamageIngredient[1] = -1;//-1 = disable check
■■m_MaxDamageIngredient[1] = 3;//-1 = disable check
■■
■■m_MinQuantityIngredient[1] = -1;//-1 = disable check
■■m_MaxQuantityIngredient[1] = -1;//-1 = disable check
■■//----------------------------------------------------------------------------------------------------------------------
■■
■■//INGREDIENTS
■■//ingredient 1
■■InsertIngredient(0,"PurificationTablets");//you can insert multiple ingredients this way
■■
■■m_IngredientAddHealth[0] = 0;// 0 = do nothing
■■m_IngredientSetHealth[0] = -1; // -1 = do nothing
■■m_IngredientAddQuantity[0] = -1;// 0 = do nothing
■■m_IngredientDestroy[0] = false;//true = destroy, false = do nothing
■■m_IngredientUseSoftSkills[0] = false;// set 'true' to allow modification of the values by softskills on this i
■■
■■//ingredient 2
■■InsertIngredient(1,"Pot");//you can insert multiple ingredients this way
■■InsertIngredient(1,"CanisterGasoline");//you can insert multiple ingredients this way
■■InsertIngredient(1,"DisinfectantAlcohol");//you can insert multiple ingredients this way
■■InsertIngredient(1,"Canteen");//you can insert multiple ingredients this way
■■InsertIngredient(1,"WaterBottle");//you can insert multiple ingredients this way
■■InsertIngredient(1,"Vodka");//you can insert multiple ingredients this way
■■InsertIngredient(1,"WaterPouch_ColorBase");//you can insert multiple ingredients this way
■■InsertIngredient(1,"Barrel_ColorBase");//you can insert multiple ingredients this way
■■
■■m_IngredientAddHealth[1] = 0;// 0 = do nothing
■■m_IngredientSetHealth[1] = -1; // -1 = do nothing
■■m_IngredientAddQuantity[1] = 0;// 0 = do nothing
■■m_IngredientDestroy[1] = false;// false = do nothing
■■m_IngredientUseSoftSkills[1] = false;// set 'true' to allow modification of the values by softskills on this i
■■//----------------------------------------------------------------------------------------------------------------------
■■
■■//result1
■■//AddResult("");//add results here

■■m_ResultSetFullQuantity[0] = false;//true = set full quantity, false = do nothing
■■m_ResultSetQuantity[0] = -1;//-1 = do nothing
■■m_ResultSetHealth[0] = -1;//-1 = do nothing
■■m_ResultInheritsHealth[0] = -1;// (value) == -1 means do nothing; a (value) >= 0 means this result will i
```

```
// ---------------------------------------------
// ------ PUScopeOptic.c - START --------------------
// ---------------------------------------------


class PUScopeOptic extends ItemOptics
{
}


// ---------------------------------------------
// ------ PUScopeOptic.c - END ----------------------
// ---------------------------------------------


// ---------------------------------------------
// ------ QuantityConversions.c - START --------------------
// ---------------------------------------------


class QuantityConversions
{
    static string GetItemQuantityText( EntityAI item, bool showMax = false )
    {
        string quantity_text = "";
        if ( item.IsInherited( InventoryItem) )
        {
            ItemBase item_base;
            Class.CastTo(item_base, item);
            float quantity = item_base.GetQuantity();
            int ammo;
            if ( item.IsMagazine() )
            {
                Magazine magazine_item;
                Class.CastTo(magazine_item, item);
                ammo = magazine_item.GetAmmoCount();

                return ammo.ToString();
            }
            else if ( item.IsInherited( ItemBook) )
            {
                return "";
            }
            int stack_max = item.GetQuantityMax();
            
            //int max = item.ConfigGetInt("varQuantityMax");
            //string unit = item.ConfigGetString("stackedUnit");

            if (stack_max > 0)
            {
                if (stack_max == 1)
                {
                    if (quantity > 1)
                    {
                        if (showMax)
                            quantity_text = string.Format("%1/%2", quantity.ToString(), stack_max.ToString() );
                        //quantity_text = string.Format("%i/%i", quantity, stack_max );
                        else
                            quantity_text = quantity.ToString();
                    }
                    else
                    {
                        quantity_text = "";
```

```
// ---------------------------------------------
// ------ QuickBarBase.c - START --------------------
// ---------------------------------------------


const int MAX_QUICKBAR_SLOTS_COUNT = 10;
// Script File
class QuickBarItem
{
■EntityAI m_entity;
■bool m_enabled;
■
■void QuickBarItem()
■{
■■m_entity = NULL;
■■m_enabled = false;
■}
}

class QuickBarBase
{
■protected ref QuickBarItem m_aQuickbarEnts[MAX_QUICKBAR_SLOTS_COUNT];
■protected PlayerBase _player;
■protected int m_slotsCount;
■
■void QuickBarBase(PlayerBase player)
■{
■■for (int i = 0; i < MAX_QUICKBAR_SLOTS_COUNT; i++)
■■{
■■■m_aQuickbarEnts[i] = new QuickBarItem;
■■■m_aQuickbarEnts[i].m_enabled = false;
■■■m_aQuickbarEnts[i].m_entity = NULL;
■■}
■■
■■_player = player;
■■m_slotsCount = 0;
■}
//-------------------------------------------------------------■■
■void SetSize(int newSize)
■{
■■int i = m_slotsCount;
■■if ( newSize == m_slotsCount )
■■■return;
■■
■■if (newSize > MAX_QUICKBAR_SLOTS_COUNT)
■■■newSize = MAX_QUICKBAR_SLOTS_COUNT;
■■
■■if ( newSize > i )
■■{■
■■■for (; i < newSize; i++)
■■■{
■■■■EntityAI entity = m_aQuickbarEnts[i].m_entity;
■■■■if ( entity != NULL && entity.GetHierarchyRootPlayer() == _player )
■■■■{
■■■■■m_aQuickbarEnts[i].m_enabled = true;
■■■■}
■■■}■
■■}
■■else
■■{■
■■■for (i--; i >= newSize; i--)
■■■■m_aQuickbarEnts[i].m_enabled = false;
■■}
```

```
// ---------------------------------------------
// ------ QuiltedJacket_ColorBase.c - START --------------------
// ---------------------------------------------


class QuiltedJacket_ColorBase extends Clothing
{
■override void SetActions()
■{
■■super.SetActions();
■■AddAction(ActionWringClothes);
■}
};
class QuiltedJacket_Black extends QuiltedJacket_ColorBase {};
class QuiltedJacket_Green extends QuiltedJacket_ColorBase {};
class QuiltedJacket_Blue extends QuiltedJacket_ColorBase {};
class QuiltedJacket_Red extends QuiltedJacket_ColorBase {};
class QuiltedJacket_Grey extends QuiltedJacket_ColorBase {};
class QuiltedJacket_Orange extends QuiltedJacket_ColorBase {};
class QuiltedJacket_Yellow extends QuiltedJacket_ColorBase {};
class QuiltedJacket_Violet extends QuiltedJacket_ColorBase {};


// ---------------------------------------------
// ------ QuiltedJacket_ColorBase.c - END ----------------------
// ---------------------------------------------


// ---------------------------------------------
// ------ Rabbit.c - START --------------------
// ---------------------------------------------


class DeadRabbit extends Edible_Base
{
■override bool CanBeCookedOnStick()
■{
■■return false;
■}

■override bool CanBeCooked()
■{
■■return false;
■}■
■
■override bool IsCorpse()
■{
■■return true;
■}

■override bool CanDecay()
■{
■■return true;
■}
}


// ---------------------------------------------
// ------ Rabbit.c - END ----------------------
// ---------------------------------------------
// ---------------------------------------------
// ------ RabbitLegMeat.c - START --------------------
// ---------------------------------------------


class RabbitLegMeat extends Edible_Base
{
■override bool CanBeCooked()
■{
■■return true;
■}■■
■
■override bool CanBeCookedOnStick()
■{
■■return true;
■}
■
■override bool IsMeat()
■{
■■return true;
■}

■override bool CanDecay()
■{
■■return true;
■}
■
■override void SetActions()
■{
■■super.SetActions();
■■
■■AddAction(ActionForceFeed);
■■AddAction(ActionEatMeat);
■■
■■AddAction(ActionCreateIndoorFireplace);
■■AddAction(ActionCreateIndoorOven);
■}
}



// ---------------------------------------------
// ------ RabbitLegMeat.c - END ----------------------
// ---------------------------------------------


// ---------------------------------------------
// ------ RadarCap_ColorBase.c - START --------------------
// ---------------------------------------------


class RadarCap_ColorBase extends Clothing
{
■override void SetActions()
■{
■■super.SetActions();
■■AddAction(ActionWringClothes);
■}
};
class RadarCap_Black extends RadarCap_ColorBase {};
class RadarCap_Blue extends RadarCap_ColorBase {};
class RadarCap_Brown extends RadarCap_ColorBase {};
class RadarCap_Green extends RadarCap_ColorBase {};
```

```
// ---------------------------------------------
// ------ RadialMenu.c - START --------------------
// ---------------------------------------------


enum RadialMenuControlType
{
■MOUSE,
■CONTROLLER■
}

class RadialMenu : ScriptedWidgetEventHandler
{
■protected Widget■■■■■m_Parent;
■protected Widget ■■■■■m_ItemCardsContainer;
■protected Widget■■■■■m_RadialSelector;
■protected ImageWidget■■■■m_RadialSelectorImage;
■protected ImageWidget ■■■■m_RadialSelectorPointerImage;
■protected int ■■■■■■m_RadialSelectorOriginalColor;
■protected int ■■■■■■m_RadialSelectorDisabledColor;
■protected Widget ■■■■■m_SelectedObject;
■protected ref map<Widget, float> ■m_RadialItemCards;
■
■protected float ■■■■■m_AngleRadOffset;
■protected ref Timer ■■■■m_UpdateTimer;
■
■//widget
■static const string ■■■■RADIAL_SELECTOR■ ■■■= "RadialSelector";
■static const string ■■■■RADIAL_SELECTOR_IMAGE■■= "SelectorImage";
■static const string ■■■■RADIAL_SELECTOR_POINTER■■= "SelectorPointer";
■static const string ■■■■RADIAL_DELIMITER_CONTAINER ■= "RadialDelimiterContainer";
■static const string ■■■■RADIAL_ITEM_CARD_CONTAINER ■= "RadialItemCardContainer";

■//controls
■protected RadialMenuControlType ■m_ControlType;
■private UAIDWrapper ■■■■m_SelectInputWrapper;
■private UAIDWrapper ■■■■m_BackInputWrapper;
■protected float ■■■■■m_ControllerAngle;
■protected float ■■■■■m_ControllerTilt;

■//controller
■protected int ■■■■■■m_ControllerTimout;
■protected bool■■■■■■m_IsControllerTimoutEnabled ■■= true;■■■//enables/disables controller desel
■protected const float■■■■CONTROLLER_DESELECT_TIMEOUT ■■= 1000;■■■//timeout [ms] after w
■protected const float ■■■■CONTROLLER_TILT_TRESHOLD_SELECT■■= 0.8;■■■//tilt value (0.0-1.0
■protected const float ■■■■CONTROLLER_TILT_TRESHOLD_EXECUTE■= 1.0;■■■//tilt value (0.0-1.0
■
■//mouse
■protected bool ■■■■■■m_WidgetInitialized;
■protected const float ■■■■MOUSE_SAFE_ZONE_RADIUS = 120;■■//Radius [px] of safe zone where
■
■//References
■protected float ■■■■■m_RadiusOffset;■■■■■//Radius [% of the main container size]
■protected float ■■■■■m_ExecuteDistanceOffset;■■//Distance offset [% of the main container size] afte
■protected float■■■■■■m_OffsetFromTop;■■■■//first item in the menu won't be directly on top but offs
■protected float■■■■■■m_ItemCardRadiusOffset;■■■//Radius [% of the main container size] for item c
■protected string■■■■■m_DelimiterLayout;■■■■//layout file name with path
■
■ref UIScriptedMenu■■■■■m_RegisteredClass;■
■ref static RadialMenu■■■■m_Instance;
■
■/*
■
```

```
// ---------------------------------------------
// ------ RadialProgressBar.c - START --------------------
// ---------------------------------------------


// ---------------------------------------------------------
class RadialProgressBar
{
■reference float speed;
■reference float start_rotation;

■protected string m_BarHider;
■protected string m_BarPart;
■protected Widget m_Root;■
■protected ref AnimatorTimer m_Anim;
■protected float x, y, z;
■protected float rotation = 0;
■protected int stage = 0;

■// ---------------------------------------------------------
■void RadialProgressBar()
■{
■■m_Anim = new AnimatorTimer();
■■GetGame().GetUpdateQueue(CALL_CATEGORY_GUI).Insert(this.Update);
■}
■
■// ---------------------------------------------------------
■void ~RadialProgressBar()
■{
■■GetGame().GetUpdateQueue(CALL_CATEGORY_GUI).Remove(this.Update);
■}
■
■void SetProgress( float progress )
■{
■■if( progress < 50 )
■■{
■■■stage = 0;
■■}
■■rotation = 360 * ( progress / 100 );
■}
■
■// ---------------------------------------------------------
■protected void Update(float tDelta)
■{
■■m_Anim.Tick(tDelta);
■■
■■Widget child = m_Root.GetChildren();
■■
■■int index = 0;
■■while ( child )
■■{
■■■UpdateChild( child, index );
■■■index++;
■■■child = child.GetSibling();
■■}
■}
■
■protected void UpdateChild( Widget child, int index )
■{
■■float rotation_value = ( m_Anim.GetTargetValue() * Math.RAD2DEG );
■■if( child.GetName() == m_BarHider )
■■{
■■■if( stage == 0 )
```

```
// ---------------------------------------------
// ------ RadialQuickbarMenu.c - START --------------------
// ---------------------------------------------


enum RadialQuickbarCategory
{
■DEFAULT,
■SPECIALIZED_LIGHTS
}

class RadialQuickbarItem
{
■protected bool ■■■■■m_IsLightSourceExtra;
■protected bool ■■■■■m_IsNVG;
■protected int ■■■■■m_Id;
■protected int ■■■■■m_Category;
■protected int ■■■■■m_CategorySwitchID;
■protected EntityAI ■■■■m_Item;
■protected string ■■■■m_ItemName;
■
■//radial menu
■protected Widget ■■■m_RadialMenuSelector;
■protected Widget■■■m_RadialMenuItemCard;
■
■void RadialQuickbarItem( int id, EntityAI item, string item_name, int category = RadialQuickbarCategory.
■{
■■m_Id■■■■= id;
■■m_Item ■■■■= item;
■■m_ItemName■■■= item_name;
■■m_Category ■■■= category;
■■m_CategorySwitchID ■= category_switch;
■■
■■//
■■if (ItemBase.Cast(m_Item))
■■{
■■■m_IsNVG = ItemBase.Cast(m_Item).IsNVG();
■■■m_IsLightSourceExtra = ItemBase.Cast(m_Item).IsLightSource();
■■}
■}
■
■EntityAI GetItem()
■{
■■return m_Item;
■}
■
■void SetItem( EntityAI item )
■{
■■m_Item = item;
■}
■
■bool IsLightSourceExtra()
■{
■■return m_IsLightSourceExtra;
■}
■
■bool IsNVGExtra()
■{
■■return m_IsNVG;
■}
■
■int GetId()
■{
```

```
// ---------------------------------------------
// ------ Radio.c - START --------------------
// ---------------------------------------------


class Radio extends ItemRadio
{
■override bool IsTransmitter()
■{
■■return true;
■}
■
■//--- COMMON
■bool CanOperate()
■{
■■return GetCompEM().IsSwitchedOn();■
■}
■
■//--- POWER EVENTS
■override void OnSwitchOn()
■{
■■//switch device on
■■SwitchOn ( true );
■}

■override void OnSwitchOff()
■{
■■//switch device off
■■SwitchOn ( false );
■}

■override void OnWorkStop()
■{
■■//turn off device
■■GetCompEM().SwitchOff();
■}
■
■//--- RADIO ACTIONS
■void TuneNextStation()
■{
■■//tune next station
■■TuneNext();
■}

■void TunePreviousStation()
■{
■■//tune previous station
■■TunePrev();
■}
■
■override void SetActions()
■{
■■super.SetActions();
■■
■■AddAction(ActionTurnOnTransmitter);
■■AddAction(ActionTurnOffTransmitter);
■■AddAction(ActionTuneRadioStation);
■}
}



// ---------------------------------------------
// ---------------------------------------------
// ------ Rag.c - START --------------------
// ---------------------------------------------


class Rag extends ItemBase
{
■override bool CanSwapEntities(EntityAI otherItem, InventoryLocation otherDestination, InventoryLocatio
■{
■■if (!super.CanSwapEntities(otherItem, otherDestination, destination))
■■{
■■■return false;
■■}

■■if (Torch.Cast(GetHierarchyParent()) && otherItem.IsInherited(Rag))
■■{
■■■return false;
■■}

■■return true;
■}
■
■
■override bool CanPutAsAttachment(EntityAI parent)
■{
■■if (!super.CanPutAsAttachment(parent))
■■{
■■■return false;
■■}

■■if (GetQuantity() > 1 && PlayerBase.Cast(parent))
■■{
■■■return false;
■■}

■■return true;
■}
■
■//================================================================
■// IGNITION ACTION
■//================================================================
■override bool HasFlammableMaterial()
■{
■■return true;
■}
■
■override bool CanBeIgnitedBy(EntityAI igniter = null)
■{
■■return GetHierarchyParent() == null;
■}
■
■override bool CanIgniteItem(EntityAI ignite_target = null)
■{
■■return false;
■}
■
■override bool CanBeCombined(EntityAI other_item, bool reservation_check = true, bool stack_max_limit
■{
■■if (!super.CanBeCombined(other_item, reservation_check, stack_max_limit))
■■{
■■■return false;
■■}
```

```
// ---------------------------------------------
// ------ Raincoat_ColorBase.c - START --------------------
// ---------------------------------------------


class Raincoat_ColorBase extends Clothing {};
class Raincoat_Orange extends Raincoat_ColorBase {};
class Raincoat_Green extends Raincoat_ColorBase {};
class Raincoat_Yellow extends Raincoat_ColorBase {};
class Raincoat_Pink extends Raincoat_ColorBase {};
class Raincoat_Red extends Raincoat_ColorBase {};
class Raincoat_Blue extends Raincoat_ColorBase {};
class Raincoat_Black extends Raincoat_ColorBase {};


// ---------------------------------------------
// ------ Raincoat_ColorBase.c - END ----------------------
// ---------------------------------------------


// ---------------------------------------------
// ------ RainProcurementManager.c - START --------------------
// ---------------------------------------------


class RainProcurementManager
{
■protected ■■■ItemBase ■m_ProcuringItem;■
■protected ■■■int ■■m_IsUnderRoof;
■protected■ref■Timer ■■m_UpdateTimer;
■protected ■const ■int ■■RAIN_COEFFICIENT = 10;

■void RainProcurementManager( ItemBase procuring_item )
■{
■■m_ProcuringItem = procuring_item;
■}
■
■// ------------------------------------------------------------------------------------------
■void InitRainProcurement()
■{
■■m_IsUnderRoof = MiscGameplayFunctions.IsUnderRoof(m_ProcuringItem);

■■//m_ProcuringItem.SetQuantity(0); /*set to 0 for debug purposses*/

■■if ( !m_IsUnderRoof )
■■{
■■■m_UpdateTimer = new Timer();
■■■m_UpdateTimer.Run( 10, this, "RainProcurementCheck", NULL, true );
■■}
■}

■// ------------------------------------------------------------------------------------------
■void RainProcurementCheck()
■{■■
■■float rain_intensity = GetGame().GetWeather().GetRain().GetActual();
■■float fill_per_update = RAIN_COEFFICIENT * rain_intensity;

■■if ( rain_intensity > 0 )
■■{
■■■if ( m_ProcuringItem.GetQuantity() < m_ProcuringItem.GetQuantityMax() )
■■■{
■■■■Liquid.FillContainerEnviro( m_ProcuringItem, LIQUID_WATER, fill_per_update );
```

```
// ---------------------------------------------
// ------ RandomGeneratorSyncManager.c - START --------------------
// ---------------------------------------------


enum RandomGeneratorSyncUsage
{
■RGSRecoil,
■RGSJam,
■RGSGeneric,
■RGSAimingModel,
■Count,
}

class RandomGeneratorSyncManager
{
■const int USAGE_COUNT = RandomGeneratorSyncUsage.Count;
■
■float randomValuesArray[USAGE_COUNT];
■DayZPlayer m_player;
■
■void RandomGeneratorSyncManager(DayZPlayer player)
■{
■■m_player = player;
■}

■void Update()
■{
■■for(int i = 0; i < USAGE_COUNT; i++ )
■■{
■■■randomValuesArray[i] = m_player.Random01();
■■}
■}■
■
■
■float GetRandom01(RandomGeneratorSyncUsage usage_index)
■{
■■if(usage_index < USAGE_COUNT)
■■{
■■■return randomValuesArray[usage_index];
■■}
■■return -1;
■}
■
■float GetRandomInRange(RandomGeneratorSyncUsage usage_index, float min, float max)
■{
■■return GetRandom01(usage_index)*(max - min) + min;
■}
}


// ---------------------------------------------
// ------ RandomGeneratorSyncManager.c - END ---------------------
// ---------------------------------------------


// ---------------------------------------------
// ------ Rangefinder.c - START --------------------
// ---------------------------------------------


class Rangefinder extends PoweredOptic_Base
{
```

```
// ---------------------------------------------
// ------ Raycaster.c - START --------------------
// ---------------------------------------------


// Item for raycast debugging. It's also fun to play around with at night.

class Raycaster extends ItemBase
{
■protected ref Timer ■m_Timer;
■
■override void OnWorkStart()
■{
■■if ( !GetGame().IsServer()  ||  !GetGame().IsMultiplayer() ) // Client side
■■{
■■■StartPeriodicMeasurement();
■■}
■}
■
■void StartPeriodicMeasurement()
■{
■■if( !m_Timer )
■■{
■■■m_Timer = new Timer;
■■}
■■
■■m_Timer.Run( 0.01, this, "PrepareMeasurement", null, true );
■}
■
■void PrepareMeasurement()
■{
■■DoMeasurement();
■■DoMeasurement();
■}
■
■void DoMeasurement()
■{
■■float dispersion = 80;
■■
■■vector ■■from ■■■= GetPosition() + ( GetMemoryPointPos("beamStart") + ((GetOrientation()+Vector
■■vector ■■ori ■■■= GetOrientation() + Vector(90,0,0) + Vector(dispersion/2 - Math.RandomFloat(0, dis
■■vector ■■to ■■■■= from + (ori.AnglesToVector() )* 30;
■■vector ■■contact_pos;
■■vector ■■contact_dir;
■■int ■■contactComponent;
■■
■■bool is_collision = DayZPhysics.RaycastRV( from, to, contact_pos, contact_dir, contactComponent, NU
■■
■■if (is_collision)
■■{
■■■vector hit_normal = contact_dir.VectorToAngles();
■■■hit_normal[1] = hit_normal[1] + 90;
■■■
■■■// Generate result
■■■Particle p = ParticleManager.GetInstance().PlayInWorld(ParticleList.DEBUG_DOT, contact_pos);
■■■p.SetOrientation(hit_normal);
■■}
■}
■
■void StopPeriodicMeasurement()
■{
■■if( m_Timer )
■■{
```

```
// ---------------------------------------------
// ------ RDG2SmokeGrenade_ColorBase.c - START --------------------
// ---------------------------------------------


class RDG2SmokeGrenade_ColorBase extends SmokeGrenadeBase
{
■const string SOUND_SMOKE_START = "SmokegGrenades_RDG2_start_loop_SoundSet";
■const string SOUND_SMOKE_LOOP = "SmokegGrenades_RDG2_active_loop_SoundSet";
■const string SOUND_SMOKE_END = "SmokegGrenades_RDG2_end_loop_SoundSet";
■
■void RDG2SmokeGrenade_ColorBase()
■{
■■SetAmmoType("");
■■SetFuseDelay(2);
■■SetParticlePosition("0 0.1 0");
■■SetSoundSmokeStart(SOUND_SMOKE_START);
■■SetSoundSmokeLoop(SOUND_SMOKE_LOOP);
■■SetSoundSmokeEnd(SOUND_SMOKE_END);
■}
■
■void ~RDG2SmokeGrenade_ColorBase() {}
};

class RDG2SmokeGrenade_Black extends RDG2SmokeGrenade_ColorBase
{
■void RDG2SmokeGrenade_Black()
■{
■■SetParticleSmokeStart(ParticleList.GRENADE_RDG2_BLACK_START);
■■SetParticleSmokeLoop(ParticleList.GRENADE_RDG2_BLACK_LOOP);
■■SetParticleSmokeEnd(ParticleList.GRENADE_RDG2_BLACK_END);
■}
}

class RDG2SmokeGrenade_White extends RDG2SmokeGrenade_ColorBase
{
■void RDG2SmokeGrenade_White()
■{
■■SetParticleSmokeStart(ParticleList.GRENADE_RDG2_WHITE_START);
■■SetParticleSmokeLoop(ParticleList.GRENADE_RDG2_WHITE_LOOP);
■■SetParticleSmokeEnd(ParticleList.GRENADE_RDG2_WHITE_END);
■}
}



// ---------------------------------------------
// ------ RDG2SmokeGrenade_ColorBase.c - END ----------------------
// ---------------------------------------------


// ---------------------------------------------
// ------ RecipeBase.c - START --------------------
// ---------------------------------------------


const int MAX_NUMBER_OF_INGREDIENTS = 2;
const int MAXIMUM_RESULTS = 10;
const float DEFAULT_SPAWN_DISTANCE = 0.5;
class RecipeBase
{
■string m_ItemsToCreate[MAXIMUM_RESULTS];
■ref array<string> m_Ingredients[MAX_NUMBER_OF_INGREDIENTS];
```

```
// ---------------------------------------------
// ------ RecoilBase.c - START --------------------
// ---------------------------------------------


class RecoilBase
{
■bool m_DebugMode;
■
■Weapon_Base m_Weapon;
■PlayerBase m_Player;
■protected bool m_DeleteRequested;
■protected float m_Time;//how much time has elapsed since first update
■protected float m_ReloadTime;//reload time config parameter of the weapon
■protected vector m_RecoilModifier;
■protected bool m_IsClient;
■// for mouse offset
■float m_MouseOffsetRangeMin;//in degrees min
■float m_MouseOffsetRangeMax;//in degrees max
■float m_MouseOffsetRelativeTime = 1;//[0..1] a time it takes to move the mouse the required distance rel
■float m_HandsOffsetRelativeTime = 1;//[0..1] a time it takes to move the hands the required distance give
■float m_CamOffsetRelativeTime = 1;//[0..1] a time it takes to move the camera the required distance rela
■float m_CamOffsetDistance = 0.05;//how far the camera will travel along the z-axis in cm
■float m_MouseOffsetDistance;//how far should the mouse travel
■float m_TimeNormalized;
■//protected float m_MouseOffsetResult;//in degrees max
■protected vector m_MouseOffsetTarget;//move the mouse towards this point
■protected vector m_MouseOffsetTargetAccum;//the overall mouse offset so far(all deltas accumulated)
■protected float m_Angle;//result between the min and max
■// mouse end
■
■protected ref array<vector> m_HandsCurvePoints;
■
■void RecoilBase(Weapon_Base weapon)
■{
■■m_Weapon = weapon;
■■//m_DebugMode = false;
■■m_DebugMode = GetDayZGame().IsAimLogEnabled();
■■m_Player = PlayerBase.Cast(weapon.GetHierarchyRootPlayer());
■■m_HandsCurvePoints = new array<vector>;
■■Init();
■■PostInit(weapon);
■}
■
■void Init();
■
■Weapon_Base GetWeapon()
■{
■■return m_Weapon;
■}
■
■void PostInit(Weapon_Base weapon)
■{
■■int muzzleIndex = weapon.GetCurrentMuzzle();
■■m_Angle = m_Player.GetRandomGeneratorSyncManager().GetRandomInRange(RandomGeneratorSy
■■m_RecoilModifier = GetRecoilModifier( GetWeapon() );
■■if(m_DebugMode) Print(m_Angle);
■■
■■m_ReloadTime = weapon.GetReloadTime(muzzleIndex);
■■m_MouseOffsetTarget = vector.YawToVector(m_Angle);
■■m_MouseOffsetTarget = m_MouseOffsetTarget * m_MouseOffsetDistance;
■■m_IsClient = !GetGame().IsDedicatedServer();
■■m_CamOffsetDistance *= m_RecoilModifier[2];
```

```
// ---------------------------------------------
// ------ Red9.c - START --------------------
// ---------------------------------------------


class Red9 : Pistol_Base
{
};


// ---------------------------------------------
// ------ Red9.c - END ----------------------
// ---------------------------------------------


// ---------------------------------------------
// ------ Red9Bttstck.c - START --------------------
// ---------------------------------------------


class Red9Bttstck extends Inventory_Base
{
■override bool CanPutAsAttachment( EntityAI parent )
■{
■■if(!super.CanPutAsAttachment(parent)) {return false;}
■■if ( !parent.IsKindOf("PlateCarrierHolster") && !parent.IsKindOf("PlateCarrierComplete") && !parent.IsK
■■{
■■■return true;
■■}
■■return false;
■}■
}


// ---------------------------------------------
// ------ Red9Bttstck.c - END ----------------------
// ---------------------------------------------


// ---------------------------------------------
// ------ ReflexVest.c - START --------------------
// ---------------------------------------------


class ReflexVest extends Clothing {};


// ---------------------------------------------
// ------ ReflexVest.c - END ----------------------
// ---------------------------------------------


// ---------------------------------------------
// ------ Refridgerator.c - START --------------------
// ---------------------------------------------


class Refridgerator extends ItemBase
{■
■SoundOnVehicle ■m_SoundLoopEntity;
■
■override bool IsElectricAppliance()
■{
```

```
// ---------------------------------------------
// ------ RefuelTorch.c - START --------------------
// ---------------------------------------------


// This recipe adds fuel to torch

class RefuelTorch extends RecipeBase
{■
■override void Init()
■{
■■m_Name = "#STR_RefuelTorch0";
■■m_IsInstaRecipe = false;//should this recipe be performed instantly without animation
■■m_AnimationLength = 0.25;//animation length in relative time units
■■m_Specialty = 0.02;// value > 0 for roughness, value < 0 for precision
■■
■■//conditions
■■m_MinDamageIngredient[0] = -1;//-1 = disable check
■■m_MaxDamageIngredient[0] = -1;//-1 = disable check
■■
■■m_MinQuantityIngredient[0] = -1;//-1 = disable check
■■m_MaxQuantityIngredient[0] = -1;//-1 = disable check
■■
■■m_MinDamageIngredient[1] = -1;//-1 = disable check
■■m_MaxDamageIngredient[1] = -1;//-1 = disable check
■■
■■m_MinQuantityIngredient[1] = -1;//-1 = disable check
■■m_MaxQuantityIngredient[1] = -1;//-1 = disable check
■■//----------------------------------------------------------------------------------------------------------------------
■■
■■//INGREDIENTS
■■//ingredient 1
■■InsertIngredient(0,"Rag");//you can insert multiple ingredients this way
■■
■■m_IngredientAddHealth[0] = 0;// 0 = do nothing
■■m_IngredientSetHealth[0] = -1; // -1 = do nothing
■■m_IngredientAddQuantity[0] = 0;// 0 = do nothing
■■m_IngredientDestroy[0] = false;//true = destroy, false = do nothing
■■m_IngredientUseSoftSkills[0] = false;// set 'true' to allow modification of the values by softskills on this i
■■
■■//ingredient 2
■■InsertIngredient(1,"Torch");//you can insert multiple ingredients this way
■■InsertIngredient(1,"LongTorch");//you can insert multiple ingredients this way
■■
■■m_IngredientAddHealth[1] = 0;// 0 = do nothing
■■m_IngredientSetHealth[1] = -1; // -1 = do nothing
■■m_IngredientAddQuantity[1] = -1;// 0 = do nothing
■■m_IngredientDestroy[1] = false;// false = do nothing
■■m_IngredientUseSoftSkills[1] = false;// set 'true' to allow modification of the values by softskills on this i
■■//----------------------------------------------------------------------------------------------------------------------
■■
■■//result1
■■//AddResult("Torch");//add results here

■■m_ResultSetFullQuantity[0] = false;//true = set full quantity, false = do nothing
■■m_ResultSetQuantity[0] = -1;//-1 = do nothing
■■m_ResultSetHealth[0] = -1;//-1 = do nothing
■■m_ResultInheritsHealth[0] = -2;// (value) == -1 means do nothing; a (value) >= 0 means this result will i
■■m_ResultInheritsColor[0] = -1;// (value) == -1 means do nothing; a (value) >= 0 means this result class
■■m_ResultToInventory[0] = -2;//(value) == -2 spawn result on the ground;(value) == -1 place anywhere i
■■m_ResultUseSoftSkills[0] = false;// set 'true' to allow modification of the values by softskills on this resu
■■m_ResultReplacesIngredient[0] = -1;// value == -1 means do nothing; a value >= 0 means this result wi
■}
```

```
// ---------------------------------------------
// ------ Remington12.c - START --------------------
// ---------------------------------------------


class Remington12 : Rifle_Base
{
};


// ---------------------------------------------
// ------ Remington12.c - END ----------------------
// ---------------------------------------------


// ---------------------------------------------
// ------ RemoteDetonator.c - START --------------------
// ---------------------------------------------


enum ERemoteDetonatorLEDState
{
■OFF = 0,
■LIT,
}

class RemoteDetonator : Inventory_Base
{
■const string COLOR_LED_OFF ■■= "#(argb,8,8,3)color(0,0,0,1.0,co)";
■const string COLOR_LED_LIT ■■= "#(argb,8,8,3)color(1,0,0,1.0,co)";
■const string SELECTION_NAME_LED = "LED";

■protected ERemoteDetonatorLEDState m_LastLEDState;
■
■bool IsKit()
■{
■■return true;
■}
■
■void UpdateLED(ERemoteDetonatorLEDState pState, bool pForced = false)
■{
■■int selectionIdx = GetHiddenSelectionIndex(SELECTION_NAME_LED);

■■switch (pState)
■■{
■■case ERemoteDetonatorLEDState.LIT:
■■■SetObjectTexture(selectionIdx, COLOR_LED_LIT);
■■break;
■■default:
■■■SetObjectTexture(selectionIdx, COLOR_LED_OFF);
■■}
■■
■■m_LastLEDState = pState;
■■SetSynchDirty();
■}

■override void SetActions()
■{
■■super.SetActions();

■■AddAction(ActionAttachExplosivesTrigger);
■■AddAction(ActionDisarmExplosiveWithRemoteDetonatorUnpaired);
■}
```

```
// ---------------------------------------------
// ------ RemotelyActivatedItemBehaviour.c - START --------------------
// ---------------------------------------------


class RemotelyActivatedItemBehaviour
{
■protected EntityAI■■m_Parent;
■protected EntityAI■■m_PairDevice;
■protected int■■■m_PairDeviceNetIdLow;
■protected int■■■m_PairDeviceNetIdHigh;

■void RemotelyActivatedItemBehaviour(notnull EntityAI pParent)
■{
■■m_Parent = pParent;
■■m_PairDeviceNetIdLow = -1;
■■m_PairDeviceNetIdHigh = -1;
■}
■
■void OnVariableSynchronized()
■{
■■Pair();
■}
■
■void Pair()
■{
■■EntityAI device = EntityAI.Cast(GetGame().GetObjectByNetworkId(GetPairDeviceNetIdLow(), GetPairD
■■if (device)
■■{
■■■Pair(device);
■■}
■}

■void Pair(notnull EntityAI device)
■{
■■m_PairDevice = device;
■■SetPairDeviceNetIds(device);
■■
■■if (!m_Parent.GetPairDevice() || m_Parent.GetPairDevice() != m_PairDevice)
■■{
■■■m_PairDevice.PairRemote(m_Parent);
■■}

■■m_PairDevice.SetSynchDirty();
■■m_Parent.SetSynchDirty();
■}
■
■void Unpair()
■{
■■m_PairDeviceNetIdLow ■= -1;
■■m_PairDeviceNetIdHigh ■= -1;

■■if (m_PairDevice)
■■{
■■■m_PairDevice.SetSynchDirty();
■■■m_PairDevice = null;
■■}

■■m_Parent.SetSynchDirty();
■}
■
■EntityAI GetPairDevice()
■{
```

```
// ---------------------------------------------
// ------ RemotePlayerDamageDebug.c - START --------------------
// ---------------------------------------------


class RemotePlayerDamageDebug
{
■const int MAX_DAMAGE_RECORDS = 5;
■PlayerBase m_Player;
■bool m_ChangedSinceSerialization;
■
■ref array<ref DamageData> m_DamageList = new array<ref DamageData>;
■
■void RemotePlayerDamageDebug(PlayerBase player)
■{
■■m_Player = player;
■}
■
■void AddDamage(float value_global, float value_blood, float value_shock)
■{
■■m_ChangedSinceSerialization = true;
■■DamageData damage_data = new DamageData( value_global, value_blood, value_shock );
■■m_DamageList.InsertAt(damage_data,0);
■■if( m_DamageList.Count() > MAX_DAMAGE_RECORDS )
■■{
■■■m_DamageList.RemoveOrdered(MAX_DAMAGE_RECORDS);
■■}
■}
■
■void InsertDamageObject(DamageData damage_object)
■{
■■m_DamageList.Insert(damage_object);
■}
■

■PlayerBase GetPlayer()
■{
■■return m_Player;
■}
■
■
■void Get(array<ref DamageData> damage_list)
■{
■■for(int i = 0; i < m_DamageList.Count(); i++)
■■{
■■■damage_list.Insert(m_DamageList.Get(i));
■■}
■}
■
■void GetReversed(array<ref DamageData> damage_list)
■{
■■int index = m_DamageList.Count() - 1;
■■for(; index >= 0; index--)
■■{
■■■damage_list.Insert(m_DamageList.Get(index));
■■}
■}
■
■void Serialize(array<ref RemotePlayerDamageDebug> list)
■{
■■if( m_ChangedSinceSerialization )
■■{
■■■list.Insert(this);
```

```
// ---------------------------------------------
// ------ RemotePlayerMeta.c - START --------------------
// ---------------------------------------------


class RemotePlayerMeta
{
■eRemoteDebugType ■m_DebugType;
■PlayerBase ■■■m_Player;
■
■void RemotePlayerMeta(PlayerBase player, eRemoteDebugType type )
■{
■■m_Player = player;
■■m_DebugType = type;
■}
■
■void SetDebugType(eRemoteDebugType type)
■{
■■m_DebugType = type;
■}
■
■eRemoteDebugType GetDebugType(eRemoteDebugType type)
■{
■■return m_DebugType;
■}
■
■void SetPlayer(PlayerBase player)
■{
■■m_Player = player;
■}
■
■PlayerBase GetPlayer()
■{
■■return m_Player;
■}
}


// ---------------------------------------------
// ------ RemotePlayerMeta.c - END ----------------------
// ---------------------------------------------


// ---------------------------------------------
// ------ RemotePlayerStatDebug.c - START --------------------
// ---------------------------------------------


enum eRemoteStatType
{
■NONE,
■DAMAGE_SYSTEM = 1,
■PLAYER_STATS = 2,
■OTHER = 4,
}


class RemotePlayerStatDebug
{
■PlayerBase m_Player;
■ref array<ref StatDebugObject> m_Stats = new array<ref StatDebugObject>;
■string m_Name;
■vector m_Pos;
```

```
// ---------------------------------------------
// ------ RenderTarget.c - START --------------------
// ---------------------------------------------


#ifdef GAME_TEMPLATE

[EditorAttribute("box", "GameLib/Scripted", "Render target", "-0.25 -0.25 -0.25", "0.25 0.25 0.25", "255 0 0 
class RenderTargetClass
{

}

RenderTargetClass RenderTargetSource;

class RenderTarget: GenericEntity
{
■[Attribute("0", "slider", "Camera index", "0 31 1")]
■int CameraIndex;
■[Attribute("0", "editbox", "Position X <0, 1>")]
■float X;
■[Attribute("0", "editbox", "Position Y <0, 1>")]
■float Y;
■[Attribute("1", "editbox", "Render target width <0, 1>")]
■float Width;
■[Attribute("1", "editbox", "Render target height <0, 1>")]
■float Height;
■[Attribute("-1", "editbox", "Sort index (the lesser the more important)")]
■int Sort;
■[Attribute("0", "combobox", "Autoinit", "", { ParamEnum("No", "0"), ParamEnum("Yes", "1") } )]
■int AutoInit;
■[Attribute("0", "combobox", "Forcing creation of render target for camera #0 in Workbench", "", { ParamE
■bool ForceCreation;
■bool m_Show = true; // when autoinit, wait with showing the render target after all entities are created (E
■ref RenderTargetWidget m_RenderWidget;
■
■void RenderTarget(IEntitySource src, IEntity parent)
■{
■■SetFlags(EntityFlags.ACTIVE, false);

■■if (AutoInit)
■■{
■■■m_Show = false;
■■■SetEventMask(EntityEvent.INIT);
■■■Init();
■■}
■}

■void ~RenderTarget()
■{
■■delete m_RenderWidget;
■}
■
■void Init()
■{
■■#ifdef WORKBENCH // Workbench is using its own renderer for main camera, it is not using render tar
■■■if (!ForceCreation && CameraIndex == 0)
■■■■return;
■■#endif

■■int screenW, screenH;
■■GetScreenSize(screenW, screenH);
```

```
// ---------------------------------------------
// ------ RepairElectric.c - START --------------------
// ---------------------------------------------


class RepairElectric extends RecipeBase
{■
■override void Init()
■{
■■m_Name = "#repair";
■■m_IsInstaRecipe = false;//should this recipe be performed instantly without animation
■■m_AnimationLength = 1;//animation length in relative time units
■■m_Specialty = 0.03;// value > 0 for roughness, value < 0 for precision
■■
■■
■■//conditions
■■m_MinDamageIngredient[0] = -1;//-1 = disable check
■■m_MaxDamageIngredient[0] = 3;//-1 = disable check
■■
■■m_MinQuantityIngredient[0] = -1;//-1 = disable check
■■m_MaxQuantityIngredient[0] = -1;//-1 = disable check
■■
■■m_MinDamageIngredient[1] = 1;//-1 = disable check
■■m_MaxDamageIngredient[1] = 3;//-1 = disable check
■■
■■m_MinQuantityIngredient[1] = -1;//-1 = disable check
■■m_MaxQuantityIngredient[1] = -1;//-1 = disable check
■■//----------------------------------------------------------------------------------------------------------------------
■■
■■//INGREDIENTS
■■//ingredient 1
■■InsertIngredient(0,"ElectronicRepairKit");//you can insert multiple ingredients this way
■■
■■m_IngredientAddHealth[0] = 0;// 0 = do nothing
■■m_IngredientSetHealth[0] = -1; // -1 = do nothing
■■m_IngredientAddQuantity[0] = 0;// 0 = do nothing
■■m_IngredientDestroy[0] = false;//true = destroy, false = do nothing
■■m_IngredientUseSoftSkills[0] = true;// set 'true' to allow modification of the values by softskills on this in
■■
■■//ingredient 2
■■InsertIngredient(1,"Inventory_Base");//you can insert multiple ingredients this way
■■
■■m_IngredientAddHealth[1] = 0;// 0 = do nothing
■■m_IngredientSetHealth[1] = -1; // -1 = do nothing
■■m_IngredientAddQuantity[1] = 0;// 0 = do nothing
■■m_IngredientDestroy[1] = false;// false = do nothing
■■m_IngredientUseSoftSkills[1] = false;// set 'true' to allow modification of the values by softskills on this i
■■
■■//----------------------------------------------------------------------------------------------------------------------
■■
■■//result1
■■//AddResult("");//add results here

■■m_ResultSetFullQuantity[0] = false;//true = set full quantity, false = do nothing
■■m_ResultSetQuantity[0] = -1;//-1 = do nothing
■■m_ResultSetHealth[0] = -1;//-1 = do nothing
■■m_ResultInheritsHealth[0] = -1;// (value) == -1 means do nothing; a (value) >= 0 means this result will i
■■m_ResultInheritsColor[0] = -1;// (value) == -1 means do nothing; a (value) >= 0 means this result classn
■■m_ResultToInventory[0] = -2;//(value) == -2 spawn result on the ground;(value) == -1 place anywhere in
■■m_ResultUseSoftSkills[0] = false;// set 'true' to allow modification of the values by softskills on this resu
■■m_ResultReplacesIngredient[0] = -1;// value == -1 means do nothing; a value >= 0 means this result wi

■■//----------------------------------------------------------------------------------------------------------------------
```

```
// ---------------------------------------------
// ------ RepairEpoxy.c - START --------------------
// ---------------------------------------------


class RepairEpoxy extends RecipeBase
{■
■override void Init()
■{
■■m_Name = "#repair";
■■m_IsInstaRecipe = false;//should this recipe be performed instantly without animation
■■m_AnimationLength = 1;//animation length in relative time units
■■m_Specialty = 0.03;// value > 0 for roughness, value < 0 for precision
■■
■■
■■//conditions
■■m_MinDamageIngredient[0] = -1;//-1 = disable check
■■m_MaxDamageIngredient[0] = 3;//-1 = disable check
■■
■■m_MinQuantityIngredient[0] = -1;//-1 = disable check
■■m_MaxQuantityIngredient[0] = -1;//-1 = disable check
■■
■■m_MinDamageIngredient[1] = 1;//-1 = disable check
■■m_MaxDamageIngredient[1] = 3;//-1 = disable check
■■
■■m_MinQuantityIngredient[1] = -1;//-1 = disable check
■■m_MaxQuantityIngredient[1] = -1;//-1 = disable check
■■//----------------------------------------------------------------------------------------------------------------------
■■
■■//INGREDIENTS
■■//ingredient 1
■■InsertIngredient(0,"EpoxyPutty");//you can insert multiple ingredients this way
■■
■■m_IngredientAddHealth[0] = 0;// 0 = do nothing
■■m_IngredientSetHealth[0] = -1; // -1 = do nothing
■■m_IngredientAddQuantity[0] = 0;// 0 = do nothing
■■m_IngredientDestroy[0] = false;//true = destroy, false = do nothing
■■m_IngredientUseSoftSkills[0] = true;// set 'true' to allow modification of the values by softskills on this in
■■
■■//ingredient 2
■■InsertIngredient(1,"Inventory_Base");//you can insert multiple ingredients this way
■■
■■m_IngredientAddHealth[1] = 0;// 0 = do nothing
■■m_IngredientSetHealth[1] = -1; // -1 = do nothing
■■m_IngredientAddQuantity[1] = 0;// 0 = do nothing
■■m_IngredientDestroy[1] = false;// false = do nothing
■■m_IngredientUseSoftSkills[1] = false;// set 'true' to allow modification of the values by softskills on this i
■■
■■//----------------------------------------------------------------------------------------------------------------------
■■
■■//result1
■■//AddResult("");//add results here

■■m_ResultSetFullQuantity[0] = false;//true = set full quantity, false = do nothing
■■m_ResultSetQuantity[0] = -1;//-1 = do nothing
■■m_ResultSetHealth[0] = -1;//-1 = do nothing
■■m_ResultInheritsHealth[0] = -1;// (value) == -1 means do nothing; a (value) >= 0 means this result will i
■■m_ResultInheritsColor[0] = -1;// (value) == -1 means do nothing; a (value) >= 0 means this result class
■■m_ResultToInventory[0] = -2;//(value) == -2 spawn result on the ground;(value) == -1 place anywhere i
■■m_ResultUseSoftSkills[0] = false;// set 'true' to allow modification of the values by softskills on this resu
■■m_ResultReplacesIngredient[0] = -1;// value == -1 means do nothing; a value >= 0 means this result wi

■■//----------------------------------------------------------------------------------------------------------------------
```

```
// ---------------------------------------------
// ------ RepairEyePatch.c - START --------------------
// ---------------------------------------------


class RepairEyePatch extends RecipeBase
{■
■override void Init()
■{
■■m_Name = "#repair";
■■m_IsInstaRecipe = false;//should this recipe be performed instantly without animation
■■m_AnimationLength = 1;//animation length in relative time units
■■m_Specialty = 0.03;// value > 0 for roughness, value < 0 for precision
■■
■■
■■//conditions
■■m_MinDamageIngredient[0] = 1;//-1 = disable check
■■m_MaxDamageIngredient[0] = 3;//-1 = disable check
■■
■■m_MinQuantityIngredient[0] = -1;//-1 = disable check
■■m_MaxQuantityIngredient[0] = -1;//-1 = disable check
■■
■■m_MinDamageIngredient[1] = -1;//-1 = disable check
■■m_MaxDamageIngredient[1] = 3;//-1 = disable check
■■
■■m_MinQuantityIngredient[1] = 1;//-1 = disable check
■■m_MaxQuantityIngredient[1] = -1;//-1 = disable check
■■//----------------------------------------------------------------------------------------------------------------------
■■
■■//INGREDIENTS
■■//ingredient 1
■■InsertIngredient(0,"EyePatch_Improvised");//you can insert multiple ingredients this way
■■
■■m_IngredientAddHealth[0] = 0;// 0 = do nothing
■■m_IngredientSetHealth[0] = -1; // -1 = do nothing
■■m_IngredientAddQuantity[0] = 0;// 0 = do nothing
■■m_IngredientDestroy[0] = false;//true = destroy, false = do nothing
■■m_IngredientUseSoftSkills[0] = false;// set 'true' to allow modification of the values by softskills on this i
■■
■■//ingredient 2
■■InsertIngredient(1,"Rag");//you can insert multiple ingredients this way
■■
■■m_IngredientAddHealth[1] = 0;// 0 = do nothing
■■m_IngredientSetHealth[1] = -1; // -1 = do nothing
■■m_IngredientAddQuantity[1] = -1;// 0 = do nothing
■■m_IngredientDestroy[1] = false;// false = do nothing
■■m_IngredientUseSoftSkills[1] = false;// set 'true' to allow modification of the values by softskills on this i
■■
■■//----------------------------------------------------------------------------------------------------------------------
■■
■■//result1
■■//AddResult("");//add results here

■■m_ResultSetFullQuantity[0] = false;//true = set full quantity, false = do nothing
■■m_ResultSetQuantity[0] = -1;//-1 = do nothing
■■m_ResultSetHealth[0] = -1;//-1 = do nothing
■■m_ResultInheritsHealth[0] = -1;// (value) == -1 means do nothing; a (value) >= 0 means this result will i
■■m_ResultInheritsColor[0] = -1;// (value) == -1 means do nothing; a (value) >= 0 means this result class
■■m_ResultToInventory[0] = -2;//(value) == -2 spawn result on the ground;(value) == -1 place anywhere i
■■m_ResultUseSoftSkills[0] = false;// set 'true' to allow modification of the values by softskills on this resu
■■m_ResultReplacesIngredient[0] = -1;// value == -1 means do nothing; a value >= 0 means this result wi

■■//----------------------------------------------------------------------------------------------------------------------
```

```
// ---------------------------------------------
// ------ RepairPlanks.c - START --------------------
// ---------------------------------------------


class RepairPlanks extends RecipeBase
{■
■override void Init()
■{
■■m_Name = "#repair";
■■m_IsInstaRecipe = false;//should this recipe be performed instantly without animation
■■m_AnimationLength = 1;//animation length in relative time units
■■m_Specialty = 0.03;// value > 0 for roughness, value < 0 for precision
■■
■■
■■//conditions
■■m_MinDamageIngredient[0] = -1;//-1 = disable check
■■m_MaxDamageIngredient[0] = 3;//-1 = disable check
■■
■■m_MinQuantityIngredient[0] = 2;//-1 = disable check
■■m_MaxQuantityIngredient[0] = -1;//-1 = disable check
■■
■■m_MinDamageIngredient[1] = 2;//-1 = disable check
■■m_MaxDamageIngredient[1] = 3;//-1 = disable check
■■
■■m_MinQuantityIngredient[1] = -1;//-1 = disable check
■■m_MaxQuantityIngredient[1] = -1;//-1 = disable check
■■//----------------------------------------------------------------------------------------------------------------------
■■
■■//INGREDIENTS
■■//ingredient 1
■■InsertIngredient(0,"WoodenPlank");//you can insert multiple ingredients this way
■■
■■m_IngredientAddHealth[0] = 0;// 0 = do nothing
■■m_IngredientSetHealth[0] = -1; // -1 = do nothing
■■m_IngredientAddQuantity[0] = 0;// 0 = do nothing
■■m_IngredientDestroy[0] = false;//true = destroy, false = do nothing
■■m_IngredientUseSoftSkills[0] = true;// set 'true' to allow modification of the values by softskills on this in
■■
■■//ingredient 2
■■InsertIngredient(1,"WoodenCrate");//you can insert multiple ingredients this way
■■InsertIngredient(1,"SeaChest");//you can insert multiple ingredients this way
■■
■■m_IngredientAddHealth[1] = 0;// 0 = do nothing
■■m_IngredientSetHealth[1] = -1; // -1 = do nothing
■■m_IngredientAddQuantity[1] = 0;// 0 = do nothing
■■m_IngredientDestroy[1] = false;// false = do nothing
■■m_IngredientUseSoftSkills[1] = false;// set 'true' to allow modification of the values by softskills on this i
■■
■■//----------------------------------------------------------------------------------------------------------------------
■■
■■//result1
■■//AddResult("");//add results here

■■m_ResultSetFullQuantity[0] = false;//true = set full quantity, false = do nothing
■■m_ResultSetQuantity[0] = -1;//-1 = do nothing
■■m_ResultSetHealth[0] = -1;//-1 = do nothing
■■m_ResultInheritsHealth[0] = -1;// (value) == -1 means do nothing; a (value) >= 0 means this result will i
■■m_ResultInheritsColor[0] = -1;// (value) == -1 means do nothing; a (value) >= 0 means this result class
■■m_ResultToInventory[0] = -2;//(value) == -2 spawn result on the ground;(value) == -1 place anywhere i
■■m_ResultUseSoftSkills[0] = false;// set 'true' to allow modification of the values by softskills on this resu
■■m_ResultReplacesIngredient[0] = -1;// value == -1 means do nothing; a value >= 0 means this result wi
```

```
// ---------------------------------------------
// ------ RepairWithRags.c - START --------------------
// ---------------------------------------------


class RepairWithRags extends RecipeBase
{■
■override void Init()
■{
■■m_Name = "#repair";
■■m_IsInstaRecipe = false;//should this recipe be performed instantly without animation
■■m_AnimationLength = 1;//animation length in relative time units
■■m_Specialty = 0.03;// value > 0 for roughness, value < 0 for precision
■■
■■
■■//conditions
■■m_MinDamageIngredient[0] = 1;//-1 = disable check
■■m_MaxDamageIngredient[0] = 3;//-1 = disable check
■■
■■m_MinQuantityIngredient[0] = -1;//-1 = disable check
■■m_MaxQuantityIngredient[0] = -1;//-1 = disable check
■■
■■m_MinDamageIngredient[1] = -1;//-1 = disable check
■■m_MaxDamageIngredient[1] = 3;//-1 = disable check
■■
■■m_MinQuantityIngredient[1] = 1;//-1 = disable check
■■m_MaxQuantityIngredient[1] = -1;//-1 = disable check
■■//----------------------------------------------------------------------------------------------------------------------
■■
■■//INGREDIENTS
■■//ingredient 1
■■InsertIngredient(0,"HeadCover_Improvised");//you can insert multi■ple ingredients this way
■■InsertIngredient(0,"HandsCover_Improvised");//you can insert multiple ingredients this way
■■InsertIngredient(0,"LegsCover_Improvised");//you can insert multiple ingredients this way
■■InsertIngredient(0,"FeetCover_Improvised");//you can insert multiple ingredients this way
■■InsertIngredient(0,"FaceCover_Improvised");//you can insert multiple ingredients this way
■■InsertIngredient(0,"TorsoCover_Improvised");//you can insert multiple ingredients this way
■■
■■m_IngredientAddHealth[0] = 0;// 0 = do nothing
■■m_IngredientSetHealth[0] = -1; // -1 = do nothing
■■m_IngredientAddQuantity[0] = 0;// 0 = do nothing
■■m_IngredientDestroy[0] = false;//true = destroy, false = do nothing
■■m_IngredientUseSoftSkills[0] = false;// set 'true' to allow modification of the values by softskills on this i
■■
■■//ingredient 2
■■InsertIngredient(1,"Rag");//you can insert multiple ingredients this way
■■
■■m_IngredientAddHealth[1] = 0;// 0 = do nothing
■■m_IngredientSetHealth[1] = -1; // -1 = do nothing
■■m_IngredientAddQuantity[1] = -1;// 0 = do nothing
■■m_IngredientDestroy[1] = false;// false = do nothing
■■m_IngredientUseSoftSkills[1] = false;// set 'true' to allow modification of the values by softskills on this i
■■
■■//----------------------------------------------------------------------------------------------------------------------
■■
■■//result1
■■//AddResult("");//add results here

■■m_ResultSetFullQuantity[0] = false;//true = set full quantity, false = do nothing
■■m_ResultSetQuantity[0] = -1;//-1 = do nothing
■■m_ResultSetHealth[0] = -1;//-1 = do nothing
■■m_ResultInheritsHealth[0] = -1;// (value) == -1 means do nothing; a (value) >= 0 means this result will i
■■m_ResultInheritsColor[0] = -1;// (value) == -1 means do nothing; a (value) >= 0 means this result class
```

```
// ---------------------------------------------
// ------ RepairWithTape.c - START --------------------
// ---------------------------------------------


class RepairWithTape extends RecipeBase
{■
■override void Init()
■{
■■m_Name = "#repair";
■■m_IsInstaRecipe = false;//should this recipe be performed instantly without animation
■■m_AnimationLength = 1;//animation length in relative time units
■■m_Specialty = 0.03;// value > 0 for roughness, value < 0 for precision
■■
■■
■■//conditions
■■m_MinDamageIngredient[0] = -1;//-1 = disable check
■■m_MaxDamageIngredient[0] = 3;//-1 = disable check
■■
■■m_MinQuantityIngredient[0] = -1;//-1 = disable check
■■m_MaxQuantityIngredient[0] = -1;//-1 = disable check
■■
■■m_MinDamageIngredient[1] = 1;//-1 = disable check
■■m_MaxDamageIngredient[1] = 3;//-1 = disable check
■■
■■m_MinQuantityIngredient[1] = -1;//-1 = disable check
■■m_MaxQuantityIngredient[1] = -1;//-1 = disable check
■■//----------------------------------------------------------------------------------------------------------------------
■■
■■//INGREDIENTS
■■//ingredient 1
■■InsertIngredient(0,"DuctTape");//you can insert multiple ingredients this way
■■
■■m_IngredientAddHealth[0] = 0;// 0 = do nothing
■■m_IngredientSetHealth[0] = -1; // -1 = do nothing
■■m_IngredientAddQuantity[0] = 0;// 0 = do nothing
■■m_IngredientDestroy[0] = false;//true = destroy, false = do nothing
■■m_IngredientUseSoftSkills[0] = true;// set 'true' to allow modification of the values by softskills on this in
■■
■■//ingredient 2
■■InsertIngredient(1,"Inventory_Base");//you can insert multiple ingredients this way
■■InsertIngredient(1,"DefaultWeapon");//you can insert multiple ingredients this way
■■InsertIngredient(1,"DefaultMagazine");//you can insert multiple ingredients this way
■■
■■m_IngredientAddHealth[1] = 0;// 0 = do nothing
■■m_IngredientSetHealth[1] = -1; // -1 = do nothing
■■m_IngredientAddQuantity[1] = 0;// 0 = do nothing
■■m_IngredientDestroy[1] = false;// false = do nothing
■■m_IngredientUseSoftSkills[1] = false;// set 'true' to allow modification of the values by softskills on this i
■■
■■//----------------------------------------------------------------------------------------------------------------------
■■
■■//result1
■■//AddResult("");//add results here

■■m_ResultSetFullQuantity[0] = false;//true = set full quantity, false = do nothing
■■m_ResultSetQuantity[0] = -1;//-1 = do nothing
■■m_ResultSetHealth[0] = -1;//-1 = do nothing
■■m_ResultInheritsHealth[0] = -1;// (value) == -1 means do nothing; a (value) >= 0 means this result will i
■■m_ResultInheritsColor[0] = -1;// (value) == -1 means do nothing; a (value) >= 0 means this result classn
■■m_ResultToInventory[0] = -2;//(value) == -2 spawn result on the ground;(value) == -1 place anywhere in
■■m_ResultUseSoftSkills[0] = false;// set 'true' to allow modification of the values by softskills on this resu
■■m_ResultReplacesIngredient[0] = -1;// value == -1 means do nothing; a value >= 0 means this result wi
```

```
// ---------------------------------------------
// ------ Repeater.c - START --------------------
// ---------------------------------------------


enum RPTAnimState
{
■DEFAULT ■■■= 0, ■///< default weapon state, closed and discharged
■CHARGED ■■■= 1,
■JAMMED ■■■■= 2,
};

enum RPTStableStateID
{
■UNKNOWN■■■■=  0,
■EmptyDischarged■■=  1,
■LoadedCharged■■=  2,
■LoadedDischarged■=  3,
■LoadedJammed■■=  4,
}


class RPTEmptyDischarged extends WeaponStableState
{
■override void OnEntry (WeaponEventBase e) { if (LogManager.IsWeaponLogEnable()) { wpnPrint("[wpnf
■override void OnExit (WeaponEventBase e) { super.OnExit(e); if (LogManager.IsWeaponLogEnable()) { v
■override int GetCurrentStateID () { return RPTStableStateID.EmptyDischarged; }
■override bool HasBullet () { return false; }
■override bool HasMagazine () { return false; }
■override bool IsJammed () { return false; }
■override bool IsRepairEnabled () { return true; }
■override void InitMuzzleArray () { m_muzzleHasBullet = {MuzzleState.E}; }
};
class RPTLoadedCharged extends WeaponStableState
{
■override void OnEntry (WeaponEventBase e) { if (LogManager.IsWeaponLogEnable()) { wpnPrint("[wpnf
■override void OnExit (WeaponEventBase e) { super.OnExit(e); if (LogManager.IsWeaponLogEnable()) { v
■override int GetCurrentStateID () { return RPTStableStateID.LoadedCharged; }
■override bool HasBullet () { return true; }
■override bool HasMagazine () { return false; }
■override bool IsJammed () { return false; }
■override bool IsRepairEnabled () { return true; }
■override void InitMuzzleArray () { m_muzzleHasBullet = {MuzzleState.L}; }
};
class RPTLoadedDischarged extends WeaponStableState
{
■override void OnEntry (WeaponEventBase e) { if (LogManager.IsWeaponLogEnable()) { wpnPrint("[wpnf
■override void OnExit (WeaponEventBase e) { super.OnExit(e); if (LogManager.IsWeaponLogEnable()) { v
■override int GetCurrentStateID () { return RPTStableStateID.LoadedDischarged; }
■override bool HasBullet () { return true; }
■override bool HasMagazine () { return false; }
■override bool IsJammed () { return false; }
■override bool IsRepairEnabled () { return true; }
■override void InitMuzzleArray () { m_muzzleHasBullet = {MuzzleState.F}; }
};
class RPTLoadedJammed extends WeaponStateJammed
{
■override void OnEntry (WeaponEventBase e) { if (LogManager.IsWeaponLogEnable()) { wpnPrint("[wpnf
■override void OnExit (WeaponEventBase e) { super.OnExit(e); if (LogManager.IsWeaponLogEnable()) { v
■override int GetCurrentStateID () { return RPTStableStateID.LoadedJammed; }
■override bool HasBullet () { return true; }
■override bool HasMagazine () { return false; }
■override bool IsJammed () { return true; }
```

```
// ---------------------------------------------
// ------ RepeaterRecoil.c - START --------------------
// ---------------------------------------------


class RepeaterRecoil: RecoilBase
{
■override void Init()
■{
■■vector point_1;
■■vector point_2;
■■vector point_3;
■■vector point_4;
■■point_1[0] = m_Player.GetRandomGeneratorSyncManager().GetRandomInRange(RandomGeneratorS
■■point_2[0] = m_Player.GetRandomGeneratorSyncManager().GetRandomInRange(RandomGeneratorS
■■point_3[0] = m_Player.GetRandomGeneratorSyncManager().GetRandomInRange(RandomGeneratorS
■■point_4[0] = m_Player.GetRandomGeneratorSyncManager().GetRandomInRange(RandomGeneratorS
■■m_HandsCurvePoints.Insert(point_1);//forms a 2 dimensional spline(z is ignored)
■■m_HandsCurvePoints.Insert(point_2);
■■m_HandsCurvePoints.Insert(point_3);
■■m_HandsCurvePoints.Insert(point_4);
■■m_HandsCurvePoints.Insert("0 0 0");
■■m_HandsOffsetRelativeTime = 0.125;■
■■
■■m_MouseOffsetRangeMin = 80;//in degrees min
■■m_MouseOffsetRangeMax = 100;//in degrees max
■■m_MouseOffsetDistance = 1.4;//how far should the mouse travel
■■m_MouseOffsetRelativeTime = 0.0625;//[0..1] a time it takes to move the mouse the required distance
■■■
■■m_CamOffsetDistance = 0.02;
■■m_CamOffsetRelativeTime = 0.125;
■}
}


// ---------------------------------------------
// ------ RepeaterRecoil.c - END ---------------------
// ---------------------------------------------



// ---------------------------------------------
// ------ ReplaceItemWithNewLambda.c - START --------------------
// ---------------------------------------------


/**@class■■ReplaceItemWithNewLambda
 * @brief■■adds automatic QuickBar handling
 **/
class ReplaceItemWithNewLambda : ReplaceItemWithNewLambdaBase
{
■PlayerBase m_Player;
■int m_IndexQB;

■void ReplaceItemWithNewLambda(EntityAI old_item, string new_item_type, PlayerBase player)
■{
■■m_Player = player;
■■m_IndexQB = -1;

■■if (m_Player)
■■■m_IndexQB = m_Player.FindQuickBarEntityIndex(old_item);
■}

■/**@fn■■CopyOldPropertiesToNew
```

```
// ---------------------------------------------
// ------ ReplaceItemWithNewLambdaBase.c - START --------------------
// ---------------------------------------------


/**@class■■ReplaceItemWithNewLambdaBase
 * @brief■■base class for transformation operations (creating one item from another)
 **/
class ReplaceItemWithNewLambdaBase
{
■EntityAI m_OldItem;
■string m_NewItemType;
■protected ref InventoryLocation m_OldLocation;
■protected ref InventoryLocation m_NewLocation;
■protected bool m_RemoveFromLocationPassed = false;
■private bool m_RemoveNetworkObjectInfoPassed = false;

■void ReplaceItemWithNewLambdaBase(EntityAI old_item, string new_item_type)
■{
■■m_OldItem = old_item;
■■m_NewItemType = new_item_type;
■}

■void OverrideNewLocation(InventoryLocation newLocation)
■{
■■m_NewLocation = newLocation;
■}
■
■void VerifyItemTypeBySlotType() {}

■protected bool WantCreateNewEntity()
■{
■■return m_NewLocation && m_NewItemType != string.Empty;
■}

■protected bool CanExecuteLambda()
■{
■■if (m_OldItem)
■■■if (GameInventory.LocationCanRemoveEntity(m_OldLocation))
■■■■//if (WantCreateNewEntity() GameInventory.LocationTestAddEntityType(m_NewItemType, true, tru
■■■■■return true;
■■return false;
■}

■/**@fn■■PrepareLocations
■ * @brief■Step A. - prepare inventory locations
■ **/
■protected bool PrepareLocations()
■{
■■hndDebugPrint("[inv] ReplaceItemWithNewLambdaBase Step A) Prepare inventory locations, old_item
■■m_OldLocation = new InventoryLocation;
■■if (m_OldItem.GetInventory().GetCurrentInventoryLocation(m_OldLocation)) // A.1) store old location
■■{
■■■if (m_NewLocation == null)
■■■{
■■■■m_NewLocation = new InventoryLocation;
■■■■m_NewLocation.CopyLocationFrom(m_OldLocation, true);■// A.2) prepare new location from old
■■■■
■■■■//setting proper location type for ground pos
■■■■if (!m_NewLocation.GetParent())
■■■■{
■■■■■vector m4[4];
■■■■■Math3D.MatrixIdentity4(m4);
```

```
// ---------------------------------------------
// ------ ResaveTool.c - START --------------------
// ---------------------------------------------


[WorkbenchPluginAttribute("Re-Save Tool", "Saves all files with given extension", "", "", {"ResourceManag
class ResavePlugin: WorkbenchPlugin
{
■[Attribute(".layout", "editbox", "File extension" )]
■string Extension;
■WBModuleDef m_module;
■
■void Resave(string file)
■{
■■Print("Resaving: " + file);
■■m_module.SetOpenedResource(file);■
■■m_module.Save();
■}
■
■override void Run()
■{
■■if (Workbench.ScriptDialog("Resave", "Which files you want to resave?", this))
■■{
■■■m_module = Workbench.GetModule("ResourceManager");
■■■Workbench.SearchResources(Extension, Resave);
■■}
■}
■
■override void RunCommandline()
■{
■■m_module = Workbench.GetModule("ResourceManager");
■■
■■if (m_module.GetCmdLine("-extension", Extension))
■■{
■■■Extension.Replace("\"", "");
■■}
■■
■■Workbench.SearchResources(Extension, Resave);
■■Workbench.GetModule("ResourceManager").Close();
■}
■
■[ButtonAttribute("Re-Save")]
■bool OK()
■{
■■return true;
■}
■
■[ButtonAttribute("Cancel")]
■bool Cancel()
■{
■■return false;
■}
};


// ---------------------------------------------
// ------ ResaveTool.c - END ----------------------
// ---------------------------------------------
// ---------------------------------------------
// ------ RespawnDialogue.c - START --------------------
// ---------------------------------------------


class RespawnDialogue extends UIScriptedMenu
{
■const int ■■■■■ID_RESPAWN_CUSTOM = 101;
■const int ■■■■■ID_RESPAWN_RANDOM = 102;
■
■//tooltips
■protected Widget■■■m_DetailsRoot;
■protected TextWidget■■m_DetailsLabel;
■protected RichTextWidget■m_DetailsText;
■
■protected Widget ■■■m_CustomRespawn;
■
■//helper
■protected Widget ■■■m_CurrentlyHighlighted;
■
■void RespawnDialogue();
■void ~RespawnDialogue();
■
■override Widget Init()
■{
■■layoutRoot ■■■■■= GetGame().GetWorkspace().CreateWidgets("gui/layouts/day_z_respawn_dialogu
■■m_DetailsRoot ■■■■= layoutRoot.FindAnyWidget("menu_details_tooltip");
■■m_DetailsLabel■■■■= TextWidget.Cast(m_DetailsRoot.FindAnyWidget("menu_details_label"));
■■m_DetailsText■■■■= RichTextWidget.Cast(m_DetailsRoot.FindAnyWidget("menu_details_tooltip_cor
■■
■■m_CustomRespawn ■■■= layoutRoot.FindAnyWidget("respawn_button_custom");
■■SetFocus(m_CustomRespawn);

■■return layoutRoot;
■}
■
■override void Update(float timeslice)
■{
■■super.Update(timeslice);
■■
■■if (GetUApi().GetInputByID(UAUIBack).LocalPress() || GetUApi().GetInputByID(UAUIMenu).LocalPress
■■■Close();
■}

■override bool OnClick(Widget w, int x, int y, int button)
■{
■■super.OnClick(w, x, y, button);
■■
■■switch (w.GetUserID())
■■{
■■■case IDC_CANCEL:
■■■■Close();
■■■■return true;

■■■case ID_RESPAWN_CUSTOM:
■■■■return RequestRespawn(false);

■■■case ID_RESPAWN_RANDOM:
■■■■return RequestRespawn(true);
■■}

■■return false;
■}
```

```
// ---------------------------------------------
// ------ RestApi.c - START --------------------
// ---------------------------------------------



/** @file */


// -----------------------------------------------------------------------
// states, (result + error) codes
// defined in C++
enum ERestResultState
{
■EREST_EMPTY,■■■■// not initialized
■EREST_PENDING,■■■■// awaiting processing
■EREST_FEEDING,■■■■// awaiting incoming data
■EREST_SUCCESS,■■■■// result and/ or data are ready (success), awaiting data processing to be finis
■EREST_PROCESSED,■■■// finished (either successfully or with failure) and eill be removed ASAP

■EREST_ERROR,■■■■// (state >= EREST_ERROR) == error happened
■EREST_ERROR_CLIENTERROR,■//  (EREST_ERROR == EREST_ERROR_CLIENTERROR)
■EREST_ERROR_SERVERERROR,
■EREST_ERROR_APPERROR,
■EREST_ERROR_TIMEOUT,
■EREST_ERROR_NOTIMPLEMENTED,
■EREST_ERROR_UNKNOWN,
};

// -----------------------------------------------------------------------
// options
// defined in C++
enum ERestOption
{
■ERESTOPTION_UNKNOWN,■■■■■// invalid option

■ERESTOPTION_READOPERATION,■■// read operation timeout (default 10sec)
■ERESTOPTION_CONNECTION,■■■// connection timeout (default 10sec)
■// note: limit for timeout is between <3 .. 120> seconds, you cannot exceed this value
};



// -----------------------------------------------------------------------
// object to be used from script for result binding
//
//■[Example:]
//
//■■RestCallback cbx1 = new RestCallback;
//■■RestContext ctx = GetRestApi().GetRestContext("http://somethingsomewhere.com/path/");
//■■ctx.GET(cbx1,"RequestPath?Argument=Something");
//
//■■Event are then called upon RestCallback()
//
class RestCallback : Managed
{
■/**
■\brief Called in case request failed (ERestResultState) - Note! May be called multiple times in case of (Re
■*/
■void OnError( int errorCode )
■{
■■// override this with your implementation
■■Print(" !!! OnError() ");
```

```
// ---------------------------------------------
// ------ RGD5Grenade.c - START --------------------
// ---------------------------------------------


class RGD5Grenade extends Grenade_Base
{
■void RGD5Grenade()
■{
■■SetAmmoType("RGD5Grenade_Ammo");
■■SetFuseDelay(4);
■■SetParticleExplosion(ParticleList.RGD5);
■}

■void ~RGD5Grenade() {}
}


// ---------------------------------------------
// ------ RGD5Grenade.c - END ----------------------
// ---------------------------------------------


// ---------------------------------------------
// ------ Rice.c - START --------------------
// ---------------------------------------------


class Rice: Edible_Base
{
■override void SetActions()
■{
■■super.SetActions();
■■
■■AddAction(ActionForceFeed);
■■AddAction(ActionEatBig);
■}
};



// ---------------------------------------------
// ------ Rice.c - END ----------------------
// ---------------------------------------------


// ---------------------------------------------
// ------ RidersJacket_ColorBase.c - START --------------------
// ---------------------------------------------


class RidersJacket_ColorBase extends Clothing
{
■override void SetActions()
■{
■■super.SetActions();
■■AddAction(ActionWringClothes);
■}
};
class RidersJacket_Black extends RidersJacket_ColorBase {};


// ---------------------------------------------
// ---------------------------------------------
// ------ RifleBoltFree_Base.c - START --------------------
// ---------------------------------------------


enum RBFAnimState
{
■DEFAULT ■■■= 0, ■///< default weapon state, closed and discharged
■JAMMED ■■■■= 1,
};

enum RBFStableStateID
{
■UNKNOWN■■■■=  0,
■RBF_CLO_BU0_MA0■■=  1,
■RBF_CLO_BU1_MA0■■=  2,
■RBF_CLO_BU1_MA1■■=  3,
■RBF_CLO_BU0_MA1■■=  4,
■RBF_JAM_BU1_MA0■■=  5,
■RBF_JAM_BU1_MA1■■=  6
}

class RBF_CLO_BU0_MA0 extends WeaponStableState
{
■override void OnEntry (WeaponEventBase e) { if (LogManager.IsWeaponLogEnable()) { wpnPrint("[wpnf
■override void OnExit (WeaponEventBase e) { super.OnExit(e); if (LogManager.IsWeaponLogEnable()) { w
■override int GetCurrentStateID () { return RBFStableStateID.RBF_CLO_BU0_MA0; }
■override bool HasBullet () { return false; }
■override bool HasMagazine () { return false; }
■override bool IsJammed () { return false; }
■override bool IsRepairEnabled () { return true; }
■override void InitMuzzleArray () { m_muzzleHasBullet = {MuzzleState.E}; }
};
class RBF_CLO_BU1_MA0 extends WeaponStableState
{
■override void OnEntry (WeaponEventBase e) { if (LogManager.IsWeaponLogEnable()) { wpnPrint("[wpnf
■override void OnExit (WeaponEventBase e) { super.OnExit(e); if (LogManager.IsWeaponLogEnable()) { w
■override int GetCurrentStateID () { return RBFStableStateID.RBF_CLO_BU1_MA0; }
■override bool HasBullet () { return true; }
■override bool HasMagazine () { return false; }
■override bool IsJammed () { return false; }
■override bool IsRepairEnabled () { return true; }
■override void InitMuzzleArray () { m_muzzleHasBullet = {MuzzleState.L}; }
};
class RBF_CLO_BU1_MA1 extends WeaponStableState
{
■override void OnEntry (WeaponEventBase e) { /*Print("[wpnfsm] " + Object.GetDebugName(m_weapon)
■override void OnExit (WeaponEventBase e) { super.OnExit(e); if (LogManager.IsWeaponLogEnable()) { w
■override int GetCurrentStateID () { return RBFStableStateID.RBF_CLO_BU1_MA1; }
■override bool HasBullet () { return true; }
■override bool HasMagazine () { return true; }
■override bool IsJammed () { return false; }
■override bool IsRepairEnabled () { return true; }
■override void InitMuzzleArray () { m_muzzleHasBullet = {MuzzleState.L}; }
};
class RBF_CLO_BU0_MA1 extends WeaponStableState
{
■override void OnEntry (WeaponEventBase e) { if (LogManager.IsWeaponLogEnable()) { wpnPrint("[wpnf
■override void OnExit (WeaponEventBase e) { super.OnExit(e); if (LogManager.IsWeaponLogEnable()) { w
■override int GetCurrentStateID () { return RBFStableStateID.RBF_CLO_BU0_MA1; }
■override bool HasBullet () { return false; }
■override bool HasMagazine () { return true; }
■override bool IsJammed () { return false; }
```

```
// ---------------------------------------------
// ------ RifleBoltLock_Base.c - START --------------------
// ---------------------------------------------


enum RBLAnimState
{
■DEFAULT ■■■= 0, ■///< default weapon state, closed and discharged
■OPENED ■■■■= 1,
■JAMMED ■■■■= 2,
};

enum RBLStableStateID
{
■UNKNOWN■■■■=  0,
■RBL_CLO_BU0_MA0■■=  1,
■RBL_CLO_BU1_MA0■■=  2,
■RBL_CLO_BU1_MA1■■=  3,
■RBL_CLO_BU0_MA1■■=  4,
■RBL_OPN_BU0_MA1■■=  5,
■RBL_OPN_BU0_MA0■■=  6,
■RBL_JAM_BU1_MA0■■=  7,
■RBL_JAM_BU1_MA1■■=  8
}

class RBL_CLO_BU0_MA0 extends WeaponStableState
{
■override void OnEntry (WeaponEventBase e) { if (LogManager.IsWeaponLogEnable()) { wpnPrint("[wpnf
■override void OnExit (WeaponEventBase e) { super.OnExit(e); if (LogManager.IsWeaponLogEnable()) { 
■override int GetCurrentStateID () { return RBLStableStateID.RBL_CLO_BU0_MA0; }
■override bool HasBullet () { return false; }
■override bool HasMagazine () { return false; }
■override bool IsJammed () { return false; }
■override bool IsRepairEnabled () { return true; }
■override void InitMuzzleArray () { m_muzzleHasBullet = {MuzzleState.E}; }
};
class RBL_CLO_BU1_MA0 extends WeaponStableState
{
■override void OnEntry (WeaponEventBase e) { if (LogManager.IsWeaponLogEnable()) { wpnPrint("[wpnf
■override void OnExit (WeaponEventBase e) { super.OnExit(e); if (LogManager.IsWeaponLogEnable()) { 
■override int GetCurrentStateID () { return RBLStableStateID.RBL_CLO_BU1_MA0; }
■override bool HasBullet () { return true; }
■override bool HasMagazine () { return false; }
■override bool IsJammed () { return false; }
■override bool IsRepairEnabled () { return true; }
■override void InitMuzzleArray () { m_muzzleHasBullet = {MuzzleState.L}; }
};
class RBL_CLO_BU1_MA1 extends WeaponStableState
{
■override void OnEntry (WeaponEventBase e) { if (LogManager.IsWeaponLogEnable()) { wpnPrint("[wpnf
■override void OnExit (WeaponEventBase e) { super.OnExit(e); if (LogManager.IsWeaponLogEnable()) { 
■override int GetCurrentStateID () { return RBLStableStateID.RBL_CLO_BU1_MA1; }
■override bool HasBullet () { return true; }
■override bool HasMagazine () { return true; }
■override bool IsJammed () { return false; }
■override bool IsRepairEnabled () { return true; }
■override void InitMuzzleArray () { m_muzzleHasBullet = {MuzzleState.L}; }
};
class RBL_CLO_BU0_MA1 extends WeaponStableState
{
■override void OnEntry (WeaponEventBase e) { if (LogManager.IsWeaponLogEnable()) { wpnPrint("[wpnf
■override void OnExit (WeaponEventBase e) { super.OnExit(e); if (LogManager.IsWeaponLogEnable()) { 
■override int GetCurrentStateID () { return RBLStableStateID.RBL_CLO_BU0_MA1; }
```

```
// ---------------------------------------------
// ------ RifleChambering.c - START --------------------
// ---------------------------------------------


class RifleChambering extends WeaponStateBase
{
■WeaponActions m_action;
■int m_actionType;
■Magazine m_srcMagazine; /// source of the cartridge

■ref WeaponStateBase m_start;
■ref WeaponEjectCasing m_eject;
■ref WeaponChambering_Cartridge m_chamber;
■ref WeaponChambering_W4T m_w4t;

■void RifleChambering (Weapon_Base w = NULL, WeaponStateBase parent = NULL, WeaponActions ac
■{
■■m_action = action;
■■m_actionType = actionType;

■■// setup nested state machine
■■m_start = new WeaponChambering_Start(m_weapon, this, m_action, m_actionType);
■■m_eject = new WeaponEjectCasing(m_weapon, this);
■■m_chamber = new WeaponChambering_Cartridge(m_weapon, this);
■■m_w4t = new WeaponChambering_W4T(m_weapon, this);
■■// events
■■WeaponEventAnimBulletEject __be_ = new WeaponEventAnimBulletEject;
■■WeaponEventAnimBulletShow  __bs_ = new WeaponEventAnimBulletShow;
■■WeaponEventAnimBulletInChamber __bc_ = new WeaponEventAnimBulletInChamber;
■■WeaponEventBase _fin_ = new WeaponEventHumanCommandActionFinished;

■■m_fsm = new WeaponFSM(this); // @NOTE: set owner of the submachine fsm
■■m_fsm.AddTransition(new WeaponTransition(m_start  , __be_, m_eject));
■■m_fsm.AddTransition(new WeaponTransition(m_eject  , __bs_, m_chamber));
■■m_fsm.AddTransition(new WeaponTransition(m_chamber, __bc_, m_w4t));
■■m_fsm.AddTransition(new WeaponTransition(m_w4t    , _fin_, NULL));
■■
■■// Safety exits
■■m_fsm.AddTransition(new WeaponTransition(m_chamber, _fin_, null));
■■m_fsm.AddTransition(new WeaponTransition(m_eject  , _fin_, null));
■■m_fsm.AddTransition(new WeaponTransition(m_start  , _fin_, null));■

■■m_fsm.SetInitialState(m_start);
■}

■override void OnEntry (WeaponEventBase e)
■{
■■if (e != NULL)
■■{
■■■if (e.m_magazine != NULL)
■■■{
■■■■if (LogManager.IsWeaponLogEnable()) { wpnDebugPrint("[wpnfsm] " + Object.GetDebugName(m_w
■■■■m_srcMagazine = e.m_magazine;
■■■■m_chamber.m_srcMagazine = m_srcMagazine;
■■■}
■■}
■■else
■■{
■■■if (LogManager.IsWeaponLogEnable()) { wpnDebugPrint("[wpnfsm] " + Object.GetDebugName(m_we
■■}
■■if (LogManager.IsWeaponLogEnable()) { wpnDebugPrint("[wpnfsm] " + Object.GetDebugName(m_wea
■■
```

```
// ---------------------------------------------
// ------ RifleEjectCasing.c - START --------------------
// ---------------------------------------------


class RifleEjectCasing extends WeaponStateBase
{
■WeaponActions m_action;
■int m_actionType;

■ref WeaponStartAction m_start;
■ref WeaponEjectCasing m_eject;
■ref BulletHide_W4T m_hideB;

■void RifleEjectCasing (Weapon_Base w = NULL, WeaponStateBase parent = NULL, WeaponActions act
■{
■■m_action = action;
■■m_actionType = actionType;

■■// setup nested state machine
■■m_start = new WeaponStartAction(m_weapon, this, m_action, m_actionType);
■■m_eject = new WeaponEjectCasing(m_weapon, this);
■■m_hideB = new BulletHide_W4T(m_weapon, this);

■■// events
■■WeaponEventBase __be_ = new WeaponEventAnimBulletEject;
■■WeaponEventBase __bh_ = new WeaponEventAnimBulletHide;
■■WeaponEventBase _fin_ = new WeaponEventHumanCommandActionFinished;

■■m_fsm = new WeaponFSM(this); // @NOTE: set owner of the submachine fsm

■■// transitions
■■m_fsm.AddTransition(new WeaponTransition(m_start, __be_, m_eject));
■■m_fsm.AddTransition(new WeaponTransition(m_eject, __bh_, m_hideB));
■■m_fsm.AddTransition(new WeaponTransition(m_hideB, _fin_, NULL));
■■
■■// Safety exits
■■m_fsm.AddTransition(new WeaponTransition(m_eject  , _fin_, null));
■■m_fsm.AddTransition(new WeaponTransition(m_start  , _fin_, null));■

■■m_fsm.SetInitialState(m_start);
■}
};





// ---------------------------------------------
// ------ RifleEjectCasing.c - END ----------------------
// ---------------------------------------------


// ---------------------------------------------
// ------ RifleReChambering.c - START --------------------
// ---------------------------------------------


class RifleReChambering extends WeaponStateBase
{
■WeaponActions m_action;
■int m_actionType;
```

```
// ---------------------------------------------
// ------ RifleSingleShotManual_Base.c - START --------------------
// ---------------------------------------------


/**@class■■Izh18_Base
 * @brief■■base for Izh18
 * @NOTE■■name copies config base class
 **/
class RifleSingleShotManual_Base extends RifleSingleShot_Base
{
■void RifleSingleShotManual_Base ()
■{
■}
■
■override void InitStateMachine()
■{
■■// setup abilities
■■m_abilities.Insert(new AbilityRecord(WeaponActions.MECHANISM, WeaponActionMechanismTypes.M
■■m_abilities.Insert(new AbilityRecord(WeaponActions.MECHANISM, WeaponActionMechanismTypes.M
■■m_abilities.Insert(new AbilityRecord(WeaponActions.CHAMBERING, WeaponActionChamberingTypes
■■m_abilities.Insert(new AbilityRecord(WeaponActions.CHAMBERING, WeaponActionChamberingTypes
■■m_abilities.Insert(new AbilityRecord(WeaponActions.UNJAMMING, WeaponActionUnjammingTypes.U
■■m_abilities.Insert(new AbilityRecord(WeaponActions.UNJAMMING, WeaponActionUnjammingTypes.U
■■m_abilities.Insert(new AbilityRecord(WeaponActions.FIRE, WeaponActionFireTypes.FIRE_NORMAL))
■■m_abilities.Insert(new AbilityRecord(WeaponActions.FIRE, WeaponActionFireTypes.FIRE_COCKED))

■■// setup state machine
■■// basic weapon states
■■WeaponStateBase E = new RSSEmpty(this, NULL, RSSAnimState.DEFAULT);
■■WeaponStateBase F = new RSSFireout(this, NULL, RSSAnimState.DEFAULT);
■■WeaponStateBase J = new RSSJammed(this, NULL, RSSAnimState.DEFAULT);
■■WeaponStateBase L = new RSSLoaded(this, NULL, RSSAnimState.DEFAULT);
■■// unstable (intermediate) states
■■WeaponStateBase Mech_E = new WeaponCharging(this, NULL, WeaponActions.MECHANISM, Weap
■■WeaponStateBase Mech_F = new WeaponCharging(this, NULL, WeaponActions.MECHANISM, Weap
■■WeaponStateBase Mech_L = new WeaponEjectBullet(this, NULL, WeaponActions.MECHANISM, Wea
■■
■■WeaponChambering Chamber_E = new WeaponChambering(this, NULL, WeaponActions.CHAMBERI
■■WeaponChambering Chamber_F = new WeaponChambering(this, NULL, WeaponActions.CHAMBERI
■■//WeaponStateBase Chamber_L = new RifleReChambering(this, NULL, WeaponActions.CHAMBERIN
■■
■■WeaponStateBase Trigger_E = new WeaponDryFire(this, NULL, WeaponActions.FIRE, WeaponAction
■■WeaponStateBase Trigger_L = new WeaponFire(this, NULL, WeaponActions.FIRE, WeaponActionFire
■■WeaponStateBase Trigger_F = new WeaponDryFire(this, NULL, WeaponActions.FIRE, WeaponAction

■■WeaponStateBase Trigger_LJ = new WeaponFireToJam(this, NULL, WeaponActions.FIRE, WeaponAc
■■
■■WeaponStateBase Unjam_J = new WeaponUnjamming(this, NULL, WeaponActions.UNJAMMING, We

■■// events
■■WeaponEventBase __M__ = new WeaponEventMechanism;
■■WeaponEventBase __T__ = new WeaponEventTrigger;
■■WeaponEventBase __TJ_ = new WeaponEventTriggerToJam;
■■WeaponEventBase __L__ = new WeaponEventLoad1Bullet;
■■WeaponEventBase __U__ = new WeaponEventUnjam;
■■WeaponEventBase _fin_ = new WeaponEventHumanCommandActionFinished;
■■WeaponEventBase _abt_ = new WeaponEventHumanCommandActionAborted;
■■WeaponEventBase _dto_ = new WeaponEventDryFireTimeout;

■■m_fsm = new WeaponFSM();
■■// charging
■■m_fsm.AddTransition(new WeaponTransition( E,■■■__M__■Mech_E));
```

```
// ---------------------------------------------
// ------ RifleSingleShot_Base.c - START --------------------
// ---------------------------------------------


enum RSSAnimState
{
■DEFAULT ■■■= 0, ■///< default weapon state, closed and discharged
};

enum RSSStableStateID
{
■UNKNOWN■■■■=  0,
■Empty■■■■=  1,
■Fireout■■■■=  2,
■Loaded■■■■=  3,
■Jammed■■■■=  4,
}

class RSSEmpty extends WeaponStableState
{
■override void OnEntry (WeaponEventBase e) { if (LogManager.IsWeaponLogEnable()) { wpnPrint("[wpnf
■override void OnExit (WeaponEventBase e) { super.OnExit(e); if (LogManager.IsWeaponLogEnable()) { w
■override int GetCurrentStateID () { return RSSStableStateID.Empty; }
■override bool HasBullet () { return false; }
■override bool HasMagazine () { return false; }
■override bool IsJammed () { return false; }
■override bool IsRepairEnabled () { return true; }
■override void InitMuzzleArray () { m_muzzleHasBullet = {MuzzleState.E}; }
};
class RSSFireout extends WeaponStableState
{
■override void OnEntry (WeaponEventBase e) { if (LogManager.IsWeaponLogEnable()) { wpnPrint("[wpnf
■override void OnExit (WeaponEventBase e) { super.OnExit(e); if (LogManager.IsWeaponLogEnable()) { w
■override int GetCurrentStateID () { return RSSStableStateID.Fireout; }
■override bool HasBullet () { return true; }
■override bool HasMagazine () { return false; }
■override bool IsJammed () { return false; }
■override bool IsRepairEnabled () { return true; }
■override void InitMuzzleArray () { m_muzzleHasBullet = {MuzzleState.F}; }
};
class RSSLoaded extends WeaponStableState
{
■override void OnEntry (WeaponEventBase e) { if (LogManager.IsWeaponLogEnable()) { wpnPrint("[wpnf
■override void OnExit (WeaponEventBase e) { super.OnExit(e); if (LogManager.IsWeaponLogEnable()) { w
■override int GetCurrentStateID () { return RSSStableStateID.Loaded; }
■override bool HasBullet () { return true; }
■override bool HasMagazine () { return false; }
■override bool IsJammed () { return false; }
■override bool IsRepairEnabled () { return true; }
■override void InitMuzzleArray () { m_muzzleHasBullet = {MuzzleState.L}; }
};
class RSSJammed extends WeaponStateJammed
{
■override void OnEntry (WeaponEventBase e) { if (LogManager.IsWeaponLogEnable()) { wpnPrint("[wpnf
■override void OnExit (WeaponEventBase e) { super.OnExit(e); if (LogManager.IsWeaponLogEnable()) { w
■override int GetCurrentStateID () { return RSSStableStateID.Jammed; }
■override bool HasBullet () { return true; }
■override bool HasMagazine () { return false; }
■override bool IsJammed () { return true; }
■override bool IsBoltOpen () { return true; }
■override bool IsRepairEnabled () { return true; }
■override void InitMuzzleArray () { m_muzzleHasBullet = {MuzzleState.F}; }
```

```
// ---------------------------------------------
// ------ Rifle_Base.c - START --------------------
// ---------------------------------------------


enum DefaultAnimState
{
■DEFAULT ■■■= 0, ■///< default weapon state, closed and discharged
};

/**@class■■Rifle_Base
 * @brief■■base for rifles
 * @NOTE■■name copies config base class
 **/
class Rifle_Base extends Weapon_Base
{
■void Rifle_Base ()
■{
■}

■override void InitStateMachine ()
■{
■■m_abilities.Insert(new AbilityRecord(WeaponActions.MECHANISM, WeaponActionMechanismTypes.M

■■// setup state machine
■■// basic weapon states
■■WeaponStateBase E = new WeaponStableState(this, NULL, DefaultAnimState.DEFAULT);

■■WeaponStateBase Mech = new WeaponCharging(this, NULL, WeaponActions.MECHANISM, Weapon

■■// events
■■WeaponEventBase __M__ = new WeaponEventMechanism;
■■WeaponEventBase _fin_ = new WeaponEventHumanCommandActionFinished;
■■WeaponEventBase _abt_ = new WeaponEventHumanCommandActionAborted;

■■m_fsm = new WeaponFSM();
■■// charging
■■m_fsm.AddTransition(new WeaponTransition(E■■, __M__,■Mech));
■■m_fsm.AddTransition(new WeaponTransition(Mech■, _fin_,■E));
■■m_fsm.AddTransition(new WeaponTransition(Mech■, _abt_,■E));
■■
■■m_fsm.SetInitialState(E);

■■SelectionBulletHide();

■■m_fsm.Start();
■}

};





// ---------------------------------------------
// ------ Rifle_Base.c - END ----------------------
// ---------------------------------------------
// ---------------------------------------------
// ------ RightArea.c - START --------------------
// ---------------------------------------------


class RightArea: Container
{
■ref PlayerContainer■m_PlayerContainer;
■protected Widget■m_ContentParent;
■
■protected ScrollWidget■■■■■■m_ScrollWidget;
■protected Widget■■■■■■■m_UpIcon;
■protected Widget■■■■■■■m_DownIcon;
■
■protected ref SizeToChild■■■■■m_ContentResize;
■protected bool■■■■■■■■m_ShouldChangeSize = true;
■protected bool■■■■■■■■m_ProcessGridMovement;
■
■void RightArea(LayoutHolder parent)
■{
■■m_MainWidget.Show(true);
■■m_ScrollWidget■= ScrollWidget.Cast( m_MainWidget.FindAnyWidget( "Scroller" ) );
■■m_MainWidget■■= m_MainWidget.FindAnyWidget("Content");
■■//m_ContentParent■■= m_RootWidget.FindAnyWidget("ContentParent");
■■m_PlayerContainer■= new PlayerContainer(this, false);
■■m_PlayerContainer.SetPlayer(PlayerBase.Cast(GetGame().GetPlayer()));
■■m_Body.Insert(m_PlayerContainer);
■■m_ActiveIndex = 0;
■■m_ProcessGridMovement = false;
■■
■■m_UpIcon■■= m_RootWidget.FindAnyWidget( "Up" );
■■m_DownIcon■■= m_RootWidget.FindAnyWidget( "Down" );
■■
■■//m_ContentParent.GetScript( m_ContentResize );
■■
■■RecomputeOpenedContainers();
■}
■
■PlayerContainer GetPlayerContainer()
■{
■■return m_PlayerContainer;
■}
■
■override void DraggingOverHeader(Widget w, int x, int y, Widget receiver)
■{
■■m_PlayerContainer.DraggingOverHeader(w, x, y, receiver);
■}
■
■override bool Select()
■{
■■return m_PlayerContainer.Select();
■}
■
■override bool SelectItem()
■{
■■return m_PlayerContainer.SelectItem();
■}
■
■override bool Combine()
■{
■■return m_PlayerContainer.Combine();
■}
■
```

```
// ---------------------------------------------
// ------ RightGap.c - START --------------------
// ---------------------------------------------


// ---------------------------------------------------------
class RightGap : ScriptedWidgetEventHandler
{
■reference int gap;
■bool kola;
■/*void OnWidgetScriptInit(Widget w)
■{
■■float width;
■■float height;
■■w.SetFlags( WidgetFlags.EXACTSIZE, false );
■■w.GetScreenSize( width, height );
■■Print(w.GetName());
■■Print(width);
■■Print(height);
■■w.SetSize( width-gap, height );
■■w.Update();
■}*/
■override bool OnUpdate( Widget  w)
■{
■■float width;
■■float height;
■■w.SetFlags( WidgetFlags.EXACTSIZE, false );
■■w.GetParent().GetScreenSize( width, height );
■■w.SetSize( width-gap, height );
■■return false;
■}
};


// ---------------------------------------------
// ------ RightGap.c - END ----------------------
// ---------------------------------------------


// ---------------------------------------------
// ------ Roadflare.c - START --------------------
// ---------------------------------------------


/*
■Author: Boris Vacula
■Description: The flare has 3 burning states during which it uses different particle effects while illumnating
■When the flare is dropped while its burning, it is stood up on its stands. This makes the shadows, illumina
*/

enum RoadflareBurningState
{
■NOT_BURNING,
■INITIAL_BURN,
■MAIN_BURN,
■FINAL_BURN,
■SMOKE_ONLY
};

enum RoadflareModelStates
{
■DEFAULT,
■UNCAPPED_UNIGNITED
```

```
// ---------------------------------------------
// ------ RoadflareLight.c - START --------------------
// ---------------------------------------------


class RoadflareLight extends PointLightBase
{
■static float ■m_RoadflareRadius = 30;
■static float ■m_RoadflareBrightness = 8.5;
■static string ■m_MemoryPoint = "light";
■
■void RoadflareLight()
■{
■■SetVisibleDuringDaylight(true);
■■SetRadiusTo( m_RoadflareRadius );
■■SetBrightnessTo(m_RoadflareBrightness);
■■SetCastShadow(true);
■■FadeIn(1);
■■SetFadeOutTime(1);
■■SetDiffuseColor(1.0, 0.3, 0.3);
■■SetAmbientColor(1.0, 0.3, 0.3);
■■SetFlareVisible(true);
■■SetFlickerAmplitude(0.2);
■■SetFlickerSpeed(1.5);
■■SetDancingShadowsMovementSpeed(0.1);
■■SetDancingShadowsAmplitude(0.015);
■■EnableHeatHaze(true);
■■SetHeatHazeRadius(0.1);
■■SetHeatHazePower(0.02);
■}
■
■/*override void OnFrameLightSource(IEntity other, float timeSlice)
■{
■■
■}*/
}


// ---------------------------------------------
// ------ RoadflareLight.c - END ----------------------
// ---------------------------------------------


// ---------------------------------------------
// ------ RockBase.c - START --------------------
// ---------------------------------------------


class RockBase: Object
{
■override bool IsRock()
■{
■■return true;
■}

■int GetAmountOfDrops(ItemBase item)
■{
■■if (item)
■■{
■■■switch (item.GetType())
■■■{
■■■case "Pickaxe":
■■■■return 4;
```

```
// ---------------------------------------------
// ------ RopeBelt.c - START --------------------
// ---------------------------------------------


class RopeBelt: Clothing
{■■
■override void SetActions()
■{
■■super.SetActions();
■■
■■AddAction(ActionDeCraftRopeBelt);
■}
■
■override bool CanPutInCargo( EntityAI parent )
■{
■■if( !super.CanPutInCargo( parent ) )
■■{
■■■return false;
■■}
■■
■■return IsEmpty();
■}
■
■override bool CanReceiveAttachment( EntityAI attachment,int slotId )
■{
■■if( !super.CanReceiveAttachment( attachment, slotId ) )
■■{
■■■return false;
■■}
■■
■■return !GetInventory().IsInCargo();
■}
};



// ---------------------------------------------
// ------ RopeBelt.c - END ----------------------
// ---------------------------------------------


// ---------------------------------------------
// ------ Rotator.c - START --------------------
// ---------------------------------------------


// -----------------------------------------------------------
class Rotator
{
■reference float speed;

■protected Widget m_root;■
■protected ref AnimatorTimer m_anim;

■// -----------------------------------------------------------
■void Rotator()
■{
■■m_anim = new AnimatorTimer();
■}
■
■// -----------------------------------------------------------
■protected void Update()
```

```
// ---------------------------------------------
// ------ Ruger1022.c - START --------------------
// ---------------------------------------------


class Ruger1022_Base : RifleBoltFree_Base
{
■override RecoilBase SpawnRecoilObject()
■{
■■return new Ruger1022Recoil(this); //TODO
■}
■
■/*override int GetWeaponSpecificCommand(int weaponAction ,int subCommand)
■{
■■if ( weaponAction == WeaponActions.RELOAD)
■■{
■■■switch (subCommand)
■■■{
■■■■case WeaponActionReloadTypes.RELOADSRIFLE_MAGAZINE_BULLET:
■■■■■return WeaponActionReloadTypes.RELOADRIFLE_MAGAZINE_BULLET;
■■■■
■■■■case WeaponActionReloadTypes.RELOADSRIFLE_NOMAGAZINE_BULLET:
■■■■■return WeaponActionReloadTypes.RELOADRIFLE_NOMAGAZINE_BULLET;
■■■■
■■■■case WeaponActionReloadTypes.RELOADSRIFLE_MAGAZINE_NOBULLET:
■■■■■return WeaponActionReloadTypes.RELOADRIFLE_MAGAZINE_NOBULLET;
■■■■
■■■■case WeaponActionReloadTypes.RELOADSRIFLE_NOMAGAZINE_NOBULLET:
■■■■■return WeaponActionReloadTypes.RELOADRIFLE_NOMAGAZINE_NOBULLET;
■■■■
■■■■default:
■■■■■return subCommand;
■■■}
■■
■■}
■■return subCommand;
■}*/
■
■//Debug menu Spawn Ground Special
■override void OnDebugSpawn()
■{■■
■■GameInventory inventory = GetInventory();

■■inventory.CreateInInventory( "HuntingOptic" );
■■
■■SpawnAttachedMagazine("Mag_Ruger1022_30Rnd");
■}■
};


// ---------------------------------------------
// ------ Ruger1022.c - END ----------------------
// ---------------------------------------------


// ---------------------------------------------
// ------ Ruger1022Recoil.c - START --------------------
// ---------------------------------------------


class Ruger1022Recoil: RecoilBase
{
■override void Init()
```

```
// ---------------------------------------------
// ------ Saiga.c - START --------------------
// ---------------------------------------------


class Saiga_Base : RifleBoltFree_Base
{
■override RecoilBase SpawnRecoilObject()
■{
■■return new SiagaRecoil(this);
■}
■
■/*override int GetWeaponSpecificCommand(int weaponAction ,int subCommand)
■{
■■if ( weaponAction == WeaponActions.RELOAD)
■■{
■■■switch (subCommand)
■■■{
■■■■case WeaponActionReloadTypes.RELOADSRIFLE_MAGAZINE_BULLET:
■■■■■return WeaponActionReloadTypes.RELOADRIFLE_MAGAZINE_BULLET;
■■■■
■■■■case WeaponActionReloadTypes.RELOADSRIFLE_NOMAGAZINE_BULLET:
■■■■■return WeaponActionReloadTypes.RELOADRIFLE_NOMAGAZINE_BULLET;
■■■■
■■■■case WeaponActionReloadTypes.RELOADSRIFLE_MAGAZINE_NOBULLET:
■■■■■return WeaponActionReloadTypes.RELOADRIFLE_MAGAZINE_NOBULLET;
■■■■
■■■■case WeaponActionReloadTypes.RELOADSRIFLE_NOMAGAZINE_NOBULLET:
■■■■■return WeaponActionReloadTypes.RELOADRIFLE_NOMAGAZINE_NOBULLET;
■■■■
■■■■default:
■■■■■return subCommand;
■■■}
■■
■■}
■■return subCommand;
■}*/
■
■override bool CanEnterIronsights()
■{
■■ItemOptics optic = GetAttachedOptics();
■■if ( optic && PSO1Optic.Cast(optic) || PSO11Optic.Cast(optic) || KashtanOptic.Cast(optic) || KazuarOpt
■■■return true;
■■return super.CanEnterIronsights();
■}
■■
■//Debug menu Spawn Ground Special
■override void OnDebugSpawn()
■{
■■super.OnDebugSpawn();
■■GameInventory inventory = GetInventory();
■■inventory.CreateInInventory( "Saiga_Bttstck" );
■■inventory.CreateInInventory( "KobraOptic" );
■■inventory.CreateInInventory( "Battery9V" );
■}■
}
class Saiga : Saiga_Base
{

};


// ---------------------------------------------
// ---------------------------------------------
// ------ Saline.c - START --------------------
// ---------------------------------------------


class SalineMdfr: ModifierBase
{
■float m_RegenTime;
■override void Init()
■{
■■m_TrackActivatedTime = true;
■■m_IsPersistent = true;
■■m_ID ■■■■■= eModifiers.MDF_SALINE;
■■m_TickIntervalInactive ■= DEFAULT_TICK_TIME_INACTIVE;
■■m_TickIntervalActive ■= DEFAULT_TICK_TIME_ACTIVE;
■■m_RegenTime = CalculateRegenTime();
■■DisableActivateCheck();
■}

■override bool ActivateCondition(PlayerBase player)
■{
■■return false;
■}

■override bool DeactivateCondition(PlayerBase player)
■{
■■float attached_time = GetAttachedTime();

■■if ( attached_time > m_RegenTime )
■■{
■■■return true;
■■}
■■else
■■{
■■■return false;
■■}
■}

■override void OnReconnect(PlayerBase player)
■{
■■OnActivate(player);
■}
■
■override void OnActivate(PlayerBase player)
■{
■■player.IncreaseHealingsCount();
■■/*
■■if( player.GetNotifiersManager() )
■■■player.GetNotifiersManager().ActivateByType(eNotifiers.NTF_PILLS);
■■*/
■}
■
■override void OnDeactivate(PlayerBase player)
■{
■■player.DecreaseHealingsCount();
■■/*
■■if( player.GetNotifiersManager() )
■■■player.GetNotifiersManager().DeactivateByType(eNotifiers.NTF_PILLS);
■■*/
■}

■override void OnTick(PlayerBase player, float deltaT)
■{
```

```
// ---------------------------------------------
// ------ SalineBag.c - START --------------------
// ---------------------------------------------


class SalineBag: Inventory_Base {};




// ---------------------------------------------
// ------ SalineBag.c - END ----------------------
// ---------------------------------------------


// ---------------------------------------------
// ------ SalineBagIV.c - START --------------------
// ---------------------------------------------


class SalineBagIV: Inventory_Base
{
■override void SetActions()
■{
■■super.SetActions();
■■
■■AddAction(ActionGiveSalineTarget);
■■AddAction(ActionGiveSalineSelf);
■}
};




// ---------------------------------------------
// ------ SalineBagIV.c - END ----------------------
// ---------------------------------------------


// ---------------------------------------------
// ------ Salmonella.c - START --------------------
// ---------------------------------------------


class SalmonellaMdfr: ModifierBase
{
■static const int AGENT_THRESHOLD_ACTIVATE = 60;
■static const int AGENT_THRESHOLD_DEACTIVATE = 20;

■static const int CHANCE_OF_VOMIT = 10;■■■// base vomit chance
■static const int CHANCE_OF_VOMIT_AGENT = 20;■// adjusted by the agent count
■static const int WATER_DRAIN_FROM_VOMIT = 450;
■static const int ENERGY_DRAIN_FROM_VOMIT = 310;
■static const float STOMACH_MIN_VOLUME = 200;■// min volume of stomach for vomit symptom
■
■static const float EVENT_INTERVAL_MIN = 12;
■static const float EVENT_INTERVAL_MAX = 18;
■
■
■
■float m_Time;
■float m_NextEvent;
■
■override void Init()
■{
```

```
// ---------------------------------------------
// ------ SalmonellaAgent.c - START --------------------
// ---------------------------------------------


class SalmonellaAgent extends AgentBase
{
■override void Init()
■{
■■m_Type ■■■■■= eAgents.SALMONELLA;
■■m_Invasibility ■■■= 0.75;
■■m_TransferabilityIn■■= 0.1;
■■m_TransferabilityOut■= 0.1;
■■m_AntibioticsResistance = 1;
■■m_MaxCount ■■■■= 300;
■■m_Potency ■■■■= EStatLevels.HIGH;
■■m_DieOffSpeed ■■■= 1;
■}
}


// ---------------------------------------------
// ------ SalmonellaAgent.c - END ----------------------
// ---------------------------------------------


// ---------------------------------------------
// ------ SambucusBerry.c - START --------------------
// ---------------------------------------------


class SambucusBerry extends Edible_Base
{
■override bool CanBeCooked()
■{
■■return true;
■}■■
■
■override bool CanBeCookedOnStick()
■{
■■return false;
■}
■
■override bool IsFruit()
■{
■■return true;
■}

■override bool CanDecay()
■{
■■return true;
■}
■
■bool ConditionAttach ( EntityAI parent )
■{
■■//if Barrel_ColorBase
■■if ( parent.IsInherited( Barrel_ColorBase ) )
■■{
■■■Barrel_ColorBase barrel = Barrel_ColorBase.Cast( parent );
■■■
■■■if ( barrel.IsOpen() && !barrel.FindAttachmentBySlotName( "Nails" ) && !barrel.FindAttachmentBySlot
■■■{
■■■■return true;
```

```
// ---------------------------------------------
// ------ SantasBeard.c - START --------------------
// ---------------------------------------------


class SantasBeard extends Clothing
{
■override bool CanPutAsAttachment( EntityAI parent )
■{
■■if (!super.CanPutAsAttachment(parent)) {return false;}
■■
■■Clothing headgear = Clothing.Cast(parent.FindAttachmentBySlotName("Headgear"));
■■if ( headgear && (headgear.ConfigGetBool("noMask") && !PumpkinHelmet.Cast(headgear) && !Weldin
■■{
■■■return false;
■■}
■■
■■return true;
■}
■
■override void SetActions()
■{
■■super.SetActions();
■■AddAction(ActionWringClothes);
■}
}


// ---------------------------------------------
// ------ SantasBeard.c - END ----------------------
// ---------------------------------------------


// ---------------------------------------------
// ------ SantasHat.c - START --------------------
// ---------------------------------------------


class SantasHat extends Clothing
{
■override void SetActions()
■{
■■super.SetActions();
■■AddAction(ActionWringClothes);
■}
};



// ---------------------------------------------
// ------ SantasHat.c - END ----------------------
// ---------------------------------------------


// ---------------------------------------------
// ------ Sardines.c - START --------------------
// ---------------------------------------------


class Sardines extends Edible_Base
{
■override bool CanBeCookedOnStick()
■{
```

```
// ---------------------------------------------
// ------ SardinesCan.c - START --------------------
// ---------------------------------------------


class SardinesCan : Edible_Base
{
■override void Open()
■{
■■//super.Open();
■■ReplaceEdibleWithNew("SardinesCan_Opened");
■}
■
■override bool IsOpen()
■{
■■return false;
■}
}



// ---------------------------------------------
// ------ SardinesCan.c - END ----------------------
// ---------------------------------------------


// ---------------------------------------------
// ------ SardinesCan_Opened.c - START --------------------
// ---------------------------------------------


class SardinesCan_Opened: Edible_Base
{
■override bool CanDecay()
■{
■■return true;
■}
■
■override bool CanProcessDecay()
■{
■■return !(GetAgents() & eAgents.FOOD_POISON);
■}
■
■override void SetActions()
■{
■■super.SetActions();
■■
■■AddAction(ActionForceFeedCan);
■■AddAction(ActionEatCan);
■}
}



// ---------------------------------------------
// ------ SardinesCan_Opened.c - END ----------------------
// ---------------------------------------------
// ---------------------------------------------
// ------ SawedoffIzh18.c - START --------------------
// ---------------------------------------------


class SawedoffIzh18 extends Izh18_Base
{
■void SawedoffIzh18 ()
■{
■}
■
■override RecoilBase SpawnRecoilObject()
■{
■■return new Izh18SawedOffRecoil(this);
■}
};



// ---------------------------------------------
// ------ SawedoffIzh18.c - END ----------------------
// ---------------------------------------------


// ---------------------------------------------
// ------ SawedoffIzh18Shotgun.c - START --------------------
// ---------------------------------------------


/**@class■■SawedoffIzh18Shotgun
 * @brief■■base for SawedoffIzh18Shotgun
 * @NOTE■■name copies config base class
 **/
class SawedoffIzh18Shotgun extends Izh18Shotgun_Base
{
■void SawedoffIzh18Shotgun ()
■{
■}
■
■override RecoilBase SpawnRecoilObject()
■{
■■return new Izh18ShotgunRecoil(this);
■}
};



// ---------------------------------------------
// ------ SawedoffIzh18Shotgun.c - END ----------------------
// ---------------------------------------------


// ---------------------------------------------
// ------ SawoffB95.c - START --------------------
// ---------------------------------------------


class SawOFFB95 extends RecipeBase
{■
■override void Init()
■{
■■m_Name = "#STR_sawoffbarrel0";
■■m_IsInstaRecipe = false;//should this recipe be performed instantly without animation
```

```
// ---------------------------------------------
// ------ SawoffFAMAS.c - START --------------------
// ---------------------------------------------


class SawoffFAMAS extends RecipeBase
{■
■override void Init()
■{
■■m_Name = "#STR_sawoffhandle0";
■■m_IsInstaRecipe = false;//should this recipe be performed instantly without animation
■■m_AnimationLength = 2;//animation length in relative time units
■■m_Specialty = 0.02;// value > 0 for roughness, value < 0 for precision
■■
■■
■■//conditions
■■m_MinDamageIngredient[0] = -1;//-1 = disable check
■■m_MaxDamageIngredient[0] = 3;//-1 = disable check
■■
■■m_MinQuantityIngredient[0] = -1;//-1 = disable check
■■m_MaxQuantityIngredient[0] = -1;//-1 = disable check
■■
■■m_MinDamageIngredient[1] = -1;//-1 = disable check
■■m_MaxDamageIngredient[1] = 3;//-1 = disable check
■■
■■m_MinQuantityIngredient[1] = -1;//-1 = disable check
■■m_MaxQuantityIngredient[1] = -1;//-1 = disable check
■■//----------------------------------------------------------------------------------------------------------------------
■■
■■//INGREDIENTS
■■//ingredient 1
■■InsertIngredient(0,"FAMAS");//you can insert multiple ingredients this way
■■
■■m_IngredientAddHealth[0] = 0;// 0 = do nothing
■■m_IngredientSetHealth[0] = -1; // -1 = do nothing
■■m_IngredientAddQuantity[0] = 0;// 0 = do nothing
■■m_IngredientDestroy[0] = false;//true = destroy, false = do nothing
■■m_IngredientUseSoftSkills[0] = false;// set 'true' to allow modification of the values by softskills on this i
■■
■■//ingredient 2
■■InsertIngredient(1,"Hacksaw");//you can insert multiple ingredients this way
■
■■m_IngredientAddHealth[1] = -20;// 0 = do nothing
■■m_IngredientSetHealth[1] = -1; // -1 = do nothing
■■m_IngredientAddQuantity[1] = 0;// 0 = do nothing
■■m_IngredientDestroy[1] = false;// false = do nothing
■■m_IngredientUseSoftSkills[1] = true;// set 'true' to allow modification of the values by softskills on this in
■■
■■//----------------------------------------------------------------------------------------------------------------------
■■
■■//result1
■■/*
■■AddResult("SawedoffMosin9130");//add results here

■■m_ResultSetFullQuantity[0] = false;//true = set full quantity, false = do nothing
■■m_ResultSetQuantity[0] = -1;//-1 = do nothing
■■m_ResultSetHealth[0] = -1;//-1 = do nothing
■■m_ResultInheritsHealth[0] = 0;// (value) == -1 means do nothing; a (value) >= 0 means this result will in
■■m_ResultInheritsColor[0] = -1;// (value) == -1 means do nothing; a (value) >= 0 means this result classn
■■m_ResultToInventory[0] = -2;//(value) == -2 spawn result on the ground;(value) == -1 place anywhere in
■■m_ResultUseSoftSkills[0] = false;// set 'true' to allow modification of the values by softskills on this resu
■■m_ResultReplacesIngredient[0] = 0;// value == -1 means do nothing; a value >= 0 means this result wil
■■*/
```

```
// ----------------------------------------------
// ------ SawOffIzh18.c - START --------------------
// ----------------------------------------------


class SawOffIzh18 extends RecipeBase
{■
■override void Init()
■{
■■m_Name = "#STR_sawoffbarrel0";
■■m_IsInstaRecipe = false;//should this recipe be performed instantly without animation
■■m_AnimationLength = 2;//animation length in relative time units
■■m_Specialty = 0.02;// value > 0 for roughness, value < 0 for precision
■■
■■
■■//conditions
■■m_MinDamageIngredient[0] = -1;//-1 = disable check
■■m_MaxDamageIngredient[0] = 3;//-1 = disable check
■■
■■m_MinQuantityIngredient[0] = -1;//-1 = disable check
■■m_MaxQuantityIngredient[0] = -1;//-1 = disable check
■■
■■m_MinDamageIngredient[1] = -1;//-1 = disable check
■■m_MaxDamageIngredient[1] = 3;//-1 = disable check
■■
■■m_MinQuantityIngredient[1] = -1;//-1 = disable check
■■m_MaxQuantityIngredient[1] = -1;//-1 = disable check
■■//----------------------------------------------------------------------------------------------------------------------
■■
■■//INGREDIENTS
■■//ingredient 1
■■InsertIngredient(0,"Izh18");//you can insert multiple ingredients this way
■■
■■m_IngredientAddHealth[0] = 0;// 0 = do nothing
■■m_IngredientSetHealth[0] = -1; // -1 = do nothing
■■m_IngredientAddQuantity[0] = 0;// 0 = do nothing
■■m_IngredientDestroy[0] = false;//true = destroy, false = do nothing
■■m_IngredientUseSoftSkills[0] = false;// set 'true' to allow modification of the values by softskills on this i
■■
■■//ingredient 2
■■InsertIngredient(1,"Hacksaw");//you can insert multiple ingredients this way
■
■■m_IngredientAddHealth[1] = -20;// 0 = do nothing
■■m_IngredientSetHealth[1] = -1; // -1 = do nothing
■■m_IngredientAddQuantity[1] = 0;// 0 = do nothing
■■m_IngredientDestroy[1] = false;// false = do nothing
■■m_IngredientUseSoftSkills[1] = true;// set 'true' to allow modification of the values by softskills on this in
■■
■■//----------------------------------------------------------------------------------------------------------------------
■■
■■//result1
■■/*
■■AddResult("SawedoffIzh18");//add results here

■■m_ResultSetFullQuantity[0] = false;//true = set full quantity, false = do nothing
■■m_ResultSetQuantity[0] = -1;//-1 = do nothing
■■m_ResultSetHealth[0] = -1;//-1 = do nothing
■■m_ResultInheritsHealth[0] = 0;// (value) == -1 means do nothing; a (value) >= 0 means this result will in
■■m_ResultInheritsColor[0] = -1;// (value) == -1 means do nothing; a (value) >= 0 means this result class
■■m_ResultToInventory[0] = -2;//(value) == -2 spawn result on the ground;(value) == -1 place anywhere i
■■m_ResultUseSoftSkills[0] = false;// set 'true' to allow modification of the values by softskills on this resu
■■m_ResultReplacesIngredient[0] = 0;// value == -1 means do nothing; a value >= 0 means this result wil
■■*/
```

```
// ---------------------------------------------
// ------ SawOffIzh18Shotgun.c - START --------------------
// ---------------------------------------------


class SawOffIzh18Shotgun extends RecipeBase
{■
■override void Init()
■{
■■m_Name = "#STR_sawoffbarrel0";
■■m_IsInstaRecipe = false;//should this recipe be performed instantly without animation
■■m_AnimationLength = 2;//animation length in relative time units
■■m_Specialty = 0.02;// value > 0 for roughness, value < 0 for precision
■■
■■
■■//conditions
■■m_MinDamageIngredient[0] = -1;//-1 = disable check
■■m_MaxDamageIngredient[0] = 3;//-1 = disable check
■■
■■m_MinQuantityIngredient[0] = -1;//-1 = disable check
■■m_MaxQuantityIngredient[0] = -1;//-1 = disable check
■■
■■m_MinDamageIngredient[1] = -1;//-1 = disable check
■■m_MaxDamageIngredient[1] = 3;//-1 = disable check
■■
■■m_MinQuantityIngredient[1] = -1;//-1 = disable check
■■m_MaxQuantityIngredient[1] = -1;//-1 = disable check
■■//----------------------------------------------------------------------------------------------------------------------
■■
■■//INGREDIENTS
■■//ingredient 1
■■InsertIngredient(0,"Izh18Shotgun");//you can insert multiple ingredients this way
■■
■■m_IngredientAddHealth[0] = 0;// 0 = do nothing
■■m_IngredientSetHealth[0] = -1; // -1 = do nothing
■■m_IngredientAddQuantity[0] = 0;// 0 = do nothing
■■m_IngredientDestroy[0] = false;//true = destroy, false = do nothing
■■m_IngredientUseSoftSkills[0] = false;// set 'true' to allow modification of the values by softskills on this i
■■
■■//ingredient 2
■■InsertIngredient(1,"Hacksaw");//you can insert multiple ingredients this way
■
■■m_IngredientAddHealth[1] = -20;// 0 = do nothing
■■m_IngredientSetHealth[1] = -1; // -1 = do nothing
■■m_IngredientAddQuantity[1] = 0;// 0 = do nothing
■■m_IngredientDestroy[1] = false;// false = do nothing
■■m_IngredientUseSoftSkills[1] = true;// set 'true' to allow modification of the values by softskills on this in
■■
■■//----------------------------------------------------------------------------------------------------------------------
■■
■■//result1
■■/*
■■AddResult("SawedoffIzh18");//add results here

■■m_ResultSetFullQuantity[0] = false;//true = set full quantity, false = do nothing
■■m_ResultSetQuantity[0] = -1;//-1 = do nothing
■■m_ResultSetHealth[0] = -1;//-1 = do nothing
■■m_ResultInheritsHealth[0] = 0;// (value) == -1 means do nothing; a (value) >= 0 means this result will in
■■m_ResultInheritsColor[0] = -1;// (value) == -1 means do nothing; a (value) >= 0 means this result classn
■■m_ResultToInventory[0] = -2;//(value) == -2 spawn result on the ground;(value) == -1 place anywhere in
■■m_ResultUseSoftSkills[0] = false;// set 'true' to allow modification of the values by softskills on this resu
■■m_ResultReplacesIngredient[0] = 0;// value == -1 means do nothing; a value >= 0 means this result wil
■■*/
```

```
// ---------------------------------------------
// ------ SawoffMagnum.c - START --------------------
// ---------------------------------------------


class SawOffMagnum extends RecipeBase
{■
■override void Init()
■{
■■m_Name = "#STR_sawoffbarrel0";
■■m_IsInstaRecipe = false;//should this recipe be performed instantly without animation
■■m_AnimationLength = 2;//animation length in relative time units
■■m_Specialty = 0.02;// value > 0 for roughness, value < 0 for precision
■■
■■
■■//conditions
■■m_MinDamageIngredient[0] = -1;//-1 = disable check
■■m_MaxDamageIngredient[0] = 3;//-1 = disable check
■■
■■m_MinQuantityIngredient[0] = -1;//-1 = disable check
■■m_MaxQuantityIngredient[0] = -1;//-1 = disable check
■■
■■m_MinDamageIngredient[1] = -1;//-1 = disable check
■■m_MaxDamageIngredient[1] = 3;//-1 = disable check
■■
■■m_MinQuantityIngredient[1] = -1;//-1 = disable check
■■m_MaxQuantityIngredient[1] = -1;//-1 = disable check
■■//----------------------------------------------------------------------------------------------------------------------
■■
■■//INGREDIENTS
■■//ingredient 1
■■InsertIngredient(0,"Magnum");//you can insert multiple ingredients this way
■■
■■m_IngredientAddHealth[0] = 0;// 0 = do nothing
■■m_IngredientSetHealth[0] = -1; // -1 = do nothing
■■m_IngredientAddQuantity[0] = 0;// 0 = do nothing
■■m_IngredientDestroy[0] = false;//true = destroy, false = do nothing
■■m_IngredientUseSoftSkills[0] = false;// set 'true' to allow modification of the values by softskills on this i
■■
■■//ingredient 2
■■InsertIngredient(1,"Hacksaw");//you can insert multiple ingredients this way
■
■■m_IngredientAddHealth[1] = -20;// 0 = do nothing
■■m_IngredientSetHealth[1] = -1; // -1 = do nothing
■■m_IngredientAddQuantity[1] = 0;// 0 = do nothing
■■m_IngredientDestroy[1] = false;// false = do nothing
■■m_IngredientUseSoftSkills[1] = true;// set 'true' to allow modification of the values by softskills on this in
■■
■■//----------------------------------------------------------------------------------------------------------------------
■■
■■/*
■■//result1
■■
■■AddResult("SawedoffMagnum");//add results here

■■m_ResultSetFullQuantity[0] = false;//true = set full quantity, false = do nothing
■■m_ResultSetQuantity[0] = -1;//-1 = do nothing
■■m_ResultSetHealth[0] = -1;//-1 = do nothing
■■m_ResultInheritsHealth[0] = 0;// (value) == -1 means do nothing; a (value) >= 0 means this result will in
■■m_ResultInheritsColor[0] = -1;// (value) == -1 means do nothing; a (value) >= 0 means this result classn
■■m_ResultToInventory[0] = -2;//(value) == -2 spawn result on the ground;(value) == -1 place anywhere in
■■m_ResultUseSoftSkills[0] = false;// set 'true' to allow modification of the values by softskills on this resu
■■m_ResultReplacesIngredient[0] = 0;// value == -1 means do nothing; a value >= 0 means this result wil
```

```
// ---------------------------------------------
// ------ SawoffMosin.c - START --------------------
// ---------------------------------------------


class SawoffMosin extends RecipeBase
{■
■override void Init()
■{
■■m_Name = "#STR_sawoffbarrel0";
■■m_IsInstaRecipe = false;//should this recipe be performed instantly without animation
■■m_AnimationLength = 2;//animation length in relative time units
■■m_Specialty = 0.02;// value > 0 for roughness, value < 0 for precision
■■
■■
■■//conditions
■■m_MinDamageIngredient[0] = -1;//-1 = disable check
■■m_MaxDamageIngredient[0] = 3;//-1 = disable check
■■
■■m_MinQuantityIngredient[0] = -1;//-1 = disable check
■■m_MaxQuantityIngredient[0] = -1;//-1 = disable check
■■
■■m_MinDamageIngredient[1] = -1;//-1 = disable check
■■m_MaxDamageIngredient[1] = 3;//-1 = disable check
■■
■■m_MinQuantityIngredient[1] = -1;//-1 = disable check
■■m_MaxQuantityIngredient[1] = -1;//-1 = disable check
■■//----------------------------------------------------------------------------------------------------------------------
■■
■■//INGREDIENTS
■■//ingredient 1
■■InsertIngredient(0,"Mosin9130");//you can insert multiple ingredients this way
■■
■■m_IngredientAddHealth[0] = 0;// 0 = do nothing
■■m_IngredientSetHealth[0] = -1; // -1 = do nothing
■■m_IngredientAddQuantity[0] = 0;// 0 = do nothing
■■m_IngredientDestroy[0] = false;//true = destroy, false = do nothing
■■m_IngredientUseSoftSkills[0] = false;// set 'true' to allow modification of the values by softskills on this i
■■
■■//ingredient 2
■■InsertIngredient(1,"Hacksaw");//you can insert multiple ingredients this way
■
■■m_IngredientAddHealth[1] = -20;// 0 = do nothing
■■m_IngredientSetHealth[1] = -1; // -1 = do nothing
■■m_IngredientAddQuantity[1] = 0;// 0 = do nothing
■■m_IngredientDestroy[1] = false;// false = do nothing
■■m_IngredientUseSoftSkills[1] = true;// set 'true' to allow modification of the values by softskills on this in
■■
■■//----------------------------------------------------------------------------------------------------------------------
■■
■■//result1
■■/*
■■AddResult("SawedoffMosin9130");//add results here

■■m_ResultSetFullQuantity[0] = false;//true = set full quantity, false = do nothing
■■m_ResultSetQuantity[0] = -1;//-1 = do nothing
■■m_ResultSetHealth[0] = -1;//-1 = do nothing
■■m_ResultInheritsHealth[0] = 0;// (value) == -1 means do nothing; a (value) >= 0 means this result will in
■■m_ResultInheritsColor[0] = -1;// (value) == -1 means do nothing; a (value) >= 0 means this result classn
■■m_ResultToInventory[0] = -2;//(value) == -2 spawn result on the ground;(value) == -1 place anywhere in
■■m_ResultUseSoftSkills[0] = false;// set 'true' to allow modification of the values by softskills on this resu
■■m_ResultReplacesIngredient[0] = 0;// value == -1 means do nothing; a value >= 0 means this result wil
■■*/
```

```
// ---------------------------------------------
// ------ SawoffMosinPainted.c - START --------------------
// ---------------------------------------------


class SawoffMosinPainted extends RecipeBase
{■
■override void Init()
■{
■■m_Name = "#STR_sawoffbarrel0";
■■m_IsInstaRecipe = false;//should this recipe be performed instantly without animation
■■m_AnimationLength = 2;//animation length in relative time units
■■m_Specialty = 0.02;// value > 0 for roughness, value < 0 for precision
■■
■■
■■//conditions
■■m_MinDamageIngredient[0] = -1;//-1 = disable check
■■m_MaxDamageIngredient[0] = 3;//-1 = disable check
■■
■■m_MinQuantityIngredient[0] = -1;//-1 = disable check
■■m_MaxQuantityIngredient[0] = -1;//-1 = disable check
■■
■■m_MinDamageIngredient[1] = -1;//-1 = disable check
■■m_MaxDamageIngredient[1] = 3;//-1 = disable check
■■
■■m_MinQuantityIngredient[1] = -1;//-1 = disable check
■■m_MaxQuantityIngredient[1] = -1;//-1 = disable check
■■//----------------------------------------------------------------------------------------------------------------------
■■
■■//INGREDIENTS
■■//ingredient 1
■■InsertIngredient(0,"Mosin9130_Black");//you can insert multiple ingredients this way
■■InsertIngredient(0,"Mosin9130_Green");//you can insert multiple ingredients this way
■■InsertIngredient(0,"Mosin9130_Camo");//you can insert multiple ingredients this way
■■
■■m_IngredientAddHealth[0] = 0;// 0 = do nothing
■■m_IngredientSetHealth[0] = -1; // -1 = do nothing
■■m_IngredientAddQuantity[0] = 0;// 0 = do nothing
■■m_IngredientDestroy[0] = false;//true = destroy, false = do nothing
■■m_IngredientUseSoftSkills[0] = false;// set 'true' to allow modification of the values by softskills on this i
■■
■■//ingredient 2
■■InsertIngredient(1,"Hacksaw");//you can insert multiple ingredients this way
■
■■m_IngredientAddHealth[1] = -20;// 0 = do nothing
■■m_IngredientSetHealth[1] = -1; // -1 = do nothing
■■m_IngredientAddQuantity[1] = 0;// 0 = do nothing
■■m_IngredientDestroy[1] = false;// false = do nothing
■■m_IngredientUseSoftSkills[1] = true;// set 'true' to allow modification of the values by softskills on this in
■■
■■//----------------------------------------------------------------------------------------------------------------------
■■
■■//result1
■■/*
■■AddResult("SawedoffMosin9130_");//add results here

■■m_ResultSetFullQuantity[0] = false;//true = set full quantity, false = do nothing
■■m_ResultSetQuantity[0] = -1;//-1 = do nothing
■■m_ResultSetHealth[0] = -1;//-1 = do nothing
■■m_ResultInheritsHealth[0] = 0;// (value) == -1 means do nothing; a (value) >= 0 means this result will in
■■m_ResultInheritsColor[0] = 0;// (value) == -1 means do nothing; a (value) >= 0 means this result classn
■■m_ResultToInventory[0] = -2;//(value) == -2 spawn result on the ground;(value) == -1 place anywhere i
■■m_ResultUseSoftSkills[0] = false;// set 'true' to allow modification of the values by softskills on this resu
```

```
// ---------------------------------------------
// ------ SawoffShotgunIzh43.c - START --------------------
// ---------------------------------------------


class SawoffShotgunIzh43 extends RecipeBase
{■
■override void Init()
■{
■■m_Name = "#STR_sawoffbarrel0";
■■m_IsInstaRecipe = false;//should this recipe be performed instantly without animation
■■m_AnimationLength = 2;//animation length in relative time units
■■m_Specialty = 0.02;// value > 0 for roughness, value < 0 for precision
■■
■■
■■//conditions
■■m_MinDamageIngredient[0] = -1;//-1 = disable check
■■m_MaxDamageIngredient[0] = 3;//-1 = disable check
■■
■■m_MinQuantityIngredient[0] = -1;//-1 = disable check
■■m_MaxQuantityIngredient[0] = -1;//-1 = disable check
■■
■■m_MinDamageIngredient[1] = -1;//-1 = disable check
■■m_MaxDamageIngredient[1] = 3;//-1 = disable check
■■
■■m_MinQuantityIngredient[1] = -1;//-1 = disable check
■■m_MaxQuantityIngredient[1] = -1;//-1 = disable check
■■//----------------------------------------------------------------------------------------------------------------------
■■
■■//INGREDIENTS
■■//ingredient 1
■■InsertIngredient(0,"Izh43Shotgun");//you can insert multiple ingredients this way
■■
■■m_IngredientAddHealth[0] = 0;// 0 = do nothing
■■m_IngredientSetHealth[0] = -1; // -1 = do nothing
■■m_IngredientAddQuantity[0] = 0;// 0 = do nothing
■■m_IngredientDestroy[0] = false;//true = destroy, false = do nothing
■■m_IngredientUseSoftSkills[0] = false;// set 'true' to allow modification of the values by softskills on this i
■■
■■//ingredient 2
■■InsertIngredient(1,"Hacksaw");//you can insert multiple ingredients this way
■
■■m_IngredientAddHealth[1] = -20;// 0 = do nothing
■■m_IngredientSetHealth[1] = -1; // -1 = do nothing
■■m_IngredientAddQuantity[1] = 0;// 0 = do nothing
■■m_IngredientDestroy[1] = false;// false = do nothing
■■m_IngredientUseSoftSkills[1] = true;// set 'true' to allow modification of the values by softskills on this in
■■
■■//----------------------------------------------------------------------------------------------------------------------
■■
■■//result1
■■/*
■■AddResult("SawedoffIzh43Shotgun");//add results here

■■m_ResultSetFullQuantity[0] = false;//true = set full quantity, false = do nothing
■■m_ResultSetQuantity[0] = -1;//-1 = do nothing
■■m_ResultSetHealth[0] = -1;//-1 = do nothing
■■m_ResultInheritsHealth[0] = 0;// (value) == -1 means do nothing; a (value) >= 0 means this result will in
■■m_ResultInheritsColor[0] = -1;// (value) == -1 means do nothing; a (value) >= 0 means this result classn
■■m_ResultToInventory[0] = -2;//(value) == -2 spawn result on the ground;(value) == -1 place anywhere in
■■m_ResultUseSoftSkills[0] = false;// set 'true' to allow modification of the values by softskills on this resu
■■m_ResultReplacesIngredient[0] = 0;// value == -1 means do nothing; a value >= 0 means this result wil
■■*/
```

```
// ---------------------------------------------
// ------ SawWoodenLog.c - START --------------------
// ---------------------------------------------


class SawWoodenLog extends RecipeBase
{■
■override void Init()
■{
■■m_Name = "#STR_sawwoodenlog0";
■■m_IsInstaRecipe = false;//should this recipe be performed instantly without animation
■■m_AnimationLength = 2;//animation length in relative time units
■■m_Specialty = 0.02;// value > 0 for roughness, value < 0 for precision
■■
■■
■■//conditions
■■m_MinDamageIngredient[0] = -1;//-1 = disable check
■■m_MaxDamageIngredient[0] = 3;//-1 = disable check
■■
■■m_MinQuantityIngredient[0] = -1;//-1 = disable check
■■m_MaxQuantityIngredient[0] = -1;//-1 = disable check
■■
■■m_MinDamageIngredient[1] = -1;//-1 = disable check
■■m_MaxDamageIngredient[1] = 3;//-1 = disable check
■■
■■m_MinQuantityIngredient[1] = -1;//-1 = disable check
■■m_MaxQuantityIngredient[1] = -1;//-1 = disable check
■■//----------------------------------------------------------------------------------------------------------------------
■■
■■//INGREDIENTS
■■//ingredient 1
■■InsertIngredient(0,"WoodenLog");//you can insert multiple ingredients this way
■■
■■m_IngredientAddHealth[0] = 0;// 0 = do nothing
■■m_IngredientSetHealth[0] = -1; // -1 = do nothing
■■m_IngredientAddQuantity[0] = -1;// 0 = do nothing
■■m_IngredientDestroy[0] = false;//true = destroy, false = do nothing
■■m_IngredientUseSoftSkills[0] = false;// set 'true' to allow modification of the values by softskills on this i
■■
■■//ingredient 2
■■InsertIngredient(1,"Hacksaw");//you can insert multiple ingredients this way
■■InsertIngredient(1,"HandSaw");//you can insert multiple ingredients this way
■■InsertIngredient(1,"WoodAxe");//you can insert multiple ingredients this way
■■InsertIngredient(1,"Hatchet");//you can insert multiple ingredients this way
■■InsertIngredient(1,"FirefighterAxe");//you can insert multiple ingredients this way
■
■■m_IngredientAddHealth[1] = -10;// 0 = do nothing
■■m_IngredientSetHealth[1] = -1; // -1 = do nothing
■■m_IngredientAddQuantity[1] = 0;// 0 = do nothing
■■m_IngredientDestroy[1] = false;// false = do nothing
■■m_IngredientUseSoftSkills[1] = true;// set 'true' to allow modification of the values by softskills on this in
■■
■■//----------------------------------------------------------------------------------------------------------------------
■■
■■//result1
■■AddResult("Firewood");//add results here

■■m_ResultSetFullQuantity[0] = false;//true = set full quantity, false = do nothing
■■m_ResultSetQuantity[0] = -1;//-1 = do nothing
■■m_ResultSetHealth[0] = -1;//-1 = do nothing
■■m_ResultInheritsHealth[0] = 0;// (value) == -1 means do nothing; a (value) >= 0 means this result will in
■■m_ResultInheritsColor[0] = -1;// (value) == -1 means do nothing; a (value) >= 0 means this result class
■■m_ResultToInventory[0] = -2;//(value) == -2 spawn result on the ground;(value) == -1 place anywhere i
```

```
// ---------------------------------------------
// ------ SceneData.c - START --------------------
// ---------------------------------------------


class SceneData
{
■protected string■m_NameScene;
■protected string■m_NameMission;
■protected float ■m_InitTime;
■protected int ■■m_DateInitYear;
■protected int■■m_DateInitMonth;
■protected int■■m_DateInitDay;
■protected int■■m_DateInitHour;
■protected int■■m_DateInitMinute;
■protected float ■m_WeaterInitOvercast;
■protected float ■m_WeaterInitRain;
■protected float ■m_WeaterInitFog;
■protected float ■m_WeaterInitWindForce;
■
■protected ref array<ref SceneObject>■m_Objects;
■protected ref array<ref ScenePlayer>■m_Players;
■protected ref array<ref SceneObject>■m_AllObjs;
■
■//=========================================
■// Constructor -> SceneData
■//=========================================
■void SceneData()
■{
■■m_Objects = new array<ref SceneObject>;
■■m_Players = new array<ref ScenePlayer>;
■■m_AllObjs = new array<ref SceneObject>;
■}
■
■//=========================================
■// GetSceneObjects
■//=========================================
■array<ref SceneObject> GetSceneObjects()
■{
■■return m_Objects;
■}
■
■//=========================================
■// GetScenePlayers
■//=========================================
■array<ref ScenePlayer> GetScenePlayers()
■{
■■return m_Players;
■}
■
■//=========================================
■// GetSceneObjectsAll
■//=========================================
■array<ref SceneObject> GetSceneObjectsAll()
■{
■■array<ref ScenePlayer> scene_players = GetScenePlayers();
■■array<ref SceneObject> scene_object = GetSceneObjects();
■■
■■m_AllObjs.Clear();
■■
■■foreach (auto p: scene_players)
■■{
■■■m_AllObjs.Insert(p);
```

```
// ---------------------------------------------
// ------ SceneEditorMenu.c - START --------------------
// ---------------------------------------------


class SceneEditorMenu extends UIScriptedMenu
{■
//------------------------------------------------------------------------------
// >> Public Scope
■static const int POPUP_ID_SCENE_MANAGER■■= 0;
■static const int POPUP_ID_SCENE_SETTINGS■= 1;
■static const int POPUP_ID_SCENE_NEW■■■= 2;
■static const int POPUP_ID_SCENE_RENAME■■= 3;
■static const int POPUP_ID_SCENE_DELETE■■= 4;
■static const int POPUP_ID_NOTIFY■■■= 5;
■static const int POPUP_ID_EDITOR_SETTINGS■= 6;
■static const int POPUP_ID_INIT_SCRIPT■■= 7;
■static const int POPUP_ID_POSITION_MANAGER■■= 8;
■static const int POPUP_ID_PRESET_NEW = 9;
■static const int POPUP_ID_PRESET_RENAME = 10;
■static const int POPUP_ID_CONFIGS = 11;
■const string CONST_DEFAULT_PRESET_PREFIX = "[Default]";■
■
■// Render specific Preset Items
■void RenderPresets()
■{
■■m_PresetsTextListbox.ClearItems();

■■int i;
■■TBoolArray preset_params;

■■// load fixed presets list
■■TStringArray presets_array = m_ConfigDebugProfileFixed.GetPresets();
■■for ( i = 0; i < presets_array.Count(); i++ )
■■{
■■■m_PresetsTextListbox.AddItem( "["+presets_array.Get(i)+"]", new PresetParams ( presets_array.Get
■■}

■■// load custom presets list
■■TStringArray custom_presets_array = m_ConfigDebugProfile.GetPresets();
■■for ( i = 0; i < custom_presets_array.Count(); i++ )
■■{
■■■m_PresetsTextListbox.AddItem( custom_presets_array.Get(i), new PresetParams ( custom_presets_
■■}

■■string default_preset = m_ConfigDebugProfile.GetDefaultPreset();
■■if ( default_preset != "" )
■■{
■■■// if is fixed
■■■int index = GetPresetIndexByName( default_preset );
■■■if ( IsPresetFixed( default_preset) )
■■■{
■■■■default_preset = "[" + default_preset + "]";
■■■}
■■■PresetParams preset_params_array;
■■■if( index > -1 && index < m_PresetsTextListbox.GetNumItems() )
■■■{
■■■■m_PresetsTextListbox.GetItemData( index, 0, preset_params_array );
■■■■m_PresetsTextListbox.SetItem( index, default_preset + CONST_DEFAULT_PRESET_PREFIX, pre
■■■}
■■}
■}
■
```

```
// ---------------------------------------------
// ------ SceneObject.c - START --------------------
// ---------------------------------------------


class SceneObject
{
■static const int■COLOR_OBJ_BBOX_NORMAL■= 0x00000000;
■static const int■COLOR_OBJ_BBOX_SELECT■= 0x1f007C00;
■
■protected EntityAI■■■■■■m_ObjectPtr;
■protected Shape■■■■■■■■m_DebugShapeBBox;
■protected string■■■■■■■m_InitScript;
■protected string■■■■■■■m_ObjectName;
■protected string■■■■■■■m_ObjectNameOrigin;
■
■protected ref array<SceneObject>■■m_LinkedSceneObjects;
■protected ref map<SceneObject, Shape>■m_LinkedSceneObjectsShapes;
■
■ref array<int>■■■■■■■m_LinkedSceneObjectsIndices;
■
■//=========================================
■// SceneObject
■//=========================================
■SceneObject Init(string obj_name, vector pos)
■{
■■if (obj_name != STRING_EMPTY)
■■{
■■■m_ObjectNameOrigin = obj_name;
■■■
■■■bool is_ai = GetGame().IsKindOf(obj_name, "DZ_LightAI");
■■■
■■■PluginDeveloper module_dev = PluginDeveloper.Cast(GetPlugin(PluginDeveloper));
■■■EntityAI e = module_dev.SpawnEntityOnGroundPos(PluginSceneManager.PLAYER, obj_name, 100,
■■■
■■■if (e != NULL)
■■■{
■■■■if (e.IsInherited(ItemBase))
■■■■{■■
■■■■■ItemBase item = ItemBase.Cast(e);
■■■■■if (item.HasQuantity())
■■■■■■item.SetQuantity(item.GetQuantityMax());
■■■■}
■■■■
■■■■m_ObjectName = e.GetType();
■■■■LinkEntityAI(e);
■■■}
■■■else if (obj_name != "player")
■■■{
■■■■return NULL;
■■■}
■■}
■■
■■m_LinkedSceneObjects = new array<SceneObject>;
■■m_LinkedSceneObjectsShapes = new map<SceneObject, Shape>;
■■m_LinkedSceneObjectsIndices = new array<int>;
■
■■return this;
■}
■
■//--------------------------------------
■// GetObject
■//--------------------------------------
```

```
// ---------------------------------------------
// ------ ScenePlayer.c - START --------------------
// ---------------------------------------------


class ScenePlayer extends SceneObject
{
■//---------------------------------------
■// IsPlayer
■//---------------------------------------
■override bool IsPlayer()
■{
■■return true;
■}
}


// ---------------------------------------------
// ------ ScenePlayer.c - END ----------------------
// ---------------------------------------------


// ---------------------------------------------
// ------ Scout.c - START --------------------
// ---------------------------------------------


class Scout_Base : BoltActionRifle_ExternalMagazine_Base
{
■override RecoilBase SpawnRecoilObject()
■{
■■return new ScoutRecoil(this);
■}
■■■
■//Debug menu Spawn Ground Special
■override void OnDebugSpawn()
■{
■■super.OnDebugSpawn();
■■
■■GetInventory().CreateAttachment("ACOGOptic");
■}
};


// ---------------------------------------------
// ------ Scout.c - END ----------------------
// ---------------------------------------------


// ---------------------------------------------
// ------ ScoutRecoil.c - START --------------------
// ---------------------------------------------


class ScoutRecoil: RecoilBase
{
■override void Init()
■{
■■vector point_1;
■■vector point_2;
■■vector point_3;
■■vector point_4;
■■point_1[0] = m_Player.GetRandomGeneratorSyncManager().GetRandomInRange(RandomGeneratorS
// ---------------------------------------------
// ------ Screwdriver.c - START --------------------
// ---------------------------------------------


class Screwdriver extends ToolBase
{
■void Screwdriver()
■{
■■m_MineDisarmRate = 80;
■}
■
■override bool IsMeleeFinisher()
■{
■■return true;
■}
■
■override array<int> GetValidFinishers()
■{
■■return {EMeleeHitType.FINISHER_LIVERSTAB,EMeleeHitType.FINISHER_NECKSTAB};
■}
■
■override void SetActions()
■{
■■super.SetActions();
■■
■■AddAction(ActionBurnSewTarget);
■■AddAction(ActionUnrestrainTarget);
■■AddAction(ActionBurnSewSelf);
■■AddAction(ActionMineTreeBark);
■■AddAction(ActionSkinning);
■■AddAction(ActionLockAttachment);
■■AddAction(ActionDisarmMine);
■■AddAction(ActionDisarmExplosive);
■■AddAction(ActionMineRock1H);
■}
}


// ---------------------------------------------
// ------ Screwdriver.c - END ----------------------
// ---------------------------------------------



// ---------------------------------------------
// ------ script.c - START --------------------
// ---------------------------------------------


//some example "reference" variables for use in material editor
reference float g_testVariable1;
reference float g_testVariable2;
reference float g_testVariable3;

class TestClass
{
■//some example "reference" variables for use in material editor
■reference float testVar1;
■reference float testVar2;
■reference float testVar3;
}
```

```
// ---------------------------------------------
// ------ ScriptAnalytics.c - START --------------------
// ---------------------------------------------


// class binded to engine
class StatsEventMeasuresData
{
■string m_CharacterId;■■//!< character ID
■int m_TimeInterval;■■■//!< amount of real time in seconds covered by this event
■int m_DaytimeHour;■■■//!< current daytime in gameplay (hour in 24h format)
■vector m_PositionStart;■■//!< player world position at the start of interval
■vector m_PositionEnd;■■//!< player world position at the end of interval
■float m_DistanceOnFoot;■■//!< traveled distance on foot (meters) during interval
■
■float m_DistanceVehicle;■//!< traveled distance (meters) in vehicle during interval
■float m_TimeVONIn;■■■//!< amount of time in seconds with inbound VON during interval
■float m_TimeVONOut;■■■//!< amount of time in seconds with outbound VON during interval
■int m_CntLootAny;■■■//!< count of any loot collected during interval
■int m_CntLootFood;■■■//!< count of any food or consumables collected during interval
■int m_CntLootCloth;■■■//!< count of cloth collected during interval
■int m_CntLootFirearm;■■//!< count of firearms collected during interval
■int m_CntLootAmmo;■■■//!< count of ammo collected during interval
■int m_CntKillInfected;■■//!< count of infected kills during interval
■int m_CntConsumedFood;■■//!< count of consumed food during interval
■int m_CntConsumedWater;■■//!< count of consumed water during interval
■int m_HealthRestored;■■//!< number of health point restored during interval
■int m_CntFiredAmmo;■■■//!< firearm rounds fired during interval
■int m_CntCraftedItem;■■//!< number of items crafted during interval
■
■// listStatus■■■■// state metric of health, hunger, thirst, etc... when event is created/send
■int m_HealthStatus;■■■//!< state of health (current state)
■int m_BloodStatus;■■■//!< state of blood (current state)
■int m_SicknessStatus;■■//!< state of sickness (current state)
■int m_TemperatureStatus;■//!< state of temperature (current state)
■int m_FoodStatus;■■■//!< state of food (hunger) (current state)
■int m_HydrationStatus;■■//!< state of hydration (thirst) (current state)
};

// class binded to engine
class StatsEventDeathData
{
■string■■m_CharacterId;■■■//!< character ID
■int■■■m_CharacterLifetime;■//!< lifetime of character in seconds
■string■■m_Cause;■■■■//!< cause of death (hunger, sickness, player-killed, zombie-killed...)
■string■■m_WeaponName;■■■//!< name of weapon which caused death
■float■■m_Distance;■■■■//!< distance in meters (rounded) from spawn position to death position
■vector■■m_Position;■■■■//!< position of player when died
■
■int■■■m_ListDamages[5];■■//!< list of damages (5 values) during last n sec. For example: {20, 80, 0, 0
};

// class binded to engine
class StatsEventScoredKillData
{
■string■■m_CharacterId;■■//!< character ID
■string■■m_WeaponName;■■//!< name of weapon which killed player (victim)
■int■■■m_KillDistance;■■//!< distance in meters (rounded) between killer and victim
■vector■■m_PositionKiller;■//!< position of killer
■vector■■m_PositionVictim;■//!< position of victim
};

// class binded to engine
// ---------------------------------------------
// ------ ScriptCamera.c - START --------------------
// ---------------------------------------------


#ifdef GAME_TEMPLATE

[EditorAttribute("box", "GameLib/Scripted", "Script camera", "-0.25 -0.25 -0.25", "0.25 0.25 0.25", "255 0 0
class ScriptCameraClass
{

}

ScriptCameraClass ScriptCameraSource;

class ScriptCamera: GenericEntity
{
■[Attribute("60", "slider", "Field of view", "0 180 1")]
■float FOV;
■[Attribute("1", "editbox", "Near plane clip")]
■float NearPlane;
■[Attribute("4000", "editbox", "Far plane clip")]
■float FarPlane;
■
■[Attribute("1", "combobox", "Projection type", "", ParamEnumArray.FromEnum(CameraType) )]
■int Type;
■[Attribute("5", "slider", "Camera speed", "0 20 1")]
■float Speed;
■[Attribute("1", "combobox", "Free Fly", "", ParamEnumArray.FromEnum(EBool) )]
■bool FreeFly;
■[Attribute("0", "combobox", "Invert vertical", "", ParamEnumArray.FromEnum(EBool) )]
■bool Inverted;
■[Attribute("0", "slider", "Camera index", "0 31 1")]
■int Index;
■float m_MouseSensitivity = 0.001; // should be somewhere else.
■float m_GamepadSensitivity = 0.2; // should be somewhere else.
■int m_GamepadFreeFly;

■// debug variables
■int m_DbgListSelection = 0;
■ref array<string> m_DbgOptions = {"Perspective", "Orthographic"};

■void ScriptCamera(IEntitySource src, IEntity parent)
■{
■■SetFlags(EntityFlags.ACTIVE, false);
■■SetEventMask(EntityEvent.FRAME);

■■SetCameraVerticalFOV(Index, FOV);
■■SetCameraFarPlane(Index, FarPlane);
■■SetCameraNearPlane(Index, NearPlane);
■■SetCameraType(Index, Type);
■■m_DbgListSelection = Type - 1;
■■SetCamera(Index, GetOrigin(), GetYawPitchRoll());

■■vector camMat[4];
■■GetTransform(camMat);
■■SetCameraEx(Index, camMat);
■■m_GamepadFreeFly = FreeFly;
■}

■override protected void EOnFrame(IEntity other, float timeSlice) //EntityEvent.FRAME
■{
■■GetGame().GetInputManager().ActivateContext("ScriptCameraContext");
```

```
// ---------------------------------------------
// ------ ScriptConsole.c - START --------------------
// ---------------------------------------------


typedef Param3<string, bool, bool> PresetParams;//  param1 - ??, param2 - ??, param3 - ??
typedef Param5<bool, string, int, string, int> ConfigParams; // param1 - isCollapsed, param2 - string name
typedef Param6<bool, string, int, string, int, string> ConfigParamsEx; // param1 - isCollapsed, param2 - str

class ScriptConsole extends UIScriptedMenu
{
■static ScriptConsole m_This;
■static EffectSound ■■■m_SoundSet;
■static ref ConfigParamsEx m_ConfigData;
■const string  DEFAULT_POS_XYZ = "<1,2,3>";
■protected ref map<CheckBoxWidget, int>■m_ClassCheckboxes = new map<CheckBoxWidget, int>;
■vector m_MapPos;
■bool m_PlayerPosRefreshBlocked;
■const string CONST_DEFAULT_PRESET_PREFIX = "[Default]";
■static float DEBUG_MAP_ZOOM = 1;
■static int ITEMS_SELECTED_ROW = -1;
■static int PRESETS_SELECTED_ROW = 0;
■static int ITEMS_IN_PRESET_SELECTED_ROW;
■static int m_ObjectsScope = 2;
■static bool SHOW_OTHERS = 0;
■static bool CLEAR_IVN;
■static float DRAW_DISTANCE = 1000;
■static ref array<Shape> m_DebugShapes = new array<Shape>;
■static EntityAI m_PreviewEntity;

■ref array<Object> m_VicinityItems = new array<Object>;
■Widget m_WgtClassesConfig;
■int m_RunColor;
■ref TStringArray m_BaseConfigClasses = new TStringArray;
■ref TStringArray m_BaseConfigClassesToggled = new TStringArray;
■PluginItemDiagnostic m_ItemDisagPlugin;
■ref array<Widget> m_CategoryButtonsWidgets = new array<Widget>;
■ref array<string> m_CategoryButtonsNames = {"FIREARMS","MELEE","ATTACHMENTS","MAGAZINES
■SliderWidget m_TimeSlider;
■ref Timer m_LateInit = new Timer();
■bool m_ScriptServer;
■bool m_UpdateMap;
■int m_CategoryMask;
■bool m_FilterOrderReversed;
■bool m_AllowScriptOutput;
■
■static ref TStringArray m_ScriptOutputHistory = new TStringArray();
■
■ref Timer m_RefreshFilterTimer = new Timer();
■
■void ScriptConsole()
■{
■■m_This = this;
■■m_ModuleLocalEnscriptHistory ■= PluginLocalEnscriptHistory.Cast(GetPlugin(PluginLocalEnscriptHist
■■m_ModuleLocalEnscriptHistoryServer ■= PluginLocalEnscriptHistoryServer.Cast(GetPlugin(PluginLoca
■■#ifndef SERVER
■■if (GetGame() && GetGame().GetMission() && GetGame().GetMission().GetHud())
■■{
■■■GetGame().GetMission().GetHud().ShowHudPlayer(false);
■■■GetGame().GetMission().GetHud().ShowQuickbarPlayer(false);
■■}
■■#endif
■}
```

```
// ---------------------------------------------
// ------ ScriptConsoleAddPosition.c - START --------------------
// ---------------------------------------------


class ScriptConsoleAddLocation extends UIScriptedMenu
{
	void ScriptConsoleAddLocation()
	{

	}

	void ~ScriptConsoleAddLocation()
	{
	}
	
	void SetPosition(vector pos)
	{
		m_EditboxPos.SetText(pos.ToString());
	}

	override Widget Init()
	{
		m_ConfigDebugProfile = PluginConfigDebugProfile.Cast( GetPlugin(PluginConfigDebugProfile) );
		layoutRoot = GetGame().GetWorkspace().CreateWidgets("gui/layouts/script_console/script_console_a
		m_EditboxName = EditBoxWidget.Cast( layoutRoot.FindAnyWidget("LocationName") );
		m_EditboxPos = EditBoxWidget.Cast( layoutRoot.FindAnyWidget("Position") );
		m_Label = TextWidget.Cast( layoutRoot.FindAnyWidget("WindowLabel") );
		m_ClearButton = ButtonWidget.Cast( layoutRoot.FindAnyWidget("ButtonClear") );
		m_Label.SetText("ADD NEW LOCATION");

		return layoutRoot;
	}

	override bool OnClick(Widget w, int x, int y, int button)
	{
		super.OnClick(w, x, y, button);

		if (w.GetUserID() == IDC_OK)
		{
			string name = m_EditboxName.GetText();
			ScriptConsole console = ScriptConsole.Cast(GetGame().GetUIManager().FindMenu(MENU_SCRIPT
			if (!console.IsLocationNameAvailable(name) || name == "" || m_EditboxPos.GetText() == "")
				return false;
			m_ConfigDebugProfile.CustomLocationsAdd(name, m_EditboxPos.GetText().BeautifiedToVector());
			Close();
			
			console.RefreshLocations();
			return true;
		}
		else if (w.GetUserID() == IDC_CANCEL)
		{
			Close();
			return true;
		}
		else if (w == m_ClearButton)
		{
			m_EditboxPos.SetText("");
			return true;
		}

		return false;
	}
```

```
// ---------------------------------------------
// ------ ScriptConsoleNewPresetDialog.c - START --------------------
// ---------------------------------------------


class ScriptConsoleNewPresetDialog extends UIScriptedMenu
{
■void ScriptConsoleNewPresetDialog()
■{

■}

■void ~ScriptConsoleNewPresetDialog()
■{
■}

■override Widget Init()
■{
■■layoutRoot = GetGame().GetWorkspace().CreateWidgets("gui/layouts/script_console/script_console_d
■■m_Editbox = EditBoxWidget.Cast( layoutRoot.FindAnyWidget("PrimaryEditBox") );
■■m_Label = TextWidget.Cast( layoutRoot.FindAnyWidget("WindowLabel") );
■■m_Label.SetText("NEW PRESET");

■■return layoutRoot;
■}

■override bool OnClick(Widget w, int x, int y, int button)
■{
■■super.OnClick(w, x, y, button);

■■if (w.GetUserID() == IDC_OK)
■■{
■■■UIScriptedMenu ui_menu = GetGame().GetUIManager().FindMenu(MENU_SCRIPTCONSOLE);
■■■if ( ui_menu != NULL && m_Editbox.GetText() )
■■■{
■■■■ScriptConsole scripted_console = ScriptConsole.Cast( ui_menu );
■■■■scripted_console.NewPreset( m_Editbox.GetText());
■■■}

■■■Close();
■■■return true;
■■}
■■else if (w.GetUserID() == IDC_CANCEL)
■■{
■■■Close();
■■■return true;
■■}

■■return false;
■}
■
■EditBoxWidget m_Editbox;
■TextWidget m_Label;
■TextWidget m_Message;
}



// ---------------------------------------------
// ------ ScriptConsoleNewPresetDialog.c - END ----------------------
// ---------------------------------------------
// ---------------------------------------------
// ------ ScriptConsoleRenamePresetDialog.c - START --------------------
// ---------------------------------------------


class ScriptConsoleRenamePresetDialog extends UIScriptedMenu
{
■void ScriptConsoleRenamePresetDialog()
■{

■}

■void ~ScriptConsoleRenamePresetDialog()
■{
■}

■override Widget Init()
■{
■■layoutRoot = GetGame().GetWorkspace().CreateWidgets("gui/layouts/script_console/script_console_d
■■m_Editbox = EditBoxWidget.Cast( layoutRoot.FindAnyWidget("PrimaryEditBox") );
■■m_Label = TextWidget.Cast( layoutRoot.FindAnyWidget("WindowLabel") );
■■m_Message = TextWidget.Cast( layoutRoot.FindAnyWidget("MessageText") );

■■MissionBase mission = MissionBase.Cast( GetGame().GetMission() );

■■UIScriptedMenu ui_menu = GetGame().GetUIManager().FindMenu(MENU_SCRIPTCONSOLE);
■■if ( ui_menu != NULL )
■■{
■■■ScriptConsole scripted_console = ScriptConsole.Cast( ui_menu );

■■■m_Editbox.SetText( scripted_console.GetCurrentPresetName() );
■■■m_Label.SetText("RENAME PRESET");
■■■m_Message.SetText( scripted_console.GetCurrentPresetName() );
■■}

■■return layoutRoot;
■}

■override bool OnClick(Widget w, int x, int y, int button)
■{
■■super.OnClick(w, x, y, button);

■■if (w.GetUserID() == IDC_OK)
■■{
■■■UIScriptedMenu ui_menu = GetGame().GetUIManager().FindMenu(MENU_SCRIPTCONSOLE);
■■■if ( ui_menu != NULL )
■■■{
■■■■ScriptConsole scripted_console = ScriptConsole.Cast( ui_menu );
■■■■scripted_console.RenamePreset( m_Editbox.GetText() );
■■■■Close();
■■■■return true;
■■■}
■■}
■■else if (w.GetUserID() == IDC_CANCEL)
■■{
■■■Close();
■■■return true;
■■}

■■return false;
■}
■
■EditBoxWidget m_Editbox;
```

```
// ---------------------------------------------
// ------ ScriptedEntity.c - START --------------------
// ---------------------------------------------


enum TriggerShape
{
■BOX,
■SPHERE,
■CYLINDER,
}

class ScriptedEntity extends EntityAI
{
■/**
■\brief Sets collision properties for object
■■\param mins \p vector Min values of box
■■\param maxs \p vector Max values of box
■■\param radius \p float Radius of bounding sphere
■■\note This function is obsolete, use rather SetCollisionBox()
■*/
■proto native void SetClippingInfo(vector mins, vector maxs, float radius);
■
■/**
■\brief Sets collision box for object
■■\param mins \p vector Min values of box
■■\param maxs \p vector Max values of box
■■\note Automatically sets TriggerShape.BOX
■■\n usage :
■■@code
■■■vector mins = "-1 -1 -1";
■■■vector maxs = "1 1 1";
■■■SetCollisionBox(mins, maxs);
■■@endcode
■*/
■proto native void SetCollisionBox(vector mins, vector maxs);
■■
■/**
■\brief Sets collision sphere for object
■■\param radius \p float Radius of cylinder
■■\note Automatically sets TriggerShape.SPHERE
■■\n usage :
■■@code
■■■SetCollisionSphere(3);
■■@endcode
■*/
■proto native void SetCollisionSphere(float radius);
■
■/**
■\brief Sets collision cylinder for object
■■\param radius \p float Radius of cylinder
■■\param height \p float Height of cylinder
■■\note Automatically sets TriggerShape.CYLINDER
■■\n usage :
■■@code
■■■SetCollisionCylinder(3, 6);
■■@endcode
■*/
■proto native void SetCollisionCylinder(float radius, float height);■
■■
■//! Set the TriggerShape to be used, default is TriggerShape.BOX
■proto native void SetTriggerShape(TriggerShape shape);
■
```

```
// ---------------------------------------------
// ------ ScriptedLightBase.c - START --------------------
// ---------------------------------------------


/*
Please remember that:
-Lights work only on client side!
-Lights with Brightness or Radius of 0 (or less) are automatically deleted
-Lights are very performance heavy. Especially if they cast shadows. Use them carefully!

Script author: Boris Vacula
*/

class ScriptedLightBase extends EntityLightSource
{
■float ■■m_LifetimeStart;
■float ■■m_LifetimeEnd = -1; // -1 makes this light permanent
■float ■■m_FadeOutTime = -1;
■float ■■m_FadeInTime = -1;
■float ■■m_Radius;
■float ■■m_RadiusTarget;
■float ■■m_Brightness;
■float ■■m_BrightnessPulse; // flicker effect
■float ■■m_BrightnessPulseSpeed;
■float ■■m_BrightnessPulseAmplitudeMax;
■float ■■m_BrightnessPulseAmplitudeMin;
■float ■■m_BrightnessTarget;
■float ■■m_BrightnessSpeedOfChange = 1;
■float ■■m_RadiusSpeedOfChange = 1;
■float ■■m_OptimizeShadowsRadius = 0; // Within this range between the light source and camera the sl
■
■float ■■m_DancingShadowsAmplitude;
■float ■■m_DancingShadowsSpeed;
■
■float■■m_BlinkingSpeed;
■
■bool ■■m_IsDebugEnabled = false;
■
■Object ■■m_Parent; // Attachment parent
■vector ■■m_LocalPos; // Local position to my attachment parent
■vector ■■m_LocalOri; // Local orientation to my attachment parent
■vector ■■m_DancingShadowsLocalPos;
■
■ref Timer ■m_DeleteTimer;
■
■static ref set<ScriptedLightBase> m_NightTimeOnlyLights = new set<ScriptedLightBase>();
■
■//! Constructor. Everything here is executed before the constructor of all children.
■void ScriptedLightBase()
■{
■■m_LifetimeStart = GetGame().GetTime();
■■SetEnabled(true);
■■SetEventMask(EntityEvent.FRAME);
■■SetEventMask(EntityEvent.INIT);
■}
■
■void ~ScriptedLightBase()
■{
■■if (m_NightTimeOnlyLights)
■■{
■■■int index = m_NightTimeOnlyLights.Find(this);
■■■if (index != -1)
```

```
// ---------------------------------------------
// ------ ScriptInvokerTests.c - START --------------------
// ---------------------------------------------


class ScriptInvokerTests : TestFramework
{
■ref ScriptInvoker m_Invoker;
■int m_InvokeCount;
■
■//---------------------------------------------------------------------------
■// Ctor - Decides the tests to run
■//---------------------------------------------------------------------------
■void ScriptInvokerTests()
■{
■■m_Invoker = new ScriptInvoker();
■■
■■//AddInitTest("TestFirstUnique");
■■//AddInitTest("TestSecondUnique");
■■//AddInitTest("TestInsertRemoveUnique");
■■//AddInitTest("TestInsertUniqueImmediate");
■■//AddInitTest("TestClearRunning");
■■//AddInitTest("TestInvokeRunning");
■■AddInitTest("TestInsertRunning");
■}
■
■//---------------------------------------------------------------------------
■// Dtor
■//---------------------------------------------------------------------------
■void ~ScriptInvokerTests()
■{
■■
■}
■
■//---------------------------------------------------------------------------
■// Tests
■//---------------------------------------------------------------------------
■// Test first insert flagged as unique
■TFResult TestFirstUnique()
■{
■■InvokeReset();
■■
■■bool insert1 = m_Invoker.Insert(InvokeLog, EScriptInvokerInsertFlags.UNIQUE);
■■Assert(insert1);
■■bool insert2 = m_Invoker.Insert(InvokeLog);
■■Assert(!insert2);
■■
■■m_Invoker.Invoke("TestFirstUnique");
■■bool count = Assert(m_InvokeCount == 1);
■■
■■InvokeReset();
■■
■■return BTFR(insert1 && !insert2 && count);
■}
■
■//---------------------------------------------------------------------------
■// Test second insert flagged as unique
■TFResult TestSecondUnique()
■{
■■InvokeReset();
■■
■■bool insert1 = m_Invoker.Insert(InvokeLog);
■■Assert(insert1);
```

```
// ---------------------------------------------
// ------ ScriptLight.c - START --------------------
// ---------------------------------------------


#ifdef GAME_TEMPLATE

[EditorAttribute("box", "GameLib/Scripted", "Script light", "-0.25 -0.25 -0.25", "0.25 0.25 0.25", "255 0 0 255
class ScriptLightClass
{

}

ScriptLightClass ScriptLightSource;

class ScriptLight: GenericEntity
{
■[Attribute("1", "flags", "Flags", "", { ParamEnum("Point", "1"), ParamEnum("Spot", "2"), ParamEnum("Dire
■int Flags;
■[Attribute("1", "combobox", "Type", "", { ParamEnum("Point", "1"), ParamEnum("Spot", "2"), ParamEnum(
■int Type;
■[Attribute("1", "editbox", "Radius", "", NULL )]
■float Radius;
■[Attribute("1 1 1", "color", "Color", "", NULL )]
■vector Color;
■[Attribute("1", "editbox", "Intensity", "", NULL )]
■float Intensity;
■HLIGHT m_light;
■
■void ScriptLight(IEntitySource src, IEntity parent)
■{
■■//SetFlags(this, EntityFlags.ACTIVE | EntityFlags.SOLID | EntityFlags.VISIBLE);
■■m_light = AddLight(this, LightType.POINT, LightFlags.DYNAMIC|LightFlags.CASTSHADOW, Radius, C
■}

■void ~ScriptLight()
■{
■■if(m_light)
■■{
■■■RemoveLight(m_light);
■■}
■}
}

#endif


// ---------------------------------------------
// ------ ScriptLight.c - END ----------------------
// ---------------------------------------------


// ---------------------------------------------
// ------ ScriptModel.c - START --------------------
// ---------------------------------------------


#ifdef GAME_TEMPLATE

[EditorAttribute("box", "GameLib/Scripted", "Script model", "-0.25 -0.25 -0.25", "0.25 0.25 0.25", "255 0 0 2
class ScriptModelClass
{
```

```
// ----------------------------------------------
// ------ ScrollBarContainer.c - START --------------------
// ----------------------------------------------


// ---------------------------------------------------------
class ScrollBarContainer : ScriptedWidgetEventHandler
{
■reference bool Invert;
■protected Widget Content;
■protected Widget ScrollBar;
■protected Widget Scroller;
■protected Widget m_root;
■
■const int WHEEL_STEP = 20;
■protected float■■m_root_height;
■protected float■■m_content_height;
■protected float■■m_position;
■protected bool■■m_scrolling;
■protected float■■m_scrolling_start_pos;
■protected int■■m_scrolling_mouse_pos;
■■
■void ~ScrollBarContainer()
■{
■■//if(GetGame() != NULL)
■■//GetGame().GetDragQueue().RemoveCalls(this);
■}
■
■void ScrollFixedAmount( bool down, float amount )
■{
■■m_root.Update();
■■Content.Update();
■■float width;
■
■■m_root.GetScreenSize(width, m_root_height);
■■Content.GetScreenSize(width, m_content_height);
■■
■■float diff = m_root_height / m_content_height;
■■float one_percent = diff / 100;
■■float percents = amount / m_content_height;
■■//float step = (1.0 / (m_content_height - m_root_height)) * WHEEL_STEP;
■■float step = (percents/100);
■■if(down)
■■m_position += 1 * ( percents + 0.05 );
■■else
■■m_position -= 1 * ( percents + 0.05 );
■■
■■if (m_position < 0) m_position = 0;
■■if (m_position > 1 - diff) m_position = 1 - diff;
■■UpdateScroller();
■}
■
■void ScrollToPos( float pos )
■{
■■m_root.Update();
■■Content.Update();
■■float width;
■
■■m_root.GetScreenSize(width, m_root_height);
■■Content.GetScreenSize(width, m_content_height);
■■
■■float diff = m_root_height / m_content_height;
■■float percents = pos / m_content_height;
```

```
// ---------------------------------------------
// ------ Sedan_02.c - START --------------------
// ---------------------------------------------


class Sedan_02 extends CarScript
{
■protected ref UniversalTemperatureSource m_UTSource;
■protected ref UniversalTemperatureSourceSettings m_UTSSettings;
■protected ref UniversalTemperatureSourceLambdaEngine m_UTSLEngine;

■void Sedan_02()
■{
■■//m_dmgContactCoef■■= 0.130;

■■m_EngineStartOK■■■= "Sedan_02_engine_start_SoundSet";
■■m_EngineStartBattery■= "Sedan_02_engine_failed_start_battery_SoundSet";
■■m_EngineStartPlug■■= "Sedan_02_engine_failed_start_sparkplugs_SoundSet";
■■m_EngineStartFuel■■= "Sedan_02_engine_failed_start_fuel_SoundSet";
■■m_EngineStopFuel■■= "offroad_engine_stop_fuel_SoundSet";

■■m_CarDoorOpenSound■■= "offroad_door_open_SoundSet";
■■m_CarDoorCloseSound■■= "offroad_door_close_SoundSet";
■■
■■m_CarHornShortSoundName = "Sedan_02_Horn_Short_SoundSet";
■■m_CarHornLongSoundName■= "Sedan_02_Horn_SoundSet";
■■
■■SetEnginePos("0 0.7 -1.7");
■}
■
■override void EEInit()
■{■■
■■super.EEInit();
■■
■■if (GetGame().IsServer() || !GetGame().IsMultiplayer())
■■{
 ■■■m_UTSSettings ■■■■■= new UniversalTemperatureSourceSettings();
■■■m_UTSSettings.m_ManualUpdate ■= true;
■■■m_UTSSettings.m_TemperatureMin■= 0;
■■■m_UTSSettings.m_TemperatureMax■= 30;
■■■m_UTSSettings.m_RangeFull■■= 0.5;
■■■m_UTSSettings.m_RangeMax■■= 2;
■■■m_UTSSettings.m_TemperatureCap■= 25;
■■■
■■■m_UTSLEngine■■■■■= new UniversalTemperatureSourceLambdaEngine();
■■■m_UTSource■■■■■■= new UniversalTemperatureSource(this, m_UTSSettings, m_UTSLEngine);
■■}■■
■}
■
■override void OnEngineStart()
■{
■■super.OnEngineStart();

■■if (GetGame().IsServer() || !GetGame().IsMultiplayer())
■■{
■■■m_UTSource.SetDefferedActive(true, 20.0);
■■}
■}
■
■override void OnEngineStop()
■{
■■super.OnEngineStop();
```

```
// ---------------------------------------------
// ------ Sedan_02FrontLight.c - START --------------------
// ---------------------------------------------


class Sedan_02FrontLight extends CarLightBase
{
■void Sedan_02FrontLight()
■{
■■m_SegregatedBrightness = 4;
■■m_SegregatedRadius = 75;
■■m_SegregatedAngle = 90;
■■m_SegregatedColorRGB = Vector(0.95, 0.95, 0.78);
■■
■■m_AggregatedBrightness = 8;
■■m_AggregatedRadius = 100;
■■m_AggregatedAngle = 120;
■■m_AggregatedColorRGB = Vector(0.95, 0.95, 0.78);
■■
■■FadeIn(0.1);
■■SetFadeOutTime(0.05);
■■
■■SegregateLight();
■}
}


// ---------------------------------------------
// ------ Sedan_02FrontLight.c - END ----------------------
// ---------------------------------------------


// ---------------------------------------------
// ------ Sedan_02RearLight.c - START --------------------
// ---------------------------------------------


class Sedan_02RearLight extends CarRearLightBase
{
■void Sedan_02RearLight()
■{
■■// Brake light only
■■m_SegregatedBrakeBrightness = 5;
■■m_SegregatedBrakeRadius = 6;
■■m_SegregatedBrakeAngle = 160;
■■m_SegregatedBrakeColorRGB = Vector(0.95, 0.16, 0.05);
■■
■■// Reverse light only
■■m_SegregatedBrightness = 2;
■■m_SegregatedRadius = 10;
■■m_SegregatedAngle = 120;
■■m_SegregatedColorRGB = Vector(0.80, 0.85, 0.9);
■■
■■// Brake & Revese lights combined
■■m_AggregatedBrightness = 5;
■■m_AggregatedRadius = 10;
■■m_AggregatedAngle = 160;
■■m_AggregatedColorRGB = Vector(0.9, 0.4, 0.5);
■■
■■FadeIn(0.1);
■■SetFadeOutTime(0.05);

■■SetVisibleDuringDaylight(false);
```

```
// ---------------------------------------------
// ------ SeedPackBase.c - START --------------------
// ---------------------------------------------


class SeedPackBase extends Inventory_Base
{
■private static const float PACK_DAMAGE_TOLERANCE = 0.5;
■
■void SeedPackBase()
■{■
■}

■void EmptySeedPack( PlayerBase player )
■{

■■string pack_type = GetType();
■■string seeds_type = "";
■■
■■GetGame().ConfigGetText( "cfgVehicles " + pack_type + " Horticulture ContainsSeedsType", seeds_typ
■■
■■int seeds_quantity_max = GetGame().ConfigGetInt( "cfgVehicles " + pack_type + " Horticulture Contair
■■int seeds_quantity = seeds_quantity_max;
■■
■■seeds_quantity = Math.Round( seeds_quantity_max * GetHealth01("","") );
■
■■if ( seeds_quantity < 1 )
■■{
■■■seeds_quantity = 1;
■■}
■■
■■if (player)
■■{
■■■EmptySeedsPackLambda lambda = new EmptySeedsPackLambda(this, seeds_type, player, seeds_
■■■player.ServerReplaceItemInHandsWithNew(lambda);
■■}
■■else
■■{
■■■vector pos = GetPosition();
■■■GetGame().CreateObjectEx(seeds_type, pos, ECE_PLACE_ON_SURFACE);
■■■GetGame().ObjectDelete( this );
■■}
■}
■
■override void SetActions()
■{
■■super.SetActions();
■■
■■AddAction(ActionEmptySeedsPack);
■}
}

class EmptySeedsPackLambda : ReplaceItemWithNewLambdaBase
{
■int m_ItemCount;
■
■void EmptySeedsPackLambda (EntityAI old_item, string new_item_type, PlayerBase player, int count)
■{
■■m_ItemCount = count;
■}

■override void CopyOldPropertiesToNew (notnull EntityAI old_item, EntityAI new_item)
■{
```

```
// ---------------------------------------------
// ------ SensesAIEvaluate.c - START --------------------
// ---------------------------------------------


class NoiseAIEvaluate
{
■static float SURFACE_NOISE_WEIGHT = 0.25;
■
■static float GetNoiseMultiplier(DayZPlayerImplement playerImplement)
■{
■■float speedNoise■= GetNoiseMultiplierByPlayerSpeed(playerImplement);
■■float shoesNoise■= GetNoiseMultiplierByShoes(playerImplement);
■■float surfaceNoise■= GetNoiseMultiplierBySurface(playerImplement);
■■
■■surfaceNoise ■■*= SURFACE_NOISE_WEIGHT;
■■float avgNoise ■■= (shoesNoise + surfaceNoise)/(1 + SURFACE_NOISE_WEIGHT);
■■avgNoise ■■■*= speedNoise;
■■
■■return avgNoise;
■}
■
■//Noise multiplier based on player speed
■static float GetNoiseMultiplierByPlayerSpeed(DayZPlayerImplement playerImplement)
■{
■■HumanMovementState hms = new HumanMovementState();
■■
■■playerImplement.GetMovementState(hms);
■■
■■if ( playerImplement.GetCommand_Move() && playerImplement.GetCommand_Move().IsInRoll() )
■■{
■■■// When rolling we are prone, so we load that Noise value, hence we multiply
■■■return PlayerConstants.AI_NOISE_ROLL;
■■}
■■
■■switch ( AITargetCallbacksPlayer.StanceToMovementIdxTranslation(hms) )
■■{■■■
■■■case DayZPlayerConstants.MOVEMENTIDX_IDLE:
■■■■return PlayerConstants.AI_NOISE_IDLE;
■■■
■■■case DayZPlayerConstants.MOVEMENTIDX_WALK:
■■■■return PlayerConstants.AI_NOISE_WALK;
■■■
■■■case DayZPlayerConstants.MOVEMENTIDX_CROUCH_RUN:
■■■■return PlayerConstants.AI_NOISE_CROUCH_RUN;
■■■
■■■case DayZPlayerConstants.MOVEMENTIDX_RUN:
■■■■return PlayerConstants.AI_NOISE_RUN;
■■■
■■■case DayZPlayerConstants.MOVEMENTIDX_SPRINT:
■■■■return PlayerConstants.AI_NOISE_SPRINT;
■■}
■■
■■//Default return
■■return PlayerConstants.AI_NOISE_SPRINT;
■}
■
■
■//Noise multiplier based on type of boots
■static float GetNoiseMultiplierByShoes(DayZPlayerImplement playerImplement)
■{
■■switch ( playerImplement.GetBootsType() )
■■{
```

```
// ---------------------------------------------
// ------ Serializer.c - START --------------------
// ---------------------------------------------


//-----------------------------------------------------------------------------
/**
 \brief Serialization general interface. Serializer API works with:
■- primitive types: int, float, string, bool, vector
■- dynamic containers: array, set, map
■- static arrays
■- complex types: classes
 \note Serializer provides deep serialization (it serialize class memebers and their members etc). To avoid
 \par usage:
 @code
■class MyData
■{
■■int m_id;
■■autoptr map<string, float> m_values;
■
■■[NonSerialized()]
■■string m_dbg; // I don't want to serialize this variable
■}

■void Serialize(Serializer s)
■{
■■int statArray[4] = {6,9,2,3};
■■array<int> dynArray = {8,5,6,4};
■■autoptr MyData data = new MyData();
■
■■data.m_id = 965;
■■data.m_values = map<string, float>;
■■data.m_values.Insert("value1", 5.98);
■■data.m_values.Insert("value2", 4.36);
■
■■s.Write(10);
■■s.Write("Hello");
■■s.Write(statArray);
■■s.Write(dynArray);
■■s.Write(data);
■}

■void Deserialize(Serializer s)
■{
■■int statArray[4];
■■array<int> dynArray;
■■MyData data;
■■int someInt;
■■string someString;■
■
■■s.Read(someInt);
■■s.Read(someString);
■■s.Read(statArray);
■■s.Read(dynArray);
■■s.Read(data);
■}

 @endcode
 */
class Serializer: Managed
{
■proto bool Write(void value_out);
■proto bool Read(void value_in);
```

```
// ---------------------------------------------
// ------ ServerBrowserEntry.c - START --------------------
// ---------------------------------------------


class ServerBrowserEntry extends ScriptedWidgetEventHandler
{
■protected Widget■■■■■m_Root;
■protected Widget■■■■■m_Favorite;
■protected Widget■■■■■m_Expand;
■
■//Basic info
■protected TextWidget■■■■m_ServerName;
■protected TextWidget■■■■m_ServerPopulation;
■protected TextWidget■■■■m_ServerSlots;
■protected TextWidget■■■■m_ServerPing;
■protected ImageWidget■■■■m_ServerTime;
■protected ImageWidget■■■■m_ServerLock;
■protected ImageWidget■■■■m_ServerModIcon;
■protected ImageWidget■■■■m_ServerMaKIcon;
■
■//Detailed info
■protected TextWidget■■■■m_ServerShard;
■protected TextWidget■■■■m_ServerCharacterAlive;
■protected TextWidget■■■■m_ServerFriends;
■protected TextWidget■■■■m_ServerMode;
■protected TextWidget■■■■m_ServerBattleye;
■protected TextWidget■■■■m_ServerIP;
■protected TextWidget■■■■m_ServerAcceleration;
■protected TextWidget■■■■m_ServerMap;
■protected TextWidget■■■■m_ServerMods;
■protected ButtonWidget■■■■m_ServerModsExpand;
■protected ref array<string>■■■m_Mods;
■
■protected bool■■■■■■m_IsExpanded;
■protected bool■■■■■■m_IsFavorited;
■protected bool■■■■■■m_IsOnline;
■
■protected ref GetServersResultRow■m_ServerData;
■protected int■■■■■■m_Index;
■protected ServerBrowserTab■■■m_Tab;
■protected bool■■■■■■m_Selected;
■protected bool■■■■■■m_FirstExpand = true;
■
■void ServerBrowserEntry( Widget parent, int index, ServerBrowserTab tab )
■{■■
■■#ifdef PLATFORM_CONSOLE
■■■m_Root■■■■= GetGame().GetWorkspace().CreateWidgets( "gui/layouts/new_ui/server_browser/xb
■■#else
■■■m_Root■■■■= GetGame().GetWorkspace().CreateWidgets( "gui/layouts/new_ui/server_browser/pc
■■#endif
■■
■■m_Root.Enable( true );
■■m_Favorite■■■■= m_Root.FindAnyWidget( "favorite_button" );
■■m_Expand■■■■= m_Root.FindAnyWidget( "expand_button" );
■■m_ServerName■■■= TextWidget.Cast( m_Root.FindAnyWidget( "server_name" ) );
■■m_ServerPopulation■■= TextWidget.Cast( m_Root.FindAnyWidget( "server_population" ) );
■■m_ServerSlots■■■= TextWidget.Cast( m_Root.FindAnyWidget( "server_slots" ) );
■■m_ServerPing■■■= TextWidget.Cast( m_Root.FindAnyWidget( "server_ping" ) );
■■m_ServerTime■■■= ImageWidget.Cast( m_Root.FindAnyWidget( "server_time" ) );
■■m_ServerLock■■■= ImageWidget.Cast( m_Root.FindAnyWidget( "lock_icon" ) );
■■m_ServerModIcon■■■= ImageWidget.Cast( m_Root.FindAnyWidget( "modded_icon" ) );
■■m_ServerMaKIcon■■■= ImageWidget.Cast( m_Root.FindAnyWidget( "mandk_icon" ) );
```

```
// ---------------------------------------------
// ------ ServerBrowserFavoritesTabConsolePages.c - START --------------------
// ---------------------------------------------


class ServerBrowserFavoritesTabConsolePages extends ServerBrowserTabConsolePages
{■
■protected override void Construct(Widget parent, ServerBrowserMenuNew menu, TabType type)
■{
■■super.Construct(parent, menu, type);
■■
■■// disabling filter section
■■m_Root.FindAnyWidget("filters_content").Show(false);
■■m_Root.FindAnyWidget("reset_filter_button").Show(false);
■}
■
■override void OnLoadServersAsyncFinished()
■{
■■// m_TotalLoadedServers for FAVORITES tab is determined by total number of favorited servers
■■TStringArray favIds = m_Menu.GetFavoritedServerIds();
■■m_TotalLoadedServers = favIds.Count();
■■super.OnLoadServersAsyncFinished();
■}
■
■protected override void LoadEntries( int cur_page_index , GetServersResultRowArray page_entries )
■{
■■if (cur_page_index == 1)
■■{
■■■m_OnlineFavServers.Clear();
■■}
■■
■■super.LoadEntries(cur_page_index, page_entries);
■}
■
■protected override void LoadExtraEntries(int index)
■{
■■if ( !m_Menu || m_Menu.GetServersLoadingTab() != m_TabType )
■■{
■■■return;
■■}
■■
■■// m_PagesCount for FAVORITES tab is determined by total number of favorited servers
■■TStringArray favIds = m_Menu.GetFavoritedServerIds();
■■m_PagesCount = Math.Ceil((float)favIds.Count() /  SERVER_BROWSER_PAGE_SIZE);
■■
■■// offlineFavIds will always have same order, even across pages,
■■// to ensure we display only fav servers that HAVEN'T been displayed yet
■■TStringArray offlineFavIds = new TStringArray();
■■offlineFavIds.Reserve(favIds.Count() - m_OnlineFavServers.Count());
■■foreach (string ipPort : favIds)
■■{
■■■if (m_OnlineFavServers.Find(ipPort) == -1)
■■■{
■■■■offlineFavIds.Insert(ipPort);
■■■}
■■}
■■
■■// appending offline servers to server list
■■int totalServersAlreadyShown = (GetCurrentPage() - 1) * SERVER_BROWSER_PAGE_SIZE + index;
■■int startingIndex = totalServersAlreadyShown - m_OnlineFavServers.Count();
■■for (int i = startingIndex; i < offlineFavIds.Count(); ++i)
■■{
■■■string favServerId = offlineFavIds[i];
```

```
// ---------------------------------------------
// ------ ServerBrowserFavoritesTabPc.c - START --------------------
// ---------------------------------------------


class ServerBrowserFavoritesTabPc extends ServerBrowserTabPc
{
■// value = ip, name, connection port, query port
■private ref array<ref CachedServerInfo> m_CachedFavoriteServerInfo;
■
■protected override void Construct(Widget parent, ServerBrowserMenuNew menu, TabType type)
■{
■■m_CachedFavoriteServerInfo = new array<ref CachedServerInfo>();
■■
■■super.Construct(parent, menu, type);
■■DisableFilters();
■}

■protected override void LoadExtraEntries(int index)
■{
■■array<ref Param2<string, CachedServerInfo>> relevantCachedInfo = new array<ref Param2<string, Ca
■■foreach (CachedServerInfo cachedInfo : m_CachedFavoriteServerInfo)
■■{
■■■string serverId = GetConnEndPoint(cachedInfo);
■■■
■■■// ensure the cached server is still favorited
■■■if (!m_Menu.IsFavorited(GetQueryEndPoint(cachedInfo)))
■■■{
■■■■continue;
■■■}
■■■
■■■if (m_OnlineFavServers.Find(serverId) > -1)
■■■{
■■■■continue;
■■■}
■■■
■■■relevantCachedInfo.Insert(new Param2<string, CachedServerInfo>(serverId, cachedInfo));
■■}
■■
■■// adding FAVORITED, OFFLINE servers to favorites tab
■■int totalServersAlreadyShown = m_PageIndex * SERVERS_VISIBLE_COUNT + index;
■■int startingIndex = totalServersAlreadyShown - m_TotalLoadedServers;
■■m_TotalLoadedServers += relevantCachedInfo.Count();
■■for (int i = startingIndex; i < relevantCachedInfo.Count(); ++i)
■■{
■■■if (index >= SERVERS_VISIBLE_COUNT)
■■■{
■■■■break;
■■■}
■■■
■■■Param2<string, CachedServerInfo> relevantEntry = relevantCachedInfo[i];
■■■
■■■cachedInfo = relevantEntry.param2;■■■
■■■string ip = cachedInfo.param1;
■■■string cachedName = cachedInfo.param2;
■■■int connPort = cachedInfo.param3;
■■■int queryPort = cachedInfo.param4;

■■■// do NOT insert offlineRow into m_EntriesSorted[m_SortType]!!
■■■// we assume that rows already in m_EntriesSorted[m_SortType] are ONLINE,
■■■// if we add offline info to m_EntriesSorted[m_SortType] then we cannot differente
■■■// between online and offline servers■■■
■■■GetServersResultRow offlineRow = new GetServersResultRow();
```

```
// ---------------------------------------------
// ------ ServerBrowserFilterContainer.c - START --------------------
// ---------------------------------------------


class ServerBrowserFilterContainer extends ScriptedWidgetEventHandler
{
■ref map<string, string>■■m_Options = new map<string, string>;
■
■protected EditBoxWidget■■■■■m_SearchByName;
■protected EditBoxWidget■■■■■m_SearchByIP;
■■■ref OptionSelectorMultistate■m_CharacterAliveFilter;
■■■ref OptionSelectorMultistate■m_SortingFilter;
■■■ref OptionSelectorMultistate■m_RegionFilter;
■■■ref OptionSelectorMultistate■m_PingFilter;
■■■ref OptionSelector■■■■m_DLCFilter;
■■■ref OptionSelector■■■■m_FavoritedFilter;
■■■ref OptionSelector■■■■m_FriendsPlayingFilter;
■■■ref OptionSelector■■■■m_BattleyeFilter;
■■■ref OptionSelector■■■■m_PasswordFilter;
■■■ref OptionSelector■■■■m_WhitelistFilter;
■■■ref OptionSelector■■■■m_KeyboardFilter;
■■■ref OptionSelector■■■■m_PreviouslyPlayedFilter;
■■■ref OptionSelector■■■■m_VersionMatchFilter;
■■■ref OptionSelector■■■■m_FullServerFilter;
■■■ref OptionSelector■■■■m_ThirdPersonFilter;
■■■ref OptionSelector■■■■m_PublicFilter;
■■■ref OptionSelector■■■■m_AcceleratedTimeFilter;
■protected ServerBrowserTab■■■■m_Tab;
■
■void ServerBrowserFilterContainer( Widget root, ServerBrowserTab parent )
■{
■■string player_name;
■■GetGame().GetPlayerName( player_name );
■■m_Tab = parent;
■■
■■ref array<string> character_name_options ={ "#server_browser_disabled", player_name };
■■ref array<string> region_options = { "#server_browser_all", "#server_browser_americas", "#server_brow
■■ref array<string> sort_options = { "#server_browser_column_host A-Z", "#server_browser_column_hos
■■ref array<string> ping_options = { "#server_browser_disabled", "<30", "<50", "<100", "<200", "<300", "<
■■ref array<string> two_options = { "#server_browser_disabled", "#server_browser_show" };
■■
■■m_SearchByName■■■■= EditBoxWidget.Cast( root.FindAnyWidget( "search_name_setting_option" )
■■m_SearchByIP■■■■= EditBoxWidget.Cast( root.FindAnyWidget( "search_ip_setting_option" ) );
■■
■■m_RegionFilter■■■■= new OptionSelectorMultistate( root.FindAnyWidget( "region_setting_option" ), 0
■■m_PingFilter■■■■= new OptionSelectorMultistate( root.FindAnyWidget( "ping_setting_option" ), 0, thi
■■#ifdef PLATFORM_CONSOLE
■■m_FavoritedFilter■■■= new OptionSelectorMultistate( root.FindAnyWidget( "favorites_setting_option"
■■#else
■■m_FavoritedFilter■■■= new OptionSelector( root.FindAnyWidget( "favorites_setting_option" ), 0, this,
■■#endif
■■
■■m_DLCFilter■■■■■= new OptionSelector( root.FindAnyWidget( "dlc_setting_option" ), 0, this, false );
■■m_FriendsPlayingFilter■■= new OptionSelector( root.FindAnyWidget( "friends_setting_option" ), 0, this
■■m_PreviouslyPlayedFilter■= new OptionSelector( root.FindAnyWidget( "prev_played_setting_option" ),
■■m_FullServerFilter■■■= new OptionSelector( root.FindAnyWidget( "full_server_setting_option" ), 0, th
■■m_PasswordFilter■■■= new OptionSelector( root.FindAnyWidget( "password_setting_option" ), 0, this
■■m_WhitelistFilter■■■= new OptionSelector( root.FindAnyWidget( "whitelist_setting_option" ), 0, this, fa
■■
■■m_DLCFilter.m_OptionChanged.Insert( OnFilterChanged );
■■m_RegionFilter.m_OptionChanged.Insert( OnFilterChanged );
■■m_PingFilter.m_OptionChanged.Insert( OnFilterChanged );
```

```
// ---------------------------------------------
// ------ ServerBrowserMenuNew.c - START --------------------
// ---------------------------------------------


const int MAX_FAVORITES = 25;

#ifdef PLATFORM_CONSOLE
const int SERVER_BROWSER_PAGE_SIZE = 22;
#else
const int SERVER_BROWSER_PAGE_SIZE = 5;
#endif

class ServerBrowserMenuNew extends UIScriptedMenu
{
■protected Widget■■■■m_Play;
■protected Widget■■■■m_Back;
■protected Widget■■■■m_CustomizeCharacter;
■protected TextWidget■■■m_PlayerName;
■protected TextWidget■■■m_Version;
■
■protected TabberUI■■■■m_Tabber;
■protected ref ServerBrowserTab■m_OfficialTab;
■protected ref ServerBrowserTab■m_CommunityTab;
■protected ref ServerBrowserTab  m_FavoritesTab;
■protected ref ServerBrowserTab■m_LANTab;
■
■protected TabType■■■■m_IsRefreshing = TabType.NONE;
■protected ref TStringArray■■m_Favorites;
■protected ServerBrowserEntry■m_SelectedServer;
■
■override Widget Init()
■{
■■#ifdef PLATFORM_CONSOLE
■■layoutRoot ■■= GetGame().GetWorkspace().CreateWidgets("gui/layouts/new_ui/server_browser/xbox
■■m_FavoritesTab  = new ServerBrowserFavoritesTabConsolePages(layoutRoot.FindAnyWidget("Tab_0
■■m_OfficialTab■= new ServerBrowserTabConsolePages(layoutRoot.FindAnyWidget("Tab_1"), this, Tab
■■m_CommunityTab■= new ServerBrowserTabConsolePages(layoutRoot.FindAnyWidget("Tab_2"), this
■■#else
■■layoutRoot ■■= GetGame().GetWorkspace().CreateWidgets("gui/layouts/new_ui/server_browser/pc/se
■■m_FavoritesTab  = new ServerBrowserFavoritesTabPc(layoutRoot.FindAnyWidget("Tab_0"), this, TabT
■■m_OfficialTab■= new ServerBrowserTabPc(layoutRoot.FindAnyWidget("Tab_1"), this, TabType.OFFIC
■■m_CommunityTab■= new ServerBrowserTabPc(layoutRoot.FindAnyWidget("Tab_2"), this, TabType.C
■■m_LANTab■■= new ServerBrowserTabPc(layoutRoot.FindAnyWidget("Tab_3"), this, TabType.LAN);
■■#endif
■■
■■layoutRoot.FindAnyWidget("Tabber").GetScript(m_Tabber);
■■
■■m_Play■■■■■= layoutRoot.FindAnyWidget("play");
■■m_Back■■■■■= layoutRoot.FindAnyWidget("back_button");
■■m_CustomizeCharacter■= layoutRoot.FindAnyWidget("customize_character");
■■m_PlayerName■■■= TextWidget.Cast(layoutRoot.FindAnyWidget("character_name_text"));
■■m_Version■■■■= TextWidget.Cast(layoutRoot.FindAnyWidget("version"));
■■m_Favorites ■■■= new TStringArray;
■■
■■#ifndef PLATFORM_CONSOLE
■■layoutRoot.FindAnyWidget("customize_character").Show(false);
■■layoutRoot.FindAnyWidget("character").Show(false);
■■#endif
■■
■■Refresh();
■■
■■string version;
```

```
// ---------------------------------------------
// ------ ServerBrowserTab.c - START --------------------
// ---------------------------------------------


enum TabType
{
■OFFICIAL,
■COMMUNITY,
■LAN,
■FAVORITE,
■NONE
}

enum SelectedPanel
{
■BROWSER,
■FILTERS,
■MENU
}

class ServerBrowserTab extends ScriptedWidgetEventHandler
{
■protected Widget■■■■■■■■■■■m_Root;
■protected ScrollWidget■■■■■■■■■■m_ServerListScroller;
■protected SpacerBaseWidget■■■■■■■■■m_ServerList;
■
■//protected ref array<ref GetServersResultRow>■■■m_Entries;
■
■protected ref map<string, ref ServerBrowserEntry>■■m_EntryWidgets;
■protected ref map<string, ref array<string>>■■■m_EntryMods;

■protected ref ServerBrowserFilterContainer■■■■m_Filters;
■
■protected ServerBrowserMenuNew■■■■■■■■m_Menu;
■protected ServerBrowserEntry■■■■■■■■m_SelectedServer;
■
■protected TabType■■■■■■■■■■■m_TabType;
■protected ESortType■■■■■■■■■■■m_SortType;
■protected ESortOrder■■■■■■■■■■m_SortOrder;
■
■protected SelectedPanel■■■■■■■■■■m_SelectedPanel;
■protected bool■■■■■■■■■■■■m_Initialized;
■protected bool■■■■■■■■■■■■m_BegunLoading;
■protected bool■■■■■■■■■■■■m_Loading;
■protected int■■■■■■■■■■■■m_TotalServers; // UNUSED
■protected int■■■■■■■■■■■■m_TotalLoadedServers;
■protected int■■■■■■■■■■■■m_LastLoadedPage;
■protected int■■■■■■■■■■■■m_TotalPages;
■protected bool■■■■■■■■■■■■m_LoadingFinished;
■protected int■■■■■■■■■■■■m_CurrentPageNum;
■
■protected string■■■■■■■■■■■m_CurrentSelectedServer;
■protected int■■■■■■■■■■■■m_CurrentLoadedPage;
■protected ref GetServersInput■■■■■■■■m_CurrentFilterInput;
■
■protected Widget■■■■■■■■■■■m_ApplyFilter;
■protected Widget■■■■■■■■■■■m_RefreshList;
■protected Widget■■■■■■■■■■■m_ResetFilters;
■protected Widget■■■■■■■■■■■m_FiltersChanged;
■protected Widget■■■■■■■■■■■m_HostSort;
■protected Widget■■■■■■■■■■■m_TimeSort;
■protected Widget■■■■■■■■■■■m_PopulationSort;
```

```
// ---------------------------------------------
// ------ ServerBrowserTabConsole.c - START --------------------
// ---------------------------------------------


class ServerBrowserTabConsole extends ServerBrowserTab
{
■private bool m_IsFilterChanged;
■private bool m_IsFilterFocused;
■
■private Widget m_WidgetNavFilters;
■private Widget m_WidgetNavServers;
■
■protected override void Construct( Widget parent, ServerBrowserMenuNew menu, TabType type )
■{
■■m_Root = GetGame().GetWorkspace().CreateWidgets( "gui/layouts/new_ui/server_browser/xbox/serve
■■
■■m_ServerListScroller■= ScrollWidget.Cast( m_Root.FindAnyWidget( "server_list_scroller" ) );
■■m_ServerList■■■= SpacerBaseWidget.Cast( m_ServerListScroller.FindAnyWidget( "server_list_conte
■■m_ServerListScroller.VScrollToPos01( 0 );
■■
■■m_EntryWidgets■■■= new map<string, ref ServerBrowserEntry>;
■■m_EntriesSorted■■■= new map<ESortType, ref array<ref GetServersResultRow>>;
■■
■■m_EntriesSorted[ESortType.HOST] = new array<ref GetServersResultRow>;
■■m_EntriesSorted[ESortType.POPULATION] = new array<ref GetServersResultRow>;
■■
■■m_Menu■■■■■= menu;
■■m_TabType■■■■= type;
■■
■■m_ApplyFilter■■■= m_Root.FindAnyWidget( "apply_filter_button" );
■■m_RefreshList■■■= m_Root.FindAnyWidget( "refresh_list_button" );
■■m_FiltersChanged■■= m_Root.FindAnyWidget( "unapplied_filters_notify" );
■■m_HostSort■■■■= m_Root.FindAnyWidget( "server_list_content_header_host" );
■■m_TimeSort■■■■= m_Root.FindAnyWidget( "server_list_content_header_time" );
■■m_PopulationSort■■= m_Root.FindAnyWidget( "server_list_content_header_population" );
■■m_SlotsSort■■■■= m_Root.FindAnyWidget( "server_list_content_header_slots" );
■■m_PingSort■■■■= m_Root.FindAnyWidget( "server_list_content_header_ping" );
■■m_LoadingText■■■= TextWidget.Cast( m_Root.FindAnyWidget( "loading_servers_info" ) );
■■m_WidgetNavFilters■■= m_Root.FindAnyWidget( "filters_root_nav_wrapper" );
■■m_WidgetNavServers■■= m_Root.FindAnyWidget( "server_list_root_nav_wrapper" );
■■
■■ShowHideConsoleWidgets();■■
■■
■■m_Filters■■■■= new ServerBrowserFilterContainer( m_Root.FindAnyWidget( "filters_content" ), this )
■■
■■SetSort( ESortType.HOST, ESortOrder.ASCENDING );
■■SetFocusFilters();
■■
■■m_Root.SetHandler( this );
■}
■
■void ShowHideConsoleWidgets()
■{■■
#ifdef PLATFORM_PS4
■■bool is_xbox = false;
#else
■■bool is_xbox = true;
#endif
■■
■■m_Root.FindAnyWidget( "filters_root_nav_img_lb_xbox" ).Show( is_xbox );
■■m_Root.FindAnyWidget( "filters_root_nav_img_rb_xbox" ).Show( is_xbox );
■■m_Root.FindAnyWidget( "server_list_root_nav_img_lb_xbox" ).Show( is_xbox );
```

```
// ---------------------------------------------
// ------ ServerBrowserTabConsolePages.c - START --------------------
// ---------------------------------------------


class ServerBrowserTabConsolePages extends ServerBrowserTab
{
■private bool m_IsFilterChanged;
■private bool m_IsFilterFocused;
■protected bool m_MouseKeyboardControlled
■
■private Widget m_WidgetNavFilters;
■private Widget m_WidgetNavServers;
■
■protected Widget■■■■■■■■m_ButtonPageLeftImg;
■protected Widget■■■■■■■■m_ButtonPageRightImg;
■
■protected int■■■■■■■■■m_PreviousPage;
■protected int■■■■■■■■■m_TotalServersCount;
■protected int■■■■■■■■■m_PageStartNum;
■protected int■■■■■■■■■m_PageEndNum;
■protected int■■■■■■■■■m_PagesCount;
■protected int■■■■■■■■■m_ServersEstimateCount;
■protected int■■■■■■■■■m_TimeLastServerRefresh;
■protected int■■■■■■■■■m_TimeLastServerRefreshHoldButton;
■
■protected Widget■■■■■■■■m_PnlPagesPanel;
■protected TextWidget■■■■■■■m_PnlPagesLoadingText;
■protected ref array<ref ServerBrowserEntry>■■m_ServerListEntiers;
■
■protected override void Construct( Widget parent, ServerBrowserMenuNew menu, TabType type )
■{
■■m_Root = GetGame().GetWorkspace().CreateWidgets( "gui/layouts/new_ui/server_browser/xbox/serve
■■
■■m_ServerListScroller■= ScrollWidget.Cast( m_Root.FindAnyWidget( "server_list_scroller" ) );
■■m_ServerList■■■= SpacerBaseWidget.Cast( m_ServerListScroller.FindAnyWidget( "server_list_conte
■■m_ServerListScroller.VScrollToPos01( 0 );
■■
■■m_ServerListEntiers■■= new array<ref ServerBrowserEntry>;
■■m_EntryWidgets■■■= new map<string, ref ServerBrowserEntry>;
■■m_EntriesSorted■■■= new map<ESortType, ref array<ref GetServersResultRow>>;
■■
■■m_EntriesSorted[ESortType.HOST] = new array<ref GetServersResultRow>;
■■m_EntriesSorted[ESortType.POPULATION] = new array<ref GetServersResultRow>;
■■
■■m_Menu■■■■■= menu;
■■m_TabType■■■■= type;
■■
■■m_ApplyFilter■■■= m_Root.FindAnyWidget( "apply_filter_button" );
■■m_ResetFilters■■■= m_Root.FindAnyWidget( "reset_filter_button" );
■■m_RefreshList■■■= m_Root.FindAnyWidget( "refresh_list_button" );
■■m_FiltersChanged■■= m_Root.FindAnyWidget( "unapplied_filters_notify" );
■■m_HostSort■■■■= m_Root.FindAnyWidget( "server_list_content_header_host" );
■■m_TimeSort■■■■= m_Root.FindAnyWidget( "server_list_content_header_time" );
■■m_PopulationSort■■= m_Root.FindAnyWidget( "server_list_content_header_population" );
■■m_SlotsSort■■■■= m_Root.FindAnyWidget( "server_list_content_header_slots" );
■■m_PingSort■■■■= m_Root.FindAnyWidget( "server_list_content_header_ping" );
■■m_FilterSearchText■■= m_Root.FindAnyWidget( "search_name_button" );
■■m_FilterSearchTextBox■= m_Root.FindAnyWidget( "search_name_setting_option" );
■■m_LoadingText■■■= TextWidget.Cast( m_Root.FindAnyWidget( "loading_servers_info" ) );
■■m_WidgetNavFilters■■= m_Root.FindAnyWidget( "filters_root_nav_wrapper" );
■■m_WidgetNavServers■■= m_Root.FindAnyWidget( "server_list_root_nav_wrapper" );
■■
```

```
// ---------------------------------------------
// ------ ServerBrowserTabPc.c - START --------------------
// ---------------------------------------------


class ServerBrowserTabPc extends ServerBrowserTab
{
■protected const int SERVERS_VISIBLE_COUNT■= 24;
■protected const int PAGES_BUTTONS_COUNT■■= 10;
■
■protected int■■■■■■■■■■m_TotalServersCount; //UNUSED
■protected int■■■■■■■■■■m_PageIndex;
■protected int■■■■■■■■■■m_PageStartNum;
■protected int■■■■■■■■■■m_PageEndNum;
■protected int■■■■■■■■■■m_PagesCount;
■protected int■■■■■■■■■■m_ServersEstimateCount;
■■
■protected Widget■■■■■■■■■m_PnlPagesPanel;
■protected TextWidget■■■■■■■■m_PnlPagesLoadingText;
■protected ButtonWidget■■■■■■■■m_BtnPagesFirst;
■protected ButtonWidget■■■■■■■■m_BtnPagesLast;
■
■protected Widget■■■■■■■■■m_FilterSearchIP;
■protected Widget■■■■■■■■■m_FilterSearchIPBox;
■protected Widget■■■■■■■■■m_FilterPanelPing;
■protected Widget■■■■■■■■■m_FilterPanelAccTime;
■
■protected ref array<ButtonWidget>■■■■m_BtnPages;
■protected ref array<ref ServerBrowserEntry>■■m_ServerListEntries;
■
■protected ref TStringArray m_TempTime = new TStringArray;
■
■protected override void Construct( Widget parent, ServerBrowserMenuNew menu, TabType type )
■{
■■#ifdef PLATFORM_CONSOLE
■■■m_Root■■■■■= GetGame().GetWorkspace().CreateWidgets( "gui/layouts/new_ui/server_browser/x
■■#else
■■#ifdef PLATFORM_WINDOWS
■■■m_Root■■■■■= GetGame().GetWorkspace().CreateWidgets( "gui/layouts/new_ui/server_browser/p
■■#endif
■■#endif
■■
■■m_ServerListScroller■= ScrollWidget.Cast( m_Root.FindAnyWidget( "server_list_scroller" ) );
■■m_ServerList■■■= SpacerBaseWidget.Cast( m_ServerListScroller.FindAnyWidget( "server_list_conte
■■m_ServerListScroller.VScrollToPos01( 0 );
■■■■
■■m_ServerListEntries■■= new array<ref ServerBrowserEntry>;■■
■■m_EntryWidgets■■■= new map<string, ref ServerBrowserEntry>;
■■m_SortInverted■■■= new map<ESortType, ESortOrder>;
■■m_EntriesSorted■■■= new map<ESortType, ref array<ref GetServersResultRow>>;
■■m_EntryMods■■■■= new map<string, ref array<string>>;
■■
■■m_EntriesSorted[ESortType.HOST] ■■= new array<ref GetServersResultRow>;
■■m_EntriesSorted[ESortType.TIME]■■■= new array<ref GetServersResultRow>;
■■m_EntriesSorted[ESortType.POPULATION]■= new array<ref GetServersResultRow>;
■■m_EntriesSorted[ESortType.SLOTS]■■= new array<ref GetServersResultRow>;
■■m_EntriesSorted[ESortType.PING]■■■= new array<ref GetServersResultRow>;
■■m_EntriesSorted[ESortType.FAVORITE]■■= new array<ref GetServersResultRow>;
■■m_EntriesSorted[ESortType.PASSWORDED]■= new array<ref GetServersResultRow>;
■■
■■m_Menu■■■■■= menu;
■■m_TabType■■■■= type;
■■
```

```
// ----------------------------------------------
// ------ settings.c - START --------------------
// ----------------------------------------------


#ifdef GAME_TEMPLATE

class Settings
{
■static void■■OnChange(string variableName) {}
■static void ■OnAnyChange() {}■
■static void ■OnLoad() {}
■static void ■OnSave() {}
■static void ■OnReset() {}
■static void ■OnRevert() {}
■static void ■OnApply() {}
■
■private void Settings() {}
■private void ~Settings() {}
};

class GameSettings: Settings
{
■[Attribute("false", "checkbox", "Is debug mode enabled")]
■static bool Debug;
■
■override static void OnAnyChange()
■{
■■GetGame().SetDebug(Debug);
■}
}

class SettingsMenu: MenuBase
{
■proto native external bool AddSettings(typename settingsClass);■
■proto native void Save();■
■proto native void Reset();
■proto native void Revert();
■proto native void Apply();
■proto native void Back();
};
#endif





// ----------------------------------------------
// ------ settings.c - END ----------------------
// ----------------------------------------------


// ----------------------------------------------
// ------ SewingKit.c - START --------------------
// ----------------------------------------------


class SewingKit: Inventory_Base
{
■override void SetActions()
■{
■■super.SetActions();
■■
■■AddAction(ActionRepairTent);
```

```
// ---------------------------------------------
// ------ SharpenBroom.c - START --------------------
// ---------------------------------------------


class SharpenBroom extends RecipeBase
{■
■override void Init()
■{
■■m_Name = "#sharpen";
■■m_IsInstaRecipe = false;//should this recipe be performed instantly without animation
■■m_AnimationLength = 1;//animation length in relative time units
■■m_Specialty = 0.01;// value > 0 for roughness, value < 0 for precision

■■//conditions
■■m_MinDamageIngredient[0] = -1;
■■m_MaxDamageIngredient[0] = 3;
■■m_MinQuantityIngredient[0] = 1;
■■m_MaxQuantityIngredient[0] = -1;
■■
■■m_MinDamageIngredient[1] = -1;
■■m_MaxDamageIngredient[1] = 3;
■■m_MinQuantityIngredient[1] = -1;
■■m_MaxQuantityIngredient[1] = -1;
■■
■■
■■//ingredient 1
■■InsertIngredient(0,"Broom");//you can insert multiple ingredients this way

■■m_IngredientAddHealth[0] = 0;
■■m_IngredientSetHealth[0] = -1; // -1 = do nothing
■■m_IngredientAddQuantity[0] = -1;
■■m_IngredientDestroy[0] = 1;
■■m_IngredientUseSoftSkills[1] = false;// set 'true' to allow modification of the values by softskills on this i
■■
■■//ingredient 2
■■InsertIngredient(1,"Sickle");//you can insert multiple ingredients this way
■■InsertIngredient(1,"KukriKnife");
■■InsertIngredient(1,"FangeKnife");
■■InsertIngredient(1,"Hacksaw");
■■InsertIngredient(1,"KitchenKnife");
■■InsertIngredient(1,"SteakKnife");
■■InsertIngredient(1,"HayHook");
■■InsertIngredient(1,"StoneKnife");
■■InsertIngredient(1,"Cleaver");
■■InsertIngredient(1,"CombatKnife");
■■InsertIngredient(1,"HuntingKnife");
■■InsertIngredient(1,"Machete");
■■InsertIngredient(1,"CrudeMachete");
■■InsertIngredient(1,"OrientalMachete");
■■InsertIngredient(1,"Screwdriver");
■■InsertIngredient(1,"Crowbar");
■■InsertIngredient(1,"Pickaxe");
■■InsertIngredient(1,"WoodAxe");
■■InsertIngredient(1,"Hatchet");
■■InsertIngredient(1,"FirefighterAxe");
■■InsertIngredient(1,"Sword");
■■InsertIngredient(1,"AK_Bayonet");
■■InsertIngredient(1,"M9A1_Bayonet");
■■InsertIngredient(1,"Mosin_Bayonet");
■■InsertIngredient(1,"SKS_Bayonet");
■■InsertIngredient(1,"BoneKnife");
■■
```

```
// ---------------------------------------------
// ------ SharpenLongStick.c - START --------------------
// ---------------------------------------------


class SharpenLongStick extends RecipeBase
{■
■override void Init()
■{
■■m_Name = "#sharpen";
■■m_IsInstaRecipe = false;//should this recipe be performed instantly without animation
■■m_AnimationLength = 1;//animation length in relative time units
■■m_Specialty = 0.01;// value > 0 for roughness, value < 0 for precision

■■//conditions
■■m_MinDamageIngredient[0] = -1;
■■m_MaxDamageIngredient[0] = 3;
■■m_MinQuantityIngredient[0] = 1;
■■m_MaxQuantityIngredient[0] = -1;
■■
■■m_MinDamageIngredient[1] = -1;
■■m_MaxDamageIngredient[1] = 3;
■■m_MinQuantityIngredient[1] = -1;
■■m_MaxQuantityIngredient[1] = -1;
■■
■■
■■//ingredient 1
■■InsertIngredient(0,"LongWoodenStick");//you can insert multiple ingredients this way

■■m_IngredientAddHealth[0] = 0;
■■m_IngredientSetHealth[0] = -1; // -1 = do nothing
■■m_IngredientAddQuantity[0] = -1;
■■m_IngredientDestroy[0] = 0;
■■m_IngredientUseSoftSkills[1] = false;// set 'true' to allow modification of the values by softskills on this i
■■
■■//ingredient 2
■■InsertIngredient(1,"Sickle");//you can insert multiple ingredients this way
■■InsertIngredient(1,"KukriKnife");
■■InsertIngredient(1,"FangeKnife");
■■InsertIngredient(1,"Hacksaw");
■■InsertIngredient(1,"KitchenKnife");
■■InsertIngredient(1,"SteakKnife");
■■InsertIngredient(1,"HayHook");
■■InsertIngredient(1,"StoneKnife");
■■InsertIngredient(1,"Cleaver");
■■InsertIngredient(1,"CombatKnife");
■■InsertIngredient(1,"HuntingKnife");
■■InsertIngredient(1,"Machete");
■■InsertIngredient(1,"CrudeMachete");
■■InsertIngredient(1,"OrientalMachete");
■■InsertIngredient(1,"Screwdriver");
■■InsertIngredient(1,"Crowbar");
■■InsertIngredient(1,"Pickaxe");
■■InsertIngredient(1,"WoodAxe");
■■InsertIngredient(1,"Hatchet");
■■InsertIngredient(1,"FirefighterAxe");
■■InsertIngredient(1,"Sword");
■■InsertIngredient(1,"AK_Bayonet");
■■InsertIngredient(1,"M9A1_Bayonet");
■■InsertIngredient(1,"Mosin_Bayonet");
■■InsertIngredient(1,"SKS_Bayonet");
■■InsertIngredient(1,"BoneKnife");
■■
```

```
// ---------------------------------------------
// ------ SharpenMelee.c - START --------------------
// ---------------------------------------------


class SharpenMelee extends RecipeBase
{■
■override void Init()
■{
■■m_Name = "#sharpen";
■■m_IsInstaRecipe = false;//should this recipe be performed instantly without animation
■■m_AnimationLength = 1;//animation length in relative time units
■■m_Specialty = -0.02;// value > 0 for roughness, value < 0 for precision
■■
■■
■■//conditions
■■m_MinDamageIngredient[0] = -1;//-1 = disable check
■■m_MaxDamageIngredient[0] = 3;//-1 = disable check
■■
■■m_MinQuantityIngredient[0] = -1;//-1 = disable check
■■m_MaxQuantityIngredient[0] = -1;//-1 = disable check
■■
■■m_MinDamageIngredient[1] = 1;//-1 = disable check
■■m_MaxDamageIngredient[1] = 3;//-1 = disable check
■■
■■m_MinQuantityIngredient[1] = -1;//-1 = disable check
■■m_MaxQuantityIngredient[1] = -1;//-1 = disable check
■■//----------------------------------------------------------------------------------------------------------------------
■■
■■//INGREDIENTS
■■//ingredient 1
■■InsertIngredient(0,"Whetstone");//you can insert multiple ingredients this way
■■
■■m_IngredientAddHealth[0] = 0;// 0 = do nothing
■■m_IngredientSetHealth[0] = -1; // -1 = do nothing
■■m_IngredientAddQuantity[0] = 0;// 0 = do nothing
■■m_IngredientDestroy[0] = false;//true = destroy, false = do nothing
■■m_IngredientUseSoftSkills[0] = true;// set 'true' to allow modification of the values by softskills on this in
■■
■■//ingredient 2
■■InsertIngredient(1,"Inventory_Base");//you can insert multiple ingredients this way
■■
■■m_IngredientAddHealth[1] = 0;// 0 = do nothing
■■m_IngredientSetHealth[1] = -1; // -1 = do nothing
■■m_IngredientAddQuantity[1] = 0;// 0 = do nothing
■■m_IngredientDestroy[1] = false;// false = do nothing
■■m_IngredientUseSoftSkills[1] = false;// set 'true' to allow modification of the values by softskills on this i
■■
■■//----------------------------------------------------------------------------------------------------------------------
■■
■■//result1
■■//AddResult("");//add results here

■■m_ResultSetFullQuantity[0] = false;//true = set full quantity, false = do nothing
■■m_ResultSetQuantity[0] = -1;//-1 = do nothing
■■m_ResultSetHealth[0] = -1;//-1 = do nothing
■■m_ResultInheritsHealth[0] = -1;// (value) == -1 means do nothing; a (value) >= 0 means this result will i
■■m_ResultInheritsColor[0] = -1;// (value) == -1 means do nothing; a (value) >= 0 means this result classn
■■m_ResultToInventory[0] = -2;//(value) == -2 spawn result on the ground;(value) == -1 place anywhere i
■■m_ResultUseSoftSkills[0] = false;// set 'true' to allow modification of the values by softskills on this resu
■■m_ResultReplacesIngredient[0] = -1;// value == -1 means do nothing; a value >= 0 means this result wi

■■//----------------------------------------------------------------------------------------------------------------------
```

```
// ---------------------------------------------
// ------ SharpenStick.c - START --------------------
// ---------------------------------------------


class SharpenStick extends RecipeBase
{■
■override void Init()
■{
■■m_Name = "#sharpen";
■■m_IsInstaRecipe = false;//should this recipe be performed instantly without animation
■■m_AnimationLength = 1;//animation length in relative time units
■■m_Specialty = 0.01;// value > 0 for roughness, value < 0 for precision

■■//conditions
■■m_MinDamageIngredient[0] = -1;
■■m_MaxDamageIngredient[0] = 3;
■■m_MinQuantityIngredient[0] = 1;
■■m_MaxQuantityIngredient[0] = -1;
■■
■■m_MinDamageIngredient[1] = -1;
■■m_MaxDamageIngredient[1] = 3;
■■m_MinQuantityIngredient[1] = -1;
■■m_MaxQuantityIngredient[1] = -1;
■■
■■
■■//ingredient 1
■■InsertIngredient(0,"WoodenStick");//you can insert multiple ingredients this way

■■m_IngredientAddHealth[0] = 0;
■■m_IngredientSetHealth[0] = -1; // -1 = do nothing
■■m_IngredientAddQuantity[0] = -1;
■■m_IngredientDestroy[0] = 0;
■■m_IngredientUseSoftSkills[1] = false;// set 'true' to allow modification of the values by softskills on this i
■■
■■//ingredient 2
■■InsertIngredient(1,"Sickle");//you can insert multiple ingredients this way■■
■■InsertIngredient(1,"KukriKnife");
■■InsertIngredient(1,"FangeKnife");
■■InsertIngredient(1,"Hacksaw");
■■InsertIngredient(1,"KitchenKnife");
■■InsertIngredient(1,"SteakKnife");
■■InsertIngredient(1,"HayHook");
■■InsertIngredient(1,"StoneKnife");
■■InsertIngredient(1,"Cleaver");
■■InsertIngredient(1,"CombatKnife");
■■InsertIngredient(1,"HuntingKnife");
■■InsertIngredient(1,"Machete");
■■InsertIngredient(1,"CrudeMachete");
■■InsertIngredient(1,"OrientalMachete");
■■InsertIngredient(1,"Screwdriver");
■■InsertIngredient(1,"Crowbar");
■■InsertIngredient(1,"Pickaxe");
■■InsertIngredient(1,"WoodAxe");
■■InsertIngredient(1,"Hatchet");
■■InsertIngredient(1,"FirefighterAxe");
■■InsertIngredient(1,"Sword");
■■InsertIngredient(1,"AK_Bayonet");
■■InsertIngredient(1,"M9A1_Bayonet");
■■InsertIngredient(1,"Mosin_Bayonet");
■■InsertIngredient(1,"SKS_Bayonet");
■■InsertIngredient(1,"BoneKnife");
■■
```

```
// ---------------------------------------------
// ------ SheepSteakMeat.c - START --------------------
// ---------------------------------------------


class SheepSteakMeat extends Edible_Base
{
■override bool CanBeCooked()
■{
■■return true;
■}■■
■
■override bool CanBeCookedOnStick()
■{
■■return true;
■}
■
■override bool IsMeat()
■{
■■return true;
■}

■override bool CanDecay()
■{
■■return true;
■}
■
■override void SetActions()
■{
■■super.SetActions();
■■
■■AddAction(ActionForceFeed);
■■AddAction(ActionEatMeat);
■■
■■AddAction(ActionCreateIndoorFireplace);
■■AddAction(ActionCreateIndoorOven);
■}
}


// ---------------------------------------------
// ------ SheepSteakMeat.c - END ----------------------
// ---------------------------------------------



// ---------------------------------------------
// ------ Shelter.c - START --------------------
// ---------------------------------------------


class ShelterBase extends TentBase
{
■static const string SITE_OBJECT_TYPE = "ShelterSite";

■void ShelterBase()
■{
■■m_ShowAnimationsWhenPitched.Insert("Body");
■■m_ShowAnimationsWhenPacked.Insert("Body");
■■
■■m_HalfExtents = Vector(0.8, 0.15, 1.3);
■}

■override void SetActions()
```

```
// ---------------------------------------------
// ------ ShelterKit.c - START --------------------
// ---------------------------------------------


class ShelterKit extends KitBase
{
■override bool CanReceiveAttachment(EntityAI attachment, int slotId)
■{
■■if ( !super.CanReceiveAttachment(attachment, slotId) )
■■■return false;
■■
■■ItemBase att = ItemBase.Cast(GetInventory().FindAttachment(slotId));
■■if (att)
■■■return false;
■■
■■return true;
■}

■//==================================================================
■// ADVANCED PLACEMENT
■//==================================================================
■override void OnPlacementComplete( Man player, vector position = "0 0 0", vector orientation = "0 0 0" )
■{
■■super.OnPlacementComplete( player, position, orientation );
■■
■■if ( GetGame().IsServer() )
■■{
■■■//Create shelter site
■■■ShelterSite site = ShelterSite.Cast( GetGame().CreateObjectEx( "ShelterSite", GetPosition(), ECE_P
■■■site.SetPosition( position );
■■■site.SetOrientation( orientation );
■■■
■■■//make the kit invisible, so it can be destroyed from deploy UA when action ends
■■■HideAllSelections();
■■■
■■■SetIsDeploySound( true );
■■}
■}

■override bool DoPlacingHeightCheck()
■{
■■return true;
■}

■override float HeightCheckOverride()
■{
■■return 1.6;
■}
■
■override string GetDeploySoundset()
■{
■■return "Shelter_Site_Build_Start_SoundSet";
■}
■
■override string GetLoopDeploySoundset()
■{
■■return "Shelter_Site_Build_Loop_SoundSet";
■}
■
■override string GetDeployFinishSoundset()
■{
■■return "Shelter_Site_Build_Finish_SoundSet";
```

```
// ---------------------------------------------
// ------ ShelterSite.c - START --------------------
// ---------------------------------------------


class ShelterSite extends BaseBuildingBase
{
■const float MAX_ACTION_DETECTION_ANGLE_RAD ■■= 1.3;■■//1.3 RAD = ~75 DEG
■const float MAX_ACTION_DETECTION_DISTANCE ■■= 2.0;■■//meters

■void ShelterSite()
■{
■}

■override string GetConstructionKitType()
■{
■■return "ShelterKit";
■}

■override int GetMeleeTargetType()
■{
■■return EMeleeTargetType.NONALIGNABLE;
■}

■//--- CONSTRUCTION KIT
■override vector GetKitSpawnPosition()
■{
■■if ( MemoryPointExists( "kit_spawn_position" ) )
■■{
■■■vector position;
■■■position = GetMemoryPointPos( "kit_spawn_position" );
■■■
■■■return ModelToWorld( position );
■■}
■■
■■return GetPosition();
■}

■//--- BUILD EVENTS
■//CONSTRUCTION EVENTS
■override void OnPartBuiltServer( notnull Man player, string part_name, int action_id )
■{
■■ConstructionPart constrution_part = GetConstruction().GetConstructionPart( part_name );
■■
■■string shelter_type = "";
■■switch (part_name)
■■{
■■■case "leather":
■■■■shelter_type = "ShelterLeather";
■■■break;
■■■
■■■case "fabric":
■■■■shelter_type = "ShelterFabric";
■■■break;
■■■
■■■case "stick":
■■■■shelter_type = "ShelterStick";
■■■break;
■■■■
■■■default: {};
■■}
■■
■■if (shelter_type != "")
```

```
// ---------------------------------------------
// ------ Shirt_ColorBase.c - START --------------------
// ---------------------------------------------


class Shirt_ColorBase extends Clothing
{
■override void SetActions()
■{
■■super.SetActions();
■■AddAction(ActionWringClothes);
■}
};
class Shirt_BlueCheck extends Shirt_ColorBase {};
class Shirt_BlueCheckBright extends Shirt_ColorBase {};
class Shirt_GreenCheck extends Shirt_ColorBase {};
class Shirt_RedCheck extends Shirt_ColorBase {};
class Shirt_WhiteCheck extends Shirt_ColorBase {};
class Shirt_PlaneBlack extends Shirt_ColorBase {};


// ---------------------------------------------
// ------ Shirt_ColorBase.c - END ----------------------
// ---------------------------------------------


// ---------------------------------------------
// ------ Shock.c - START --------------------
// ---------------------------------------------


class ShockMdfr: ModifierBase
{
■private const float■ ■UNCONSCIOUS_LIMIT = 50;
■private const float ■SHOCK_INCREMENT_PER_SEC = 1;
■
■override void Init()
■{
■■m_ID ■■■■■= eModifiers.MDF_SHOCK;
■■m_TickIntervalInactive ■= 1;
■■m_TickIntervalActive ■= 0.35;
■■//DisableActivateCheck();
■■DisableDeactivateCheck();
■}■

■override bool ActivateCondition(PlayerBase player)
■{
■■return true;
■}

■override bool DeactivateCondition(PlayerBase player)
■{
■■return false;
■}

■override void OnActivate(PlayerBase player)
■{
■}
■
■override void OnDeactivate(PlayerBase player)
■{
■}
■
```

```
// ---------------------------------------------
// ------ ShockDamage.c - START --------------------
// ---------------------------------------------


class ShockDamageMdfr: ModifierBase
{
■const int UNCONSIOUSS_COOLDOWN_TIME = 60;//in s
■override void Init()
■{
■■m_TrackActivatedTime■= false;
■■m_ID ■■■■■= eModifiers.MDF_SHOCK_DAMAGE;
■■m_TickIntervalInactive ■= DEFAULT_TICK_TIME_INACTIVE;
■■m_TickIntervalActive ■= 0.5;
■}

■override void OnActivate(PlayerBase player)
■{
■■
■}

■override void OnReconnect(PlayerBase player)
■{

■}

■override bool ActivateCondition(PlayerBase player)
■{
■■if( !player.IsUnconscious() && player.GetHealth("","Blood") <= PlayerConstants.SHOCK_DAMAGE_BL
■■{
■■■return true;
■■}
■■else
■■{
■■■return false;
■■}
■}

■override bool DeactivateCondition(PlayerBase player)
■{
■■return !ActivateCondition(player);
■}
■// ----------------------------------------------------------------------------

■override void OnTick(PlayerBase player, float deltaT)
■{
■■float blood =  player.GetHealth("","Blood");
■■float value = Math.InverseLerp( PlayerConstants.SHOCK_DAMAGE_BLOOD_THRESHOLD_HIGH, Pl
■■value = Math.Clamp(value,0,1);
■■float dmg = Math.Lerp( PlayerConstants.SHOCK_DAMAGE_HIGH, PlayerConstants.SHOCK_DAMAG
■■float damage =  -dmg * deltaT;
■■player.AddHealth("","Shock", damage);
■■//PrintString(damage.ToString());
■}
};


// ---------------------------------------------
// ------ ShockDamage.c - END ----------------------
// ---------------------------------------------
// ---------------------------------------------
// ------ ShockHandler.c - START --------------------
// ---------------------------------------------


class ShockHandler
{■
■protected float ■■■■■■m_Shock;
■protected float ■■■■■■m_ShockValueMax;
■protected float ■■■■■■m_ShockValueThreshold;
■protected PlayerBase■■■■■m_Player;
■
■protected const float ■■■■■UPDATE_THRESHOLD = 3; //NOTE : The lower, the more precise but the
■const float ■■■■■■■VALUE_CHECK_INTERVAL = 0.95; //in seconds
■protected float ■■■■■■m_CumulatedShock;
■private float■■■■■■■m_TimeSinceLastTick = VALUE_CHECK_INTERVAL + 1;
■private float■■■■■■■m_ShockMultiplier = 1;
■private float ■■■■■■■m_PrevVignette; //Previous vignette shock value
■private■float ■■■■■■■m_LerpRes; //Lerp result
■
■private const int■■■■■■LIGHT_SHOCK_HIT = 33;
■private const int■■■■■■MID_SHOCK_HIT = 67;
■private const int■■■■■■HEAVY_SHOCK_HIT = 100;
■private const int ■■■■■■INTENSITY_FACTOR = 1; //How intense the vignette effect will be, the higher
■
■//Pulsing effect
■private const float ■■■■■PULSE_PERIOD = 0.5; //The time it takes for pulse to do a full cycle
■private const float ■■■■■PULSE_AMPLITUDE = 0.05; //This is a multiplier, keep below 1 or expect the
■private ■  float ■■■■■m_PulseTimer;
■
■protected ref Param1<float> ■■■m_Param;
■
■void ShockHandler(PlayerBase player)
■{
■■m_Player = player;
■■m_Player.m_CurrentShock = m_Player.GetMaxHealth("", "Shock");
■■m_PrevVignette = m_Player.m_CurrentShock * 0.01; //Equivalent to divided by 100
■■m_ShockValueMax = m_Player.GetMaxHealth("", "Shock");
■■m_ShockValueThreshold = m_ShockValueMax * 0.95;
■■m_Param = new Param1<float>(0);
■}

■void Update(float deltaT)
■{
■■m_TimeSinceLastTick += deltaT;
■■
■■m_PulseTimer += deltaT;
■■
■■if ( GetGame().IsClient() )
■■{
■■■//Deactivate tunnel vignette when player falls unconscious
■■■if ( m_Player.IsUnconscious() )
■■■{
■■■■PPERequesterBank.GetRequester(PPERequester_TunnelVisionEffects).Stop();
■■■■return;
■■■}
■■■
■■■//Deactivate if above visible threshold (also stops "zero bobbing" being sent all the time
■■■if ( m_Player.m_CurrentShock >= m_ShockValueThreshold)
■■■{
■■■■PPERequesterBank.GetRequester(PPERequester_TunnelVisionEffects).Stop();
■■■■return;
■■■}
```

```
// ---------------------------------------------
// ------ ShockHitReaction.c - START --------------------
// ---------------------------------------------


class ShockDealtEffect
{
■const float INTENSITY_COEF_OVERALL = 0.8; //intensity percentage multiplier
■const float INTENSITY_COEF_BLUR = 1;
■const float INTENSITY_COEF_COLOR = 0.2;
■const float INTENSITY_COEF_VIGNETTE = 0.7;
■const float DURATION_MIN = 0.6;
■
■ref Param3<float,float,float> m_EffectParam;
■
■float m_HitDuration;
■float m_BreakPoint;
■float m_TimeActive;
■float m_ShockIntensityMax;
■
■void ShockDealtEffect(float intensity_max)
■{
■■float duration_coef = Math.Clamp(intensity_max,DURATION_MIN,1);
■■m_HitDuration = 1 * duration_coef;
■■m_BreakPoint = 0.2 * duration_coef;
■■m_ShockIntensityMax = Math.Clamp(intensity_max,0,1);
■■
■■m_EffectParam = new Param3<float,float,float>(0,0,0);
■■//Print(intensity_max);
■■//Print("HitSyncDebug | ShockDealtEffect: " + GetGame().GetPlayer().GetSimulationTimeStamp());
■}
■
■void ~ShockDealtEffect()
■{
■■if (GetGame())
■■■PPERequesterBank.GetRequester(PPERequester_ShockHitReaction).Stop();
■}
■
■void Update(float deltatime)
■{
■■float value;
■■
■■if ( m_TimeActive <= m_BreakPoint )
■■{
■■■value = Math.InverseLerp(0, m_BreakPoint, m_TimeActive);
■■}
■■else
■■{
■■■float tmp_value = Math.InverseLerp(m_BreakPoint, m_HitDuration, m_TimeActive);
■■■value = 1 - tmp_value;
■■}
■■
■■m_TimeActive += deltatime;
■■value = Math.Clamp(value,0,1);
■■
■■//value calculations
■■float val = Math.Clamp(Math.Lerp(0, m_ShockIntensityMax, value),0,m_ShockIntensityMax);
■■float val_color = Math.Clamp(val * INTENSITY_COEF_COLOR,0,m_ShockIntensityMax);
■■
■■//Postprocess application
■■float blur = val * INTENSITY_COEF_OVERALL * INTENSITY_COEF_BLUR;
■■float factor = val_color * INTENSITY_COEF_OVERALL;
■■float vignette = val * INTENSITY_COEF_OVERALL * INTENSITY_COEF_VIGNETTE;
```

```
// ---------------------------------------------
// ------ Shovel.c - START --------------------
// ---------------------------------------------


class Shovel extends ItemBase
{
■override bool CanMakeGardenplot()
■{
■■return true;
■}
■
■override void SetActions()
■{
■■super.SetActions();
■■
■■AddAction(ActionClapBearTrapWithThisItem);
■■AddAction(ActionTogglePlaceObject);
■■AddAction(ActionDigGardenPlot);
■■AddAction(ActionDismantleGardenPlot);
■■AddAction(ActionDismantlePart);
■■AddAction(ActionBuildPart);
■■AddAction(ActionBuryBody);
■■AddAction(ActionBuryAshes);
■■AddAction(ActionDigOutStash);
■■AddAction(ActionDigInStash);
■■AddAction(ActionFillObject);
■■AddAction(ActionDigWorms);
■■AddAction(ActionCreateGreenhouseGardenPlot);
■}
}


// ---------------------------------------------
// ------ Shovel.c - END ----------------------
// ---------------------------------------------


// ---------------------------------------------
// ------ ShowLog.c - START --------------------
// ---------------------------------------------


[WorkbenchPluginAttribute("SVN ShowLog", "Just for testing", "ctrl+=", "", {"ScriptEditor"})]
class SVNShowLogPlugin: WorkbenchPlugin
{
■[Attribute("TortoiseProc /command:log /path:$path", "editbox")]
■string CommandLine;
■
■override void Run()
■{
■■ScriptEditor mod = Workbench.GetModule("ScriptEditor");
■■if (mod)
■■{
■■■string file;
■■■string absPath;
■■■if (mod.GetCurrentFile(file) && Workbench.GetAbsolutePath(file, absPath))
■■■{
■■■■string command = CommandLine;
■■■■command.Replace("$path", absPath);
■■■■Workbench.RunCmd(command);
■■■■//Print( command );
■■■■//Print( absPath );
```

```
// ---------------------------------------------
// ------ SiagaRecoil.c - START --------------------
// ---------------------------------------------


class SiagaRecoil: RecoilBase
{
■override void Init()
■{
■■vector point_1;
■■vector point_2;
■■vector point_3;
■■vector point_4;
■■point_1[0] = m_Player.GetRandomGeneratorSyncManager().GetRandomInRange(RandomGeneratorS
■■point_2[0] = m_Player.GetRandomGeneratorSyncManager().GetRandomInRange(RandomGeneratorS
■■point_3[0] = m_Player.GetRandomGeneratorSyncManager().GetRandomInRange(RandomGeneratorS
■■point_4[0] = m_Player.GetRandomGeneratorSyncManager().GetRandomInRange(RandomGeneratorS
■■m_HandsCurvePoints.Insert(point_1);//forms a 2 dimensional spline(z is ignored)
■■m_HandsCurvePoints.Insert(point_2);
■■m_HandsCurvePoints.Insert(point_3);
■■m_HandsCurvePoints.Insert(point_4);
■■m_HandsCurvePoints.Insert("0 0 0");
■■m_HandsOffsetRelativeTime = 0.4;
■■
■■m_MouseOffsetRangeMin = 45;//in degrees min
■■m_MouseOffsetRangeMax = 130;//in degrees max
■■m_MouseOffsetDistance = 8.5;//how far should the mouse travel
■■m_MouseOffsetRelativeTime = 0.25;//[0..1] a time it takes to move the mouse the required distance rel
■
■■m_CamOffsetDistance = 0.05; //0.0125;
■■m_CamOffsetRelativeTime = 0.5;
■}
}


// ---------------------------------------------
// ------ SiagaRecoil.c - END ----------------------
// ---------------------------------------------


// ---------------------------------------------
// ------ Sickle.c - START --------------------
// ---------------------------------------------


class Sickle extends ToolBase
{
■void Sickle()
■{
■}

■override bool IsMeleeFinisher()
■{
■■return true;
■}
■
■override array<int> GetValidFinishers()
■{
■■return {EMeleeHitType.FINISHER_LIVERSTAB,EMeleeHitType.FINISHER_NECKSTAB};
■}
■
■override void SetActions()
■{
```

```
// ---------------------------------------------
// ------ SickNotfr.c - START --------------------
// ---------------------------------------------


class SickNotfr: NotifierBase
{
■void SickNotfr(NotifiersManager manager)
■{
■■m_Active = false;
■}

■override int GetNotifierType()
■{
■■return eNotifiers.NTF_SICK;
■}

■override void DisplayBadge()
■{
■■
■■DisplayElementBadge dis_elm = DisplayElementBadge.Cast(GetVirtualHud().GetElement(eDisplayEle
■■
■■if( dis_elm )
■■{
■■■dis_elm.SetLevel(eBadgeLevel.FIRST);
■■}
■}

■override void HideBadge()
■{
■■
■■DisplayElementBadge dis_elm = DisplayElementBadge.Cast(GetVirtualHud().GetElement(eDisplayEle
■■
■■if( dis_elm )
■■{
■■■dis_elm.SetLevel(eBadgeLevel.NONE);
■■}
■}
};


// ---------------------------------------------
// ------ SickNotfr.c - END ----------------------
// ---------------------------------------------


// ---------------------------------------------
// ------ SimpleMovingAverage.c - START --------------------
// ---------------------------------------------


class SimpleMovingAverage<Class T>
{
■private T m_Sum ■■■■= 0;
■private int m_Pointer ■■■= 0;
■private int m_Size ■■■■= 0;
■private ref array<T> m_Samples ■= new array<T>();

■/**
■\brief Initialize Simple Moving Average Cyclic Buffer
■■\param pSize \p size of the buffer
■■\param pDefaultValue \p initial value stored in buffer
■*/
```

```
// ---------------------------------------------
// ------ SingleShotPistol_Base.c - START --------------------
// ---------------------------------------------


enum SSPAnimState
{
■COCKED ■■= 0, ■///< default weapon state, closed and discharged
■UNCOCKED ■= 1, ■///< default weapon state, closed and discharged
};

enum SSPStableStateID
{
■UNKNOWN■■■■=  0,
■Empty■■■■=  1,
■Fireout■■■■=  2,
■Loaded■■■■=  3,
}

class SSPEmpty extends WeaponStableState
{
■override void OnEntry (WeaponEventBase e) { if (LogManager.IsWeaponLogEnable()) { wpnPrint("[wpnf
■override void OnExit (WeaponEventBase e) { super.OnExit(e); if (LogManager.IsWeaponLogEnable()) { w
■override int GetCurrentStateID () { return SSPStableStateID.Empty; }
■override bool HasBullet () { return false; }
■override bool HasMagazine () { return false; }
■override bool IsJammed () { return false; }
■override bool IsBoltOpen () { return false; }
■override bool IsRepairEnabled () { return true; }
■override void InitMuzzleArray () { m_muzzleHasBullet = {MuzzleState.E}; }
};
class SSPFireout extends WeaponStableState
{
■override void OnEntry (WeaponEventBase e) { if (LogManager.IsWeaponLogEnable()) { wpnPrint("[wpnf
■override void OnExit (WeaponEventBase e) { super.OnExit(e); if (LogManager.IsWeaponLogEnable()) { w
■override int GetCurrentStateID () { return SSPStableStateID.Fireout; }
■override bool HasBullet () { return true; }
■override bool HasMagazine () { return false; }
■override bool IsJammed () { return false; }
■override bool IsBoltOpen () { return false; }
■override bool IsRepairEnabled () { return true; }
■override void InitMuzzleArray () { m_muzzleHasBullet = {MuzzleState.F}; }
};
class SSPLoaded extends WeaponStableState
{
■override void OnEntry (WeaponEventBase e) { if (LogManager.IsWeaponLogEnable()) { wpnPrint("[wpnf
■override void OnExit (WeaponEventBase e) { super.OnExit(e); if (LogManager.IsWeaponLogEnable()) { w
■override int GetCurrentStateID () { return SSPStableStateID.Loaded; }
■override bool HasBullet () { return true; }
■override bool HasMagazine () { return false; }
■override bool IsJammed () { return false; }
■override bool IsBoltOpen () { return false; }
■override bool IsRepairEnabled () { return true; }
■override void InitMuzzleArray () { m_muzzleHasBullet = {MuzzleState.L}; }
};

class SingleShotPistol_Base: Weapon_Base
{
■override void InitStateMachine()
■{
■■// setup abilities
■■m_abilities.Insert(new AbilityRecord(WeaponActions.MECHANISM, WeaponActionMechanismTypes.M
■■m_abilities.Insert(new AbilityRecord(WeaponActions.MECHANISM, WeaponActionMechanismTypes.M
// ---------------------------------------------
// ------ SizeToChild.c - START --------------------
// ---------------------------------------------


class SizeToChild extends ScriptedWidgetEventHandler
{
■reference string■■m_ChildName;
■reference float■■■m_HorizontalOffset;
■reference float■■■m_VerticalOffset;
■reference bool■■■m_ResizeHorizontal;
■reference bool■■■m_ResizeVertical;
■
■protected Widget■■m_Root;
■protected Widget■■m_Child;
■
■protected static bool■m_IgnoredBool;
■
■void OnWidgetScriptInit(Widget w)
■{
■■m_Root■= w;
■■
■■m_Child■= m_Root.FindAnyWidget( m_ChildName );
■■if ( m_Child )
■■{
■■■ResizeParentToChild();
■■}
■}
■
■bool ResizeParentToChild()
■{
■■return ResizeParentToChild( m_IgnoredBool, -1, false );
■}
■
■bool ResizeParentToChild( out bool changed_size, int limit = -1, bool immedUpdate = true )
■{
■■float x, y, o_x, o_y, new_x, new_y;
■■if ( m_Child )
■■{
■■■m_Child.Update();
■■■m_Child.GetScreenSize( x, y );
■■■m_Root.GetScreenSize( new_x, new_y );
■■■m_Root.GetSize( o_x, o_y );
■■■
■■■bool changed■= false;
■■■bool hit_limit■= false;
■■■
■■■if ( m_ResizeHorizontal && x != new_x )
■■■{
■■■■new_x = x + m_HorizontalOffset;
■■■■changed = true;
■■■}
■■■else
■■■■new_x = o_x;
■■■
■■■if ( m_ResizeVertical && y != new_y )
■■■{
■■■■new_y = y + m_VerticalOffset;
■■■■changed = true;
■■■}
■■■else
■■■■new_y = o_y;
■■■
```

```
// ---------------------------------------------
// ------ SkateHelmet_ColorBase.c - START --------------------
// ---------------------------------------------


class SkateHelmet_ColorBase extends HelmetBase
{
}

class SkateHelmet_Black extends SkateHelmet_ColorBase {};
class SkateHelmet_Blue extends SkateHelmet_ColorBase {};
class SkateHelmet_Gray extends SkateHelmet_ColorBase {};
class SkateHelmet_Green extends SkateHelmet_ColorBase {};
class SkateHelmet_Red extends SkateHelmet_ColorBase {};


// ---------------------------------------------
// ------ SkateHelmet_ColorBase.c - END ----------------------
// ---------------------------------------------


// ---------------------------------------------
// ------ Skirt_ColorBase.c - START --------------------
// ---------------------------------------------


class Skirt_ColorBase extends Clothing
{
■override void SetActions()
■{
■■super.SetActions();
■■AddAction(ActionWringClothes);
■}
};
class Skirt_Blue extends Skirt_ColorBase {};
class Skirt_Red extends Skirt_ColorBase {};
class Skirt_White extends Skirt_ColorBase {};
class Skirt_Yellow extends Skirt_ColorBase {};


// ---------------------------------------------
// ------ Skirt_ColorBase.c - END ----------------------
// ---------------------------------------------


// ---------------------------------------------
// ------ SkorpionRecoil.c - START --------------------
// ---------------------------------------------


class Cz61Recoil: RecoilBase
{
■override void Init()
■{
■■vector point_1;
■■vector point_2;
■■vector point_3;
■■vector point_4;
■■point_1[0] = m_Player.GetRandomGeneratorSyncManager().GetRandomInRange(RandomGeneratorS
■■point_2[0] = m_Player.GetRandomGeneratorSyncManager().GetRandomInRange(RandomGeneratorS
■■point_3[0] = m_Player.GetRandomGeneratorSyncManager().GetRandomInRange(RandomGeneratorS
■■point_4[0] = m_Player.GetRandomGeneratorSyncManager().GetRandomInRange(RandomGeneratorS
■■m_HandsCurvePoints.Insert(point_1);//forms a 2 dimensional spline(z is ignored)
```

```
// ---------------------------------------------
// ------ SKS.c - START --------------------
// ---------------------------------------------


enum SKSAnimState
{
■DEFAULT ■■■= 0, ■///< default weapon state, closed and discharged
■OPENED ■■■■= 1,
■JAMMED ■■■■= 2,
};

enum SKSStableStateID
{
■UNKNOWN■■■=  0,
■SKS_CLO_BU0■■=  1,
■SKS_CLO_BU1■■=  2,
■SKS_OPN_BU0■■=  3,
■SKS_JAM_BU1 ■=  4
}

class SKS_CLO_BU0 extends WeaponStableState
{
■override void OnEntry (WeaponEventBase e) { if (LogManager.IsWeaponLogEnable()) { wpnPrint("[wpnf
■override void OnExit (WeaponEventBase e) { super.OnExit(e); if (LogManager.IsWeaponLogEnable()) { w
■override int GetCurrentStateID () { return SKSStableStateID.SKS_CLO_BU0; }
■override bool HasBullet () { return false; }
■override bool HasMagazine () { return false; }
■override bool IsJammed () { return false; }
■override bool IsRepairEnabled () { return true; }
■override void InitMuzzleArray () { m_muzzleHasBullet = {MuzzleState.E}; }

};
class SKS_CLO_BU1 extends WeaponStableState
{
■override void OnEntry (WeaponEventBase e) { if (LogManager.IsWeaponLogEnable()) { wpnPrint("[wpnf
■override void OnExit (WeaponEventBase e) { super.OnExit(e); if (LogManager.IsWeaponLogEnable()) { w
■override int GetCurrentStateID () { return SKSStableStateID.SKS_CLO_BU1; }
■override bool HasBullet () { return true; }
■override bool HasMagazine () { return false; }
■override bool IsJammed () { return false; }
■override bool IsRepairEnabled () { return true; }
■override void InitMuzzleArray () { m_muzzleHasBullet = {MuzzleState.L}; }
};
class SKS_OPN_BU0 extends WeaponStableState
{
■override void OnEntry (WeaponEventBase e) { if (LogManager.IsWeaponLogEnable()) { wpnPrint("[wpnf
■override void OnExit (WeaponEventBase e) { super.OnExit(e); if (LogManager.IsWeaponLogEnable()) { w
■override int GetCurrentStateID () { return SKSStableStateID.SKS_OPN_BU0; }
■override bool HasBullet () { return false; }
■override bool HasMagazine () { return false; }
■override bool IsJammed () { return false; }
■override bool IsBoltOpen () { return true; }
■override bool IsRepairEnabled () { return true; }
■override void InitMuzzleArray () { m_muzzleHasBullet = {MuzzleState.E}; }
};
class SKS_JAM_BU1 extends WeaponStateJammed
{
■override void OnEntry (WeaponEventBase e) { if (LogManager.IsWeaponLogEnable()) { wpnPrint("[wpnf
■override void OnExit (WeaponEventBase e) { super.OnExit(e); if (LogManager.IsWeaponLogEnable()) { w
■override int GetCurrentStateID () { return SKSStableStateID.SKS_JAM_BU1; }
■override bool HasBullet () { return true; }
■override bool HasMagazine () { return false; }
```

```
// ---------------------------------------------
// ------ SKSRecoil.c - START --------------------
// ---------------------------------------------


class SKSRecoil: RecoilBase
{
■override void Init()
■{
■■vector point_1;
■■vector point_2;
■■vector point_3;
■■vector point_4;
■■point_1[0] = m_Player.GetRandomGeneratorSyncManager().GetRandomInRange(RandomGeneratorS
■■point_2[0] = m_Player.GetRandomGeneratorSyncManager().GetRandomInRange(RandomGeneratorS
■■point_3[0] = m_Player.GetRandomGeneratorSyncManager().GetRandomInRange(RandomGeneratorS
■■point_4[0] = m_Player.GetRandomGeneratorSyncManager().GetRandomInRange(RandomGeneratorS
■■m_HandsCurvePoints.Insert(point_1);//forms a 2 dimensional spline(z is ignored)
■■m_HandsCurvePoints.Insert(point_2);
■■m_HandsCurvePoints.Insert(point_3);
■■m_HandsCurvePoints.Insert(point_4);
■■m_HandsCurvePoints.Insert("0 0 0");
■■m_HandsOffsetRelativeTime = 1;
■■
■■m_MouseOffsetRangeMin = 50;//in degrees min
■■m_MouseOffsetRangeMax = 120;//in degrees max
■■m_MouseOffsetDistance = 1.4;//how far should the mouse travel
■■m_MouseOffsetRelativeTime = 0.5;//[0..1] a time it takes to move the mouse the required distance rela
■
■■m_CamOffsetDistance = 0.015;
■■m_CamOffsetRelativeTime = 1;
■}
}


// ---------------------------------------------
// ------ SKSRecoil.c - END ----------------------
// ---------------------------------------------


// ---------------------------------------------
// ------ SKS_Bayonet.c - START --------------------
// ---------------------------------------------


class SKS_Bayonet extends Inventory_Base
{
■override bool IsMeleeFinisher()
■{
■■return true;
■}
■
■override array<int> GetValidFinishers()
■{
■■return {EMeleeHitType.FINISHER_LIVERSTAB,EMeleeHitType.FINISHER_NECKSTAB};
■}
■
■override bool CanPutAsAttachment( EntityAI parent )
■{
■■if(!super.CanPutAsAttachment(parent)) {return false;}
■■if ( parent.FindAttachmentBySlotName("suppressorImpro") == null && parent.FindAttachmentBySlotNa
■■{
■■■return true;
```

```
// ---------------------------------------------
// ------ SlacksPants_ColorBase.c - START --------------------
// ---------------------------------------------


class SlacksPants_ColorBase extends Clothing
{
■override void SetActions()
■{
■■super.SetActions();
■■AddAction(ActionWringClothes);
■}
};
class SlacksPants_Beige extends SlacksPants_ColorBase {};
class SlacksPants_Black extends SlacksPants_ColorBase {};
class SlacksPants_Blue extends SlacksPants_ColorBase {};
class SlacksPants_Brown extends SlacksPants_ColorBase {};
class SlacksPants_DarkGrey extends SlacksPants_ColorBase {};
class SlacksPants_Khaki extends SlacksPants_ColorBase {};
class SlacksPants_LightGrey extends SlacksPants_ColorBase {};
class SlacksPants_White extends SlacksPants_ColorBase {};


// ---------------------------------------------
// ------ SlacksPants_ColorBase.c - END ----------------------
// ---------------------------------------------


// ---------------------------------------------
// ------ SledgeHammer.c - START --------------------
// ---------------------------------------------


class SledgeHammer extends Inventory_Base
{
■override void SetActions()
■{
■■super.SetActions();
■■
■■AddAction(ActionClapBearTrapWithThisItem);
■■AddAction(ActionMineRock);
■■AddAction(ActionDismantlePart);
■■AddAction(ActionBuildPart);
■■//AddAction(ActionDestroyPart);
■}
}


// ---------------------------------------------
// ------ SledgeHammer.c - END ----------------------
// ---------------------------------------------


// ---------------------------------------------
// ------ SlicedPumpkin.c - START --------------------
// ---------------------------------------------


class SlicedPumpkin : Edible_Base
{
■override bool CanBeCooked()
■{
■■return true;
```

```
// ---------------------------------------------
// ------ Slot.c - START --------------------
// ---------------------------------------------


enum eFertlityState
{
■NONE = 0,
■FERTILIZED = 1
■//This will be used to set bit values (DO NOT ADD MORE VALUES)
}

enum eWateredState
{
■DRY = 0,
■WET = 1
■//Used to improve readability of watered state changes
}

class Slot
{
■static const int ■■STATE_DIGGED ■■= 1;
■static const int ■■STATE_PLANTED ■■= 2;
■
■private int ■■■m_WaterQuantity;
■static private int ■■m_WaterNeeded ■■= 190; // How much water is needed to water a plant from a bott
■static private int ■■m_WaterMax ■■■= 200;
■
■float m_Fertility;
■float m_FertilizerUsage;
■float m_FertilizerQuantity;
■int m_slotIndex;
■int m_slotId;
■
■string m_FertilizerType;
■int    m_FertilityState = eFertlityState.NONE;
■int    m_WateredState = eWateredState.DRY;
■string m_DiggedSlotComponent; // example: "Component02" for the 1st slot in GardenBase
■string m_PlantType;
■private ItemBase m_Seed;
■private GardenBase m_Garden;
■
■float m_HarvestingEfficiency;
■
■int m_State;
■
■private PlantBase m_Plant;
■
■void Slot( float base_fertility )
■{
■■m_Seed = NULL;
■■m_Plant = NULL;
■■m_WaterQuantity = 0.0;
■■Init( base_fertility );
■}

■void ~Slot()
■{
■■if (m_Plant  &&  GetGame()) // GetGame() returns NULL when the game is being quit!
■■{
■■■GetGame().ObjectDelete( GetPlant() );
■■}
■}
```

```
// ---------------------------------------------
// ------ SlotsContainer.c - START --------------------
// ---------------------------------------------


class SlotsContainer: Container
{
■protected ref array<ref SlotsIcon>■m_Icons;
■int m_VisibleColumnCount;
■
■void SlotsContainer( LayoutHolder parent, EntityAI slot_parent )
■{
■■m_Icons = new array<ref SlotsIcon>;
■■for ( int i = 0; i < ITEMS_IN_ROW; i++ )
■■{
■■■if ( GetRootWidget().FindAnyWidget( "Icon" + i ) )
■■■■m_Icons.Insert( new SlotsIcon( this, GetRootWidget().FindAnyWidget( "Icon" + i ), i , slot_parent) );
■■■else
■■■{
■■■■Widget child = GetRootWidget().GetChildren();
■■■■while ( child )
■■■■{
■■■■■child = child.GetSibling();
■■■■}
■■■}
■■}
■}
■
■override void SetDefaultFocus( bool while_micromanagment_mode = false )
■{■■■
■■m_FocusedColumn = 0;
■■for (int i = 0; i < GetColumnCount(); i++)
■■{
■■■SlotsIcon icon = m_Icons[i];
■■■if (icon.IsVisible())
■■■{
■■■■m_FocusedColumn = i;
■■■■break;
■■■}
■■}
■■■■■
■■SetFocus( m_FocusedColumn );
■}
■
■override void SetLastFocus()
■{
■■SetDefaultFocus();
■}
■
■void SetFocus( int index )
■{
■■if ( index >= 0 && index < m_Icons.Count() )
■■{
■■■m_Icons.Get( index ).GetCursorWidget().Show( true );
■■}
■■UpdateIcon();
■}
■
■void SetVisibleFocus( int index )
■{
■■Unfocus();
■■int visible_icons_count = 0;
■■
```

```
// ---------------------------------------------
// ------ SlotsIcon.c - START --------------------
// ---------------------------------------------


class SlotsIcon: LayoutHolder
{
■protected static int ■■■m_NormalWidth;
■protected static int ■■■m_NormalHeight;
■
■protected bool■■■■■m_IsWeapon■■■= false;
■protected bool■■■■■m_IsMagazine■■= false;
■protected bool■■■■■m_HasTemperature■= false;
■protected bool■■■■■m_HasQuantity■■= false;
■protected bool■■■■■m_HasItemSize■■= false;
■protected float■■■■■m_CurrQuantity■■= -1;
■
■protected EntityAI■■■■m_Obj;
■protected ItemBase■■■■m_Item;
■protected EntityAI■■■■m_SlotParent;
■protected Container■■■■m_Container;
■protected int■■■■■m_SlotID;
■protected bool■■■■■m_IsDragged■■■= false;
■
■protected Widget■■■■m_PanelWidget;
■
■protected Widget■■■■m_CursorWidget;
■protected Widget■■■■m_ColWidget;
■protected Widget■■■■m_MountedWidget;
■protected Widget■■■■m_OutOfReachWidget;
■protected Widget■■■■m_ReservedWidget;
■
■protected ItemPreviewWidget■■m_ItemPreview;
■protected ImageWidget■■■m_GhostSlot;
■
■protected Widget■■■■m_ColorWidget;
■protected Widget■■■■m_SelectedPanel;
■protected Widget■■■■m_EmptySelectedPanel;
■
■protected Widget■■■■m_QuantityPanel;
■protected TextWidget■■■m_QuantityItem;
■protected ProgressBarWidget■■m_QuantityProgress;
■protected Widget■■■■m_QuantityStack;
■
■protected string■■■■m_SlotDisplayName;
■protected string■■■■m_SlotDesc;
■
■protected Widget■■■■m_ItemSizePanel;
■protected TextWidget■■■m_ItemSizeWidget;
■
■protected Widget■■■■m_AmmoIcon;
■
■protected Widget■■■■m_RadialIconPanel;
■protected Widget■■■■m_RadialIconClosed;
■protected Widget■■■■m_RadialIcon;
■
■protected bool■■■■■m_Reserved;

■void SlotsIcon( LayoutHolder parent, Widget root, int index, EntityAI slot_parent )
■{
■■m_MainWidget■■■= root;
■■
■■m_PanelWidget■■■= m_MainWidget.FindAnyWidget( "PanelWidget" + index );
```

```
// ---------------------------------------------
// ------ SmallGasCannister.c - START --------------------
// ---------------------------------------------


class SmallGasCannister extends ItemBase
{
■override bool CanExplodeInFire()
■{
■■return true;
■}
}


// ---------------------------------------------
// ------ SmallGasCannister.c - END ----------------------
// ---------------------------------------------


// ---------------------------------------------
// ------ SmallGuts.c - START --------------------
// ---------------------------------------------


class SmallGuts extends Edible_Base
{
■override bool CanPutAsAttachment( EntityAI parent )
■{
■■if(!super.CanPutAsAttachment(parent)) {return false;}
■■const int SLOTS_ARRAY = 8;
■■bool is_barrel = false;
■■bool is_opened_barrel = false;
■■bool slot_test = true;
■■string slot_names[SLOTS_ARRAY] = { "BerryR", "BerryB", "Nails", "OakBark", "BirchBark", "Lime", "Dis

■■
■■// is barrel
■■if ( parent.IsKindOf("Barrel_ColorBase") )
■■{
■■■is_barrel = true;
■■}

■■// is opened barrel■■■■
■■if ( is_barrel && parent.GetAnimationPhase("Lid") == 1 )
■■{
■■■is_opened_barrel = true;
■■}

■■// all of the barrel slots are empty
■■for ( int i = 0; i < SLOTS_ARRAY ; i++ )
■■{
■■■if ( parent.FindAttachmentBySlotName(slot_names[i]) != NULL )
■■■{
■■■■slot_test = false;
■■■■break;
■■■}
■■}
■■
■■if ( ( is_opened_barrel && slot_test ) || !is_barrel )
■■{
■■■return true;
■■}
■■return false;
```

```
// ---------------------------------------------
// ------ SmershBag.c - START --------------------
// ---------------------------------------------


class SmershBag extends Clothing
{
■
};


// ---------------------------------------------
// ------ SmershBag.c - END ----------------------
// ---------------------------------------------


// ---------------------------------------------
// ------ SmershVest.c - START --------------------
// ---------------------------------------------


class SmershVest extends Clothing
{
■override void EEItemAttached(EntityAI item, string slot_name)
■{
■■super.EEItemAttached(item,slot_name);
■■
■■if (SmershBag.Cast(item))
■■{
■■■ShowSelection("Buttpack");
■■}
■}
■
■override void EEItemDetached(EntityAI item, string slot_name)
■{
■■super.EEItemDetached(item,slot_name);
■■
■■if (SmershBag.Cast(item))
■■{
■■■HideSelection("Buttpack");
■■}
■}
■
};


// ---------------------------------------------
// ------ SmershVest.c - END ----------------------
// ---------------------------------------------


// ---------------------------------------------
// ------ SmokeGrenadeBase.c - START --------------------
// ---------------------------------------------


enum ESmokeGrenadeState
{
■NO_SMOKE ■= 0
■START ■ ■= 1,
■LOOP ■ ■= 2,
■END■■■= 3,
■//! ---
```

```
// ----------------------------------------------
// ------ SmokeSimulation.c - START --------------------
// ----------------------------------------------


class SmokeSimulation : Entity
{
■protected Particle ■■■m_ParMainSmoke;
■int particle_id;
■
■void SmokeSimulation()
■{
■■particle_id = ParticleList.GRENADE_M18_PURPLE_LOOP;
■}

■void OnFire( Entity flare)
■{
■■m_ParMainSmoke = ParticleManager.GetInstance().PlayOnObject( particle_id, flare);
■■m_ParMainSmoke.SetWiggle( 7, 0.3);
■}

■void ~SmokeSimulation()
■{
■■if (m_ParMainSmoke)
■■■m_ParMainSmoke.Stop();
■}
}

class SmokeSimulation_Black : SmokeSimulation
{
■void SmokeSimulation_Black()
■{
■■particle_id = ParticleList.GRENADE_M18_BLACK_LOOP;
■}
}
class SmokeSimulation_White : SmokeSimulation
{
■void SmokeSimulation_White()
■{
■■particle_id = ParticleList.GRENADE_M18_WHITE_LOOP;
■}
}
class SmokeSimulation_Red : SmokeSimulation
{
■void SmokeSimulation_Red()
■{
■■particle_id = ParticleList.GRENADE_M18_RED_LOOP;
■}
}
class SmokeSimulation_Green : SmokeSimulation
{
■void SmokeSimulation_Green()
■{
■■particle_id = ParticleList.GRENADE_M18_GREEN_LOOP;
■}
}



// ----------------------------------------------
// ------ SmokeSimulation.c - END ----------------------
// ----------------------------------------------
// ---------------------------------------------
// ------ SmptAnimMeta.c - START --------------------
// ---------------------------------------------


enum eAnimFinishType
{
■SUCCESS,
■FAILURE,
}

class SmptAnimMetaBase
{
■bool■m_IsPlaying;
■SymptomManager m_Manager;
■PlayerBase m_Player;
■int m_AnimID;
■int m_SymptomType;
■bool m_DestroyRequested;
■bool m_Canceled;
■
■void SmptAnimMetaBase()
■{
■}
■
■void Init(ParamsReadContext ctx, SymptomManager manager, PlayerBase player)
■{
■■m_Manager = manager;
■■m_Player = player;
■}
■
■bool IsPlaying()
■{
■■return m_IsPlaying;
■}
■
■bool IsDestroyReqested()
■{
■■return m_DestroyRequested;
■}
■
■void AnimFinished(eAnimFinishType type)
■{
■■m_DestroyRequested = true;
■■SymptomBase Symptom = m_Manager.GetCurrentPrimaryActiveSymptom();
■■
■■if( type == eAnimFinishType.FAILURE)//   <--------------- FAILED
■■{
■■■if( m_Player.GetInstanceType() == DayZPlayerInstanceType.INSTANCETYPE_SERVER || !GetGam
■■■{
■■■■if( Symptom )
■■■■{
■■■■■Symptom.AnimationPlayFailed();
■■■■}
■■■}
■■}
■■else if( type == eAnimFinishType.SUCCESS)//   <--------------- SUCCESS
■■{
■■■if( m_Player.GetInstanceType() == DayZPlayerInstanceType.INSTANCETYPE_SERVER || !GetGam
■■■{
■■■■if( Symptom )
■■■■{
■■■■■Symptom.AnimationFinish();
```

```
// ---------------------------------------------
// ------ Sneakers_ColorBase.c - START --------------------
// ---------------------------------------------


class Sneakers_ColorBase extends Clothing {};
class Sneakers_Black extends Sneakers_ColorBase {};
class Sneakers_Gray extends Sneakers_ColorBase {};
class Sneakers_Green extends Sneakers_ColorBase {};
class Sneakers_Red extends Sneakers_ColorBase {};
class Sneakers_White extends Sneakers_ColorBase {};


// ---------------------------------------------
// ------ Sneakers_ColorBase.c - END ----------------------
// ---------------------------------------------


// ---------------------------------------------
// ------ SneezeState.c - START --------------------
// ---------------------------------------------


class SneezeSymptom extends SymptomBase
{
■//this is just for the Symptom parameters set-up and is called even if the Symptom doesn't execute, don't
■override void OnInit()
■{
■■m_SymptomType = SymptomTypes.PRIMARY;
■■m_Priority = 100;
■■m_ID = SymptomIDs.SYMPTOM_SNEEZE;
■■m_DestroyOnAnimFinish = true;
■■m_SyncToClient = false;
■}
■
■//!gets called every frame
■override void OnUpdateServer(PlayerBase player, float deltatime)
■{

■}

■override void OnUpdateClient(PlayerBase player, float deltatime)
■{
■}
■
■//!gets called once on an Symptom which is being activated
■override void OnGetActivatedServer(PlayerBase player)
■{
■■if (LogManager.IsSymptomLogEnable()) Debug.SymptomLog("n/a", this.ToString(), "n/a", "OnGetActiva
■■if( m_Manager.GetCurrentCommandID() == DayZPlayerConstants.COMMANDID_MOVE && !player.Is
■■{
■■■PlayAnimationADD(0);
■■}
■■else
■■{
■■■PlaySound(EPlayerSoundEventID.SYMPTOM_SNEEZE);
■■}
■■player.SpreadAgentsEx(3);
■}

■//!gets called once on an Symptom which is being activated
■override void OnGetActivatedClient(PlayerBase player)
■{
```

```
// ---------------------------------------------
// ------ SodaCan_ColorBase.c - START --------------------
// ---------------------------------------------


class SodaCan_ColorBase : Edible_Base
{
■override void SetActions()
■{
■■super.SetActions();
■■
■■//AddAction(ActionWashHandsItem);
■■AddAction(ActionForceDrink);
■■AddAction(ActionDrinkCan);
■}
};




// ---------------------------------------------
// ------ SodaCan_ColorBase.c - END ----------------------
// ---------------------------------------------


// ---------------------------------------------
// ------ SodaCan_Empty.c - START --------------------
// ---------------------------------------------


class SodaCan_Empty extends ItemBase {}


// ---------------------------------------------
// ------ SodaCan_Empty.c - END ----------------------
// ---------------------------------------------


// ---------------------------------------------
// ------ SoftSkillsManager.c - START --------------------
// ---------------------------------------------


class SoftSkillsManager
{■
■protected PlayerBase■■■m_Player;
■protected float ■■■■m_SpecialtyLevel;
■protected float ■■■■m_RoughLevel;
■protected float ■■■■m_PreciseLevel;
■
■protected bool■■■■■m_IsLinear;
■protected bool■■■■■m_IsActive;
■protected bool■■■■■m_IsCoolDown;

■protected int ■■■■■m_UserActionsCounter;■

■static protected const int ■■DEFAULT_EFFICIENCY = 0;
■static protected const float ■PRECISE_WEIGHT_LIMIT = -1;
■static protected const float ■ROUGH_WEIGHT_LIMIT = 1;
■static protected const float ■COOLDOWN_TIMER = 5; //default 5,

■protected ref Timer ■■■m_CoolDownTimer = new Timer();

■protected bool ■■■■m_IsDebugMode;
```

```
// ---------------------------------------------
// ------ someMission.c - START --------------------
// ---------------------------------------------


Mission CreateMission(string path)
{
■Print("Creating Mission: "+ path);
■
■// g_Game.SetMissionPath(path); Done from C++ now

■if (g_Game.IsMultiplayer() && g_Game.IsServer())
■{
■■return new MissionServer;
■}

#ifdef NO_GUI
■return new MissionDummy;
#endif
■MissionMainMenu m;
■if (path.Contains("NoCutscene"))
■{
■■m = new MissionMainMenu();
■■m.m_NoCutscene = true;
■■return m;
■}
■
■if (path.Contains("intro"))
■{
■■m = new MissionMainMenu();
■■m.m_NoCutscene = false;
■■return m;
■}
■else
■{
■■if( path == "" )
■■{
■■■return new MissionDummy;
■■}
#ifndef NO_GUI_INGAME
■■return new MissionGameplay;
#else
■■return new MissionDummy;
#endif
■}
}



// ---------------------------------------------
// ------ someMission.c - END ----------------------
// ---------------------------------------------


// ---------------------------------------------
// ------ Sound.c - START --------------------
// ---------------------------------------------


enum WaveKind
{
■WAVEEFFECT,
■WAVEEFFECTEX
```

```
// ----------------------------------------------
// ------ SoundEvents.c - START --------------------
// ----------------------------------------------


class SoundEventBase
{
■AbstractWave ■m_SoundSetCallback;
■int ■■■m_Type;
■int ■■■m_ID;
■int ■■■m_SoundVoiceAnimEventClassID;
■bool ■■■m_RequestDestroy;
■string ■■■m_SoundSetNameRoot;
■bool ■■■m_SkipForControlledCharacter;
■int■■■■m_Param;
■
■
■void ~SoundEventBase()
■{
■■if(m_SoundSetCallback) m_SoundSetCallback.Stop();
■}
■
■//obsolete function, now possible to set a param
■bool IsSkipForControlled()
■{
■■return m_SkipForControlledCharacter;
■}
■
■void Tick()
■{
■■if(!m_SoundSetCallback)
■■{
■■■m_RequestDestroy = true;
■■}
■}
■
■bool IsSoundCallbackExist()
■{
■■if( m_SoundSetCallback )
■■{
■■■return true;
■■}
■■
■■return false;
■}
■
■bool IsDestroyRequested()
■{
■■return m_RequestDestroy;
■}
■
■bool CanPlay()
■{
■■return true;
■}
■
■bool Play()
■{
■■return true;
■}
■
■void OnPlay(PlayerBase player);
■
```

```
// ---------------------------------------------
// ------ SoundOnVehicle.c - START --------------------
// ---------------------------------------------


class SoundOnVehicle extends Entity
{
■proto native float GetSoundLength();
};

class SoundWaveOnVehicle extends Entity
{
};




// ---------------------------------------------
// ------ SoundOnVehicle.c - END ----------------------
// ---------------------------------------------


// ---------------------------------------------
// ------ SoundSetMap.c - START --------------------
// ---------------------------------------------


class SoundSetMap
{
■ref static map<int, string> m_SoundSetMapIDByID = new map<int, string>;
■ref static map<string, int> m_SoundSetMapIDByName = new map<string, int>;
■
■static void Init()
■{
■■string path = "CfgSoundSets";
■■
■■int soundCount = GetGame().ConfigGetChildrenCount(path);
■■
■■for (int i = 1; i < soundCount; i++)
■■{
■■■string soundClassName;
■■■GetGame().ConfigGetChildName(path, i, soundClassName);
■■■m_SoundSetMapIDByID.Insert(i,soundClassName);
■■■m_SoundSetMapIDByName.Insert(soundClassName,i);
■■■//PrintString(soundClassName);
■■}
■}
■
■static string GetSoundSetName(int id)
■{
■■return m_SoundSetMapIDByID.Get(id);
■}
■
■static int GetSoundSetID(string name)
■{
■■return m_SoundSetMapIDByName.Get(name);
■}

}


// ---------------------------------------------
// ------ SoundSetMap.c - END ----------------------
// ---------------------------------------------
// ---------------------------------------------
// ------ SpacerBase.c - START --------------------
// ---------------------------------------------


// -------------------------------------------------------
class SpacerBase : ScriptedWidgetEventHandler
{
■protected Widget m_root;
■protected int m_count;

■// -------------------------------------------------------
■void OnWidgetScriptInit(Widget w)
■{
■■m_root = w;
■■m_count = 0;
■
■■Widget child = m_root.GetChildren();
■■while (child)
■■{
■■■m_count++;
■■■child.SetFlags(WidgetFlags.EXACTPOS | WidgetFlags.EXACTSIZE, false);
■■■child = child.GetSibling();
■■}
■
■■m_root.SetHandler(this);
■}
■
■// -------------------------------------------------------
■override bool OnUpdate(Widget w)
■{
■■if (w == m_root) UpdateLayout();
■■return false;
■}
■
■// -------------------------------------------------------
■override bool OnChildAdd( Widget  w, Widget  child)
■{
■■m_count++;
■■child.SetFlags(WidgetFlags.EXACTPOS | WidgetFlags.EXACTSIZE, false);
■■return false;
■}
■
■// -------------------------------------------------------
■override bool OnChildRemove( Widget  w, Widget  child)
■{
■■m_count--;
■■return false;
■}
■
■// -------------------------------------------------------
■protected int GetChildIndex(Widget w)
■{
■■Widget child = m_root.GetChildren();
■
■■int index = 0;
■■while (child)
■■{
■■■if (child == w) return index;
■
■■■index++;
■■■child = child.GetSibling();
■■}
```

```
// ---------------------------------------------
// ------ SpaghettiCan.c - START --------------------
// ---------------------------------------------


class SpaghettiCan : Edible_Base
{
■override void Open()
■{
■■ReplaceEdibleWithNew("SpaghettiCan_Opened");
■}
■
■override bool IsOpen()
■{
■■return false;
■}
}



// ---------------------------------------------
// ------ SpaghettiCan.c - END ---------------------
// ---------------------------------------------


// ---------------------------------------------
// ------ SpaghettiCan_Opened.c - START --------------------
// ---------------------------------------------


class SpaghettiCan_Opened: Edible_Base
{
■override bool CanDecay()
■{
■■return true;
■}
■
■override bool CanProcessDecay()
■{
■■return !(GetAgents() & eAgents.FOOD_POISON);
■}
■
■override void SetActions()
■{
■■super.SetActions();
■■
■■AddAction(ActionForceFeedCan);
■■AddAction(ActionEatCan);
■}
}



// ---------------------------------------------
// ------ SpaghettiCan_Opened.c - END ---------------------
// ---------------------------------------------
// ---------------------------------------------
// ------ SphereTrigger.c - START --------------------
// ---------------------------------------------


//! Trigger with spherical shape
class SphereTrigger : Trigger
{
■override void EOnInit(IEntity other, int extra)
■{
■■SetCollisionSphere(3);
■}
■
#ifdef DEVELOPER
■override protected Shape DrawDebugShape(vector pos, vector min, vector max, float radius, int color)
■{
■■Shape dbgShape = Debug.DrawSphere(pos, radius, color, ShapeFlags.TRANSP|ShapeFlags.NOZWR
■■dbgTargets.Insert( dbgShape );
■■return dbgShape;
■}
#endif
};


// ---------------------------------------------
// ------ SphereTrigger.c - END ----------------------
// ---------------------------------------------


// ---------------------------------------------
// ------ Splint.c - START --------------------
// ---------------------------------------------


class Splint: Inventory_Base
{
■override void SetActions()
■{
■■super.SetActions();
■■
■■AddAction(ActionSplintTarget);
■■AddAction(ActionSplintSelf);
■}
};

class Splint_Applied: Clothing
{
■void ~Splint_Applied()
■{
■■
■}
};



// ---------------------------------------------
// ------ Splint.c - END ----------------------
// ---------------------------------------------
// ---------------------------------------------
// ------ SplitBroom.c - START --------------------
// ---------------------------------------------


class SplitBroom extends RecipeBase
{■
■override void Init()
■{
■■m_Name = "#STR_split0";
■■m_IsInstaRecipe = false;//should this recipe be performed instantly without animation
■■m_AnimationLength = 1;//animation length in relative time units
■■m_Specialty = 0.01;// value > 0 for roughness, value < 0 for precision
■■
■■
■■//conditions
■■m_MinDamageIngredient[0] = -1;//-1 = disable check
■■m_MaxDamageIngredient[0] = 3;//-1 = disable check
■■
■■m_MinQuantityIngredient[0] = -1;//-1 = disable check
■■m_MaxQuantityIngredient[0] = -1;//-1 = disable check
■■
■■m_MinDamageIngredient[1] = -1;//-1 = disable check
■■m_MaxDamageIngredient[1] = 3;//-1 = disable check
■■
■■m_MinQuantityIngredient[1] = -1;//-1 = disable check
■■m_MaxQuantityIngredient[1] = -1;//-1 = disable check
■■//----------------------------------------------------------------------------------------------------------------------
■■
■■//INGREDIENTS
■■//ingredient 1
■■InsertIngredient(0,"Broom");//you can insert multiple ingredients this way
■■InsertIngredient(0,"Broom_Birch");//you can insert multiple ingredients this way
■■
■■m_IngredientAddHealth[0] = 0;// 0 = do nothing
■■m_IngredientSetHealth[0] = -1; // -1 = do nothing
■■m_IngredientAddQuantity[0] = 0;// 0 = do nothing
■■m_IngredientDestroy[0] = true;//true = destroy, false = do nothing
■■m_IngredientUseSoftSkills[0] = false;// set 'true' to allow modification of the values by softskills on this i
■■
■■//ingredient 2
■■InsertIngredient(1,"Sickle");//you can insert multiple ingredients this way
■■InsertIngredient(1,"KukriKnife");
■■InsertIngredient(1,"FangeKnife");
■■InsertIngredient(1,"Hacksaw");
■■InsertIngredient(1,"HandSaw");
■■InsertIngredient(1,"KitchenKnife");
■■InsertIngredient(1,"SteakKnife");
■■InsertIngredient(1,"HayHook");
■■InsertIngredient(1,"StoneKnife");
■■InsertIngredient(1,"Cleaver");
■■InsertIngredient(1,"CombatKnife");
■■InsertIngredient(1,"HuntingKnife");
■■InsertIngredient(1,"Machete");
■■InsertIngredient(1,"CrudeMachete");
■■InsertIngredient(1,"OrientalMachete");
■■InsertIngredient(1,"Crowbar");
■■InsertIngredient(1,"Pickaxe");
■■InsertIngredient(1,"WoodAxe");
■■InsertIngredient(1,"Hatchet");
■■InsertIngredient(1,"FirefighterAxe");
■■InsertIngredient(1,"Sword");
■■InsertIngredient(1,"AK_Bayonet");
```

```
// ---------------------------------------------
// ------ SplitFirewood.c - START --------------------
// ---------------------------------------------


class SplitFirewood extends RecipeBase
{■
■override void Init()
■{
■■m_Name = "#STR_split0";
■■m_IsInstaRecipe = false;//should this recipe be performed instantly without animation
■■m_AnimationLength = 1;//animation length in relative time units
■■m_Specialty = 0.01;// value > 0 for roughness, value < 0 for precision
■■
■■
■■//conditions
■■m_MinDamageIngredient[0] = -1;//-1 = disable check
■■m_MaxDamageIngredient[0] = 3;//-1 = disable check
■■
■■m_MinQuantityIngredient[0] = 1;//-1 = disable check
■■m_MaxQuantityIngredient[0] = -1;//-1 = disable check
■■
■■m_MinDamageIngredient[1] = -1;//-1 = disable check
■■m_MaxDamageIngredient[1] = 3;//-1 = disable check
■■
■■m_MinQuantityIngredient[1] = -1;//-1 = disable check
■■m_MaxQuantityIngredient[1] = -1;//-1 = disable check
■■//----------------------------------------------------------------------------------------------------------------------
■■
■■//INGREDIENTS
■■//ingredient 1
■■InsertIngredient(0,"Firewood");//you can insert multiple ingredients this way
■■
■■m_IngredientAddHealth[0] = 0;// 0 = do nothing
■■m_IngredientSetHealth[0] = -1; // -1 = do nothing
■■m_IngredientAddQuantity[0] = -1;// 0 = do nothing
■■m_IngredientDestroy[0] = false;//true = destroy, false = do nothing
■■m_IngredientUseSoftSkills[0] = false;// set 'true' to allow modification of the values by softskills on this i
■■
■■//ingredient 2
■■InsertIngredient(1,"Hacksaw");
■■InsertIngredient(1,"HandSaw");
■■InsertIngredient(1,"Pickaxe");
■■InsertIngredient(1,"WoodAxe");
■■InsertIngredient(1,"Hatchet");
■■InsertIngredient(1,"FirefighterAxe");
■■
■■m_IngredientAddHealth[1] = -4;// 0 = do nothing
■■m_IngredientSetHealth[1] = -1; // -1 = do nothing
■■m_IngredientAddQuantity[1] = 0;// 0 = do nothing
■■m_IngredientDestroy[1] = false;// false = do nothing
■■m_IngredientUseSoftSkills[1] = true;// set 'true' to allow modification of the values by softskills on this in
■■
■■//----------------------------------------------------------------------------------------------------------------------
■■
■■//result1
■■AddResult("WoodenStick");//add results here

■■m_ResultSetFullQuantity[0] = false;//true = set full quantity, false = do nothing
■■m_ResultSetQuantity[0] = 3;//-1 = do nothing
■■m_ResultSetHealth[0] = -1;//-1 = do nothing
■■m_ResultInheritsHealth[0] = 0;// (value) == -1 means do nothing; a (value) >= 0 means this result will ir
■■m_ResultInheritsColor[0] = -1;// (value) == -1 means do nothing; a (value) >= 0 means this result class
```

```
// ---------------------------------------------
// ------ SplitItemUtils.c - START --------------------
// ---------------------------------------------


class SplitItemUtils
{
■static void TakeOrSplitToInventory ( notnull PlayerBase player, notnull EntityAI target, notnull EntityAI ite
■{
■■ItemBase item_base = ItemBase.Cast( item );
■■
■■if( !item.GetInventory().CanRemoveEntity() || !player.CanManipulateInventory() )
■■■return;
■■
■■InventoryLocation il = new InventoryLocation;
■■if( target.GetInventory().FindFreeLocationFor( item, FindInventoryLocationType.ANY, il) )
■■{
■■■if( item_base.GetTargetQuantityMax(il.GetSlot()) >= item_base.GetQuantity() )
■■■{
■■■■if( il.GetType() == InventoryLocationType.ATTACHMENT )
■■■■{
■■■■■player.PredictiveTakeEntityToTargetAttachmentEx(il.GetParent(), item, il.GetSlot());
■■■■}
■■■■else
■■■■{
■■■■■InventoryLocation src = new InventoryLocation;
■■■■■if (item.GetInventory().GetCurrentInventoryLocation(src))
■■■■■■player.PredictiveTakeToDst(src, il);
■■■■
■■■■}
■■■}
■■■else
■■■{
■■■■item_base.SplitIntoStackMaxClient( il.GetParent(), il.GetSlot() );
■■■}
■■}
■}
■
■static void TakeOrSplitToInventoryLocation ( notnull PlayerBase player, notnull InventoryLocation dst)
■{
■■ItemBase item_base = ItemBase.Cast( dst.GetItem() );■
■■int slot = dst.GetSlot();

■■if( !dst.GetItem().GetInventory().CanRemoveEntity() || !player.CanManipulateInventory() )
■■■return;
■■
■■float stack_max = item_base.GetTargetQuantityMax(slot);
■■
■■if( stack_max >= item_base.GetQuantity() )
■■{
■■■InventoryLocation src = new InventoryLocation;
■■■if (dst.GetItem().GetInventory().GetCurrentInventoryLocation(src))
■■■{
■■■■player.PredictiveTakeToDst(src, dst);
■■■}
■■■else
■■■■Error("TakeIntoCargoEx cannot get src for dst=" + dst.DumpToString());
■■}
■■else
■■{
■■■item_base.SplitIntoStackMaxToInventoryLocationClient( dst );
■■}
■}
```

```
// ---------------------------------------------
// ------ SplitLongWoodenStick.c - START --------------------
// ---------------------------------------------


class SplitLongWoodenStick extends RecipeBase
{■
■override void Init()
■{
■■m_Name = "#STR_split0";
■■m_IsInstaRecipe = false;//should this recipe be performed instantly without animation
■■m_AnimationLength = 1;//animation length in relative time units
■■m_Specialty = 0.01;// value > 0 for roughness, value < 0 for precision
■■
■■
■■//conditions
■■m_MinDamageIngredient[0] = -1;//-1 = disable check
■■m_MaxDamageIngredient[0] = 3;//-1 = disable check
■■
■■m_MinQuantityIngredient[0] = -1;//-1 = disable check
■■m_MaxQuantityIngredient[0] = -1;//-1 = disable check
■■
■■m_MinDamageIngredient[1] = -1;//-1 = disable check
■■m_MaxDamageIngredient[1] = 3;//-1 = disable check
■■
■■m_MinQuantityIngredient[1] = -1;//-1 = disable check
■■m_MaxQuantityIngredient[1] = -1;//-1 = disable check
■■//----------------------------------------------------------------------------------------------------------------------
■■
■■//INGREDIENTS
■■//ingredient 1
■■InsertIngredient(0,"LongWoodenStick");//you can insert multiple ingredients this way
■■InsertIngredient(0,"SharpWoodenStick");//you can insert multiple ingredients this way
■■
■■m_IngredientAddHealth[0] = 0;// 0 = do nothing
■■m_IngredientSetHealth[0] = -1; // -1 = do nothing
■■m_IngredientAddQuantity[0] = 0;// 0 = do nothing
■■m_IngredientDestroy[0] = true;//true = destroy, false = do nothing
■■m_IngredientUseSoftSkills[0] = false;// set 'true' to allow modification of the values by softskills on this i
■■
■■//ingredient 2
■■InsertIngredient(1,"Sickle");//you can insert multiple ingredients this way
■■InsertIngredient(1,"KukriKnife");
■■InsertIngredient(1,"FangeKnife");
■■InsertIngredient(1,"Hacksaw");
■■InsertIngredient(1,"HandSaw");
■■InsertIngredient(1,"KitchenKnife");
■■InsertIngredient(1,"SteakKnife");
■■InsertIngredient(1,"HayHook");
■■InsertIngredient(1,"StoneKnife");
■■InsertIngredient(1,"Cleaver");
■■InsertIngredient(1,"CombatKnife");
■■InsertIngredient(1,"HuntingKnife");
■■InsertIngredient(1,"Machete");
■■InsertIngredient(1,"CrudeMachete");
■■InsertIngredient(1,"OrientalMachete");
■■InsertIngredient(1,"Crowbar");
■■InsertIngredient(1,"Pickaxe");
■■InsertIngredient(1,"WoodAxe");
■■InsertIngredient(1,"Hatchet");
■■InsertIngredient(1,"FirefighterAxe");
■■InsertIngredient(1,"Sword");
■■InsertIngredient(1,"AK_Bayonet");
```

```
// ---------------------------------------------
// ------ SplitStones.c - START --------------------
// ---------------------------------------------


class SplitStones extends RecipeBase
{■
■override void Init()
■{
■■m_Name = "#STR_split0";
■■m_IsInstaRecipe = false;//should this recipe be performed instantly without animation
■■m_AnimationLength = 1.5;//animation length in relative time units
■■m_Specialty = 0.02;// value > 0 for roughness, value < 0 for precision
■■
■■
■■//conditions
■■m_MinDamageIngredient[0] = -1;//-1 = disable check
■■m_MaxDamageIngredient[0] = 3;//-1 = disable check
■■
■■m_MinQuantityIngredient[0] = 1;//-1 = disable check
■■m_MaxQuantityIngredient[0] = -1;//-1 = disable check
■■
■■m_MinDamageIngredient[1] = -1;//-1 = disable check
■■m_MaxDamageIngredient[1] = 3;//-1 = disable check
■■
■■m_MinQuantityIngredient[1] = -1;//-1 = disable check
■■m_MaxQuantityIngredient[1] = -1;//-1 = disable check
■■//----------------------------------------------------------------------------------------------------------------------
■■
■■//INGREDIENTS
■■//ingredient 1
■■InsertIngredient(0,"Stone");//you can insert multiple ingredients this way

■■m_IngredientAddHealth[0] = 0;// 0 = do nothing
■■m_IngredientSetHealth[0] = -1; // -1 = do nothing
■■m_IngredientAddQuantity[0] = -1;// 0 = do nothing
■■m_IngredientDestroy[0] = false;//true = destroy, false = do nothing
■■m_IngredientUseSoftSkills[0] = false;// set 'true' to allow modification of the values by softskills on this i
■■
■■//ingredient 2
■■InsertIngredient(1,"Pickaxe");//you can insert multiple ingredients this way
■■InsertIngredient(1,"SledgeHammer");//you can insert multiple ingredients this way
■■InsertIngredient(1,"Hammer");//you can insert multiple ingredients this way
■■
■■m_IngredientAddHealth[1] = -4;// 0 = do nothing
■■m_IngredientSetHealth[1] = -1; // -1 = do nothing
■■m_IngredientAddQuantity[1] = 0;// 0 = do nothing
■■m_IngredientDestroy[1] = false;// false = do nothing
■■m_IngredientUseSoftSkills[1] = true;// set 'true' to allow modification of the values by softskills on this in
■■//----------------------------------------------------------------------------------------------------------------------
■■
■■//result1
■■AddResult("SmallStone");//add results here

■■m_ResultSetFullQuantity[0] = false;//true = set full quantity, false = do nothing
■■m_ResultSetQuantity[0] = -1;//-1 = do nothing
■■m_ResultSetHealth[0] = -1;//-1 = do nothing
■■m_ResultInheritsHealth[0] = -1;// (value) == -1 means do nothing; a (value) >= 0 means this result will i
■■m_ResultInheritsColor[0] = -1;// (value) == -1 means do nothing; a (value) >= 0 means this result class
■■m_ResultToInventory[0] = -2;//(value) == -2 spawn result on the ground;(value) == -1 place anywhere i
■■m_ResultUseSoftSkills[0] = false;// set 'true' to allow modification of the values by softskills on this resu
■■m_ResultReplacesIngredient[0] = -1;// value == -1 means do nothing; a value >= 0 means this result wi
■■
```

```
// ---------------------------------------------
// ------ SpookyArea.c - START --------------------
// ---------------------------------------------


// The base area for Spooky land, spooky particles and spooky triggers for a spooky halloween
class SpookyArea : EffectArea
{
■// ---------------------------------------------
■// ■■■■INITIAL SETUP
■// ---------------------------------------------
■
■override void SetupZoneData(  EffectAreaParams params )
■{
■■super.SetupZoneData( params );
■}
■
■override void EEInit()
■{
■■// We make sure we have the particle array
■■if ( !m_ToxicClouds )
■■■m_ToxicClouds = new array<Particle>;
■■
■■SetSynchDirty();
■■
■■#ifdef DEVELOPER
■■// Debugs when placing entity by hand using internal tools
■■if ( GetGame().IsServer() && !GetGame().IsMultiplayer() )
■■{
■■■Debug.Log("YOU CAN IGNORE THE FOLLOWING DUMP");
■■■InitZone();
■■■Debug.Log("YOU CAN USE FOLLOWING DATA PROPERLY");
■■}
■■#endif
■■
■■if ( GetGame().IsClient() && GetGame().IsMultiplayer() )
■■■InitZone();
■■
■■super.EEInit();
■}
■
■// We spawn particles and setup trigger
■override void InitZone()
■{
■■super.InitZone();
■}
■
■override void InitZoneServer()
■{
■■super.InitZoneServer();
■■
■■// We create the trigger on server
■■if ( m_TriggerType != "" )
■■■CreateTrigger( m_Position, m_Radius );
■}
■
■override void InitZoneClient()
■{
■■super.InitZoneClient();
■■
■■// We spawn VFX on client
■■PlaceParticles( GetWorldPosition(), m_Radius, m_InnerRings, m_InnerSpacing, m_OuterRingToggle,
■}
```

```
// ---------------------------------------------
// ------ SpookyTrigger.c - START --------------------
// ---------------------------------------------


// Halloween related, create a spooky ambiance in designated areas
class SpookyTrigger extends EffectTrigger
{
■
}


// ---------------------------------------------
// ------ SpookyTrigger.c - END ----------------------
// ---------------------------------------------


// ---------------------------------------------
// ------ SportGlasses_ColorBase.c - START --------------------
// ---------------------------------------------


class SportGlasses_ColorBase extends Clothing
{
■override bool CanPutAsAttachment(EntityAI parent)
■{
■■if (!super.CanPutAsAttachment(parent))
■■■return false;
■■
■■Clothing mask = Clothing.Cast(parent.FindAttachmentBySlotName("Mask"));
■■if (mask && mask.ConfigGetBool("noEyewear"))
■■{
■■■return false;
■■}
■■
■■return true;
■}
};

class SportGlasses_Black extends SportGlasses_ColorBase
{
■override int GetGlassesEffectID()
■{
■■return PPERequesterBank.REQ_GLASSESSPORTBLACK;
■}
};

class SportGlasses_Blue extends SportGlasses_ColorBase
{
■override int GetGlassesEffectID()
■{
■■return PPERequesterBank.REQ_GLASSESSPORTBLUE;
■}
};

class SportGlasses_Green extends SportGlasses_ColorBase
{
■override int GetGlassesEffectID()
■{
■■return PPERequesterBank.REQ_GLASSESSPORTGREEN;
■}
};
```

```
// ---------------------------------------------
// ------ Spotlight.c - START --------------------
// ---------------------------------------------


class Spotlight extends ItemBase
{
■private bool■m_IsFolded;
■private bool ■m_EvaluateDeployment;
■SpotlightLight ■m_Light;
■
■static vector ■m_LightLocalPosition;
■static vector ■m_LightLocalOrientation = "0 0 0";
■
■// Spotlight can be extended and compressed
■static const string SEL_REFLECTOR_COMP_U ■■= "reflector";
■static const string SEL_CORD_FOLDED_U ■■■= "cord_folded";
■static const string SEL_CORD_PLUGGED_U ■■■= "cord_plugged";
■static const string SEL_CORD_PLUGGED_F ■■■= "spotlight_folded_cord_plugged";
■static const string SEL_CORD_FOLDED_F ■■■= "spotlight_folded_cord_folded";
■
■static const string SEL_INVENTORY ■■■■= "inventory";
■static const string SEL_PLACING ■■■■= "placing";
■static const string SEL_GLASS_F ■■■■= "glass_folded";
■static const string SEL_GLASS_U ■■■■= "glass_unfolded";
■static const string SEL_REFLECTOR_F■■■■= "reflector_folded";
■static const string SEL_REFLECTOR_U■■■■= "reflector_unfolded";
■
■static const int ■ID_GLASS_UNFOLDED■■■= 3;
■static const int ■ID_REFLECTOR_UNFOLDED■■= 4;
■static const int ■ID_GLASS_FOLDED■■■■= 5;
■static const int ■ID_REFLECTOR_FOLDED■■■= 6;
■
■static string ■■LIGHT_OFF_GLASS ■■■= "dz\\gear\\camping\\Data\\spotlight_glass.rvmat";
■static string ■■LIGHT_OFF_REFLECTOR ■■= "dz\\gear\\camping\\Data\\spotlight.rvmat";
■static string ■■LIGHT_ON_GLASS ■■■■= "dz\\gear\\camping\\Data\\spotlight_glass_on.rvmat";
■static string ■■LIGHT_ON_REFLECTOR ■■■= "dz\\gear\\camping\\Data\\spotlight_glass_on.rvmat";
■
■//sound
■const string ■■■■SOUND_TURN_ON■■= "spotlight_turn_on_SoundSet";
■const string ■■■■SOUND_TURN_OFF■■= "spotlight_turn_off_SoundSet";
■
■protected EffectSound ■m_SoundTurnOn;
■protected EffectSound ■m_SoundTurnOff;
■protected EffectSound ■m_DeployLoopSound;

■//Spotlight, folded and unfolded
■void Spotlight()
■{
■■m_EvaluateDeployment = false;

■■RegisterNetSyncVariableBool("m_IsSoundSynchRemote");
■■RegisterNetSyncVariableBool("m_IsDeploySound");
■■RegisterNetSyncVariableBool("m_IsFolded");
■}
■
■void ~Spotlight()
■{
■■if (m_DeployLoopSound)
■■■SEffectManager.DestroyEffect(m_DeployLoopSound);
■}
■
■override bool IsElectricAppliance()
```

```
// ---------------------------------------------
// ------ SpotLightBase.c - START --------------------
// ---------------------------------------------


class SpotLightBase extends ScriptedLightBase
{
■void SpotLightBase()
■{
■■SetLightType(LightSourceType.SpotLight); // This function must be called in constructor of the light!
■}
}


// ---------------------------------------------
// ------ SpotLightBase.c - END ----------------------
// ---------------------------------------------


// ---------------------------------------------
// ------ SpotlightLight.c - START --------------------
// ---------------------------------------------


class SpotlightLight extends SpotLightBase
{
■void SpotlightLight()
■{
■■SetVisibleDuringDaylight(true);
■■SetRadiusTo(60);
■■SetSpotLightAngle(140);
■■SetCastShadow(true);
■■SetBrightnessTo(20.0);
■■SetFadeOutTime(0.3);
■■FadeIn(0.2);
■■SetAmbientColor(0.8, 0.9, 1.0);
■■SetDiffuseColor(0.8, 0.9, 1.0);
■}
}


// ---------------------------------------------
// ------ SpotlightLight.c - END ----------------------
// ---------------------------------------------


// ---------------------------------------------
// ------ SqfDebugWatcher.c - START --------------------
// ---------------------------------------------


class SqfDebugWatcher
{
■private int m_Id;
■private string m_SqfCommand;
■
■private bool m_IsRunning;
■■

■void SqfDebugWatcher( int id, string sqf_command )
■{
■■m_Id = id;
■■m_SqfCommand = sqf_command;
```

```
// ---------------------------------------------
// ------ SSG82.c - START --------------------
// ---------------------------------------------


class SSG82_Base : BoltActionRifle_ExternalMagazine_Base
{
■override bool CanEnterIronsights()
■{
■■return false;
■}
■
■override void AssembleGun()
■{
■■super.AssembleGun();
■■
■■if ( !FindAttachmentBySlotName("weaponOpticsAug") )
■■{
■■■GetInventory().CreateAttachment("SSG82Optic");
■■}
■}
■
■override RecoilBase SpawnRecoilObject()
■{
■■return new SSG82Recoil(this);
■}
■■■
■//Debug menu Spawn Ground Special
■/*override void OnDebugSpawn()
■{
■■super.OnDebugSpawn();
■}*/
};


class SSG82Optic: ItemOptics
{
■override bool CanPutAsAttachment( EntityAI parent )
■{
■■return true;
■}
};


// ---------------------------------------------
// ------ SSG82.c - END ----------------------
// ---------------------------------------------


// ---------------------------------------------
// ------ SSG82Recoil.c - START --------------------
// ---------------------------------------------


class SSG82Recoil: RecoilBase
{
■override void Init()
■{
■■vector point_1;
■■vector point_2;
■■vector point_3;
■■vector point_4;
■■point_1[0] = m_Player.GetRandomGeneratorSyncManager().GetRandomInRange(RandomGeneratorS
// ---------------------------------------------
// ------ Ssh68Helmet.c - START --------------------
// ---------------------------------------------


class Ssh68Helmet extends HelmetBase
{
}


// ---------------------------------------------
// ------ Ssh68Helmet.c - END ----------------------
// ---------------------------------------------


// ---------------------------------------------
// ------ StaminaEvents.c - START --------------------
// ---------------------------------------------


class StaminaSoundEventBase extends PlayerSoundEventBase
{
■const int MAX_VOLUME = 4;
■void StaminaSoundEventBase()
■{
■■m_Type = EPlayerSoundEventType.STAMINA;
■■m_HasPriorityOverTypes = EPlayerSoundEventType.STAMINA;
■}
■
■override bool HasPriorityOverCurrent(PlayerBase player, EPlayerSoundEventID other_state_id, EPlayer
■{
■■return false;
■}
■
■override void OnPlay(PlayerBase player)
■{
■■super.OnPlay(player);
■■
■■MaskBase mask = MaskBase.Cast(player.GetInventory().FindAttachment(InventorySlots.GetSlotIdFron
■■
■■float rel_volume = 1;
■■if (mask)
■■{
■■■if (mask.IsExternalFilterAttached() || mask.HasIntegratedFilter())
■■■{
■■■■rel_volume = Math.Lerp(MAX_VOLUME,1,mask.GetFilterQuantity01());
■■■}
■■}
■■
■■if (m_SoundSetCallback)
■■■m_SoundSetCallback.SetVolumeRelative(rel_volume);
■■
■■#ifdef DIAG_DEVELOPER
■■if (!DiagMenu.GetBool(DiagMenuIDs.MISC_BREATH_VAPOR_LVL))//disabled in debug, do not spawn
■■■return;
■■#endif
■■
■■if ( player.CanSpawnBreathVaporEffect() )
■■■player.SpawnBreathVaporEffect();
■■
■■
■■//!!! beware of the ifdef above , most likely you want to put your code above it, otherwise it might not ge
■■
```

```
// ---------------------------------------------
// ------ StaminaHandler.c - START --------------------
// ---------------------------------------------


enum EStaminaMultiplierTypes
{
■MASK = 1,
■FATIGUE,
■EPINEPHRINE,
■DROWNING,
}


/**@class■Stamina Consumer
 * @brief■Holds information about Stamina Consumer
 *
 * @param[in]■threshold■value needed to allow consume stamina
 * @param[in]■state■■keeps state of the consumer non-depleted/depleted
 */
class StaminaConsumer
{
■protected float m_ActivationThreshold;
■protected float m_DrainThreshold
■protected bool m_State;
■
■void StaminaConsumer(float threshold, float threshold2, bool state)
■{
■■m_ActivationThreshold = threshold; //can be activated if above this threshold
■■m_DrainThreshold = threshold2; //can continually drain until it reaches this threshold
■■m_State = state;
■}
■
■bool GetState() { return m_State; }
■void SetState(bool state) { m_State = state; }
■
■float GetActivationThreshold() { return m_ActivationThreshold; }
■void SetActivationThreshold(float threshold) { m_ActivationThreshold = threshold; }
■
■float GetDrainThreshold() { return m_DrainThreshold; }
■void SetDrainThreshold(float threshold) { m_DrainThreshold = threshold; }■
}

class StaminaConsumers
{
■protected ref map<EStaminaConsumers, ref StaminaConsumer> m_StaminaConsumers;

■void StaminaConsumers()
■{
■■m_StaminaConsumers = new map<EStaminaConsumers, ref StaminaConsumer>;
■}

■void RegisterConsumer(EStaminaConsumers consumer, float threshold, float depletion_threshold = -1)
■{
■■if (depletion_threshold == -1)
■■{
■■■depletion_threshold = threshold;
■■}
■■
■■if ( !m_StaminaConsumers.Contains(consumer) )
■■{
■■■//! init of StaminaConsumer - threshold, state
■■■StaminaConsumer sc = new StaminaConsumer(threshold, depletion_threshold, true);
```

```
// ----------------------------------------------
// ------ StaminaSoundHandler.c - START --------------------
// ----------------------------------------------


const float TICK_INTERVAL = 1;

const float STAMINA_SOUND_TR1 = 0.55;
const float STAMINA_SOUND_TR2 = 0.25;

enum eStaminaZones
{
■ZONE0,
■ZONE1,
■ZONE2,
}

enum eStaminaTendency
{
■UP,
■DOWN,
}

enum eStaminaState
{
■ZONE0_UP = 1,
■ZONE0_DOWN,
■ZONE1_UP,
■ZONE1_DOWN,
■ZONE2_UP,
■ZONE2_DOWN,
■//count bellow, add above
■COUNT
}


class SoundHandlerBase
{
■eSoundHandlers m_Id;
■PlayerBase m_Player;
■
■void Update();
■
■eSoundHandlers GetID()
■{
■■return m_Id;
■}
■
■void SoundHandlerBase(PlayerBase player)
■{
■■m_Player = player;
■■Init();
■}
■
■void Init();
}

class StaminaSoundHandlerBase extends SoundHandlerBase
{

■override void Init()
■{
■■m_Id = eSoundHandlers.STAMINA;
```

```
// ---------------------------------------------
// ------ StanceIndicator.c - START --------------------
// ---------------------------------------------


class StanceIndicator
{■
■protected ref HumanMovementState■m_State;
■protected PlayerBase■■■■m_Player;
■
■void StanceIndicator(PlayerBase player)
■{
■■m_State = new HumanMovementState();
■■m_Player = player;
■}
■
■void Update()
■{
■■if ( m_Player )
■■{
■■■m_Player.GetMovementState(m_State);
■■■int player_stance = m_State.m_iStanceIdx;
■■■int hud_stance_id = 1;
■■■//if ( player_stance == DayZPlayerConstants.STANCEIDX_ERECT  || player_stance == DayZPlayerC
■■■if ( player_stance == DayZPlayerConstants.STANCEIDX_CROUCH || player_stance == DayZPlayerC
■■■{
■■■■hud_stance_id = 2;
■■■}
■■■if ( player_stance == DayZPlayerConstants.STANCEIDX_PRONE  || player_stance == DayZPlayerCo
■■■{
■■■■hud_stance_id = 3;
■■■}
■■■DisplayStance(hud_stance_id);
■■■//Debug
■■■//m_Player.MessageStatus(ToString(player) + "StanceIndicator.c || stance: " + ToString(player_stanc
■■}
■}
■
■void DisplayStance(int stance)
■{■
■■if ( m_Player )
■■{
■■■DisplayElementBase stance_element = m_Player.GetVirtualHud().GetElement(eDisplayElements.DE
■■■if(stance_element)
■■■{
■■■■stance_element.SetValue(stance);
■■■}
■■■//m_Player.GetVirtualHud().SetValue(eDisplayElements.DELM_STANCE, stance);
■■}
■}
};


// ---------------------------------------------
// ------ StanceIndicator.c - END ----------------------
// ---------------------------------------------
// ---------------------------------------------
// ------ StartKitIV.c - START --------------------
// ---------------------------------------------


class StartKitIV: Inventory_Base {};



// ---------------------------------------------
// ------ StartKitIV.c - END ----------------------
// ---------------------------------------------


// ---------------------------------------------
// ------ StartupMenu.c - START --------------------
// ---------------------------------------------


class StartupMenu extends UIScriptedMenu
{
■void StartupMenu()
■{
■}

■void ~StartupMenu()
■{
■}

■override Widget Init()
■{
■■layoutRoot = GetGame().GetWorkspace().CreateWidgets("gui/layouts/startup.layout");
■■
■■m_label = TextWidget.Cast( layoutRoot.FindAnyWidget("TextWidget") );

■■return layoutRoot;
■}

■TextWidget m_label;
}



// ---------------------------------------------
// ------ StartupMenu.c - END ----------------------
// ---------------------------------------------


// ---------------------------------------------
// ------ StatDebugObject.c - START --------------------
// ---------------------------------------------


class StatDebugObject
{
■string m_Name;
■float m_Value;
■eRemoteDebugType m_Type;
■
■void StatDebugObject(string name, float value, eRemoteDebugType type)
■{
■■m_Name = name;
■■m_Value = value;
```

```
// ---------------------------------------------
// ------ StateBase.c - START --------------------
// ---------------------------------------------


class SymptomBase
{
■const float MAX_TIME_ACTIVE_SAVEGUARD = 20;
■int m_Priority;
■SoundOnVehicle m_SoundObject;
■bool m_PlayedSound;
■bool m_IsActivated;
■PlayerBase m_Player;
■float m_ServerUpdateInterval = 1;
■float m_ServerUpdateDelta;
■bool m_IsTemplate = true;
■float m_ActivatedTime;
■int m_ID;//ID for the type of Symptom
■int m_UID;//unique ID
■bool m_IsClientOnly;
■bool m_DestroyOnAnimFinish = true;
■bool m_DestroyRequested = false;
■int m_SymptomType = -1;
■bool m_IsPersistent = false;
■SymptomManager m_Manager;
■bool m_SyncToClient = false;
■float m_Duration;
■bool m_AnimPlayRequested;
■int m_MaxCount = -1;//how many symptoms of this type can be queued up at the same time, '-1' for unli

■SymptomCB m_AnimCallback;
■
■ref array<Param> m_PersistentParams = new array<Param>;
■
■void SymptomBase()
■{
■}
■
■void ~SymptomBase()
■{

■}

■void Init(SymptomManager manager, PlayerBase player, int uid)
■{
■■m_Manager = manager;
■■m_Player = player;
■■m_UID = uid;
■■m_IsTemplate = false;
■■OnInit();
■}
■
■int GetMaxCount()
■{
■■return m_MaxCount;
■}
■
■int GetUID()
■{
■■return m_UID;
■}

■void OnOwnerKilled()
```

```
// ---------------------------------------------
// ------ StateCB.c - START --------------------
// ---------------------------------------------


class SymptomCB extends HumanCommandActionCallback
{
■//int m_SymptomUID;
■float m_RunTime;
■float m_StartingTime;
■PlayerBase m_Player;
■
■override void OnFinish(bool pCanceled)
■{
■■if( m_Player && m_Player.GetSymptomManager())
■■{
■■■m_Player.GetSymptomManager().OnAnimationFinished();
■■}
■}
■
■
■void Init(float run_time, PlayerBase player)
■{
■■EnableCancelCondition(true);
■■m_RunTime = run_time * 1000;
■■m_StartingTime = GetGame().GetTime();
■■m_Player = player;
■■
■■if( m_Player && m_Player.GetSymptomManager())
■■{
■■■m_Player.GetSymptomManager().OnAnimationStarted();
■■}
■}
■
■bool CancelCondition()
■{
■■if(m_RunTime > 0 && (GetGame().GetTime() > m_StartingTime + m_RunTime))
■■{
■■■return true;
■■}
■■else
■■{
■■■return false;
■■}
■}

■override bool IsSymptomCallback()
■{
■■return true;
■}
■
};



// ---------------------------------------------
// ------ StateCB.c - END ----------------------
// ---------------------------------------------
// ---------------------------------------------
// ------ StateManager.c - START --------------------
// ---------------------------------------------


enum AnimType
{
■FULL_BODY = 1,
■ADDITIVE,
}

enum SymptomIDs
{
■SYMPTOM_COUGH = 1,
■SYMPTOM_VOMIT,
■SYMPTOM_BLINDNESS,
■SYMPTOM_BULLET_HIT,
■SYMPTOM_BLEEDING_SOURCE,
■SYMPTOM_BLOODLOSS,
■SYMPTOM_SNEEZE,
■SYMPTOM_FEVERBLUR,
■SYMPTOM_LAUGHTER,
■SYMPTOM_UNCONSCIOUS,
■SYMPTOM_FREEZE,
■SYMPTOM_HOT,
■SYMPTOM_PAIN_LIGHT,
■SYMPTOM_PAIN_HEAVY,
■SYMPTOM_HAND_SHIVER,
};

enum SymptomTypes
{
■PRIMARY,
■SECONDARY
};

enum EAnimPlayState
{
■OK,
■POSTPONED,
■FAILED,
};

const int DEBUG_PADDING_OFFSET = 2;
const int MAX_QUEUE_SIZE = 5;

class SymptomManager
{
■PlayerBase m_Player;
■ref map<int, ref SymptomBase> m_AvailableSymptoms;
■ref map<int, int> m_ActiveSymptomTypes;//for each type(symptom id), keep track of how many times it i
■
■ref array<ref SymptomBase> m_SymptomQueuePrimary;
■ref array<ref SymptomBase> m_SymptomQueueSecondary;

■ref map<int , SymptomBase > m_SymptomsUIDs;

■
■ref array<ref Param> m_SymptomQueueServerDbg;
■ref array<ref Param> m_SymptomQueueServerDbgPrimary;
■ref array<ref Param> m_SymptomQueueServerDbgSecondary;
■//ref array<string> m_SymptomQueueSecondaryServerDbg;
■ref Timer m_Timer;
```

```
// ----------------------------------------------
// ------ staticDefinesDoc.c - START --------------------
// ----------------------------------------------


#ifdef DOXYGEN
/**
| %Defines                                                  | Short description
| --------------------------------------------------------------------- | -------------------------------------------------------------------
| DEVELOPER                                                 | Internal developer version
| RELEASE                                                   | Retail build of the game
| PREVIEW_BUILD                                             | Preview build. Currently used only in console build
| PLATFORM_WINDOWS \|\| PLATFORM_MACOSX \|\| ENF_LINUX               | Platform specific defines
| PLATFORM_CONSOLE && (PLATFORM_XBOX \|\| PLATFORM_PS4)              | Platform specific defin
| X1_TODO_TEMP_GUI                                          | Temporary XBOX GUI hacks
| BULDOZER                                                  | Game is launched in land editor mode (-buldozer)
| SERVER                                                    | Game runs as server
| NO_GUI, NO_GUI_INGAME                                     | Game runs without gui - Server, client simula
| WORKBENCH                                                 | Game runs from workbench
| COMPONENT_SYSTEM                                          | Enfusion entity component system
| GAME_TEMPLATE                                             | Game template implementation see [link](https://
| ENF_DONE                                                  | Looks like commented out wip or dead code
| DOXYGEN                                                   | Doxygen documentation, never compiled
| _DAYZ_CREATURE_DEBUG_SHADOW                               | Game is built with network debug fo
| BOT                                                       | ? Script player debug bot ?
| PS3                                                       | ? obsolete ?

| PC builds                  | %Defines used
| ---------------------------- | ------------------------------------------------------------------------------------------------------
| release, releaseAsserts      | DEVELOPER, (PLATFORM_WINDOWS \|\| PLATFORM_MACOSX \|\| ENF
| retail client                | RELEASE, (PLATFORM_WINDOWS \|\| PLATFORM_MACOSX \|\| ENF_LINUX),
| retail server, profile server | RELEASE, (PLATFORM_WINDOWS \|\| PLATFORM_MACOSX \|\| ENF_LIN

| Console builds | %Defines used                                                    |
| ------------- | ---------------------------------------------------------------------- |
| preview       | RELEASE, PLATFORM_CONSOLE, (PLATFORM_XBOX \|\| PLATFORM_PS4), PREVIEW
| retail        | RELEASE, PLATFORM_CONSOLE, (PLATFORM_XBOX \|\| PLATFORM_PS4)             |
| release       | DEVELOPER, PLATFORM_CONSOLE, (PLATFORM_XBOX \|\| PLATFORM_PS4)
*/

enum StaticDefines
{
■DEVELOPER,
■RELEASE,
■PREVIEW_BUILD,
■PLATFORM_WINDOWS,
■PLATFORM_MACOSX,
■ENF_LINUX,
■PLATFORM_CONSOLE,
■PLATFORM_XBOX,
■PLATFORM_PS4,
■X1_TODO_TEMP_GUI,
■BULDOZER,
■SERVER,
■NO_GUI,
■NO_GUI_INGAME,
■WORKBENCH,
■GAME_TEMPLATE,
■ENF_DONE,
■COMPONENT_SYSTEM,
■DOXYGEN,
■_DAYZ_CREATURE_DEBUG_SHADOW,
■BOT,
// ---------------------------------------------
// ------ StaticGUIUtils.c - START --------------------
// ---------------------------------------------


class StaticGUIUtils
{
■static const int IMAGESETGROUP_INVENTORY = 0;
■
■
■//! Checks for improperly formated, legacy image names and corrects them to default format.
■static string VerifyIconImageString(int imageset_group = IMAGESETGROUP_INVENTORY, string icon_
■{
■■if (icon_name == "")
■■{
■■■return "set:dayz_inventory image:missing";
■■}
■■
■■if ( !icon_name.Contains("image:") )
■■{
■■■switch (imageset_group)
■■■{
■■■■case IMAGESETGROUP_INVENTORY:
■■■■■return "set:dayz_inventory image:" + icon_name;
■■■}
■■■
■■}
■■return icon_name;
■}
}


// ---------------------------------------------
// ------ StaticGUIUtils.c - END ----------------------
// ---------------------------------------------


// ---------------------------------------------
// ------ StaticObj_Roadblock_Wood_Small.c - START --------------------
// ---------------------------------------------


class StaticObj_Roadblock_Wood_Small: House
{■■
■PointLightBase m_BlinkingLight;
■
■override void EEInit()
■{
■■super.EEInit();
■
■■if ( !GetGame().IsDedicatedServer() )
■■{
■■■m_BlinkingLight = EntranceLight.Cast(ScriptedLightBase.CreateLightAtObjMemoryPoint(Roadblock_
■■}
■}
■
■override void EEDelete(EntityAI parent)
■{
■■super.EEDelete(parent);
■■
■■if ( !GetGame().IsDedicatedServer() )
■■{
■■■if ( m_BlinkingLight )
```

```
// ---------------------------------------------
// ------ SteakKnife.c - START --------------------
// ---------------------------------------------


class SteakKnife extends ToolBase
{
■override bool IsMeleeFinisher()
■{
■■return true;
■}
■
■override array<int> GetValidFinishers()
■{
■■return {EMeleeHitType.FINISHER_LIVERSTAB,EMeleeHitType.FINISHER_NECKSTAB};
■}
■
■override void SetActions()
■{
■■super.SetActions();
■■
■■AddAction(ActionBurnSewTarget);
■■AddAction(ActionUnrestrainTarget);
■■AddAction(ActionSkinning);
■■AddAction(ActionMineBush);
■■AddAction(ActionMineTreeBark);
■■AddAction(ActionBurnSewSelf);
■■AddAction(ActionDigWorms);
■■AddAction(ActionShaveTarget);
■■AddAction(ActionDisarmMine);
■■AddAction(ActionDisarmExplosive);
■■AddAction(ActionShave);
■}
}


// ---------------------------------------------
// ------ SteakKnife.c - END ----------------------
// ---------------------------------------------


// ---------------------------------------------
// ------ Stomach.c - START --------------------
// ---------------------------------------------


class StomachMdfr: ModifierBase
{
■override void Init()
■{
■■m_TrackActivatedTime = false;
■■m_ID ■■■■■= eModifiers.MDF_STOMACH;
■■m_TickIntervalInactive ■= DEFAULT_TICK_TIME_INACTIVE;
■■m_TickIntervalActive ■= DEFAULT_TICK_TIME_ACTIVE;
■■
■■DisableDeactivateCheck();
■}

■override bool ActivateCondition(PlayerBase player)
■{
■■return true;
■}
```

```
// ---------------------------------------------
// ------ Stone.c - START --------------------
// ---------------------------------------------


class Stone extends ItemBase
{
■override void SetActions()
■{
■■super.SetActions();

■■AddAction(ActionAttach);
■■AddAction(ActionDetach);
■}
}


// ---------------------------------------------
// ------ Stone.c - END ----------------------
// ---------------------------------------------


// ---------------------------------------------
// ------ StoneKnife.c - START --------------------
// ---------------------------------------------


class StoneKnife extends ToolBase
{
■override bool IsMeleeFinisher()
■{
■■return true;
■}
■
■override array<int> GetValidFinishers()
■{
■■return {EMeleeHitType.FINISHER_LIVERSTAB,EMeleeHitType.FINISHER_NECKSTAB};
■}
■
■override void SetActions()
■{
■■super.SetActions();
■■
■■AddAction(ActionUnrestrainTarget);
■■AddAction(ActionSkinning);
■■AddAction(ActionMineBush);
■■AddAction(ActionMineTreeBark);
■■AddAction(ActionDigWorms);
■}
}


// ---------------------------------------------
// ------ StoneKnife.c - END ----------------------
// ---------------------------------------------


// ---------------------------------------------
// ------ StoveLight.c - START --------------------
// ---------------------------------------------


class StoveLight extends PointLightBase
```

```
// ---------------------------------------------
// ------ StuffedNotfr.c - START --------------------
// ---------------------------------------------


class StuffedNotfr: NotifierBase
{
■void StuffedNotfr(NotifiersManager manager)
■{
■■m_Active = true;
■}

■override int GetNotifierType()
■{
■■return eNotifiers.NTF_STUFFED;
■}

■override void DisplayBadge()
■{
■■float volume = m_Player.m_PlayerStomach.GetStomachVolume();
■■eBadgeLevel lvl = DetermineBadgeLevel(volume, PlayerConstants.BT_STOMACH_VOLUME_LVL3, P
■■DisplayElementBadge dis_elm = DisplayElementBadge.Cast(GetVirtualHud().GetElement(eDisplayEle
■■
■■if( dis_elm )
■■{
■■■dis_elm.SetLevel(lvl);
■■}
■}

■override void HideBadge()
■{
■■DisplayElementBadge dis_elm = DisplayElementBadge.Cast(GetVirtualHud().GetElement(eDisplayEle
■■
■■if( dis_elm )
■■{
■■■dis_elm.SetLevel(eBadgeLevel.NONE);
■■}
■}
};


// ---------------------------------------------
// ------ StuffedNotfr.c - END ----------------------
// ---------------------------------------------


// ---------------------------------------------
// ------ StuffedStomach.c - START --------------------
// ---------------------------------------------


class StuffedStomachMdfr: ModifierBase
{
■protected const int■ ■STUFFED_TRESHOLD = PlayerConstants.BT_STOMACH_VOLUME_LVL3;
■override void Init()
■{
■■m_TrackActivatedTime = false;
■■m_ID ■■■■■= eModifiers.MDF_STUFFED;
■■m_TickIntervalInactive ■= 1;
■■m_TickIntervalActive ■= DEFAULT_TICK_TIME_ACTIVE;
■}
■
■override void OnTick(PlayerBase player, float deltaT)
```

```
// ---------------------------------------------
// ------ StunBaton.c - START --------------------
// ---------------------------------------------


class StunBaton : ItemBase
{
■protected int m_MeleeMode;
■protected int m_MeleeHeavyMode;
■protected int m_MeleeSprintMode;

■void StunBaton()
■{
■■m_MeleeMode = 0;
■■m_MeleeHeavyMode = 1;
■■m_MeleeSprintMode = 2;
■}

■override void OnWorkStart()
■{
■■//! melee modes in cfg (switched on state)
■■m_MeleeMode = 3;
■■m_MeleeHeavyMode = 4;
■■m_MeleeSprintMode = 5;
■}
■
■override void OnWorkStop()
■{
■■//! melee modes in cfg (switched off state)
■■m_MeleeMode = 0;
■■m_MeleeHeavyMode = 1;
■■m_MeleeSprintMode = 2;
■}
■
■override int GetMeleeMode()
■{
■■return m_MeleeMode;
■}

■override int GetMeleeHeavyMode()
■{
■■return m_MeleeHeavyMode;
■}
■
■override int GetMeleeSprintMode()
■{
■■return m_MeleeSprintMode;
■}
■
■override void SetActions()
■{
■■super.SetActions();
■■AddAction(ActionTurnOnWhileInHands);
■■AddAction(ActionTurnOffWhileInHands);
■}
}


// ---------------------------------------------
// ------ StunBaton.c - END ----------------------
// ---------------------------------------------
// ---------------------------------------------
// ------ SuppressorBase.c - START --------------------
// ---------------------------------------------


class SuppresorBase extends ItemBase
{
}


// ---------------------------------------------
// ------ SuppressorBase.c - END ----------------------
// ---------------------------------------------


// ---------------------------------------------
// ------ Surface.c - START --------------------
// ---------------------------------------------


class Surface
{
■static float GetParamFloat(string surface_name, string param_name)
■{
■■return GetGame().ConfigGetFloat("CfgSurfaces " + surface_name + " " + param_name);
■}
■
■static bool AllowedWaterSurface(float pHeight, string pSurface, array<string> pAllowedSurfaceList)
■{
■■if (pSurface)
■■{
■■■pSurface.Replace("_ext", "");
■■■pSurface.Replace("_int", "");
■■}

■■bool isSeaCheck = false;
■■
■■foreach (string allowedSurface : pAllowedSurfaceList)
■■{
■■■if (pSurface == "" && allowedSurface == UAWaterType.SEA)
■■■■isSeaCheck = pHeight <= g_Game.SurfaceGetSeaLevel() + 0.001;

■■■if (isSeaCheck || allowedSurface == pSurface)
■■■■return true;
■■}
■■
■■return false;
■}
}


// ---------------------------------------------
// ------ Surface.c - END ----------------------
// ---------------------------------------------


// ---------------------------------------------
// ------ SurgicalGloves_ColorBase.c - START --------------------
// ---------------------------------------------


class SurgicalGloves_ColorBase extends Clothing {};
class SurgicalGloves_Blue extends SurgicalGloves_ColorBase {};
```

```
// ---------------------------------------------
// ------ SurgicalMask.c - START --------------------
// ---------------------------------------------


class SurgicalMask extends Clothing
{
■override void SetActions()
■{
■■super.SetActions();
■■AddAction(ActionWringClothes);
■}
};


// ---------------------------------------------
// ------ SurgicalMask.c - END ----------------------
// ---------------------------------------------


// ---------------------------------------------
// ------ SurrenderDummyItem.c - START --------------------
// ---------------------------------------------


class SurrenderDummyItem extends DummyItem
{
■protected PlayerBase m_player;
■
■void ~SurrenderDummyItem()
■{
■■/*m_player = PlayerBase.Cast(GetHierarchyRootPlayer());
■■if (m_player && m_player.m_EmoteManager)
■■■PlayerBase.Cast(m_player).m_EmoteManager.PlaySurrenderInOut(false); //has to be triggered sepa
■■else
■■■Print("Error, no player owns  SurrenderDummyItem");*/
■}
■
■override void OnItemLocationChanged(EntityAI old_owner, EntityAI new_owner)
■{
■■super.OnItemLocationChanged(old_owner,new_owner);
■■
■■//Player somehow got a hold of the item (can be done by quickly pressing tab after F5), item should not
■■if (old_owner)
■■■Delete();
■■
■■/*
■■//enters player's inventory
■■if (new_owner)
■■{
■■■//m_player = PlayerBase.Cast(new_owner);
■■■//PlayerBase.Cast(new_owner).m_EmoteManager.PlaySurrenderInOut(true);
■■}
■■// may not work on item destroy?
■■else
■■{
■■■Delete();
■■■//PlayerBase.Cast(old_owner).m_EmoteManager.PlaySurrenderInOut(false);
■■}
■■*/
■}
}
```

```
// ---------------------------------------------
// ------ SurvivorBase.c - START --------------------
// ---------------------------------------------


class SurvivorBase extends PlayerBaseClient
{
■override int GetHideIconMask()
■{
■■return EInventoryIconVisibility.HIDE_VICINITY;
■}
}


// ---------------------------------------------
// ------ SurvivorBase.c - END ----------------------
// ---------------------------------------------


// ---------------------------------------------
// ------ SVD.c - START --------------------
// ---------------------------------------------


class SVD_Base : RifleBoltLock_Base
{
■override RecoilBase SpawnRecoilObject()
■{
■■return new SvdRecoil(this);
■}
■
■override bool CanEnterIronsights()
■{
■■ItemOptics optic = GetAttachedOptics();
■■if (optic && PSO1Optic.Cast(optic) || PSO11Optic.Cast(optic) || KashtanOptic.Cast(optic) || KazuarOpti
■■■return true;
■■return super.CanEnterIronsights();
■}
■■■
■//Debug menu Spawn Ground Special
■override void OnDebugSpawn()
■{
■■GameInventory inventory = GetInventory();

■■inventory.CreateInInventory( "PSO1Optic" );
■■inventory.CreateInInventory( "AK_Suppressor" );
■■inventory.CreateInInventory( "Battery9V" );
■■
■■SpawnAttachedMagazine("Mag_SVD_10Rnd");
■}
};


// ---------------------------------------------
// ------ SVD.c - END ---------------------
// ---------------------------------------------
// ---------------------------------------------
// ------ SvdRecoil.c - START --------------------
// ---------------------------------------------


class SvdRecoil: RecoilBase
{
■override void Init()
■{
■■vector point_1;
■■vector point_2;
■■vector point_3;
■■vector point_4;
■■point_1[0] = m_Player.GetRandomGeneratorSyncManager().GetRandomInRange(RandomGeneratorS
■■point_2[0] = m_Player.GetRandomGeneratorSyncManager().GetRandomInRange(RandomGeneratorS
■■point_3[0] = m_Player.GetRandomGeneratorSyncManager().GetRandomInRange(RandomGeneratorS
■■point_4[0] = m_Player.GetRandomGeneratorSyncManager().GetRandomInRange(RandomGeneratorS
■■m_HandsCurvePoints.Insert(point_1);//forms a 2 dimensional spline(z is ignored)
■■m_HandsCurvePoints.Insert(point_2);
■■m_HandsCurvePoints.Insert(point_3);
■■m_HandsCurvePoints.Insert(point_4);
■■m_HandsCurvePoints.Insert("0 0 0");
■■m_HandsOffsetRelativeTime = 1;
■■
■■m_MouseOffsetRangeMin = 45;//in degrees min
■■m_MouseOffsetRangeMax = 110;//in degrees max
■■m_MouseOffsetDistance = 1.8;//how far should the mouse travel
■■m_MouseOffsetRelativeTime = 0.5;//[0..1] a time it takes to move the mouse the required distance rela
■
■■m_CamOffsetDistance = 0.015;
■■m_CamOffsetRelativeTime = 1;
■}
}


// ---------------------------------------------
// ------ SvdRecoil.c - END ----------------------
// ---------------------------------------------


// ---------------------------------------------
// ------ SwarmingFlies.c - START --------------------
// ---------------------------------------------


class EffSwarmingFlies : EffectParticle
{
■void EffSwarmingFlies()
 ■{
■■SetParticleID(ParticleList.ENV_SWARMING_FLIES);
■}
}



// ---------------------------------------------
// ------ SwarmingFlies.c - END ----------------------
// ---------------------------------------------
// ---------------------------------------------
// ------ Sweater_ColorBase.c - START --------------------
// ---------------------------------------------


class Sweater_ColorBase extends Clothing
{
■override void SetActions()
■{
■■super.SetActions();
■■AddAction(ActionWringClothes);
■}
};
class Sweater_Gray extends Sweater_ColorBase {};
class Sweater_Blue extends Sweater_ColorBase {};
class Sweater_Green extends Sweater_ColorBase {};
class Sweater_Red extends Sweater_ColorBase {};


// ---------------------------------------------
// ------ Sweater_ColorBase.c - END ----------------------
// ---------------------------------------------


// ---------------------------------------------
// ------ Switchable_Base.c - START --------------------
// ---------------------------------------------


class Switchable_Base extends ItemBase
{
■override void OnWasAttached( EntityAI parent, int slot_id )
■{
■■super.OnWasAttached( parent, slot_id );
■■
■■ItemBase parent_item;
■■if ( Class.CastTo(parent_item,parent) )
■■{
■■■parent_item.AddLightSourceItem(this);
■■}
■}
■
■override void OnWasDetached( EntityAI parent, int slot_id )
■{
■■super.OnWasDetached( parent, slot_id );
■■
■■ItemBase parent_item;
■■if ( Class.CastTo(parent_item,parent) )
■■{
■■■parent_item.RemoveLightSourceItem();
■■}
■}
■
■override void EEItemLocationChanged (notnull InventoryLocation oldLoc, notnull InventoryLocation newL
■{
■■super.EEItemLocationChanged(oldLoc, newLoc);
■■
■■if (GetLight())
■■{
■■■GetLight().UpdateMode();
■■}
■}
}
```

```
// ---------------------------------------------
// ------ Sword.c - START --------------------
// ---------------------------------------------


class Sword extends ToolBase
{
■void Sword()
■{
■}

■override bool IsMeleeFinisher()
■{
■■return true;
■}
■
■override array<int> GetValidFinishers()
■{
■■return {EMeleeHitType.FINISHER_LIVERSTAB,EMeleeHitType.FINISHER_NECKSTAB};
■}
■
■override void SetActions()
■{
■■super.SetActions();
■■AddAction(ActionUnrestrainTarget);
■■AddAction(ActionMineBush);
■}
}


// ---------------------------------------------
// ------ Sword.c - END ----------------------
// ---------------------------------------------


// ---------------------------------------------
// ------ SymptomEvents.c - START --------------------
// ---------------------------------------------


class SymptomSoundEventBase extends PlayerSoundEventBase
{
■void SymptomSoundEventBase()
■{
■■m_Type = EPlayerSoundEventType.GENERAL;
■■m_HasPriorityOverTypes = -1;
■}
■
■override bool HasPriorityOverCurrent(PlayerBase player, EPlayerSoundEventID other_state_id, EPlayer
■{
■■return true;
■}
}

class CoughSoundEvent extends SymptomSoundEventBase
{
■void CoughSoundEvent()
■{
■■m_ID = EPlayerSoundEventID.SYMPTOM_COUGH;
■■m_SoundVoiceAnimEventClassID = 8;
■}
}
```

```
// ---------------------------------------------
// ------ SyncData.c - START --------------------
// ---------------------------------------------


class SyncData
{
■int m_SyncInt;
■ref SyncPlayerList m_ServerPlayerList;
■ref SyncEntityKillInfo m_EntityKill;
}


// ---------------------------------------------
// ------ SyncData.c - END ----------------------
// ---------------------------------------------


// ---------------------------------------------
// ------ SyncedValue.c - START --------------------
// ---------------------------------------------


class SyncedValue
{
■string m_Name;
■float m_Value;
■float m_ValueNorm;
■bool m_State;
■
■void SyncedValue( string name, float value, bool state, float val_norm )
■{
■■m_Name = name;
■■m_Value = value;
■■m_State = state;
■■m_ValueNorm = val_norm;
■}
■
■string GetName()
■{
■■return m_Name;
■}
■
■float GetValue()
■{
■■return m_Value;
■}■
■
■float GetValueNorm()
■{
■■return m_ValueNorm;
■}
■
■bool GetState()
■{
■■return m_State;
■}
}

class SyncedValueLevel
{
■string m_Name;
■float m_Value;
```

```
// ---------------------------------------------
// ------ SyncEntityKill.c - START --------------------
// ---------------------------------------------


class SyncEntityKillInfo
{
■EntityAI■m_EntityVictim;
■EntityAI■m_EntityKiller;
■EntityAI■m_EntitySource;
■bool■■m_IsHeadShot;
}


// ---------------------------------------------
// ------ SyncEntityKill.c - END ----------------------
// ---------------------------------------------


// ---------------------------------------------
// ------ SyncEvents.c - START --------------------
// ---------------------------------------------


class SyncEvents
{
■static void RegisterEvents()
■{
■■DayZGame dz_game = DayZGame.Cast( GetGame() );
■■
■■dz_game.Event_OnRPC.Insert( Event_OnRPC );
■■Print("SyncEvents -> RegisterEvents");
■}
■
■static void Event_OnRPC(PlayerIdentity sender, Object target, int rpc_type, ParamsReadContext ctx)
■{■■
■■if ( rpc_type == ERPCs.RPC_SYNC_EVENT && GetGame() && GetGame().IsMultiplayer() && GetGar
■■{■■■
■■■Param2<ESyncEvent, ref SyncData> event_data = new Param2<ESyncEvent, ref SyncData>( -1, nu
■■■■
■■■if ( ctx.Read( event_data ) )
■■■{
■■■■OnSyncEvent( event_data.param1, event_data.param2, target );
■■■}
■■}
■}
■
■static void OnSyncEvent( ESyncEvent event_type, SyncData data, Object target )
■{
■■switch ( event_type )
■■{
■■■case ESyncEvent.PlayerList:
■■■{
■■■■ClientData.SyncEvent_OnRecievedPlayerList( data.m_ServerPlayerList );
■■■■break;
■■■}
■■■case ESyncEvent.EntityKill:
■■■{
■■■■ClientData.SyncEvent_OnEntityKilled( data.m_EntityKill );
■■■■break;
■■■}
■■■case ESyncEvent.PlayerIgnateFireplayce:
■■■{
```

```
// ---------------------------------------------
// ------ SyncHitInfo.c - START --------------------
// ---------------------------------------------


//! extendable type to allow for smarter hit data syncing
class SyncHitInfo extends Managed
{
■int m_AnimType;
■float m_HitDir;
■float m_HealthDamage;
■bool m_Fullbody;
■//ref TotalDamageResult m_DamageResult;
■bool m_HasSource;
};


// ---------------------------------------------
// ------ SyncHitInfo.c - END ----------------------
// ---------------------------------------------


// ---------------------------------------------
// ------ SyncPlayer.c - START --------------------
// ---------------------------------------------


class SyncPlayer
{
■string m_UID;
■string m_PlayerName;
}


// ---------------------------------------------
// ------ SyncPlayer.c - END ----------------------
// ---------------------------------------------


// ---------------------------------------------
// ------ SyncPlayerList.c - START --------------------
// ---------------------------------------------


class SyncPlayerList
{
■ref array<ref SyncPlayer> m_PlayerList;

■void CreatePlayerList()
■{
■■if (GetGame().IsServer())
■■{
■■■m_PlayerList = new array<ref SyncPlayer>;

■■■array<Man> players = new array<Man>;
■■■GetGame().GetWorld().GetPlayerList(players);

■■■for (int i = 0; i < players.Count(); ++i)
■■■{
■■■■Man player = players[i];
■■■■PlayerIdentity p_identity = player.GetIdentity();

■■■■if (p_identity)
```

```
// ---------------------------------------------
// ------ Syringe.c - START --------------------
// ---------------------------------------------


class Syringe extends BloodContainerBase
{
■override void SetActions()
■{
■■super.SetActions();
■■
■■AddAction(ActionCollectSampleTarget);
■■AddAction(ActionCollectSampleSelf);
■}
}


// ---------------------------------------------
// ------ Syringe.c - END ----------------------
// ---------------------------------------------


// ---------------------------------------------
// ------ Tabber.c - START --------------------
// ---------------------------------------------


// ---------------------------------------------------------
class Tabber : ScriptedWidgetEventHandler
{
■Widget ButtonsPanel;
■Widget ContentPanel;
■reference int SelectedTab;
■
■// ---------------------------------------------------------
■void OnWidgetScriptInit(Widget w)
■{
■■w.SetHandler(this);
■■SelectTab(SelectedTab);
■}
■
■// ---------------------------------------------------------
■override bool OnClick(Widget w, int x, int y, int button)
■{
■■if (w && w.GetParent() == ButtonsPanel)
■■{
■■■SelectedTab = 0;
■■■Widget iw = ButtonsPanel.GetChildren();
■■■while(iw)
■■■{
■■■■if (iw == w) break;
■■■■SelectedTab++;
■■■■iw = iw.GetSibling();
■■■}
■
■■■SelectTab(SelectedTab);
■■}
■■
■■return false;
■}
■
■// ---------------------------------------------------------
■protected void SelectTab(int index)
```

```
// ----------------------------------------------
// ------ TabberUI.c - START --------------------
// ----------------------------------------------


class TabberUI extends ScriptedWidgetEventHandler
{
■protected bool■■■■■m_FirstInit = true;
■protected Widget■■■■m_Root;
■protected Widget■■■■m_TabControlsRoot;
■
■protected int ■■■■■m_TabsCount;
■protected ref map<int, Widget>■m_TabControls;
■protected ref map<int, Widget>■m_Tabs;
■
■protected int■■■■■m_SelectedIndex;
■protected float■■■■■m_ResolutionMultiplier;
■protected bool■■■■■m_CanSwitch;
■
■ref ScriptInvoker■■■■m_OnTabSwitch = new ScriptInvoker();
■ref ScriptInvoker■■■■m_OnAttemptTabSwitch = new ScriptInvoker();
■ref Timer■■■■■■m_InitTimer;
■
■protected void OnInputPresetChanged()
■{
■■#ifdef PLATFORM_CONSOLE
■■UpdateControlsElements();
■■#endif
■}
■
■protected void OnInputDeviceChanged(EInputDeviceType pInputDeviceType)
■{
■■switch (pInputDeviceType)
■■{
■■case EInputDeviceType.CONTROLLER:
■■■UpdateControlsElements();
■■■m_TabControlsRoot.FindAnyWidget("ConsoleControls").Show(true);
■■break;

■■default:
■■■if (GetGame().GetInput().IsEnabledMouseAndKeyboardEvenOnServer())
■■■{
■■■■m_TabControlsRoot.FindAnyWidget("ConsoleControls").Show(false);
■■■}
■■break;
■■}
■}
■
■void Init()
■{
■■int x, y;
■■GetScreenSize( x, y );
■■m_ResolutionMultiplier■= y / 1080;
■■m_CanSwitch = true;
■■m_TabsCount■= 0;
■■
■■Widget tab_controls■= m_Root.FindAnyWidget("Tab_Control_Container");
■■if ( tab_controls )
■■{
■■■Widget tab_child■= tab_controls.GetChildren();
■■■
■■■while ( tab_child )
■■■■{
```

```
// ---------------------------------------------
// ------ TacticalBaconCan.c - START --------------------
// ---------------------------------------------


class TacticalBaconCan : Edible_Base
{
■override void Open()
■{
■■ReplaceEdibleWithNew("TacticalBaconCan_Opened");
■}
■
■
■override bool IsOpen()
■{
■■return false;
■}
}



// ---------------------------------------------
// ------ TacticalBaconCan.c - END ----------------------
// ---------------------------------------------


// ---------------------------------------------
// ------ TacticalBaconCan_Opened.c - START --------------------
// ---------------------------------------------


class TacticalBaconCan_Opened: Edible_Base
{
■override void SetActions()
■{
■■super.SetActions();
■■
■■AddAction(ActionForceFeedCan);
■■AddAction(ActionEatCan);
■}
}



// ---------------------------------------------
// ------ TacticalBaconCan_Opened.c - END ----------------------
// ---------------------------------------------


// ---------------------------------------------
// ------ TacticalGloves_ColorBase.c - START --------------------
// ---------------------------------------------


class TacticalGloves_ColorBase extends Clothing {};
class TacticalGloves_Black extends TacticalGloves_ColorBase {};
class TacticalGloves_Beige extends TacticalGloves_ColorBase {};
class TacticalGloves_Green extends TacticalGloves_ColorBase {};


// ---------------------------------------------
// ------ TacticalGloves_ColorBase.c - END ----------------------
// ---------------------------------------------
// ---------------------------------------------
// ------ TacticalGoggles.c - START --------------------
// ---------------------------------------------


class TacticalGoggles extends Clothing
{
■override int GetGlassesEffectID()
■{
■■return PPERequesterBank.REQ_GLASSESTACTICAL;
■}
■
■override bool CanPutAsAttachment(EntityAI parent)
■{
■■if (!super.CanPutAsAttachment(parent))
■■■return false;
■■
■■Clothing mask = Clothing.Cast(parent.FindAttachmentBySlotName("Mask"));
■■if (mask && mask.ConfigGetBool("noEyewear"))
■■{
■■■return false;
■■}
■■
■■return true;
■}
};


// ---------------------------------------------
// ------ TacticalGoggles.c - END ----------------------
// ---------------------------------------------


// ---------------------------------------------
// ------ TacticalShirt_ColorBase.c - START --------------------
// ---------------------------------------------


class TacticalShirt_ColorBase extends Clothing
{
■override void SetActions()
■{
■■super.SetActions();
■■AddAction(ActionWringClothes);
■}
};
class TacticalShirt_Grey extends TacticalShirt_ColorBase {};
class TacticalShirt_Black extends TacticalShirt_ColorBase {};
class TacticalShirt_Olive extends TacticalShirt_ColorBase {};
class TacticalShirt_Tan extends TacticalShirt_ColorBase {};


// ---------------------------------------------
// ------ TacticalShirt_ColorBase.c - END ----------------------
// ---------------------------------------------


// ---------------------------------------------
// ------ TaloonBag_ColorBase.c - START --------------------
// ---------------------------------------------


class TaloonBag_ColorBase extends Clothing {};
```

```
// ---------------------------------------------
// ------ TankerHelmet.c - START --------------------
// ---------------------------------------------


class TankerHelmet extends Clothing
{
■override void SetActions()
■{
■■super.SetActions();
■■AddAction(ActionWringClothes);
■}
};



// ---------------------------------------------
// ------ TankerHelmet.c - END ----------------------
// ---------------------------------------------


// ---------------------------------------------
// ------ TelescopicBaton.c - START --------------------
// ---------------------------------------------


class TelescopicBaton : ItemBase
{
■protected int m_MeleeMode;
■protected int m_MeleeHeavyMode;
■protected int m_MeleeSprintMode;
■
■protected ref OpenableBehaviour m_Openable;
■
■void TelescopicBaton()
■{
■■//!set default melee modes on init
■■m_MeleeMode = 0;
■■m_MeleeHeavyMode = 1;
■■m_MeleeSprintMode = 2;

■■m_Openable = new OpenableBehaviour(false);
■■
■■RegisterNetSyncVariableBool("m_Openable.m_IsOpened");

■■UpdateVisualState();
■}
■
■override void Open()
■{
■■m_Openable.Open();
■■SetSynchDirty();

■■//! sets different set of melee modes for opened state
■■m_MeleeMode = 3;
■■m_MeleeHeavyMode = 4;
■■m_MeleeSprintMode = 5;

■■UpdateVisualState();
■}

■override void Close()
■{
```

```
// ---------------------------------------------
// ------ TelnyashkaShirt.c - START --------------------
// ---------------------------------------------


class TelnyashkaShirt extends Clothing
{
■override void SetActions()
■{
■■super.SetActions();
■■AddAction(ActionWringClothes);
■}
};


// ---------------------------------------------
// ------ TelnyashkaShirt.c - END ----------------------
// ---------------------------------------------


// ---------------------------------------------
// ------ TendencyBacteria.c - START --------------------
// ---------------------------------------------


class TendencyBacteria extends DisplayElementTendency
{
■void TendencyBacteria(PlayerBase player)
■{
■■m_Type = eDisplayElements.DELM_TDCY_BACTERIA;
■■m_Key = NTFKEY_BACTERIA;
■}
■
}


// ---------------------------------------------
// ------ TendencyBacteria.c - END ----------------------
// ---------------------------------------------


// ---------------------------------------------
// ------ TendencyBlood.c - START --------------------
// ---------------------------------------------


class TendencyBlood extends DisplayElementTendency
{
■void TendencyBlood(PlayerBase player)
■{
■■m_Type■=■eDisplayElements.DELM_TDCY_BLOOD;
■■m_Key = NTFKEY_BLEEDISH;
■}
}


// ---------------------------------------------
// ------ TendencyBlood.c - END ----------------------
// ---------------------------------------------
// ---------------------------------------------
// ------ TendencyHealth.c - START --------------------
// ---------------------------------------------


class TendencyHealth extends DisplayElementTendency
{
■void TendencyHealth(PlayerBase player)
■{
■■m_Type■=■eDisplayElements.DELM_TDCY_HEALTH;
■■m_Key = NTFKEY_SICK;
■}
■
}


// ---------------------------------------------
// ------ TendencyHealth.c - END ----------------------
// ---------------------------------------------


// ---------------------------------------------
// ------ TendencyHunger.c - START --------------------
// ---------------------------------------------


class TendencyHunger extends DisplayElementTendency
{
■void TendencyHunger(PlayerBase player)
■{
■■m_Type■=■eDisplayElements.DELM_TDCY_ENERGY;
■■m_Key = NTFKEY_HUNGRY;
■}
}


// ---------------------------------------------
// ------ TendencyHunger.c - END ----------------------
// ---------------------------------------------


// ---------------------------------------------
// ------ TendencyTemperature.c - START --------------------
// ---------------------------------------------


class TendencyTemperature extends DisplayElementTendency
{
■void TendencyTemperature(PlayerBase player)
■{
■■TENDENCY_MASK = 7;// 3 bits
■■SERIOUSNESS_BIT_MASK = 7;// 3 bits
■■SERIOUSNESS_BIT_OFFSET = 3;//3 bit offset
■■NUM_OF_BITS = 6;■■

■■m_Type■= eDisplayElements.DELM_TDCY_TEMPERATURE;
■■m_Key = NTFKEY_FEVERISH;
■}
■
■override int TranslateLevelToStatus(int level)
■{
■■if( level == DSLevelsTemp.WARNING_PLUS ) return 2;
■■if( level == DSLevelsTemp.CRITICAL_PLUS ) return 3;
```

```
// ---------------------------------------------
// ------ TendencyThirst.c - START --------------------
// ---------------------------------------------


class TendencyThirst extends DisplayElementTendency
{
■void TendencyThirst(PlayerBase player)
■{
■■m_Type■=■eDisplayElements.DELM_TDCY_WATER;
■■m_Key = NTFKEY_THIRSTY;
■}
}


// ---------------------------------------------
// ------ TendencyThirst.c - END ----------------------
// ---------------------------------------------


// ---------------------------------------------
// ------ TentBase.c - START --------------------
// ---------------------------------------------


class TentBase extends ItemBase
{
■const int OPENING_0 = 1;
■const int OPENING_1 = 2;
■const int OPENING_2 = 4;
■const int OPENING_3 = 8;
■const int OPENING_4 = 16;
■const int OPENING_5 = 32;
■const int OPENING_6 = 64;
■const int OPENING_7 = 128;
■const int OPENING_8 = 256;
■const int OPENING_9 = 512;
■const int OPENING_10 = 1024;
■const int OPENING_11 = 2048;
■const int OPENING_12 = 4096;
■const int OPENING_13 = 8192;
■const int OPENING_14 = 16384;
■const int OPENING_15 = 32768;
■
■static const int PACKED ■= 0;
■static const int PITCHED ■= 1;
■const float MAX_PLACEMENT_HEIGHT_DIFF = 1.5;
■
■//bool m_DamageSystemStarted = false;
■protected int m_State;
■protected int m_StateLocal = -1;
■protected bool m_IsEntrance;
■protected bool m_IsWindow;
■protected bool m_IsToggle;
■protected bool m_IsBeingPacked = false;
■protected int m_OpeningMask = 0;
■protected int m_OpeningMaskLocal = -1;
■
■protected ref map< ref ToggleAnimations, bool> m_ToggleAnimations;
■protected ref array<string> m_ShowAnimationsWhenPitched;
■protected ref array<string> m_ShowAnimationsWhenPacked;
■protected Object■■■m_ClutterCutter;
■ref protected EffectSound ■m_DeployLoopSound;
```

```
// ---------------------------------------------
// ------ Test.c - START --------------------
// ---------------------------------------------


class Test extends RecipeBase
{■
■override void Init()
■{
■■m_Name = "Test 3 ingredients";
■■m_IsInstaRecipe = false;//should this recipe be performed instantly without animation
■■m_AnimationLength = 1;//animation length in relative time units
■■m_Specialty = 0;// value > 0 for roughness, value < 0 for precision
■■
■■
■■//conditions
■■m_MinDamageIngredient[0] = -1;//-1 = disable check
■■m_MaxDamageIngredient[0] = 3;//-1 = disable check
■■
■■m_MinQuantityIngredient[0] = -1;//-1 = disable check
■■m_MaxQuantityIngredient[0] = -1;//-1 = disable check
■■
■■m_MinDamageIngredient[1] = -1;//-1 = disable check
■■m_MaxDamageIngredient[1] = 3;//-1 = disable check
■■
■■m_MinQuantityIngredient[1] = -1;//-1 = disable check
■■m_MaxQuantityIngredient[1] = -1;//-1 = disable check
■■/*
■■m_MinDamageIngredient[2] = 1;//-1 = disable check
■■m_MaxDamageIngredient[2] = -1;//-1 = disable check
■■
■■m_MinQuantityIngredient[2] = -1;//-1 = disable check
■■m_MaxQuantityIngredient[2] = -1;//-1 = disable check
■■*/
■■//----------------------------------------------------------------------------------------------------------------------
■■
■■//INGREDIENTS
■■//ingredient 1
■■InsertIngredient(0,"BloodTestKit");//you can insert multiple ingredients this way
■■
■■m_IngredientAddHealth[0] = 0;// 0 = do nothing
■■m_IngredientSetHealth[0] = -1; // -1 = do nothing
■■m_IngredientAddQuantity[0] = 0;// 0 = do nothing
■■m_IngredientDestroy[0] = true;//true = destroy, false = do nothing
■■m_IngredientUseSoftSkills[0] = false;// set 'true' to allow modification of the values by softskills on this i
■■
■■//ingredient 2
■■InsertIngredient(1,"BloodSyringe");//you can insert multiple ingredients this way
■■m_IngredientAddHealth[1] = 0;// 0 = do nothing
■■m_IngredientSetHealth[1] = -1; // -1 = do nothing
■■m_IngredientAddQuantity[1] = 0;// 0 = do nothing
■■m_IngredientDestroy[1] = false;// false = do nothing
■■m_IngredientUseSoftSkills[1] = false;// set 'true' to allow modification of the values by softskills on this i
■■
■■/*
■■//ingredient 3
■■InsertIngredient(2,"SalineBagIV");//you can insert multiple ingredients this way
■■
■■m_IngredientAddHealth[2] = 0;// 0 = do nothing
■■m_IngredientSetHealth[2] = -1; // -1 = do nothing
■■m_IngredientAddQuantity[2] = 0;// 0 = do nothing
■■m_IngredientDestroy[2] = false;// false = do nothing
■■m_IngredientUseSoftSkills[1] = false;// set 'true' to allow modification of the values by softskills on this i
```

```
// ---------------------------------------------
// ------ TestDisease.c - START --------------------
// ---------------------------------------------


class TestDiseaseMdfr: ModifierBase
{
	float m_Interval;
	ref Param1<float> m_SomeValue;

	override void Init()
	{
		m_TrackActivatedTime			= false;
		m_ID 							= 9999;
		m_TickIntervalInactive 	= DEFAULT_TICK_TIME_INACTIVE;
		m_TickIntervalActive 	= 5;
		m_SomeValue		= new Param1<float>(0);
		MakeParamObjectPersistent(m_SomeValue);
		
		DisableDeactivateCheck();
		DisableActivateCheck();
	}
	
	override bool ActivateCondition(PlayerBase player)
	{
		return false;
	}

	override void OnActivate(PlayerBase player)
	{
		//player.GetBleedingManagerServer().AttemptAddBleedingSource(Math.RandomInt(0, 100));
	}
	
	override void OnReconnect(PlayerBase player)
	{
		//OnActivate(player);
		//player.GetSymptomManager().QueueUpSecondaryState( SymptomIDs.SYMPTOM_BLINDNESS );
	}

	override void OnDeactivate(PlayerBase player)
	{
	}

	override bool DeactivateCondition(PlayerBase player)
	{
		return false;
	}

	override void OnTick(PlayerBase player, float deltaT)
	{
		player.GetSymptomManager().QueueUpPrimarySymptom(SymptomIDs.SYMPTOM_LAUGHTER);
	}
};


// ---------------------------------------------
// ------ TestDisease.c - END ---------------------
// ---------------------------------------------
// ---------------------------------------------
// ------ TestFramework.c - START --------------------
// ---------------------------------------------


enum TFR
{
■FAIL = -1,
■SUCCESS,
■PENDING,
}

class TFResult // Pretty much just to be able to keep a reference...
{
■TFR Result;
■
■void TFResult(TFR result)
■{
■■Result = result;
■}
■
■TFResult And(TFResult other)
■{
■■if (Result == TFR.PENDING || other.Result == TFR.PENDING)
■■■ErrorEx("Trying to And while one of the results are PENDING.");
■■
■■if (Result == TFR.SUCCESS && other.Result == TFR.SUCCESS)
■■■Result = TFR.SUCCESS;
■■else
■■■Result = TFR.FAIL;
■■
■■return this;
■}
■
■TFResult Or(TFResult other)
■{
■■if (Result == TFR.PENDING || other.Result == TFR.PENDING)
■■■ErrorEx("Trying to Or while one of the results are PENDING.");
■■
■■if (Result == TFR.SUCCESS || other.Result == TFR.SUCCESS)
■■■Result = TFR.SUCCESS;
■■else
■■■Result = TFR.FAIL;
■■
■■return this;
■}
}
typedef array<ref TFResult> TFResultArr;

class TFCaller
{
■private Class ■■■m_Instance;
■private string ■■■m_Test;
■private ref TFResult ■m_Result;
■
■void TFCaller(Class instance, string test, TFResult result)
■{
■■m_Instance ■= instance;
■■m_Test ■■= test;
■■m_Result ■= result;
■}
■
■TFResult Run(float dt)
```

```
// ---------------------------------------------
// ------ Testing.c - START --------------------
// ---------------------------------------------


class TestingMdfr: ModifierBase
{
■float stuff;
■bool swch;
■override void Init()
■{
■■m_TrackActivatedTime■■■= true;
■■m_ID ■■■■■= eModifiers.MDF_TESTING;
■■m_TickIntervalInactive ■= DEFAULT_TICK_TIME_INACTIVE;
■■m_TickIntervalActive ■= DEFAULT_TICK_TIME_ACTIVE;
■}

■override void OnActivate(PlayerBase player)
■{
■■//m_Timer1.Run(1, this, "BadaBang");
■■player.GetSymptomManager().QueueUpPrimarySymptom( SymptomIDs.SYMPTOM_COUGH );
■■
■}

■override void OnReconnect(PlayerBase player)
■{

■}

■// ----------------------------------------------------------------------------

■// ----------------------------------------------------------------------------

■override bool ActivateCondition(PlayerBase player)
■{
■■return swch;
■■if(stuff < 40)
■■{
■■■return true;
■■}
■■else return false;
■}

■override bool DeactivateCondition(PlayerBase player)
■{
■■if (GetAttachedTime() > 100) {swch = true; return true;}
■■else return false;
■■if(stuff > 40)
■■{
■■■return true;
■■}
■■else return false;
■}
■// ----------------------------------------------------------------------------

■override void OnTick(PlayerBase player, float deltaT)
■{
■■stuff += deltaT;
■}
};


// ---------------------------------------------
// ---------------------------------------------
// ------ TetracyclineAntibiotics.c - START --------------------
// ---------------------------------------------


class TetracyclineAntibiotics : Edible_Base
{
■override void OnConsume(float amount, PlayerBase consumer)
■{
■■if (consumer.GetModifiersManager().IsModifierActive(eModifiers.MDF_ANTIBIOTICS)) //effectively res
■■{
■■■consumer.GetModifiersManager().DeactivateModifier(eModifiers.MDF_ANTIBIOTICS);
■■}

■■consumer.GetModifiersManager().ActivateModifier(eModifiers.MDF_ANTIBIOTICS);
■}
■
■override void SetActions()
■{
■■super.SetActions();
■■
■■AddAction(ActionForceConsumeSingle);
■■AddAction(ActionConsumeSingle);
■}
}



// ---------------------------------------------
// ------ TetracyclineAntibiotics.c - END ----------------------
// ---------------------------------------------


// ---------------------------------------------
// ------ Thermometer.c - START --------------------
// ---------------------------------------------


class Thermometer extends ItemBase
{
■float GetTemperatureValue(PlayerBase player)
■{
■■string temp;
■■float value;
■■if( player.GetModifiersManager() && player.GetModifiersManager().IsModifierActive(eModifiers.MDF_F
■■{
■■■value = Math.RandomFloatInclusive(PlayerConstants.HIGH_TEMPERATURE_L, PlayerConstants.H
■■}
■■else
■■{
■■■value = Math.RandomFloatInclusive(PlayerConstants.NORMAL_TEMPERATURE_L, PlayerConstan
■■}
■■value = Math.Round(value * 10) / 10;
■■return value;
■■//temp = value.ToString();
■■//return temp + "C";
■}
■
■override void SetActions()
■{
■■super.SetActions();
■■
■■AddAction(ActionMeasureTemperatureTarget);
```

```
// ---------------------------------------------
// ------ ThickFramesGlasses.c - START --------------------
// ---------------------------------------------


class ThickFramesGlasses extends Clothing
{
■override bool CanPutAsAttachment(EntityAI parent)
■{
■■if (!super.CanPutAsAttachment(parent))
■■■return false;
■■
■■Clothing mask = Clothing.Cast(parent.FindAttachmentBySlotName("Mask"));
■■if (mask && mask.ConfigGetBool("noEyewear"))
■■{
■■■return false;
■■}
■■
■■return true;
■}
};


// ---------------------------------------------
// ------ ThickFramesGlasses.c - END ----------------------
// ---------------------------------------------


// ---------------------------------------------
// ------ ThinFramesGlasses.c - START --------------------
// ---------------------------------------------


class ThinFramesGlasses extends Clothing
{
■override bool CanPutAsAttachment(EntityAI parent)
■{
■■if (!super.CanPutAsAttachment(parent))
■■■return false;
■■
■■Clothing mask = Clothing.Cast(parent.FindAttachmentBySlotName("Mask"));
■■if (mask && mask.ConfigGetBool("noEyewear"))
■■{
■■■return false;
■■}
■■
■■return true;
■}
};


// ---------------------------------------------
// ------ ThinFramesGlasses.c - END ----------------------
// ---------------------------------------------


// ---------------------------------------------
// ------ Thirst.c - START --------------------
// ---------------------------------------------


class ThirstMdfr: ModifierBase
{
```

```
// ---------------------------------------------
// ------ ThirstNotfr.c - START --------------------
// ---------------------------------------------


class ThirstNotfr: NotifierBase
{
■private const float ■HYDRATED_TRESHOLD ■■■= 3500;
■private const float ■THIRSTY_TRESHOLD ■■■= 2500;
■private const float ■VERY_THIRSTY_TRESHOLD ■■= 1500;
■private const float ■FATALLY_THIRSTY_TRESHOLD ■= 0;
■private const float ■DEC_TRESHOLD_LOW ■■■= 0;
■private const float ■DEC_TRESHOLD_MED ■■■= -0.2;
■private const float ■DEC_TRESHOLD_HIGH■■■= -0.85;
■private const float ■INC_TRESHOLD_LOW ■■■= 0;
■private const float ■INC_TRESHOLD_MED ■■■= 0.2;
■private const float ■INC_TRESHOLD_HIGH■■■= 0.85;
■■
■void ThirstNotfr(NotifiersManager manager)
■{
■}

■override int GetNotifierType()
■{
■■return eNotifiers.NTF_THIRSTY;
■}

■
■override void DisplayTendency(float delta)
■{
■■//PrintString(delta.ToString());
■■int tendency = CalculateTendency(delta, INC_TRESHOLD_LOW, INC_TRESHOLD_MED, INC_TRES
■■//GetVirtualHud().SetStatus(eDisplayElements.DELM_TDCY_WATER,tendency);
■■
■■//DSLevels level = DetermineLevel( GetObservedValue(), PlayerConstants.THRESHOLD_WATER_WA
■■EStatLevels water_level = m_Player.GetStatLevelWater();
■■DisplayElementTendency dis_elm = DisplayElementTendency.Cast(GetVirtualHud().GetElement(eDisp
■■
■■if( dis_elm )
■■{
■■■dis_elm.SetTendency(tendency);
■■■dis_elm.SetSeriousnessLevel(water_level);
■■■
■■}
■}
■
■override void DisplayBadge()
■{
■■float water = m_Player.GetStatWater().Get();
■■if (water >= HYDRATED_TRESHOLD)
■■{
■■■//GetVirtualHud().SetStatus(eDisplayElements.DELM_NTFR_THIRST,DELM_LVL_1);
■■}
■■else if (water <= THIRSTY_TRESHOLD)
■■{
■■■//GetVirtualHud().SetStatus(eDisplayElements.DELM_NTFR_THIRST,DELM_LVL_2);
■■}
■■else if (water <= VERY_THIRSTY_TRESHOLD)
■■{
■■■//GetVirtualHud().SetStatus(eDisplayElements.DELM_NTFR_THIRST,DELM_LVL_3);
■■}
■■else if (water <= FATALLY_THIRSTY_TRESHOLD)
■■{
```

```
// ---------------------------------------------
// ------ TimeAccel.c - START --------------------
// ---------------------------------------------


#ifdef DIAG_DEVELOPER
typedef Param3<bool, float, int> TimeAccelParam;//enabled, timeAccel,category mask

enum ETimeAccelCategories
{
■//DO NOT FORGET TO INCLUDE IN THE MAPPING 'Init()' FUNCTION
■UNDERGROUND_ENTRANCE ■= 0x00000001,
■UNDERGROUND_RESERVOIR ■= 0x00000002,
■ENERGY_CONSUMPTION ■■= 0x00000004,
■ENERGY_RECHARGE ■■= 0x00000008,
■FOOD_DECAY ■■■■= 0x00000010,
■//SYSTEM5 = 0x00000020,
■//SYSTEM6 = 0x00000040,
■//SYSTEM6 = 0x00000080,
}

class FeatureTimeAccel
{
■static ref TimeAccelParam ■■m_CurrentTimeAccel;// ■= new TimeAccelParam(false, 0, 0);
■static ref map<int,int> ■■m_Mapping ■■■= new map<int,int>();
■static ref array<int> ■■■m_IDs■■■■= new array<int>();
■static bool ■■■■■m_Initializeded ■= Init();
■
■static bool Init()
■{
■■Bind(DiagMenuIDs.FEATURE_TIME_ACCEL_UG_ENTRANCES, ■■ETimeAccelCategories.UNDERG
■■Bind(DiagMenuIDs.FEATURE_TIME_ACCEL_UG_RESERVOIR, ■■ETimeAccelCategories.UNDERG
■■Bind(DiagMenuIDs.FEATURE_TIME_ACCEL_ENERGY_CONSUME, ■ETimeAccelCategories.ENERG
■■Bind(DiagMenuIDs.FEATURE_TIME_ACCEL_ENERGY_RECHARGE, ■ETimeAccelCategories.ENER
■■Bind(DiagMenuIDs.FEATURE_TIME_ACCEL_FOOD_DECAY, ■■ETimeAccelCategories.FOOD_DEC
■■return true;
■}
■
■//-------------------------------
■
■static void Bind(DiagMenuIDs id, ETimeAccelCategories catBit)
■{
■■m_Mapping.Insert(id, catBit);
■■m_IDs.Insert(id);
■}
■
■//-------------------------------
■
■static int GetCategoryByDiagID(DiagMenuIDs id)
■{
■■return m_Mapping.Get(id);
■}
■
■//-------------------------------
■
■static int GetDiagIDByCategory(ETimeAccelCategories category)
■{
■■for (int i = 0; i < m_Mapping.Count();i++)
■■{
■■■if (m_Mapping.GetElement(i) == category)
■■■{
■■■■return m_Mapping.GetKey(i);
■■■}
```

```
// ---------------------------------------------
// ------ TitleScreenMenu.c - START --------------------
// ---------------------------------------------


/*! Xbox menu */
class TitleScreenMenu extends UIScriptedMenu
{
■RichTextWidget m_TextPress;
■
■void TitleScreenMenu()
■{
■■g_Game.SetGameState(DayZGameState.MAIN_MENU);
■■g_Game.SetLoadState(DayZLoadState.MAIN_MENU_START);
■}
■
■void ~TitleScreenMenu()
■{
■}
■
■override Widget Init()
■{
■■layoutRoot = GetGame().GetWorkspace().CreateWidgets("gui/layouts/xbox/day_z_title_screen.layout")
■■
■■MissionMainMenu mission = MissionMainMenu.Cast(g_Game.GetMission());
■■
■■m_TextPress = RichTextWidget.Cast(layoutRoot.FindAnyWidget("InputPromptText"));
■■if (m_TextPress)
■■{
■■■string gamertag;
■■■string text = Widget.TranslateString("#console_start_game");
■■■GetGame().GetPlayerName(gamertag);
■■■#ifdef PLATFORM_XBOX
■■■■BiosUserManager user_manager = GetGame().GetUserManager();
■■■■if (user_manager && user_manager.GetSelectedUser())
■■■■■m_TextPress.SetText(string.Format(text, "<image set=\"xbox_buttons\" name=\"A\" />"));
■■■■else
■■■■■m_TextPress.SetText(string.Format(text, "<image set=\"xbox_buttons\" name=\"A\" />"));
■■■#endif
■■■■■
■■■#ifdef PLATFORM_PS4
■■■■string confirm = "cross";
■■■■if (GetGame().GetInput().GetEnterButton() == GamepadButton.A)
■■■■{
■■■■■confirm = "cross";
■■■■}
■■■■else
■■■■{
■■■■■confirm = "circle";
■■■■}
■■■■m_TextPress.SetText(string.Format(text, "<image set=\"playstation_buttons\" name=\"" + confirm +
■■■#endif
■■}
■■return layoutRoot;
■}
■
■override void OnShow()
■{
■■if (g_Game.GetGameState() != DayZGameState.CONNECTING)
■■{
■■■#ifdef PLATFORM_CONSOLE
■■■■g_Game.GamepadCheck();
■■■#endif
```

```
// ---------------------------------------------
// ------ TLRLight.c - START --------------------
// ---------------------------------------------


class TLRLight extends Switchable_Base
{
■PistollightLight ■m_Light;
■
■static int■■■REFLECTOR_ID = 1;
■static int■■■GLASS_ID = 0;
■
■static string ■■LIGHT_OFF_GLASS = "dz\\gear\\tools\\data\\flashlight_glass.rvmat";
■static string ■■LIGHT_OFF_REFLECTOR = "dz\\weapons\\attachments\\data\\TLS3.rvmat";
■static string ■■LIGHT_ON_GLASS = "dz\\gear\\tools\\data\\flashlight_glass_on.rvmat";
■static string ■■LIGHT_ON_REFLECTOR = "dz\\weapons\\attachments\\data\\TLS3_on.rvmat";
■
■override ScriptedLightBase GetLight()
■{
■■return m_Light;
■}
■
■override bool CanPutAsAttachment( EntityAI parent )
■{
■■if(!super.CanPutAsAttachment(parent)) {return false;}
■■if ( !parent.IsKindOf("PlateCarrierHolster") && !parent.IsKindOf("PlateCarrierComplete") && !parent.IsK
■■{
■■■return true;
■■}
■■
■■return false;
■}

■override void OnWorkStart()
■{
■■if ( !GetGame().IsServer()  ||  !GetGame().IsMultiplayer() ) // Client side
■■{
■■■m_Light = PistollightLight.Cast(  ScriptedLightBase.CreateLight(PistollightLight, "0 0 0", 0.08)  ); // Po
■■■m_Light.AttachOnMemoryPoint(this, "beamStart", "beamEnd");
■■■SetObjectMaterial(GLASS_ID, LIGHT_ON_GLASS);
■■■SetObjectMaterial(REFLECTOR_ID, LIGHT_ON_REFLECTOR);
■■}
■}

■override void OnWork( float consumed_energy )
■{
■■if ( !GetGame().IsServer()  ||  !GetGame().IsMultiplayer() ) // Client side
■■{
■■■Battery9V battery = Battery9V.Cast( GetCompEM().GetEnergySource() );
■■■
■■■if (battery  &&  m_Light)
■■■{
■■■■float efficiency = battery.GetEfficiency0To1();
■■■■
■■■■if ( efficiency < 1 )
■■■■{
■■■■■m_Light.SetIntensity( efficiency, GetCompEM().GetUpdateInterval() );
■■■■}
■■■■else
■■■■{
■■■■■m_Light.SetIntensity( 1, 0 );
■■■■}
■■■}
```

```
// ----------------------------------------------
// ------ ToggleSelections.c - START --------------------
// ----------------------------------------------


class ToggleAnimations
{
■protected string m_ToggleOff;
■protected string m_ToggleOn;
■int m_OpeningBit;
■//ref protected array<string> ■m_ToggleOffLinkedAnimations;
■//ref protected array<string> ■m_ToggleOnLinkedAnimations;
■
■void ToggleAnimations( string toggle_off, string toggle_on, int mask, array<string> linked_anims_off = nu
■{
■■m_ToggleOff = toggle_off;
■■m_ToggleOn = toggle_on;
■■m_OpeningBit = mask;
■■//m_ToggleOffLinkedAnimations = linked_anims_off;
■■//m_ToggleOnLinkedAnimations = linked_anims_on;
■}

■string GetToggleOff()
■{
■■return m_ToggleOff;
■}

■string GetToggleOn()
■{
■■return m_ToggleOn;
■}
■
■int GetOpeningBit()
■{
■■return m_OpeningBit;
■}
■
■/*array<string> GetLinkedOff()
■{
■■return m_ToggleOffLinkedAnimations;
■}
■
■array<string> GetLinkedOn()
■{
■■return m_ToggleOnLinkedAnimations;
■}*/
}



// ----------------------------------------------
// ------ ToggleSelections.c - END ----------------------
// ----------------------------------------------


// ----------------------------------------------
// ------ Tomato.c - START --------------------
// ----------------------------------------------


class Tomato : Edible_Base
{
■override bool CanBeCookedOnStick()
```

```
// ---------------------------------------------
// ------ TomatoSeedsPack.c - START --------------------
// ---------------------------------------------


class TomatoSeedsPack extends SeedPackBase
{
}


// ---------------------------------------------
// ------ TomatoSeedsPack.c - END ----------------------
// ---------------------------------------------


// ---------------------------------------------
// ------ ToolBase.c - START --------------------
// ---------------------------------------------


//TODO trees are static objects, there is no script event for playing sounds on clients when they are chopp
class ToolBase extends ItemBase
{■
■protected int m_MineDisarmRate = 60; //Success rate when disarming with this tool
■
■void ToolBase()
■{
■■
■}
■
■int GetDisarmRate()
■{
■■return m_MineDisarmRate;
■}
■■■
■override void OnRPC(PlayerIdentity sender, int rpc_type,ParamsReadContext  ctx)
■{
■■super.OnRPC(sender, rpc_type,ctx);
■■
■■switch(rpc_type)
■■{
■■■case PlantType.TREE_HARD:
■■■■SoundHardTreeFallingPlay();
■■■break;
■■■
■■■case PlantType.TREE_SOFT:
■■■■SoundSoftTreeFallingPlay();
■■■break;
■■■
■■■case PlantType.BUSH_HARD:
■■■■SoundHardBushFallingPlay();
■■■break;
■■■
■■■case PlantType.BUSH_SOFT:
■■■■SoundSoftBushFallingPlay();
■■■break;
■■}
■}
}


// ---------------------------------------------
// ------ ToolBase.c - END ----------------------
// ---------------------------------------------
// ------ tools.c - START --------------------
// ---------------------------------------------


/**
 * \defgroup Tools
 * \desc Helpful functions & classes
 * @{
 */

//-----------------------------------------------------------------------
const int CALL_CATEGORY_SYSTEM = 0; // Runs always
const int CALL_CATEGORY_GUI = 1; // Runs always (on client)
const int CALL_CATEGORY_GAMEPLAY = 2; // Runs unless ingame menu is opened

const int CALL_CATEGORY_COUNT = 3;

// ----------------------------------------------------------------------
class CallQueueContext
{
■Class m_target;
■string m_function;
■ref Param m_params;
■bool m_valid;

■void CallQueueContext(Class target, string fn, Param params)
■{
■■m_target = target;
■■m_function = fn;
■■m_params = params;
■■m_valid = true;
■}
■■
■void Call()
■{
■■CallParams(m_params);
■}
■■
■void CallParams(Param params)
■{
■■if (params)
■■{
■■■GetGame().GameScript.CallFunctionParams(m_target, m_function, NULL, params);
■■}
■■else
■■{
■■■GetGame().GameScript.CallFunction(m_target, m_function, NULL, 0);
■■}
■}
■
■void Invalidate() {
■■m_valid = false;
■}

■bool IsValid(){
■■return m_valid;
■}
};

//-----------------------------------------------------------------------
/**
 \brief CallQueue Class provide "lazy" calls - when we don't want to execute function immediately but later
```

```
// ----------------------------------------------
// ------ Torch.c - START --------------------
// ----------------------------------------------


class FlammableBase : ItemBase
{
■void FlammableBase()
■{
■■Init();
■}
■
■private SoundOnVehicle■m_LoopSoundEntity;
■Particle ■■■■m_FireParticle;
■bool■■■■■m_CanReceiveUpgrade; // Synchronized variable
■bool■■■■■m_IsBeingDestructed = false;
■
■float ■■■■■m_BurnTimePerRagEx;
■float ■■■■■m_BurnTimePerFullLardEx;
■float ■■■■■m_BurnTimePerFullFuelDoseEx;
■float ■■■■■m_MaxConsumableLardQuantityEx;
■float ■■■■■m_MaxConsumableFuelQuantityEx;
■float ■■■■■m_WaterEvaporationByFireIntensityEx = 0.001;
■int ■■■■■m_StartFadeOutOfLightAtQuantityEx = 3;
■
■int ■■■■■m_RagsUpgradedCount;//how many rags were upgraded with fuel/lard
■bool■■■■■m_ConsumeRagFlipFlop;//are we burning rag or fuel/lard
■vector ■■■■■m_ParticleLocalPos = Vector(0, 0.50, 0);
■
■string■■■■■m_DecraftResult = "WoodenStick";
■TorchLight■■■■m_Light;
■bool■■■■■m_WasLit;//was this item ever lit ? (used for correct material setting after reconnect for extin
■
■protected ref UniversalTemperatureSource m_UTSource;
■protected ref UniversalTemperatureSourceSettings m_UTSSettings;
■protected ref UniversalTemperatureSourceLambdaConstant m_UTSLConstant;
■
■override void DeferredInit()
■{
■■if(m_RagsUpgradedCount)
■■{
■■■LockRags(true);
■■}
■}
■
■void Init()
■{
■■if ( m_BurnTimePerRagEx == 0 || m_BurnTimePerFullLardEx == 0 || m_MaxConsumableLardQuantityE
■■{
■■■string cfg_path = "CfgVehicles " + GetType();
■■■m_BurnTimePerRagEx = GetGame().ConfigGetFloat( cfg_path + " burnTimePerRag" );
■■■m_BurnTimePerFullLardEx = GetGame().ConfigGetFloat( cfg_path + " burnTimePerFullLardDose" );
■■■m_BurnTimePerFullFuelDoseEx = GetGame().ConfigGetFloat( cfg_path + " burnTimePerFullFuelDos
■■■m_MaxConsumableLardQuantityEx = GetGame().ConfigGetFloat( cfg_path + " maxConsumableLard
■■■m_MaxConsumableFuelQuantityEx = GetGame().ConfigGetFloat( cfg_path + " maxConsumableFuel
■■}
■■RegisterNetSyncVariableBool("m_CanReceiveUpgrade");
■}
■
■override void EEInit()
■{■■
■■super.EEInit();
■■
```

```
// ---------------------------------------------
// ------ TorchLight.c - START --------------------
// ---------------------------------------------


class TorchLight extends PointLightBase
{
■static float m_TorchRadius = 30;
■static float m_TorchBrightness = 5.0;
■
■void TorchLight()
■{
■■SetVisibleDuringDaylight(false);
■■SetRadiusTo( m_TorchRadius );
■■SetBrightnessTo(m_TorchBrightness);
■■SetCastShadow(true);
■■SetFadeOutTime(1);
■■SetDiffuseColor(1.0, 0.45, 0.25);
■■SetAmbientColor(1.0, 0.45, 0.25);
■■SetFlareVisible(false);
■■SetFlickerAmplitude(0.55);
■■SetFlickerSpeed(0.75);
■■SetDancingShadowsMovementSpeed(0.1);
■■SetDancingShadowsAmplitude(0.03);
■■EnableHeatHaze(true);
■■SetHeatHazeRadius(0.08);
■■SetHeatHazePower(0.015);
■}
■
■override void OnFrameLightSource(IEntity other, float timeSlice)
■{
■■
■}
}


// ---------------------------------------------
// ------ TorchLight.c - END ----------------------
// ---------------------------------------------


// ---------------------------------------------
// ------ Torso.c - START --------------------
// ---------------------------------------------


class MaleTorso_Base extends InventoryItem {};
class FemaleTorso_Base extends InventoryItem {};

class MaleAdamTorso extends MaleTorso_Base {};
class MaleBorisTorso extends MaleTorso_Base {};
class MaleCyrilTorso extends MaleTorso_Base {};
class MaleDenisTorso extends MaleTorso_Base {};
class MaleEliasTorso extends MaleTorso_Base {};
class MaleFrancisTorso extends MaleTorso_Base {};
class MaleGuoTorso extends MaleTorso_Base {};
class MaleHassanTorso extends MaleTorso_Base {};
class MaleIndarTorso extends MaleTorso_Base {};
class MaleJoseTorso extends MaleTorso_Base {};
class MaleKaitoTorso extends MaleTorso_Base {};
class MaleLewisTorso extends MaleTorso_Base {};
class MaleManuaTorso extends MaleTorso_Base {};
class MaleNikiTorso extends MaleTorso_Base {};
```

```
// ---------------------------------------------
// ------ TorsoCover_improvised.c - START --------------------
// ---------------------------------------------


class TorsoCover_Improvised extends Clothing
{
■override void SetActions()
■{
■■super.SetActions();
■■AddAction(ActionWringClothes);
■}
};



// ---------------------------------------------
// ------ TorsoCover_improvised.c - END ----------------------
// ---------------------------------------------


// ---------------------------------------------
// ------ TortillaBag.c - START --------------------
// ---------------------------------------------


class TortillaBag : Clothing {};



// ---------------------------------------------
// ------ TortillaBag.c - END ----------------------
// ---------------------------------------------


// ---------------------------------------------
// ------ Totem.c - START --------------------
// ---------------------------------------------


class TerritoryFlag extends BaseBuildingBase
{
■const float MAX_ACTION_DETECTION_ANGLE_RAD ■■= 1.3;■■//1.3 RAD = ~75 DEG
■const float MAX_ACTION_DETECTION_DISTANCE ■■= 2.0;■■//meters
■
■bool m_RefresherActive;
■bool m_RefresherActiveLocal;
■bool m_RefresherInitialized;
■int m_RefresherTimeRemaining;
■int m_RefreshTimeCounter;
■
■protected int m_FlagRefresherFrequency = GameConstants.REFRESHER_FREQUENCY_DEFAULT; ■
■protected int m_FlagRefresherMaxDuration = GameConstants.REFRESHER_MAX_DURATION_DEFAU
■
■void TerritoryFlag()
■{
■■m_RefresherActive = false;
■■m_RefresherActiveLocal = false;
■■m_RefresherInitialized = false;
■■m_RefresherTimeRemaining = 0;
■■
■■if ( GetCEApi() )
■■{
```

```
// ---------------------------------------------
// ------ TotemKit.c - START --------------------
// ---------------------------------------------


class TerritoryFlagKit extends KitBase
{
■override bool PlacementCanBeRotated()
■{
■■return false;
■}
■
■override bool DoPlacingHeightCheck()
■{
■■return true;
■}
■
■override float HeightCheckOverride()
■{
■■return 11.3;
■}

■override bool CanReceiveAttachment(EntityAI attachment, int slotId)
■{
■■if ( !super.CanReceiveAttachment(attachment, slotId) )
■■■return false;
■■
■■ItemBase att = ItemBase.Cast(GetInventory().FindAttachment(slotId));
■■if (att)
■■■return false;
■■
■■return true;
■}
■
■//=================================================================
■// ADVANCED PLACEMENT
■//=================================================================■■■
■■
■override void OnPlacementComplete( Man player, vector position = "0 0 0", vector orientation = "0 0 0" )
■{
■■super.OnPlacementComplete( player, position, orientation );
■■
■■if ( GetGame().IsServer() )
■■{
■■■//Create TerritoryFlag
■■■TerritoryFlag totem = TerritoryFlag.Cast( GetGame().CreateObjectEx( "TerritoryFlag", GetPosition(),
■■■totem.SetPosition( position );
■■■totem.SetOrientation( orientation );
■■■
■■■//make the kit invisible, so it can be destroyed from deploy UA when action ends
■■■HideAllSelections();
■■■
■■■SetIsDeploySound( true );
■■}
■}

■override void DisassembleKit(ItemBase item)
■{
■■if (!IsHologram())
■■{
■■■ItemBase stick = ItemBase.Cast(GetGame().CreateObjectEx("WoodenStick",GetPosition(),ECE_PLA
■■■MiscGameplayFunctions.TransferItemProperties(this, stick);
■■■stick.SetQuantity(3);
```

```
// ---------------------------------------------
// ------ Toxicity.c - START --------------------
// ---------------------------------------------


class ToxicityMdfr: ModifierBase
{
■private const float■ ■TOXICITY_CLEANUP_PER_SEC = 1;
■private const float ■VOMIT_THRESHOLD = 70;
■
■override void Init()
■{
■■m_TrackActivatedTime = false;
■■m_ID ■■■■■= eModifiers.MDF_TOXICITY;
■■m_TickIntervalInactive ■= DEFAULT_TICK_TIME_INACTIVE;
■■m_TickIntervalActive ■= DEFAULT_TICK_TIME_ACTIVE;
■■DisableDeactivateCheck();
■}■

■override bool ActivateCondition(PlayerBase player)
■{
■■return true;
■}

■override bool DeactivateCondition(PlayerBase player)
■{
■■return false;
■}

■override void OnTick(PlayerBase player, float deltaT)
■{■
■■player.GetStatToxicity().Add( -TOXICITY_CLEANUP_PER_SEC * deltaT );
■■if( player.GetStatToxicity().Get() > VOMIT_THRESHOLD )
■■{
■■■SymptomBase symptom = player.GetSymptomManager().QueueUpPrimarySymptom(SymptomIDs.S
■■
■■■if( symptom )
■■■{
■■■■symptom.SetDuration(Math.RandomIntInclusive(4,8));
■■■}
■■}
■}
■
■override void OnReconnect(PlayerBase player)
■{

■}
■
■override void OnActivate(PlayerBase player)
■{
■■
■}
};


// ---------------------------------------------
// ------ Toxicity.c - END ----------------------
// ---------------------------------------------
// ---------------------------------------------
// ------ TrackSuitJacket_ColorBase.c - START --------------------
// ---------------------------------------------


class TrackSuitJacket_ColorBase extends Clothing
{
■override void SetActions()
■{
■■super.SetActions();
■■AddAction(ActionWringClothes);
■}
};
class TrackSuitJacket_Black extends TrackSuitJacket_ColorBase {};
class TrackSuitJacket_Blue extends TrackSuitJacket_ColorBase {};
class TrackSuitJacket_Green extends TrackSuitJacket_ColorBase {};
class TrackSuitJacket_LightBlue extends TrackSuitJacket_ColorBase {};
class TrackSuitJacket_Red extends TrackSuitJacket_ColorBase {};


// ---------------------------------------------
// ------ TrackSuitJacket_ColorBase.c - END ----------------------
// ---------------------------------------------


// ---------------------------------------------
// ------ TrackSuitPants_ColorBase.c - START --------------------
// ---------------------------------------------


class TrackSuitPants_ColorBase extends Clothing
{
■override void SetActions()
■{
■■super.SetActions();
■■AddAction(ActionWringClothes);
■}
};
class TrackSuitPants_Black extends TrackSuitPants_ColorBase {};
class TrackSuitPants_Blue extends TrackSuitPants_ColorBase {};
class TrackSuitPants_Green extends TrackSuitPants_ColorBase {};
class TrackSuitPants_Red extends TrackSuitPants_ColorBase {};
class TrackSuitPants_LightBlue extends TrackSuitPants_ColorBase {};


// ---------------------------------------------
// ------ TrackSuitPants_ColorBase.c - END ----------------------
// ---------------------------------------------


// ---------------------------------------------
// ------ TransferValues.c - START --------------------
// ---------------------------------------------


class TransferValues extends Managed
{
■const int TYPE_HEALTH ■= 1;
■const int TYPE_BLOOD ■= 2;
■
■const float VALUE_CHECK_INTERVAL■ ■= 5;
■const float SENSITIVTY_PERCENTAGE ■■= 1;//how much the value needs to change up/down from p
```

```
// ----------------------------------------------
// ------ TransmitterBase.c - START --------------------
// ----------------------------------------------


//TRANSMITTER BASE
class TransmitterBase extends ItemTransmitter
{
■//Sounds
■string SOUND_RADIO_TURNED_ON ■■= "";

■protected EffectSound m_SoundLoop;
■
■// --- SYSTEM EVENTS
■override void OnStoreSave( ParamsWriteContext ctx )
■{
■■super.OnStoreSave( ctx );
■■
■■//store tuned frequency
■■ctx.Write( GetTunedFrequencyIndex() );
■}
■
■override bool OnStoreLoad( ParamsReadContext ctx, int version )
■{
■■if ( !super.OnStoreLoad( ctx, version ) )
■■■return false;
■■
■■//--- Transmitter data ---
■■//load and set tuned frequency
■■int tuned_frequency_idx;
■■if ( !ctx.Read( tuned_frequency_idx ) )
■■{
■■■SetFrequencyByIndex( 0 );■■//set default
■■■return false;
■■}
■■SetFrequencyByIndex( tuned_frequency_idx );
■■//---
■■
■■return true;
■}■
■
■override bool IsTransmitter()
■{
■■return true;
■}
■
■//--- ACTIONS
■void SetNextFrequency( PlayerBase player = NULL )
■{
■■SetNextChannel();
■■
■■/*
■■if ( player )
■■{
■■■DisplayRadioInfo( GetTunedFrequency().ToString(), player );
■■}
■■*/
■}
■
■//--- HUD
■/*
■protected Hud GetHud( PlayerBase player )
■{
```

```
// ----------------------------------------------
// ------ Transport.c - START --------------------
// ----------------------------------------------


/*!
■Base native class of all vehicles in game.
*/
class Transport extends EntityAI
{
■ref TIntArray m_SingleUseActions;
■ref TIntArray m_ContinuousActions;
■ref TIntArray m_InteractActions;

■void Transport()
■{
■}

■override int GetMeleeTargetType()
■{
■■return EMeleeTargetType.NONALIGNABLE;
■}


■//! Synchronizes car's state in case the simulation is not running.
■proto native void Synchronize();


■//! Returns crew capacity of this vehicle.
■proto native int CrewSize();

■//! Returns crew member index based on action component index.
■//! -1 is returned when no crew position corresponds to given component index.
■proto native int CrewPositionIndex( int componentIdx );

■//! Returns crew member index based on player's instance.
■//! -1 is returned when the player is not isnide.
■proto native int CrewMemberIndex( Human player );

■//! Returns crew member based on position index.
■//! Null can be returned if no Human is present on the given position.
■proto native Human CrewMember( int posIdx );

■//! Reads entry point and direction into vehicle on given position in model space.
■proto void CrewEntry( int posIdx, out vector pos, out vector dir );

■//! Reads entry point and direction into vehicle on given position in world space.
■proto void CrewEntryWS( int posIdx, out vector pos, out vector dir );

■//! Returns crew transformation indside vehicle in model space
■proto void CrewTransform( int posIdx, out vector mat[4] );

■//! Returns crew transformation indside vehicle in world space
■proto void CrewTransformWS( int posIdx, out vector mat[4] );

■//! Performs transfer of player from world into vehicle on given position.
■proto native void CrewGetIn( Human player, int posIdx );

■//! Performs transfer of player from vehicle into world from given position.
■proto native Human CrewGetOut( int posIdx );
■
■//! Handles death of player in vehicle and awakes its physics if needed
■proto native void CrewDeath( int posIdx );
```

```
// ---------------------------------------------
// ------ TransportInventory.c - START --------------------
// ---------------------------------------------


/**@class■■TransportInventory
 * @brief■■inventory for transport ("man cargo")
 **/
class TransportInventory : GameInventory
{
};





// ---------------------------------------------
// ------ TransportInventory.c - END ----------------------
// ---------------------------------------------


// ---------------------------------------------
// ------ TrapBase.c - START --------------------
// ---------------------------------------------


enum SoundTypeTrap
{
■ACTIVATING■■■= 5,
}

class TrapBase extends ItemBase
{
■#ifdef SERVER
■protected const int SPAWN_FLAGS ■■■■= ECE_CREATEPHYSICS;
■#else
■protected const int SPAWN_FLAGS ■■■■= ECE_LOCAL;
■#endif
■
■protected const int DAMAGE_TRIGGER_MINE ■■= 75;
■protected const float UPDATE_TIMER_INTERVAL ■= 0.05;

■int   m_InitWaitTime; ■■■//After this time after deployment, the trap is activated
■bool m_NeedActivation;■■■//If activation of trap is needed
■float m_DefectRate; ■■■//Added damage after trap activation
■float m_DamagePlayers; ■■■//How much damage player gets when caught
■float m_DamageOthers; ■■■//How much damage player gets when caught

■bool m_AddActivationDefect;■■// Damage trap after activation
■bool m_AddDeactivationDefect;■// Damage trap after deactivation
■protected bool m_IsActive;■■// True means that the trap is ready to detonate
■protected bool m_IsInProgress;

■protected bool m_Disarmed = false; //! DEPRECATED Used for explosive traps to prevent detonation aft

■bool m_WasActivatedOrDeactivated;

■string m_AnimationPhaseGrounded;
■string m_AnimationPhaseSet;
■string m_AnimationPhaseTriggered;

■string m_InfoSetup;
■string m_InfoDeactivated;
■string m_InfoDamageManipulation;
```

```
// ---------------------------------------------
// ------ TrapSpawnBase.c - START --------------------
// ---------------------------------------------


class TrapSpawnBase extends ItemBase
{
■int   ■■■■■■■m_InitWaitTime; ■■■//After this time after deployment, the trap is activated
■int  ■■■■■■■m_UpdateWaitTime; ■■■//Time when timer will be called again until success or not succ
■float ■■■■■■■m_DefectRate; ■■■■//Added damage after trap activation
■bool ■■■■■■■m_IsFoldable;
■bool■■■■■■■m_CanCatch = false;
■float ■■■■■■■m_MinimalDistanceFromPlayersToCatch;
■
■float■■■■■■■m_BaitCatchProb; // The probability (0-100) to catch something when bait is attached
■float■■■■■■■m_NoBaitCatchProb; // The probability (0-100) to catch something when no bait is attach
■float■■■■■■■m_FinalCatchProb; // The selected catch probability (0-100) -> Will be overriden no need
■
■vector ■■■■■■■m_PreyPos; // The position where prey will be spawned -> Will be overriden later
■
■protected EntityAI■■■■m_Bait;

■protected bool ■■■■■m_IsActive;
■protected bool ■■■■■m_IsDeployed;
■protected bool ■■■■■m_IsInProgress;
■
■ref Timer ■■■■■■m_Timer;
■
■string ■■■■■■■m_AnimationPhaseSet;
■string ■■■■■■■m_AnimationPhaseTriggered;
■string ■■■■■■■m_AnimationPhaseUsed;

■const string ■■■■■m_PlaceableWaterType //! DEPRECATED
■protected ref array<string>■■m_PlaceableWaterSurfaceList;

■bool m_WaterSurfaceForSetup;■//if trap can be installed on water surface (cannot be detected via getsu
■ref multiMap<string, float>■m_CatchesPond;■//array of catches that can be catched in trap - string key,
■ref multiMap<string, float>■m_CatchesSea;■//array of catches that can be catched in trap - string key, fl
■ref multiMap<string, float>■m_CatchesGroundAnimal;■//array of catches that can be catched in trap - st
■
■protected ref EffectSound ■m_DeployLoopSound;
■
■// ===========================================================
■// ====================  DEPRECATED  ===========================
■// ===========================================================
■ref Timer ■■■■■■m_PrevTimer;
■bool ■■■■■■■m_NeedInstalation;
■bool ■■■■■■■m_BaitNeeded;
■bool ■■■■■■■m_IsUsable;
■private ItemBase ■■■■m_Catch;
■ref Timer ■■■■■■m_AlignCatchTimer;
■string ■■■■■■■m_InfoSetup;
■// ===========================================================
■
■void TrapSpawnBase()
■{
■■m_DefectRate ■■■■■■■= 10; ■■■//Added damage after trap activation
■■m_UpdateWaitTime ■■■■■■= 10;
■■m_InitWaitTime ■■■■■■■= 10;
■■m_NeedInstalation ■■■■■■= true;
■■m_BaitNeeded ■■■■■■■= true;
■■m_IsFoldable ■■■■■■■= false;
■■m_MinimalDistanceFromPlayersToCatch ■= 0;
```

```
// ---------------------------------------------
// ------ TrapTrigger.c - START --------------------
// ---------------------------------------------


//! Trigger used by traps
class TrapTrigger : Trigger
{
■TrapBase m_ParentObj;
■
■#ifdef DEVELOPER
■bool m_CanSendDbg = true;
■#endif
■
■void SetParentObject( TrapBase obj )
■{
■■if (GetGame().IsServer())
■■{
■■■m_ParentObj = obj;
■■}
■}
■
■override protected bool CanAddObjectAsInsider(Object object)
■{
■■return object.IsInherited(EntityAI) && m_ParentObj && m_ParentObj.IsActive();
■}
■
■override protected void OnEnterServerEvent(TriggerInsider insider)
■{
■■#ifdef DEVELOPER
■■m_CanSendDbg = false;
■■#endif
■■
■■if (m_ParentObj && m_ParentObj.IsActive())
■■{
■■■m_ParentObj.SnapOnObject(EntityAI.Cast(insider.GetObject()));
■■}
■}
■
■override protected void OnLeaveServerEvent(TriggerInsider insider)
■{
■■if (m_ParentObj && m_ParentObj.IsActive())
■■{
■■■m_ParentObj.RemoveFromObject(EntityAI.Cast(insider.GetObject()));
■■}
■}
■
■#ifdef DEVELOPER
■
■override void DebugSendDmgTrigger()
■{
■■if ( m_CanSendDbg )
■■■super.DebugSendDmgTrigger();
■}
■
■#endif
}

// Used for tripwire type traps, where we want to allow players to go through
class TripWireTrigger : TrapTrigger
{
■override protected void OnEnterServerEvent( TriggerInsider insider )
■{
```

```
// ---------------------------------------------
// ------ Trap_Bear.c - START --------------------
// ---------------------------------------------


class BearTrap extends TrapBase
{
■static const int RAYCAST_SOURCES_COUNT = 5;
■// Raycasts start positions:
■// Positions are local to model. Vertical offset prevents ground collision.
■static const vector m_RaycastSources[RAYCAST_SOURCES_COUNT] = {
■■"0.0 0.1 0.0",■// center
■■"0.2 0.1 0.2",■// north east
■■"-.2 0.1 0.2",■// north west
■■"0.2 0.1 -0.2",■// south east
■■"-0.2 0.1 -0.2"■// south west
■};
■
■void BearTrap()
■{
■■m_DamagePlayers ■■■= 5; ■■■■// How much damage player gets when caught
■■m_DamageOthers ■■■■= 5;         ■■// How much damage other entities(CreatureAI) gets when caug
■■m_DefectRate ■■■■= 0;
■■m_InitWaitTime ■■■■= 0; ■■■■// After this time after deployment, the trap is activated
■■m_AnimationPhaseGrounded ■= "placing";
■■m_AnimationPhaseSet ■■= "BearTrap_Set";
■■m_AnimationPhaseTriggered ■= "placing";
■}
■
■override bool CanBeDisarmed()
■{
■■return true;
■}
■
■override void EEHealthLevelChanged(int oldLevel, int newLevel, string zone)
■{
■■super.EEHealthLevelChanged(oldLevel, newLevel, zone);

■■if (GetGame().IsServer())
■■{
■■■if (newLevel == GameConstants.STATE_RUINED)
■■■{
■■■■SetInactive();
■■■}
■■}
■}
■
■override void CreateTrigger()
■{
■■super.CreateTrigger();

■■vector mins■■= "-0.1 -0.05 -0.1";
■■vector maxs ■= "0.1 0.4 0.1";

■■m_TrapTrigger.SetOrientation(GetOrientation());
■■m_TrapTrigger.SetExtents(mins, maxs);
■■m_TrapTrigger.SetParentObject(this);
■}
■
■override void OnUpdate(EntityAI victim)
■{
■■if (victim && victim.IsInherited(CarScript))
■■{
```

```
// ---------------------------------------------
// ------ Trap_FishNet.c - START --------------------
// ---------------------------------------------


class Trap_FishNet extends TrapSpawnBase
{
■void Trap_FishNet()
■{
■■m_DefectRate = 15; ■■■//Added damage after trap activation
■■
■■m_InitWaitTime = Math.RandomFloat(900,1500);
■■m_UpdateWaitTime = 30;
■■m_IsFoldable = true;
■■
■■m_MinimalDistanceFromPlayersToCatch ■= 15;
■■
■■m_BaitCatchProb ■■■■■■= 85;
■■m_NoBaitCatchProb■■■■■■= 15;

■■m_AnimationPhaseSet ■■■■■= "inventory";
■■m_AnimationPhaseTriggered ■■■■= "placing";
■■m_AnimationPhaseUsed ■■■■■= "triggered";
■■
■■m_WaterSurfaceForSetup = true;

■■m_CatchesPond = new multiMap<string, float>;
■■m_CatchesPond.Insert("Carp",1);
■■
■■m_CatchesSea = new multiMap<string, float>;
■■m_CatchesSea.Insert("Mackerel",1);
■}
■
■override void OnVariablesSynchronized()
■{
■■super.OnVariablesSynchronized();
■■■■
■■if ( IsPlaceSound() )
■■{
■■■PlayPlaceSound();
■■}
■}
■
■//================================================================
■//======================== CATCHING ========================
■//================================================================
■
■override void SpawnCatch()
■{
■■super.SpawnCatch();
■■
■■if ( m_CanCatch )
■■{
■■■multiMap<string, float>■catches;
■■■vector pos = GetPosition();
■■■
■■■ItemBase catch;
■■■// Select catch type depending on water type ( FRESH VS SALT )
■■■if ( GetGame().SurfaceIsSea( pos[0], pos[2] ) )
■■■{
■■■■catches = m_CatchesSea;
■■■}
■■■else if ( GetGame().SurfaceIsPond( pos[0], pos[2] ) )
```

```
// ----------------------------------------------
// ------ Trap_LandMine.c - START --------------------
// ----------------------------------------------


enum SoundTypeMine
{
■DISARMING = 0
}

class LandMineTrap extends TrapBase
{
■protected ref EffectSound m_TimerLoopSound;
■protected ref EffectSound m_DisarmingLoopSound;
■protected ref Timer m_DeleteTimer;
■
■private const int BROKEN_LEG_PROB ■= 90;
■private const int BLEED_SOURCE_PROB = 50;
■private const int MAX_BLEED_SOURCE ■= 1;
■
■void LandMineTrap()
■{
■■m_DefectRate ■■■= 15;
■■m_DamagePlayers ■■= 0; ■■■//How much damage player gets when caught
■■m_InitWaitTime ■■■= 10; ■■■//After this time after deployment, the trap is activated
■■m_InfoActivationTime ■= string.Format("#STR_LandMineTrap0%1#STR_LandMineTrap1", m_InitWait
■■
■■m_AddDeactivationDefect = true;
■■
■■//Order is important and must match clothing array in DamageClothing method
■■m_ClothingDmg = new array<int>;
■■m_ClothingDmg.Insert(60); ■//Trousers
■■m_ClothingDmg.Insert(100);■//BackPack
■■m_ClothingDmg.Insert(40);■//Vest
■■m_ClothingDmg.Insert(10);■//HeadGear
■■m_ClothingDmg.Insert(10);■//Mask
■■m_ClothingDmg.Insert(40);■//Body
■■m_ClothingDmg.Insert(50);■//Feet
■■m_ClothingDmg.Insert(5);■//Gloves
■}
■
■void ~LandMineTrap()
■{
■■SEffectManager.DestroyEffect(m_TimerLoopSound);
■■SEffectManager.DestroyEffect(m_DisarmingLoopSound);
■}
■■
■override void StartActivate(PlayerBase player)
■{
■■super.StartActivate(player);
■■
■■if (!GetGame().IsDedicatedServer())
■■{
■■■EffectSound sound = SEffectManager.PlaySound("landmine_safetyPin_SoundSet", GetPosition(), 0,
■■■sound.SetAutodestroy( true );
■■■m_TimerLoopSound = SEffectManager.PlaySound("landmine_timer2_SoundSet", GetPosition(), 0, 0
■■}
■}
■
■override void OnActivatedByItem(notnull ItemBase item)
■{
■■SetHealth("", "", 0.0);
■■DeleteThis();
```

```
// ---------------------------------------------
// ------ Trap_RabbitSnare.c - START --------------------
// ---------------------------------------------


class Trap_RabbitSnare extends TrapSpawnBase
{
■void Trap_RabbitSnare()
■{
■■m_DefectRate = 15; ■■■//Added damage after trap activation
■■
■■m_InitWaitTime ■■■■■■■= Math.RandomInt(900, 1500);
■■m_UpdateWaitTime ■■■■■■= 30;
■■m_IsFoldable ■■■■■■■= true;
■■m_IsUsable ■■■■■■■■= true;
■■m_MinimalDistanceFromPlayersToCatch ■= 30;
■■
■■m_BaitCatchProb ■■■■■■= 85;
■■m_NoBaitCatchProb■■■■■■= 15;

■■m_AnimationPhaseSet ■■■■■= "inventory";
■■m_AnimationPhaseTriggered ■■■■= "placing";
■■m_AnimationPhaseUsed ■■■■■= "triggered";

■■m_WaterSurfaceForSetup = false;
■■
■■m_CatchesGroundAnimal = new multiMap<string, float>;
■■m_CatchesGroundAnimal.Insert("DeadRooster", 1);
■■m_CatchesGroundAnimal.Insert("DeadChicken_White", 1);
■■m_CatchesGroundAnimal.Insert("DeadChicken_Spotted", 1);
■■m_CatchesGroundAnimal.Insert("DeadChicken_Brown", 1);
■■// ALWAYS keep rabbit last as that is how it gets the rabbit in case of rabbit specific bait
■■m_CatchesGroundAnimal.Insert("DeadRabbit", 1);
■}
■
■override bool CanBePlaced(Man player, vector position)
■{
■■if (m_IsBeingPlaced)
■■■return true;

■■int liquidType;
■■string surfaceType;
■■g_Game.SurfaceUnderObject(PlayerBase.Cast(player).GetHologramLocal().GetProjectionEntity(), surf

■■return g_Game.IsSurfaceDigable(surfaceType);
■}
■
■override void OnVariablesSynchronized()
■{
■■super.OnVariablesSynchronized();
■■■■
■■if (IsPlaceSound())
■■{
■■■PlayPlaceSound();
■■}
■}
■
■override void SetupTrap()
■{
■■if ( GetGame().IsServer() )
■■{
■■■if ( GetHierarchyRootPlayer().CanDropEntity( this ) )
■■■{
```

```
// ---------------------------------------------
// ------ Trap_SmallFish.c - START --------------------
// ---------------------------------------------


class Trap_SmallFish extends TrapSpawnBase
{
■void Trap_SmallFish()
■{
■■m_DefectRate = 15; ■■■//Added damage after trap activation
■■
■■m_InitWaitTime = Math.RandomFloat(900,1500);
■■m_UpdateWaitTime = 30;

■■m_AnimationPhaseSet ■■■■■= "inventory";
■■m_AnimationPhaseTriggered ■■■■= "placing";
■■m_AnimationPhaseUsed ■■■■■= "triggered";
■■
■■m_MinimalDistanceFromPlayersToCatch ■= 15;
■■
■■m_BaitCatchProb ■■■■■■= 85;
■■m_NoBaitCatchProb■■■■■■= 15;

■■m_WaterSurfaceForSetup = true;

■■m_CatchesPond = new multiMap<string, float>;
■■m_CatchesPond.Insert("Bitterlings", 1);
■■
■■m_CatchesSea = new multiMap<string, float>;
■■m_CatchesSea.Insert("Sardines",1);
■}
■
■override void OnVariablesSynchronized()
■{
■■super.OnVariablesSynchronized();
■■■■
■■if ( IsPlaceSound() )
■■{
■■■PlayPlaceSound();
■■}
■}
■
■override void SpawnCatch()
■{
■■super.SpawnCatch();
■■
■■if ( m_CanCatch )
■■{
■■■// We get the prey position for spawning
■■■multiMap<string, float>■catches;
■■■vector pos = GetPosition();
■■■
■■■ItemBase catch;
■■■// Select catch type depending on water type ( FRESH VS SALT )
■■■if ( GetGame().SurfaceIsSea( pos[0], pos[2] ) )
■■■{
■■■■catches = m_CatchesSea;
■■■}
■■■else if ( GetGame().SurfaceIsPond( pos[0], pos[2] ) )
■■■{
■■■■catches = m_CatchesPond;
■■■}
```

```
// ---------------------------------------------
// ------ Trap_TripWire.c - START --------------------
// ---------------------------------------------


// Wire type is used in the case of decrafting to give back the correct base Ingredient
enum eWireMaterial
{
■WIRE ■■= 0,
■BARBED_WIRE = 1,
■ROPE ■■= 2
}

class TripwireTrap : TrapBase
{
■// Current state of the tripwire
■static const int ■FOLDED = 3;
■static const int ■DEPLOYED = 2;
■static const int ■TRIGGERED = 1;

■int ■■m_State = FOLDED;
■private int m_WireMaterial;
■
■protected bool   m_ResultOfAdvancedPlacing;
■protected vector m_TriggerPosition;
■protected vector m_TriggerOrientation;
■
■void TripwireTrap()
■{
■■m_DamagePlayers = 0; ■■■//How much damage player gets when caught
■■m_InitWaitTime = 0; ■■■//After this time after deployment, the trap is activated
■■m_DefectRate = 15;
■■m_NeedActivation = false;
■■m_AnimationPhaseGrounded = "inventory";
■■m_AnimationPhaseSet = "placing";
■■m_AnimationPhaseTriggered = "triggered";
■■m_InfoActivationTime = string.Format("#STR_TripwireTrap0%1#STR_TripwireTrap1", m_InitWaitTime.
■
■■RegisterNetSyncVariableInt("m_State");
■}

■override void OnVariablesSynchronized()
■{
■■super.OnVariablesSynchronized();
■■■■
■■if ( IsPlaceSound() )
■■{
■■■PlayPlaceSound();
■■}
■}
■
■override void OnStoreSave(ParamsWriteContext ctx)
■{
■■super.OnStoreSave(ctx);
■■
■■ctx.Write( m_State );
■}
■
■//---------------------------------------------------------------
■override bool OnStoreLoad(ParamsReadContext ctx, int version)
■{
■■if ( !super.OnStoreLoad(ctx, version) )
■■■return false;
```

```
// ---------------------------------------------
// ------ Trees.c - START --------------------
// ---------------------------------------------


//------------------------------------
//Birch tree baseclasses (white bark)
class TreeHard_BetulaPendula: TreeHard
{
■
};

class TreeSoft_BetulaPendula_Base: TreeSoft
{
■
};

//----------------------------------------------
//TreeSoft
class TreeSoft_t_BetulaPendula_1f: TreeSoft_BetulaPendula_Base
{
■
};
class TreeSoft_t_BetulaPendula_1fb: TreeSoft_BetulaPendula_Base
{
■
};
class TreeSoft_t_BetulaPendula_1s: TreeSoft_BetulaPendula_Base
{
■
};
class TreeSoft_t_BetulaPendula_2f: TreeSoft_BetulaPendula_Base
{
■
};
class TreeSoft_t_BetulaPendula_2fb: TreeSoft_BetulaPendula_Base
{
■
};
class TreeSoft_t_BetulaPendula_2fc: TreeSoft_BetulaPendula_Base
{
■
};
class TreeSoft_t_BetulaPendula_2w: TreeSoft_BetulaPendula_Base
{
■
};
class TreeSoft_t_FagusSylvatica_1f: TreeSoft
{
■
};
class TreeSoft_t_FagusSylvatica_1fc: TreeSoft
{
■
};
class TreeSoft_t_FagusSylvatica_1fd: TreeSoft
{
■
};
class TreeSoft_t_FagusSylvatica_1s: TreeSoft
{
■
};
```

```
// ---------------------------------------------
// ------ Tremor.c - START --------------------
// ---------------------------------------------


class TremorMdfr: ModifierBase
{
■private const float■ ■TREMOR_DECREMENT_PER_SEC = 0.008;

■override void Init()
■{
■■m_TrackActivatedTime = false;
■■m_ID ■■■■■= eModifiers.MDF_TREMOR;
■■m_TickIntervalInactive ■= DEFAULT_TICK_TIME_INACTIVE;
■■m_TickIntervalActive ■= DEFAULT_TICK_TIME_ACTIVE;
■}

■override bool ActivateCondition(PlayerBase player)
■{
■■if ( player.GetStatTremor().Get() > player.GetStatTremor().GetMin() )
■■{
■■■return true;
■■}
■■else
■■{
■■■return false;
■■}
■}

■override bool DeactivateCondition(PlayerBase player)
■{
■■if ( player.GetStatTremor().Get() == player.GetStatTremor().GetMin() )
■■{
■■■return true;
■■}
■■else
■■{
■■■return false;
■■}
■}

■override void OnTick(PlayerBase player, float deltaT)
■{
■■player.GetStatTremor().Add( (TREMOR_DECREMENT_PER_SEC*deltaT) );
■■
■■//Mirek: SetShakeBodyFactor is removed now, because it worked only on legacy animation system
■■//player.SetShakeBodyFactor(player.GetStatTremor().Get());
■■//_person SetBodyShaking tremor; ASK GAMEPLAY PROGRAMMERS TO EXPOSE THIS ENGINE F
■■//PrintString( "Tremor:" + ToString(tremor) );
■}
};


// ---------------------------------------------
// ------ Tremor.c - END ---------------------
// ---------------------------------------------
// ---------------------------------------------
// ------ TrialService.c - START --------------------
// ---------------------------------------------



//! TrialService is used to query if the game is trial version or not
class TrialService
{
■/*!
■■Xbox: Retrieves if the game license is trial
■■■
■■@param bIsSimulator switch in between the testing simulator for trial and the retail version.
■■■■■■■If the game is in retail mode it automatically ignores this switch and
■■■■■■■uses the correct app runtime version.
■■@return boolean if the game is in trial mode.■■
■*/
■proto native bool IsGameTrial(bool bIsSimulator);
■
■/*!
■■Xbox: Retrieves if the game license is active
■■■
■■@param bIsSimulator switch in between the testing simulator for trial and the retail version.
■■■■■■■If the game is in retail mode it automatically ignores this switch and
■■■■■■■uses the correct app runtime version.
■■@return boolean if the game is in trial mode.■■
■*/
■proto native bool IsGameActive(bool bIsSimulator);
};



// ---------------------------------------------
// ------ TrialService.c - END ----------------------
// ---------------------------------------------


// ---------------------------------------------
// ------ Trigger.c - START --------------------
// ---------------------------------------------


//! The object which is in a trigger and its metadata
class TriggerInsider
{
■ref OLinkT insider; // DEPRECATED
■
■//! Object that data belongs to
■protected Object m_Object;
■
■//! Last time the object was seen in ms
■int timeStamp;
■
■//! Time the object was first seen in seconds
■float timeEntered;
■
■//! Last time the object was updated in seconds, is used for calculating deltaTime
■float lastUpdated;
■
■void TriggerInsider(Object obj)
■{
■■insider = new OLinkT(obj);
■■m_Object = obj;
```

```
// ---------------------------------------------
// ------ TriggerCarrierBase.c - START --------------------
// ---------------------------------------------


class UndergroundTriggerCarrierBase extends ScriptedEntity
{
■int m_TriggerIndex = -1;
■
■void SetIndex(int index)
■{
■■m_TriggerIndex = index;
■■SetSynchDirty();
■}
}


// ---------------------------------------------
// ------ TriggerCarrierBase.c - END ----------------------
// ---------------------------------------------


// ---------------------------------------------
// ------ TriggerEffectManager.c - START --------------------
// ---------------------------------------------


/*
DISCLAIMER: may undergo some changes in the course of 1.14 experimental stage.
*/

// Used for Effect based triggers ( such as Contaminated areas )
// Allows us to unify these triggers and treat Enter / Exit events as if all triggers were one
class TriggerEffectManager
{
■static ref TriggerEffectManager m_This;
■
■// Player map will only be handled client side as it is most relevant
■ref map<string, ref map<PlayerBase, int>> m_PlayerInsiderMap; // Used to keep track of which players a

■void TriggerEffectManager()
■{
■■m_PlayerInsiderMap = new map<string, ref map<PlayerBase, int>>;
■}
■
■static void DestroyInstance()
■{
■■m_This = null;
■}
■
■// This is a SINGLETON so, if we can't get the instance, we create it
■static TriggerEffectManager GetInstance()
■{
■■if ( !m_This )
■■■m_This = new TriggerEffectManager();

■■return m_This;
■}
■
■void RegisterTriggerType( EffectTrigger effectTrigger )
■{
■■if ( !m_PlayerInsiderMap.Contains( effectTrigger.GetType() ) )
■■{
```

```
// ----------------------------------------------
// ------ TriggerEvents.c - START --------------------
// ----------------------------------------------


//! Events API for triggers, keep in sync with AreaDamageEvents for consistency
//! NOTE: The deltaTime passed in is the one for the Insider specifically.
class TriggerEvents : ScriptedEntity
{■
■/** \name OnEnter
 ■■Called when an object enters the trigger
■*/
■///@{
■protected void Enter(TriggerInsider insider)
■{
■■#ifdef TRIGGER_DEBUG_BASIC
■■Debug.TriggerLog(GetDebugName(insider.GetObject()), "TriggerEvents", "", "Enter", GetDebugName(t
■■#endif
■■
■■OnEnterBeginEvent(insider);
■■
■■if ( GetGame().IsServer() )
■■■OnEnterServerEvent(insider);
■■else
■■■OnEnterClientEvent(insider);
■■
■■OnEnterEndEvent(insider);
■}■
■protected void OnEnterBeginEvent(TriggerInsider insider) {}
■protected void OnEnterServerEvent(TriggerInsider insider) {}
■protected void OnEnterClientEvent(TriggerInsider insider) {}
■protected void OnEnterEndEvent(TriggerInsider insider) {}
■///@}
■
■/** \name OnStayStart
 ■■Called at the beginning of an update loop
■*/
■///@{
■protected void StayStart(int nrOfInsiders)
■{
■■#ifdef TRIGGER_DEBUG_SPAM
■■Debug.TriggerLog(nrOfInsiders.ToString(), "TriggerEvents", "", "StayStart", GetDebugName(this));
■■#endif
■■
■■OnStayStartBeginEvent(nrOfInsiders);
■■
■■if ( GetGame().IsServer() )
■■■OnStayStartServerEvent(nrOfInsiders);
■■else
■■■OnStayStartClientEvent(nrOfInsiders);
■■
■■OnStayStartEndEvent(nrOfInsiders);
■}■
■protected void OnStayStartBeginEvent(int nrOfInsiders) {}
■protected void OnStayStartServerEvent(int nrOfInsiders) {}
■protected void OnStayStartClientEvent(int nrOfInsiders) {}
■protected void OnStayStartEndEvent(int nrOfInsiders) {}
■///@}
■
■/** \name OnStay
 ■■Called in the update loop
■*/
■///@{
```

```
// ---------------------------------------------
// ------ Tripod.c - START --------------------
// ---------------------------------------------


class TripodBase : ItemBase
{
■override bool HasProxyParts()
■{
■■return true;
■}
■
■override bool CanDetachAttachment( EntityAI parent )
■{
■■FireplaceBase fireplace = FireplaceBase.Cast(parent);
■■if(fireplace)
■■{
■■■if ( fireplace.GetCookingEquipment() != null )
■■■{
■■■■return false;
■■■}
■■}
■■return true;
■}

■override bool CanSwapEntities(EntityAI otherItem, InventoryLocation otherDestination, InventoryLocation
■{
■■if (GetHierarchyParent() && GetHierarchyParent().IsFireplace() && otherItem)
■■{
■■■if (otherItem.IsInherited(Pot) || otherItem.IsInherited(Cauldron))
■■■{
■■■■return false;
■■■}
■■}
■■return true;
■}
■
■
■override void OnDebugSpawn()
■{
■■HideAllSelections();
■■ShowSelection( "Deployed" );
■}
■
■override void SetActions()
■{
■■super.SetActions();

■■AddAction(ActionAttach);
■■AddAction(ActionDetach);
■}
■
■override void OnWasAttached( EntityAI parent, int slot_id )
■{
■■super.OnWasAttached(parent, slot_id);
■■if (parent.IsFireplace())
■■{
■■■HideAllSelections();
■■■ShowSelection( "Deployed" );
■■}
■}
■
■override void OnWasDetached( EntityAI parent, int slot_id )
```

```
// ---------------------------------------------
// ------ TruckBattery.c - START --------------------
// ---------------------------------------------


class TruckBattery : VehicleBattery {}


// ---------------------------------------------
// ------ TruckBattery.c - END ----------------------
// ---------------------------------------------


// ---------------------------------------------
// ------ Truck_01FrontLight.c - START --------------------
// ---------------------------------------------


class Truck_01FrontLight extends CarLightBase
{
■void Truck_01FrontLight()
■{
■■m_SegregatedBrightness = 4;
■■m_SegregatedRadius = 75;
■■m_SegregatedAngle = 90;
■■m_SegregatedColorRGB = Vector(0.90, 0.90, 0.68);
■■
■■m_AggregatedBrightness = 8;
■■m_AggregatedRadius = 100;
■■m_AggregatedAngle = 120;
■■m_AggregatedColorRGB = Vector(0.90, 0.90, 0.68);
■■
■■FadeIn(0.1);
■■SetFadeOutTime(0.05);
■■
■■SegregateLight();
■}
}


// ---------------------------------------------
// ------ Truck_01FrontLight.c - END ----------------------
// ---------------------------------------------


// ---------------------------------------------
// ------ Truck_01RearLight.c - START --------------------
// ---------------------------------------------


class Truck_01RearLight extends CarRearLightBase
{
■void Truck_01RearLight()
■{
■■// Brake light only
■■m_SegregatedBrakeBrightness = 5;
■■m_SegregatedBrakeRadius = 6;
■■m_SegregatedBrakeAngle = 160;
■■m_SegregatedBrakeColorRGB = Vector(0.75, 0.16, 0.05);
■■
■■// Reverse light only
■■m_SegregatedBrightness = 2;
■■m_SegregatedRadius = 10;
```

```
// ---------------------------------------------
// ------ Truck_01_Base.c - START --------------------
// ---------------------------------------------


class Truck_01_Base extends CarScript
{
■protected ref UniversalTemperatureSource m_UTSource;
■protected ref UniversalTemperatureSourceSettings m_UTSSettings;
■protected ref UniversalTemperatureSourceLambdaEngine m_UTSLEngine;

■void Truck_01_Base()
■{
■■//m_dmgContactCoef ■■= 0.018;
■■m_enginePtcPos ■■■= "0 1.346 2.205";
■■
■■m_EngineStartOK ■■= "Truck_01_engine_start_SoundSet";
■■m_EngineStartBattery■= "Truck_01_engine_failed_start_battery_SoundSet";
■■m_EngineStartPlug ■■= "Truck_01_engine_failed_start_sparkplugs_SoundSet";
■■m_EngineStartFuel ■■= "Truck_01_engine_failed_start_fuel_SoundSet";
■■m_EngineStopFuel ■■= "Truck_01_engine_stop_fuel_SoundSet";

■■m_CarDoorOpenSound ■■= "Truck_01_door_open_SoundSet";
■■m_CarDoorCloseSound ■= "Truck_01_door_close_SoundSet";
■■
■■m_CarHornShortSoundName = "Truck_01_Horn_Short_SoundSet";
■■m_CarHornLongSoundName■= "Truck_01_Horn_SoundSet";
■■
■■SetEnginePos("0 1.4 2.25");
■}

■override void EEInit()
■{■■
■■super.EEInit();
■■
■■if (GetGame().IsServer() || !GetGame().IsMultiplayer())
■■{
 ■■■m_UTSSettings ■■■■■= new UniversalTemperatureSourceSettings();
■■■m_UTSSettings.m_ManualUpdate ■= true;
■■■m_UTSSettings.m_TemperatureMin■= 0;
■■■m_UTSSettings.m_TemperatureMax■= 30;
■■■m_UTSSettings.m_RangeFull■■= 0.5;
■■■m_UTSSettings.m_RangeMax■■= 2;
■■■m_UTSSettings.m_TemperatureCap■= 25;
■■■
■■■m_UTSLEngine■■■■■= new UniversalTemperatureSourceLambdaEngine();
■■■m_UTSource■■■■■■= new UniversalTemperatureSource(this, m_UTSSettings, m_UTSLEngine);
■■}■■
■}
■
■override void OnEngineStart()
■{
■■super.OnEngineStart();

■■if (GetGame().IsServer() || !GetGame().IsMultiplayer())
■■{
■■■m_UTSource.SetDefferedActive(true, 20.0);
■■}
■}
■
■override void OnEngineStop()
■{
■■super.OnEngineStop();
```

```
// ---------------------------------------------
// ------ Truck_01_Cargo.c - START --------------------
// ---------------------------------------------


class Truck_01_Cargo extends Truck_01_Chassis {};
/*
{
■override int GetAnimInstance()
■{
■■return VehicleAnimInstances.V3S;
■}

■override int GetSeatAnimationType( int posIdx )
■{
■■switch( posIdx )
■■{
■■case 0:
■■■return DayZPlayerConstants.VEHICLESEAT_DRIVER;
■■case 1:
■■■return DayZPlayerConstants.VEHICLESEAT_CODRIVER;
■■}

■■return 0;
■}
■
■override bool CrewCanGetThrough( int posIdx )
■{
■■CarDoor carDoor;
■■switch( posIdx )
■■{
■■■case 0:
■■■■Class.CastTo( carDoor, FindAttachmentBySlotName("V3SDriverDoors") );
■■■■if ( carDoor )
■■■■{
■■■■■if ( carDoor.GetAnimationPhase("DoorsSource") > 0.5 ) return true;
■■■■}
■■■■else
■■■■{
■■■■■return true;
■■■■}
■■■■return false;
■■■break;
■■■
■■■case 1:
■■■■Class.CastTo( carDoor, FindAttachmentBySlotName("V3SCoDriverDoors") );
■■■■if ( carDoor )
■■■■{
■■■■■if ( carDoor.GetAnimationPhase("DoorsSource") > 0.5 ) return true;
■■■■}
■■■■else
■■■■{
■■■■■return true;
■■■■}
■■■■return false;
■■■break;
■■}

■■return false;
■}

■override float OnSound( CarSoundCtrl ctrl, float oldValue )
■{
```

```
// ---------------------------------------------
// ------ Truck_01_Chassis.c - START --------------------
// ---------------------------------------------


class Truck_01_Chassis extends Truck_01_Base {};


// ---------------------------------------------
// ------ Truck_01_Chassis.c - END ----------------------
// ---------------------------------------------


// ---------------------------------------------
// ------ Truck_01_Covered.c - START --------------------
// ---------------------------------------------


class Truck_01_Covered extends Truck_01_Base {};

class Truck_01_Covered_Orange extends Truck_01_Covered
{
■override void OnDebugSpawn()
■{
■■SpawnUniversalParts();
■■SpawnAdditionalItems();
■■FillUpCarFluids();

■■GetInventory().CreateInInventory("Truck_01_Wheel");
■■GetInventory().CreateInInventory("Truck_01_Wheel");

■■GetInventory().CreateInInventory("Truck_01_WheelDouble");
■■GetInventory().CreateInInventory("Truck_01_WheelDouble");
■■GetInventory().CreateInInventory("Truck_01_WheelDouble");
■■GetInventory().CreateInInventory("Truck_01_WheelDouble");

■■GetInventory().CreateInInventory("Truck_01_Door_1_1_Orange");
■■GetInventory().CreateInInventory("Truck_01_Door_2_1_Orange");
■■GetInventory().CreateInInventory("Truck_01_Hood_Orange");
■■■
■■//-----IN CAR CARGO
■■GetInventory().CreateInInventory("Truck_01_Wheel");
■■GetInventory().CreateInInventory("Truck_01_Wheel");
■■GetInventory().CreateInInventory("Truck_01_WheelDouble");
■■GetInventory().CreateInInventory("Truck_01_WheelDouble");
■}
}

class Truck_01_Covered_Blue extends Truck_01_Covered
{
■override void OnDebugSpawn()
■{
■■SpawnUniversalParts();
■■SpawnAdditionalItems();
■■FillUpCarFluids();

■■GetInventory().CreateInInventory("Truck_01_Wheel");
■■GetInventory().CreateInInventory("Truck_01_Wheel");

■■GetInventory().CreateInInventory("Truck_01_WheelDouble");
■■GetInventory().CreateInInventory("Truck_01_WheelDouble");
■■GetInventory().CreateInInventory("Truck_01_WheelDouble");
■■GetInventory().CreateInInventory("Truck_01_WheelDouble");
```

```
// ---------------------------------------------
// ------ Truck_02.c - START --------------------
// ---------------------------------------------


class Truck_02 extends CarScript
{
■protected ref UniversalTemperatureSource m_UTSource;
■protected ref UniversalTemperatureSourceSettings m_UTSSettings;
■protected ref UniversalTemperatureSourceLambdaEngine m_UTSLEngine;

■void Truck_02()
■{
■■//m_dmgContactCoef = 0.018;
■■m_enginePtcPos = "0 1.346 2.205";
■}
■
■override void EEInit()
■{■■
■■super.EEInit();
■■
■■if (GetGame().IsServer() || !GetGame().IsMultiplayer())
■■{
 ■■■m_UTSSettings ■■■■■= new UniversalTemperatureSourceSettings();
■■■m_UTSSettings.m_ManualUpdate ■= true;
■■■m_UTSSettings.m_TemperatureMin■= 0;
■■■m_UTSSettings.m_TemperatureMax■= 30;
■■■m_UTSSettings.m_RangeFull■■= 0.5;
■■■m_UTSSettings.m_RangeMax■■= 2;
■■■m_UTSSettings.m_TemperatureCap■= 25;
■■■
■■■m_UTSLEngine■■■■■= new UniversalTemperatureSourceLambdaEngine();
■■■m_UTSource■■■■■■= new UniversalTemperatureSource(this, m_UTSSettings, m_UTSLEngine);
■■}■■
■}
■
■override void OnEngineStart()
■{
■■super.OnEngineStart();

■■if (GetGame().IsServer() || !GetGame().IsMultiplayer())
■■{
■■■m_UTSource.SetDefferedActive(true, 20.0);
■■}
■}
■
■override void OnEngineStop()
■{
■■super.OnEngineStop();

■■if (GetGame().IsServer() || !GetGame().IsMultiplayer())
■■{
■■■m_UTSource.SetDefferedActive(false, 10.0);
■■}
■}
■
■override void EOnPostSimulate(IEntity other, float timeSlice)
■{
■■if (GetGame().IsServer() || !GetGame().IsMultiplayer())
■■{
■■■if (m_UTSource.IsActive())
■■■{
■■■■m_UTSource.Update(m_UTSSettings, m_UTSLEngine);
```

```
// ---------------------------------------------
// ------ Trumpet.c - START --------------------
// ---------------------------------------------


class Trumpet : Rifle_Base
{
};


// ---------------------------------------------
// ------ Trumpet.c - END ----------------------
// ---------------------------------------------


// ---------------------------------------------
// ------ TShirt_ColorBase.c - START --------------------
// ---------------------------------------------


class TShirt_ColorBase extends Clothing
{
■override void SetActions()
■{
■■super.SetActions();
■■AddAction(ActionWringClothes);
■}
};
class TShirt_Beige extends TShirt_ColorBase {};
class TShirt_Black extends TShirt_ColorBase {};
class TShirt_Blue extends TShirt_ColorBase {};
class TShirt_Dyed extends TShirt_ColorBase {};
class TShirt_Green extends TShirt_ColorBase {};
class TShirt_OrangeWhiteStripes extends TShirt_ColorBase {};
class TShirt_Red extends TShirt_ColorBase {};
class TShirt_RedBlackStripes extends TShirt_ColorBase {};
class TShirt_White extends TShirt_ColorBase {};
class TShirt_Grey extends TShirt_ColorBase {};



// ---------------------------------------------
// ------ TShirt_ColorBase.c - END ----------------------
// ---------------------------------------------


// ---------------------------------------------
// ------ TShirt_Dyed.c - START --------------------
// ---------------------------------------------


/*
class TShirt_Dyed : Clothing
{
■void TShirt_Dyed()
■{
■■string color = GetItemVariableString("varColor"); //SYNCFAIL
■■if ( color != "" )
■■{
■■■SetObjectTexture (0,color);
■■■SetObjectTexture (1,color);
■■■SetObjectTexture (2,color);
■■}
```

```
// ---------------------------------------------
// ------ TShirt_White.c - START --------------------
// ---------------------------------------------


/*
class TShirt_White : Clothing
{
■void TShirt_White()
■{
■■string color = GetItemVariableString("varColor"); //SYNCFAIL
■■if ( color != "" )
■■{
■■■SetObjectTexture (0,color);
■■■SetObjectTexture (1,color);
■■■SetObjectTexture (2,color);
■■}
■}
■
■override bool IsClothing()
■{
■■return true;
■}
}
*/


// ---------------------------------------------
// ------ TShirt_White.c - END ----------------------
// ---------------------------------------------


// ---------------------------------------------
// ------ TTSKOBoots.c - START --------------------
// ---------------------------------------------


class TTSKOBoots extends Clothing {};


// ---------------------------------------------
// ------ TTSKOBoots.c - END ----------------------
// ---------------------------------------------


// ---------------------------------------------
// ------ TTsKOJacket_ColorBase.c - START --------------------
// ---------------------------------------------


class TTsKOJacket_ColorBase extends Clothing
{
■override void SetActions()
■{
■■super.SetActions();
■■AddAction(ActionWringClothes);
■}
};
class TTsKOJacket_Camo extends TTsKOJacket_ColorBase {};


// ---------------------------------------------
// ------ TTsKOJacket_ColorBase.c - END ----------------------
// ---------------------------------------------
// ------ TTSKOPants.c - START --------------------
// ---------------------------------------------


class TTSKOPants extends Clothing
{
■override void SetActions()
■{
■■super.SetActions();
■■AddAction(ActionWringClothes);
■}
};


// ---------------------------------------------
// ------ TTSKOPants.c - END ----------------------
// ---------------------------------------------


// ---------------------------------------------
// ------ TunaCan.c - START --------------------
// ---------------------------------------------


class TunaCan : Edible_Base
{
■override void Open()
■{
■■ReplaceEdibleWithNew("TunaCan_Opened");
■}
■
■override void SetActions()
■{
■■super.SetActions();
■■
■■AddAction(ActionOpen);
■}
■
■override bool IsOpen()
■{
■■return false;
■}
}



// ---------------------------------------------
// ------ TunaCan.c - END ----------------------
// ---------------------------------------------


// ---------------------------------------------
// ------ TunaCan_Opened.c - START --------------------
// ---------------------------------------------


class TunaCan_Opened: Edible_Base
{
■override bool CanDecay()
■{
■■return true;
■}
```

```
// ---------------------------------------------
// ------ TutorialsMenu.c - START --------------------
// ---------------------------------------------


class TutorialsMenu extends UIScriptedMenu
{
■protected const string PATH_MOUSEKEY = "Scripts/data/PageDataTutorials.json";
■protected const string PATH_X1_OLD = "Xbox/PageDataTutorials.json";
■protected const string PATH_X1_NEW = "Xbox/PageDataTutorialsAlternate.json";
■protected const string PATH_PS_OLD = "PS4/PageDataTutorials.json";
■protected const string PATH_PS_NEW = "PS4/PageDataTutorialsAlternate.json";
■
■protected string ■■■■■m_BackButtonTextID;
■
■protected Widget■■■■■m_InfoTextLeft;
■protected Widget■■■■■m_InfoTextRight;
■protected ButtonWidget■■■■m_Back;
■
■protected ImageWidget ■■■■m_ControlsLayoutImage;
■protected const int ■■■■TABS_COUNT = 4;
■protected ImageWidget ■■■■m_tab_images[TABS_COUNT];
■protected TabberUI■■■■■m_TabScript;
■protected ref TutorialKeybinds■■m_KeybindsTab;
■
■//==============================================
■// Init
■//==============================================
■override Widget Init()
■{
■■#ifdef PLATFORM_CONSOLE
■■layoutRoot■= GetGame().GetWorkspace().CreateWidgets("gui/layouts/new_ui/tutorials/xbox/tutorials.la
■■#else
■■layoutRoot■= GetGame().GetWorkspace().CreateWidgets("gui/layouts/new_ui/tutorials/pc/tutorials.lay
■■#endif
■■
■■m_InfoTextLeft■= layoutRoot.FindAnyWidget("InfoTextLeft");
■■m_InfoTextRight■= layoutRoot.FindAnyWidget("InfoTextRight");
■■
■■m_Back■■■= ButtonWidget.Cast(layoutRoot.FindAnyWidget("back"));
■■
■■layoutRoot.FindAnyWidget("Tabber").GetScript(m_TabScript);
■■m_TabScript.m_OnTabSwitch.Insert(DrawConnectingLines);
■■
■■#ifdef PLATFORM_CONSOLE
■■if (GetGame().GetInput().IsEnabledMouseAndKeyboard())
■■{
■■■m_KeybindsTab = new TutorialKeybinds(layoutRoot.FindAnyWidget("Tab_6"), this);
■■■m_TabScript.EnableTabControl(6, true);
■■}
■■#endif
■■
■■m_tab_images[0] = ImageWidget.Cast(layoutRoot.FindAnyWidget("MovementTabBackdropImageWidg
■■m_tab_images[1] = ImageWidget.Cast(layoutRoot.FindAnyWidget("WeaponsAndActionsBackdropImag
■■m_tab_images[2] = ImageWidget.Cast(layoutRoot.FindAnyWidget("InventoryTabBackdropImageWidge
■■m_tab_images[3] = ImageWidget.Cast(layoutRoot.FindAnyWidget("MenusTabBackdropImageWidget")
■■
■■UpdateControlsElements();
■■UpdateControlsElementVisibility();
■■
■■PPERequesterBank.GetRequester(PPERequester_TutorialMenu).Start(new Param1<float>(0.6));
■■DrawConnectingLines(0);
■■
```

```
// ---------------------------------------------
// ------ UAInput.c - START --------------------
// ---------------------------------------------



/** @file */


// constants for device (binding device) type determine:
//
//■EUAINPUT_DEVICE_KEYBOARD
//■EUAINPUT_DEVICE_MOUSE
//■EUAINPUT_DEVICE_KEYBOARDMOUSE
//■EUAINPUT_DEVICE_CONTROLLER
//■EUAINPUT_DEVICE_IR

// -------------------------------------------------------------------------
class UAIDWrapper
{
■private void UAIDWrapper() {}
■private void ~UAIDWrapper() {}
■
■proto native UAInput InputP();■■■// get input pointer
};

// -------------------------------------------------------------------------
class UAInput
{
■proto native int ID();■■■// return input index
■proto native int NameHash();■// return input hash

■proto native int BindingCount();■// return binding count
■proto native int Binding( int iIndex );■// return binding at index

■proto native void ClearBinding();■// remove all bindings

■proto native void BindCombo( string sButtonName );■■// bind combo to this input by name (single/ or ap
■proto native void BindComboByHash( int iHash );■■■// bind combo to this input by hash (single/ or appe

■proto native void AddAlternative();■■■■■■// add new alternative keybind
■proto native void ClearAlternative( int iIndex );■■// clear binding alternative by index
■proto native void SelectAlternative( int iIndex );■■// select binding alternative by index
■proto native int AlternativeCount();■■■■■// get currently assigned alternative count
■proto native int AlternativeIndex();■■■■■// get currently selected alternative index
■proto native void ClearDeviceBind( int iDeviceFlags );■// clear binding for specific device(s)

■proto native int BindKeyCount(); // binded key count (for selected alternative)
■proto native int GetBindKey( int iIndex ); // binded "control" at index (for selected alternative)
■proto native int GetBindDevice( int iIndex ); // binded "control" device type at index (for selected alternativ
■proto native bool CheckBindDevice( int iIndex, int iDeviceFlags ); // compare binded device "control" type

■proto native float LocalValue();

■proto native bool LocalPress();
■proto native bool LocalRelease();
■proto native bool LocalHold();
■proto native bool LocalHoldBegin();
■proto native bool LocalDoubleClick();
■proto native bool LocalClick();

■proto native bool IsCombo();■// return true if there is currently combo bind - use Binding() result !!!
```

```
// ---------------------------------------------
// ------ UiHintPanel.c - START --------------------
// ---------------------------------------------


/*
■Ui class for hints in in-game-menu
*/
class UiHintPanel extends ScriptedWidgetEventHandler
{
■// Const
■private const int ■■■■m_SlideShowDelay■= 25000;■■■■■■■■■■■■■// The speed of the slideshow
■private const string ■■■m_RootPath■■■= "Gui/layouts/new_ui/hints/in_game_hints.layout";■■// Layou
■private const string ■■■m_DataPath■■■= "Scripts/data/hints.json";■■■■■■■■// Json path
■// Widgets
■private Widget ■■■■■m_RootFrame;
■private Widget ■■■■■m_SpacerFrame;
■private ButtonWidget ■■■m_UiLeftButton;
■private ButtonWidget ■■■m_UiRightButton;
■private RichTextWidget■■■m_UiDescLabel;
■private TextWidget■■■■m_UiHeadlineLabel;
■private ImageWidget ■■■m_UiHintImage;
■private TextWidget ■■■■m_UiPageingLabel;
■// Data■■
■private ref array<ref HintPage>■m_ContentList;
■private int ■■■■■m_PageIndex;
■
■// ------------------------------------------------------
■
■// Constructor
■void UiHintPanel(Widget parent_widget)
■{■■
■■// Load Json File
■■LoadContentList();
■■// If load successful
■■if (m_ContentList)■
■■{
■■■// Build the layout
■■■BuildLayout(parent_widget);
■■■// Get random page index
■■■RandomizePageIndex();
■■■// Populate the layout with data
■■■PopulateLayout();
■■■// Start the slideshow
■■■StartSlideshow();■■■
■■}
■■else
■■{
■■■Print("ERROR: UiHintPanel - Could not create the hint panel. The data are missing!");
■■}
■}
■// Destructor
■void ~UiHintPanel()
■{
■■StopSlideShow();
■}
■
■// ---------------------------------------------------
■
■// Load content data from json file
■private void LoadContentList()
■{
■■JsonFileLoader<array<ref HintPage>>.JsonLoadFile( m_DataPath, m_ContentList );
```

```
// ---------------------------------------------
// ------ UIManager.c - START --------------------
// ---------------------------------------------


class UIManager
{
■//! Create & open menu with specific id (see \ref MenuID) and set its parent
■proto native UIScriptedMenu EnterScriptedMenu(int id, UIMenuPanel parent);
■proto native void EnterServerBrowser(UIMenuPanel parentMenu);

■proto native UIScriptedMenu ShowScriptedMenu(UIScriptedMenu menu, UIMenuPanel parent);
■proto native void HideScriptedMenu(UIScriptedMenu menu);

■proto native Widget GetWidgetUnderCursor();
■proto native bool IsDialogVisible();
■proto native bool IsModalVisible();
■proto native void CloseSpecificDialog(int id);
■proto native void CloseDialog();
■proto native void HideDialog();
■
■UIScriptedMenu CreateScriptedMenu(int id, UIMenuPanel parent)
■{
■■UIScriptedMenu menu = EnterScriptedMenu(id, parent);
■■HideScriptedMenu( menu );
■■return menu;
■}
■
■/**
  \brief Shows message dialog
■@param caption
■@param text
■@param id custom user id
■@param butts \ref DialogBoxType
■@param def \ref DialogBoxButton
■@param type \ref DialogMessageType
■@param handler
■\n usage :
■@code
■const int QUIT_DIALOG_ID = 76;
■GetGame().GetUIManager().ShowDialog("Quit", "Do You really want to quit?", QUIT_DIALOG_ID, DBT_
■...
■// after user pass dialog, callback on menu/event handler is called
■ScriptedWidgetEventHandler::OnModalResult( Widget  w,■int  x,■int  y,■int  code,■int  result■)
■{
■■if (code == QUIT_DIALOG_ID && result == DBB_YES) // yes this is callback for dialog we show earlier
■■{
■■■Quit();
■■}
■}
■@endcode
■*/
■proto native void ShowDialog(string caption, string text, int id, int butts /*DBT_*/, int def/*DBB_*/, int type
■proto native bool ShowCursor(bool visible);
■proto native bool IsCursorVisible();
■proto native bool IsDialogQueued();
■proto native bool ShowQueuedDialog();
■proto native int  GetLoginQueuePosition();
■proto native bool ScreenFadeVisible();
■proto native void ScreenFadeIn(float duration, string text, int backgroundColor, int textColor);
■proto native void ScreenFadeOut(float duration);
■proto native bool IsScaledMode();
■proto native void SetScaledMode(bool enabled);
```

```
// ---------------------------------------------
// ------ UIPopupScript.c - START --------------------
// ---------------------------------------------


class UIPopupScript
{
■//================================================
■// UIPopupScript
■//================================================
■void UIPopupScript(Widget wgt)
■{
■■m_WgtRoot = wgt;
■}

■void OnOpen(Param param)
■{
■}
■
■void OnClose()
■{
■}
■
■void Show(bool show)
■{
■■m_WgtRoot.Show(show);
■}
■
■bool OnClick(Widget w, int x, int y, int button)
■{
■■return false;
■}
■
■bool OnChange(Widget w, int x, int y, bool finished)
■{
■■return false;
■}
■
■//~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
■// PopupBack
■//~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
■protected UIPopupScript PopupBack()
■{
■■SceneEditorMenu menu = SceneEditorMenu.Cast( GetGame().GetUIManager().GetMenu() );
■■return menu.PopupBack();
■}

■//~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
■// PopupOpen
■//~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~■
■protected UIPopupScript PopupOpen(int popup_id, Param param)
■{
■■SceneEditorMenu menu = SceneEditorMenu.Cast( GetGame().GetUIManager().GetMenu() );
■■return menu.PopupOpen(popup_id, param);
■}

■//~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
■// GetSceneEditor
■//~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~■
■protected PluginSceneManager GetSceneEditor()
■{
■■PluginSceneManager m = PluginSceneManager.Cast( GetPlugin(PluginSceneManager) );
■■return m;
```

```
// ---------------------------------------------
// ------ UIPopupScriptConfigs.c - START --------------------
// ---------------------------------------------


class UIPopupScriptConfigs extends UIPopupScript
{
■private ButtonWidget m_BtnOk;
■private ButtonWidget m_BtnCancel;
■private ButtonWidget m_BtnCopyToClipboard;
■
■private TextListboxWidget m_ConfigHierarchyTextListbox;
■private TextListboxWidget m_ConfigVariablesTextListbox;
■
■private PluginConfigViewer m_ModuleConfigViewer;
■
■private EditBoxWidget m_ObjectConfigFilter;
■
■void UIPopupScriptConfigs( Widget wgt )
■{
■■m_BtnOk■■= ButtonWidget.Cast( wgt.FindAnyWidget("btn_ppp_pc_ok") );
■■m_ConfigHierarchyTextListbox = TextListboxWidget.Cast( wgt.FindAnyWidget("ConfigHierarchy") );
■■m_ConfigVariablesTextListbox = TextListboxWidget.Cast( wgt.FindAnyWidget("ConfigVariables") );
■■m_BtnCopyToClipboard = ButtonWidget.Cast( wgt.FindAnyWidget("btn_ppp_pc_copy") );
■■m_ObjectConfigFilter = EditBoxWidget.Cast( wgt.FindAnyWidget("ObjectConfigFilter") );
■}
■
■void ChangeConfigFilter()
■{
■■if ( m_ObjectConfigFilter.GetText() == "" )
■■{
■■■ClearHierarchy();
■■}
■■else
■■{
■■■FindInHierarchy( m_ObjectConfigFilter.GetText() );
■■}
■}
■
■override void OnOpen(Param param)
■{
■■m_ModuleConfigViewer■= PluginConfigViewer.Cast( GetPlugin( PluginConfigViewer ) );
■■ClearHierarchy();
■}
■
■void AddItemToClipboard( TextListboxWidget text_listbox_widget )
■{
■■int selected_row_index = text_listbox_widget.GetSelectedRow();
■■if ( selected_row_index != -1 )
■■{
■■■string item_name;
■■■text_listbox_widget.GetItemText( selected_row_index, 0, item_name );
■■■GetGame().CopyToClipboard( item_name );
■■}
■}
■
■void RenderVariables( int row )
■{
■■ConfigParams config_params;
■■if( row > -1 && row < m_ConfigHierarchyTextListbox.GetNumItems() )
■■{
■■■m_ConfigHierarchyTextListbox.GetItemData( row, 0, config_params );
■■■m_ConfigVariablesTextListbox.ClearItems();
```

```
// ---------------------------------------------
// ------ UIPopupScriptEditorSettings.c - START --------------------
// ---------------------------------------------


class UIPopupScriptEditorSettings extends UIPopupScript
{
■private CheckBoxWidget■■m_WgtTglSeleHighlight;
■private CheckBoxWidget■■m_WgtTglSavePlayerPos;
■private EditBoxWidget■■m_EdxRotationDelta;
■
■private ButtonWidget■■m_BtnCancel;
■
■private PluginSceneManager■m_ModuleSceneManager;
■
■//================================================
■// UIPopupScriptEditorSettings
■//================================================■
■void UIPopupScriptEditorSettings(Widget wgt)
■{
■■m_ModuleSceneManager = PluginSceneManager.Cast( GetPlugin(PluginSceneManager) );
■■
■■m_WgtTglSeleHighlight■= CheckBoxWidget.Cast( wgt.FindAnyWidget("cbx_ppp_est_flag_selection")
■■m_WgtTglSavePlayerPos■= CheckBoxWidget.Cast( wgt.FindAnyWidget("cbx_ppp_est_flag_load_play
■■m_EdxRotationDelta■■= EditBoxWidget.Cast( wgt.FindAnyWidget("ebx_ppp_est_rotation_delta_value
■■
■■m_BtnCancel■■■■= ButtonWidget.Cast( wgt.FindAnyWidget("btn_ppp_est_cancel") );
■}

■//================================================
■// OnOpen
■//================================================■
■override void OnOpen(Param param)
■{
■■m_WgtTglSeleHighlight.SetChecked( m_ModuleSceneManager.GetDrawSelection() );
■■m_WgtTglSavePlayerPos.SetChecked( m_ModuleSceneManager.GetLoadPlayerPos() );
■■m_EdxRotationDelta.SetText( m_ModuleSceneManager.GetRotationAngle().ToString() );
■}

■//================================================
■// OnClick
■//================================================■
■override bool OnClick(Widget w, int x, int y, int button)
■{
■■super.OnClick(w, x, y, button);
■■
■■if ( w == m_WgtTglSeleHighlight )
■■{
■■■m_ModuleSceneManager.SetDrawSelection(m_WgtTglSeleHighlight.IsChecked());
■■}
■■else if ( w == m_WgtTglSavePlayerPos )
■■{
■■■m_ModuleSceneManager.SetLoadPlayerPos(m_WgtTglSavePlayerPos.IsChecked());
■■}
■■else if ( w == m_BtnCancel )
■■{
■■■PopupBack();
■■■
■■■return true;
■■}
■■
■■return false;
■}
```

```
// --------------------------------------------
// ------ UIPopupScriptInitScript.c - START --------------------
// --------------------------------------------


class UIPopupScriptInitScript extends UIPopupScript
{
■private MultilineEditBoxWidget■m_MedxInitScript;
■
■private ButtonWidget■■■m_BtnRun;
■private ButtonWidget■■■m_BtnSave;
■private ButtonWidget■■■m_BtnCancel;
■
■private int■■■■■■m_SceneObjectIndex;
■private SceneObject■■■■m_SceneObject;
■
■private PluginSceneManager■■m_ModuleSceneManager;
■
■//================================================
■// UIPopupScriptInitScript
■//================================================■
■void UIPopupScriptInitScript(Widget wgt)
■{
■■m_ModuleSceneManager = PluginSceneManager.Cast( GetPlugin(PluginSceneManager) );
■■
■■m_MedxInitScript■= MultilineEditBoxWidget.Cast( wgt.FindAnyWidget("pnl_ppp_is_init_script_value")
■■
■■m_BtnRun■■■= ButtonWidget.Cast( wgt.FindAnyWidget("btn_ppp_is_run") );
■■m_BtnSave■■■= ButtonWidget.Cast( wgt.FindAnyWidget("btn_ppp_is_save") );
■■m_BtnCancel■■■= ButtonWidget.Cast( wgt.FindAnyWidget("btn_ppp_is_cancel") );
■}

■//================================================
■// OnOpen
■//================================================■
■override void OnOpen(Param param)
■{
■■m_MedxInitScript.SetText( "" );
■■
■■if ( param != NULL )
■■{
■■■Param2<int, SceneObject> param_scene_object = Param2<int, SceneObject>.Cast( param );
■■■int index = param_scene_object.param1;
■■■SceneObject scene_object = param_scene_object.param2;
■■■
■■■if ( index > -1 && scene_object != NULL )
■■■{
■■■■m_SceneObjectIndex = index;
■■■■m_SceneObject = scene_object;
■■■■m_MedxInitScript.SetText( m_SceneObject.GetInitScript() );
■■■}
■■}
■}

■//================================================
■// OnClick
■//================================================■
■override bool OnClick(Widget w, int x, int y, int button)
■{
■■super.OnClick(w, x, y, button);
■■
■■if ( w == m_BtnRun )
■■{
```

```
// ---------------------------------------------
// ------ UIPopupScriptNotify.c - START --------------------
// ---------------------------------------------


class UIPopupScriptNotify extends UIPopupScript
{
■private ButtonWidget■m_BtnOk;
■private TextWidget■■m_TxtLabel;
■
■//================================================
■// UIPopupScriptNotify
■//================================================■
■void UIPopupScriptNotify(Widget wgt)
■{
■■m_BtnOk■■= ButtonWidget.Cast( wgt.FindAnyWidget("btn_ppp_notify_ok") );
■■m_TxtLabel■= TextWidget.Cast( wgt.FindAnyWidget("txt_ppp_notify_label") );
■}

■//================================================
■// OnClick
■//================================================■
■override bool OnClick(Widget w, int x, int y, int button)
■{
■■super.OnClick(w, x, y, button);
■■
■■if ( w == m_BtnOk )
■■{
■■■PopupBack();
■■■
■■■return true;
■■}
■■
■■return false;
■}

■//================================================
■// OnClick
■//================================================■
■void SetLabelText(string text)
■{
■■m_TxtLabel.SetText(text);
■}
}



// ---------------------------------------------
// ------ UIPopupScriptNotify.c - END ----------------------
// ---------------------------------------------


// ---------------------------------------------
// ------ UIPopupScriptPositionManager.c - START --------------------
// ---------------------------------------------


class UIPopupScriptPositionManager extends UIPopupScript
{
■private TextListboxWidget■m_LstPositionList;
■private EditBoxWidget m_TxtSelectedX;
■private EditBoxWidget m_TxtSelectedY;
■private EditBoxWidget m_TxtCurrentX;
```

```
// ---------------------------------------------
// ------ UIPopupScriptPresetNew.c - START --------------------
// ---------------------------------------------


class UIPopupScriptPresetNew extends UIPopupScript
{
■private ButtonWidget m_BtnOk;
■private ButtonWidget m_BtnCancel;
■
■//==================================================
■// UIPopupScriptSceneNew
■//==================================================■
■void UIPopupScriptPresetNew( Widget wgt )
■{
■■m_BtnOk■■= ButtonWidget.Cast( wgt.FindAnyWidget("btn_ppp_pn_ok") );
■■m_BtnCancel■= ButtonWidget.Cast( wgt.FindAnyWidget("btn_ppp_pn_cancel") );
■}

■//==================================================
■// OnClick
■//==================================================■
■override bool OnClick( Widget w, int x, int y, int button )
■{
■■super.OnClick( w, x, y, button );
■■
■■if ( w == m_BtnOk )
■■{
■■■EditBoxWidget wgt_text = EditBoxWidget.Cast( m_WgtRoot.FindAnyWidget("fld_ppp_pn_new_name
■■■PluginConfigDebugProfile m_ConfigDebugProfile = PluginConfigDebugProfile.Cast( GetPlugin(Plugin
■■■m_ConfigDebugProfile.PresetAdd( wgt_text.GetText() );
■■■
■■■SceneEditorMenu menu = SceneEditorMenu.Cast( GetGame().GetUIManager().GetMenu() );
■■■menu.RefreshLists();
■■■PopupBack();
■■■
■■■return true;
■■}
■■else if ( w == m_BtnCancel )
■■{
■■■PopupBack();
■■■
■■■return true;
■■}
■■
■■return false;
■}
}



// ---------------------------------------------
// ------ UIPopupScriptPresetNew.c - END ----------------------
// ---------------------------------------------


// ---------------------------------------------
// ------ UIPopupScriptPresetRename.c - START --------------------
// ---------------------------------------------


class UIPopupScriptPresetRename extends UIPopupScript
{
```

```
// ---------------------------------------------
// ------ UIPopupScriptSceneDelete.c - START --------------------
// ---------------------------------------------


class UIPopupScriptSceneDelete extends UIPopupScript
{
■private ButtonWidget ■m_BtnYes;
■private ButtonWidget ■m_BtnNo;
■private string■■■m_DeleteScene;
■
■//================================================
■// UIPopupScriptSceneDelete
■//================================================■
■void UIPopupScriptSceneDelete(Widget wgt)
■{
■■m_BtnYes■= ButtonWidget.Cast( wgt.FindAnyWidget("btn_ppp_sn_dlt_yes") );
■■m_BtnNo■■= ButtonWidget.Cast( wgt.FindAnyWidget("btn_ppp_sn_dlt_no") );
■}

■//================================================
■// OnClick
■//================================================■
■override bool OnClick(Widget w, int x, int y, int button)
■{
■■super.OnClick(w, x, y, button);
■■
■■if ( w == m_BtnYes )
■■{
■■■PluginSceneManager editor = PluginSceneManager.Cast( GetPlugin(PluginSceneManager) );
■■■
■■■editor.SceneDelete(m_DeleteScene);
■■■
■■■PopupBack();
■■■
■■■return true;
■■}
■■else if ( w == m_BtnNo )
■■{
■■■m_DeleteScene = STRING_EMPTY;
■■■PopupBack();
■■■return true;
■■}
■■
■■return false;
■}

■void SetDeleteName(string scene_name)
■{
■■m_DeleteScene = scene_name;
■}
}



// ---------------------------------------------
// ------ UIPopupScriptSceneDelete.c - END ----------------------
// ---------------------------------------------
// ---------------------------------------------
// ------ UIPopupScriptSceneManager.c - START --------------------
// ---------------------------------------------


class UIPopupScriptSceneManager extends UIPopupScript
{
■private ButtonWidget ■■m_BtnCancel;
■private ButtonWidget ■■m_BtnSceneNew;
■private ButtonWidget ■■m_BtnSceneLoad;
■private ButtonWidget ■■m_BtnSceneRename;
■private ButtonWidget ■■m_BtnSceneDuplicate;
■private ButtonWidget ■■m_BtnSceneDelete;
■private TextListboxWidget■m_LstListScenes;
■
■private static const int m_DaysInMonth[12] = { 31, 28, 31, 30, 31, 30, 31, 31, 30, 31, 30, 31 };
■
■private ButtonWidget■m_BtnSave;
■
■private TextWidget■■m_TxtWeatherTime;
■
■private SliderWidget ■m_SldStartTime;
■private TextWidget■■m_TxtStartTimeValue;
■private SliderWidget ■m_SldStartDay;
■private TextWidget■■m_TxtStartDayValue;
■private SliderWidget■m_SldOvercast;
■private TextWidget■■m_TxtOvercastValue;
■private SliderWidget■m_SldRain;
■private TextWidget■■m_TxtRainValue;
■private SliderWidget■m_SldFog;
■private TextWidget■■m_TxtFogValue;
■private SliderWidget■m_SldWindForce;
■private TextWidget■■m_TxtWindForceValue;
■
■private int■■■■m_OrigYear;
■private int■■■■m_OrigMonth;
■private int■■■■m_OrigDay;
■private int■■■■m_OrigHour;
■private int■■■■m_OrigMinute;
■private float■■■m_OrigOvercast;
■private float■■■m_OrigRain;
■private float■■■m_OrigFog;
■private float■■■m_OrigWindForce;
■
■private int■■■■m_CurrYear;
■private int■■■■m_CurrMonth;
■private int■■■■m_CurrDay;
■private int■■■■m_CurrHour;
■private int■■■■m_CurrMinute;
■private float■■■m_CurrOvercast;
■private float■■■m_CurrRain;
■private float■■■m_CurrFog;
■private float■■■m_CurrWindForce;
■
■//================================================
■// UIPopupScriptSceneManager
■//================================================■
■void UIPopupScriptSceneManager(Widget wgt)
■{
■■m_BtnCancel■■■= ButtonWidget.Cast( wgt.FindAnyWidget("btn_ppp_sm_cancel") );
■■m_BtnSceneNew■■= ButtonWidget.Cast( wgt.FindAnyWidget("btn_ppp_sm_mission_new") );
■■m_BtnSceneLoad■■= ButtonWidget.Cast( wgt.FindAnyWidget("btn_ppp_sm_mission_load") );
■■m_BtnSceneRename■= ButtonWidget.Cast( wgt.FindAnyWidget("btn_ppp_sm_mission_rename") );
```

```
// ---------------------------------------------
// ------ UIPopupScriptSceneNew.c - START --------------------
// ---------------------------------------------


class UIPopupScriptSceneNew extends UIPopupScript
{
■private ButtonWidget m_BtnOk;
■private ButtonWidget m_BtnCancel;
■
■//================================================
■// UIPopupScriptSceneNew
■//================================================■
■void UIPopupScriptSceneNew(Widget wgt)
■{
■■m_BtnOk■■= ButtonWidget.Cast( wgt.FindAnyWidget("btn_ppp_sn_ok") );
■■m_BtnCancel■= ButtonWidget.Cast( wgt.FindAnyWidget("btn_ppp_sn_cancel") );
■}

■//================================================
■// OnClick
■//================================================■
■override bool OnClick(Widget w, int x, int y, int button)
■{
■■super.OnClick(w, x, y, button);
■■
■■if ( w == m_BtnOk )
■■{
■■■EditBoxWidget wgt_text = EditBoxWidget.Cast( m_WgtRoot.FindAnyWidget("fld_ppp_sn_new_name
■■■
■■■PluginSceneManager editor = PluginSceneManager.Cast( GetPlugin(PluginSceneManager) );

■■■Log("OnClick -> SceneLoad");
■■■
■■■editor.SceneLoad(wgt_text.GetText());
■■■editor.SceneSave();
■■■
■■■PopupBack();
■■■
■■■return true;
■■}
■■else if ( w == m_BtnCancel )
■■{
■■■PopupBack();
■■■
■■■return true;
■■}
■■
■■return false;
■}
}



// ---------------------------------------------
// ------ UIPopupScriptSceneNew.c - END ---------------------
// ---------------------------------------------
// ---------------------------------------------
// ------ UIPopupScriptSceneRename.c - START --------------------
// ---------------------------------------------


class UIPopupScriptSceneRename extends UIPopupScript
{
■private ButtonWidget■m_BtnOk;
■private ButtonWidget■m_BtnCancel;
■private string■■■m_RenameName;
■
■//===============================================
■// UIPopupScriptSceneRename
■//===============================================■
■void UIPopupScriptSceneRename(Widget wgt)
■{
■■m_BtnOk■■■= ButtonWidget.Cast( wgt.FindAnyWidget("btn_ppp_sn_rnm_ok") );
■■m_BtnCancel■■= ButtonWidget.Cast( wgt.FindAnyWidget("btn_ppp_sn_rnm_cancel") );
■}

■//===============================================
■// OnClick
■//===============================================■
■override bool OnClick(Widget w, int x, int y, int button)
■{
■■super.OnClick(w, x, y, button);
■■
■■if ( w == m_BtnOk )
■■{■■
■■■EditBoxWidget wgt_text = EditBoxWidget.Cast( m_WgtRoot.FindAnyWidget("fld_ppp_sn_rnm_new_r
■■■
■■■PluginSceneManager editor = PluginSceneManager.Cast( GetPlugin(PluginSceneManager) );
■■■
■■■editor.SceneRename(m_RenameName, wgt_text.GetText());
■■■m_RenameName = STRING_EMPTY;
■■■
■■■PopupBack();
■■■
■■■return true;
■■}
■■else if ( w == m_BtnCancel )
■■{
■■■PopupBack();
■■■
■■■return true;
■■}
■■
■■return false;
■}

■//===============================================
■// SetRenameName
■//===============================================■
■void SetRenameName(string rename_name)
■{
■■m_RenameName = rename_name;
■}
}



// ---------------------------------------------
// ------ UIPopupScriptSceneRename.c - END ----------------------
// ---------------------------------------------
// ------ UIPopupScriptSceneSettings.c - START --------------------
// ---------------------------------------------


class UIPopupScriptSceneSettings extends UIPopupScript
{
■private static const int m_DaysInMonth[12] = { 31, 28, 31, 30, 31, 30, 31, 31, 30, 31, 30, 31 };
■
■private ButtonWidget■m_BtnSave;
■private ButtonWidget■m_BtnCancel;
■
■private TextWidget■■m_TxtWeatherTime;
■
■private SliderWidget ■m_SldStartTime;
■private TextWidget■■m_TxtStartTimeValue;
■private SliderWidget ■m_SldStartDay;
■private TextWidget■■m_TxtStartDayValue;
■private SliderWidget■m_SldOvercast;
■private TextWidget■■m_TxtOvercastValue;
■private SliderWidget■m_SldRain;
■private TextWidget■■m_TxtRainValue;
■private SliderWidget■m_SldFog;
■private TextWidget■■m_TxtFogValue;
■private SliderWidget■m_SldWindForce;
■private TextWidget■■m_TxtWindForceValue;
■
■private int■■■■m_OrigYear;
■private int■■■■m_OrigMonth;
■private int■■■■m_OrigDay;
■private int■■■■m_OrigHour;
■private int■■■■m_OrigMinute;
■private float■■■m_OrigOvercast;
■private float■■■m_OrigRain;
■private float■■■m_OrigFog;
■private float■■■m_OrigWindForce;
■
■private int■■■■m_CurrYear;
■private int■■■■m_CurrMonth;
■private int■■■■m_CurrDay;
■private int■■■■m_CurrHour;
■private int■■■■m_CurrMinute;
■private float■■■m_CurrOvercast;
■private float■■■m_CurrRain;
■private float■■■m_CurrFog;
■private float■■■m_CurrWindForce;
■
■//================================================
■// UIPopupScriptSceneSettings
■//================================================■
■void UIPopupScriptSceneSettings(Widget wgt)
■{
■■m_BtnSave■■■= ButtonWidget.Cast( wgt.FindAnyWidget("btn_ppp_st_save") );
■■m_BtnCancel■■■= ButtonWidget.Cast( wgt.FindAnyWidget("btn_ppp_st_cancel") );
■■
■■m_TxtWeatherTime■= TextWidget.Cast( wgt.FindAnyWidget("txt_ppp_st_w_time_value") );
■■
■■m_SldStartTime■■= SliderWidget.Cast( wgt.FindAnyWidget("sld_ppp_st_start_time") );
■■m_TxtStartTimeValue■= TextWidget.Cast( wgt.FindAnyWidget("txt_ppp_st_start_time_value") );
■■
■■m_SldStartDay■■= SliderWidget.Cast( wgt.FindAnyWidget("sld_ppp_st_start_day") );
■■m_TxtStartDayValue■= TextWidget.Cast( wgt.FindAnyWidget("txt_ppp_st_start_day_value") );
■■
```

```
// ---------------------------------------------
// ------ UIPropertyAttachment.c - START --------------------
// ---------------------------------------------


class UIPropertyAttachment
{
■private Widget■■■m_WgtRoot;
■private Widget■■■m_WgtThis;
■private XComboBoxWidget■m_WgtComboBox;
■private TextWidget■■m_WgtSlotName;
■private ref TStringArray m_ComboItems;
■private int■■■■m_PrevIndex;
■private EntityAI■■m_Obj;
■private int■■■■m_SlotID;
■
■
■void UIPropertyAttachment(Widget root)
■{
■■m_WgtRoot■■= root;
■■
■■m_ComboItems = new TStringArray;
■■
■■m_WgtThis = GetGame().GetWorkspace().CreateWidgets("gui/layouts/scene_editor/day_z_scene_edit
■■m_WgtComboBox■= XComboBoxWidget.Cast( m_WgtThis.FindAnyWidget("combo_box") );
■■m_WgtSlotName■= TextWidget.Cast( m_WgtThis.FindAnyWidget("txt_slot_name") );
■}
■
■void ~UIPropertyAttachment()
■{
■■m_WgtRoot■■= NULL;
■■m_WgtComboBox■= NULL;
■■
■■delete m_WgtThis;
■}
■
■bool OnClick(Widget w, int x, int y, int button)
■{
■■if ( w == m_WgtComboBox )
■■{
■■■if ( m_PrevIndex != 0 )
■■■{
■■■■EntityAI attachment = m_Obj.GetInventory().FindAttachment(m_SlotID);
■■■■GetGame().ObjectDelete(attachment);
■■■}
■■■
■■■int curr_index = m_WgtComboBox.GetCurrentItem();
■■■
■■■if ( curr_index != 0 )
■■■{■■■■
■■■■PluginDeveloper ■module_dev = PluginDeveloper.Cast( GetPlugin(PluginDeveloper) );
■■■■■
■■■■EntityAI e = module_dev.SpawnEntityAsAttachment(PluginSceneManager.PLAYER, m_Obj, m_Co
■■■}
■■■■
■■■m_PrevIndex = curr_index;
■■■
■■■return true;
■■}
■■
■■return false;
■}
■
```

```
// ---------------------------------------------
// ------ UIScriptedMenu.c - START --------------------
// ---------------------------------------------


//-----------------------------------------------------------------------------
class UIMenuPanel: Managed
{
■proto native UIMenuPanel GetSubMenu();
■proto native UIMenuPanel GetParentMenu();
■proto native UIMenuPanel GetVisibleMenu();
■proto native void SetSubMenu(UIMenuPanel submenu);
■proto native void SetParentMenu(UIMenuPanel parent);
■proto native bool CanClose();
■proto native bool CanCloseOnEscape();
■//! Create & open menu with specific id (see \ref MenuID) and set this menu as its parent
■proto native UIScriptedMenu EnterScriptedMenu(int id);
■
■proto native void DestroySubmenu();
■proto native bool IsAnyMenuVisible();
■proto native bool IsVisible();
■
■//! If visibility of application is changed. On console it is called when application is suspended or constrain
■//! @param isVisible indicate if application is visible in foreground
■void OnVisibilityChanged(bool isVisible)
■{
■}
■
■//! Safe way to close window, using this function can even window safely close itself
■proto native void Close();

■bool UseMouse()■{
■■#ifdef PLATFORM_CONSOLE
■■return GetGame().GetInput().IsEnabledMouseAndKeyboardEvenOnServer();
■■#else
■■return true;
■■#endif
■}

■bool UseKeyboard()■{
■■#ifdef PLATFORM_CONSOLE
■■return GetGame().GetInput().IsEnabledMouseAndKeyboardEvenOnServer();
■■#else
■■return true;
■■#endif
■}
■
■bool UseGamepad()■{
■■return true;
■}

■//! Returns \ref MenuID
■int GetID()■{
■■return MENU_UNKNOWN;
■}
■
■//! Refresh request, called from anywhere
■void Refresh()
■{
■}
};

//-----------------------------------------------------------------------------
```

```
// ----------------------------------------------
// ------ UIScriptedWindow.c - START --------------------
// ----------------------------------------------


class UIScriptedWindow
{
■Widget■m_WgtRoot;
■int m_Id;

■//---MOVE TO UIMANAGER WHEN FIXED
■static ref map<int, UIScriptedWindow> m_ActiveWindows;
■
■static void AddToActiveWindows( int id, UIScriptedWindow window )
■{
■■if ( m_ActiveWindows == NULL )
■■{
■■■m_ActiveWindows = new map<int, UIScriptedWindow>;
■■}
■■■
■■m_ActiveWindows.Insert( id, window );
■}
■
■static void RemoveFromActiveWindows( int id )
■{
■■if ( m_ActiveWindows )
■■{
■■■m_ActiveWindows.Remove( id );■
■■}
■}
■
■static UIScriptedWindow GetWindow( int id )
■{
■■if ( m_ActiveWindows )
■■{
■■■return m_ActiveWindows.Get( id );
■■}
■■
■■return NULL;
■}
■
■static map<int, UIScriptedWindow> GetActiveWindows()
■{
■■return m_ActiveWindows;
■}
■//---

■
■void UIScriptedWindow(int id)
■{
■■m_Id = id;
■}
■■
■void ~UIScriptedWindow()
■{
■■GetWidgetRoot().Show(false);
■■delete GetWidgetRoot();
■}
■
■Widget GetWidgetRoot()
■{
■■return m_WgtRoot;
■}
```

```
// ---------------------------------------------
// ------ UKAssVest_ColorBase.c - START --------------------
// ---------------------------------------------


class UKAssVest_ColorBase extends Clothing {};
class UKAssVest_Black extends UKAssVest_ColorBase {};
class UKAssVest_Camo extends UKAssVest_ColorBase {};
class UKAssVest_Khaki extends UKAssVest_ColorBase {};
class UKAssVest_Olive extends UKAssVest_ColorBase {};


// ---------------------------------------------
// ------ UKAssVest_ColorBase.c - END ----------------------
// ---------------------------------------------


// ---------------------------------------------
// ------ UMP45.c - START --------------------
// ---------------------------------------------


class UMP45_Base : RifleBoltLock_Base
{
■void UMP_Base()
■{■
■}
■
■override RecoilBase SpawnRecoilObject()
■{
■■return new Ump45Recoil(this);
■}
■
■//Debug menu Spawn Ground Special
■override void OnDebugSpawn()
■{
■■GameInventory inventory = GetInventory();
■■inventory.CreateInInventory( "PistolSuppressor" );
■■inventory.CreateInInventory( "ReflexOptic" );
■■inventory.CreateInInventory( "UniversalLight" );
■■inventory.CreateInInventory( "Battery9V" );
■■inventory.CreateInInventory( "Battery9V" );
■■
■■SpawnAttachedMagazine("Mag_UMP_25Rnd");
■}
};


// ---------------------------------------------
// ------ UMP45.c - END ----------------------
// ---------------------------------------------


// ---------------------------------------------
// ------ Ump45Recoil.c - START --------------------
// ---------------------------------------------


class Ump45Recoil: RecoilBase
{
■override void Init()
■{
■■vector point_1;
```

```
// ---------------------------------------------
// ------ Unconsciousness.c - START --------------------
// ---------------------------------------------


class UnconsciousnessMdfr: ModifierBase
{
■const int UNCONSIOUSS_COOLDOWN_TIME = 60;//in s
■
■override void Init()
■{
■■m_TrackActivatedTime = false;
■■m_ID ■■■■■= eModifiers.MDF_UNCONSCIOUSNESS;
■■m_TickIntervalInactive ■= 0.5;
■■m_TickIntervalActive ■= 0.5;
■}■■

■override bool ActivateCondition(PlayerBase player)
■{
■■if( player.GetHealth("","Shock") <=  PlayerConstants.UNCONSCIOUS_THRESHOLD )
■■{
■■■return true;
■■}
■■return false;
■}
■
■override void OnActivate(PlayerBase player)
■{
■■DayZPlayerSyncJunctures.SendPlayerUnconsciousness(player, true);
■}
■
■override void OnDeactivate(PlayerBase player)
■{
■■player.m_UnconsciousEndTime = GetGame().GetTime();
■■DayZPlayerSyncJunctures.SendPlayerUnconsciousness(player, false);
■}
■
■override string GetDebugText()
■{
■■string text_pulse = "pulse type:"+m_Player.GetPulseType()+"|";
■■string text_shock = (PlayerConstants.CONSCIOUS_THRESHOLD - m_Player.GetHealth("","Shock")).T
■■string debug_text = text_pulse + text_shock;
■■return debug_text;
■}
■
■override void OnReconnect(PlayerBase player)
■{
■}

■override bool DeactivateCondition(PlayerBase player)
■{
■■if( player.GetHealth("","Shock") >= PlayerConstants.CONSCIOUS_THRESHOLD && player.GetPulseT
■■{
■■■return true;
■■}
■■return false;
■}

■override void OnTick(PlayerBase player, float deltaT)
■{
■■
■}
};
```

```
// ----------------------------------------------
// ------ UndergroundAreaLoader.c - START --------------------
// ----------------------------------------------


class JsonUndergroundTriggers
{
■ref array<ref JsonUndergroundAreaTriggerData> Triggers;
}


class JsonUndergroundAreaBreadcrumb
{
■vector GetPosition()
■{
■■return Vector(Position[0],Position[1],Position[2]);
■}
■
■ref array<float> ■Position;
■float ■■■■EyeAccommodation;
■bool ■■■■UseRaycast;
■float ■■■■Radius;
}

class JsonUndergroundAreaTriggerData
{
■vector GetPosition()
■{
■■return Vector(Position[0],Position[1],Position[2]);
■}
■
■vector GetOrientation()
■{
■■return Vector(Orientation[0],Orientation[1],Orientation[2]);
■}
■vector GetSize()
■{
■■return Vector(Size[0],Size[1],Size[2]);
■}
■
■ref array<float> Position;
■ref array<float> Orientation;
■ref array<float> Size;
■float  EyeAccommodation;
■float  InterpolationSpeed;
■
■ref array<ref JsonUndergroundAreaBreadcrumb> Breadcrumbs;
■
};


class UndergroundAreaLoader
{
■private static string m_Path = "$mission:cfgundergroundtriggers.json";
■
■static ref JsonUndergroundTriggers m_JsonData;
■
■
■static JsonUndergroundTriggers GetData()
■{
■■if ( !FileExist( m_Path ) )
■■{
■■■// We fallback to check in data and notify user file was not found in mission
```

```
// ---------------------------------------------
// ------ UndergroundHandlerClient.c - START --------------------
// ---------------------------------------------


enum EUndergroundPresence
{
■NONE,//player is not interacting with underground at any level
■OUTER,//player is on the outskirts of the underdound, some effects are already in effect, while others mi
■TRANSITIONING,//player has entered underground and is in the process of screen darkening transition
■FULL//the player is now fully entered underground
}

class UndergroundHandlerClient
{
■const float LIGHT_BLEND_SPEED_IN = 5;
■const float LIGHT_BLEND_SPEED_OUT = 1.75;
■const float MAX_RATIO = 0.9;//how much max ratio between 0..1 can a single breadcrumb occupy
■const float RATIO_CUTOFF = 0;//what's the minimum ratio a breadcrumb needs to have to be considere
■const float DISTANCE_CUTOFF = 5;//we ignore breadcrumbs further than this distance
■const float ACCO_MODIFIER = 1;//when we calculate eye accommodation between 0..1 based on the b
■const float DEFAULT_INTERPOLATION_SPEED = 7;
■const string UNDERGROUND_LIGHTING = "dz\\data\\lighting\\lighting_underground.txt";
■private ref AnimationTimer ■m_AnimTimerLightBlend;
■
■private PlayerBase ■■■■m_Player;
■private PPERUndergroundAcco ■m_Requester;
■private PPERequester_CameraNV ■m_NVRequester;
■private ref set<UndergroundTrigger> m_InsideTriggers = new set<UndergroundTrigger>();
■
■private float ■■m_EyeAccoTarget = 1;
■private float ■■m_AccoInterpolationSpeed;
■private float ■■m_EyeAcco = 1;
■private float ■■m_LightingLerpTarget;
■private float ■■m_LightingLerp;
■private EffectSound m_AmbientSound;
■
■private UndergroundTrigger m_TransitionalTrigger;
■
■void UndergroundHandlerClient(PlayerBase player)
■{
■■GetGame().GetWorld().LoadUserLightingCfg(UNDERGROUND_LIGHTING, "Underground");
■■m_Player = player;
■■m_NVRequester = PPERequester_CameraNV.Cast(PPERequesterBank.GetRequester( PPERequeste
■}
■
■void ~UndergroundHandlerClient()
■{
■■if (GetGame())
■■{
■■■GetGame().GetWorld().SetExplicitVolumeFactor_EnvSounds2D(1, 0.5);
■■■GetGame().GetWeather().SuppressLightningSimulation(false);
■■■GetGame().GetWorld().SetUserLightingLerp(0);
■■■if (m_AmbientSound)
■■■■m_AmbientSound.Stop();
■■}
■}
■
■private PPERUndergroundAcco GetRequester()
■{
■■if (!m_Requester)
■■{
■■■m_Requester = PPERUndergroundAcco.Cast(PPERequesterBank.GetRequester( PPERequesterBa
// ----------------------------------------------
// ------ UndergroundStash.c - START --------------------
// ----------------------------------------------


class UndergroundStash extends ItemBase
{
■void UndergroundStash() {}
■
■void PlaceOnGround()
■{
■■vector pos = GetPosition();
■■pos[1] = GetGame().SurfaceRoadY(pos[0], pos[2]);
■■pos[1] = pos[1] + 0.22;
■■SetPosition(pos);
■}
■
■ItemBase GetStashedItem()
■{
■■ItemBase item;■■
■■if (GetInventory().GetCargo().GetItemCount() > 0)
■■{
■■■item = ItemBase.Cast(GetInventory().GetCargo().GetItem(0));
■■}

■■return item;
■}
■
■override bool CanDisplayCargo()
■{
■■return false;
■}
■
■override bool CanPutInCargo(EntityAI parent)
■{
■■return false;
■}
■
■override bool CanReleaseCargo(EntityAI cargo)
■{
■■return false;
■}
■
■override bool CanReceiveItemIntoHands(EntityAI item_to_hands)
■{
■■return false;
■}
■
■override bool CanSaveItemInHands(EntityAI item_in_hands)
■{
■■return false;
■}
■
■override bool CanPutIntoHands(EntityAI parent)
■{
■■return false;
■}
■
■override bool IsInventoryVisible()
■{
■■return false;
■}
}
```

```
// ---------------------------------------------
// ------ UndergroundTrigger.c - START --------------------
// ---------------------------------------------


//! This entity exists both client andserver side
//! when it enters into player's bubble and gets instantiated client-side, it will locally spawn UndergroundTrig
class UndergroundTriggerCarrier extends UndergroundTriggerCarrierBase
{
■ref JsonUndergroundAreaTriggerData m_Data;
■UndergroundTrigger m_Trigger;
■
■void UndergroundTriggerCarrier()
■{
■■RegisterNetSyncVariableInt("m_TriggerIndex", -1, 255);
■}
■
■void ~UndergroundTriggerCarrier()
■{
■■if (m_Trigger && !m_Trigger.IsSetForDeletion() && GetGame())
■■{
■■■//RemoveChild(m_Trigger);
■■■m_Trigger.Delete();
■■}
■}
■
■override void OnVariablesSynchronized()
■{
■■super.OnVariablesSynchronized();
■■if (!m_Trigger)
■■{
■■■SpawnTrigger();
■■}
■}
■■
■bool CanSpawnTrigger()
■{
■■return UndergroundAreaLoader.m_JsonData && m_TriggerIndex != -1;
■}
■
■void RequestDelayedTriggerSpawn()
■{
■■//Print("RequestDelayedTriggerSpawn() " + this);
■■GetGame().GetCallQueue(CALL_CATEGORY_SYSTEM).CallLater(SpawnTrigger, 100);
■}
■
■void SpawnTrigger()
■{
■■
■■if (!CanSpawnTrigger())
■■{
■■■RequestDelayedTriggerSpawn();
■■■return;
■■}
■■
■■if (UndergroundAreaLoader.m_JsonData.Triggers && UndergroundAreaLoader.m_JsonData.Triggers.I
■■{
■■■JsonUndergroundAreaTriggerData data = UndergroundAreaLoader.m_JsonData.Triggers[m_Trigger
■■■UndergroundTrigger trigger = UndergroundTrigger.Cast(GetGame().CreateObjectEx( "UndergroundT
■■■if (trigger)
■■■{
■■■■#ifdef DEVELOPER
■■■■trigger.m_Local = true;
```

```
// ---------------------------------------------
// ------ UniversalLight.c - START --------------------
// ---------------------------------------------


class UniversalLight extends Switchable_Base
{
■UniversallightLight ■m_Light;
■
■static int■■■REFLECTOR_ID = 1;
■static int■■■GLASS_ID = 2;
■
■static string ■■LIGHT_OFF_GLASS ■= "dz\\gear\\tools\\data\\flashlight_glass.rvmat";
■static string ■■LIGHT_OFF_REFLECTOR = "dz\\weapons\\attachments\\data\\m4_flashlight.rvmat";
■static string ■■LIGHT_ON_GLASS ■■= "dz\\gear\\tools\\data\\flashlight_glass_on.rvmat";
■static string ■■LIGHT_ON_REFLECTOR ■= "dz\\weapons\\attachments\\data\\m4_flashlight_on.rvmat"
■
■ref array<int> ■■m_AttachmentSlotsCheck;
■
■void UniversalLight()
■{
■■InitAttachmentsSlotsToCheck(m_AttachmentSlotsCheck);
■}
■
■override ScriptedLightBase GetLight()
■{
■■return m_Light;
■}
■
■override bool CanPutAsAttachment( EntityAI parent )
■{
■■if ( !super.CanPutAsAttachment(parent) ) {return false;}
■■
■■bool req_attachment ■■= false;
■■bool rail_attachment_found   = false;
■■int slot_id;
■■ItemBase attachment;
■■for ( int i = 0; i < parent.GetInventory().GetAttachmentSlotsCount(); i++ )
■■{
■■■slot_id = parent.GetInventory().GetAttachmentSlotId(i);
■■■if ( m_AttachmentSlotsCheck.Find(slot_id) != -1 )
■■■{
■■■■req_attachment = true;
■■■■attachment = ItemBase.Cast(parent.GetInventory().FindAttachment(slot_id));
■■■■if ( attachment && attachment.ConfigIsExisting("hasRailFunctionality") && attachment.ConfigGetBo
■■■■■rail_attachment_found = true;
■■■}
■■}
■■return !req_attachment || (req_attachment && rail_attachment_found);
■}
■
■override void OnWorkStart()
■{
■■if ( !GetGame().IsServer()  ||  !GetGame().IsMultiplayer() ) // Client side
■■{
■■■m_Light = UniversallightLight.Cast(  ScriptedLightBase.CreateLight(UniversallightLight, "0 0 0", 0.08)
■■■m_Light.AttachOnMemoryPoint(this, "beamStart", "beamEnd");
■■■SetObjectMaterial(GLASS_ID, LIGHT_ON_GLASS);
■■■SetObjectMaterial(REFLECTOR_ID, LIGHT_ON_REFLECTOR);
■■}
■}

■override void OnWork( float consumed_energy )
```

```
// ----------------------------------------------
// ------ UniversallightLight.c - START --------------------
// ----------------------------------------------


class UniversallightLight extends SpotLightBase
{
■private static float m_DefaultBrightness = 6.05;
■private static float m_DefaultRadius = 25;
■private static float m_DefaultAngle = 100;
■
■void UniversallightLight()
■{
■■SetVisibleDuringDaylight( true );
■■SetRadiusTo( m_DefaultRadius );
■■SetSpotLightAngle( m_DefaultAngle );
■■SetCastShadow( true );
■■SetBrightnessTo( m_DefaultBrightness );
■■SetFadeOutTime( 0.15 );
■■SetAmbientColor( 0.95, 0.88, 0.8 );
■■SetDiffuseColor( 0.95, 0.88, 0.8 );
■■SetDisableShadowsWithinRadius( 1.2 );
■}
■
■void SetIntensity( float coef, float time )
■{
■■FadeBrightnessTo( m_DefaultBrightness * coef, time );
■■FadeRadiusTo( m_DefaultRadius * coef, time );
■}
■
■override void UpdateLightMode( string slotName )
■{
■■switch (slotName)
■■{
■■■case "weaponFlashlight":
■■■■SetSpotLightAngle( m_DefaultAngle/2 );
■■■■SetRadiusTo( m_DefaultRadius * 1.8 );
■■■break;
■■■
■■■default:
■■■■SetSpotLightAngle( m_DefaultAngle );
■■■■SetRadiusTo( m_DefaultRadius );
■■■break;
■■}
■}
}


// ----------------------------------------------
// ------ UniversallightLight.c - END ----------------------
// ----------------------------------------------


// ----------------------------------------------
// ------ UniversalTemperatureSource.c - START --------------------
// ----------------------------------------------


class UniversalTemperatureSourceSettings
{
■float m_UpdateInterval■■= 1.0;■■■//! how often the Update is ticking
■float m_TemperatureMin■■= 0;■■■//! min temperature you can get from the TemperatureSource
■float m_TemperatureMax■■= 100;■■■//! max temperature you can get from the TemperatureSource
```

```
// ---------------------------------------------
// ------ UniversalTemperatureSourceLambdaBase.c - START --------------------
// ---------------------------------------------


class UniversalTemperatureSourceLambdaBase
{
    const float HEAT_THROUGH_AIR_COEF = 0.003;

    void UniversalTemperatureSourceLambdaBase() {};
    void ~UniversalTemperatureSourceLambdaBase() {};

    void Execute(UniversalTemperatureSourceSettings pSettings, UniversalTemperatureSourceResult resul
    
    void DryItemsInVicinity(UniversalTemperatureSourceSettings pSettings)
    {
        float distance;
        array<Object> nearestObjects = new array<Object>;

        vector pos = pSettings.m_Position;
        if (pSettings.m_Parent != null)
        {
            pos = pSettings.m_Parent.GetPosition();
        }

        GetGame().GetObjectsAtPosition(pos, pSettings.m_RangeMax, nearestObjects, null);
        for (int i = 0; i < nearestObjects.Count(); i++)
        {
            EntityAI nearestEnt = EntityAI.Cast(nearestObjects.Get(i));
            //! heat transfer to items (no in player possession)
            if (nearestEnt && nearestEnt.HasWetness() && nearestEnt != pSettings.m_Parent && nearestEnt.Ge
            {
                float wetness = nearestEnt.GetWet();
    
                //! drying of items around the fireplace (based on distance)
                if (wetness > 0)
                {
                    distance = vector.Distance(nearestEnt.GetPosition(), pSettings.m_Position);
                    distance = Math.Max(distance, 0.1);    //min distance cannot be 0 (division by zero)
                    
                    wetness = wetness * (HEAT_THROUGH_AIR_COEF / distance);
                    wetness = Math.Clamp(wetness, nearestEnt.GetWetMin(), nearestEnt.GetWetMax());
                    nearestEnt.AddWet(-wetness);
                }
            }
        }
    }
}

class UniversalTemperatureSourceLambdaConstant : UniversalTemperatureSourceLambdaBase
{
    override void Execute(UniversalTemperatureSourceSettings pSettings, UniversalTemperatureSourceRes
    {
        resultValues.m_Temperature = pSettings.m_TemperatureMax;
        
        DryItemsInVicinity(pSettings);
    }
}

class UniversalTemperatureSourceLambdaEngine : UniversalTemperatureSourceLambdaBase
{
    override void Execute(UniversalTemperatureSourceSettings pSettings, UniversalTemperatureSourceRes
    {
```

```
// ---------------------------------------------
// ------ UniversalTemperatureSourceLambdaFireplace.c - START --------------------
// ---------------------------------------------


class UniversalTemperatureSourceLambdaFireplace : UniversalTemperatureSourceLambdaBase
{
■int m_FuelCount;
■int m_SmallFireplaceTemperatureMax;
■int m_NormalFireplaceTemperatureMax;
■float m_Temperature;

■void UniversalTemperatureSourceLambdaFireplace()
■{
■■m_FuelCount■■■■■■■= 0;
■■m_SmallFireplaceTemperatureMax■■= 0;
■■m_NormalFireplaceTemperatureMax■■= 0;
■■m_Temperature■■■■■■= 0;
■}
■
■void SetSmallFireplaceTemperatureMax(int value)
■{
■■m_SmallFireplaceTemperatureMax = value;
■}
■
■void SetNormalFireplaceTemperatureMax(int value)
■{
■■m_NormalFireplaceTemperatureMax = value;
■}
■
■void SetFuelCount(int value)
■{
■■m_FuelCount = value;
■}
■
■void SetCurrentTemperature(float temperature)
■{
■■//no fuel present, temperature should be low but there can be high temperature from previous fuel burn
■■if (m_FuelCount == 0 || temperature <= m_SmallFireplaceTemperatureMax)
■■{
■■■temperature = Math.Clamp(temperature, 0, m_SmallFireplaceTemperatureMax); //small fire
■■}
■■else
■■{
■■■temperature = Math.Clamp(temperature, 0, m_NormalFireplaceTemperatureMax); //normal fire
■■}
■■
■■m_Temperature = temperature;
■}

■override void Execute(UniversalTemperatureSourceSettings pSettings, UniversalTemperatureSourceRes
■{
■■resultValues.m_Temperature = m_Temperature;
■■//Debug.Log(string.Format("Execute: temperature: %1", resultValues.m_Temperature), "UTS Fireplace
■■
■■if (pSettings.m_AffectStat)
■■{
■■■//! set temperature to the item stat
■■■pSettings.m_Parent.SetTemperature(m_Temperature);
■■}
■■
■■DryItemsInVicinity(pSettings);
■}
```

```
// ---------------------------------------------
// ------ UpgradeTorchWithLard.c - START --------------------
// ---------------------------------------------


class UpgradeTorchWithLard extends RecipeBase
{■
■override void Init()
■{
■■m_Name = "#STR_UpgradeTorchWithLard0";
■■m_IsInstaRecipe = false;//should this recipe be performed instantly without animation
■■m_AnimationLength = 1;//animation length in relative time units
■■m_Specialty = 0.02;// value > 0 for roughness, value < 0 for precision
■■
■■//conditions
■■m_MinDamageIngredient[0] = -1;//-1 = disable check
■■m_MaxDamageIngredient[0] = -1;//-1 = disable check
■■
■■m_MinQuantityIngredient[0] = -1;//-1 = disable check
■■m_MaxQuantityIngredient[0] = -1;//-1 = disable check
■■
■■m_MinDamageIngredient[1] = -1;//-1 = disable check
■■m_MaxDamageIngredient[1] = -1;//-1 = disable check
■■
■■m_MinQuantityIngredient[1] = -1;//-1 = disable check
■■m_MaxQuantityIngredient[1] = -1;//-1 = disable check
■■//----------------------------------------------------------------------------------------------------------------------
■■
■■//INGREDIENTS
■■//ingredient 1
■■InsertIngredient(0,"Lard");//you can insert multiple ingredients this way
■■
■■m_IngredientAddHealth[0] = 0;// 0 = do nothing
■■m_IngredientSetHealth[0] = -1; // -1 = do nothing
■■m_IngredientAddQuantity[0] = 0;// 0 = do nothing
■■m_IngredientDestroy[0] = false;//true = destroy, false = do nothing
■■m_IngredientUseSoftSkills[0] = false;// set 'true' to allow modification of the values by softskills on this i
■■
■■//ingredient 2
■■InsertIngredient(1,"Torch");//you can insert multiple ingredients this way
■■InsertIngredient(1,"LongTorch");//you can insert multiple ingredients this way
■■
■■m_IngredientAddHealth[1] = 0;// 0 = do nothing
■■m_IngredientSetHealth[1] = -1; // -1 = do nothing
■■m_IngredientAddQuantity[1] = 0;// 0 = do nothing
■■m_IngredientDestroy[1] = false;// false = do nothing
■■m_IngredientUseSoftSkills[1] = false;// set 'true' to allow modification of the values by softskills on this i
■■//----------------------------------------------------------------------------------------------------------------------
■■
■■//result1
■■//AddResult("Torch");//add results here

■■m_ResultSetFullQuantity[0] = false;//true = set full quantity, false = do nothing
■■m_ResultSetQuantity[0] = -1;//-1 = do nothing
■■m_ResultSetHealth[0] = -1;//-1 = do nothing
■■m_ResultInheritsHealth[0] = -2;// (value) == -1 means do nothing; a (value) >= 0 means this result will i
■■m_ResultInheritsColor[0] = -1;// (value) == -1 means do nothing; a (value) >= 0 means this result class
■■m_ResultToInventory[0] = -2;//(value) == -2 spawn result on the ground;(value) == -1 place anywhere i
■■m_ResultUseSoftSkills[0] = false;// set 'true' to allow modification of the values by softskills on this resu
■■m_ResultReplacesIngredient[0] = -1;// value == -1 means do nothing; a value >= 0 means this result wi
■}

■override bool CanDo(ItemBase ingredients[], PlayerBase player)//final check for recipe's validity
```

```
// ---------------------------------------------
// ------ UpgradeTorchWithLiquidFuel.c - START --------------------
// ---------------------------------------------


class UpgradeTorchWithLiquidFuel extends RecipeBase
{■
■override void Init()
■{
■■m_Name = "#str_upgrade_torch_fuel";
■■m_IsInstaRecipe = false;//should this recipe be performed instantly without animation
■■m_AnimationLength = 1;//animation length in relative time units
■■m_Specialty = 0.02;// value > 0 for roughness, value < 0 for precision
■■
■■//conditions
■■m_MinDamageIngredient[0] = -1;//-1 = disable check
■■m_MaxDamageIngredient[0] = -1;//-1 = disable check
■■
■■m_MinQuantityIngredient[0] = -1;//-1 = disable check
■■m_MaxQuantityIngredient[0] = -1;//-1 = disable check
■■
■■m_MinDamageIngredient[1] = -1;//-1 = disable check
■■m_MaxDamageIngredient[1] = -1;//-1 = disable check
■■
■■m_MinQuantityIngredient[1] = -1;//-1 = disable check
■■m_MaxQuantityIngredient[1] = -1;//-1 = disable check
■■//----------------------------------------------------------------------------------------------------------------------
■■
■■//INGREDIENTS
■■//ingredient 1
■■InsertIngredient(0,"Bottle_Base");
■■InsertIngredient(0,"Barrel_ColorBase");
■■
■■m_IngredientAddHealth[0] = 0;// 0 = do nothing
■■m_IngredientSetHealth[0] = -1; // -1 = do nothing
■■m_IngredientAddQuantity[0] = 0;// 0 = do nothing
■■m_IngredientDestroy[0] = false;//true = destroy, false = do nothing
■■m_IngredientUseSoftSkills[0] = false;// set 'true' to allow modification of the values by softskills on this i
■■
■■//ingredient 2
■■InsertIngredient(1,"Torch");//you can insert multiple ingredients this way
■■InsertIngredient(1,"LongTorch");//you can insert multiple ingredients this way
■
■■m_IngredientAddHealth[1] = 0;// 0 = do nothing
■■m_IngredientSetHealth[1] = -1; // -1 = do nothing
■■m_IngredientAddQuantity[1] = -1;// 0 = do nothing
■■m_IngredientDestroy[1] = false;// false = do nothing
■■m_IngredientUseSoftSkills[1] = false;// set 'true' to allow modification of the values by softskills on this i
■■//----------------------------------------------------------------------------------------------------------------------
■■
■■//result1
■■//AddResult("Torch");//add results here

■■m_ResultSetFullQuantity[0] = false;//true = set full quantity, false = do nothing
■■m_ResultSetQuantity[0] = -1;//-1 = do nothing
■■m_ResultSetHealth[0] = -1;//-1 = do nothing
■■m_ResultInheritsHealth[0] = -2;// (value) == -1 means do nothing; a (value) >= 0 means this result will i
■■m_ResultInheritsColor[0] = -1;// (value) == -1 means do nothing; a (value) >= 0 means this result class
■■m_ResultToInventory[0] = -2;//(value) == -2 spawn result on the ground;(value) == -1 place anywhere i
■■m_ResultUseSoftSkills[0] = false;// set 'true' to allow modification of the values by softskills on this resu
■■m_ResultReplacesIngredient[0] = -1;// value == -1 means do nothing; a value >= 0 means this result wi
■}
```

```
// ---------------------------------------------
// ------ Ushanka_ColorBase.c - START --------------------
// ---------------------------------------------


class Ushanka_ColorBase extends Clothing
{
■override void SetActions()
■{
■■super.SetActions();
■■AddAction(ActionWringClothes);
■}
};
class Ushanka_Black extends Ushanka_ColorBase {};
class Ushanka_Blue extends Ushanka_ColorBase {};
class Ushanka_Green extends Ushanka_ColorBase {};



// ---------------------------------------------
// ------ Ushanka_ColorBase.c - END ----------------------
// ---------------------------------------------


// ---------------------------------------------
// ------ USMCJacket_ColorBase.c - START --------------------
// ---------------------------------------------


class USMCJacket_ColorBase extends Clothing
{
■override void SetActions()
■{
■■super.SetActions();
■■AddAction(ActionWringClothes);
■}
};
class USMCJacket_Desert extends USMCJacket_ColorBase {};
class USMCJacket_Woodland extends USMCJacket_ColorBase {};


// ---------------------------------------------
// ------ USMCJacket_ColorBase.c - END ----------------------
// ---------------------------------------------


// ---------------------------------------------
// ------ USMCPants_ColorBase.c - START --------------------
// ---------------------------------------------


class USMCPants_ColorBase extends Clothing
{
■override void SetActions()
■{
■■super.SetActions();
■■AddAction(ActionWringClothes);
■}
};
class USMCPants_Desert extends USMCPants_ColorBase {};
class USMCPants_Woodland extends USMCPants_ColorBase {};
```

```
// ---------------------------------------------
// ------ UtilityClasses.c - START --------------------
// ---------------------------------------------


class ItemVariableFlags
{
■static const int NONE ■■■■■= 0;
■static const int FLOAT■■■ ■■= 0x0001;
■static const int STRING■■ ■■■= 0x0002;
■static const int BOOL ■■■■■= 0x0004;
};

class CachedObjectsParams
{■
■static ref Param1<int> ■■PARAM1_INT;//CachedObjectsParams.PARAM1_INT
■static ref Param1<bool> ■PARAM1_BOOL;//CachedObjectsParams.PARAM1_BOOL
■static ref Param1<float> ■PARAM1_FLOAT;//CachedObjectsParams.PARAM1_FLOAT
■static ref Param1<string> ■PARAM1_STRING;//CachedObjectsParams.PARAM1_STRING
■
■static ref Param2<int,int> ■■PARAM2_INT_INT;//CachedObjectsParams.PARAM2_INT_INT
■static ref Param2<int,float> ■PARAM2_INT_FLOAT;//CachedObjectsParams.PARAM2_INT_FLOAT
■static ref Param2<int,string> ■PARAM2_INT_STRING;//CachedObjectsParams.PARAM2_INT_STRING
■static ref Param2<string,float> PARAM2_STRING_FLOAT;//CachedObjectsParams.PARAM2_STRING_
■static ref Param2<string,string> PARAM2_STRING_STRING;//CachedObjectsParams.PARAM2_STRIN
■static ref Param2<float,float> ■PARAM2_FLOAT_FLOAT;//CachedObjectsParams.PARAM2_STRING_
■
■static void Init()
■{
■■PARAM1_INT = new Param1<int>(0);
■■PARAM1_BOOL = new Param1<bool>(false);
■■PARAM1_FLOAT = new Param1<float>(0);
■■PARAM1_STRING = new Param1<string>("");
■■
■■PARAM2_INT_INT = new Param2<int,int>(0,0);
■■PARAM2_INT_FLOAT = new Param2<int,float>(0,0);
■■PARAM2_INT_STRING = new Param2<int,string>(0,"");
■■PARAM2_STRING_FLOAT = new Param2<string,float>("",0);
■■PARAM2_STRING_STRING = new Param2<string,string>("","");
■■PARAM2_FLOAT_FLOAT = new Param2<float,float>(0,0);
■}
};


class CachedObjectsArrays//don't forget to .Clear() your cache object before using it
{■
■static void Init()
■{
■■ARRAY_STRING ■= new TStringArray;
■■ARRAY_FLOAT ■= new TFloatArray;
■■ARRAY_INT ■■= new TIntArray;
■}
■
■static ref TStringArray ARRAY_STRING;//CachedObjectsArrays.ARRAY_STRING
■static ref TFloatArray ARRAY_FLOAT;//CachedObjectsArrays.ARRAY_FLOAT
■static ref TIntArray ARRAY_INT;//CachedObjectsArrays.ARRAY_INT
};


// ---------------------------------------------
// ------ UtilityClasses.c - END ----------------------
// ---------------------------------------------
// ---------------------------------------------
// ------ Van_01.c - START --------------------
// ---------------------------------------------


class Van_01 extends CarScript
{
■protected ref UniversalTemperatureSource m_UTSource;
■protected ref UniversalTemperatureSourceSettings m_UTSSettings;
■protected ref UniversalTemperatureSourceLambdaEngine m_UTSLEngine;

■void Van_01()
■{
■■//m_dmgContactCoef = 0.070; //TODO::Set proper value
■}
■
■override void EEInit()
■{■■
■■super.EEInit();
■■
■■if (GetGame().IsServer() || !GetGame().IsMultiplayer())
■■{
 ■■■m_UTSSettings ■■■■■= new UniversalTemperatureSourceSettings();
■■■m_UTSSettings.m_ManualUpdate ■= true;
■■■m_UTSSettings.m_TemperatureMin■= 0;
■■■m_UTSSettings.m_TemperatureMax■= 30;
■■■m_UTSSettings.m_RangeFull■■= 0.5;
■■■m_UTSSettings.m_RangeMax■■= 2;
■■■m_UTSSettings.m_TemperatureCap■= 25;
■■■
■■■m_UTSLEngine■■■■■= new UniversalTemperatureSourceLambdaEngine();
■■■m_UTSource■■■■■■= new UniversalTemperatureSource(this, m_UTSSettings, m_UTSLEngine);
■■}■■
■}
■
■override void OnEngineStart()
■{
■■super.OnEngineStart();

■■if (GetGame().IsServer() || !GetGame().IsMultiplayer())
■■{
■■■m_UTSource.SetDefferedActive(true, 20.0);
■■}
■}
■
■override void OnEngineStop()
■{
■■super.OnEngineStop();

■■if (GetGame().IsServer() || !GetGame().IsMultiplayer())
■■{
■■■m_UTSource.SetDefferedActive(false, 10.0);
■■}
■}
■
■override void EOnPostSimulate(IEntity other, float timeSlice)
■{
■■if (GetGame().IsServer() || !GetGame().IsMultiplayer())
■■{
■■■if (m_UTSource.IsActive())
■■■{
■■■■m_UTSource.Update(m_UTSSettings, m_UTSLEngine);
■■■}
```

```
// ---------------------------------------------
// ------ Vector2.c - START --------------------
// ---------------------------------------------


class Vector2
{■
■void Vector2(float value_x, float value_y)
■{
■■x = value_x;
■■y = value_y;
■}
■
■float x;
■float y;
};


// ---------------------------------------------
// ------ Vector2.c - END ----------------------
// ---------------------------------------------


// ---------------------------------------------
// ------ VehicleAnimInstances.c - START --------------------
// ---------------------------------------------


enum VehicleAnimInstances
{
■CIVVAN = 0,
■V3S = 1,
■SEDAN = 2,
■HATCHBACK = 3,
■BUS = 4,
■S120 = 5,
■MULTICAR = 6,
■GOLF = 7,
■HMMWV = 8
};


// ---------------------------------------------
// ------ VehicleAnimInstances.c - END ----------------------
// ---------------------------------------------


// ---------------------------------------------
// ------ VehicleBattery.c - START --------------------
// ---------------------------------------------


class VehicleBattery : ItemBase
{
■override bool CanPutAsAttachment(EntityAI parent)
■{
■■if (!super.CanPutAsAttachment(parent))
■■■return false;
■■
■■if (parent.IsInherited(BatteryCharger))
■■{
■■■BatteryCharger charger = BatteryCharger.Cast(parent);
■■■return charger.CanReceiveAttachment(this, InventorySlots.INVALID);
```

```
// ----------------------------------------------
// ------ VehicleManager.c - START --------------------
// ----------------------------------------------


/*class VehicleManager
{■■
■PlayerBase ■■■■■■m_Player;
■HumanInputController ■■■m_HIC;
■protected ActionManagerBase■ ■m_AM;
■■
■void VehicleManager( PlayerBase player, ActionManagerBase mngr)
■{
■■m_Player = player;
■■m_HIC = m_Player.GetInputController();
■■m_AM = mngr;
■}
■
■//Called from players commandhandler each frame, checks input
■void Update( float deltaT )
■{
■■PickAction();
■}
■
■protected void PerformAction( int actionID )
■{
■■if ( !GetGame().IsDedicatedServer() )
■■{
■■■if ( !m_Player.GetCommandModifier_Action() && !m_Player.GetCommand_Action() )
■■■{
■■■■ActionBase action = m_AM.GetAction(actionID);
■■■■if ( action )
■■■■{■■■
■■■■■//ActionTarget target = new ActionTarget( m_AM.FindActionTarget().GetObject(), -1, Vector(0,0,0)
■■■■■ActionTarget target = new ActionTarget( m_Player.GetHierarchyParent(), null, -1, Vector(0,0,0), -
■■■■■if ( target && action.Can(m_Player, target, null ) )
■■■■■{
■■■■■■action.Start(m_Player, target, null );
■■■■■■if ( GetGame().IsClient() )
■■■■■■{
■■■■■■■ScriptInputUserData ctx = new ScriptInputUserData;
■■■■■■■ctx.Write(INPUT_UDT_STANDARD_ACTION);
■■■■■■■ctx.Write(action.GetType());
■■■■■■■action.WriteToContext(ctx, target);
■■■■■■■ctx.Send();
■■■■■■}
■■■■■}
■■■■}
■■■}
■■}
■■else
■■{
■■■m_AM.StartDeliveredAction();
■■}
■}
■
■protected void PickAction()
■{
■■Car car;
■■if ( !Class.CastTo(car,  m_Player.GetHierarchyParent()) ) return;
■■
■■if ( m_Player )
■■{
```

```
// ---------------------------------------------
// ------ VehicleSmoke.c - START --------------------
// ---------------------------------------------


class EffVehicleSmoke : EffectParticle
{
■void EffVehicleSmoke()
■{
■■SetParticleStateLight();
■}
■
■
■void SetParticleStateLight()
■{
■■SetParticleState( ParticleList.HATCHBACK_COOLANT_OVERHEATING );
■}
■
■
■void SetParticleStateHeavy()
■{
■■SetParticleState( ParticleList.HATCHBACK_COOLANT_OVERHEATED );
■}
■
■void SetParticleState( int state )
■{
■■bool was_playing = IsPlaying();
■
■■Stop();
■■
■■SetParticleID(state);
■■
■■if (was_playing)
■■{
■■■Start(); // resume effect
■■}
■}
}


// ---------------------------------------------
// ------ VehicleSmoke.c - END ----------------------
// ---------------------------------------------


// ---------------------------------------------
// ------ VerticalSpacer.c - START --------------------
// ---------------------------------------------


// ---------------------------------------------------------
class VerticalSpacer : SpacerBase
{
■reference int border;
■reference int gap;

■override protected void UpdateChild(Widget child, float w, float h, int index)
■{
■■float itemWidth = w - (2 * border);
■■float itemHeight = (h - (border * 2) - ((m_count - 1) * gap)) / m_count;
■
■■child.SetPos(border, border + ((itemHeight + gap) * index));
■■child.SetSize(itemWidth, itemHeight);
```

```
// ---------------------------------------------
// ------ VicinityContainer.c - START --------------------
// ---------------------------------------------


class VicinityContainer: CollapsibleContainer
{
■ref VicinitySlotsContainer■■■m_VicinityIconsContainer;
■ref map<EntityAI, ref Container>■m_ShowedItems■■■■= new ref map<EntityAI, ref Container>;
■ref map<int, ref Container>■■■m_ShowedItemsIDs■■■= new ref map<int, ref Container>;
■ref array<EntityAI>■■■■■m_ShowedItemIcons■■■= new array<EntityAI>;
■ref map<CargoBase, ref Container>■m_ShowedCargos■■■■= new ref map<CargoBase, ref Container
■protected bool■■■■■■m_IsProcessing = false; // Prevents refreshing every time a child is added while
■
■const float DISTANCE_TO_ENTITIES ■= 1.0;
■const float DISTANCE_TO_THE_REST ■= 0.5;
■
■void VicinityContainer( LayoutHolder parent, int sort = -1 )
■{
■■m_VicinityIconsContainer = new VicinitySlotsContainer( this );
■■m_Body.Insert( m_VicinityIconsContainer );
■■m_VicinityIconsContainer.GetRootWidget().SetColor(166 << 24 | 120 << 16 | 120 << 8 | 120);
■■m_MainWidget = m_RootWidget.FindAnyWidget( "body" );
■■WidgetEventHandler.GetInstance().RegisterOnChildAdd( m_MainWidget, this, "OnChildAdd" );
■■WidgetEventHandler.GetInstance().RegisterOnChildRemove( m_MainWidget, this, "OnChildRemove" )
■■
■■RecomputeOpenedContainers();
■■m_CollapsibleHeader.SetName("#container_vicinity");
■■LoadDefaultState();
■}
■
■bool IsVicinityContainerIconsActive()
■{
■■return m_VicinityIconsContainer.IsActive();
■}
■
■VicinitySlotsContainer GetVicinityIconsContainer()
■{
■■return m_VicinityIconsContainer;
■}
■
■bool IsContainerWithCargoActive()
■{
■■return ( ContainerWithCargo.Cast( GetFocusedContainer() ) != null );
■}
■
■bool IsItemWithAttachmentsActive()
■{
■■return ( ContainerWithCargoAndAttachments.Cast( GetFocusedContainer() ) != null );
■}
■
■bool IsItemWithCategoriesActive()
■{
■■return ( AttachmentCategoriesContainer.Cast( GetFocusedContainer() ) != null );
■}

■void TraverseShowedItems()
■{
■■map<string, bool> serialized_types = new map<string, bool>();
■■for ( int i = 0; i < m_ShowedItems.Count(); i++ )
■■{
■■■EntityAI entity = m_ShowedItems.GetKey( i );
■■■Container container = m_ShowedItems.GetElement( i );
```

```
// ---------------------------------------------
// ------ VicinityItemManager.c - START --------------------
// ---------------------------------------------


class VicinityItemManager
{
■private const float UPDATE_FREQUENCY ■■■■= 0.25;
■private const float VICINITY_DISTANCE■■■■= 0.5;
■private const float VICINITY_ACTOR_DISTANCE■■■= 2.0;
■private const float VICINITY_LARGE_ACTOR_DISTANCE ■= 3.0;
■private const float VICINITY_CONE_DISTANCE■■■= 2.0;
■private const float VICINITY_CONE_REACH_DISTANCE■= 2.0;
■private const float VICINITY_CONE_ANGLE ■■■= 30;
■private const float VICINITY_CONE_RADIANS ■■■= 0.5;
■private const string CE_CENTER ■■■■■■= "ce_center";
■private const float HEIGHT_OFFSET ■■■■■= 0.2;
■private const int OBJECT_OBSTRUCTION_WEIGHT■■■= 10000; //in grams
■private const float CONE_HEIGHT_MIN ■■■■= -0.5;
■private const float CONE_HEIGHT_MAX ■■■■= 3.0;

■private ref array<EntityAI> m_VicinityItems ■■= new array<EntityAI>();
■private ref array<CargoBase> m_VicinityCargos ■■= new array<CargoBase>();
■private float m_RefreshCounter;
■private static ref VicinityItemManager s_Instance;
■
■static VicinityItemManager GetInstance ()
■{
■■if (!s_Instance)
■■■s_Instance = new VicinityItemManager();

■■return s_Instance;
■}
■
■void Init()
■{
■}
■
■array<EntityAI> GetVicinityItems()
■{
■■return m_VicinityItems;
■}
■
■void AddVicinityItems(Object object)
■{
■■EntityAI entity = EntityAI.Cast(object);
■■if (entity)
■■{
■■■if (m_VicinityItems.Find(entity) == INDEX_NOT_FOUND && GameInventory.CheckManipulatedObjec
■■■{
■■■■m_VicinityItems.Insert(entity);
■■■}
■■}
■}

■array<CargoBase> GetVicinityCargos()
■{
■■return m_VicinityCargos;
■}
■
■void AddVicinityCargos(CargoBase object)
■{
■■if (m_VicinityCargos.Find(object) == INDEX_NOT_FOUND)
```

```
// ---------------------------------------------
// ------ VicinitySlotsContainer.c - START --------------------
// ---------------------------------------------


class VicinitySlotsContainer: Container
{
■protected ref AttachmentsGroupContainer■■m_Container;
■protected int■■■■■■■■m_ItemsCount;
■protected int■■■■■■■■m_SlotsCount;
■protected ref array<EntityAI>■■■■m_ShowedItems;
■
■void VicinitySlotsContainer( LayoutHolder parent )
■{
■■m_Container = new AttachmentsGroupContainer(this);
■■ref SlotsContainer con = new SlotsContainer( m_Container, null );
■■m_Container.Insert( con );
■■m_Body.Insert( m_Container );
■■for( int j = 0; j < ITEMS_IN_ROW; j++ )
■■{
■■■SlotsIcon icon = con.GetSlotIcon( j );
■■■WidgetEventHandler.GetInstance().RegisterOnDropReceived( icon.GetPanelWidget(),  m_Parent, "C
■■■WidgetEventHandler.GetInstance().RegisterOnDropReceived( icon.GetGhostSlot(),  m_Parent, "OnD
■■■WidgetEventHandler.GetInstance().RegisterOnDropReceived( icon.GetMainWidget(),  m_Parent, "Or
■■■
■■■WidgetEventHandler.GetInstance().RegisterOnDraggingOver( icon.GetPanelWidget(),  m_Parent, "D
■■■WidgetEventHandler.GetInstance().RegisterOnDraggingOver( icon.GetGhostSlot(),  m_Parent, "Drag
■■■WidgetEventHandler.GetInstance().RegisterOnDraggingOver( icon.GetMainWidget(),  m_Parent, "Dra
■■■
■■■WidgetEventHandler.GetInstance().RegisterOnDoubleClick( icon.GetPanelWidget(),  this, "DoubleCli
■■■WidgetEventHandler.GetInstance().RegisterOnMouseButtonUp( icon.GetPanelWidget(),  this, "Mouse
■■■WidgetEventHandler.GetInstance().RegisterOnMouseButtonDown( icon.GetPanelWidget(),  this, "Mo
■■}
■■
■■con.SetColumnCount(0);
■■con.SetForceShow(true);
■■
■■WidgetEventHandler.GetInstance().RegisterOnDropReceived( m_Container.GetMainWidget(),  m_Pare
■■WidgetEventHandler.GetInstance().RegisterOnDraggingOver( m_Container.GetMainWidget(),  m_Pare
■■
■■m_ShowedItems = new array<EntityAI>;
■}
■
■bool IsItemWithContainerActive()
■{
■■EntityAI ent = GetFocusedItem();
■■return ent && ( ent.GetInventory().GetCargo() || (ent.GetSlotsCountCorrect() > 0 && ent.CanDisplayAny
■■//TODO: also check for cargo visibility maybe?
■}
■
■override bool IsItemWithQuantityActive()
■{
■■EntityAI ent = GetFocusedItem();
■■return ent && QuantityConversions.HasItemQuantity( ent ) && ent.CanBeSplit();
■}
■
■override bool IsItemActive()
■{
■■EntityAI ent = GetFocusedItem();
■■return ent && !IsItemWithQuantityActive() && !IsItemWithContainerActive();
■}
■
■bool IsEmptyItemActive()
```

```
// ---------------------------------------------
// ------ VitaminBottle.c - START --------------------
// ---------------------------------------------


class VitaminBottle : Edible_Base
{
■//Specify this item can only be combined but not split
■override void InitItemVariables()
■{
■■super.InitItemVariables();
■■can_this_be_combined = true;
■}
■
■override void OnConsume(float amount, PlayerBase consumer)
■{
■■if (consumer.GetModifiersManager().IsModifierActive(eModifiers.MDF_IMMUNITYBOOST)) //effectivel
■■{
■■■consumer.GetModifiersManager().DeactivateModifier(eModifiers.MDF_IMMUNITYBOOST);
■■}

■■consumer.GetModifiersManager().ActivateModifier(eModifiers.MDF_IMMUNITYBOOST);
■}
■
■override void SetActions()
■{
■■super.SetActions();
■■
■■AddAction(ActionForceConsumeSingle);
■■AddAction(ActionConsumeSingle);
■}
}



// ---------------------------------------------
// ------ VitaminBottle.c - END ----------------------
// ---------------------------------------------


// ---------------------------------------------
// ------ Vodka.c - START --------------------
// ---------------------------------------------


class Vodka: Bottle_Base
{
■override void SetActions()
■{
■■super.SetActions();
■■
■■RemoveAction(ActionWashHandsItem);
■}
};



// ---------------------------------------------
// ------ Vodka.c - END ----------------------
// ---------------------------------------------
// ---------------------------------------------
// ------ Vomit.c - START --------------------
// ---------------------------------------------


class EffVomit : EffectParticle
{
■void EffVomit()
■{
■■SetParticleID(ParticleList.VOMIT);
■}
}


// ---------------------------------------------
// ------ Vomit.c - END ----------------------
// ---------------------------------------------


// ---------------------------------------------
// ------ VomitBlood.c - START --------------------
// ---------------------------------------------


class EffVomitBlood : EffectParticle
{
■void EffVomitBlood()
■{
■■SetParticleID(ParticleList.VOMIT_BLOOD);
■}
}


// ---------------------------------------------
// ------ VomitBlood.c - END ----------------------
// ---------------------------------------------


// ---------------------------------------------
// ------ VomitState.c - START --------------------
// ---------------------------------------------


class VomitSymptom extends SymptomBase
{
■//just for the Symptom parameters set-up and gets called even if the Symptom doesn't execute, don't put
■const int BLOOD_LOSS = 250;
■override void OnInit()
■{
■■m_SymptomType = SymptomTypes.PRIMARY;
■■m_Priority = 100;
■■m_ID = SymptomIDs.SYMPTOM_VOMIT;
■■m_DestroyOnAnimFinish = true;
■■m_SyncToClient = false;
■■m_Duration = 5;
■■m_MaxCount = 1;
■}
■
■//!gets called every frame
■override void OnUpdateServer(PlayerBase player, float deltatime)
■{

■}
```

```
// ---------------------------------------------
// ------ VomitStuffed.c - START --------------------
// ---------------------------------------------


class VomitStuffedMdfr: ModifierBase
{
■override void Init()
■{
■■m_TrackActivatedTime = false;
■■m_ID ■■■■■= eModifiers.MDF_VOMITSTUFFED;
■■m_TickIntervalInactive ■= DEFAULT_TICK_TIME_INACTIVE;
■■m_TickIntervalActive ■= DEFAULT_TICK_TIME_ACTIVE;
■}
■
■override void OnTick(PlayerBase player, float deltaT)
■{

■}
■
■override bool ActivateCondition(PlayerBase player)
■{
■■float stomach = m_Player.m_PlayerStomach.GetStomachVolume();
■■if ( stomach >= PlayerConstants.VOMIT_THRESHOLD )
■■{
■■■return true;
■■}
■■else
■■{
■■■return false;
■■}
■}

■override void OnActivate(PlayerBase player)
■{
■■player.GetSymptomManager().QueueUpPrimarySymptom( SymptomIDs.SYMPTOM_VOMIT );
■}


■override bool DeactivateCondition(PlayerBase player)
■{
■■return !ActivateCondition(player);
■}
};


// ---------------------------------------------
// ------ VomitStuffed.c - END ----------------------
// ---------------------------------------------


// ---------------------------------------------
// ------ VONManager.c - START --------------------
// ---------------------------------------------


class VONManagerBase : Managed
{
■protected bool■■■■■m_VoNToggled;
■ref ■  ScriptInvoker■■■m_OnVonStateEvent;
■ref ■  ScriptInvoker■■■m_OnPartyChatChangedEvent;
■
■void VONManagerBase()
```

```
// ---------------------------------------------
// ------ VSS.c - START --------------------
// ---------------------------------------------


class VSS_Base : RifleBoltFree_Base
{
■override RecoilBase SpawnRecoilObject()
■{
■■return new VSSRecoil(this);
■}
■
■/*override int GetWeaponSpecificCommand(int weaponAction ,int subCommand)
■{
■■if ( weaponAction == WeaponActions.RELOAD)
■■{
■■■switch (subCommand)
■■■{
■■■■case WeaponActionReloadTypes.RELOADSRIFLE_MAGAZINE_BULLET:
■■■■■return WeaponActionReloadTypes.RELOADRIFLE_MAGAZINE_BULLET;
■■■■
■■■■case WeaponActionReloadTypes.RELOADSRIFLE_NOMAGAZINE_BULLET:
■■■■■return WeaponActionReloadTypes.RELOADRIFLE_NOMAGAZINE_BULLET;
■■■■
■■■■case WeaponActionReloadTypes.RELOADSRIFLE_MAGAZINE_NOBULLET:
■■■■■return WeaponActionReloadTypes.RELOADRIFLE_MAGAZINE_NOBULLET;
■■■■
■■■■case WeaponActionReloadTypes.RELOADSRIFLE_NOMAGAZINE_NOBULLET:
■■■■■return WeaponActionReloadTypes.RELOADRIFLE_NOMAGAZINE_NOBULLET;
■■■■
■■■■default:
■■■■■return subCommand;
■■■}
■■
■■}
■■return subCommand;
■}*/
■
■//Debug menu Spawn Ground Special
■override void OnDebugSpawn()
■{
■■EntityAI entity;
■■if ( Class.CastTo(entity, this) )
■■{
■■■entity.GetInventory().CreateInInventory( "PSO1Optic" );■
■■■entity.SpawnEntityOnGroundPos("Mag_VSS_10Rnd", entity.GetPosition());
■■}
■}
};

class VSS : VSS_Base
{
■override bool CanEnterIronsights()
■{
■■ItemOptics optic = GetAttachedOptics();
■■if ( optic && PSO1Optic.Cast(optic) || PSO11Optic.Cast(optic) || KashtanOptic.Cast(optic) || KazuarOpt
■■■return true;
■■return super.CanEnterIronsights();
■}
};
class ASVAL : VSS_Base {};
```

```
// ---------------------------------------------
// ------ VSSRecoil.c - START --------------------
// ---------------------------------------------


class VSSRecoil: RecoilBase
{
■override void Init()
■{
■■vector point_1;
■■vector point_2;
■■vector point_3;
■■vector point_4;
■■point_1[0] = m_Player.GetRandomGeneratorSyncManager().GetRandomInRange(RandomGeneratorS
■■point_2[0] = m_Player.GetRandomGeneratorSyncManager().GetRandomInRange(RandomGeneratorS
■■point_3[0] = m_Player.GetRandomGeneratorSyncManager().GetRandomInRange(RandomGeneratorS
■■point_4[0] = m_Player.GetRandomGeneratorSyncManager().GetRandomInRange(RandomGeneratorS
■■m_HandsCurvePoints.Insert(point_1);//forms a 2 dimensional spline(z is ignored)
■■m_HandsCurvePoints.Insert(point_2);
■■m_HandsCurvePoints.Insert(point_3);
■■m_HandsCurvePoints.Insert(point_4);
■■m_HandsCurvePoints.Insert("0 0 0");
■■m_HandsOffsetRelativeTime = 0.25;■
■■
■■m_MouseOffsetRangeMin = 60;//in degrees min
■■m_MouseOffsetRangeMax = 100;//in degrees max
■■m_MouseOffsetDistance = 0.6;//how far should the mouse travel
■■m_MouseOffsetRelativeTime = 0.25;//[0..1] a time it takes to move the mouse the required distance rel
■■
■■m_CamOffsetDistance = 0.0075;
■■m_CamOffsetRelativeTime = 1;
■}
}


// ---------------------------------------------
// ------ VSSRecoil.c - END ----------------------
// ---------------------------------------------


// ---------------------------------------------
// ------ WarmthNotfr.c - START --------------------
// ---------------------------------------------


class WarmthNotfr: NotifierBase
{
■private const float ■DEC_TRESHOLD_LOW ■■■= 0;
■private const float ■DEC_TRESHOLD_MED ■■■= -0.2;
■private const float ■DEC_TRESHOLD_HIGH■■■= -0.3;
■private const float ■INC_TRESHOLD_LOW ■■■= 0;
■private const float ■INC_TRESHOLD_MED ■■■= 0.2;
■private const float ■INC_TRESHOLD_HIGH■■■= 0.3;
■■
■void WarmthNotfr(NotifiersManager manager)
■{
■■m_TendencyBufferSize = 6;
■}

■override int GetNotifierType()
■{
■■return eNotifiers.NTF_WARMTH;
■}
```

```
// ---------------------------------------------
// ------ Watchtower.c - START --------------------
// ---------------------------------------------


class Watchtower extends BaseBuildingBase
{■
■typename ATTACHMENT_BARBED_WIRE■■■= BarbedWire;
■typename ATTACHMENT_CAMONET ■■■= CamoNet;
■
■const float MAX_FLOOR_VERTICAL_DISTANCE ■■= 0.5;
■
■const float MIN_ACTION_DETECTION_ANGLE_RAD ■■= 0.35;■■//0.35 RAD = 20 DEG
■const float MAX_ACTION_DETECTION_DISTANCE ■■= 2.0;■■//meters
■
■static const string BASE_VIEW_NAME■■■■= "level_";
■static const string BASE_WALL_NAME■■■■= "_wall_";
■static const string BASE_ROOF_NAME■■■■= "_roof";
■static const int■MAX_WATCHTOWER_FLOORS■■= 3;
■static const int■MAX_WATCHTOWER_WALLS■■= 3;
■
■void Watchtower()
■{
■}
■
■override string GetConstructionKitType()
■{
■■return "WatchtowerKit";
■}■■
■
■override int GetMeleeTargetType()
■{
■■return EMeleeTargetType.NONALIGNABLE;
■}
■
■/*override void EEHitBy(TotalDamageResult damageResult, int damageType, EntityAI source, int compo
■{
■■super.EEHitBy(damageResult, damageType, source, component, dmgZone, ammo, modelPos, speedC
■■
■■if (component == -1)
■■{
■■■Print("EEHitBy: " + this + "; damageType: "+ damageType +"; source: "+ source +"; component: "+ co
■■■Print("GetDamage " + damageResult.GetDamage("","Health"));
■■■Print("GetHighestDamage " + damageResult.GetHighestDamage("Health"));
■■}
■}*/
■
■//overriden for the express purpose of handling view geometry (interaction) animations
■override void UpdateVisuals()
■{
■■super.UpdateVisuals();
■■
■■SetAnimationPhase( "level_1", 0); //always visible
■■SetAnimationPhase( "level_1_wall_1", 0); //always visible
■■SetAnimationPhase( "level_1_wall_2", 0); //always visible
■■SetAnimationPhase( "level_1_wall_3", 0); //always visible
■■
■■string part_name = "";
■■bool built = false;
■■
■■for ( int i = 1; i < MAX_WATCHTOWER_FLOORS; ++i )
■■{
■■■//roof checks
```

```
// ---------------------------------------------
// ------ WatchtowerKit.c - START --------------------
// ---------------------------------------------


class WatchtowerKit extends KitBase
{
■override bool CanReceiveAttachment(EntityAI attachment, int slotId)
■{
■■ItemBase att = ItemBase.Cast(GetInventory().FindAttachment(slotId));
■■if (att)
■■■return false;
■■
■■return super.CanReceiveAttachment(attachment, slotId);
■}

■//================================================================
■// ADVANCED PLACEMENT
■//================================================================■■■
■■
■override void OnPlacementComplete( Man player, vector position = "0 0 0", vector orientation = "0 0 0" )
■{
■■super.OnPlacementComplete( player, position, orientation );
■■
■■if ( GetGame().IsServer() )
■■{
■■■//Create watchtower
■■■Watchtower watchtower = Watchtower.Cast( GetGame().CreateObjectEx( "Watchtower", GetPosition
■■■watchtower.SetPosition( position );
■■■watchtower.SetOrientation( orientation );
■■■
■■■//make the kit invisible, so it can be destroyed from deploy UA when action ends
■■■HideAllSelections();
■■■
■■■SetIsDeploySound( true );
■■}
■}
■
■override bool DoPlacingHeightCheck()
■{
■■return true;
■}
■
■override float HeightCheckOverride()
■{
■■return 2.83;//9.56;
■}
■
■override void DisassembleKit(ItemBase item)
■{
■■if (!IsHologram())
■■{
■■■ItemBase stick = ItemBase.Cast(GetGame().CreateObjectEx("WoodenStick",GetPosition(),ECE_PLA
■■■MiscGameplayFunctions.TransferItemProperties(this, stick);
■■■stick.SetQuantity(4);
■■■Rope rope = Rope.Cast(item);
■■■CreateRope(rope);
■■}
■}
■
■//Debug menu Spawn Ground Special
■override void OnDebugSpawn()
■{
```

```
// ---------------------------------------------
// ------ WaterBottle.c - START --------------------
// ---------------------------------------------


class WaterBottle extends Bottle_Base
{
■void WaterBottle()
■{

■}
■
■void ~WaterBottle()
■{
■■
■}
■
■override bool IsContainer()
■{
■■return true;
■}
■
■override string GetPouringSoundset()
■{
■■return "emptyVessle_WaterBottle_SoundSet";
■}
■
■override string GetEmptyingLoopSoundsetHard()
■{
■■return "pour_HardGround_WatterBottle_SoundSet";
■}
■
■override string GetEmptyingLoopSoundsetSoft()
■{
■■return "pour_SoftGround_WatterBottle_SoundSet";
■}
■
■override string GetEmptyingLoopSoundsetWater()
■{
■■return "pour_Water_WatterBottle_SoundSet";
■}
■
■override string GetEmptyingEndSoundsetHard()
■{
■■return "pour_End_HardGround_WatterBottle_SoundSet";
■}
■
■override string GetEmptyingEndSoundsetSoft()
■{
■■return "pour_End_SoftGround_WatterBottle_SoundSet";
■}
■
■override string GetEmptyingEndSoundsetWater()
■{
■■return "pour_End_Water_WatterBottle_SoundSet";
■}
■
■override bool CanPutInCargo( EntityAI parent )
■{
■■if( !super.CanPutInCargo(parent) ) {return false;}■
■■if ( parent && (parent.IsKindOf("WatterBottle"))/* && !(parent.IsKindOf("Container_Base"))*/)
■■{
■■■return false;
```

```
// ---------------------------------------------
// ------ WaterPouch_ColorBase.c - START --------------------
// ---------------------------------------------


class WaterPouch_ColorBase: Bottle_Base{};



// ---------------------------------------------
// ------ WaterPouch_ColorBase.c - END ----------------------
// ---------------------------------------------


// ---------------------------------------------
// ------ Weapon.c - START --------------------
// ---------------------------------------------


/**@class■■Weapon
 * @brief■■script counterpart to engine's class Weapon
 **/
class Weapon extends InventoryItemSuper
{
■override bool IsWeapon() { return true; }
■
■/**@fn■■Synchronize
■ * @brief ■Force synchronizes the weapon state from Server to Client. Use with caution.
■ **/
■proto native void Synchronize();

■/**@fn■■GetMuzzleCount
■ * @return■number of muzzles
■ **/
■proto native int GetMuzzleCount();
■
■/**@fn■■GetMuzzleIndexFromMagazineSlot
■ * @brief ■conversion slotId -> muzzleIndex
■ * @param[in] magazineSlotId■■slotId of the magazine slot
■ * @return■muzzleIndex on success, -1 otherwise
■ **/
■//proto native int GetMuzzleIndexFromMagazineSlot (int magazineSlotId);
■
■/**@fn■■GetSlotFromMuzzleIndex
■ * @brief ■conversion muzzleIndex -> slotId
■ * @param[in] muzzleIndex■■muzzle index
■ * @return■slotId on success, -1 otherwise
■ **/
■proto native int GetSlotFromMuzzleIndex(int muzzleIndex);
■
■/**@fn■■GetCurrentMuzzle
■ * @return■returns index of current muzzle
■ **/
■proto native int GetCurrentMuzzle();
■
■/**@fn■■SetCurrentMuzzle
■ * @brief ■sets index of active muzzle
■ * @param[in] muzzleIndex
■ **/
■proto native void SetCurrentMuzzle(int muzzleIndex);
■
■proto native int GetMuzzleModeCount(int muzzleIndex);
■proto native void SetMuzzleMode(int muzzleIndex, int modeIndex);
```

```
// ---------------------------------------------
// ------ WeaponAttachMagazine.c - START --------------------
// ---------------------------------------------


class RemoveNewMagazineFromInventory extends WeaponStateBase
{
■Magazine m_newMagazine; /// magazine that will be removed from inventory
■ref InventoryLocation m_newSrc;

■void RemoveNewMagazineFromInventory (Weapon_Base w = NULL, WeaponStateBase parent = NULL
■{
■■m_newMagazine = NULL;
■■m_newSrc = NULL;
■}

■override void OnEntry (WeaponEventBase e)
■{
■■if(e)
■■{
■■■if (!m_newSrc.IsValid())
■■■■Error("[wpnfsm] " + Object.GetDebugName(m_weapon) + " RemoveNewMagazineFromInventory m

■■■if (m_newMagazine && m_newSrc && m_newSrc.IsValid())
■■■{
■■■■InventoryLocation curr = new InventoryLocation;
■■■■m_newMagazine.GetInventory().GetCurrentInventoryLocation(curr);

■■■■if (m_newSrc.GetType() == InventoryLocationType.GROUND && curr.GetType() == InventoryLocat
■■■■{
■■■■■// already in LH
■■■■}
■■■■else
■■■■{
■■■■■InventoryLocation lhand = new InventoryLocation;
■■■■■lhand.SetAttachment(e.m_player, m_newMagazine, InventorySlots.LEFTHAND);
■■■■■if (GameInventory.LocationSyncMoveEntity(m_newSrc, lhand))
■■■■■{
■■■■■■if (LogManager.IsWeaponLogEnable()) { wpnDebugPrint("[wpnfsm] " + Object.GetDebugName(
■■■■■}
■■■■■else
■■■■■■Error("[wpnfsm] " + Object.GetDebugName(m_weapon) + " RemoveNewMagazineFromInventor
■■■■}
■■■}
■■■else
■■■■Error("[wpnfsm] " + Object.GetDebugName(m_weapon) + " RemoveNewMagazineFromInventory, e
■■}
■■super.OnEntry(e);
■}

■override void OnAbort (WeaponEventBase e)
■{
■■m_newMagazine = NULL;
■■m_newSrc = NULL;

■■super.OnAbort(e);
■}

■override void OnExit (WeaponEventBase e)
■{
■■m_weapon.ShowMagazine();

■■m_newMagazine = NULL;
```

```
// ---------------------------------------------
// ------ WeaponChamberFromAttMag.c - START --------------------
// ---------------------------------------------



// load bullet from att mag (no anim)
class WeaponChamberFromAttMag extends WeaponStateBase
{
■void WeaponChamberFromAttMag (Weapon_Base w = NULL, WeaponStateBase parent = NULL, int act
■{ }

■override void OnEntry (WeaponEventBase e)
■{
■■super.OnEntry(e);
■■if(e)
■■{
■■■int mi = m_weapon.GetCurrentMuzzle();
■■■pushToChamberFromAttachedMagazine(m_weapon, mi);
■■}
■}
■override void OnExit (WeaponEventBase e)
■{
■■super.OnExit(e);
■}
};

// load bullet from att mag (no anim)
class WeaponChamberFromInnerMag extends WeaponStateBase
{
■void WeaponChamberFromInnerMag (Weapon_Base w = NULL, WeaponStateBase parent = NULL, int a
■{ }

■override void OnEntry (WeaponEventBase e)
■{
■■super.OnEntry(e);
■■if (e)
■■{
■■■int mi = m_weapon.GetCurrentMuzzle();
■■■pushToChamberFromInnerMagazine(m_weapon, mi);
■■}
■}
■override void OnExit (WeaponEventBase e)
■{
■■super.OnExit(e);
■}
};

// load bullet from att mag (no anim)
class WeaponChamberFromAttMagOnExit extends WeaponStateBase
{
■void WeaponChamberFromAttMagOnExit (Weapon_Base w = NULL, WeaponStateBase parent = NULL
■{ }

■override void OnExit (WeaponEventBase e)
■{
■■int mi = m_weapon.GetCurrentMuzzle();
■■pushToChamberFromAttachedMagazine(m_weapon, mi);

■■super.OnExit(e);
■}
};
```

```
// ---------------------------------------------
// ------ WeaponChambering.c - START --------------------
// ---------------------------------------------


// load 1 bullet
class WeaponChambering_Start extends WeaponStartAction
{
■override void OnEntry (WeaponEventBase e)
■{
■■super.OnEntry(e);
■■if (e)
■■{
■■■m_weapon.SelectionBulletHide();
■■■m_weapon.ForceSyncSelectionState();
■■}
■}
■
■override bool IsWaitingForActionFinish()
■{
■■return true;
■}
};

class WeaponChambering_Base extends WeaponStateBase
{
■float m_damage;
■string m_type;
■string m_magazineType;
■Magazine m_srcMagazine; /// source of the cartridge
■
■override bool SaveCurrentFSMState (ParamsWriteContext ctx)
■{
■■if (!super.SaveCurrentFSMState(ctx))
■■■return false;

■■if (!ctx.Write(m_damage))
■■{
■■■Error("[wpnfsm] " + Object.GetDebugName(m_weapon) + " WeaponChambering.SaveCurrentFSMSt
■■■return false;
■■}
■■if (!ctx.Write(m_type))
■■{
■■■Error("[wpnfsm] " + Object.GetDebugName(m_weapon) + " WeaponChambering.SaveCurrentFSMSt
■■■return false;
■■}
■■if (!ctx.Write(m_magazineType))
■■{
■■■Error("[wpnfsm] " + Object.GetDebugName(m_weapon) + " WeaponChambering.SaveCurrentFSMSt
■■■return false;
■■}
■■if (!ctx.Write(m_srcMagazine))
■■{
■■■Error("[wpnfsm] " + Object.GetDebugName(m_weapon) + " WeaponChambering.SaveCurrentFSMSt
■■■return false;
■■}
■■return true;
■}

■override bool LoadCurrentFSMState (ParamsReadContext ctx, int version)
■{
■■if (!super.LoadCurrentFSMState(ctx, version))
■■■return false;
```

```
// ---------------------------------------------
// ------ WeaponChamberingLooped.c - START --------------------
// ---------------------------------------------


// load x bullets
class LoopedChambering_EndLoop extends WeaponStartAction
{
■override bool IsWaitingForActionFinish () { return true; }
};

class LoopedChambering_Wait4ShowBullet2 extends WeaponStateBase
{
■override bool IsWaitingForActionFinish () { return false; }
};

class LoopedChambering extends WeaponStateBase
{
■WeaponActions m_action;
■int m_startActionType;
■int m_endActionType;
■Magazine m_srcMagazine; /// source of the cartridge
■ref InventoryLocation m_srcMagazinePrevLocation;

■ref WeaponStateBase m_start;
■ref WeaponEjectCasing m_eject;
■ref WeaponChambering_Base m_chamber;
■ref LoopedChambering_Wait4ShowBullet2 m_w4sb2;
■ref WeaponStartAction m_endLoop;
■ref BulletHide_W4T m_hideB;

■void LoopedChambering (Weapon_Base w = NULL, WeaponStateBase parent = NULL, WeaponActions
■{
■■m_action = action;
■■m_startActionType = startActionType;
■■m_endActionType = endActionType;

■■// setup nested state machine
■■m_start = new WeaponChambering_Start(m_weapon, this, m_action, m_startActionType);
■■m_eject = new WeaponEjectCasing(m_weapon, this);
■■m_chamber = new WeaponChambering_Cartridge_ChambToMag(m_weapon, this);
■■m_w4sb2 = LoopedChambering_Wait4ShowBullet2(m_weapon, this);
■■m_hideB = new BulletHide_W4T(m_weapon, this);
■■m_endLoop = new LoopedChambering_EndLoop(m_weapon, this, m_action, m_endActionType); // @
■■// events
■■WeaponEventBase■■■■■■■_fin_ = new WeaponEventHumanCommandActionFinished;
■■WeaponEventContinuousLoadBulletStart■__lS_ = new WeaponEventContinuousLoadBulletStart;
■■WeaponEventContinuousLoadBulletEnd■■__lE_ = new WeaponEventContinuousLoadBulletEnd;
■■WeaponEventAnimBulletShow■■■■__bs_ = new WeaponEventAnimBulletShow;
■■WeaponEventAnimBulletHide■■■■__bh_ = new WeaponEventAnimBulletHide;
■■WeaponEventAnimBulletEject■■■■__be_ = new WeaponEventAnimBulletEject;
■■WeaponEventAnimBulletInMagazine■■■__bM_ = new WeaponEventAnimBulletInMagazine;
■■WeaponEventAnimBulletShow2■■■■_bs2_ = new WeaponEventAnimBulletShow2;

■■m_fsm = new WeaponFSM(this); // @NOTE: set owner of the submachine fsm
■■m_fsm.AddTransition(new WeaponTransition(m_start,■__be_, m_eject));
■■m_fsm.AddTransition(new WeaponTransition(m_start,■__bs_, m_chamber));
■■m_fsm.AddTransition(new WeaponTransition(m_eject,■__bs_, m_chamber));
■■

■■m_fsm.AddTransition(new WeaponTransition(m_chamber, __bM_, m_w4sb2, NULL, new GuardAnd(n
■■m_fsm.AddTransition(new WeaponTransition(m_chamber, __bM_, m_endLoop));
■■m_fsm.AddTransition(new WeaponTransition(m_w4sb2, ■__bh_, m_hideB));
```

```
// ---------------------------------------------
// ------ WeaponCharging.c - START --------------------
// ---------------------------------------------


/**@class ■WeaponCharging
 * @brief■charging of weapon without ammo to be chambered
 */
class WeaponCharging extends WeaponStateBase
{
■WeaponActions m_action;
■int m_actionType;

■ref WeaponCharging_Start m_start;
■ref WeaponCharging_CK m_onCK;
■ref WeaponChamberFromAttMag_W4T m_chamber;
■ref WeaponEjectAllMuzzles m_eject;

■void WeaponCharging (Weapon_Base w = NULL, WeaponStateBase parent = NULL, WeaponActions ac
■{
■■m_action = action;
■■m_actionType = actionType;

■■// setup nested state machine
■■m_start = new WeaponCharging_Start(m_weapon, this, m_action, m_actionType);
■■m_onCK = new WeaponCharging_CK(m_weapon, this);
■■m_chamber = new WeaponChamberFromAttMag_W4T(m_weapon, this);
■■m_eject = new WeaponEjectAllMuzzles(m_weapon, this);

■■// events
■■WeaponEventBase __be_ = new WeaponEventAnimBulletEject;
■■WeaponEventBase __bh_ = new WeaponEventAnimBulletHide;
■■WeaponEventBase __ck_ = new WeaponEventAnimCocked;
■■WeaponEventBase _fin_ = new WeaponEventHumanCommandActionFinished;

■■m_fsm = new WeaponFSM(this); // @NOTE: set owner of the submachine fsm
■■// transitions
■■m_fsm.AddTransition(new WeaponTransition(  m_start, __be_, m_eject ));
■■m_fsm.AddTransition(new WeaponTransition(  m_start, _fin_, NULL ));
■■
■■m_fsm.AddTransition(new WeaponTransition(  m_start, __ck_, m_eject, NULL, new GuardNot(new We
■■m_fsm.AddTransition(new WeaponTransition(  m_start, __ck_, m_chamber, NULL, new WeaponGuard
■■m_fsm.AddTransition(new WeaponTransition(  m_start, __ck_, m_onCK)); // some anims do not send I
■■
■■m_fsm.AddTransition(new WeaponTransition(   m_eject, __ck_, m_chamber, NULL, new WeaponGuar
■■m_fsm.AddTransition(new WeaponTransition(   m_eject, __ck_, m_onCK));
■■m_fsm.AddTransition(new WeaponTransition(   m_eject, _fin_, NULL));
■■
■■m_fsm.AddTransition(new WeaponTransition(   m_onCK, _fin_, NULL));
■■
■■m_fsm.AddTransition(new WeaponTransition(m_chamber, _fin_, NULL));

■■m_fsm.SetInitialState(m_start);
■}
};

class WeaponCharging_Start extends WeaponStartAction
{
■override bool IsWaitingForActionFinish ()
■{
■■return true;
■}
};
```

```
// ---------------------------------------------
// ------ WeaponCleaningKit.c - START --------------------
// ---------------------------------------------


class WeaponCleaningKit: Inventory_Base {};



// ---------------------------------------------
// ------ WeaponCleaningKit.c - END ----------------------
// ---------------------------------------------


// ---------------------------------------------
// ------ WeaponDebug.c - START --------------------
// ---------------------------------------------


enum eDebugMode
{
    NORMAL,
■MUZZLE_FIRE,
■CAMERA_MUZZLE_HYBRID,
■COUNT
};



class WeaponDebug
{
■const int BUFFER_SIZE = 1000;
■const float COLLISIONS_DISTANCE_TOLERANCE = 0.01;
■const float MAX_MUZZLE_DISTANCE_TOLERANCE = 20;
■const float CAMERA_BULLET_ORIGIN_OFFSET = 1;
■const float CAMERA_TRACE_MIN_DISTANCE_TOLERANCE = 0.3;
■
■Weapon m_WeaponInHands;
■int m_BufferIndex;
■bool m_IsDrawKeyHeldDown;
■bool m_IsLMBPressed;
■bool m_IsToggleKeyPressed;
■bool m_IsFireKeyPressed;
■float m_TargetDistance;
■eDebugMode m_CurrentMode;
■vector m_AimTrailCyclic[BUFFER_SIZE];■
■vector m_AimTrailOrdered[BUFFER_SIZE];
■
■ref map<int, string> m_DebugModesNames = new map<int, string>;
■
■//ref array<Selection> m_Selections = new array<Selection>();
■
■Shape m_Shape_usti;
■Shape m_Shape_konec;
■Shape m_ShapeFireDirection1;
■Shape m_ShapeFireDirection2;
■Shape m_HitShape;
■Shape m_ShapeEye;
■Shape m_ShapeTrailLines;
■Shape m_ShapeFireDirCamera;
■Shape m_HitShape2;
■Shape m_HitShape3;
■Shape m_HitShape4;
```

```
// ---------------------------------------------
// ------ WeaponDetachingMag.c - START --------------------
// ---------------------------------------------



// detach magazine composite state
class WeaponDetachingMag_1 extends WeaponStartAction
{ };

class WeaponDetachingMag_Store extends WeaponStateBase
{
■Magazine m_magazine; /// magazine that will be detached
■ref InventoryLocation m_dst;

■override void OnEntry (WeaponEventBase e)
■{
■■//if (LogManager.IsWeaponLogEnable()) { wpnDebugPrint("[wpnfsm] " + Object.GetDebugName(m_we
■■super.OnEntry(e);
■■if (e)
■■{
■■■if (!m_magazine || !m_dst)
■■■{
■■■■Error("[wpnfsm] " + Object.GetDebugName(m_weapon) + " WeaponDetachingMag_Store, error - nc
■■■}
■■}
■}

■override void OnAbort (WeaponEventBase e)
■{
■■m_magazine = NULL;
■■m_dst = NULL;

■■super.OnAbort(e);
■}

■override void OnExit (WeaponEventBase e)
■{
■■InventoryLocation il = new InventoryLocation;
■■if (m_magazine.GetInventory().GetCurrentInventoryLocation(il))
■■{
■■■if (GameInventory.LocationSyncMoveEntity(il, m_dst))
■■■{
■■■■m_weapon.HideMagazine();
■■■■if (LogManager.IsWeaponLogEnable()) { wpnDebugPrint("[wpnfsm] " + Object.GetDebugName(m_v
■■■}
■■■else
■■■{
■■■■// @TODO: drop on gnd
■■■■Error("[wpnfsm] " + Object.GetDebugName(m_weapon) + " WeaponDetachingMag_Store, error - ca
■■■}
■■}
■■else
■■{
■■■Error("[wpnfsm] " + Object.GetDebugName(m_weapon) + " WeaponDetachingMag_Store, error - can
■■}

■■m_magazine = NULL;
■■m_dst = NULL;
■■super.OnExit(e);
■}

■override bool SaveCurrentFSMState (ParamsWriteContext ctx)
```

```
// ---------------------------------------------
// ------ WeaponEjectBullet.c - START --------------------
// ---------------------------------------------


// eject
//TODO MW delete this file
class WeaponEjectBullet_Start extends WeaponStartAction
{ };
/*
class WeaponEjectBullet_Cartridge extends WeaponStateBase
{
■Magazine m_dstMagazine; /// destination of the cartridge

■override void OnEntry (WeaponEventBase e)
■{
■■super.OnEntry(e);

■■DayZPlayer p = e.m_player;
■■int mi = m_weapon.GetCurrentMuzzle();

■■ejectBulletAndStoreInMagazine(m_weapon, mi, m_dstMagazine, p); // MP-safe
■}

■override void OnAbort (WeaponEventBase e)
■{
■■m_dstMagazine = NULL;
■■super.OnAbort(e);
■}

■override void OnExit (WeaponEventBase e)
■{
■■m_dstMagazine = NULL;
■■super.OnExit(e);
■}
};
*/
/*
class WeaponEjectBullet_Cartridge_W4T extends WeaponEjectBullet_Cartridge
{
■override bool IsWaitingForActionFinish () { return true; }
};
*/
class WeaponEjectBullet extends WeaponStateBase
{
■WeaponActions m_action;
■int m_actionType;
■Magazine m_dstMagazine; /// destination of the cartridge

■ref WeaponEjectBullet_Start m_start;
■ref WeaponEjectBullet_Cartridge_W4T m_eject; // @TODO: workaround for missing BH event
■ref BulletHide_W4T m_hideB;
■ref WeaponChamberFromAttMag_W4T m_chamber;

■void WeaponEjectBullet (Weapon_Base w = NULL, WeaponStateBase parent = NULL, WeaponActions a
■{
■■m_action = action;
■■m_actionType = actionType;

■■// setup nested state machine
■■m_start = new WeaponEjectBullet_Start(m_weapon, this, m_action, m_actionType);
■■m_eject = new WeaponEjectBullet_Cartridge_W4T(m_weapon, this); // @TODO: workaround for missi
■■m_hideB = new BulletHide_W4T(m_weapon, this);
```

```
// ---------------------------------------------
// ------ WeaponEjectCasingAndChamberFromAttMag.c - START --------------------
// ---------------------------------------------



class WeaponEjectCasing extends WeaponStateBase
{
■void WeaponEjectCasing (Weapon_Base w = NULL, WeaponStateBase parent = NULL) { }

■override void OnEntry (WeaponEventBase e)
■{
■■super.OnEntry(e);
■■if (e)
■■{
■■■if (LogManager.IsWeaponLogEnable()) { wpnDebugPrint("[wpnfsm] " + Object.GetDebugName(m_we
■■■int mi = m_weapon.GetCurrentMuzzle();
■■■if(m_weapon.IsChamberFiredOut(mi))
■■■{
■■■■m_weapon.EjectCasing(mi);
■■■}
■■■m_weapon.EffectBulletHide(mi);
■■■m_weapon.SelectionBulletHide();
■■}
■}
};

class WeaponEjectCasingMultiMuzzle extends WeaponStateBase
{
■void WeaponEjectCasingMultiMuzzle (Weapon_Base w = NULL, WeaponStateBase parent = NULL) { }

■override void OnEntry (WeaponEventBase e)
■{
■■super.OnEntry(e);
■■if (e)
■■{
■■■if (LogManager.IsWeaponLogEnable()) { wpnDebugPrint("[wpnfsm] " + Object.GetDebugName(m_we
■■■for( int i = 0; i < m_weapon.GetMuzzleCount(); i++ )
■■■{
■■■■if(m_weapon.IsChamberFiredOut(i))
■■■■{
■■■■■m_weapon.EjectCasing(i);
■■■■■m_weapon.EffectBulletHide(i);
■■■■■m_weapon.HideBullet(i);
■■■■}
■■■}
■■}
■}
■
■override bool IsWaitingForActionFinish()
■{
■■return false;
■}
};

class WeaponEjectCasing_W4T extends WeaponEjectCasingMultiMuzzle
{
■override bool IsWaitingForActionFinish () { return true; }
};

class WeaponEjectAllMuzzles extends WeaponStateBase
{
■Magazine m_dstMagazine;
```

```
// ---------------------------------------------
// ------ WeaponFire.c - START --------------------
// ---------------------------------------------


// fire
class WeaponDryFire extends WeaponStartAction
{
■float m_dtAccumulator;

■override void OnEntry (WeaponEventBase e)
■{
■■super.OnEntry(e);
■■if (e)
■■{
■■■m_dtAccumulator = 0;
■■}
■■m_weapon.ResetBurstCount();
■}

■override bool IsWaitingForActionFinish () { return true; }

■override void OnUpdate (float dt)
■{
■■m_dtAccumulator += dt;
■■DayZPlayer p;
■■Class.CastTo(p, m_weapon.GetHierarchyParent());
■■if (p)
■■{
■■■HumanInputController hic = p.GetInputController();

■■■int muzzleIndex = m_weapon.GetCurrentMuzzle();
■■■float reloadTime = m_weapon.GetReloadTime(muzzleIndex);
■■■if ( hic.IsAttackButton() && m_dtAccumulator >= reloadTime)
■■■■if (m_weapon.CanProcessWeaponEvents())
■■■■■m_weapon.ProcessWeaponEvent(new WeaponEventDryFireTimeout(p));
■■}
■}

■override void OnExit (WeaponEventBase e)
■{
■■m_dtAccumulator = 0;
■■super.OnExit(e);
■}
■
};

// fire
class WeaponFire extends WeaponStartAction
{
■float m_dtAccumulator;

■override bool IsWaitingForActionFinish () { return true; }

■override void OnEntry (WeaponEventBase e)
■{
■■if (e)
■■{
■■■m_dtAccumulator = 0;
■
■■■if (LogManager.IsWeaponLogEnable()) { wpnPrint("[wpnfsm] " + Object.GetDebugName(m_weapon)
■■■//m_weapon.Fire();
■■■int mi = m_weapon.GetCurrentMuzzle();
```

```
// ----------------------------------------------
// ------ WeaponFireAndChamberNext.c - START --------------------
// ----------------------------------------------


class WeaponFireAndChamberNext extends WeaponStateBase
{
■WeaponActions m_action;
■int m_actionType;

■float m_dtAccumulator;
■ref WeaponFire m_fire;

■void WeaponFireAndChamberNext (Weapon_Base w = NULL, WeaponStateBase parent = NULL, Weap
■{
■■m_action = action;
■■m_actionType = actionType;

■■// setup nested state machine
■■m_fire = new WeaponFireAndChamber(m_weapon, this, m_action, m_actionType);

■■// events
■■WeaponEventBase _fin_ = new WeaponEventHumanCommandActionFinished;
■■WeaponEventAnimBulletEject  __be_ = new WeaponEventAnimBulletEject;
■■WeaponEventReloadTimeout __to_ = new WeaponEventReloadTimeout;

■■m_fsm = new WeaponFSM(this); // @NOTE: set owner of the submachine fsm

■■// transitions
■■m_fsm.AddTransition(new WeaponTransition(m_fire, _fin_, NULL));
■■m_fsm.AddTransition(new WeaponTransition(m_fire, __to_, NULL));

■■m_fsm.SetInitialState(m_fire);
■}

■override void OnEntry (WeaponEventBase e)
■{
■■super.OnEntry(e);
■■if (e)
■■■m_dtAccumulator = 0;
■}

■override void OnUpdate (float dt)
■{
■■m_dtAccumulator += dt;
■■DayZPlayerImplement p;
■■Class.CastTo(p, m_weapon.GetHierarchyParent());
■■if( p )
■■{
■■■HumanInputController hic = p.GetInputController();

■■■int muzzleIndex = m_weapon.GetCurrentMuzzle();
■■■float reloadTime = m_weapon.GetReloadTime(muzzleIndex);
■■■
■■■if ( m_dtAccumulator >= reloadTime && ( hic.IsAttackButton() || (m_weapon.GetBurstCount() < m_we
■■■{
■■■■if (m_weapon.CanProcessWeaponEvents())
■■■■{
■■■■■m_weapon.ProcessWeaponEvent(new WeaponEventReloadTimeout(p));
■■■■}
■■■}
■■}
■}
```

```
// ---------------------------------------------
// ------ WeaponFireAndChamberNextFromInnerMag.c - START --------------------
// ---------------------------------------------


class WeaponFireAndChamberNextFromInnerMag extends WeaponStateBase
{
■WeaponActions m_action;
■int m_actionType;

■float m_dtAccumulator;
■ref WeaponFire m_fire;

■void WeaponFireAndChamberNextFromInnerMag (Weapon_Base w = NULL, WeaponStateBase parent
■{
■■m_action = action;
■■m_actionType = actionType;

■■// setup nested state machine
■■m_fire = new WeaponFireAndChamber(m_weapon, this, m_action, m_actionType);

■■// events
■■WeaponEventBase _fin_ = new WeaponEventHumanCommandActionFinished;
■■WeaponEventAnimBulletEject  __be_ = new WeaponEventAnimBulletEject;
■■WeaponEventReloadTimeout __to_ = new WeaponEventReloadTimeout;

■■m_fsm = new WeaponFSM(this); // @NOTE: set owner of the submachine fsm

■■// transitions
■■m_fsm.AddTransition(new WeaponTransition(m_fire, _fin_, NULL));
■■m_fsm.AddTransition(new WeaponTransition(m_fire, __to_, NULL));

■■m_fsm.SetInitialState(m_fire);
■}

■override void OnEntry (WeaponEventBase e)
■{
■■super.OnEntry(e);
■■if (e)
■■■m_dtAccumulator = 0;
■}

■override void OnUpdate (float dt)
■{
■■m_dtAccumulator += dt;
■■DayZPlayer p;
■■Class.CastTo(p, m_weapon.GetHierarchyParent());
■■if( p )
■■{
■■■HumanInputController hic = p.GetInputController();

■■■int muzzleIndex = m_weapon.GetCurrentMuzzle();
■■■float reloadTime = m_weapon.GetReloadTime(muzzleIndex);
■■■if ( hic.IsAttackButton() && m_dtAccumulator >= reloadTime)
■■■■if (m_weapon.CanProcessWeaponEvents())
■■■■■m_weapon.ProcessWeaponEvent(new WeaponEventReloadTimeout(p));
■■}
■}

■override void OnExit (WeaponEventBase e)
■{
■■m_dtAccumulator = 0;
■■super.OnExit(e);
```

```
// ---------------------------------------------
// ------ WeaponFireLast.c - START --------------------
// ---------------------------------------------



class WeaponFireLast extends WeaponStateBase
{
■WeaponActions m_action;
■int m_actionType;
■
■float m_dtAccumulator;
■ref WeaponFireWithEject m_fire;

■void WeaponFireLast (Weapon_Base w = NULL, WeaponStateBase parent = NULL, WeaponActions act
■{
■■m_action = action;
■■m_actionType = actionType;

■■// setup nested state machine
■■m_fire = new WeaponFireWithEject(m_weapon, this, m_action, m_actionType);

■■// events
■■WeaponEventBase _fin_ = new WeaponEventHumanCommandActionFinished;
■■WeaponEventReloadTimeout __to_ = new WeaponEventReloadTimeout;

■■m_fsm = new WeaponFSM(this); // @NOTE: set owner of the submachine fsm

■■// transitions
■■m_fsm.AddTransition(new WeaponTransition(m_fire, _fin_, NULL));
■■m_fsm.AddTransition(new WeaponTransition(m_fire, __to_, NULL));

■■m_fsm.SetInitialState(m_fire);
■}

■override void OnEntry (WeaponEventBase e)
■{
■■super.OnEntry(e);
■■if (e)
■■■m_dtAccumulator = 0;
■}

■override void OnExit (WeaponEventBase e)
■{
■■m_dtAccumulator = 0;
■■super.OnExit(e);
■}
};





// ---------------------------------------------
// ------ WeaponFireLast.c - END ----------------------
// ---------------------------------------------
// ---------------------------------------------
// ------ WeaponFSM.c - START --------------------
// ---------------------------------------------


/**@class■■WeaponFSM
 * @brief■■weapon finite state machine
 **/
class WeaponFSM extends HFSMBase<WeaponStateBase, WeaponEventBase, WeaponActionBase, We
{
■private static const int MAX_SYNCHRONIZE_ATTEMPTS = 12;
■private static const int MIN_SYNCHRONIZE_INTERVAL = 3000; // ms
■private static const int RESET_SYNCHRONIZE_THRESHOLD = 3600000; // ms
■private int m_SynchronizeAttempts;
■private int m_LastSynchronizeTime;
■
■protected int m_NextStateId = 0; /// counter for InternalID: each state in a fsm is assigned an unique num
■protected ref array<WeaponStateBase> m_UniqueStates = new array<WeaponStateBase>; /// unique lis

■protected void SetInternalID(WeaponStateBase state)
■{
■■if (state && state.GetInternalStateID() == -1)
■■{
■■■state.SetInternalStateID(m_NextStateId);

■■■//if (LogManager.IsWeaponLogEnable()) { wpnDebugSpam("[wpnfsm] " + Object.GetDebugName(m_
■■■m_UniqueStates.Insert(state);
■■■++m_NextStateId;
■■}
■}

■/**@fn■■AddTransition
■ * @brief■adds transition into transition table
■ * As a side effect registers the state(s) into m_UniqueStates
■ **/
■override void AddTransition(FSMTransition<WeaponStateBase, WeaponEventBase, WeaponActionBase
■{
■■super.AddTransition(t);

■■SetInternalID(t.m_srcState);
■■SetInternalID(t.m_dstState);
■}
■
■override protected ProcessEventResult ProcessLocalTransition(FSMTransition<WeaponStateBase, Wea
■{
■■ProcessEventResult res = super.ProcessLocalTransition(t, e);
■■ValidateAndRepair();
■■return res;
■}
■
■override WeaponStateBase ProcessAbortEvent(WeaponEventBase e, out ProcessEventResult result)
■{
■■WeaponStateBase res = super.ProcessAbortEvent(e, result);
■■ValidateAndRepair();
■■return res;
■}
■
■override protected ProcessEventResult ProcessAbortTransition(FSMTransition<WeaponStateBase, Wea
■{
■■ProcessEventResult res = super.ProcessAbortTransition(t, e);
■■ValidateAndRepair();
■■return res;
■}
```

```
// ---------------------------------------------
// ------ WeaponInventory.c - START --------------------
// ---------------------------------------------


/**@class■■WeaponInventory
 * @brief■■inventory for weapons
 **/
class WeaponInventory : ItemInventory
{
};

proto native bool TryFireWeapon(EntityAI weapon, int muzzleIndex);



// ---------------------------------------------
// ------ WeaponInventory.c - END ----------------------
// ---------------------------------------------


// ---------------------------------------------
// ------ WeaponManager.c - START --------------------
// ---------------------------------------------


class WeaponManager
{
■const float MAX_DROP_MAGAZINE_DISTANCE_SQ = 4;
■
■protected PlayerBase ■■■■■m_player;
■
■protected int■■■■■■■m_LastAcknowledgmentID;
■
■protected int ■■■■■■■m_PendingWeaponActionAcknowledgmentID;
■protected Magazine ■■■■■■m_PendingTargetMagazine;
■protected ref InventoryLocation■■■m_TargetInventoryLocation;
■protected int ■■■■■■■m_PendingWeaponAction;
■protected ref InventoryLocation■■■m_PendingInventoryLocation;
■
■protected bool■■■■■■■m_canEnd;
■protected bool■■■■■■■m_justStart;
■protected bool■■■■■■■m_InProgress;
■protected bool■■■■■■■m_IsEventSended;
■protected bool■■■■■■■m_WantContinue;
■protected bool■■■■■■■m_InIronSight;
■protected bool■■■■■■■m_InOptic;
■protected bool ■■■■■■■m_readyToStart;
■protected Weapon_Base■■■■■m_WeaponInHand;
■protected MagazineStorage■■■■m_MagazineInHand;
■protected ActionBase■■■■■m_ControlAction;
■protected int ■■■■■■■m_ForceEjectBulletTimestamp;
■
■protected const int ■■■■■FORCE_EJECT_BULLET_TIMEOUT = 2000;
#ifdef DIAG_DEVELOPER■
■protected int ■■■■■■■m_BurstOption;
#endif
■//Reload
■protected ref array<Magazine>■■■m_MagazinePilesInInventory;
■protected ref array<MagazineStorage>■m_MagazineStorageInInventory;
■protected ref array<Magazine>■■■m_SuitableMagazines;
■protected Magazine■■■■■■m_PreparedMagazine;
■
```

```
// ---------------------------------------------
// ------ WeaponParticles.c - START --------------------
// ---------------------------------------------


/*
■Author: Boris Vacula
■For documentation go to: DayZ Confluence -> How-to articles -> Weapon muzzle flash particle system co
■This system plays effect(s) on any weapon that is fired/jammed/ruined/...
*/

class WeaponParticlesBase // This class represents every particle effect you see in config within OnFire or
{
■bool■■■m_IlluminateWorld;
■bool■■■m_IgnoreIfSuppressed;
■bool■■■m_OnlyIfBoltIsOpen;
■int ■■■m_MuzzleIndex;
■int ■■■m_OverrideParticle;
■int ■■■m_OnlyWithinHealthLabelMin;
■int ■■■m_OnlyWithinHealthLabelMax;
■float ■■■m_OnlyWithinOverheatLimitsMin;
■float■■■m_OnlyWithinOverheatLimitsMax;
■float ■■■m_OnlyWithinRainLimitsMin;
■float■■■m_OnlyWithinRainLimitsMax;
■string ■■■m_OverrideDirectionPoint;
■string ■■■m_OnlyIfBulletIs;
■string ■■■m_OnlyIfWeaponIs;
■string ■■■m_OverridePoint;
■vector ■■■m_OverrideDirectionVector;
■vector ■■■m_PositionOffset;
■
■string ■■■m_Name;
■
■//======================================
■//  ■■PRELOAD EVERYTHING
■//======================================
■
■void WeaponParticlesBase(ItemBase muzzle_owner, string config_OnFire_entry)
■{■■
■■m_Name = config_OnFire_entry;
■■
■■// ignoreIfSuppressed
■■m_IgnoreIfSuppressed = GetGame().ConfigGetFloat(string.Format("%1 ignoreIfSuppressed", m_Name
■■
■■// onlyIfBoltIsOpen
■■m_OnlyIfBoltIsOpen = GetGame().ConfigGetFloat(string.Format("%1 onlyIfBoltIsOpen", m_Name));
■■
■■// illuminateWorld
■■m_IlluminateWorld = GetGame().ConfigGetFloat(string.Format("%1 illuminateWorld", m_Name));
■■
■■m_MuzzleIndex = -1;
■■if (GetGame().ConfigIsExisting(string.Format("%1 muzzleIndex", m_Name)))
■■{
■■■m_MuzzleIndex = GetGame().ConfigGetInt(string.Format("%1 muzzleIndex", m_Name));
■■}
■■
■■// onlyIfWeaponIs
■■m_OnlyIfWeaponIs = "";
■■GetGame().ConfigGetText(string.Format("%1 onlyIfWeaponIs", m_Name), m_OnlyIfWeaponIs);
■■
■■// onlyIfBulletIs
■■m_OnlyIfBulletIs = "";
■■GetGame().ConfigGetText(string.Format("%1 onlyIfBulletIs", m_Name), m_OnlyIfBulletIs);
```

```
// ---------------------------------------------
// ------ WeaponReChamber.c - START --------------------
// ---------------------------------------------



// rechamber (== eject cartridge + load another + store the old one)
class WeaponRechamber extends WeaponStateBase
{
■int m_actionEject;
■int m_actionTypeEject;
■int m_actionLoad;
■int m_actionTypeLoad;
■Magazine m_dstMagazine; /// destination of the ejected cartridge
■Magazine m_srcMagazine; /// source of the loaded cartridge

■ref WeaponEjectBullet m_eje;
■ref WeaponChambering m_loa;

■void WeaponRechamber (Weapon_Base w = NULL, WeaponStateBase parent = NULL, int actionEject =
■{
■■m_actionEject = actionEject;
■■m_actionTypeEject = actionTypeEject;
■■m_actionLoad = actionLoad;
■■m_actionTypeLoad = actionTypeLoad;

■■// setup nested state machine
■■m_eje = new WeaponEjectBullet(m_weapon, this, m_actionEject, m_actionTypeEject);
■■m_loa = new WeaponChambering(m_weapon, this, m_actionLoad, m_actionTypeLoad);
■■// events
■■WeaponEventBase _fin_ = new WeaponEventHumanCommandActionFinished;

■■m_fsm = new WeaponFSM(this); // @NOTE: set owner of the submachine fsm
■■m_fsm.AddTransition(new WeaponTransition(m_eje, _fin_, m_loa));
■■m_fsm.AddTransition(new WeaponTransition(m_loa, _fin_, NULL));

■■m_fsm.SetInitialState(m_eje);
■}

■override void OnEntry (WeaponEventBase e)
■{
■■if (e)
■■{
■■■if (LogManager.IsWeaponLogEnable()) { wpnDebugPrint("[wpnfsm] " + Object.GetDebugName(m_we
■■■m_srcMagazine = e.m_magazine;
■■■m_loa.m_srcMagazine = m_srcMagazine;

■■■// prepare magazine for ejected ammo
■■■int mi = m_weapon.GetCurrentMuzzle();
■■■string magazineTypeName = m_weapon.GetChamberedCartridgeMagazineTypeName(mi);
■■■float damage = 0.0;
■■■string type;
■■■if (m_weapon.GetCartridgeInfo(mi, damage, type))
■■■{
■■■■m_dstMagazine = DayZPlayerUtils.SelectStoreCartridge(e.m_player, m_weapon, mi, m_srcMagazi
■■■■if (!m_dstMagazine)
■■■■{
■■■■■Error("[wpnfsm] " + Object.GetDebugName(m_weapon) + "   WeaponRechamber - error, cannot p
■■■■}
■■■}

■■■e.m_magazine = m_dstMagazine; // @NOTE: override event mag - @TODO
■■}
```

```
// ---------------------------------------------
// ------ WeaponReplacingMagAndChamberNext.c - START --------------------
// ---------------------------------------------



/**@class■■DetachOldMagazine
 * @brief■■detaches old magazine from weapon and stores it in left hand (LH)
 **/
class DetachOldMagazine extends WeaponStateBase
{
■Magazine m_oldMagazine; /// magazine that will be detached
■ref InventoryLocation m_newDst;

■void DetachOldMagazine (Weapon_Base w = NULL, WeaponStateBase parent = NULL)
■{
■■m_oldMagazine = NULL;
■■m_newDst = NULL;
■}
■
■override void OnEntry (WeaponEventBase e)
■{
■■super.OnEntry(e);
■}

■override void OnAbort (WeaponEventBase e)
■{
■■super.OnAbort(e);
■■m_oldMagazine = NULL;
■■m_newDst = NULL;
■}

■override void OnExit (WeaponEventBase e)
■{
■■if (m_oldMagazine)
■■{
■■■InventoryLocation il = new InventoryLocation();
■■■e.m_player.GetInventory().FindFreeLocationFor( m_oldMagazine , FindInventoryLocationType.CARG
■■■
■■■if (!m_newDst || !m_newDst.IsValid() || m_newDst.GetType() == InventoryLocationType.GROUND)
■■■{
■■■■if (DayZPlayerUtils.HandleDropMagazine(e.m_player, m_oldMagazine))
■■■■{
■■■■■if (LogManager.IsWeaponLogEnable()) { wpnDebugPrint("[wpnfsm] " + Object.GetDebugName(m
■■■■}
■■■■else
■■■■■Error("[wpnfsm] " + Object.GetDebugName(m_weapon) + " DetachOldMagazine, error - cannot dr
■■■■
■■■}
■■■else
■■■{
■■■■InventoryLocation oldSrc = new InventoryLocation();
■■■■m_oldMagazine.GetInventory().GetCurrentInventoryLocation(oldSrc);
■■■■
■■■■if (GameInventory.LocationSyncMoveEntity(oldSrc, m_newDst))
■■■■{
■■■■■if (LogManager.IsWeaponLogEnable()) { wpnDebugPrint("[wpnfsm] " + Object.GetDebugName(m
■■■■}
■■■■else
■■■■■Error("[wpnfsm] " + Object.GetDebugName(m_weapon) + " DetachOldMagazine, error - cannot re
■■■}
■■}
```

```
// ---------------------------------------------
// ------ WeaponStableState.c - START --------------------
// ---------------------------------------------


/**@class■■WeaponStableState
 * @brief■  represents weapon's stable state (i.e. the basic states that the weapon will spend the most tim
 *
 * Stable states have associated animation states that they supposed to be in.
 * If they are not, the SetWeaponAnimState is called on the weapon in order to
 * set to required (configured) state
 *
 * @NOTE: OnExit from stable state, the weapon's animation state (@see
 * Weapon_Base::m_weaponAnimState) is set to -1 (unknown) In case of action
 * abort the final stable state is forced to set proper animation state
 * according to configuration (@see m_animState)
 **/

enum MuzzleState
{
■//! UNKNOWN
■U■= -1,
■//! EMPTY
■E■=  0,
■//! FIRED
■F■=  1,
■//! LOADED
■L■=  2
}

class WeaponStableState extends WeaponStateBase
{
■int m_animState;
■
■ref array<MuzzleState> m_muzzleHasBullet = new array<MuzzleState>();

■void WeaponStableState(Weapon_Base w = NULL, WeaponStateBase parent = NULL, int anim_state =
■{
■■m_animState = anim_state;
■■InitMuzzleArray();
■■ValidateMuzzleArray();
■}

■void SyncAnimState()
■{
■■int curr = m_weapon.GetWeaponAnimState();
■■if (curr != m_animState)
■■{■■■
■■■//fsmDebugSpam("[wpnfsm] " + Object.GetDebugName(m_weapon) + " synchronizing anim state: " +
■■■DayZPlayer p;
■■■if (Class.CastTo(p, m_weapon.GetHierarchyParent()))
■■■{
■■■■HumanCommandWeapons hcw = p.GetCommandModifier_Weapons();
■■■■if (hcw)
■■■■{
■■■■■hcw.SetInitState(m_animState);
■■■■■m_weapon.SetWeaponAnimState(m_animState);
■■■■■fsmDebugSpam("[wpnfsm] " + Object.GetDebugName(m_weapon) + " state=" + m_weapon.GetC
■■■■}
■■■■else
■■■■{
■■■■■Human wpnOwner = Human.Cast(m_weapon.GetHierarchyRootPlayer());
■■■■■HumanCommandWeapons.StaticSetInitState(wpnOwner, m_animState);
```

```
// ---------------------------------------------
// ------ WeaponStartAction.c - START --------------------
// ---------------------------------------------


/**@class ■WeaponStartAction
 * @brief■simple class starting animation action specified by m_action and m_actionType
 */
class WeaponStartAction extends WeaponStateBase
{
■WeaponActions m_action; /// action to be played
■int m_actionType; /// specific action sub-type

■void WeaponStartAction (Weapon_Base w = NULL, WeaponStateBase parent = NULL, WeaponActions
■{
■■m_action = action;
■■m_actionType = actionType;
■}

■override void OnEntry (WeaponEventBase e)
■{
■■super.OnEntry(e);
■■if (e)
■■{
■■■if (e.m_player)
■■■{
■■■■HumanCommandWeapons hcw = e.m_player.GetCommandModifier_Weapons();
■■■■if (hcw)
■■■■{
■■■■■HumanCommandAdditives ad = e.m_player.GetCommandModifier_Additives();
■■■■■if (ad)
■■■■■■ad.CancelModifier();
■■■■■
■■■■■hcw.StartAction(m_action, m_actionType);
■■
■■■■■if (hcw.GetRunningAction() == m_action && hcw.GetRunningActionType() == m_actionType)
■■■■■{
■■■■■■if (LogManager.IsWeaponLogEnable()) { wpnDebugPrint("HCW: playing A=" + typename.Enum
■■■■■}
■■■■■else
■■■■■■Error("HCW: NOT playing A=" + typename.EnumToString(WeaponActions, m_action) + " AT=" 
■■■■}
■■■■else
■■■■■if (LogManager.IsWeaponLogEnable()) { wpnDebugPrint("---: remote playing A=" + typename.Enu
■■■}
■■■else
■■■{
■■■■if (LogManager.IsWeaponLogEnable()) { wpnDebugPrint("---: warning, no player wants to play A=" 
■■■}
■■}
■}
■override void OnExit (WeaponEventBase e)
■{
■■super.OnExit(e);
■}
};




// ---------------------------------------------
// ------ WeaponStartAction.c - END ----------------------
// ---------------------------------------------
// ------ WeaponStateBase.c - START --------------------
// ---------------------------------------------


/**@class■WeaponStateBase
 * @brief■represent weapon state base
 *
 * Class comes with entry/update/abort/exit hooks that can be overriden in custom states
 *
 * Class is ready for hierarchic composition, i.e. this state having a sub-machine running
 * under hood. If no m_fsm member is configured, class acts as ordinary plain
 * finite machine state.
 **/
class WeaponStateBase
{
■Weapon_Base m_weapon; /// weapon that this state relates to
■WeaponStateBase m_parentState; /// hierarchical parent state of this state (or null)
■ref WeaponFSM m_fsm; /// nested state machine (or null)
■int m_InternalID = -1; /// internal state id used for load/restore

■void WeaponStateBase (Weapon_Base w = NULL, WeaponStateBase parent = NULL) { m_weapon = w

■/**@fn■■SetParentState
■ * @brief■allows construction of hierarchical state machine
■ **/
■void SetParentState (WeaponStateBase parent) { m_parentState = parent; }
■/**@fn■■GetParentState
■ * @return■state that owns this sub-state (or null if plain state)
■ **/
■WeaponStateBase GetParentState () { return m_parentState; }

■bool HasFSM () { return m_fsm != NULL; }
■WeaponFSM GetFSM () { return m_fsm; }

■void SetInternalStateID (int i) { m_InternalID = i; }
■int GetInternalStateID () { return m_InternalID; }

■bool SaveCurrentFSMState (ParamsWriteContext ctx)
■{
■■if (HasFSM())
■■{
■■■if (IsIdle())
■■■{
■■■■if (LogManager.IsWeaponLogEnable()) { wpnDebugSpam("[wpnfsm] " + Object.GetDebugName(m_
■■■■return m_fsm.SaveCurrentFSMState(ctx);
■■■}
■■■else
■■■{
■■■■// if parent state is !idle (unstable) then save whole machine
■■■■if (LogManager.IsWeaponLogEnable()) { wpnDebugSpam("[wpnfsm] " + Object.GetDebugName(m_
■■■■return m_fsm.SaveCurrentUnstableFSMState(ctx);
■■■}
■■■return false;
■■}
■■return true;
■}

■bool LoadCurrentFSMState (ParamsReadContext ctx, int version)
■{
■■if (HasFSM())
■■{
■■■if (IsIdle())
```

```
// ---------------------------------------------
// ------ WeaponStateJammed.c - START --------------------
// ---------------------------------------------


/**@class■WeaponStateJammed
 * @brief■handle jamming state set jam/unjam state for weapon
 **/
class WeaponStateJammed extends WeaponStableState
{
■/**@fn■■OnEntry
■ * @brief■called upon entry to state
■ * @NOTE■if state has (non-running) sub-machine, it's started on entry
■ * @param[in] e■the event that triggered transition to this state
■ **/
■override void OnEntry (WeaponEventBase e)
■{
■■super.OnEntry(e);
■■m_weapon.SetJammed(true);
■}


■/**@fn■■OnExit
■ * @brief called on exit from state
■ * @param[in] e■the event that triggered transition from this state
■ **/
■override void OnExit (WeaponEventBase e)
■{
■■super.OnExit(e);
■■//m_weapon.SetJammed(false);
■}
};






// ---------------------------------------------
// ------ WeaponStateJammed.c - END ----------------------
// ---------------------------------------------


// ---------------------------------------------
// ------ WeaponUnjamming.c - START --------------------
// ---------------------------------------------


class WeaponChamberFromAttMag_W4T extends WeaponChamberFromAttMag
{
■override bool IsWaitingForActionFinish () { return true; }
};

class WeaponChamberFromInnerMag_W4T extends WeaponChamberFromInnerMag
{
■override bool IsWaitingForActionFinish () { return true; }
};

class WeaponUnjamming_Start extends WeaponStartAction
{
■float m_dtAccumulator;
■float m_jamTime;

■void WeaponUnjamming_Start (Weapon_Base w = NULL, WeaponStateBase parent = NULL, WeaponA
// ---------------------------------------------
// ------ Weapon_Base.c - START --------------------
// ---------------------------------------------


/**@class■■AbilityRecord
 * @brief■■pair ( action, actionType )
 **/
class AbilityRecord
{
■int m_action; /// corresponds to Human::actions == RELOAD, MECHANISM, ...
■int m_actionType; /// corresponds to Human::actionTypes == CHAMBERING_ONEBULLET_CLOSED, M
■void AbilityRecord (int a, int at) { m_action = a; m_actionType = at; }
};

enum WeaponWithAmmoFlags
{
■//! Attached magazine will be full and no round will be chambered
■NONE = 0,
■//! Chambers bullets
■CHAMBER = 1,
■//! Maybe chambers bullets (sequential rng) example: 1 1 1 0 0 0
■CHAMBER_RNG = 2,
■//! Maybe chambers bullets (full random) example: 0 1 0 0 1 1
■CHAMBER_RNG_SPORADIC = 4,
■//! Randomizes the quantity of the bullets in the spawned magazine
■QUANTITY_RNG = 8,
■//! Fully randomizes the ammo type instead of picking one random for the entire mag (needs to have type
■AMMO_MAG_RNG = 16,
■//! Fully randomizes the ammo type instead of picking one random for all chambers (needs to have type a
■AMMO_CHAMBER_RNG = 32,
■//! Instead of randomizing when type is empty, it looks for the one which has the highest capacity
■MAX_CAPACITY_MAG = 64,
}

typedef FSMTransition<WeaponStateBase, WeaponEventBase, WeaponActionBase, WeaponGuardBase

/**@class■■Weapon_Base
 * @brief■■script base for all weapons
 *
 * @NOTE: this class is bound to core-config "Weapon_Base" config class
 **/
class Weapon_Base extends Weapon
{
■//! Full highest capacity magazine + chambered round
■const int SAMF_DEFAULT = WeaponWithAmmoFlags.CHAMBER | WeaponWithAmmoFlags.MAX_CAP
■//! Random bullet quantity + maybe chambered round
■const int SAMF_RNG = WeaponWithAmmoFlags.CHAMBER_RNG | WeaponWithAmmoFlags.QUANTIT
■
■protected const float DEFAULT_DAMAGE_ON_SHOT = 0.05;
■protected ref array<ref AbilityRecord> m_abilities = new array<ref AbilityRecord>;■■/// weapon abilities
■protected ref WeaponFSM m_fsm;■/// weapon state machine
■protected bool m_isJammed = false;
■protected bool m_LiftWeapon = false;
■protected bool m_BayonetAttached;
■protected bool m_ButtstockAttached;
■protected int m_BurstCount;
■protected int m_BayonetAttachmentIdx;
■protected int m_ButtstockAttachmentIdx;
■protected int m_weaponAnimState = -1; /// animation state the weapon is in, -1 == uninitialized
■protected int m_magazineSimpleSelectionIndex = -1;
■protected int m_weaponHideBarrelIdx = -1; //index in simpleHiddenSelections cfg array
■protected float m_DmgPerShot = 0; //default is set to zero, since C++ solution has been implemented. Se
```

```
// ---------------------------------------------
// ------ weapon_utils.c - START --------------------
// ---------------------------------------------


bool pushToChamberFromAttachedMagazine(Weapon_Base weapon, int muzzleIndex)
{
■Magazine mag = weapon.GetMagazine(muzzleIndex);
■if (mag)
■{
■■if (LogManager.IsWeaponLogEnable()) { wpnDebugPrint("[wpnfsm] " + Object.GetDebugName(weapon
■■float damage;
■■string type;
■■if (mag && mag.LocalAcquireCartridge(damage, type))
■■{
■■■weapon.SelectionBulletShow();
■■■if (LogManager.IsWeaponLogEnable()) { wpnDebugPrint("[wpnfsm] " + Object.GetDebugName(weap
■■}
■■else
■■■Error("[wpnfsm] " + Object.GetDebugName(weapon) + " chamberFromAttachedMagazine, error - can

■■if (weapon.PushCartridgeToChamber(muzzleIndex, damage, type))
■■{
■■■if (LogManager.IsWeaponLogEnable()) { wpnDebugPrint("[wpnfsm] " + Object.GetDebugName(weap
■■■return true;
■■}
■■else
■■■Error("[wpnfsm] " + Object.GetDebugName(weapon) + " chamberFromAttachedMagazine, error - can
■}
■else
■{
■■Error("[wpnfsm] " + Object.GetDebugName(weapon) + " chamberFromAttachedMagazine, error - no m
■}
■return false;
}

bool pushToChamberFromInnerMagazine(Weapon_Base weapon, int muzzleIndex)
{

■if (LogManager.IsWeaponLogEnable()) { wpnDebugPrint("[wpnfsm] " + Object.GetDebugName(weapon)
■float damage;
■string type;
■if (weapon.PopCartridgeFromInternalMagazine(muzzleIndex,damage, type))
■{
■■weapon.SelectionBulletShow();
■■if (LogManager.IsWeaponLogEnable()) { wpnDebugPrint("[wpnfsm] " + Object.GetDebugName(weapon
■}
■else
■■Error("[wpnfsm] " + Object.GetDebugName(weapon) + " chamberFromInnerMagazine, error - cannot ta

■if (weapon.PushCartridgeToChamber(muzzleIndex, damage, type))
■{
■■if (LogManager.IsWeaponLogEnable()) { wpnDebugPrint("[wpnfsm] " + Object.GetDebugName(weapon
■■return true;
■}
■else
■■Error("[wpnfsm] " + Object.GetDebugName(weapon) + " chamberFromInnerMagazine, error - cannot lo

■return false;
}

void ejectBulletAndStoreInMagazine(Weapon_Base weapon, int muzzleIndex, Magazine mag, DayZPlaye
{
```

```
// ---------------------------------------------
// ------ Weather.c - START --------------------
// ---------------------------------------------


/*!
■All classes related to game weather
*/


//--------------------------------------------------------------------------
/*!
■Weather phenomenon type
*/
enum EWeatherPhenomenon
{
■OVERCAST,
■FOG,
■RAIN
};


//--------------------------------------------------------------------------
/*!
■Weather phenomenon
*/
class WeatherPhenomenon
{
■private void WeatherPhenomenon() {}
■private void ~WeatherPhenomenon() {}
■
■//! Returns type of this phenomenon.
■proto native EWeatherPhenomenon GetType();

■//! Returns actual value of phenomenon in range <0, 1>.
■proto native float GetActual();

■//! Returns a forecast value the phenomenon is heading towards.
■proto native float GetForecast();

■/*!
■■\brief Sets the forecast.
■■\param forecast     Desired forecast value that should be met in given time.
■■\param time         A time of the next change (how long it takes in seconds to interpolate to given value).
■■\param minDuration  A minimal time in seconds the change will last.
■*/
■proto native void Set( float forecast, float time = 0, float minDuration = 0 );

■//! Reads the time in seconds when the next forecast will be computed.
■proto native float GetNextChange();
■//! Sets the time in seconds when the next forecast will be computed.
■proto native void SetNextChange( float time );

■/*!
■■\brief Reads limits of this phenomenon.
■■\param fnMin Function minimum (in range <0, 1>).
■■\param fnMax Function maximum (in range <0, 1>).
■*/
■proto void GetLimits( out float fnMin, out float fnMax );
■/*!
■■\brief Sets limits of this phenomenon.

■■Actual value of this phenomenon will be always held in range <fnMin, fnMax>
```

```
// ---------------------------------------------
// ------ WeldingMask.c - START --------------------
// ---------------------------------------------


class WeldingMask extends ClothingBase
{
■override array<int> GetEffectWidgetTypes()
■{
■■return {EffectWidgetsTypes.HELMET_OCCLUDER/*,EffectWidgetsTypes.HELMET_BREATH*/};
■}
■
■override bool CanPutAsAttachment( EntityAI parent )
■{
■■if(!super.CanPutAsAttachment(parent)) {return false;}
■■
■■Clothing eyewear = Clothing.Cast(parent.FindAttachmentBySlotName("Eyewear"));
■■if ( eyewear && eyewear.ConfigGetBool("isStrap") )
■■{
■■■return false;
■■}
■■
■■Clothing mask = Clothing.Cast(parent.FindAttachmentBySlotName("Mask"));
■■if ( mask && mask.ConfigGetBool("noHelmet") && !SantasBeard.Cast(mask) )
■■{
■■■return false;
■■}
■■
■■return true;
■}
■
■override int GetGlassesEffectID()
■{
■■return PPERequesterBank.REQ_GLASSESWELDING;
■}
}


// ---------------------------------------------
// ------ WeldingMask.c - END ----------------------
// ---------------------------------------------


// ---------------------------------------------
// ------ Well.c - START --------------------
// ---------------------------------------------


class Well extends BuildingSuper
{
■override bool IsBuilding()
■{
■■return false;
■}
■
■override bool IsWell()
■{
■■return true;
■}
■
■override float GetLiquidThroughputCoef()
■{
■■return LIQUID_THROUGHPUT_WELL;
```

```
// ---------------------------------------------
// ------ Wellies_ColorBase.c - START --------------------
// ---------------------------------------------


class Wellies_ColorBase extends Clothing {};
class Wellies_Black extends Wellies_ColorBase {};
class Wellies_Brown extends Wellies_ColorBase {};
class Wellies_Green extends Wellies_ColorBase {};
class Wellies_Grey extends Wellies_ColorBase {};


// ---------------------------------------------
// ------ Wellies_ColorBase.c - END ----------------------
// ---------------------------------------------


// ---------------------------------------------
// ------ Wet.c - START --------------------
// ---------------------------------------------


class WetMdfr: ModifierBase
{
■override void Init()
■{
■■m_TrackActivatedTime ■= false;
■■m_ID ■■■■■= eModifiers.MDF_WETNESS;
■■m_TickIntervalInactive ■= DEFAULT_TICK_TIME_INACTIVE;
■■m_TickIntervalActive ■= DEFAULT_TICK_TIME_ACTIVE;
■}
■override bool ActivateCondition(PlayerBase player)
■{
■■if (player.GetStatWet().Get() == player.GetStatWet().GetMax())
■■■return true;
■■
■■return false;
■}
■
■override bool DeactivateCondition(PlayerBase player)
■{
■■if (player.GetStatWet().Get() == player.GetStatWet().GetMin())
■■■return true;
■■
■■return false;
■}


■override void OnActivate(PlayerBase player)
■{
■■if( player.m_NotifiersManager ) player.m_NotifiersManager.ActivateByType(eNotifiers.NTF_WETNESS
■}

■override void OnReconnect(PlayerBase player)
■{
■■this.OnActivate(player);
■}


■override void OnDeactivate(PlayerBase player)
■{
■■if( player.m_NotifiersManager ) player.m_NotifiersManager.DeactivateByType(eNotifiers.NTF_WETNE
■}
```

```
// ---------------------------------------------
// ------ WetnessNotfr.c - START --------------------
// ---------------------------------------------


class WetnessNotfr: NotifierBase
{
■void WetnessNotfr(NotifiersManager manager)
■{
■■m_Active = false;
■}

■override int GetNotifierType()
■{
■■return eNotifiers.NTF_WETNESS;
■}

■override void DisplayBadge()
■{
■■DisplayElementBadge dis_elm = DisplayElementBadge.Cast(GetVirtualHud().GetElement(eDisplayEle
■■
■■if( dis_elm )
■■{
■■■dis_elm.SetLevel(eBadgeLevel.FIRST);
■■}
■■
■}

■override void HideBadge()
■{
■■DisplayElementBadge dis_elm = DisplayElementBadge.Cast(GetVirtualHud().GetElement(eDisplayEle
■■
■■if( dis_elm )
■■{
■■■dis_elm.SetLevel(eBadgeLevel.NONE);
■■}
■}
};


// ---------------------------------------------
// ------ WetnessNotfr.c - END ----------------------
// ---------------------------------------------


// ---------------------------------------------
// ------ Whetstone.c - START --------------------
// ---------------------------------------------


class Whetstone: Inventory_Base {};



// ---------------------------------------------
// ------ Whetstone.c - END ----------------------
// ---------------------------------------------
// ---------------------------------------------
// ------ WidgetEventHandler.c - START --------------------
// ---------------------------------------------


class WidgetEventHandler: ScriptedWidgetEventHandler
{
■private ref static WidgetEventHandler s_instance;

■protected ref map<Widget, ref Param> m_OnMouseButtonDownRegister;
■protected ref map<Widget, ref Param> m_OnMouseButtonUpRegister;
■protected ref map<Widget, ref Param> m_OnMouseWheel;
■protected ref map<Widget, ref Param> m_OnDropReceived;
■protected ref map<Widget, ref Param> m_OnDrag;
■protected ref map<Widget, ref Param> m_OnDrop;
■protected ref map<Widget, ref Param> m_OnDraggingOver;
■protected ref map<Widget, ref Param> m_OnMouseEnter;
■protected ref map<Widget, ref Param> m_OnMouseButtonLeave;
■protected ref map<Widget, ref Param> m_OnClick;
■protected ref map<Widget, ref Param> m_OnDoubleClick;
■protected ref map<Widget, ref Param> m_OnFocus;
■protected ref map<Widget, ref Param> m_OnFocusLost;
■protected ref map<Widget, ref Param> m_OnController;
■protected ref map<Widget, ref Param> m_OnChildAdd;
■protected ref map<Widget, ref Param> m_OnChildRemove;

■static WidgetEventHandler GetInstance()
■{
■■return s_instance;
■}

■void WidgetEventHandler()
■{
■■s_instance = this;
■■m_OnMouseButtonDownRegister = new map<Widget, ref Param>;
■■m_OnMouseButtonUpRegister = new map<Widget, ref Param>;
■■m_OnMouseWheel = new map<Widget, ref Param>;
■■m_OnDropReceived = new map<Widget, ref Param>;
■■m_OnDrag = new map<Widget, ref Param>;
■■m_OnDrop = new map<Widget, ref Param>;
■■m_OnDraggingOver = new map<Widget, ref Param>;
■■m_OnMouseEnter = new map<Widget, ref Param>;
■■m_OnMouseButtonLeave = new map<Widget, ref Param>;
■■m_OnClick = new map<Widget, ref Param>;
■■m_OnDoubleClick = new map<Widget, ref Param>;
■■m_OnFocus = new map<Widget, ref Param>;
■■m_OnFocusLost = new map<Widget, ref Param>;
■■m_OnController = new map<Widget, ref Param>;
■■m_OnChildAdd = new map<Widget, ref Param>;
■■m_OnChildRemove = new map<Widget, ref Param>;
■}
■
■void UnregisterWidget( Widget w )
■{
■■m_OnMouseButtonDownRegister.Remove( w );
■■m_OnMouseButtonUpRegister.Remove( w );
■■m_OnMouseWheel.Remove( w );
■■m_OnDropReceived.Remove( w );
■■m_OnDrag.Remove( w );
■■m_OnDrop.Remove( w );
■■m_OnDraggingOver.Remove( w );
■■m_OnMouseEnter.Remove( w );
■■m_OnMouseButtonLeave.Remove( w );
```

```
// ---------------------------------------------
// ------ WidgetLayoutName.c - START --------------------
// ---------------------------------------------


class WidgetLayoutName
{
■#ifdef PLATFORM_CONSOLE
■const string InventoryXbox■■■■■= "gui/layouts/inventory_new/xbox/day_z_inventory_new.layout";
■#else
■const string InventoryNarrow■■■■= "gui/layouts/inventory_new/narrow/day_z_inventory_new.layout";
■const string InventoryMedium■■■■= "gui/layouts/inventory_new/medium/day_z_inventory_new.layout"
■const string InventoryWide■■■■■= "gui/layouts/inventory_new/wide/day_z_inventory_new.layout";
■#endif
■
■const string HandsPreview■■■■■= "gui/layouts/inventory_new/hands_preview.layout";
■
■const string InventorySlotsContainerNarrow■= "gui/layouts/inventory_new/narrow/inventory_slots_conta
■const string InventorySlotsContainerMedium■= "gui/layouts/inventory_new/medium/inventory_slots_con
■const string InventorySlotsContainerWide■= "gui/layouts/inventory_new/wide/inventory_slots_container.
■const string InventorySlotsContainerXbox■= "gui/layouts/inventory_new/xbox/inventory_slots_container.
■
■const string CargoContainerNarrow■■■= "gui/layouts/inventory_new/narrow/cargo_container.layout";
■const string CargoContainerMedium■■■= "gui/layouts/inventory_new/medium/cargo_container.layout";
■const string CargoContainerWide■■■■= "gui/layouts/inventory_new/wide/cargo_container.layout";
■const string CargoContainerXbox■■■■= "gui/layouts/inventory_new/xbox/cargo_container.layout";
■
■const string CargoContainerRowNarrow■■= "gui/layouts/inventory_new/narrow/cargo_container_row.la
■const string CargoContainerRowMedium■■= "gui/layouts/inventory_new/medium/cargo_container_row.
■const string CargoContainerRowWide■■■= "gui/layouts/inventory_new/wide/cargo_container_row.layou
■const string CargoContainerRowXbox■■■= "gui/layouts/inventory_new/xbox/cargo_container_row.layou
■
■const string IconNarrow■■■■■■= "gui/layouts/inventory_new/narrow/icon.layout";
■const string IconMedium■■■■■■= "gui/layouts/inventory_new/medium/icon.layout";
■const string IconWide■■■■■■= "gui/layouts/inventory_new/wide/icon.layout";
■const string IconXbox■■■■■■= "gui/layouts/inventory_new/xbox/icon.layout";
■
■const string LeftAreaNarrow■■■■■= "gui/layouts/inventory_new/narrow/left_area.layout";
■const string LeftAreaMedium■■■■■= "gui/layouts/inventory_new/medium/left_area.layout";
■const string LeftAreaWide■■■■■= "gui/layouts/inventory_new/wide/left_area.layout";
■const string LeftAreaXbox■■■■■= "gui/layouts/inventory_new/xbox/left_area.layout";
■
■const string RightAreaNarrow■■■■= "gui/layouts/inventory_new/narrow/right_area.layout";
■const string RightAreaMedium■■■■= "gui/layouts/inventory_new/medium/right_area.layout";
■const string RightAreaWide■■■■■= "gui/layouts/inventory_new/wide/right_area.layout";
■const string RightAreaXbox■■■■■= "gui/layouts/inventory_new/xbox/right_area.layout";
■
■const string CollapsibleHeader■■■■= "gui/layouts/inventory_new/collapsible_header.layout";
■const string ClosableHeader■■■■■= "gui/layouts/inventory_new/closable_header.layout";
■const string HandsHeader■■■■■= "gui/layouts/inventory_new/hands_header.layout";
■const string HandsArea■■■■■■= "gui/layouts/inventory_new/hands_area.layout";
■const string Container■■■■■■= "gui/layouts/inventory_new/container.layout";
■const string CollapsibleContainer■■■= "gui/layouts/inventory_new/collapsible_container.layout";
■const string ClosableContainer■■■■= "gui/layouts/inventory_new/closable_container.layout";
■const string AttachmentsGroupContainer■■= "gui/layouts/inventory_new/attachments_group_container.
};



// ---------------------------------------------
// ------ WidgetLayoutName.c - END ----------------------
// ---------------------------------------------
// ---------------------------------------------
// ------ Winchester70.c - START --------------------
// ---------------------------------------------


class Winchester70_Base : BoltActionRifle_InnerMagazine_Base
{
■override RecoilBase SpawnRecoilObject()
■{
■■return new Winchester70Recoil(this);
■}
■
■■■
■//Debug menu Spawn Ground Special
■override void OnDebugSpawn()
■{
■■super.OnDebugSpawn();
■■EntityAI entity;
■■if ( Class.CastTo(entity, this) )
■■{
■■■entity.GetInventory().CreateInInventory( "HuntingOptic" );■
■■■entity.SpawnEntityOnGroundPos("Ammo_308Win", entity.GetPosition());
■■}
■}
};


// ---------------------------------------------
// ------ Winchester70.c - END ----------------------
// ---------------------------------------------


// ---------------------------------------------
// ------ Winchester70Recoil.c - START --------------------
// ---------------------------------------------


class Winchester70Recoil: RecoilBase
{
■override void Init()
■{
■■vector point_1;
■■vector point_2;
■■vector point_3;
■■vector point_4;
■■point_1[0] = m_Player.GetRandomGeneratorSyncManager().GetRandomInRange(RandomGeneratorS
■■point_2[0] = m_Player.GetRandomGeneratorSyncManager().GetRandomInRange(RandomGeneratorS
■■point_3[0] = m_Player.GetRandomGeneratorSyncManager().GetRandomInRange(RandomGeneratorS
■■point_4[0] = m_Player.GetRandomGeneratorSyncManager().GetRandomInRange(RandomGeneratorS
■■m_HandsCurvePoints.Insert(point_1);//forms a 2 dimensional spline(z is ignored)
■■m_HandsCurvePoints.Insert(point_2);
■■m_HandsCurvePoints.Insert(point_3);
■■m_HandsCurvePoints.Insert(point_4);
■■m_HandsCurvePoints.Insert("0 0 0");
■■m_HandsOffsetRelativeTime = 0.125;
■■
■■m_MouseOffsetRangeMin = 80;//in degrees min
■■m_MouseOffsetRangeMax = 100;//in degrees max
■■m_MouseOffsetDistance = 1.25;//how far should the mouse travel
■■m_MouseOffsetRelativeTime = 0.0625;//[0..1] a time it takes to move the mouse the required distance
■
■■m_CamOffsetDistance = 0.02;
■■m_CamOffsetRelativeTime = 0.125;
```

```
// ---------------------------------------------
// ------ WitchHat.c - START --------------------
// ---------------------------------------------


class WitchHat extends Clothing
{
■override void SetActions()
■{
■■super.SetActions();
■■AddAction(ActionWringClothes);
■}
};



// ---------------------------------------------
// ------ WitchHat.c - END ----------------------
// ---------------------------------------------


// ---------------------------------------------
// ------ WitchHood.c - START --------------------
// ---------------------------------------------


class WitchHood extends Clothing
{
■override void SetActions()
■{
■■super.SetActions();
■■AddAction(ActionWringClothes);
■}
};



// ---------------------------------------------
// ------ WitchHood.c - END ----------------------
// ---------------------------------------------


// ---------------------------------------------
// ------ WolfMask.c - START --------------------
// ---------------------------------------------


class WolfMask extends ClothingBase
{
■override bool CanPutAsAttachment( EntityAI parent )
■{
■■if(!super.CanPutAsAttachment(parent)) {return false;}
■■bool headgear_present = false;
■■
■■if ( parent.FindAttachmentBySlotName( "Headgear" ) )
■■{
■■■headgear_present = parent.FindAttachmentBySlotName( "Headgear" ).ConfigGetBool( "noMask" );
■■}
■■
■■if ( ( GetNumberOfItems() == 0 || !parent || parent.IsMan() ) && !headgear_present )
■■{
■■■return true;
■■}
```

```
// ---------------------------------------------
// ------ WolfSteakMeat.c - START --------------------
// ---------------------------------------------


class WolfSteakMeat extends Edible_Base
{
■override bool CanBeCooked()
■{
■■return true;
■}■■
■
■override bool CanBeCookedOnStick()
■{
■■return true;
■}
■
■override bool IsMeat()
■{
■■return true;
■}
■
■override bool CanDecay()
■{
■■return true;
■}
■
■override void SetActions()
■{
■■super.SetActions();
■■
■■AddAction(ActionForceFeed);
■■AddAction(ActionEatMeat);
■■
■■AddAction(ActionCreateIndoorFireplace);
■■AddAction(ActionCreateIndoorOven);
■}
}



// ---------------------------------------------
// ------ WolfSteakMeat.c - END ----------------------
// ---------------------------------------------


// ---------------------------------------------
// ------ WomanSuit_ColorBase.c - START --------------------
// ---------------------------------------------


class WomanSuit_ColorBase extends Clothing
{
■override void SetActions()
■{
■■super.SetActions();
■■AddAction(ActionWringClothes);
■}
};
class WomanSuit_Beige extends WomanSuit_ColorBase {};
class WomanSuit_Black extends WomanSuit_ColorBase {};
class WomanSuit_Blue extends WomanSuit_ColorBase {};
class WomanSuit_Brown extends WomanSuit_ColorBase {};
```

```
// ---------------------------------------------
// ------ WoodAxe.c - START --------------------
// ---------------------------------------------


class WoodAxe extends ToolBase
{
■override void SetActions()
■{
■■super.SetActions();
■■
■■//AddAction(ActionBuildPartSwitch);
■■AddAction(ActionMineTree);
■■AddAction(ActionMineTreeBark);
■■AddAction(ActionMineBush);
■■//AddAction(ActionRepairPart);
■■AddAction(ActionDismantlePart);
■■//AddAction(ActionBuildPart);
■■AddAction(ActionUnrestrainTarget);
■■AddAction(ActionSkinning);
■}
}


// ---------------------------------------------
// ------ WoodAxe.c - END ----------------------
// ---------------------------------------------


// ---------------------------------------------
// ------ WoodBase.c - START --------------------
// ---------------------------------------------


enum EHarvestType
{
■NORMAL,
■BARK
}

class WoodBase extends Plant
{
■
■static bool ■m_IsCuttable;
■static int ■■m_PrimaryDropsAmount = -1;
■static int ■■m_SecondaryDropsAmount = -1;
■static float ■m_ToolDamage = -1.0;
■static float ■m_CycleTimeOverride = -1.0;
■static string ■m_PrimaryOutput = ""; //some nonsensical item for debugging purposes
■static string ■m_SecondaryOutput = ""; //some nonsensical item for debugging purposes
■static string ■m_BarkType = "";
■
■/*
■ bool ■m_IsCuttable;
■ int ■m_PrimaryDropsAmount = -1;
■ int ■m_SecondaryDropsAmount = -1;
■ float ■m_ToolDamage = -1.0;
■ float ■m_CycleTimeOverride = -1.0;
■ string ■m_PrimaryOutput = ""; //some nonsensical item for debugging purposes
■ string ■m_SecondaryOutput = ""; //some nonsensical item for debugging purposes
■ string ■m_BarkType = "";
■*/
■
```

```
// ---------------------------------------------
// ------ WoodenLog.c - START --------------------
// ---------------------------------------------


class WoodenLog extends ItemBase
{
■override void SetActions()
■{
■■super.SetActions();
■■
■■AddAction(ActionAttachToConstruction);
■■AddAction(ActionAttachOnSelection);
■}
}


// ---------------------------------------------
// ------ WoodenLog.c - END ----------------------
// ---------------------------------------------


// ---------------------------------------------
// ------ WoodenPlank.c - START --------------------
// ---------------------------------------------


class WoodenPlank extends ItemBase
{
■override void SetActions()
■{
■■super.SetActions();
■■
■■AddAction(ActionClapBearTrapWithThisItem);
■■AddAction(ActionAttachToConstruction);
■■AddAction(ActionRepairCarChassis);
■■AddAction(ActionAttachOnSelection);
■}
}


// ---------------------------------------------
// ------ WoodenPlank.c - END ----------------------
// ---------------------------------------------


// ---------------------------------------------
// ------ WoodenStick.c - START --------------------
// ---------------------------------------------


class WoodenStick extends ItemBase
{
■override void SetActions()
■{
■■super.SetActions();

■■AddAction(ActionCreateIndoorFireplace);
■■AddAction(ActionCreateIndoorOven);
■■AddAction(ActionAttach);
■■AddAction(ActionDetach);
■■AddAction(ActionAttachToConstruction);
■■
```

```
// ---------------------------------------------
// ------ WoolCoat_ColorBase.c - START --------------------
// ---------------------------------------------


class WoolCoat_ColorBase extends Clothing
{
■override void SetActions()
■{
■■super.SetActions();
■■AddAction(ActionWringClothes);
■}
};
class WoolCoat_Black extends WoolCoat_ColorBase {};
class WoolCoat_Red extends WoolCoat_ColorBase {};
class WoolCoat_Blue extends WoolCoat_ColorBase {};
class WoolCoat_Green extends WoolCoat_ColorBase {};
class WoolCoat_Beige extends WoolCoat_ColorBase {};
class WoolCoat_RedCheck extends WoolCoat_ColorBase {};
class WoolCoat_BlackCheck extends WoolCoat_ColorBase {};
class WoolCoat_BlueCheck extends WoolCoat_ColorBase {};
class WoolCoat_GreyCheck extends WoolCoat_ColorBase {};
class WoolCoat_BrownCheck extends WoolCoat_ColorBase {};


// ---------------------------------------------
// ------ WoolCoat_ColorBase.c - END ----------------------
// ---------------------------------------------


// ---------------------------------------------
// ------ workbenchApi.c - START --------------------
// ---------------------------------------------


typedef int[] WBModuleDef;
typedef int[] ScriptEditor;
typedef int[] ResourceBrowser;
typedef int[] WorldEditor;

class Workbench
{
■static proto native WBModuleDef GetModule(string type);
■static proto native bool OpenModule(string type);
■static proto native bool CloseModule(string type);
■static proto native void Dialog(string caption, string text);
■static proto int ScriptDialog(string caption, string text, Class data);
■static proto bool SearchResources(string filter, func callback);
■static proto native int RunCmd(string command, bool wait = false);
■static proto void GetCwd(out string currentDir);
■static proto bool GetAbsolutePath(string relativePath, out string absPath);
};

class WBModuleDef
{
■proto native external bool SetOpenedResource(string filename);
■proto native external int GetNumContainers();
■proto native external BaseContainer GetContainer(int index = 0);
■proto external bool GetCmdLine(string name, out string value);
■proto native external bool Save();
■proto native external bool Close();
};
```

```
// ---------------------------------------------
// ------ WorkingBoots_ColorBase.c - START --------------------
// ---------------------------------------------


class WorkingBoots_ColorBase extends Clothing {};
class WorkingBoots_Grey extends WorkingBoots_ColorBase {};
class WorkingBoots_Brown extends WorkingBoots_ColorBase {};
class WorkingBoots_Green extends WorkingBoots_ColorBase {};
class WorkingBoots_Yellow extends WorkingBoots_ColorBase {};
class WorkingBoots_Beige extends WorkingBoots_ColorBase {};


// ---------------------------------------------
// ------ WorkingBoots_ColorBase.c - END ----------------------
// ---------------------------------------------


// ---------------------------------------------
// ------ WorkingGloves_ColorBase.c - START --------------------
// ---------------------------------------------


class WorkingGloves_ColorBase extends Clothing {};
class WorkingGloves_Black extends WorkingGloves_ColorBase {};
class WorkingGloves_Beige extends WorkingGloves_ColorBase {};
class WorkingGloves_Brown extends WorkingGloves_ColorBase {};
class WorkingGloves_Yellow extends WorkingGloves_ColorBase {};


// ---------------------------------------------
// ------ WorkingGloves_ColorBase.c - END ----------------------
// ---------------------------------------------


// ---------------------------------------------
// ------ World.c - START --------------------
// ---------------------------------------------


class World: Managed
{
■//proto private void ~World();
■//proto private void World();

■proto void CheckSoundObstruction(EntityAI source,  bool inSource, out float obstruction, out float occlus

■proto native void■GetPlayerList(out array<Man> players);
■
■/**
■\brief Get actual ingame world time
■■\param year
■■\param month in range <1, 12>
■■\param day in range <1, 31>
■■\param hour in range <0, 23>
■■\param minute in range <0, 59>
■■@code
■■■int year, month, day, hour, minute;
■■■GetGame().GetWorld().GetDate(year, month, day, hour, minute);
■■@endcode
■*/
■proto void■■■GetDate(out int year, out int month, out int day, out int hour, out int minute);
■
```

```
// ---------------------------------------------
// ------ WorldContainer_Base.c - START --------------------
// ---------------------------------------------


class WorldContainer_Base extends ItemBase
{
}


// ---------------------------------------------
// ------ WorldContainer_Base.c - END ----------------------
// ---------------------------------------------


// ---------------------------------------------
// ------ WorldData.c - START --------------------
// ---------------------------------------------


//! Keeps information about currently loaded world, like temperature
class WorldData
{
■protected float m_DayTemperature;■// legacy, no longer used
■protected float m_NightTemperature;■// legacy, no longer used
■protected Weather m_Weather;
■protected float m_EnvironmentTemperature;
■protected float m_Timer;
■protected float■m_MaxTemps[12];
■protected float m_MinTemps[12];
■protected float m_Sunrise_Jan;
■protected float m_Sunset_Jan;
■protected float m_Sunrise_Jul;
■protected float m_Sunset_Jul;
■protected ref array<vector> m_FiringPos; // Where we should fire from. On Init set the relevant data

■void WorldData()
■{
■■Init();
■}
■
■void Init()
■{
■■m_DayTemperature = 10; ■// legacy, no longer used
■■m_NightTemperature = 6;■// legacy, no longer used
■■m_Weather = g_Game.GetWeather();
■■m_EnvironmentTemperature = 12.0;
■■m_Timer = 0.0;
■■m_Sunrise_Jan = 8.54;
■■m_Sunset_Jan = 15.52;
■■m_Sunrise_Jul = 3.26;
■■m_Sunset_Jul = 20.73;
■■m_MaxTemps = {3,5,7,14,19,24,26,25,21,16,10,5};
■■m_MinTemps = {-3,-2,0,4,9,14,18,17,12,7,4,0};
■}

■float GetApproxSunriseTime( float monthday )
■{
■■if ( monthday <= 8.0 )
■■■return ( ( m_Sunrise_Jan - m_Sunrise_Jul ) / ( 8 - 1 ) ) * ( 1 - monthday ) + m_Sunrise_Jan;
■■else
■■■return ( ( ( monthday - 8 ) * ( m_Sunrise_Jan - m_Sunrise_Jul ) ) / ( 13 - 8 ) ) + m_Sunrise_Jul;
■}
```

```
// ---------------------------------------------
// ------ WorldLighting.c - START --------------------
// ---------------------------------------------


class WorldLighting
{
■string lighting_default = "dz\\data\\lighting\\lighting_default.txt";
■string lighting_darknight = "dz\\data\\lighting\\lighting_darknight.txt";
■
■void WorldLighting() {}
■void ~WorldLighting() {}
■■
■// sets global lighting config by given value (sent from server, defined in server config)
■void SetGlobalLighting( int lightingID )
■{
■■switch ( lightingID )
■■{
■■■case 0:
■■■■GetGame().GetWorld().LoadNewLightingCfg( lighting_default );
■■■■break;
■■■■
■■■case 1:
■■■■GetGame().GetWorld().LoadNewLightingCfg( lighting_darknight );
■■■■break;
■■}
■}
}

/*modded class WorldLighting
{
■protected string lighting_modded = "your\\path\\to\\lighting\\cfg.txt";
■
■override void SetGlobalLighting( int lightingID )
■{
■■switch ( lightingID )
■■{
■■■case 3:
■■■■GetGame().GetWorld().LoadNewLightingCfg( lighting_modded );
■■■■break;
■■}
■}
}*/


// ---------------------------------------------
// ------ WorldLighting.c - END ----------------------
// ---------------------------------------------


// ---------------------------------------------
// ------ WorldsMenu.c - START --------------------
// ---------------------------------------------


#ifdef GAME_TEMPLATE

[EditorAttribute("box", "GameLib/Scripted", "Worlds menu", "-0.25 -0.25 -0.25", "0.25 0.25 0.25", "255 0 0 2
class WorldsMenuClass
{

}
```

```
// ---------------------------------------------
// ------ Worm.c - START --------------------
// ---------------------------------------------


class Worm extends Edible_Base
{
■override bool CanBeCookedOnStick()
■{
■■return false;
■}

■override bool CanBeCooked()
■{
■■return false;
■}■
■
■override bool IsMeat()
■{
■■return true;
■}
}



// ---------------------------------------------
// ------ Worm.c - END ----------------------
// ---------------------------------------------


// ---------------------------------------------
// ------ WoundAgent.c - START --------------------
// ---------------------------------------------


class WoundAgent extends AgentBase
{
■static const float RESISTANCE_STAGE_1 = 1;
■static const float RESISTANCE_STAGE_2 = 0.5;
■
■override void Init()
■{
■■m_Type ■■■■■= eAgents.WOUND_AGENT;
■■m_Invasibility ■■■= 0.208;//to reach 250 in 20 mins
■■m_TransferabilityIn■■= 1;
■■m_TransferabilityOut■= 0;
■■m_AntibioticsResistance = 0.5;//override in a func. GetAntiboticsResistance()
■■m_MaxCount ■■■■= 500;
■■m_Potency ■■■■= EStatLevels.GREAT;
■■m_DieOffSpeed ■■■= 1;
■}
■
■override float GetAntibioticsResistanceEx(PlayerBase player)
■{
■■if(player.GetModifiersManager().IsModifierActive(eModifiers.MDF_WOUND_INFECTION1))
■■■return RESISTANCE_STAGE_1;
■■else
■■■return RESISTANCE_STAGE_2;
■}
■
■override bool GrowDuringAntibioticsAttack(PlayerBase player)
■{
■■if(player.GetModifiersManager().IsModifierActive(eModifiers.MDF_WOUND_INFECTION1))
```

```
// ---------------------------------------------
// ------ WoundInfection.c - START --------------------
// ---------------------------------------------


class WoundInfectionMdfr: ModifierBase
{
■static const int AGENT_THRESHOLD_ACTIVATE = 100;
■static const int AGENT_THRESHOLD_DEACTIVATE = 20;
■
■void WoundInfectionMdfr()
■{
■■Error("[ERROR] :: WoundInfectionMdfr is deprecated.");
■}
};



class WoundInfectStage1Mdfr: ModifierBase
{
■static const int AGENT_THRESHOLD_ACTIVATE = 1;
■static const int AGENT_THRESHOLD_DEACTIVATE = 250;
■static const int AGENT_THRESHOLD_FEVER = 250;
■
■static const int PAIN_EVENT_INTERVAL_MIN = 18;
■static const int PAIN_EVENT_INTERVAL_MAX = 26;
■
■protected float m_NextEvent;
■protected float m_Time;
■
■override void Init()
■{
■■m_TrackActivatedTime■= false;
■■m_ID ■■■■■= eModifiers.MDF_WOUND_INFECTION1;
■■m_TickIntervalInactive ■= DEFAULT_TICK_TIME_INACTIVE;
■■m_TickIntervalActive ■= DEFAULT_TICK_TIME_ACTIVE;
■■m_SyncID■■■■= eModifierSyncIDs.MODIFIER_SYNC_WOUND_INFECT_1;
■}
■
■override string GetDebugText()
■{
■■return ("Activate threshold: "+AGENT_THRESHOLD_ACTIVATE + "| " +"Deativate threshold: "+AGENT
■}
■
■override protected bool ActivateCondition(PlayerBase player)
■{
■■if( player.GetSingleAgentCount(eAgents.WOUND_AGENT) >= AGENT_THRESHOLD_ACTIVATE &&
■■{
■■■return true;
■■}
■■else
■■{
■■■return false;
■■}
■}

■override protected void OnActivate(PlayerBase player)
■{
■■player.IncreaseDiseaseCount();
■■m_NextEvent = Math.RandomFloatInclusive( PAIN_EVENT_INTERVAL_MIN, PAIN_EVENT_INTERVA
■}

■override protected void OnDeactivate(PlayerBase player)
```

```
// ---------------------------------------------
// ------ WoundInfection2.c - START --------------------
// ---------------------------------------------


class WoundInfectStage2Mdfr: ModifierBase
{
■static const int AGENT_THRESHOLD_ACTIVATE = 250;
■static const int AGENT_THRESHOLD_DEACTIVATE = 0;

■static const int PAIN_EVENT_INTERVAL_MIN = 6;
■static const int PAIN_EVENT_INTERVAL_MAX = 12;
■
■
■static const float DAMAGE_PER_SEC = 0.04;
■
■protected float m_NextEvent;
■protected float m_Time;
■
■override void Init()
■{
■■m_TrackActivatedTime■= false;
■■m_ID ■■■■■= eModifiers.MDF_WOUND_INFECTION2;
■■m_TickIntervalInactive ■= DEFAULT_TICK_TIME_INACTIVE;
■■m_TickIntervalActive ■= DEFAULT_TICK_TIME_ACTIVE;
■■m_SyncID■■■■= eModifierSyncIDs.MODIFIER_SYNC_WOUND_INFECT_2;
■}
■
■override string GetDebugText()
■{
■■return ("Activate threshold: "+AGENT_THRESHOLD_ACTIVATE + "| " +"Deativate threshold: "+AGENT
■}
■
■override protected bool ActivateCondition(PlayerBase player)
■{
■■if(player.GetSingleAgentCount(eAgents.WOUND_AGENT) >= AGENT_THRESHOLD_ACTIVATE)
■■{
■■■return true;
■■}
■■else
■■{
■■■return false;
■■}
■}

■override protected void OnActivate(PlayerBase player)
■{
■■player.IncreaseDiseaseCount();
■■m_NextEvent = Math.RandomFloatInclusive( PAIN_EVENT_INTERVAL_MIN, PAIN_EVENT_INTERVA
■■
■■SymptomBase shivers = player.GetSymptomManager().QueueUpSecondarySymptomEx(SymptomIDs
■■■if ( shivers )
■■■{
■■■■CachedObjectsParams.PARAM1_INT.param1 = 3;
■■■■shivers.SetParam(CachedObjectsParams.PARAM1_INT);
■■■}
■■
■}

■override protected void OnDeactivate(PlayerBase player)
■{
■■player.DecreaseDiseaseCount();
■■player.GetSymptomManager().RemoveSecondarySymptom(SymptomIDs.SYMPTOM_HAND_SHIVER
// ---------------------------------------------
// ------ Wreck_MI8.c - START --------------------
// ---------------------------------------------


//New russian helicopter crash site
class Wreck_Mi8_Crashed extends CrashBase
{
■void Wreck_Mi8_Crashed()
■{
■■if ( !GetGame().IsDedicatedServer() )
■■{
■■■m_ParticleEfx = ParticleManager.GetInstance().PlayOnObject(ParticleList.SMOKING_HELI_WRECK
■■}
■}
}

//Old Russian helicopter crash site
class Wreck_Mi8 extends CrashBase
{

}


// ---------------------------------------------
// ------ Wreck_MI8.c - END ----------------------
// ---------------------------------------------


// ---------------------------------------------
// ------ Wreck_SantasSleigh.c - START --------------------
// ---------------------------------------------


//Christmas Event: Santa's Sleigh
class Wreck_SantasSleigh extends CrashBase
{
■XmasSleighLight ■■m_SleighLight;
■
■int m_MaxDeersAmount = 4;
■int m_MinDeersAmount = 2;
■int m_MaxDeersSpawnRange = 25;
■int m_MinDeersSpawnRange = 5;

■void Wreck_SantasSleigh()
■{
■■if ( !GetGame().IsDedicatedServer() )
■■{
■■■//particles - Aurora trail
■■■m_ParticleEfx = ParticleManager.GetInstance().PlayOnObject(ParticleList.AURORA_SANTA_WREC
■■■
■■■//lights - green light
■■■m_SleighLight = XmasSleighLight.Cast( ScriptedLightBase.CreateLight( XmasSleighLight, Vector(0,
■■■m_SleighLight.AttachOnMemoryPoint( this, "light" );
■■}
■}
■
■// needs to have the soundset registered in CrashBase.Init()
■override string GetSoundSet()
■{
■■return "SledgeCrash_Distant_SoundSet";
■}
■
```

```
// ---------------------------------------------
// ------ Wreck_UH1Y.c - START --------------------
// ---------------------------------------------


class Wreck_UH1Y extends CrashBase
{
■void Wreck_UH1Y()
■{
■■if ( !GetGame().IsDedicatedServer() )
■■{
■■■m_ParticleEfx = ParticleManager.GetInstance().PlayOnObject(ParticleList.SMOKING_HELI_WRECK
■■}
■}
}


// ---------------------------------------------
// ------ Wreck_UH1Y.c - END ----------------------
// ---------------------------------------------


// ---------------------------------------------
// ------ Wrench.c - START --------------------
// ---------------------------------------------


class Wrench: Inventory_Base
{
■override void SetActions()
■{
■■super.SetActions();
■■
■■AddAction(ActionLockAttachment);
■■AddAction(ActionMineRock1H);
■}
};



// ---------------------------------------------
// ------ Wrench.c - END ----------------------
// ---------------------------------------------


// ---------------------------------------------
// ------ WriteLetter.c - START --------------------
// ---------------------------------------------


class WriteLetter extends RecipeBase
{■
■override void Init()
■{
■■m_Name = "#write_note";
■■m_IsInstaRecipe = true;//should this recipe be performed instantly without animation
■■m_AnimationLength = 1.5;//animation length in relative time units
■■m_Specialty = -0.01;// value > 0 for roughness, value < 0 for precision
■■
■■
■■//conditions
■■m_MinDamageIngredient[0] = -1;//-1 = disable check
■■m_MaxDamageIngredient[0] = 3;//-1 = disable check
```

```
// ---------------------------------------------
// ------ WrittenNoteData.c - START --------------------
// ---------------------------------------------


class WrittenNoteData
{
■protected ItemBase ■m_WritingImplement;
■protected ItemBase■m_Paper;
■//protected int ■■m_Handwriting = -1;
■protected string ■m_SimpleText;
■
■void WrittenNoteData(ItemBase parent)
■{
■■
■}
■
■void OnRPC( PlayerIdentity sender, int rpc_type, ParamsReadContext  ctx)
■{
■■Param1<string> param = new Param1<string>("");
■■
■■//sent from server, executed on client
■■if (rpc_type == ERPCs.RPC_WRITE_NOTE)
■■{
■■■if (ctx.Read(param))
■■■{
■■■■SetNoteText(param.param1);
■■■}
■■■
■■■g_Game.GetMission().SetNoteMenu(g_Game.GetUIManager().EnterScriptedMenu(MENU_NOTE, G
■■■
■■■ItemBase pen;
■■■ItemBase paper;
■■■//int handwriting;
■■■
■■■if (GetNoteInfo(pen,paper))
■■■{
■■■■g_Game.GetMission().GetNoteMenu().InitNoteWrite(paper,pen,m_SimpleText);
■■■}
■■}
■■//sent from client (NoteMenu), executed on server
■■if (rpc_type == ERPCs.RPC_WRITE_NOTE_CLIENT)
■■{
■■■if (ctx.Read(param))
■■■{
■■■■string old_string = m_SimpleText;
■■■■SetNoteText(param.param1);
■■■■DepleteWritingImplement(m_WritingImplement,old_string,m_SimpleText);
■■■}
■■}
■■if (rpc_type == ERPCs.RPC_READ_NOTE)
■■{
■■■if (ctx.Read(param))
■■■{
■■■■SetNoteText(param.param1);
■■■}
■■■
■■■g_Game.GetMission().SetNoteMenu(g_Game.GetUIManager().EnterScriptedMenu(MENU_NOTE, G
■■■g_Game.GetMission().GetNoteMenu().InitNoteRead(m_SimpleText);
■■}
■}
■
■void InitNoteInfo(ItemBase pen = null, ItemBase paper = null)
```

```
// ---------------------------------------------
// ------ XboxDemoGame.c - START --------------------
// ---------------------------------------------


//global variables for xbox demo purposes
int g_DemoPlayerClass ■= 0;
int g_DemoWeather ■■= 0;
int g_DemoTimeOfDay ■= 0;
//


// ---------------------------------------------
// ------ XboxDemoGame.c - END ----------------------
// ---------------------------------------------


// ---------------------------------------------
// ------ XmasLights.c - START --------------------
// ---------------------------------------------


class XmasLights extends Inventory_Base
{
■private ItemBase m_AttachedTo = NULL;
■
■void XmasLights()
■{
■■TurnOffItem( this );
■}
■
■override bool IsElectricAppliance()
■{
■■return true;
■}

■override void OnWorkStart()
■{
■■TurnOnItem( m_AttachedTo );
■■TurnOnItem( this );
■}

■override void OnWorkStop()
■{
■■TurnOffItem( m_AttachedTo );
■■TurnOffItem( this );
■}

■void AttachToObject(ItemBase parent)
■{
■■//SetPilotLight( false );■
■■
■■m_AttachedTo = parent;
■■TurnOnItem( parent );

■■if ( GetCompEM().IsPlugged() )
■■{
■■■parent.SetAnimationPhase( "Cord_plugged", 0);
■■■parent.SetAnimationPhase( "Cord_folded", 1);
■■}
■■else
■■{
■■■parent.SetAnimationPhase( "Cord_plugged", 1);
```

```
// ---------------------------------------------
// ------ XmasSleighLight.c - START --------------------
// ---------------------------------------------


class XmasSleighLight extends PointLightBase
{
■void XmasSleighLight()
■{
■■SetVisibleDuringDaylight(false);
■■SetRadiusTo(24);
■■SetBrightnessTo(3);
■■SetCastShadow(false);
■■SetDiffuseColor(0.7, 0.7, 1.0);
■■SetFlareVisible(false);
■■SetFlickerAmplitude(0.5);
■■SetFlickerSpeed(1);
■}
}


// ---------------------------------------------
// ------ XmasSleighLight.c - END ----------------------
// ---------------------------------------------


// ---------------------------------------------
// ------ XmasTreeLight.c - START --------------------
// ---------------------------------------------


class XmasTreeLight extends PointLightBase
{
■void XmasTreeLight()
■{
■■SetVisibleDuringDaylight(false);
■■SetRadiusTo( 12 );
■■SetBrightnessTo( 0.50 );
■■SetCastShadow(false);
■■SetDiffuseColor(1.2, 1.0, 0.7);
■■SetDancingShadowsMovementSpeed(0.005);
■■SetDancingShadowsAmplitude(0.003);
■}
}


// ---------------------------------------------
// ------ XmasTreeLight.c - END ----------------------
// ---------------------------------------------


// ---------------------------------------------
// ------ ZmijovkaCap_ColorBase.c - START --------------------
// ---------------------------------------------


class ZmijovkaCap_ColorBase extends Clothing
{
■override void SetActions()
■{
■■super.SetActions();
■■AddAction(ActionWringClothes);
■}
```

```
// ---------------------------------------------
// ------ ZombieBase.c - START --------------------
// ---------------------------------------------


class ZombieBase extends DayZInfected
{
■const float TARGET_CONE_ANGLE_CHASE = 20;
■const float TARGET_CONE_ANGLE_FIGHT = 30;
■const float ORIENTATION_SYNC_THRESHOLD = 30; //threshold for local heading/orientation sync
■
■const float■SHOCK_TO_STUN_MULTIPLIER = 2.82;

■//! server / singleplayer properties
■protected int m_StanceVariation = 0;
■protected int m_LastMindState = -1;
■protected float m_LastMovementSpeed = -1;
■
■protected bool m_KnuckleLand = false;
■protected float m_KnuckleOutTimer = 0;

■protected int m_MindState = -1;
■protected int m_OrientationLocal = -1; //local 'companion' value for sync checking
■protected int m_OrientationSynced = -1;
■protected float m_OrientationTimer;
■protected float m_MovementSpeed = -1;
■
■protected vector m_DefaultHitPosition;
■
■protected AbstractWave m_LastSoundVoiceAW;
■protected ref InfectedSoundEventHandler■m_InfectedSoundEventHandler;

■protected ref array<Object> m_AllTargetObjects;
■protected ref array<typename>m_TargetableObjects;
■
■//static ref map<int,ref array<string>> ■m_FinisherSelectionMap; //! which selections in the FireGeometry
■
■protected bool m_IsCrawling; //'DayZInfectedCommandCrawl' is transition to crawl only, 'DayZInfectedCo
■
■protected bool m_FinisherInProgress = false; //is this object being backstabbed?



■//-------------------------------------------------------------
■void ZombieBase()
■{
■■Init();
■}
■
■void Init()
■{
■■SetEventMask(EntityEvent.INIT);
■■
■■m_IsCrawling = false;
■■
■■RegisterNetSyncVariableInt("m_MindState", -1, 4);
■■RegisterNetSyncVariableInt("m_OrientationSynced", 0, 359);
■■RegisterNetSyncVariableFloat("m_MovementSpeed", -1, 3);
■■RegisterNetSyncVariableBool("m_IsCrawling");
■■
■■//! sets default hit position and cache it here (mainly for impact particles)
■■m_DefaultHitPosition = SetDefaultHitPosition(GetDayZInfectedType().GetDefaultHitPositionComponen
// ---------------------------------------------
// ------ ZombieContainer.c - START --------------------
// ---------------------------------------------


class ZombieContainer: CollapsibleContainer
{
■protected ref AttachmentsGroupContainer■■m_Container;
■protected ref ContainerWithCargo■■■m_CargoGrid;
■protected ref map<int, SlotsIcon>■■■m_InventorySlots;
■protected ref map<EntityAI, ref Container>■m_ShowedItems = new ref map<EntityAI, ref Container>;
■protected EntityAI■■■■■■■m_ZombieEntity;
■
■void ZombieContainer( LayoutHolder parent, int sort = -1 )
■{
■■m_InventorySlots = new ref map<int, SlotsIcon>;
■■m_Container = new AttachmentsGroupContainer(this);
■■
■■m_Container.SetHeader(GetHeader());
■■SetHeaderName();
■■SetHeader(null);
■■m_Body.Insert( m_Container );
■■
■■m_MainWidget = m_RootWidget.FindAnyWidget( "body" );
■■
■■WidgetEventHandler.GetInstance().RegisterOnChildAdd( m_MainWidget, this, "OnChildAdd" );
■■WidgetEventHandler.GetInstance().RegisterOnChildRemove( m_MainWidget, this, "OnChildRemove" )
■■
■■
■■RecomputeOpenedContainers();
■}
■
■void ~ZombieContainer()
■{
■■if( m_ZombieEntity )
■■{
■■■m_ZombieEntity.GetOnItemAttached().Remove(ItemAttached);
■■■m_ZombieEntity.GetOnItemDetached().Remove(ItemDetached);
■■}
■}
■
■void SetEntity( EntityAI zombie_entity )
■{
■■m_ZombieEntity = zombie_entity;
■■m_ZombieEntity.GetOnItemAttached().Insert(ItemAttached);
■■m_ZombieEntity.GetOnItemDetached().Insert(ItemDetached);
■■InitGhostSlots();
■■m_Parent.Refresh();
■}
■
■override void UpdateRadialIcon()
■{
■■if ( m_SlotIcon )
■■{
■■■bool show_radial_icon;
■■■show_radial_icon = IsHidden();
■■■Widget rip = m_SlotIcon.GetRadialIconPanel();
■■■Widget icon_open = m_SlotIcon.GetRadialIcon();
■■■Widget icon_closed = m_SlotIcon.GetRadialIconClosed();
■■■rip.Show( !m_ZombieEntity.GetInventory().IsInventoryLockedForLockType( HIDE_INV_FROM_SCRI
■■■icon_open.Show( show_radial_icon );
■■■icon_closed.Show( !show_radial_icon );
■■}
```

```
// ---------------------------------------------
// ------ ZombieFemaleBase.c - START --------------------
// ---------------------------------------------


class ZombieFemaleBase extends ZombieBase
{
■override bool IsMale()
■{
■■return false;
■}
■
■override string CaptureSound()
■{
■■return "ZmbF_Normal_HeavyHit_Soundset";
■}
■
■override string ReleaseSound()
■{
■■return "ZmbF_Normal_CallToArmsShort_Soundset";
■}
};

class ZmbF_BlueCollarFat_Base extends ZombieFemaleBase
{
};

class ZmbF_CitizenANormal_Base extends ZombieFemaleBase
{
};

class ZmbF_CitizenANormal_LT_Base extends ZombieFemaleBase
{
};

class ZmbF_CitizenBSkinny_Base extends ZombieFemaleBase
{
};

class ZmbF_Clerk_Normal_Base extends ZombieFemaleBase
{
};

class ZmbF_ClerkFat_Base extends ZombieFemaleBase
{
};

class ZmbF_Clerk_Normal_LT_Base extends ZombieFemaleBase
{
};

class ZmbF_DoctorSkinny_Base extends ZombieFemaleBase
{
};

class ZmbF_HikerSkinny_Base extends ZombieFemaleBase
{
};

class ZmbF_JoggerSkinny_Base extends ZombieFemaleBase
{
};
```

```
// ---------------------------------------------
// ------ ZombieMaleBase.c - START --------------------
// ---------------------------------------------


class ZombieMaleBase extends ZombieBase
{
■override string CaptureSound()
■{
■■return "ZmbM_Normal_HeavyHit_Soundset";
■}
■
■override string ReleaseSound()
■{
■■return "ZmbM_Normal_CallToArmsShort_Soundset";
■}
■
};

//! Base class for eatch Infected type
class ZmbM_CitizenASkinny_Base extends ZombieMaleBase
{
};

class ZmbM_CitizenASkinny_LT_Base extends ZombieMaleBase
{
};

class ZmbM_CitizenBFat_Base extends ZombieMaleBase
{
};

class ZmbM_ClerkFat_Base extends ZombieMaleBase
{
};

class ZmbM_ClerkFat_LT_Base extends ZombieMaleBase
{
};

class ZmbM_CommercialPilotOld_Base extends ZombieMaleBase
{
};

class ZmbM_CommercialPilotOld_LT_Base extends ZombieMaleBase
{
};

class ZmbM_ConstrWorkerNormal_Base extends ZombieMaleBase
{
};

class ZmbM_DoctorFat_Base extends ZombieMaleBase
{
};

class ZmbM_FarmerFat_Base extends ZombieMaleBase
{
};

class ZmbM_FarmerFat_LT_Base extends ZombieMaleBase
{
};
```

```
// ---------------------------------------------
// ------ ZombieMummyLight.c - START --------------------
// ---------------------------------------------


class ZombieMummyLight extends PointLightBase
{
■void ZombieMummyLight()
■{
■■SetVisibleDuringDaylight(false);
■■SetRadiusTo(2.1);
■■SetBrightnessTo(3.1);
■■SetCastShadow(false);
■■SetFadeOutTime(15);
■■SetDiffuseColor(0.6, 0.5, 0.1);
■■SetAmbientColor(0.6, 0.5, 0.1);
■■SetFlareVisible(false);
■■SetFlickerAmplitude(0.35);
■■SetFlickerSpeed(0.25);
■■SetDancingShadowsMovementSpeed(0.05);
■■SetDancingShadowsAmplitude(0.03);
■}
};


// ---------------------------------------------
// ------ ZombieMummyLight.c - END ----------------------
// ---------------------------------------------


// ---------------------------------------------
// ------ ZSh3PilotHelmet.c - START --------------------
// ---------------------------------------------


class ZSh3PilotHelmet extends HelmetBase
{
}


// ---------------------------------------------
// ------ ZSh3PilotHelmet.c - END ----------------------
// ---------------------------------------------


// ---------------------------------------------
// ------ Zucchini.c - START --------------------
// ---------------------------------------------


class Zucchini : Edible_Base
{
■override bool CanBeCookedOnStick()
■{
■■return true;
■}

■override bool CanBeCooked()
■{
■■return true;
■}■

■override bool IsFruit()
```

```
// ---------------------------------------------
// ------ ZucchiniSeedsPack.c - START --------------------
// ---------------------------------------------


class ZucchiniSeedsPack extends SeedPackBase
{■
}


// ---------------------------------------------
// ------ ZucchiniSeedsPack.c - END ----------------------
// ---------------------------------------------


// ---------------------------------------------
// ------ _constants.c - START --------------------
// ---------------------------------------------


enum eBadgeLevel
{
■NONE,
■FIRST,
■SECOND,
■THIRD,
}

enum eDisplayElements
{
■DELM_BADGE_STUFFED,
■DELM_BADGE_WET,
■DELM_BADGE_SICK,
■DELM_BADGE_POISONED,
■DELM_BADGE_FRACTURE,
■DELM_BADGE_BLEEDING,
■DELM_BADGE_PILLS,
■DELM_BADGE_HEARTBEAT,
■//------------------
■DELM_TDCY_HEALTH,
■DELM_TDCY_BLOOD,
■DELM_TDCY_TEMPERATURE,
■DELM_TDCY_ENERGY,
■DELM_TDCY_WATER,
■DELM_TDCY_BACTERIA,
■//------------------
■DELM_STANCE,
■COUNT
}

//constants related to ui
//key constants
const int■NTFKEY_THIRSTY■= 1;
const int■NTFKEY_HUNGRY■= 2;
const int■NTFKEY_WARMTH■= 3;
const int■NTFKEY_WETNESS■= 4;
const int■NTFKEY_FRACTURE■= 5;
const int■NTFKEY_HEALTHY■= 6;
const int■NTFKEY_FEVERISH■= 7;
const int■NTFKEY_SICK■■= 8;
const int■NTFKEY_STUFFED■= 9;
const int■NTFKEY_BLEEDISH■= 10;
const int■NTFKEY_BLOOD■= 11;
```