Ultimate Starter Kit

Documentation

Henry Jooste

None

Table of contents

1. Getting Started

1.1 Requirements

The Ultimate Starter Kit plugin is only available for Unreal Engine 4.27 and newer. The plugin also depends on the following plugins:

1. Niagara

2. Enchanced Input

1.2 Installation

1. Download the latest release from GitHub

2. Navigate to `C:\Program Files\Epic Games\UE_{VERSION}\Engine\Plugins`

3. Create a `Marketplace` folder if needed

4. Extract the release and copy to the `Marketplace` folder

5. Open Unreal Engine

6. Click on `Edit > Plugins`

7. Enable the plugin under the `Built-in > Other` category

8. Restart Unreal Engine

1.3 Plugin Content

The Ultimate Starter Kit plugin includes content that can be used in your Blueprints. You might need to enable this first:

1. Open the `Content Browser`

2. Click on the settings button

3. Enable the `Show Plugin Content` setting

2. Special Thanks

This project uses a few assets from third parties:

• Generic Graph by jinyuliao

• Comic Helvetic Font by Alexander Pravdin

• Menu Sound Effects by Broumbroum

• Land & Jump Sound Effects by Felixyadomi

• Item Sound Effects by Scrampunk

• Slide Sound Effects by Aquafeniz

• Music by InAudio

If you liked these assets and want to see more, please follow the links and show them some love!

3. Console Codenames

3.1 Code names

Due to copyright issues, the plugin is not allowed to mention Xbox, Playstation or Switch anywhere in the code (including file names). To get around this issue, the consoles are refered to by the following code names:

Console	Code name
Xbox	Console MX
Playstation	Console SP
Switch	Console NS

3.2 Obfuscated code

The plugin will also use the console names to determine the platform. To avoid any copyright issues, the application will instead use an obfuscated string to determine the platform. The obfuscation and deobfuscation code can be found in `PlatformUtils.cpp`

Console	Obfuscated Text	Deobfuscated Text
Xbox	PT1BZWk5R2U=	xbox
Playstation	PUEzYw== PT13YzBGR2RwOW1i	ps station
Switch	PTRXYXVSWFp1UjJi emRYYTBOR2E=	nintendoswitch

4. Core Functionality

- 8/176 -

4.1 Input Devices

4.1.1 Introduction

Supported input devices. This is used to update the input indicators when using different input devices

4.1.2 Values

Value	Description
KeyboardMouse	Using a keyboard and mouse
GenericController	Using a controller on a desktop build
MxController	Using an Console MX controller
SpController	Using a Console SP controller
NsController	Using a Console NS controller
Unknown	Unknown device (used before initializing the input indicators)

4.2 Game Instance

4.2.1 Introduction

A base game instance with support for saving and loading game data using multiple save slots

4.2.2 Dependencies

The `USKGameInstance` relies on other components of this plugin to work:

- Logger: Used to log useful information to help you debug any issues you might experience

4.2.3 Using the Game Instance

You need to create a blueprint using the `USKGameInstance_Implementation` as a parent before using the game instance. The input indicators feature is already configured if you use this base class. If you prefer to set this up manually, you can use `USKGameInstance` instead. After creating your own game instance blueprint, set this as the default game instance:

1. Open the Project Settings
2. Go to Project > Maps & Modes
3. Change the `Game Instance Class` value to your own blueprint

4.2.4 Save Data

You need to create a `USK Save Game` object before you can save/load data. This object contains all the data that you want to save. Just add the data you want to save as variables to the blueprint. The `Game Instance` will handle the rest. You also need to set the following properties before you can save/load data:

- **Save Game Class:** A reference to the `USK Save Game` class that contains the data you want to save

NB: You are required to set the save slot before you can save/load data. If not, you will get a `nullptr` and might cause your game to crash

4.2.5 Input Indicators

The Game Instance will automatically detect input events and update the current input device if needed. If the input device is changed, other classes will be notified through the `OnInputDeviceUpdated` event

4.2.6 API Reference

Properties

Property	Description	Type	Default Value
LogConfigEditor	The log configuration used when running the game in the editor	ULogConfig*	`nullptr`
LogConfigDebug	The log configuration used by debug builds	ULogConfig*	`nullptr`
LogConfigDevelopment	The log configuration used by development builds	ULogConfig*	`nullptr`
LogConfigShipping	The log configuration used by shipping builds	ULogConfig*	`nullptr`
SaveGameClass	The class that holds the data that should be saved/loaded	TSubclassOf<UUSKSaveGame>	
SettingsConfig	The configuration for the settings	USettingsConfig*	`nullptr`
IsInputIndicatorsEnabled	Is the input indicators feature enabled?	bool	true
InputMappingContext	The input mapping context used to extract the keys based on specific input actions	UInputMappingContext*	`nullptr`
KeyboardMouseInputMappings	A map of all keyboard/mouse keys and the texture displayed in the indicator	TMap<FKey, UTexture2D*>	
GenericControllerInputMappings	A map of all generic controller keys and the texture displayed in the indicator	TMap<FKey, UTexture2D*>	
MxControllerInputMappings	A map of all Console MX controller keys and the texture displayed in the indicator	TMap<FKey, UTexture2D*>	
SpControllerInputMappings	A map of all Console SP controller keys and the texture displayed in the indicator	TMap<FKey, UTexture2D*>	
NsControllerInputMappings	A map of all Console NS controller keys and the texture displayed in the indicator	TMap<FKey, UTexture2D*>	
MessagePopupWidgetClass	The widget class used to display message popups	TSubclassOf<UMessagePopupWidget>	

Events

Name	Description	Params
OnDataLoadedEvent	Event used to notify other classes when the save data is loaded	
OnInputDeviceUpdated	Event used to notify other classes when the current input device is updated	
OnGamePaused	Event used to notify other classes when the game is paused	
OnGameUnpaused	Event used to notify other classes when the game is unpaused	
OnDifficultyUpdated	Event used to notify other classes when the difficulty is updated	**Difficulty (int)** The new difficulty value

Functions

Name	Description	Params	Return
GetSaveData	Get the save data that is currently loaded		**UUSKSaveGame*** A reference to the current save data
SaveData	Save the modified data currently in memory		
SetCurrentSaveSlot	Set the current save slot	**Index (int)** The index of the save slot	
IsSaveSlotUsed	Check if a save slot is used	**Index (int)** The index of the save slot to check	**bool** A boolean value indicating if the save slot is used
EnableInputIndicators	Enable the input indicators feature		
DisableInputIndicators	Disable the input indicators feature		
GetInputIndicatorIcon	Get the input indicator icon for a specific action	**InputAction (UInputAction*)** The input action **Amount (int)** The amount of icons to retrieve	**TArray<UTexture2D*>** An array of input indicator icons for the specified action
GetInputIndicatorIconForKey	Get the input indicator icon for a specific key	**Key (FKey)** The key used to retrieve the input indicator icon **InputDevice (EInputDevice)** The input device used to retrieve the input indicator icon	**UTexture2D*** The input indicator icon for the specified key
GetKeyForInputAction	Get the key used by a specific input action	**Context (UInputMappingContext*)** The input mapping context **InputAction (UInputAction*)** The input action **MappableName (FName)** The player mappable name for the action	**FKey** The key used by the specified input action
UpdateKeyBindings	Update the key bindings that was changed by the player		
PauseGame	Pause the game		
UnpauseGame	Unpause the game		
UpdateDifficulty	Update the difficulty	**Difficulty (int)** The new difficulty	
GetDifficulty	Get the current difficulty		**int** The current difficulty
ShowMessagePopup	Show a message popup	**Data (FMessagePopupData)** The data displayed in the message popup	**UMessagePopupWidget*** A reference to the message popup
HideMessagePopup	Hide the message popup		

4.2.7 Blueprint Usage

You can use the `USKGameInstance` using Blueprints by adding one of the following nodes:

- Ultimate Starter Kit > Save Data > Get Save Data

- Ultimate Starter Kit > Save Data > Save Data

- Ultimate Starter Kit > Save Data > Set Current Save Slot

- Ultimate Starter Kit > Save Data > Is Save Slot Used

- Ultimate Starter Kit > Input > Enable Input Indicators

- Ultimate Starter Kit > Input > Disable Input Indicators

- Ultimate Starter Kit > Input > Get Input Indicator Icon

- Ultimate Starter Kit > Input > Get Input Indicator Icon For Key

- Ultimate Starter Kit > Input > Get Key For Input Action

- Ultimate Starter Kit > Input > Update Key Bindings

- Ultimate Starter Kit > Pause > Pause Game

- Ultimate Starter Kit > Pause > Unpause Game

- Ultimate Starter Kit > Difficulty > Update Difficulty

- Ultimate Starter Kit > Difficulty > Get Difficulty

- Ultimate Starter Kit > UI > Show Message Popup

- Ultimate Starter Kit > UI > Hide Message Popup

4.2.8 C++ Usage

Before you can use the plugin, you first need to enable the plugin in your `Build.cs` file:

```
PublicDependencyModuleNames.Add("USK");
```

The `USKGameInstance` can now be used in any of your C++ files:

```
#include "USK/Core/USKGameInstance.h"

void ATestActor::Test()
{
    // USKGameInstance is a pointer to the UUSKGameInstance
    UUSKSaveGame* SaveData = USKGameInstance->GetSaveData();
    USKGameInstance->SaveData();
    USKGameInstance->SetCurrentSaveSlot(Index);
    bool IsSaveSlotUsedValue = USKGameInstance->IsSaveSlotUsed(Index);
    USKGameInstance->EnableInputIndicators();
    USKGameInstance->DisableInputIndicators();
    TArray<UTexture2D*> InputIndicatorIcon = USKGameInstance->GetInputIndicatorIcon(InputAction, Amount);
    UTexture2D* InputIndicatorIconForKey = USKGameInstance->GetInputIndicatorIconForKey(Key, InputDevice);
    FKey KeyForInputAction = USKGameInstance->GetKeyForInputAction(Context, InputAction, MappableName);
    USKGameInstance->UpdateKeyBindings();
    USKGameInstance->PauseGame();
    USKGameInstance->UnpauseGame();
    USKGameInstance->UpdateDifficulty(Difficulty);
    int Difficulty = USKGameInstance->GetDifficulty();
    UMessagePopupWidget* ShowMessagePopupValue = USKGameInstance->ShowMessagePopup(Data);
    USKGameInstance->HideMessagePopup();
}
```

5. Logging System

5.1 Logger

5.1.1 Introduction

A system used to easily log info to file and via on-screen messages

5.1.2 Log Levels

This plugin supports the following log types:

1. **Trace:** Logs that contain the most detailed messages. These messages may contain sensitive application data. These messages are disabled by default and should never be enabled in a production environment

2. **Debug:** Logs that are used for interactive investigation during development. These logs should primarily contain information useful for debugging and have no long-term value

3. **Information:** Logs that track the general flow of the application. These logs should have long-term value

4. **Warning:** Logs that highlight an abnormal or unexpected event in the application flow, but do not otherwise cause the application execution to stop

5. **Error:** Logs that highlight when the current flow of execution is stopped due to a failure. These should indicate a failure in the current activity, not an application-wide failure

The log levels corresponds to the following verbosity level in Unreal Engine:

Log Level	Log Verbosity
Trace	VeryVerbose
Debug	Verbose
Information	Display
Warning	Warning
Error	Error

5.1.3 API Reference

Functions

Name	Description	Params	Return
Configure	Configure the logger	**Config (ULogConfig*)** The new config file used by the logger	
Error	Log an error	**Tag (FString)** The category of the log entry **Text (FString)** The text to log out	
Warning	Log a warning	**Tag (FString)** The category of the log entry **Text (FString)** The text to log out	
Info	Log info	**Tag (FString)** The category of the log entry **Text (FString)** The text to log out	
Debug	Log debug information	**Tag (FString)** The category of the log entry **Text (FString)** The text to log out	
Trace	Log trace information	**Tag (FString)** The category of the log entry **Text (FString)** The text to log out	

5.1.4 Blueprint Usage

You can easily log info using Blueprints by adding one of the following nodes:

- Ultimate Starter Kit > Logger > Configure
- Ultimate Starter Kit > Logger > Log Trace
- Ultimate Starter Kit > Logger > Log Debug
- Ultimate Starter Kit > Logger > Log Info
- Ultimate Starter Kit > Logger > Log Warning
- Ultimate Starter Kit > Logger > Log Error

5.1.5 C++ Usage

The logging is handled through a static class/functions. You first need to enable the plugin in your `Build.cs` file:

```
PublicDependencyModuleNames.Add("USK");
```

The logger can now be used in any of your C++ files:

```
#include "USK/Logger/Log.h"
```

```cpp
void ATestActor::Test()
{
    USK_LOG_TRACE("Testing trace logging");
    USK_LOG_DEBUG("Testing debug logging");
    USK_LOG_INFO("Testing info logging");
    USK_LOG_WARNING("Testing warning logging");
    USK_LOG_ERROR("Testing error logging");

    ULog::Configure(Config);
    ULog::Trace("Custom Tag", "Testing trace logging");
    ULog::Debug("Custom Tag", "Testing debug logging");
    ULog::Info("Custom Tag", "Testing info logging");
    ULog::Warning("Custom Tag", "Testing warning logging");
    ULog::Error("Custom Tag", "Testing error logging");
}
```

```cpp
void ATestActor::Test()
{
    USK_LOG_TRACE("Testing trace logging");
    USK_LOG_DEBUG("Testing debug logging");
    USK_LOG_INFO("Testing info logging");
    USK_LOG_WARNING("Testing warning logging");
    USK_LOG_ERROR("Testing error logging");
```

```cpp
    ULog::Configure(Config);
```

5.2 Log Configuration

5.2.1 Introduction

Config file for the logger

5.2.2 API Reference

Properties

Property	Description	Type	Default Value
bErrorWriteToFile	Should error logs be written to file?	bool	true
bErrorPrintToScreen	Should error logs be printed to screen?	bool	true
bWarningWriteToFile	Should warning logs be written to file?	bool	true
bWarningPrintToScreen	Should warning logs be printed to screen?	bool	true
bInfoWriteToFile	Should info logs be written to file?	bool	true
bInfoPrintToScreen	Should info logs be printed to screen?	bool	true
bDebugWriteToFile	Should debug logs be written to file?	bool	true
bDebugPrintToScreen	Should debug logs be printed to screen?	bool	true
bTraceWriteToFile	Should debug logs be written to file?	bool	true
bTracePrintToScreen	Should debug logs be printed to screen?	bool	true

6. Dialogue System

6.1 Dialogue System

6.1.1 Introduction

The Ultimate Starter Kit plugin contains a full dialogue system with support for different participants, choices and brances

6.1.2 Creating a new dialogue

You can create a new dialogue by following these steps: 1. Right click in the Content Browser 2. Go to the Ultimate Starter Kit section 3. Click on Dialogue

This will open the newly created dialogue in the dialogue editor

6.1.3 Using the editor

The dialogue editor works similar to a Behaviour Tree. You start by creating a single root node. You can then drag connections to a new dialogue entry. The game will start off with the root and follow the path automatically until it reaches the end or a choice. If a choice is encountered, it will present the options to the player and follow the correct branch after a choice is selected

6.2 Dialogue

6.2.1 Introduction

An array of all the root entries contained by this dialogue

6.2.2 API Reference

Properties

Property	Description	Type	Default Value
RootEntries	An array of all the root entries contained by this dialogue	TArray<UDialogueEntry*>	
AllEntries	An array of all the entries contained by this dialogue	TArray<UDialogueEntry*>	
EditorGraph	A reference to the editor graph used by this dialogue	UEdGraph*	nullptr

6.3 Participant

6.3.1 Introduction

A participant in the dialogue

6.3.2 API Reference

Properties

Property	Description	Type	Default Value
Name	The name of the dialogue participant	FText	
PortraitImage	The portrait image for the dialogue participant	UTexture2D*	`nullptr`
Color	The color used to represent the dialogue participant	FLinearColor	FLinearColor::Black
bCustomPortraitImageBorderColor	Should a custom border color be used for the portrait image instead of the participant color?	bool	false
PortraitImageBorderColor	The custom portrait image border color	FSlateColor	FSlateColo

6.4 Entry

6.4.1 Introduction

A single entry in a dialogue

6.4.2 API Reference

Properties

Property	Description	Type	Default Value
Owner	The owner participant of the dialogue entry	UDialogueParticipant*	`nullptr`
Transition	The type of transition for this dialogue entry	EDialogueTransitionType	
Id	The ID of the dialogue entry	FName	FNam
Text	The text to display	FText	
Speed	The speed of the dialogue	float	12.5f
Audio	The audio to play with this dialogue entry	USoundBase*	`nullptr`
bOverrideRichTextStyle	Should the rich text style be overridden?	bool	false
RichTextStyle	The rich text style applied to the widget	UDataTable*	`nullptr`
bOverridePortraitImage	Should the participant portrait image be overridden?	bool	false
CustomPortraitImage	The custom portrait image for this dialogue entry	UTexture2D*	`nullptr`
CustomPortraitBorderColor	The custom portrait border color for this dialogue entry	FSlateColor	FSlateColo
Dialogue	A reference to the dialogue containing this entry	UDialogue*	`nullptr`
ParentNodes	An array of all the parent entries for this dialogue entry	TArray<UDialogueEntry*>	
ChildrenNodes	An array of all the child entries for this dialogue entry	TArray<UDialogueEntry*>	
Edges	A map of all the possible edges for this dialogue entry	TMap<UDialogueEntry*, UDialogueTransition*>	

Functions

Name	Description	Params	Return
IsLeafNode	Check if this entry is a leaf node		**bool** A boolean value indicating if this entry is a leaf node
GetTitle	Get the title displayed for this dialogue entry		**FText** The title displayed for this dialogue entry
GetText	Get the text displayed for this dialogue entry		**FText** The text displayed for this dialogue entry

6.4.3 Blueprint Usage

You can use the `DialogueEntry` using Blueprints by adding one of the following nodes:

- • Ultimate Starter Kit > Dialogue > Entry > Is Leaf Node

- • Ultimate Starter Kit > Dialogue > Entry > Get Title

- • Ultimate Starter Kit > Dialogue > Entry > Get Text

6.4.4 C++ Usage

Before you can use the plugin, you first need to enable the plugin in your `Build.cs` file:

```
PublicDependencyModuleNames.Add("USK");
```

The `DialogueEntry` can now be used in any of your C++ files:

```
#include "USK/Dialogue/DialogueEntry.h"

void ATestActor::Test()
{
    // DialogueEntry is a pointer to the UDialogueEntry
    bool IsLeafNodeValue = DialogueEntry->IsLeafNode();
    FText Title = DialogueEntry->GetTitle();
    FText Text = DialogueEntry->GetText();
}
```

6.5 Transition

6.5.1 Transition Type

Introduction

The type of transition used by the dialogue entry

Values

Value	Description
Auto	Automatically advance to the next dialogue entry
Choice	Give the player a few choices used to create branches in the dialogue

6.5.2 Transition Data

Introduction

A transition from one dialogue entry to the next

API Reference

PROPERTIES

Property	Description	Type	Default Value
Text	The text displayed for this transition (choice)	FText	
StartEntry	The dialogue entry where the transition starts	UDialogueEntry*	`nullptr`
EndEntry	The dialogue entry where the transition ends	UDialogueEntry*	`nullptr`

EVENTS

Name	Description	Params
OnMarkedForDeletion	Event used to notify other classes that the transition is marked for deletion	

6.6 Manager

6.6.1 Introduction

The pawn responsible for managing the dialogue

6.6.2 Dependencies

The `DialogueManager` relies on other components of this plugin to work:

- Logger: Used to log useful information to help you debug any issues you might experience

- Audio: Used to play sound effects either 2D or at a specified location

6.6.3 Components

The `DialogueManager` uses the following components:

Name	Description	Type
AudioComponent	The audio component responsible for playing the audio files of the dialogue entries	UAudioComponent*

6.6.4 API Reference

Properties

Property	Description	Type	Default Value
Dialogue	The dialogue that should be played by the dialogue manager	UDialogue*	nullptr
DialogueWidgetClass	The class of the widget used to display the dialogue	TSubclassOf<UDialogueWidget>	
PlayOnStart	A boolean value indicating if the dialogue should automatically play when the level is started	bool	true
DestroyOnComplete	A boolean value indicating if the dialogue should automatically be destroyed when completed	bool	true
StopOnComplete	A boolean value indicating if the dialogue should automatically be stopped when completed	bool	true
SkipSFX	The sound effect to play when an entry is skipped	USoundBase*	nullptr
AdvanceSFX	The sound effect to play when advancing to the next entry	USoundBase*	nullptr
InputMappingContext	The input mapping context used to interact with the dialogue	UInputMappingContext*	nullptr
SkipAction	The input action used to skip the current dialogue entry	UInputAction*	nullptr

Events

Name	Description	Params
OnDialogueEnded	Event used to notify other classes when the dialogue has ended	**LastEntryId (FName)** The ID of the last entry in the dialogue
OnDialogueEntryStarted	Event used to notify other classes when a dialogue entry has started	**LastEntryId (FName)** The ID of the dialogue entry
OnDialogueEntryEnded	Event used to notify other classes when a dialogue entry has ended	**LastEntryId (FName)** The ID of the dialogue entry

Functions

Name	Description	Params	Return
PlayDialogue	Play the dialogue		
StopDialogue	Stop playing the dialogue		
DestroyDialogue	Stop playing the dialogue and destroy the dialogue manager		
SkipEntry	Skip the current entry in the dialogue		
GetDialogueWidget	Get the dialogue widget		**UDialogueWidget*** A reference to the dialogue widget

6.6.5 Blueprint Usage

You can use the `DialogueManager` using Blueprints by adding one of the following nodes:

- Ultimate Starter Kit > Dialogue > Play Dialogue

- Ultimate Starter Kit > Dialogue > Stop Dialogue

- Ultimate Starter Kit > Dialogue > Destroy Dialogue

- Ultimate Starter Kit > Dialogue > Skip Entry

- Ultimate Starter Kit > Dialogue > Get Dialogue Widget

6.6.6 C++ Usage

Before you can use the plugin, you first need to enable the plugin in your `Build.cs` file:

```
PublicDependencyModuleNames.Add("USK");
```

The `DialogueManager` can now be used in any of your C++ files:

```
#include "USK/Dialogue/DialogueManager.h"

void ATestActor::Test()
{
    // DialogueManager is a pointer to the ADialogueManager
    DialogueManager->PlayDialogue();
    DialogueManager->StopDialogue();
    DialogueManager->DestroyDialogue();
    DialogueManager->SkipEntry();
    UDialogueWidget* DialogueWidget = DialogueManager->GetDialogueWidget();
}
```

6.7 Widget

6.7.1 Introduction

Widget used to display a dialogue

6.7.2 Dependencies

The `DialogueTransitionType` relies on other components of this plugin to work:

- Logger: Used to log useful information to help you debug any issues you might experience

6.7.3 Required Widgets

You need to add the following before you can compile the `DialogueTransitionType` widget:

Name	Description	Type
DialogueTitle	The text block used to display the dialogue title	UTextBlock*
DialogueText	The text block used to display the dialogue text	URichTextBlock*
ChoiceMenu	The menu used to display the dialogue choices	UMenu*

6.7.4 Optional Widgets

You can add the following widgets to enable extra functionality:

Name	Description	Type
ParticipantPortrait	The image used to display the portrait image of the participant	UImage*
ParticipantPortraitBorder	The border image displayed around the portrait image of the participant	UImage*
SkipEntryImage	The image displayed when the dialogue entry is completed	UImage*

6.7.5 API Reference

Properties

Property	Description	Type	Default Value
ChoiceMenuItemClass	The class for all choice menu items	TSubclassOf<UMenuItem>	
RichTextStyle	The rich text style applied to the dialogue text	UDataTable*	`nullptr`

Events

Name	Description	Params
OnChoiceSelected	Event used to notify other classes when a choice is selected	**Index (int)** The index of the choice that was selected

Functions

Name	Description	Params	Return
UpdateEntry	Update the current entry and refresh the UI	**Entry (UDialogueEntry*)** The new dialogue entry	
SkipEntry	Skip the current entry		**bool** A boolean value indicating if the entry was skipped

6.7.6 Blueprint Usage

You can use the `DialogueTransitionType` using Blueprints by adding one of the following nodes:

• Ultimate Starter Kit > UI > Update Entry

• Ultimate Starter Kit > UI > Skip Entry

6.7.7 C++ Usage

Before you can use the plugin, you first need to enable the plugin in your `Build.cs` file:

```
PublicDependencyModuleNames.Add("USK");
```

The `DialogueTransitionType` can now be used in any of your C++ files:

```
#include "USK/Dialogue/DialogueWidget.h"

void ATestActor::Test()
{
    // DialogueTransitionType is a pointer to the EDialogueTransitionType
    DialogueTransitionType->UpdateEntry(Entry);
    bool SkipEntryValue = DialogueTransitionType->SkipEntry();
}
```

7. Inventory

7.1 Inventory Component

7.1.1 Introduction

Actor component responsible for tracking data in an inventory

7.1.2 Dependencies

The `InventoryComponent` relies on other components of this plugin to work:

• Logger: Used to log useful information to help you debug any issues you might experience

• Game Instance: Used to monitor for input device changes and handle saving/loading game data

7.1.3 API Reference

Properties

Property	Description	Type	Default Value
InventoryId	The ID of the inventory used when saving/loading the data	FName	
AutoSave	Should the data in the inventory automatically be saved/loaded	bool	true
EnforceMaxAmount	Should a maximum amount be enforced for each item?	bool	false
MaxAmount	The maximum amount of each item	int	99

Events

Name	Description	Params
OnInventoryItemUpdated	Event used to notify other classes every time an item in the inventory was updated	**Id (FName)** The ID of the item that was updated **Amount (FName)** The new amount of the item

Functions

Name	Description	Params	Return
GetItems	Get all the item currently in the inventory		**TArray<FInventoryItem>** An array of all the items in the inventory
AddItem	Add an item to the inventory	**Id (FName)** The ID of the item to add **Amount (int)** The amount to add	
RemoveItem	Remove an item from the inventory	**Id (FName)** The ID of the item to remove **Amount (int)** The amount to remove	
RemoveAll	Remove all the items with the specified ID	**Id (FName)** The ID of the item to remove	
Clear	Remove all items from the inventory		
LoadInventory	Load the inventory data		
SaveInventory	Save the inventory data		

7.1.4 Blueprint Usage

You can use the `InventoryComponent` using Blueprints by adding one of the following nodes:

- Ultimate Starter Kit > Inventory > Get Items
- Ultimate Starter Kit > Inventory > Add Item
- Ultimate Starter Kit > Inventory > Remove Item
- Ultimate Starter Kit > Inventory > Remove All
- Ultimate Starter Kit > Inventory > Clear
- Ultimate Starter Kit > Inventory > Load Inventory
- Ultimate Starter Kit > Inventory > Save Inventory

7.1.5 C++ Usage

Before you can use the plugin, you first need to enable the plugin in your `Build.cs` file:

```
PublicDependencyModuleNames.Add("USK");
```

The `InventoryComponent` can now be used in any of your C++ files:

```
#include "USK/Inventory/InventoryComponent.h"

void ATestActor::Test()
{
    // InventoryComponent is a pointer to the UInventoryComponent
    TArray<FInventoryItem> Items = InventoryComponent->GetItems();
    InventoryComponent->AddItem(Id, Amount);
    InventoryComponent->RemoveItem(Id, Amount);
    InventoryComponent->RemoveAll(Id);
    InventoryComponent->Clear();
    InventoryComponent->LoadInventory();
    InventoryComponent->SaveInventory();
}
```

7.2 Inventory Data

7.2.1 Introduction

The inventory data for a single inventory

7.2.2 Properties

Property	Description	Type	Default Value
Items	The array of items in the inventory	TArray<FInventoryItem>	

7.2 Inventory Data

7.3 Inventory Item

7.3.1 Introduction

The information about a specific inventory item

7.3.2 Properties

Property	Description	Type	Default Value
Id	The ID of the inventory item	FName	
Amount	The amount of the inventory item	int	

7.3 Inventory Item

7.4 Inventory Item Data

7.4.1 Introduction

The data used to describe an inventory item

7.4.2 Properties

Property	Description	Type	Default Value
Name	The name of the inventory item	FText	
Description	The description of the inventory item	FText	
InventoryImage	The image displayed in the inventory	UTexture2D*	`nullptr`
PreviewImage	The preview image displayed when the item is highlighted	UTexture2D*	`nullptr`

7.5 Inventory Menu Item

7.5.1 Introduction

The menu item used to display an inventory item

7.5.2 Dependencies

The `InventoryMenuItem` relies on other components of this plugin to work:

- Logger: Used to log useful information to help you debug any issues you might experience

7.5.3 Optional Widgets

You can add the following widgets to enable extra functionality:

Name	Description	Type
InventoryImage	The image of the inventory item	UImage*

7.5.4 API Reference

Properties

Property	Description	Type	Default Value

Functions

Name	Description	Params	Return
InitializeEmptyInventoryItem	Initialize an empty inventory item	**Widget (UInventoryWidget*)** The widget that owns this menu item	
InitializeInventoryItem	Initialize a non-empty inventory item	**Widget (UInventoryWidget*)** The widget that owns this menu item **Item (FInventoryItem)** The item tracked by this menu item **Data (FInventoryItemData)** The data for this menu item	
UpdateInventoryGridPosition	Update the grid position of the inventory menu item	**CurrentColumn (int)** The current column of the inventory menu item **CurrentRow (int)** The current row of the inventory menu item	
GetInventoryItem	Get the inventory item tracked by this menu item		**FInventoryItem** The inventory item tracked by this menu item
UpdateAmount	Update the amount of the inventory item	**Amount (int)** The new amount of the inventory item	

7.5.5 Blueprint Usage

You can use the `InventoryMenuItem` using Blueprints by adding one of the following nodes:

• Ultimate Starter Kit > Inventory > Initialize Empty Inventory Item

• Ultimate Starter Kit > Inventory > Initialize Inventory Item

• Ultimate Starter Kit > Inventory > Update Inventory Grid Position

• Ultimate Starter Kit > Inventory > Get Inventory Item

• Ultimate Starter Kit > Inventory > Update Amount

7.5.6 C++ Usage

Before you can use the plugin, you first need to enable the plugin in your `Build.cs` file:

```
PublicDependencyModuleNames.Add("USK");
```

The `InventoryMenuItem` can now be used in any of your C++ files:

```
#include "USK/Inventory/InventoryMenuItem.h"

void ATestActor::Test()
{
    // InventoryMenuItem is a pointer to the UInventoryMenuItem
    InventoryMenuItem->InitializeEmptyInventoryItem(Widget);
    InventoryMenuItem->InitializeInventoryItem(Widget, Item, Data);
    InventoryMenuItem->UpdateInventoryGridPosition(CurrentColumn, CurrentRow);
    FInventoryItem InventoryItem = InventoryMenuItem->GetInventoryItem();
    InventoryMenuItem->UpdateAmount(Amount);
}
```

7.6 Inventory Size

7.6.1 Introduction

The size restrictions of the inventory

7.6.2 Values

Value	Description
FixedSize	Limit both the width and height of the inventory
FixedRows	Limit only the rows of the inventory and allow the columns to increase
FixedColumns	Limit only the columns of the inventory and allow the rows to increase

7.7 Inventory Widget

7.7.1 Introduction

The widget responsible for displaying the inventory

7.7.2 Dependencies

The `InventoryWidget` relies on other components of this plugin to work:

- Logger: Used to log useful information to help you debug any issues you might experience

7.7.3 Required Widgets

You need to add the following before you can compile the `InventoryWidget` widget:

Name	Description	Type
InventoryMenu	The menu responsible for controlling all the menu items	UMenu*

7.7.4 Optional Widgets

You can add the following widgets to enable extra functionality:

Name	Description	Type
NameText	The widget responsible for displaying the highlighted item's name	UTextBlock*
DescriptionText	The widget responsible for displaying the highlighted item's description	UTextBlock*
PreviewImage	The widget responsible for displaying the highlighted item's image	UImage*
AmountText	The widget responsible for displaying the highlighted item's amount	UTextBlock*

7.7.5 API Reference

Properties

Property	Description	Type	Default Value
MenuItemClass	The inventory menu item class used to display the inventory items	TSubclassOf<UInventoryMenuItem>	
ItemData	The data table containing all the information about the inventory items	UDataTable*	`nullptr`
InventorySize	The size restrictions of the inventory	EInventorySize	
Rows	The amount of rows in the inventory	int	
Columns	The amount of columns in the inventory	int	

Events

Name	Description	Params
OnInventoryItemSelected	Event used to notify other classes every time an inventory item is selected	**Name (FName)** The ID of the selected inventory item

Functions

Name	Description	Params	Return
LoadInventory	Load a specific inventory	**InventoryComponent (UInventoryComponent*)** The inventory to load	
UpdatePreview	Update the preview of the inventory	**Item (FInventoryItem)** The item to preview	
UpdateHighlightedIndex	Update the highlighted index	**Column (int)** The column index of the item that is highlighted **Row (int)** The row index of the item that is highlighted	
SelectItem	Select an inventory item	**Id (FName)** The ID of the item to select	
GetInventory	Get the inventory managed by the widget		**UInventoryComponent*** The inventory managed by the widget
RefreshItem	Refresh a specific item in the inventory	**Id (FName)** The ID of the item to refresh **Amount (int)** The amount of the item	
RefreshInventory	Refresh the entire inventory		

7.7.6 Blueprint Usage

You can use the `InventoryWidget` using Blueprints by adding one of the following nodes:

- Ultimate Starter Kit > Inventory > Load Inventory
- Ultimate Starter Kit > Inventory > Update Preview
- Ultimate Starter Kit > Inventory > Update Highlighted Index
- Ultimate Starter Kit > Inventory > Select Item
- Ultimate Starter Kit > Inventory > Get Inventory
- Ultimate Starter Kit > Inventory > Refresh Item
- Ultimate Starter Kit > Inventory > Refresh Inventory

7.7.7 C++ Usage

Before you can use the plugin, you first need to enable the plugin in your `Build.cs` file:

```
PublicDependencyModuleNames.Add("USK");
```

The `InventoryWidget` can now be used in any of your C++ files:

```
#include "USK/Inventory/InventoryWidget.h"

void ATestActor::Test()
{
    // InventoryWidget is a pointer to the UInventoryWidget
    InventoryWidget->LoadInventory(InventoryComponent);
    InventoryWidget->UpdatePreview(Item);
    InventoryWidget->UpdateHighlightedIndex(Column, Row);
```

```
    InventoryWidget->SelectItem(Id);
    UInventoryComponent* Inventory = InventoryWidget->GetInventory();
    InventoryWidget->RefreshItem(Id, Amount);
    InventoryWidget->RefreshInventory();
}
```

8. Trackable Data

8.1 Overview

8.1.1 Introduction

A system that is used to easily manage different types of actor data

8.1.2 Trackable Data Component

Before you can manage the data, you need to create a Trackable Data Component and add it to the actor/character containing the data

8.1.3 Built-in data

The following data can automatically be managed without creating custom components:

1. Currency (using the `Currency Component`)

2. Stats (using the `Stats Component`)

8.2 Data

8.2.1 Introduction

All trackable data use the FTrackableData struct to specify the default values and behaviours

8.2.2 Properties

Property	Description	Type	Default Value
InitialValue	The initial value of the data	float	
EnforceMaxValue	Should we enforce a maximum value?	bool	false
MaxValue	The maximum value of the data	float	100.0f
AutoSave	Should all value updates automatically be saved using the game instance?	bool	false
AutoGenerate	Should we automatically generate value every second?	bool	false
GenerateAmount	The amount of value to generate every second	float	
GenerateDelay	The delay before the value starts generating after losing value	float	

8.3 Component

8.3.1 Introduction

A component that is used to easily manage/track different types of actor data

8.3.2 Dependencies

The `TrackableDataComponent` relies on other components of this plugin to work:

- Logger: Used to log useful information to help you debug any issues you might experience

- Game Instance: Used to monitor for input device changes and handle saving/loading game data

8.3.3 Data

The data to track is configured by adding items to the `Data` map. The component should be added to the actor/character containing the data

8.3.4 API Reference

Properties

Property	Description	Type	Default Value
Data	The map of data to track	TMap<FName, FTrackableData>	

Events

Name	Description	Params
OnValueZero	Event used to notify other classes every time the data value reaches 0	**Name (FName)** The name of the data item
OnValueUpdated	Event used to notify other classes every time the data value is updated	**Name (FName)** The name of the data item **Value (FName)** The current value of the data item **ValuePercentage (FName)** The percentage of the current value compared to the max value of the data item

Functions

Name	Description	Params	Return
GetValue	Get the amount of the data	**Name (FName)** The name of the data item	**float** The current amount of the data item
GetValuePercentage	Get the value of the data as a percentage of to the max value	**Name (FName)** The name of the data item	**float** The value of the data as a percentage of to the max value
Add	Add an amount to the data	**Name (FName)** The name of the data item **Amount (float)** The amount to add	**float** The new amount of the data item
Remove	Remove an amount from the data	**Name (FName)** The name of the data item **Amount (float)** The amount to remove	**float** The new amount of the data item

8.3.5 Blueprint Usage

You can use the `TrackableDataComponent` using Blueprints by adding one of the following nodes:

- Ultimate Starter Kit > Trackable Data > Get Value

- Ultimate Starter Kit > Trackable Data > Get Value Percentage

- Ultimate Starter Kit > Trackable Data > Add

- Ultimate Starter Kit > Trackable Data > Remove

8.3.6 C++ Usage

Before you can use the plugin, you first need to enable the plugin in your `Build.cs` file:

```
PublicDependencyModuleNames.Add("USK");
```

The `TrackableDataComponent` can now be used in any of your C++ files:

```
#include "USK/Data/TrackableDataComponent.h"

void ATestActor::Test()
{
    // TrackableDataComponent is a pointer to the UTrackableDataComponent
    float Value = TrackableDataComponent->GetValue(Name);
    float ValuePercentage = TrackableDataComponent->GetValuePercentage(Name);
    float AddValue = TrackableDataComponent->Add(Name, Amount);
    float RemoveValue = TrackableDataComponent->Remove(Name, Amount);
}
```

9. Audio

9.1 Audio Overview

9.1.1 Introduction

A system used to manage the basic properties of audio files. It includes different sound classes, a sound mix and sound attenuation settings

9.1.2 Sound Classes

The audio system includes a few basic preconfigured sound classes:

Class name	Group	Volume
USK_EffectsSoundClass	Effects	1.0
USK_MusicSoundClass	Music	0.5
USK_UISoundClass	UI	1.0
USK_VoiceSoundClass	Voice	3.0

9.2 Audio Utils

9.2.1 Introduction

The audio utils class is used to easily play sound effects

9.2.2 Dependencies

The `AudioUtils` relies on other components of this plugin to work:

- Logger: Used to log useful information to help you debug any issues you might experience

9.2.3 API Reference

Functions

Name	Description	Params	Return
PlaySound2D	Play a 2D sound	**WorldContext (UObject*)** The top level object representing a map **SoundFX (USoundBase*)** The USoundBase to play	
PlayRandomSound2D	Play a random 2D sound	**WorldContext (UObject*)** The top level object representing a map **SoundFX (TArray)** The array of USoundBase to select the random sound from	
PlaySound	Play a sound at the specified actor's location	**Actor (AActor*)** The actor where the sound will be played **SoundFX (USoundBase*)** The USoundBase to play	
PlayRandomSound	Play a random sound at the specified actor's location	**Actor (AActor*)** The actor where the sound will be played **SoundFX (TArray)** The array of USoundBase to select the random sound from	

9.2.4 Blueprint Usage

You can use the `AudioUtils` using Blueprints by adding one of the following nodes:

- Ultimate Starter Kit > Audio > Play Sound2D
- Ultimate Starter Kit > Audio > Play Random Sound2D
- Ultimate Starter Kit > Audio > Play Sound
- Ultimate Starter Kit > Audio > Play Random Sound

9.2.5 C++ Usage

Before you can use the plugin, you first need to enable the plugin in your `Build.cs` file:

```
PublicDependencyModuleNames.Add("USK");
```

The `AudioUtils` can now be used in any of your C++ files:

```cpp
#include "USK/Audio/AudioUtils.h"

void ATestActor::Test()
{
    UAudioUtils::PlaySound2D(WorldContext, SoundFX);
    UAudioUtils::PlayRandomSound2D(WorldContext, SoundFX);
    UAudioUtils::PlaySound(Actor, SoundFX);
    UAudioUtils::PlayRandomSound(Actor, SoundFX);
}
```

9.3 Music Player

9.3.1 Introduction

Actor responsible for playing, pausing and stopping music. It also allows you to adjust music volume

9.3.2 Dependencies

The `inal` relies on other components of this plugin to work:

- Logger: Used to log useful information to help you debug any issues you might experience

9.3.3 Components

The `inal` uses the following components:

Name	Description	Type
AudioPlayer	Actor responsible for playing, pausing and stopping music. It also allows you to adjust music volume	UAudioComponent*

9.3.4 API Reference

Properties

Property	Description	Type	Default Value
PlayOnStart	Should the music automatically play when the actor is spawned?	bool	true

Functions

Name	Description	Params	Return
GetAudioComponent	Get a reference to the audio component		**UAudioComponent*** A reference to the audio component
SetVolume	Adjust the playback volume of the music	**NewVolume (float)** The new volume of the music	
Play	Play the music		
Pause	Pause the music		
Stop	Stop the music		
FadeOut	Fade out the music	**FadeDuration (float)** The duration of the fade	
FadeIn	Fade in the music	**FadeDuration (float)** The duration of the fade	

9.3.5 Blueprint Usage

You can use the `inal` using Blueprints by adding one of the following nodes:

- Ultimate Starter Kit > Audio > Get Audio Component

- Ultimate Starter Kit > Audio > Set Volume

- Ultimate Starter Kit > Audio > Play

- Ultimate Starter Kit > Audio > Pause

- Ultimate Starter Kit > Audio > Stop

- Ultimate Starter Kit > Audio > Fade Out

- Ultimate Starter Kit > Audio > Fade In

9.3.6 C++ Usage

Before you can use the plugin, you first need to enable the plugin in your `Build.cs` file:

```
PublicDependencyModuleNames.Add("USK");
```

The `inal` can now be used in any of your C++ files:

```
#include "USK/Audio/MusicPlayer.h"

void ATestActor::Test()
{
    // inal is a pointer to the final
    UAudioComponent* AudioComponent = inal->GetAudioComponent();
    inal->SetVolume(NewVolume);
    inal->Play();
    inal->Pause();
    inal->Stop();
    inal->FadeOut(FadeDuration);
    inal->FadeIn(FadeDuration);
}
```

10. Characters

10.1 Overview

10.1.1 Introduction

The plugin includes a basic 3D platformer character and animation template. This can easily be extended to add unique functionality

10.2 Base Character

10.2.1 Introduction

Base character class

10.2.2 Dependencies

The `USKCharacter` relies on other components of this plugin to work:

- Logger: Used to log useful information to help you debug any issues you might experience

- Audio: Used to play sound effects either 2D or at a specified location

10.2.3 Camera Clipping

Sometimes parts of the character mesh or weapon might clip when close to the camera (especially while aiming). This can be fixed by changing `Set Near Clip Plane` to `0.1` in your project settings

10.2.4 Components

The `USKCharacter` uses the following components:

Name	Description	Type
CameraComponent	The camera used by the character	UCameraComponent*
SpringArmComponent	The spring arm component responsible for controlling the distance of the camera	USpringArmComponent*
CrouchTimeline	The timeline component used for smooth crouching	UTimelineComponent*
ProneTimeline	The timeline component used for smooth proning	UTimelineComponent*
AimTimeline	The timeline component used for aiming	UTimelineComponent*
LookAtCenterTimeline	The timeline component used to look at the center of the screen	UTimelineComponent*

10.2.5 API Reference

Properties

Property	Description	Type	Default Value
InputMappingContext	The input mapping context used by the character	UInputMappingContext*	`nullptr`
MoveAction	The move input action	UInputAction*	`nullptr`
LookAroundAction	The camera rotation input action	UInputAction*	`nullptr`
JumpAction	The jump input action	UInputAction*	`nullptr`
SprintAction	The sprint input action	UInputAction*	`nullptr`
FireWeaponAction	The fire weapon input action	UInputAction*	`nullptr`
AimAction	The aim down sights input action	UInputAction*	`nullptr`
EquipNextWeaponAction	The equip next weapon input action	UInputAction*	`nullptr`
EquipPreviousWeaponAction	The equip previous weapon input action	UInputAction*	`nullptr`
ReloadWeaponAction	The reload weapon input action	UInputAction*	`nullptr`
CrouchAction	The crouch input action	UInputAction*	`nullptr`
ProneAction	The prone input action	UInputAction*	`nullptr`
LeanAction	The lean input action	UInputAction*	`nullptr`
InteractAction	The interact input action	UInputAction*	`nullptr`
SwitchCameraPerspectiveAction	The switch camera perspective input action	UInputAction*	`nullptr`
CameraPerspective	The camera perspective used by the character	ECameraPerspective	
bCanSwitchCameraPerspectives	Can the character switch perspectives?	bool	true
bSmoothCameraRotation	Should the camera rotation be smooth or instant?	bool	false
SmoothCameraRotationSpeed	The speed used to smooth the camera rotations	float	5.0f
HeadSocketName	The name of the head socket used to attach the camera in the first person perspective	FName	
CameraAttachOffset	The offset of the camera after attaching to the head	FVector	
TargetArmLength	Length of the spring arm component	float	350.0f
ArmLengthMultiplier	The multiplier applied to the spring arm component when the character is moving	float	0.4f
CameraAdjustmentSpeed	The speed used when adjusting the camera distance	float	3.0f
ShadowDecalClass	The shadow decal class used to draw a shadow below the character while in the air	TSubclassOf<AShadowDecal>	
JumpSoundEffects		TArray<USoundBase*>	

	An array of sound effects played when jumping		
JumpParticleFx	The particle effects spawned when jumping	UNiagaraSystem*	`nullptr`
JumpParticleFxSpawnOffset	The offset applied to the location of the jump particles when spawning	FVector	
LandedSoundEffects	An array of sound effects played when landing	TArray<USoundBase*>	
LandParticleFx	The particle effects spawned when landing	UNiagaraSystem*	`nullptr`
LandParticleFxSpawnOffset	The offset applied to the location of the land particles when spawning	FVector	
VariableJumpHeight	Does the character support variable jump height?	bool	true
VariableJumpHeightMaxHoldTime	The amount of time to hold the jump button to reach the max height	float	0.3f
JumpVelocity	The velocity applied to the character when jumping	float	500.0f
AirControl	The amount of lateral movement control available to the character while in the air	float	1000.0f
FallingFriction	The amount of friction to apply to lateral air movement when falling	float	3.5f
Gravity	The amount of gravity applied to the character	float	2.0f
CanDoubleJump	Can the character perform a double jump?	bool	true
CanCoyoteJump	Does the character support coyote time when trying to jump?	bool	true
CoyoteJumpTime	The amount of coyote time for the character	float	0.375f
CoyoteJumpVelocity	The velocity applied to the character when performing a coyote jump	float	700.0f
bCanStompJump	Can the character perform a high jump directly after landing from a stomp?	bool	true
StompJumpDuration	The duration after the stomp that the character can perform the stomp jump	float	0.35f
StompJumpVelocity	The velocity applied to the character when performing a stomp jump	float	1250.0f
bCanCrouchJump	Can the character perform a high jump while crouching?	bool	true
CrouchJumpVelocity	The velocity applied to the character when performing a crouch jump	float	1250.0f
IsDoubleJumping	Is the character double jumping?	bool	false
bCanLongJump		bool	true

	Is the character allowed to perform a long jump?		
LongJumpVelocity	The velocity applied to the character during a long jump	FVector	FVecto
MovementSpeed	The default movement speed	float	600.0f
BrakingFriction	Friction coefficient applied when braking	float	10.0f
MaxAcceleration	The rate of change of velocity	float	2500.0f
bCanSprint	Can the character sprint?	bool	true
SprintSpeed	The movement speed while the character is sprinting	float	950.0f
bSprintRequiresStamina	Does sprinting require stamina?	bool	false
SprintStaminaStatName	The name of the stamina stat used for sprinting	FName	
SprintStaminaUseRate	The amount of stamina used per second while sprinting	float	20.0f
bCanCrouch	Can the character crouch?	bool	true
bHoldToCrouch	Should the crouch/prone action be held down to crouch?	bool	true
CrouchSpeed	The movement speed while the character is crouching	float	300.0f
CrouchCurve	The float curve used for smooth crouching	UCurveFloat*	`nullptr`
bCanProne	Can the character prone?	bool	true
bHoldToProne	Should the crouch/prone action be held down to prone?	bool	true
ProneTraceHeight	The height of the trace used to check if the character can prone	float	5.0f
ProneTraceOffsetZ	The Z-offset of the trace used to check if the character can prone	float	-20.0f
ProneMoveTraceSizeMultiplier	The multiplier applied to the trace size when the character is moving while proning	float	0.5f
ProneSpeed	The movement speed while the character is proning	float	100.0f
ProneCurve	The float curve used for smooth proning	UCurveFloat*	`nullptr`
bCanStomp	Can the character perform a stomp?	bool	true
MinAirTimeBeforeStomping	The minimum air time before the character can perform a stomp	float	0.3f
StompVelocity	The velocity applied to the character after performing a stomp	float	-5000.0f
StompZeroGravityDuration	The amount of time the character will be in zero gravity after performing a stomp	float	0.5f

StompLandVelocity	The velocity applied to the character when landing after performing a stomp	float	750.0f
StompRecoveryTime	The time it takes (in seconds) before the character recovers from a stomp and can move again	float	0.25f
StompCameraShake	The camera shake applied when performing a stomp	TSubclassOf<UCameraShakeBase>	
bCanLean	Can the character lean?	bool	true
LeanSpeed	The speed used when leaning	float	5.0f
LeanOffset	The offset applied to the camera when leaning	float	50.0f
LeanRotation	The rotation applied to the camera when leaning	float	25.0f
bLookAtCenterWhileIdle	Should the character automatically rotate to look at the center of the screen while idle?	bool	true
MaxLookAtCenterRotationWhileIdle	The maximum rotation that can be applied while looking at the center of the screen while idle	float	90.0f
bLookAtCenterWhileRunning	Should the character automatically rotate to look at the center of the screen while running?	bool	true
MaxLookAtCenterRotationWhileRunning	The maximum rotation that can be applied while looking at the center of the screen while running	float	90.0f
bLookAtCenterWhileSprinting	Should the character automatically rotate to look at the center of the screen while sprinting?	bool	true
MaxLookAtCenterRotationWhileSprinting	The maximum rotation that can be applied while looking at the center of the screen while sprinting	float	90.0f
bLookAtCenterWhileCrouching	Should the character automatically rotate to look at the center of the screen while crouching?	bool	true
MaxLookAtCenterRotationWhileCrouching	The maximum rotation that can be applied while looking at the center of the screen while crouching	float	60.0f
bLookAtCenterWhileProning	Should the character automatically rotate to look at the center of the screen while proning?	bool	true
MaxLookAtCenterRotationWhileProning	The maximum rotation that can be applied while looking at the center of the screen while proning	float	45.0f
LookAtCenterCurve	The float curve used to look at the center of the screen	UCurveFloat*	`nullptr`
bCanSlide	Can the character perform a slide?	bool	true

SlideSpeed	The movement speed while the character is sliding	float	2500.0f
bSlideRequiresSprinting	Does sliding require the character to sprint?	bool	false
SlideMinSpeed	The minimum movement speed before a slide is allowed	float	400.0f
SlidingTime	The time the character is allowed to slide	float	0.5f
SlidingCooldown	The cooldown after a slide before another slide can be performed	float	0.5f
SlideSoundEffects	The sound effect played when the character is sliding	TArray<USoundBase*>	
MaxWeapons	The maximum amount of weapons that the character can carry	int	2
DefaultWeaponClass	The default weapon the character will equip on spawn	TSubclassOf<AWeapon>	
AimCurve	The float curve used for aiming	UCurveFloat*	`nullptr`
DefaultCrosshair	The default crosshair to use if no weapon is equipped	UCrosshairConfig*	`nullptr`

Events

Name	Description	Params
OnCurrentWeaponUpdated	Event used to notify other classes when the weapon is updated	**Weapon (AWeapon*)** The current weapon used by the character **Ammo (AWeapon*)** The amount of ammo remaining **ReloadAmmo (AWeapon*)** The amount of ammo that can be used to reload the weapon
OnCrosshairUpdated	Event used to notify other classes that the crosshair is updated	**Crosshair (UCrosshairConfig*)** The current crosshair config

Functions

Name	Description	Params	Return
GetCameraComponent	Get the camera used by the character		**UCameraComponent*** The camera used by the character
GetSpringArmComponent	Get the spring arm component of the character		**USpringArmComponent*** The spring arm component responsible for controlling the distance of the camera
GetCameraPerspective	Get the current camera perspective		**ECameraPerspective** The current camera perspective
UpdateCameraPerspective	Update the current camera perspective	**NewCameraPerspective (ECameraPerspective)** The new camera perspective	
SwitchCameraPerspective	Switch the current camera perspective		
SetWeapon	Set the current weapon used by the character	**NewWeapon (AWeapon*)** The new weapon	
GetWeapon	Get the current weapon used by the character		**AWeapon*** The current weapon used by the character
HasWeapon	Check if the character has a weapon		**bool** A boolean value indicating if the character has a weapon
IsCrouching	Check if the character is crouching		**bool** A boolean value indicating if the character is crouching
IsProning	Check if the character is proning		**bool** A boolean value indicating if the character is proning
IsEndingCrouch	Check if the character is busy ending the crouch		**bool** A boolean value indicating if the character is busy ending the crouch
IsEndingProne	Check if the character is busy ending the prone		**bool** A boolean value indicating if the character is busy ending the prone
IsStomping	Check if the character is stomping		**bool** A boolean value indicating if the character is stomping
IsStompStarting	Check if the character is starting to stomp		**bool** A boolean value indicating if the character is starting to stomp
GetLeanCameraRoll	Get the current lean camera roll		**float** The current lean camera roll
GetLookAtCenterRotation	Get the look at center rotation		**float** The look at center rotation

IsSliding	Check if the character is sliding		**bool** A boolean value indicating if the character is sliding
IsEndingSlide	Check if the character is busy ending the slide		**bool** A boolean value indicating if the character is busy ending the slide
GetInteractTrigger	Get the current interact trigger		**UInteractTrigger*** The current interact trigger
UpdateInteractTrigger	Update the current interact trigger	**NewInteractTrigger (UInteractTrigger*)** The new interact trigger	
OnWeaponUpdated	Called when the current weapon is updated		
OnNewWeaponEquipped	Called when a new weapon is equipped for the first time	**Weapon (AWeapon*)** The new weapon that was equipped	
IsAiming	Check if the character is aiming		**bool** A boolean value indicating if the character is aiming
GetCrosshair	Get the crosshair configuration		**UCrosshairConfig*** The crosshair configuration
GetWeaponSway	Get the current weapon sway rotation		**FRotator** The current weapon sway rotation
StartFiringWeapon	Start firing the current weapon		
StopFiringWeapon	Stop firing the current weapon		
EquipNextWeapon	Equip the next weapon in the list of available weapons		
EquipPreviousWeapon	Equip the previous weapon in the list of available weapons		
StartCrouching	Start crouching		
StopCrouching	Stop crouching		
StartProning	Start proning		
StopProning	Stop proning		
UpdateCharacterMeshLocationWhileCrouching	Update the character mesh location while crouching	**SizeDifference (float)** The difference between the original capsule size and the crouched capsule size	

10.2.6 Blueprint Usage

There is no additional functions exposed to Blueprints. Just create the character and add it to your level

10.3 FPS Character

10.3.1 Introduction

Base character that can be used for first person shooter (FPS) games

10.3.2 API Reference

10.3.3 Blueprint Usage

There is no additional functions exposed to Blueprints. Just create the character and add it to your level

10.4 3D Platformer Character

10.4.1 Introduction

Base character that can be used for 3D platformer games

10.4.2 Dependencies

The `PlatformerCharacter` relies on other components of this plugin to work:

• Logger: Used to log useful information to help you debug any issues you might experience

10.4.3 Components

The `PlatformerCharacter` uses the following components:

Name	Description	Type
SpringArmComponent	Base character that can be used for 3D platformer games	USpringArmComponent*

10.4.4 API Reference

Properties

Property	Description	Type	Default Value
TargetArmLength	Length of the spring arm component	float	350.0f
ArmLengthMultiplier	The multiplier applied to the spring arm component when the character is moving	float	0.4f
CameraAdjustmentSpeed	The speed used when adjusting the camera distance	float	3.0f

Functions

Name	Description	Params	Return
GetSpringArmComponent	Get the spring arm component of the character		**USpringArmComponent*** The spring arm component responsible for controlling the distance of the camera

10.4.5 Blueprint Usage

There is no additional functions exposed to Blueprints. Just create the character and add it to your level

10.5 Animation Instance

10.5.1 Introduction

Base animation instance for USK characters

10.5.2 Dependencies

The `USKCharacterAnimationInstance` relies on other components of this plugin to work:

- Logger: Used to log useful information to help you debug any issues you might experience

10.5.3 Animation Montages

The animation blueprint contains a `USK` slot that can be used to play animation montages

10.5.4 API Reference

Properties

Property	Description	Type	Default Value
BlendBoneName	The name of the bone used to blend the character's animations	FName	
bMeshSpaceRotationBlend	Whether to blend bone rotations in mesh space or in local space	bool	false
bMeshSpaceScaleBlend	Whether to blend bone scales in mesh space or in local space	bool	false
bBlendRootMotionBasedOnRootBone	Should we incorporate the per-bone blend weight of the root bone when lending root motion?	bool	true
bEnableDirectionalMovementAnimations	Should directional movement animations be used?	bool	false
IdleBaseAnimation	The base animation used when the character is in the idle state	UAnimSequence*	`nullptr`
WalkBaseAnimation	The base animation used when the character is walking	UAnimSequence*	`nullptr`
WalkForwardBaseAnimation	The base animation used when the character is walking forward	UAnimSequence*	`nullptr`
WalkBackwardsBaseAnimation	The base animation used when the character is walking backwards	UAnimSequence*	`nullptr`
WalkLeftBaseAnimation	The base animation used when the character is walking left	UAnimSequence*	`nullptr`
WalkRightBaseAnimation	The base animation used when the character is walking right	UAnimSequence*	`nullptr`
RunBaseAnimation	The base animation used when the character is running	UAnimSequence*	`nullptr`
RunForwardBaseAnimation	The base animation used when the character is running forward	UAnimSequence*	`nullptr`
RunBackwardsBaseAnimation	The base animation used when the character is running backwards	UAnimSequence*	`nullptr`
RunLeftBaseAnimation	The base animation used when the character is running left	UAnimSequence*	`nullptr`
RunRightBaseAnimation	The base animation used when the character is running right	UAnimSequence*	`nullptr`
SprintBaseAnimation	The base animation used when the character is sprinting	UAnimSequence*	`nullptr`
SprintForwardBaseAnimation	The base animation used when the character is sprinting forward	UAnimSequence*	`nullptr`
SprintBackwardsBaseAnimation	The base animation used when the character is sprinting backwards	UAnimSequence*	`nullptr`
SprintLeftBaseAnimation	The base animation used when the character is sprinting left	UAnimSequence*	`nullptr`
SprintRightBaseAnimation	The base animation used when the character is sprinting right	UAnimSequence*	`nullptr`
JumpBaseAnimation		UAnimSequence*	`nullptr`

	The base animation used when the character is jumping		
DoubleJumpBaseAnimation	The base animation used when the character is double jumping	UAnimSequence*	`nullptr`
FallBaseAnimation	The base animation used when the character is falling	UAnimSequence*	`nullptr`
LandBaseAnimation	The base animation used when the character is landing	UAnimSequence*	`nullptr`
CrouchStartBaseAnimation	The base animation used when the character starts crouching	UAnimSequence*	`nullptr`
CrouchEndBaseAnimation	The base animation used when the character stops crouching	UAnimSequence*	`nullptr`
CrouchIdleBaseAnimation	The base animation used when the character is crouching and idle	UAnimSequence*	`nullptr`
CrouchWalkBaseAnimation	The base animation used when the character is crouching and walking	UAnimSequence*	`nullptr`
CrouchWalkForwardBaseAnimation	The base animation used when the character is crouching and walking forward	UAnimSequence*	`nullptr`
CrouchWalkBackwardsBaseAnimation	The base animation used when the character is crouching and walking backwards	UAnimSequence*	`nullptr`
CrouchWalkLeftBaseAnimation	The base animation used when the character is crouching and walking left	UAnimSequence*	`nullptr`
CrouchWalkRightBaseAnimation	The base animation used when the character is crouching and walking right	UAnimSequence*	`nullptr`
ProneStartBaseAnimation	The base animation used when the character starts proning	UAnimSequence*	`nullptr`
ProneEndBaseAnimation	The base animation used when the character stops proning	UAnimSequence*	`nullptr`
ProneIdleBaseAnimation	The base animation used when the character is proning and idle	UAnimSequence*	`nullptr`
ProneMoveBaseAnimation	The base animation used when the character is proning and moving	UAnimSequence*	`nullptr`
StompStartBaseAnimation	The base animation used when the character is starting to stomp	UAnimSequence*	`nullptr`
StompFallBaseAnimation	The base animation used when the character is falling after a stomp	UAnimSequence*	`nullptr`
StompLandBaseAnimation	The base animation used when the character is landing after a stomp	UAnimSequence*	`nullptr`
SlideStartBaseAnimation	The base animation used when the character starts sliding	UAnimSequence*	`nullptr`
SlideEndBaseAnimation	The base animation used when the character stops sliding	UAnimSequence*	`nullptr`
SlideLoopBaseAnimation	The base animation used when the character is sliding	UAnimSequence*	`nullptr`

IdleAnimation	The animation used when the character is in the idle state while unarmed	UAnimSequence*	`nullptr`
WalkAnimation	The animation used when the character is walking while unarmed	UAnimSequence*	`nullptr`
RunAnimation	The animation used when the character is running while unarmed	UAnimSequence*	`nullptr`
SprintAnimation	The animation used when the character is sprinting while unarmed	UAnimSequence*	`nullptr`
JumpAnimation	The animation used when the character is jumping while unarmed	UAnimSequence*	`nullptr`
DoubleJumpAnimation	The animation used when the character is double jumping while unarmed	UAnimSequence*	`nullptr`
FallAnimation	The animation used when the character is falling while unarmed	UAnimSequence*	`nullptr`
LandAnimation	The animation used when the character is landing while unarmed	UAnimSequence*	`nullptr`
CrouchStartAnimation	The animation used when the character starts crouching while unarmed	UAnimSequence*	`nullptr`
CrouchEndAnimation	The animation used when the character stops crouching while unarmed	UAnimSequence*	`nullptr`
CrouchIdleAnimation	The animation used when the character is crouching and idle while unarmed	UAnimSequence*	`nullptr`
CrouchWalkAnimation	The animation used when the character is crouching and walking while unarmed	UAnimSequence*	`nullptr`
ProneStartAnimation	The animation used when the character starts proning while unarmed	UAnimSequence*	`nullptr`
ProneEndAnimation	The animation used when the character stops proning while unarmed	UAnimSequence*	`nullptr`
ProneIdleAnimation	The animation used when the character is proning and idle while unarmed	UAnimSequence*	`nullptr`
ProneMoveAnimation	The animation used when the character is proning and moving while unarmed	UAnimSequence*	`nullptr`
StompStartAnimation	The animation used when the character is starting to stomp while unarmed	UAnimSequence*	`nullptr`
StompFallAnimation	The animation used when the character is falling after a stomp while unarmed	UAnimSequence*	`nullptr`
StompLandAnimation	The animation used when the character is landing after a stomp while unarmed	UAnimSequence*	`nullptr`
SlideStartAnimation	The animation used when the character starts sliding while unarmed	UAnimSequence*	`nullptr`
SlideEndAnimation	The animation used when the character stops sliding while unarmed	UAnimSequence*	`nullptr`
SlideLoopAnimation	The animation used when the character is sliding while unarmed	UAnimSequence*	`nullptr`

IdleWeaponOneHandedAnimation	The animation used when the character is in the idle state with a one handed weapon	UAnimSequence*	`nullptr`
WalkWeaponOneHandedAnimation	The animation used when the character is walking with a one handed weapon	UAnimSequence*	`nullptr`
RunWeaponOneHandedAnimation	The animation used when the character is running with a one handed weapon	UAnimSequence*	`nullptr`
SprintWeaponOneHandedAnimation	The animation used when the character is sprinting with a one handed weapon	UAnimSequence*	`nullptr`
JumpWeaponOneHandedAnimation	The animation used when the character is jumping with a one handed weapon	UAnimSequence*	`nullptr`
DoubleJumpWeaponOneHandedAnimation	The animation used when the character is double jumping with a one handed weapon	UAnimSequence*	`nullptr`
FallWeaponOneHandedAnimation	The animation used when the character is falling with a one handed weapon	UAnimSequence*	`nullptr`
LandWeaponOneHandedAnimation	The animation used when the character is landing with a one handed weapon	UAnimSequence*	`nullptr`
CrouchStartWeaponOneHandedAnimation	The animation used when the character starts crouching with a one handed weapon	UAnimSequence*	`nullptr`
CrouchEndWeaponOneHandedAnimation	The animation used when the character stops crouching with a one handed weapon	UAnimSequence*	`nullptr`
CrouchIdleWeaponOneHandedAnimation	The animation used when the character is crouching and idle with a one handed weapon	UAnimSequence*	`nullptr`
CrouchWalkWeaponOneHandedAnimation	The animation used when the character is crouching and walking with a one handed weapon	UAnimSequence*	`nullptr`
ProneStartWeaponOneHandedAnimation	The animation used when the character starts proning with a one handed weapon	UAnimSequence*	`nullptr`
ProneEndWeaponOneHandedAnimation	The animation used when the character stops proning with a one handed weapon	UAnimSequence*	`nullptr`
ProneIdleWeaponOneHandedAnimation	The animation used when the character is proning and idle with a one handed weapon	UAnimSequence*	`nullptr`
ProneMoveWeaponOneHandedAnimation	The animation used when the character is proning and moving with a one handed weapon	UAnimSequence*	`nullptr`
StompStartWeaponOneHandedAnimation	The animation used when the character is starting to stomp with a one handed weapon	UAnimSequence*	`nullptr`
StompFallWeaponOneHandedAnimation	The animation used when the character is falling after a stomp with a one handed weapon	UAnimSequence*	`nullptr`
StompLandWeaponOneHandedAnimation	The animation used when the character is landing after a stomp with a one handed weapon	UAnimSequence*	`nullptr`
SlideStartWeaponOneHandedAnimation	The animation used when the character starts sliding with a one handed weapon	UAnimSequence*	`nullptr`
SlideEndWeaponOneHandedAnimation	The animation used when the character stops sliding with a one handed weapon	UAnimSequence*	`nullptr`
SlideLoopWeaponOneHandedAnimation	The animation used when the character is sliding with a one handed weapon	UAnimSequence*	`nullptr`

IdleWeaponAimOneHandedAnimation	The animation used when the character is in the idle state with a one handed weapon while aiming	UAnimSequence*	nullptr
WalkWeaponAimOneHandedAnimation	The animation used when the character is walking with a one handed weapon while aiming	UAnimSequence*	nullptr
RunWeaponAimOneHandedAnimation	The animation used when the character is running with a one handed weapon while aiming	UAnimSequence*	nullptr
SprintWeaponAimOneHandedAnimation	The animation used when the character is sprinting with a one handed weapon while aiming	UAnimSequence*	nullptr
JumpWeaponAimOneHandedAnimation	The animation used when the character is jumping with a one handed weapon while aiming	UAnimSequence*	nullptr
DoubleJumpWeaponAimOneHandedAnimation	The animation used when the character is double jumping with a one handed weapon while aiming	UAnimSequence*	nullptr
FallWeaponAimOneHandedAnimation	The animation used when the character is falling with a one handed weapon while aiming	UAnimSequence*	nullptr
LandWeaponAimOneHandedAnimation	The animation used when the character is landing with a one handed weapon while aiming	UAnimSequence*	nullptr
CrouchStartWeaponAimOneHandedAnimation	The animation used when the character starts crouching with a one handed weapon while aiming	UAnimSequence*	nullptr
CrouchEndWeaponAimOneHandedAnimation	The animation used when the character stops crouching with a one handed weapon while aiming	UAnimSequence*	nullptr
CrouchIdleWeaponAimOneHandedAnimation	The animation used when the character is crouching and idle with a one handed weapon while aiming	UAnimSequence*	nullptr
CrouchWalkWeaponAimOneHandedAnimation	The animation used when the character is crouching and walking with a one handed weapon while aiming	UAnimSequence*	nullptr
ProneStartWeaponAimOneHandedAnimation	The animation used when the character starts proning with a one handed weapon while aiming	UAnimSequence*	nullptr
ProneEndWeaponAimOneHandedAnimation	The animation used when the character stops proning with a one handed weapon while aiming	UAnimSequence*	nullptr
ProneIdleWeaponAimOneHandedAnimation	The animation used when the character is proning and idle with a one handed weapon while aiming	UAnimSequence*	nullptr
ProneMoveWeaponAimOneHandedAnimation	The animation used when the character is proning and moving with a one handed weapon while aiming	UAnimSequence*	nullptr
StompStartWeaponAimOneHandedAnimation	The animation used when the character is starting to stomp with a one handed weapon while aiming	UAnimSequence*	nullptr
StompFallWeaponAimOneHandedAnimation	The animation used when the character is falling after a stomp with a one handed weapon while aiming	UAnimSequence*	nullptr
StompLandWeaponAimOneHandedAnimation	The animation used when the character is landing after a stomp with a one handed weapon while aiming	UAnimSequence*	nullptr
SlideStartWeaponAimOneHandedAnimation	The animation used when the character starts sliding with a one handed weapon while aiming	UAnimSequence*	nullptr
SlideEndWeaponAimOneHandedAnimation	The animation used when the character stops sliding with a one handed weapon while aiming	UAnimSequence*	nullptr

SlideLoopWeaponAimOneHandedAnimation	The animation used when the character is sliding with a one handed weapon while aiming	UAnimSequence*	nullptr
IdleWeaponTwoHandedAnimation	The animation used when the character is in the idle state with a two handed weapon	UAnimSequence*	nullptr
WalkWeaponTwoHandedAnimation	The animation used when the character is walking with a two handed weapon	UAnimSequence*	nullptr
RunWeaponTwoHandedAnimation	The animation used when the character is running with a two handed weapon	UAnimSequence*	nullptr
SprintWeaponTwoHandedAnimation	The animation used when the character is sprinting with a two handed weapon	UAnimSequence*	nullptr
JumpWeaponTwoHandedAnimation	The animation used when the character is jumping with a two handed weapon	UAnimSequence*	nullptr
DoubleJumpWeaponTwoHandedAnimation	The animation used when the character is double jumping with a two handed weapon	UAnimSequence*	nullptr
FallWeaponTwoHandedAnimation	The animation used when the character is falling with a two handed weapon	UAnimSequence*	nullptr
LandWeaponTwoHandedAnimation	The animation used when the character is landing with a two handed weapon	UAnimSequence*	nullptr
CrouchStartWeaponTwoHandedAnimation	The animation used when the character starts crouching with a two handed weapon	UAnimSequence*	nullptr
CrouchEndWeaponTwoHandedAnimation	The animation used when the character stops crouching with a two handed weapon	UAnimSequence*	nullptr
CrouchIdleWeaponTwoHandedAnimation	The animation used when the character is crouching and idle with a two handed weapon	UAnimSequence*	nullptr
CrouchWalkWeaponTwoHandedAnimation	The animation used when the character is crouching and walking with a two handed weapon	UAnimSequence*	nullptr
ProneStartWeaponTwoHandedAnimation	The animation used when the character starts proning with a two handed weapon	UAnimSequence*	nullptr
ProneEndWeaponTwoHandedAnimation	The animation used when the character stops proning with a two handed weapon	UAnimSequence*	nullptr
ProneIdleWeaponTwoHandedAnimation	The animation used when the character is proning and idle with a two handed weapon	UAnimSequence*	nullptr
ProneMoveWeaponTwoHandedAnimation	The animation used when the character is proning and moving with a two handed weapon	UAnimSequence*	nullptr
StompStartWeaponTwoHandedAnimation	The animation used when the character is starting to stomp with a two handed weapon	UAnimSequence*	nullptr
StompFallWeaponTwoHandedAnimation	The animation used when the character is falling after a stomp with a two handed weapon	UAnimSequence*	nullptr
StompLandWeaponTwoHandedAnimation	The animation used when the character is landing after a stomp with a two handed weapon	UAnimSequence*	nullptr
SlideStartWeaponTwoHandedAnimation	The animation used when the character starts sliding with a two handed weapon	UAnimSequence*	nullptr
SlideEndWeaponTwoHandedAnimation	The animation used when the character stops sliding with a two handed weapon	UAnimSequence*	nullptr

SlideLoopWeaponTwoHandedAnimation	The animation used when the character is sliding with a two handed weapon	UAnimSequence*	`nullptr`
IdleWeaponAimTwoHandedAnimation	The animation used when the character is in the idle state with a two handed weapon while aiming	UAnimSequence*	`nullptr`
WalkWeaponAimTwoHandedAnimation	The animation used when the character is walking with a two handed weapon while aiming	UAnimSequence*	`nullptr`
RunWeaponAimTwoHandedAnimation	The animation used when the character is running with a two handed weapon while aiming	UAnimSequence*	`nullptr`
SprintWeaponAimTwoHandedAnimation	The animation used when the character is sprinting with a two handed weapon while aiming	UAnimSequence*	`nullptr`
JumpWeaponAimTwoHandedAnimation	The animation used when the character is jumping with a two handed weapon while aiming	UAnimSequence*	`nullptr`
DoubleJumpWeaponAimTwoHandedAnimation	The animation used when the character is double jumping with a two handed weapon while aiming	UAnimSequence*	`nullptr`
FallWeaponAimTwoHandedAnimation	The animation used when the character is falling with a two handed weapon while aiming	UAnimSequence*	`nullptr`
LandWeaponAimTwoHandedAnimation	The animation used when the character is landing with a two handed weapon while aiming	UAnimSequence*	`nullptr`
CrouchStartWeaponAimTwoHandedAnimation	The animation used when the character starts crouching with a two handed weapon while aiming	UAnimSequence*	`nullptr`
CrouchEndWeaponAimTwoHandedAnimation	The animation used when the character stops crouching with a two handed weapon while aiming	UAnimSequence*	`nullptr`
CrouchIdleWeaponAimTwoHandedAnimation	The animation used when the character is crouching and idle with a two handed weapon while aiming	UAnimSequence*	`nullptr`
CrouchWalkWeaponAimTwoHandedAnimation	The animation used when the character is crouching and walking with a two handed weapon while aiming	UAnimSequence*	`nullptr`
ProneStartWeaponAimTwoHandedAnimation	The animation used when the character starts proning with a two handed weapon while aiming	UAnimSequence*	`nullptr`
ProneEndWeaponAimTwoHandedAnimation	The animation used when the character stops proning with a two handed weapon while aiming	UAnimSequence*	`nullptr`
ProneIdleWeaponAimTwoHandedAnimation	The animation used when the character is proning and idle with a two handed weapon while aiming	UAnimSequence*	`nullptr`
ProneMoveWeaponAimTwoHandedAnimation	The animation used when the character is proning and moving with a two handed weapon while aiming	UAnimSequence*	`nullptr`
StompStartWeaponAimTwoHandedAnimation	The animation used when the character is starting to stomp with a two handed weapon while aiming	UAnimSequence*	`nullptr`
StompFallWeaponAimTwoHandedAnimation	The animation used when the character is falling after a stomp with a two handed weapon while aiming	UAnimSequence*	`nullptr`
StompLandWeaponAimTwoHandedAnimation	The animation used when the character is landing after a stomp with a two handed weapon while aiming	UAnimSequence*	`nullptr`
SlideStartWeaponAimTwoHandedAnimation	The animation used when the character starts sliding with a two handed weapon while aiming	UAnimSequence*	`nullptr`

SlideEndWeaponAimTwoHandedAnimation	The animation used when the character stops sliding with a two handed weapon while aiming	UAnimSequence*	`nullptr`
SlideLoopWeaponAimTwoHandedAnimation	The animation used when the character is sliding with a two handed weapon while aiming	UAnimSequence*	`nullptr`
MovementBlendSpeed	The blend speed used when updating the movement speed	float	10.0f
MovementSpeed	The movement speed of the character	float	
MovementDirection	The movement direction of the character multiplied by the movement speed	FVector2D	
bIsCrouching	Is the character currently crouching?	bool	false
bIsProning	Is the character currently proning?	bool	false
bIsStomping	Is the character currently stomping?	bool	false
StompBlendValue	The value used to blend stomp animations	float	
bIsStompStarting	Is the character starting the stomp?	bool	false
bIsEndingCrouch	Is the character busy ending the crouch?	bool	false
bIsEndingProne	Is the character busy ending the prone?	bool	false
IsInAir	Is the character currently in the air?	bool	false
IsDoubleJumping	Is the character double jumping?	bool	false
LeanCameraRotationModifier	The modifier applied to the lean camera rotation	float	1.0f
LeanCameraRoll	The lean camera roll	float	
LeanBoneName	The bone to rotate while the character is leaning	FName	
bIsSliding	Is the character currently sliding?	bool	false
bIsEndingSlide	Is the character busy ending the slide?	bool	false
bIsAiming	Is the character aiming?	bool	false
AimRotation	The rotation applied to the character while aiming	float	
LookAtCenterRotation	The rotation applied to the character to look at the center of the screen	float	
AimBoneName	The bone to rotate while the player is aiming up or down using the first person camera perspective	FName	
WeaponSwayBoneName	The bone to rotate to simulate weapon sway	FName	
WeaponSway	The weapon sway rotation applied to the character	FRotator	

Functions

Name	Description	Params	Return
GetIdleBaseAnimation	Get the base idle animation		**UAnimSequence*** The idle animation to play
GetWalkBaseAnimation	Get the base walk animation		**UAnimSequence*** The walk animation to play
GetWalkForwardBaseAnimation	Get the base walk forward animation		**UAnimSequence*** The walk forward animation to play
GetWalkBackwardsBaseAnimation	Get the base walk backwards animation		**UAnimSequence*** The walk backwards animation to play
GetWalkLeftBaseAnimation	Get the base walk left animation		**UAnimSequence*** The walk left animation to play
GetWalkRightBaseAnimation	Get the base walk right animation		**UAnimSequence*** The walk right animation to play
GetRunBaseAnimation	Get the base run animation		**UAnimSequence*** The run animation to play
GetRunForwardBaseAnimation	Get the base run forward animation		**UAnimSequence*** The run forward animation to play
GetRunBackwardsBaseAnimation	Get the base run backwards animation		**UAnimSequence*** The run backwards animation to play
GetRunLeftBaseAnimation	Get the base run left animation		**UAnimSequence*** The run left animation to play
GetRunRightBaseAnimation	Get the base run right animation		**UAnimSequence*** The run right animation to play
GetSprintBaseAnimation	Get the base sprint animation		**UAnimSequence*** The sprint animation to play
GetSprintForwardBaseAnimation	Get the base sprint forward animation		**UAnimSequence*** The sprint forward animation to play
GetSprintBackwardsBaseAnimation	Get the base sprint backwards animation		**UAnimSequence*** The sprint backwards animation to play
GetSprintLeftBaseAnimation	Get the base sprint left animation		**UAnimSequence*** The sprint left animation to play
GetSprintRightBaseAnimation	Get the base sprint right animation		**UAnimSequence*** The sprint right animation to play
GetJumpBaseAnimation	Get the base jump animation		**UAnimSequence*** The jump animation to play
GetDoubleJumpBaseAnimation	Get the base double jump animation		**UAnimSequence*** The double jump animation to play
GetFallBaseAnimation	Get the base fall animation		**UAnimSequence*** The fall animation to play

GetLandBaseAnimation	Get the base land animation		**UAnimSequence*** The land animation to play
GetCrouchStartBaseAnimation	Get the base crouch start animation		**UAnimSequence*** The crouch start animation to play
GetCrouchEndBaseAnimation	Get the base crouch end animation		**UAnimSequence*** The crouch end animation to play
GetCrouchIdleBaseAnimation	Get the base crouch idle animation		**UAnimSequence*** The crouch idle animation to play
GetCrouchWalkBaseAnimation	Get the base crouch walk animation		**UAnimSequence*** The crouch walk animation to play
GetCrouchWalkForwardBaseAnimation	Get the base crouch walk forward animation		**UAnimSequence*** The crouch walk forward animation to play
GetCrouchWalkBackwardsBaseAnimation	Get the base crouch walk backwards animation		**UAnimSequence*** The crouch walk backwards animation to play
GetCrouchWalkLeftBaseAnimation	Get the base crouch walk left animation		**UAnimSequence*** The crouch walk left animation to play
GetCrouchWalkRightBaseAnimation	Get the base crouch walk right animation		**UAnimSequence*** The crouch walk right animation to play
GetProneStartBaseAnimation	Get the base prone start animation		**UAnimSequence*** The prone start animation to play
GetProneEndBaseAnimation	Get the base prone end animation		**UAnimSequence*** The prone end animation to play
GetProneIdleBaseAnimation	Get the base prone idle animation		**UAnimSequence*** The prone idle animation to play
GetProneMoveBaseAnimation	Get the base prone walk animation		**UAnimSequence*** The prone walk animation to play
GetStompStartBaseAnimation	Get the base stomp start animation		**UAnimSequence*** The stomp start animation to play
GetStompFallBaseAnimation	Get the base stomp fall animation		**UAnimSequence*** The stomp fall animation to play
GetStompLandBaseAnimation	Get the base stomp land animation		**UAnimSequence*** The stomp land animation to play
GetSlideStartBaseAnimation	Get the base slide start animation		**UAnimSequence*** The slide start animation to play
GetSlideEndBaseAnimation	Get the base slide end animation		**UAnimSequence*** The slide end animation to play
GetSlideLoopBaseAnimation	Get the base slide loop animation		**UAnimSequence*** The slide loop animation to play
GetIdleAnimation	Get the idle animation based on the current armed state		**UAnimSequence*** The idle animation to play

GetWalkAnimation	Get the walk animation based on the current armed state		**UAnimSequence*** The walk animation to play
GetRunAnimation	Get the run animation based on the current armed state		**UAnimSequence*** The run animation to play
GetSprintAnimation	Get the sprint animation based on the current armed state		**UAnimSequence*** The sprint animation to play
GetJumpAnimation	Get the jump animation based on the current armed state		**UAnimSequence*** The jump animation to play
GetDoubleJumpAnimation	Get the double jump animation based on the current armed state		**UAnimSequence*** The double jump animation to play
GetFallAnimation	Get the fall animation based on the current armed state		**UAnimSequence*** The fall animation to play
GetLandAnimation	Get the land animation based on the current armed state		**UAnimSequence*** The land animation to play
GetCrouchStartAnimation	Get the crouch start animation based on the current armed state		**UAnimSequence*** The crouch start animation to play
GetCrouchEndAnimation	Get the crouch end animation based on the current armed state		**UAnimSequence*** The crouch end animation to play
GetCrouchIdleAnimation	Get the crouch idle animation based on the current armed state		**UAnimSequence*** The crouch idle animation to play
GetCrouchWalkAnimation	Get the crouch walk animation based on the current armed state		**UAnimSequence*** The crouch walk animation to play
GetProneStartAnimation	Get the prone start animation based on the current armed state		**UAnimSequence*** The prone start animation to play
GetProneEndAnimation	Get the prone end animation based on the current armed state		**UAnimSequence*** The prone end animation to play
GetProneIdleAnimation	Get the prone idle animation based on the current armed state		**UAnimSequence*** The prone idle animation to play
GetProneMoveAnimation	Get the prone walk animation based on the current armed state		**UAnimSequence*** The prone walk animation to play
GetStompStartAnimation	Get the stomp start animation based on the current armed state		**UAnimSequence*** The stomp start animation to play
GetStompFallAnimation	Get the stomp fall animation based on the current armed state		**UAnimSequence*** The stomp fall animation to play
GetStompLandAnimation	Get the stomp fall animation based on the current armed state		**UAnimSequence*** The stomp fall animation to play
GetSlideStartAnimation	Get the slide start animation based on the current armed state		**UAnimSequence*** The slide start animation to play
GetSlideEndAnimation	Get the slide end animation based on the current armed state		**UAnimSequence*** The slide end animation to play
GetSlideLoopAnimation	Get the slide loop animation based on the current armed state		**UAnimSequence*** The slide loop animation to play

10.5.5 Blueprint Usage

You can use this template by creating your own animation blueprint and selecting `UPlatformerAnimationInstance` as the parent class. Set your animations and use this for your characters

10.6 Shadow Decal

10.6.1 Introduction

Decal used to draw a shadow beneath a character when the character is in the air

10.6.2 Dependencies

The `ShadowDecal` relies on other components of this plugin to work:

• Logger: Used to log useful information to help you debug any issues you might experience

10.6.3 API Reference

Functions

Name	Description	Params	Return
Initialize	Initialize the shadow decal	**OwnerCharacter (ACharacter*)** The character owning this shadow decal	

10.6.4 Blueprint Usage

You can use the `ShadowDecal` using Blueprints by adding one of the following nodes:

• Ultimate Starter Kit > Shadow Decal > Initialize

10.6.5 C++ Usage

Before you can use the plugin, you first need to enable the plugin in your `Build.cs` file:

```
PublicDependencyModuleNames.Add("USK");
```

The `ShadowDecal` can now be used in any of your C++ files:

```
#include "USK/Character/ShadowDecal.h"

void ATestActor::Test()
{
    // ShadowDecal is a pointer to the AShadowDecal
    ShadowDecal->Initialize(OwnerCharacter);
}
```

11. Weapons

11.1 Weapon

11.1.1 Introduction

The weapon attached to characters

11.1.2 Dependencies

The `Weapon` relies on other components of this plugin to work:

• Logger: Used to log useful information to help you debug any issues you might experience

• Audio: Used to play sound effects either 2D or at a specified location

11.1.3 Components

The `Weapon` uses the following components:

Name	Description	Type
WeaponMesh	The skeletal mesh of the weapon	USkeletalMeshComponent*
MuzzleFlash	The muzzle flash of the weapon	USceneComponent*

11.1.4 API Reference

Properties

Property	Description	Type	Default Value
WeaponType	The type of weapon	EWeaponType	
WeaponFireMode	The fire mode of weapon	EWeaponFireMode	
AimFov	The FOV of the camera while aiming	float	70
Crosshair	The crosshair used by the weapon	UCrosshairConfig*	`nullptr`
FireRate	The fire rate of the weapon (amount of seconds between each shot)	float	0.2f
MaxShotsPerFireEvent	The amount of shots fired per fire event	int	3
Projectiles	The projectiles spawned by the weapon	TArray<FWeaponProjectileData>	
bInfiniteAmmo	Does the weapon have an infinite amount of ammo?	bool	false
bRequireReloading	Does the weapon require reloading?	bool	true
bAutoReloadWhenFiringWhileEmpty	Should the weapon automatically be reloaded when firing while empty?	bool	true
ReloadDuration	The delay after reloading and before ammo is added to the weapon	float	0.65f
Ammo	The amount of ammo for the weapon	int	50
AmmoPerClip	The amount of ammo per clip	int	20
WeaponAttachPoint	The attach point used by all weapons	FName	
WeaponTransform	The relative transform of the weapon after it is attached to a character	FTransform	
WeaponAimTransform	The relative transform of the weapon while aiming	FTransform	
RecoilCurve	The curve used to add recoil to the weapon	UCurveVector*	`nullptr`
RecoveryTime	The recovery time after recoil was applied	float	1.0f
RecoilRecoveryDelay	The delay before we start recovering from recoil	float	0.15f
bWeaponSway	Should weapon sway be enabled?	bool	true
WeaponSwayMultiplier	The multiplier applied to the weapon sway rotation	float	1.0f
WeaponSwayInterpSpeed	The speed used to interpolate the weapon sway rotation	float	10.0f
FireCameraShake	The camera shake applied when firing the weapon	TSubclassOf<UCameraShakeBase>	
MuzzleFlashParticleFx	The muzzle flash particle effects	UNiagaraSystem*	`nullptr`
MuzzleFlashParticleFxTransform	The transform of the muzzle flash particle effects	FTransform	
FireSound	The sound played each time the weapon is fired	TArray<USoundBase*>	
EmptyClipFireSound		TArray<USoundBase*>	

	The sound played each time the weapon is fired with an empty clip		
ReloadSound	The sound played when the weapon is reloaded	TArray<USoundBase*>	
FireAnimation	The animation played when the weapon is fired	UAnimMontage*	`nullptr`
EmptyClipFireAnimation	The animation played when the weapon is fired with an empty clip	UAnimMontage*	`nullptr`
EquipAnimation	The animation played when the weapon is equipped	UAnimMontage*	`nullptr`
ReloadAnimation	The animation played when the weapon is reloaded	UAnimMontage*	`nullptr`
ReloadAimAnimation	The animation played when the weapon is reloaded while aiming	UAnimMontage*	`nullptr`

Events

Name	Description	Params
OnWeaponEquipped	Event used to notify other classes when the weapon is equipped	
OnWeaponUnequipped	Event used to notify other classes when the weapon is unequipped	
OnWeaponFired	Event used to notify other classes when the weapon is fired	
OnWeaponFiredEmptyClip	Event used to notify other classes when the weapon is fired with an empty clip	
OnWeaponAmmoUpdated	Event used to notify other classes when the weapon ammo is updated	**Weapon (AWeapon*)** The current weapon used by the character **Ammo (AWeapon*)** The amount of ammo remaining **ReloadAmmo (AWeapon*)** The amount of ammo that can be used to reload the weapon
OnWeaponAmmoEmpty	Event used to notify other classes when the ammo is empty	

Functions

Name	Description	Params	Return
Equip	Equip the weapon	**TargetCharacter (AUSKCharacter*)** The character that will use the weapon **IsNewWeapon (bool)** Is this a new weapon?	
Unequip	Unequip the weapon		
StartFiring	Start firing the weapon		
StopFiring	Stop firing the weapon		
AddAmmo	Add more ammo to the weapon	**Amount (int)** The amount of ammo to add	
RemoveAmmo	Remove ammo from the weapon	**Amount (int)** The amount of ammo to remove	
GetAmmoRemaining	Get the amount of ammo remaining		**int** The amount of ammo remaining
GetReloadAmmoRemaining	Get the amount of ammo that can be used when reloading		**int** The amount of ammo that can be used when reloading
Reload	Reload the weapon		
StartRecoil	Start applying recoil to the weapon		
StopRecoil	Stop applying recoil to the weapon		
ApplyRecoil	Apply recoil to the weapon	**DeltaSeconds (float)** Game time elapsed during last frame modified by the time dilation	
StartRecoilRecovery	Start recovering from recoil		
StopRecoilRecovery	Stop recovering from recoil		
ApplyRecoilRecovery	Recover from recoil	**DeltaSeconds (float)** Game time elapsed during last frame modified by the time dilation	

11.1.5 Blueprint Usage

You can use the `Weapon` using Blueprints by adding one of the following nodes:

- Ultimate Starter Kit > Weapon > Equip
- Ultimate Starter Kit > Weapon > Unequip
- Ultimate Starter Kit > Weapon > Start Firing
- Ultimate Starter Kit > Weapon > Stop Firing
- Ultimate Starter Kit > Weapon > Add Ammo
- Ultimate Starter Kit > Weapon > Remove Ammo
- Ultimate Starter Kit > Weapon > Get Ammo Remaining
- Ultimate Starter Kit > Weapon > Get Reload Ammo Remaining
- Ultimate Starter Kit > Weapon > Reload
- Ultimate Starter Kit > Weapon > Start Recoil
- Ultimate Starter Kit > Weapon > Stop Recoil
- Ultimate Starter Kit > Weapon > Apply Recoil
- Ultimate Starter Kit > Weapon > Start Recoil Recovery
- Ultimate Starter Kit > Weapon > Stop Recoil Recovery
- Ultimate Starter Kit > Weapon > Apply Recoil Recovery

11.1.6 C++ Usage

Before you can use the plugin, you first need to enable the plugin in your `Build.cs` file:

```
PublicDependencyModuleNames.Add("USK");
```

The `Weapon` can now be used in any of your C++ files:

```
#include "USK/Weapons/Weapon.h"

void ATestActor::Test()
{
    // Weapon is a pointer to the AWeapon
    Weapon->Equip(TargetCharacter, IsNewWeapon);
    Weapon->Unequip();
    Weapon->StartFiring();
    Weapon->StopFiring();
    Weapon->AddAmmo(Amount);
    Weapon->RemoveAmmo(Amount);
    int AmmoRemaining = Weapon->GetAmmoRemaining();
    int ReloadAmmoRemaining = Weapon->GetReloadAmmoRemaining();
    Weapon->Reload();
    Weapon->StartRecoil();
    Weapon->StopRecoil();
    Weapon->ApplyRecoil(DeltaSeconds);
    Weapon->StartRecoilRecovery();
    Weapon->StopRecoilRecovery();
    Weapon->ApplyRecoilRecovery(DeltaSeconds);
}
```

11.2 Weapon Type

11.2.1 Introduction

The types of weapons

11.2.2 Values

Value	Description
WeaponOneHanded	A one handed weapon
WeaponTwoHanded	A two handed weapon

11.3 Weapon Fire Mode

11.3.1 Introduction

The types of weapon fire modes

11.3.2 Values

Value	Description
SingleShot	Fires a single shot each time the trigger is pulled
SemiAuto	Fires multiple projectiles each time the trigger is pulled
FullAuto	Continuously fires projectiles while the trigger is held

11.4 Weapon Item

11.4.1 Introduction

A weapon item that can be picked up by a character

11.4.2 API Reference

Properties

Property	Description	Type	Default Value
WeaponClass	The weapon assigned to this item	TSubclassOf<AWeapon>	

11.5 Weapon Ammo Item

11.5.1 Introduction

A weapon ammo item that can be picked up by a character

11.5.2 Dependencies

The `WeaponAmmoItem` relies on other components of this plugin to work:

• Logger: Used to log useful information to help you debug any issues you might experience

11.5.3 API Reference

Properties

Property	Description	Type	Default Value
Ammo	The amount of ammo to add to the weapon	int	10
bAddToAnyWeapon	Should the ammo be added to any weapon?	bool	false
AllowedWeapons	The weapons that can use this ammo	TArray<TSubclassOf<AWeapon>>	

11.6 Weapon Projectile

11.6.1 Introduction

The projectile spawned by weapons

11.6.2 Dependencies

The `WeaponProjectile` relies on other components of this plugin to work:

- Logger: Used to log useful information to help you debug any issues you might experience

- Audio: Used to play sound effects either 2D or at a specified location

11.6.3 Components

The `WeaponProjectile` uses the following components:

Name	Description	Type
CollisionComponent	The projectile spawned by weapons	USphereComponent*
ProjectileMovementComponent	The projectile movement component used to move the projectile	UProjectileMovementComponent*

11.6.4 API Reference

Properties

Property	Description	Type	Default Value
bDestroyOnHit	Should the projectile be destroyed after hitting something?	bool	true
HitImpulse	The impulse applied to the component that was hit	float	
DefaultHitReaction	The default hit reaction of the projectile	FWeaponProjectileHitData	
HitReactions	A list of hit reactions for specific actors	TMap<TSubclassOf<AActor>, FWeaponProjectileHitData>	
DefaultDecal	The default decal that is spawned when the projectile hits something	TSubclassOf<AWeaponProjectileDecal>	
Decals	A list of decals for specific actors	TMap<TSubclassOf<AWeaponProjectileDecal>, TSubclassOf<AWeaponProjectileDecal>>	

Functions

Name	Description	Params	Return
GetCollisionComponent	Get the collision component used by the projectile		**USphereComponent*** The collision component used by the projectile
GetProjectileMovementComponent	Get the projectile movement component used to move the projectile		**UProjectileMovementComponent*** The projectile movement component used to move the projectile
NormalImpulse, const FHitResult& HitResult);	Called after the projectile hits something	**HitComponent (HitResult);)** The component responsible for the hit **OtherActor (HitResult);)** The actor that was hit **OtherComponent (HitResult);)** The component that was hit **NormalImpulse (FVector)** The normal impulse of the hit **HitResult (FHitResult&)** Result describing the hit	

11.6.5 Blueprint Usage

You can use the `WeaponProjectile` using Blueprints by adding one of the following nodes:

- Ultimate Starter Kit > Weapon Projectile > Get Collision Component

- Ultimate Starter Kit > Weapon Projectile > Get Projectile Movement Component

- Ultimate Starter Kit > Weapon Projectile > Normal Impulse, const FHit Result& Hit Result);

11.6.6 C++ Usage

Before you can use the plugin, you first need to enable the plugin in your `Build.cs` file:

```
PublicDependencyModuleNames.Add("USK");
```

The `WeaponProjectile` can now be used in any of your C++ files:

```
#include "USK/Weapons/WeaponProjectile.h"

void ATestActor::Test()
{
    // WeaponProjectile is a pointer to the AWeaponProjectile
    USphereComponent* CollisionComponent = WeaponProjectile->GetCollisionComponent();
    UProjectileMovementComponent* ProjectileMovementComponent = WeaponProjectile->GetProjectileMovementComponent();
    WeaponProjectile->NormalImpulse, const FHitResult& HitResult);(HitComponent, OtherActor, OtherComponent, NormalImpulse, HitResult);
}
```

11.7 Weapon Projectile Data

11.7.1 Introduction

Structure describing a projectile spawned by a weapon

11.7.2 Properties

Property	Description	Type	Default Value
ProjectileClass	The class of the projectile to spawn	TSubclassOf<AWeaponProjectile>	
SpawnTransform	The relative transform of the projectile	FTransform	

11.8 Weapon Projectile Hit Data

11.8.1 Introduction

Structure describing the hit data of a projectile

11.8.2 Properties

Property	Description	Type	Default Value
HitParticleFx	The particle FX to spawn when the projectile hits the actor	UNiagaraSystem*	`nullptr`
HitParticleFxTransform	The transform of the particle FX	FTransform	
HitSfx	The sound effect to play when the projectile hits the actor	TArray<USoundBase*>	

11.9 Weapon Projectile Decal

11.9.1 Introduction

A decal spawned when a projectile hits something

11.9.2 Dependencies

The `WeaponProjectileDecal` relies on other components of this plugin to work:

- Logger: Used to log useful information to help you debug any issues you might experience

11.9.3 API Reference

Properties

Property	Description	Type	Default Value
bAutomaticallyDestroy	Should the decal automatically be destroyed?	bool	true
Lifetime	The lifetime of the decal	float	10.0f

Functions

Name	Description	Params	Return
DestroyDecal	Destroy the decal actor		

11.9.4 Blueprint Usage

You can use the `WeaponProjectileDecal` using Blueprints by adding one of the following nodes:

- Ultimate Starter Kit > Weapon Projectile Decal > Destroy Decal

11.9.5 C++ Usage

Before you can use the plugin, you first need to enable the plugin in your `Build.cs` file:

```
PublicDependencyModuleNames.Add("USK");
```

The `WeaponProjectileDecal` can now be used in any of your C++ files:

```
#include "USK/Weapons/WeaponProjectileDecal.h"

void ATestActor::Test()
{
    // WeaponProjectileDecal is a pointer to the AWeaponProjectileDecal
    WeaponProjectileDecal->DestroyDecal();
}
```

11.10 Weapon Utils

11.10.1 Introduction

A Blueprint Function Library class used to manage weapons

11.10.2 Dependencies

The `WeaponUtils` relies on other components of this plugin to work:

• Logger: Used to log useful information to help you debug any issues you might experience

11.10.3 API Reference

Functions

Name	Description	Params	Return
EquipWeapon	Equip a weapon to a character	**Owner (AUSKCharacter*)** The owner character **WeaponClass (TSubclassOf)** The weapon class to equip	

11.10.4 Blueprint Usage

You can use the `WeaponUtils` using Blueprints by adding one of the following nodes:

• Ultimate Starter Kit > Weapons > Equip Weapon

11.10.5 C++ Usage

Before you can use the plugin, you first need to enable the plugin in your `Build.cs` file:

```
PublicDependencyModuleNames.Add("USK");
```

The `WeaponUtils` can now be used in any of your C++ files:

```
#include "USK/Weapons/WeaponUtils.h"

void ATestActor::Test()
{
    UWeaponUtils::EquipWeapon(Owner, WeaponClass);
}
```

12. Dynamic Crosshair

12.1 Config

12.1.1 Introduction

The crosshair configuration data asset

12.1.2 API Reference

Properties

Property	Description	Type	Default Value
bDisplayCenterImage	Should the center image be displayed?	bool	true
CenterImage	The image to display in the center of the crosshair	UTexture2D*	`nullptr`
CenterColor	The color of the center image	FLinearColor	FLinearColor::White
bDisplayCenterShadow	Should a shadow be displayed behind the center image?	bool	true
CenterShadowColor	The color of the shadow behind the center image	FLinearColor	FLinearColor::Black
CenterShadowScale	The scale of the shadow behind the center image	float	1.15f
Rotation	The rotation applied to the sides of the crosshair	float	
SpreadBlendSpeed	The blend speed used when updating the spread of the crosshair	float	10.0f
bUseGlobalColor	Should a global color be used for the sides of the crosshair?	bool	true
GlobalColor	The global color used for the sides of the crosshair	FLinearColor	FLinearColor::White
bDisplayTop	Should the top side of the crosshair be displayed?	bool	true
TopColor	The color of the top side of the crosshair	FLinearColor	FLinearColor::White
TopLength	The length of the top side of the crosshair	float	20.0f
TopThickness	The thickness of the top side of the crosshair	float	4.0f
TopSpread	The spread of the top side of the crosshair	float	50.0f
TopOffset	The offset of the top side of the crosshair	float	10.0f
bDisplayTopShadow	Should a shadow be displayed behind the top side of the crosshair?	bool	true
TopShadowColor	The color of the shadow behind the top side of the crosshair	FLinearColor	FLinearColor::Black
TopShadowScale	The scale of the shadow behind the top side of the crosshair	float	1.15f
bDisplayBottom	Should the bottom side of the crosshair be displayed?	bool	true
BottomColor	The color of the bottom side of the crosshair	FLinearColor	FLinearColor::White
BottomLength	The length of the bottom side of the crosshair	float	20.0f
BottomThickness	The thickness of the bottom side of the crosshair	float	4.0f
BottomSpread	The spread of the bottom side of the crosshair	float	50.0f
BottomOffset	The offset of the bottom side of the crosshair	float	10.0f
bDisplayBottomShadow	Should a shadow be displayed behind the bottom side of the crosshair?	bool	true
BottomShadowColor	The color of the shadow behind the bottom side of the crosshair	FLinearColor	FLinearColor::Black
BottomShadowScale	The scale of the shadow behind the bottom side of the crosshair	float	1.15f
bDisplayLeft	Should the left side of the crosshair be displayed?	bool	true
LeftColor	The color of the left side of the crosshair	FLinearColor	FLinearColor::White
LeftLength	The length of the left side of the crosshair	float	20.0f
LeftThickness	The thickness of the left side of the crosshair	float	4.0f
LeftSpread	The spread of the left side of the crosshair	float	50.0f

LeftOffset	The offset of the left side of the crosshair	float	10.0f
bDisplayLeftShadow	Should a shadow be displayed behind the left side of the crosshair?	bool	true
LeftShadowColor	The color of the shadow behind the left side of the crosshair	FLinearColor	FLinearColor::Black
LeftShadowScale	The scale of the shadow behind the left side of the crosshair	float	1.15f
bDisplayRight	Should the right side of the crosshair be displayed?	bool	true
RightColor	The color of the right side of the crosshair	FLinearColor	
RightLength	The length of the right side of the crosshair	float	20.0f
RightThickness	The thickness of the right side of the crosshair	float	4.0f
RightSpread	The spread of the right side of the crosshair	float	50.0f
RightOffset	The offset of the right side of the crosshair	float	10.0f
bDisplayRightShadow	Should a shadow be displayed behind the right side of the crosshair?	bool	true
RightShadowColor	The color of the shadow behind the right side of the crosshair	FLinearColor	FLinearColor::Black
RightShadowScale	The scale of the shadow behind the right side of the crosshair	float	1.15f

12.2 Widget

12.2.1 Introduction

The dynamic crosshair widget

12.2.2 Dependencies

The `DynamicCrosshair` relies on other components of this plugin to work:

- Logger: Used to log useful information to help you debug any issues you might experience

12.2.3 Required Widgets

There is already a `DynamicCrosshair_Implementation` that you can use in your projects. But if you create your own instance of this widget, you need to add the following before you can compile:

Name	Description	Type
CenterImage	The center image of the crosshair	UImage*
CenterShadowImage	The center shadow image of the crosshair	UImage*
SidesContainer	The container used by all the sides	UCanvasPanel*
TopContainer	The top container of the crosshair	UCanvasPanel*
TopImage	The top image of the crosshair	UImage*
TopShadowImage	The top shadow image of the crosshair	UImage*
BottomContainer	The bottom container of the crosshair	UCanvasPanel*
BottomImage	The bottom image of the crosshair	UImage*
BottomShadowImage	The bottom shadow image of the crosshair	UImage*
LeftContainer	The left container of the crosshair	UCanvasPanel*
LeftImage	The left image of the crosshair	UImage*
LeftShadowImage	The left shadow image of the crosshair	UImage*
RightContainer	The right container of the crosshair	UCanvasPanel*
RightImage	The right image of the crosshair	UImage*
RightShadowImage	The right shadow image of the crosshair	UImage*

12.2.4 API Reference

Properties

Property	Description	Type	Default Value

13. Items

13.1 Allowed Collector

13.1.1 Introduction

The types of actor(s) that can collect an item

13.1.2 Values

Value	Description
AnyActor	Any actor can collect the item
AnyPawn	Any pawn can collect the item
AnyCharacter	Any character can collect the item
AnyUskCharacter	Any USK character can collect the item
PossessedPawn	Only the possessed pawn can collect the item
Custom	A custom array of actor types can collect the item

13.2 Collectable Item

13.2.1 Introduction

An item that can be collected by an actor

13.2.2 Dependencies

The `CollectableItem` relies on other components of this plugin to work:

- Logger: Used to log useful information to help you debug any issues you might experience

- Audio: Used to play sound effects either 2D or at a specified location

13.2.3 Collision

The item requires an actor to overlap with the item before it can be collected. Make sure you have some collider on the actor and that the intended collector can overlap with the item/collider

13.2.4 API Reference

Properties

Property	Description	Type	Default Value
DestroyOnCollected	Should the item be destroyed after it has been collected	bool	true
AllowedCollector	The type of actor that can collect the item	EAllowedCollector	EAllowedCollector::PossessedPawn
AllowedCollectorTypes	The array of actor types that can collect the item	TArray<TSubclassOf<AActor>>	
CollectedSoundEffects	An array of sound effects played when collecting the item	TArray<USoundBase*>	
CollectedParticleFx	The particle effects spawned when collecting the item	UNiagaraSystem*	`nullptr`
CollectedParticleFxSpawnOffset	The offset applied to the location of the collected particles when spawning	FVector	
bCompleteQuestPointAfterCollecting	Should the quest point be completed after the item is collected?	bool	true

Functions

Name	Description	Params	Return
CanCollectItem	Check if the item can be collected	**Collector (AActor*)** A pointer to the actor that is trying to collect the item	**bool** A boolean value indicating if the item can be collected
CollectItem	Collect the item	**Collector (AActor*)** A pointer to the actor that collected the item	
OnItemCollected	Called after the item is collected	**Collector (AActor*)** A pointer to the actor that collected the item	

13.2.5 Blueprint Usage

You can use the `CollectableItem` using Blueprints by adding one of the following nodes:

• Ultimate Starter Kit > Item > Can Collect Item

• Ultimate Starter Kit > Item > Collect Item

• Ultimate Starter Kit > Item > On Item Collected

13.2.6 C++ Usage

Before you can use the plugin, you first need to enable the plugin in your `Build.cs` file:

```
PublicDependencyModuleNames.Add("USK");
```

The `CollectableItem` can now be used in any of your C++ files:

```
#include "USK/Items/CollectableItem.h"

void ATestActor::Test()
{
    // CollectableItem is a pointer to the ACollectableItem
    bool CanCollectItemValue = CollectableItem->CanCollectItem(Collector);
    CollectableItem->CollectItem(Collector);
    CollectableItem->OnItemCollected(Collector);
}
```

13.3 Attract Component

13.3.1 Introduction

A component that can be used to attract the owning object to the player

13.3.2 Dependencies

The `AttractComponent` relies on other components of this plugin to work:

• Logger: Used to log useful information to help you debug any issues you might experience

13.3.3 API Reference

Properties

Property	Description	Type	Default Value
AttractSpeed	The speed at which the object is attracted to the player	float	5.0f
CollectDistance	The distance at which the object is collected by the player	float	75.0f

14. Interact System

14.1 Interact Trigger

14.1.1 Introduction

A trigger that can be used to interact with an object

14.1.2 Dependencies

The `InteractTrigger` relies on other components of this plugin to work:

• Logger: Used to log useful information to help you debug any issues you might experience

14.1.3 API Reference

Properties

Property	Description	Type	Default Value
bRequireCurrency	Is this a paid item that should be bought using currency?	bool	false
CurrencyRequirements	The currency requirements before the object can be interacted with	TMap<FName, int>	
InteractWidgetClass	The class of the interact widget	TSubclassOf<UInteractWidget>	
BeforeText	The text displayed before the input indicator	FText	
AfterText	The text displayed after the input indicator	FText	

Functions

Name	Description	Params	Return
CanInteract	Check if the actor can interact with the object	**Actor (AActor*)** The actor trying to interact with the object	**bool** A boolean value indicating if the actor can interact with the object
OnInteracted	Called when the actor interacts with the object	**Actor (AActor*)** The actor that interacted with the object	
ReceiveOnInteracted	Called when the actor interacts with the object	**Actor (AActor*)** The actor that interacted with the object	

14.1.4 Blueprint Usage

You can use the `InteractTrigger` using Blueprints by adding one of the following nodes:

• Ultimate Starter Kit > Interact Trigger > Can Interact

• Ultimate Starter Kit > Interact Trigger > On Interacted

• Ultimate Starter Kit > Interact Trigger > OnInteracted

14.1.5 C++ Usage

Before you can use the plugin, you first need to enable the plugin in your `Build.cs` file:

```
PublicDependencyModuleNames.Add("USK");
```

The `InteractTrigger` can now be used in any of your C++ files:

```cpp
#include "USK/Components/InteractTrigger.h"

void ATestActor::Test()
{
    // InteractTrigger is a pointer to the UInteractTrigger
    bool CanInteractValue = InteractTrigger->CanInteract(Actor);
    InteractTrigger->OnInteracted(Actor);
    InteractTrigger->ReceiveOnInteracted(Actor);
}
```

14.2 Interact Widget

14.2.1 Introduction

The widget displayed when the player can interact with an object

14.2.2 Dependencies

The `InteractWidget` relies on other components of this plugin to work:

- Logger: Used to log useful information to help you debug any issues you might experience

14.2.3 Required Widgets

You need to add the following before you can compile the `InteractWidget` widget:

Name	Description	Type
BeforeTextBlock	The text block used to display the text before the input indicator	UTextBlock*
AfterTextBlock	The text block used to display the text after the input indicator	UTextBlock*

14.2.4 Optional Animations

You can add the following widgets to enable extra functionality:

Name	Description
ShowAnimation	The animation played when the widget is shown

14.2.5 API Reference

Properties

Property	Description	Type	Default Value
ShowAnimation	The animation played when the widget is shown	UWidgetAnimation*	nullptr

Functions

Name	Description	Params	Return
Show	Show the interact widget	**BeforeText (FText)** The text displayed before the input indicator **AfterText (FText)** The text displayed after the input indicator	
Hide	Hide the interact widget		

14.2.6 Blueprint Usage

You can use the `InteractWidget` using Blueprints by adding one of the following nodes:

- Ultimate Starter Kit > UI > Show
- Ultimate Starter Kit > UI > Hide

14.2.7 C++ Usage

Before you can use the plugin, you first need to enable the plugin in your `Build.cs` file:

```
PublicDependencyModuleNames.Add("USK");
```

The `InteractWidget` can now be used in any of your C++ files:

```cpp
#include "USK/Widgets/InteractWidget.h"

void ATestActor::Test()
{
    // InteractWidget is a pointer to the UInteractWidget
    InteractWidget->Show(BeforeText, AfterText);
    InteractWidget->Hide();
}
```

15. Quest System

15.1 Quest

15.1.1 Introduction

The information about a specific quest

15.1.2 API Reference

Properties

Property	Description	Type	Default Value
Name	The name of the quest	FText	
Id	The ID of the quest	FGuid	FGuid::NewGui
Points	The points of the quest	TArray<FText>	

15.2 Quest Component

15.2.1 Introduction

Component used to update the actor based on the current quest and point

15.2.2 Dependencies

The `QuestComponent` relies on other components of this plugin to work:

- Logger: Used to log useful information to help you debug any issues you might experience

15.2.3 API Reference

Properties

Property	Description	Type	Default Value
Quest	The quest associated with the component	UQuest*	`nullptr`
RequiredPoint	The point of the quest associated with the component	int	
bHideIfQuestNotActive	Should the actor be hidden if the quest is not active?	bool	true
bHideAfterPointCompleted	Should the actor be hidden if the quest point has been completed?	bool	true

15.3 Quest Manager

15.3.1 Introduction

Actor responsible for managing quests

15.3.2 API Reference

Properties

Property	Description	Type	Default Value
QuestWidgetClass	The widget class used to display the current quest	TSubclassOf<UQuestWidget>	

Events

Name	Description	Params
OnQuestCompleted	Event used to notify other classes when a quest is completed	**Quest (UQuest*)** The quest that was completed
OnQuestUpdated	Event used to notify other classes when a quest is updated	**CurrentQuest (UQuest*)** The current active quest **CurrentPoint (UQuest*)** The current point of the active quest

Functions

Name	Description	Params	Return
StartQuest	Start the specified quest	**Quest (UQuest*)** The quest to start	
OnQuestPointCompleted	Complete the current quest point		
GetCurrentQuest	Get the current quest		**UQuest*** The current quest
GetCurrentQuestPo	Get the current quest point		**int** The current quest point
IsQuestActive	Check if the specified quest is active	**Quest (UQuest*)** The quest to check	**bool** A boolean value indicating if the quest is active

15.3.3 Blueprint Usage

You can use the `QuestManager` using Blueprints by adding one of the following nodes:

- Ultimate Starter Kit > Quests > Start Quest
- Ultimate Starter Kit > Quests > On Quest Point Completed
- Ultimate Starter Kit > Quests > Get Current Quest
- Ultimate Starter Kit > Quests > Get Current Quest Po
- Ultimate Starter Kit > Quests > Is Quest Active

15.3.4 C++ Usage

Before you can use the plugin, you first need to enable the plugin in your `Build.cs` file:

```
PublicDependencyModuleNames.Add("USK");
```

The `QuestManager` can now be used in any of your C++ files:

```cpp
#include "USK/Quests/QuestManager.h"

void ATestActor::Test()
{
    // QuestManager is a pointer to the AQuestManager
    QuestManager->StartQuest(Quest);
    QuestManager->OnQuestPointCompleted();
    UQuest* CurrentQuest = QuestManager->GetCurrentQuest();
    int CurrentQuestPo = QuestManager->GetCurrentQuestPo();
    bool IsQuestActiveValue = QuestManager->IsQuestActive(Quest);
}
```

15.4 Quest Widget

15.4.1 Introduction

The widget displayed for the current quest point

15.4.2 Dependencies

The `QuestWidget` relies on other components of this plugin to work:

- Logger: Used to log useful information to help you debug any issues you might experience

15.4.3 Required Widgets

You need to add the following before you can compile the `QuestWidget` widget:

Name	Description	Type
QuestTitleText	The text block used to quest title	UTextBlock*
QuestDetailsText	The text block used to quest details	UTextBlock*

15.4.4 Optional Animations

You can add the following widgets to enable extra functionality:

Name	Description
ShowAnimation	The animation played when the widget is shown

15.4.5 API Reference

Properties

Property	Description	Type	Default Value
ShowAnimation	The animation played when the widget is shown	UWidgetAnimation*	nullptr

Functions

Name	Description	Params	Return
Show	Show the quest widget	**Quest (UQuest*)** The current quest **Point (int)** The current quest point	
Hide	Hide the quest widget		

15.4.6 Blueprint Usage

You can use the `QuestWidget` using Blueprints by adding one of the following nodes:

- Ultimate Starter Kit > UI > Show
- Ultimate Starter Kit > UI > Hide

15.4.7 C++ Usage

Before you can use the plugin, you first need to enable the plugin in your `Build.cs` file:

```
PublicDependencyModuleNames.Add("USK");
```

The `QuestWidget` can now be used in any of your C++ files:

```cpp
#include "USK/Widgets/QuestWidget.h"

void ATestActor::Test()
{
    // QuestWidget is a pointer to the UQuestWidget
    QuestWidget->Show(Quest, Point);
    QuestWidget->Hide();
}
```

16. UI & Widgets

16.1 Collectable Item Icon

16.1.1 Introduction

A widget used to display the collection state of an item by showing/hiding an image

16.1.2 Dependencies

The `CollectableItemIcon` relies on other components of this plugin to work:

• Logger: Used to log useful information to help you debug any issues you might experience

16.1.3 Required Widgets

You need to add the following before you can compile the `CollectableItemIcon` widget:

Name	Description	Type
Icon	The Icon displayed if the item is collected	UImage*

16.1.4 API Reference

Properties

Property	Description	Type	Default Value
RequiredValue	The required value before the item is considered collected	float	

Functions

Name	Description	Params	Return
UpdateState	Update the collected state (and visibility) of the icon	**IsCollected (bool)** Is the item collected?	
UpdateValue	Update the value of the item and adjust the collected state if necessary	**Value (float)** The current value of the item	
MonitorTrackableValue	Monitor the trackable data and automatically update the icon state whenever the value changes	**TrackableDataComponent (UTrackableDataComponent*)** A reference to the TrackableDataComponent **DataName (FName)** The name of the data item to monitor	

16.1.5 Blueprint Usage

You can use the `CollectableItemIcon` using Blueprints by adding one of the following nodes:

• Ultimate Starter Kit > UI > Update State

• Ultimate Starter Kit > UI > Update Value

• Ultimate Starter Kit > UI > Monitor Trackable Value

16.1.6 C++ Usage

Before you can use the plugin, you first need to enable the plugin in your `Build.cs` file:

```
PublicDependencyModuleNames.Add("USK");
```

The `CollectableItemIcon` can now be used in any of your C++ files:

```cpp
#include "USK/Widgets/CollectableItemIcon.h"

void ATestActor::Test()
{
    // CollectableItemIcon is a pointer to the UCollectableItemIcon
    CollectableItemIcon->UpdateState(IsCollected);
    CollectableItemIcon->UpdateValue(Value);
    CollectableItemIcon->MonitorTrackableValue(TrackableDataComponent, DataName);
}
```

16.2 Credits

16.2.1 Credits Entry

Introduction

An single credits entry displayed by the credits widget

Properties

Property	Description	Type	Default Value
Title	The title for the credits entry	FText	
Text	The text for the credits entry	FText	
Duration	The duration of the credits entry	float	5.0f
HorizontalAlignment	The horizontal alignment applied to the credits widget when displaying this entry	TEnumAsByte<EHorizontalAlignment>	HAlign_Center
VerticalAlignment	The vertical alignment applied to the credits widget when displaying this entry	TEnumAsByte<EVerticalAlignment>	VAlign_Center

16.2.2 Credits Widget

Introduction

Widget used to display multiple animated credits entries using different alignment options and durations

Dependencies

The `CreditsWidget` relies on other components of this plugin to work:

• Logger: Used to log useful information to help you debug any issues you might experience

Required Widgets

You need to add the following before you can compile the `CreditsWidget` widget:

Name	Description	Type
Root	The root container of the widget	UPanelWidget*
Container	The container used to display the credits entries	UPanelWidget*
Text	The text block used to display the text of the credits entry	UTextBlock*

Optional Widgets

You can add the following widgets to enable extra functionality:

Name	Description	Type
Title	The text block used to display the title of the credits entry	UTextBlock*

Optional Animations

You can add the following widgets to enable extra functionality:

Name	Description
ShowAnimation	The animation played each time a new entry is shown
HideAnimation	The animation played at the end of each entry

API Reference

PROPERTIES

Property	Description	Type	Default Value
ShowAnimation	The animation played each time a new entry is shown	UWidgetAnimation*	`nullptr`
HideAnimation	The animation played at the end of each entry	UWidgetAnimation*	`nullptr`
AutoStart	Should the credits automatically be started when the widget is constructed?	bool	true
RemoveOnCompletion	Should the widget automatically be removed from the viewport when the credits are finished?	bool	true
StartDelay	The delay in seconds before the first credits entry is shown after starting	float	1.0f
DelayBetweenEntries	The delay in seconds between the previous hide animation and the next show animation	float	3.0f
Credits	The array of credits entries to be displayed	TArray<FCreditsEntry>	

EVENTS

Name	Description	Params
OnCreditsFinished	Event used to notify other classes when the credits are finished	
OnCreditsNextEntryStarted	Event used to notify other classes every time a next entry is started	**Name (int)** The index of the entry that was started

FUNCTIONS

Name	Description	Params	Return
Start	Start showing the credits		

Blueprint Usage

You can use the `CreditsWidget` using Blueprints by adding one of the following nodes:

• Ultimate Starter Kit > UI > Start

C++ Usage

Before you can use the plugin, you first need to enable the plugin in your `Build.cs` file:

```
PublicDependencyModuleNames.Add("USK");
```

The `CreditsWidget` can now be used in any of your C++ files:

```
#include "USK/Widgets/CreditsWidget.h"

void ATestActor::Test()
{
    // CreditsWidget is a pointer to the UCreditsWidget
    CreditsWidget->Start();
}
```

16.3 FPS Counter

16.3.1 Introduction

A widget used to display the current framerate

16.3.2 Dependencies

The `FpsCounter` relies on other components of this plugin to work:

- Logger: Used to log useful information to help you debug any issues you might experience
- Game Instance: Used to monitor for input device changes and handle saving/loading game data

16.3.3 Required Widgets

There is already a `FpsCounter_Implementation` that you can use in your projects. But if you create your own instance of this widget, you need to add the following before you can compile:

Name	Description	Type
FramerateText	The text block used to display the framerate	UTextBlock*

16.3.4 API Reference

Properties

Property	Description	Type	Default Value
UpdateDelay	The delay in seconds between each update	float	0.125f
HighFramerate	A framerate that is considered high and will use the high color	int	60
MediumFramerate	A framerate that is considered medium and will use the medium color	int	30
HighColor	The color used to display high framerates	FLinearColor	FLinearColor::Green
MediumColor	The color used to display medium framerates	FLinearColor	FLinearColor::Yellow
LowColor	The color used to display low framerates	FLinearColor	FLinearColor::Red

Functions

Name	Description	Params	Return
UpdateVisibility	Update the visibility of the widget	**IsVisible (bool)** Is the widget visible?	

16.3.5 Blueprint Usage

You can use the `FpsCounter` using Blueprints by adding one of the following nodes:

- Ultimate Starter Kit > UI > Update Visibility

16.3.6 C++ Usage

Before you can use the plugin, you first need to enable the plugin in your `Build.cs` file:

```
PublicDependencyModuleNames.Add("USK");
```

The `FpsCounter` can now be used in any of your C++ files:

```cpp
#include "USK/Widgets/FpsCounter.h"

void ATestActor::Test()
{
    // FpsCounter is a pointer to the UFpsCounter
    FpsCounter->UpdateVisibility(IsVisible);
}
```

16.4 Input Indicator

16.4.1 Introduction

A widget used to display input indicators based on the current input device and input action

16.4.2 Dependencies

The `InputIndicator` relies on other components of this plugin to work:

- Logger: Used to log useful information to help you debug any issues you might experience

- Game Instance: Used to monitor for input device changes and handle saving/loading game data

16.4.3 Required Widgets

There is already a `InputIndicator_Implementation` that you can use in your projects. But if you create your own instance of this widget, you need to add the following before you can compile:

Name	Description	Type
Container	The container used to display multiple images	UHorizontalBox*

16.4.4 API Reference

Properties

Property	Description	Type	Default Value
InputIndicatorIconClass	The input indicator icon class	TSubclassOf<UInputIndicatorIcon>	
Action	The input action displayed by widget	UInputAction*	`nullptr`
Size	The size of the image	float	50.0f
Amount	The amount of images to display for the input action	int	1

Functions

Name	Description	Params	Return
UpdateAction	Update the input action displayed by the widget	**NewAction (UInputAction*)** The new action **NewAmount (int)** The new amount of images to display	

16.4.5 Blueprint Usage

You can use the `InputIndicator` using Blueprints by adding one of the following nodes:

- Ultimate Starter Kit > UI > Update Action

16.4.6 C++ Usage

Before you can use the plugin, you first need to enable the plugin in your `Build.cs` file:

```
PublicDependencyModuleNames.Add("USK");
```

The `InputIndicator` can now be used in any of your C++ files:

```cpp
#include "USK/Widgets/InputIndicator.h"

void ATestActor::Test()
{
    // InputIndicator is a pointer to the UInputIndicator
    InputIndicator->UpdateAction(NewAction, NewAmount);
}
```

```cpp
#include "USK/Widgets/InputIndicator.h"
```

16.5 Input Indicator Icon

16.5.1 Introduction

A widget used to display a single input indicator image

16.5.2 Dependencies

The `InputIndicatorIcon` relies on other components of this plugin to work:

• Logger: Used to log useful information to help you debug any issues you might experience

16.5.3 Required Widgets

There is already a `InputIndicatorIcon_Implementation` that you can use in your projects. But if you create your own instance of this widget, you need to add the following before you can compile:

Name	Description	Type
Container	The size box container used to resize the image	USizeBox*
Image	The image used to display the input indicator	UImage*

16.5.4 API Reference

Properties

Property	Description	Type	Default Value

Functions

Name	Description	Params	Return
UpdateIcon	Update the icon	**Size (float)** The size of the image **Icon (UTexture2D*)** The new icon	

16.5.5 Blueprint Usage

You can use the `InputIndicatorIcon` using Blueprints by adding one of the following nodes:

• Ultimate Starter Kit > UI > Update Icon

16.5.6 C++ Usage

Before you can use the plugin, you first need to enable the plugin in your `Build.cs` file:

```
PublicDependencyModuleNames.Add("USK");
```

The `InputIndicatorIcon` can now be used in any of your C++ files:

```
#include "USK/Widgets/InputIndicatorIcon.h"

void ATestActor::Test()
{
    // InputIndicatorIcon is a pointer to the UInputIndicatorIcon
```

```
    InputIndicatorIcon->UpdateIcon(Size, Icon);
}
```

16.6 Menu

16.6.1 Introduction

A widget used to display menu items and handle navigation between the items

16.6.2 Dependencies

The `Menu` relies on other components of this plugin to work:

- Logger: Used to log useful information to help you debug any issues you might experience

- Game Instance: Used to monitor for input device changes and handle saving/loading game data

- Audio: Used to play sound effects either 2D or at a specified location

16.6.3 Optional Widgets

You can add the following widgets to enable extra functionality:

Name	Description	Type
ScrollContainer	Scroll container used for large menus with many items	UScrollBox*
Container	The container used to display the menu items	UPanelWidget*

16.6.4 API Reference

Properties

Property	Description	Type	Default Value
AddInputBindingOnLoad	Should the input binding automatically be added as soon as the widget is loaded?	bool	false
PauseGameWhileVisible	Should the game automatically be paused/resumed based on the visibility of the menu?	bool	false
DisableWhilePaused	Should the menu be disabled while the game is paused?	bool	false
SelectedSFX	The sound effect played when a menu item is selected	USoundBase*	`nullptr`
BackSFX	The sound effect played when trying to go back to a previous menu or closing the menu through the back button	USoundBase*	`nullptr`
InputMappingContext	The input mapping context used to navigate the menu	UInputMappingContext*	`nullptr`
MenuUpInputAction	The input action used to navigate up	UInputAction*	`nullptr`
MenuDownInputAction	The input action used to navigate down	UInputAction*	`nullptr`
MenuLeftInputAction	The input action used to navigate left	UInputAction*	`nullptr`
MenuRightInputAction	The input action used to navigate right	UInputAction*	`nullptr`
MenuSelectInputAction	The input action used to select a menu item	UInputAction*	`nullptr`
MenuBackInputAction	The input action used to go back to a previous menu or close the menu	UInputAction*	`nullptr`

Events

Name	Description	Params
OnBackEvent	Event used to handle the back/close action of the menu	

Functions

Name	Description	Params	Return
OnMenuUp	Navigate up or increase the value		
OnMenuUpHold	Increase the value while holding the menu up key		
OnMenuDown	Navigate down or decrease the value		
OnMenuDownHold	Decrease the value while holding the menu down key		
OnMenuLeft	Navigate left or decrease the value		
OnMenuLeftHold	Decrease the value while holding the menu left key		
OnMenuRight	Navigate right or increase the value		
OnMenuRightHold	Increase the value while holding the menu right key		
OnMenuSelected	Select the current menu item		
OnMenuBack	Go back to a previous menu or close the menu		
RequestHighlight	Request to highlight a specific menu item	**MenuItem (UMenuItem*)** The menu item to highlight	
RemoveHighlight	Request to remove the highlighted state from a specific menu item	**MenuItem (UMenuItem*)** The menu item to remove the highlighted state from	
AddMenuItem	Add a menu item to the container	**MenuItem (UMenuItem*)** The menu item to add	

16.6.5 Blueprint Usage

You can use the `Menu` using Blueprints by adding one of the following nodes:

- Ultimate Starter Kit > UI > On Menu Up
- Ultimate Starter Kit > UI > On Menu Up Hold
- Ultimate Starter Kit > UI > On Menu Down
- Ultimate Starter Kit > UI > On Menu Down Hold
- Ultimate Starter Kit > UI > On Menu Left
- Ultimate Starter Kit > UI > On Menu Left Hold
- Ultimate Starter Kit > UI > On Menu Right
- Ultimate Starter Kit > UI > On Menu Right Hold
- Ultimate Starter Kit > UI > On Menu Selected
- Ultimate Starter Kit > UI > On Menu Back
- Ultimate Starter Kit > UI > Request Highlight
- Ultimate Starter Kit > UI > Remove Highlight
- Ultimate Starter Kit > UI > Add Menu Item

16.6.6 C++ Usage

Before you can use the plugin, you first need to enable the plugin in your `Build.cs` file:

```
PublicDependencyModuleNames.Add("USK");
```

The `Menu` can now be used in any of your C++ files:

```cpp
#include "USK/Widgets/Menu.h"

void ATestActor::Test()
{
    // Menu is a pointer to the UMenu
    Menu->OnMenuUp();
    Menu->OnMenuUpHold();
    Menu->OnMenuDown();
    Menu->OnMenuDownHold();
    Menu->OnMenuLeft();
    Menu->OnMenuLeftHold();
    Menu->OnMenuRight();
    Menu->OnMenuRightHold();
    Menu->OnMenuSelected();
    Menu->OnMenuBack();
    Menu->RequestHighlight(MenuItem);
    Menu->RemoveHighlight(MenuItem);
    Menu->AddMenuItem(MenuItem);
}
```

16.7 Menu Item

16.7.1 Navigation

Introduction

All the supported menu navigation types

Values

Value	Description
Disabled	No navigation allowed
HighlightItem	Highlight a different menu item
IncreaseDecreaseValue	Increase or decrease the value

16.7.2 Value Update Method

Introduction

The method used to update the value of a menu item

Values

Value	Description
SinglePress	The value is only updated when the button is pressed
Hold	The value is updated while the button is held down

16.7.3 Menu Item Widget

Introduction

A widget used to display a title, text and value in the form of a menu item

Dependencies

The `MenuItem` relies on other components of this plugin to work:

• Logger: Used to log useful information to help you debug any issues you might experience

• Game Instance: Used to monitor for input device changes and handle saving/loading game data

• Audio: Used to play sound effects either 2D or at a specified location

Optional Widgets

You can add the following widgets to enable extra functionality:

Name	Description	Type
Title	The TextBlock used to display the title of the menu item	UTextBlock*
NormalText	The TextBlock used to display the text of the menu item while not highlighted	UTextBlock*
HighlightedText	The TextBlock used to display the text of the menu item while highlighted	UTextBlock*
ValueText	The TextBlock used to display the current value of the menu item	UTextBlock*
HighlightedValueText	The TextBlock used to display the current value of the menu item while highlighted	UTextBlock*
SelectButton	The button used to select the menu item	UButton*
ValueSlider	The slider used to display and update the current value of the menu item	USlider*
IncreaseValueButton	The button used to increase the value of the menu item	UButton*
DecreaseValueButton	The button used to decrease the value of the menu item	UButton*
BorderLeft	The border displayed on the left of the menu item	UImage*
BorderRight	The border displayed on the right of the menu item	UImage*
BorderBackground	The background border display in the menu item	UImage*
ButtonLeft	The button displayed on the left of the menu item	UImage*
ButtonRight	The button displayed on the right of the menu item	UImage*
ButtonBackground	The background button display in the menu item	UImage*
InputIndicator	The background button display in the menu item	UImage*

Optional Animations

You can add the following widgets to enable extra functionality:

Name	Description
HighlightedAnimation	The animation played when the menu item is highlighted

API Reference

PROPERTIES

API Reference

PROPERTIES

Property	Description	Type	Default Value
HighlightedAnimation	The animation played when the menu item is highlighted	UWidgetAnimation*	`nullptr`
FocusByDefault	Should the menu item be focused by default?	bool	false
HideOnConsoles	Should the menu item be hidden on consoles?	bool	false
TitleText	The title text displayed in the menu item	FText	
MenuItemText	The text displayed in the menu item	FText	
HighlightedSFX	The sound effect played when the menu item is highlighted	USoundBase*	`nullptr`
BorderNormalColor	The color of the border when not highlighted	FLinearColor	
BorderHighlightedColor	The color of the border when highlighted	FLinearColor	
BorderNormalImage	The image of the border when not highlighted	UTexture2D*	`nullptr`
BorderHighlightedImage	The image of the border when highlighted	UTexture2D*	`nullptr`
BorderLeftNormalImage	The image of the left border when not highlighted	UTexture2D*	`nullptr`
BorderLeftHighlightedImage	The image of the left border when highlighted	UTexture2D*	`nullptr`
BorderRightNormalImage	The image of the right border	UTexture2D*	`nullptr`

	when not highlighted		
BorderRightHighlightedImage	The image of the right border when highlighted	UTexture2D*	`nullptr`
BackgroundNormalColor	The color of the button when not highlighted	FLinearColor	
BackgroundHighlightedColor	The color of the button when highlighted	FLinearColor	
BackgroundNormalImage	The image of the button when not highlighted	UTexture2D*	`nullptr`
BackgroundHighlightedImage	The image of the button when highlighted	UTexture2D*	`nullptr`
BackgroundLeftNormalImage	The image of the left button when not highlighted	UTexture2D*	`nullptr`
BackgroundLeftHighlightedImage	The image of the left button when highlighted	UTexture2D*	`nullptr`
BackgroundRightNormalImage	The image of the right button when not highlighted	UTexture2D*	`nullptr`
BackgroundRightHighlightedImage	The image of the right button when highlighted	UTexture2D*	`nullptr`
ValueUpdateMethod	The method used to update the value of the menu item	EMenuItemValueUpdateMethod	EMenuItemValueUpdateMethod::SinglePress
IncrementSinglePress	The increment used when updating the value when the key is pressed	float	1.0f
IncrementHold	The increment used when updating the value when the	float	0.15f

	key is held down		
SettingsItemType	The type of setting item managed by this menu item (changing this will overwrite other settings)	ESettingsItemType	ESettingsItemType::None
AutoSaveSettingsOnValueChanged	Should the settings managed by this menu item automatically be saved when the value is changed?	bool	true
AutoSaveSettingsOnHighlightRemoved	Should the settings managed by this menu item automatically be saved when the highlight state is removed?	bool	true
AutoSaveSettingsOnSelected	Should the settings managed by this menu item automatically be saved when the menu item is selected?	bool	true
InputDevice	The input device associated with the action to rebind	EInputDevice	EInputDevice::Unknown
InputMappingContext	The input mapping context containing the action to rebind	UInputMappingContext*	`nullptr`
InputAction	The input action to rebind	UInputAction*	`nullptr`
MappableName	The player mappable name for the action to rebind	FName	
ShowValueSlider	Should the value slider be	bool	false

	shown for this menu item?		
ShowValueButtons	Should the increase/ decrease value buttons be shown for this menu item?	bool	false
ValueMapping	A mapping of possible values to text	TMap<int, FText>	
DefaultValue	The default value of the menu item	int	100
MinValue	The minimum value of the menu item	int	0
MaxValue	The maximum value of the menu item	int	100
AllowSelection	Can the menu item be selected?	bool	true
VerticalNavigation	The type of navigation used by the menu item when pressing the up or down key	EMenuNavigation	EMenuNavigation::HighlightItem
MenuItemUp	The menu item highlighted when the up key is pressed	UMenuItem*	`nullptr`
MenuItemDown	The menu item highlighted when the down key is pressed	UMenuItem*	`nullptr`
HorizontalNavigation	The type of navigation used by the menu item when pressing the left or right key	EMenuNavigation	EMenuNavigation::HighlightItem
MenuItemLeft	The menu item highlighted when the left key is pressed	UMenuItem*	`nullptr`
MenuItemRight	The menu item highlighted when the right key is pressed	UMenuItem*	`nullptr`

Menu	A reference to the menu that contains this menu item	UMenu*	`nullptr`

EVENTS

Name	Description	Params
OnSelected	Event used to notify other classes that the menu item was selected	
OnSelectedInContainer	Event used to notify other classes that a specific menu item in the container was selected	**Index (int)** The index of the menu item that was selected
OnHighlighted	Event used to notify other classes that the menu item was highlighted	
OnHighlightRemoved	Event used to notify other classes that the menu item's highlighted state was removed	
OnValueChanged	Event used to notify other classes that the menu item's value was updated	**Value (int)** The new value of the menu item

FUNCTIONS

Name	Description	Params	Return
SetText	Set the text displayed in the menu item	**Text (FText&)** The new text displayed in the menu item	
SetTitle	Set the title displayed in the menu item	**Text (FText&)** The new title displayed in the menu item	
SetHighlightedState	Set the highlighted state of the menu item	**IsHighlighted (bool)** Is the menu item highlighted? **PlayHighlightedAnimation (bool)** Should the highlighted animation be played? **PlayHighlightedSound (bool)** Should the highlighted sound effect be played?	
IsHighlighted	Check if the menu item is highlighted		**bool** A boolean value indicating if the menu item is highlighted
GetValue	Get the current value of the menu item		**int** The current value of the menu item
UpdateValue	Update the value of the menu item	**Increment (float)** The amount added to the current value of the menu item	
SelectItem	Select the menu item		
SaveSettings	Save the settings managed by this menu item		
ApplySettings	Apply the settings managed by this menu item		
GetInputActionKey	Get the key used by the specified input action		**FKey** The key used by the specified input action
OnMenuBack	Called when trying to go back in the menu		**bool** A boolean value indicating if the back event was handled
AnyKeyPressed	Called after any key is pressed by the player (used to remap controls)	**Key (FKey)** The key pressed by the player	
ApplyKeyBinding	Apply the key binding for the input action		
IsWaitingForKeyPress	Is the menu item waiting for a key press?		**bool** A boolean value indicating if the menu item is waiting for a key press

Blueprint Usage

You can use the `MenuItem` using Blueprints by adding one of the following nodes:

- Ultimate Starter Kit > UI > Set Text

- Ultimate Starter Kit > UI > Set Title

- Ultimate Starter Kit > UI > Set Highlighted State

- Ultimate Starter Kit > UI > Is Highlighted

- Ultimate Starter Kit > UI > Get Value

- Ultimate Starter Kit > UI > Update Value

- Ultimate Starter Kit > UI > Select Item

- Ultimate Starter Kit > UI > Save Settings

- Ultimate Starter Kit > UI > Apply Settings

- Ultimate Starter Kit > UI > Get Input Action Key

- Ultimate Starter Kit > UI > On Menu Back

- Ultimate Starter Kit > UI > Any Key Pressed

- Ultimate Starter Kit > UI > Apply Key Binding

- Ultimate Starter Kit > UI > Is Waiting For Key Press

C++ Usage

Before you can use the plugin, you first need to enable the plugin in your `Build.cs` file:

```
PublicDependencyModuleNames.Add("USK");
```

The `MenuItem` can now be used in any of your C++ files:

```
#include "USK/Widgets/MenuItem.h"

void ATestActor::Test()
{
    // MenuItem is a pointer to the UMenuItem
    MenuItem->SetText(Text);
    MenuItem->SetTitle(Text);
    MenuItem->SetHighlightedState(IsHighlighted, PlayHighlightedAnimation, PlayHighlightedSound);
    bool IsHighlightedValue = MenuItem->IsHighlighted();
    int Value = MenuItem->GetValue();
    MenuItem->UpdateValue(Increment);
    MenuItem->SelectItem();
    MenuItem->SaveSettings();
    MenuItem->ApplySettings();
    FKey InputActionKey = MenuItem->GetInputActionKey();
    bool OnMenuBackValue = MenuItem->OnMenuBack();
    MenuItem->AnyKeyPressed(Key);
    MenuItem->ApplyKeyBinding();
    bool IsWaitingForKeyPressValue = MenuItem->IsWaitingForKeyPress();
}
```

16.8 Message Popups

16.8.1 Message Popup Data

Introduction

The data used to display a message popup

Properties

Property	Description	Type	Default Value
Title	The title displayed in the message popup	FText	
Message	The text displayed in the message popup	FText	
bShowPositiveButton	Should the positive button be shown?	bool	false
PositiveButtonText	The text displayed on the positive button	FText	
bShowNegativeButton	Should the negative button be shown?	bool	false
NegativeButtonText	The text displayed on the negative button	FText	
bShowNeutralButton	Should the neutral button be shown?	bool	false
NeutralButtonText	The text displayed on the neutral button	FText	

16.8.2 Message Popup Widget

Introduction

The widget used to display a message popup

Dependencies

The `MessagePopupWidget` relies on other components of this plugin to work:

• Logger: Used to log useful information to help you debug any issues you might experience

Required Widgets

You need to add the following before you can compile the `MessagePopupWidget` widget:

Name	Description	Type
TitleText	The text block used to display the title	UTextBlock*
MessageText	The text block used to display the message	UTextBlock*
ButtonMenu	The menu used to display the buttons	UMenu*

Optional Animations

You can add the following widgets to enable extra functionality:

Name	Description
ShowAnimation	The animation used to show the popup
HideAnimation	The animation used to hide the popup

API Reference

PROPERTIES

Property	Description	Type	Default Value
ShowAnimation	The animation used to show the popup	UWidgetAnimation*	`nullptr`
HideAnimation	The animation used to hide the popup	UWidgetAnimation*	`nullptr`
ButtonMenuItemClass	The button menu item class used when creating the buttons	TSubclassOf<UMenuItem>	

EVENTS

Name	Description	Params
OnShown	Event used to notify other classes that the popup was shown	
OnHidden	Event used to notify other classes that the popup was hidden	
OnPositiveButtonSelected	Event used to notify other classes that the positive button was selected	
OnNegativeButtonSelected	Event used to notify other classes that the negative button was selected	
OnNeutralButtonSelected	Event used to notify other classes that the neutral button was selected	

FUNCTIONS

Name	Description	Params	Return
Show	Show the message popup	**Data (FMessagePopupData)** The data used to display the popup	
Hide	Hide the message popup		

Blueprint Usage

You can use the `MessagePopupWidget` using Blueprints by adding one of the following nodes:

• Ultimate Starter Kit > UI > Show

• Ultimate Starter Kit > UI > Hide

C++ Usage

Before you can use the plugin, you first need to enable the plugin in your `Build.cs` file:

```
PublicDependencyModuleNames.Add("USK");
```

The `MessagePopupWidget` can now be used in any of your C++ files:

```
#include "USK/Widgets/MessagePopupWidget.h"

void ATestActor::Test()
{
    // MessagePopupWidget is a pointer to the UMessagePopupWidget
    MessagePopupWidget->Show(Data);
    MessagePopupWidget->Hide();
}
```

17. Settings

17.1 Data

17.1.1 Introduction

The settings data that is saved/loaded

17.1.2 API Reference

Properties

Property	Description	Type	Default Value
GameplayDifficultyModified	Was the gameplay difficulty setting modified?	bool	false
GameplayDifficulty	The gameplay difficulty setting	int	
AudioMasterModified	Was the master audio volume modified?	bool	false
AudioMaster	The master audio volume	float	
AudioMusicModified	Was the music volume modified?	bool	false
AudioMusic	The music volume	float	
AudioEffectsModified	Was the effects volume modified?	bool	false
AudioEffects	The effects volume	float	
AudioUiModified	Was the UI volume modified?	bool	false
AudioUi	The UI volume	float	
AudioVoiceModified	Was the voice volume modified?	bool	false
AudioVoice	The voice volume	float	
GraphicsResolutionX	The X value of the saved resolution	int	
GraphicsResolutionY	The Y value of the saved resolution	int	
GraphicsFullscreenModified	Was the graphics fullscreen setting modified?	bool	false
GraphicsFullscreen	The graphics fullscreen value	bool	false
GraphicsViewDistanceModified	Was the graphics view distance setting modified?	bool	false
GraphicsViewDistance	The graphics view distance value	int	
GraphicsAntiAliasingModified	Was the graphics anti-aliasing setting modified?	bool	false
GraphicsAntiAliasing	The graphics anti-aliasing value	int	
GraphicsPostProcessingModified	Was the graphics post processing setting modified?	bool	false
GraphicsPostProcessing	The graphics post processing value	int	
GraphicsShadowQualityModified	Was the graphics shadow quality setting modified?	bool	false
GraphicsShadowQuality	The graphics shadow quality value	int	
GraphicsTextureQualityModified	Was the graphics texture quality setting modified?	bool	false
GraphicsTextureQuality	The graphics texture quality value	int	
GraphicsVisualEffectsModified	Was the graphics visual effects setting modified?	bool	false
GraphicsVisualEffects	The graphics visual effects value	int	
GraphicsShadingQualityModified	Was the graphics shading quality setting modified?	bool	false
GraphicsShadingQuality	The graphics shading quality value	int	
GraphicsVsyncModified	Was the graphics vsync setting modified?	bool	false
GraphicsVsync	The graphics vsync value	bool	false
GraphicsFpsIndicatorModified	Was the graphics FPS indicator setting modified?	bool	false
GraphicsFpsIndicator	The graphics FPS indicator value	bool	false

AccessibilityColorBlindMode	The color blind mode	int	
AccessibilityColorBlindModeSeverity	The severity of the color blind mode	float	100.0f
KeyBindings	A map of all key bindings changed by the player	TMap<FName, FKey>	

17.2 Config

17.2.1 Introduction

The configuration used for managing settings

17.2.2 API Reference

Properties

Property	Description	Type	Default Value
GameplayDifficultyImplementation	The implementation for the gameplay difficulty setting	TSubclassOf<USettingsItem>	USettingsItemGameplayDifficulty::StaticClas
GameplayDifficultyText	The text displayed in the gameplay difficulty settings item	FText	
GameplayDifficultyOptions	The array of gameplay difficulty options	TArray<FText>	
GameplayDifficultyDefault	The default gameplay difficulty value	int	0
AudioMasterImplementation	The implementation for the audio master settings item	TSubclassOf<USettingsItem>	USettingsItemAudioMaster::StaticClas
AudioMasterSoundMix	The sound mix used to manage all sound classes	USoundMix*	`nullptr`
AudioMasterText	The text displayed in the master audio settings item	FText	
AudioMasterMin	The minimum value for the master audio settings item	float	0.0f
AudioMasterMax	The maximum value for the master audio settings item	float	100.0f
AudioMasterDefault	The default value for the master audio settings item	float	100.0f
AudioMusicImplementation	The implementation for the audio music settings item	TSubclassOf<USettingsItem>	USettingsItemAudioMusic::StaticClas

AudioMusicSoundClass	The sound class used by all music	USoundClass*	`nullptr`
AudioMusicText	The text displayed in the music audio settings item	FText	
AudioMusicMin	The minimum value for the music audio settings item	float	0.0f
AudioMusicMax	The maximum value for the music audio settings item	float	100.0f
AudioMusicDefault	The default value for the music audio settings item	float	100.0f
AudioEffectsImplementation	The implementation for the audio effects settings item	TSubclassOf<USettingsItem>	USettingsItemAudioEffects::StaticClas
AudioEffectsSoundClass	The sound class used by all effects	USoundClass*	`nullptr`
AudioEffectsText	The text displayed in the effects audio settings item	FText	
AudioEffectsMin	The minimum value for the effects audio settings item	float	0.0f
AudioEffectsMax	The maximum value for the effects audio settings item	float	100.0f
AudioEffectsDefault	The default value for the effects audio settings item	float	100.0f
AudioUiImplementation	The implementation for the audio UI settings item	TSubclassOf<USettingsItem>	USettingsItemAudioUi::StaticClas

AudioUiSoundClass	The sound class used by all UI	USoundClass*	`nullptr`
AudioUiText	The text displayed in the UI audio settings item	FText	
AudioUiMin	The minimum value for the UI audio settings item	float	0.0f
AudioUiMax	The maximum value for the UI audio settings item	float	100.0f
AudioUiDefault	The default value for the UI audio settings item	float	100.0f
AudioVoiceImplementation	The implementation for the audio voice settings item	TSubclassOf<USettingsItem>	USettingsItemAudioVoice::StaticClas
AudioVoiceSoundClass	The sound class used by all voice	USoundClass*	`nullptr`
AudioVoiceText	The text displayed in the voice audio settings item	FText	
AudioVoiceMin	The minimum value for the voice audio settings item	float	0.0f
AudioVoiceMax	The maximum value for the voice audio settings item	float	100.0f
AudioVoiceDefault	The default value for the voice audio settings item	float	100.0f
GraphicsResolutionImplementation	The implementation for the graphics resolution settings item	TSubclassOf<USettingsItem>	USettingsItemGraphicsResolution::StaticClas
GraphicsResolutionText	The text displayed in the graphics	FText	

	resolution settings item		
GraphicsFullscreenImplementation	The implementation for the graphics fullscreen settings item	TSubclassOf<USettingsItem>	USettingsItemGraphicsFullscreen::StaticClas
GraphicsFullscreenText	The text displayed in the graphics fullscreen settings item	FText	
GraphicsFullscreenEnabledText	The text displayed when fullscreen is enabled	FText	
GraphicsFullscreenDisabledText	The text displayed when fullscreen is disabled	FText	
GraphicsFullscreenDefault	The default value of the fullscreen setting	bool	true
GraphicsViewDistanceImplementation	The implementation for the graphics view distance settings item	TSubclassOf<USettingsItem>	USettingsItemGraphicsViewDistance::StaticClas
GraphicsViewDistanceText	The text displayed in the graphics view distance settings item	FText	
GraphicsViewDistanceNearValueText	The text displayed when the near value is used for the view distance setting	FText	
GraphicsViewDistanceMediumValueText	The text displayed when the medium value is used for the view distance setting	FText	
GraphicsViewDistanceFarValueText	The text displayed when the far value is used for the view distance setting	FText	

GraphicsViewDistanceEpicValueText	The text displayed when the epic value is used for the view distance setting	FText	
GraphicsViewDistanceCinematicValueText	The text displayed when the cinematic value is used for the view distance setting	FText	
GraphicsViewDistanceDefault	The default value of the view distance setting	int	2
GraphicsAntiAliasingImplementation	The implementation for the graphics anti-aliasing settings item	TSubclassOf<USettingsItem>	USettingsItemGraphicsAntiAliasing::StaticClas
GraphicsAntiAliasingText	The text displayed in the graphics anti-aliasing settings item	FText	
GraphicsAntiAliasingLowValueText	The text displayed when the low value is used for the anti-aliasing setting	FText	
GraphicsAntiAliasingMediumValueText	The text displayed when the medium value is used for the anti-aliasing setting	FText	
GraphicsAntiAliasingHighValueText	The text displayed when the high value is used for the anti-aliasing setting	FText	
GraphicsAntiAliasingEpicValueText	The text displayed when the epic value is used for the anti-aliasing setting	FText	
GraphicsAntiAliasingCinematicValueText	The text displayed when the cinematic	FText	

	value is used for the anti-aliasing setting		
GraphicsAntiAliasingDefault	The default value of the anti-aliasing setting	int	2
GraphicsPostProcessingImplementation	The implementation for the graphics post processing settings item	TSubclassOf<USettingsItem>	USettingsItemGraphicsPostProcessing::StaticClas
GraphicsPostProcessingText	The text displayed in the graphics post processing settings item	FText	
GraphicsPostProcessingLowValueText	The text displayed when the low value is used for the post processing setting	FText	
GraphicsPostProcessingMediumValueText	The text displayed when the medium value is used for the post processing setting	FText	
GraphicsPostProcessingHighValueText	The text displayed when the high value is used for the post processing setting	FText	
GraphicsPostProcessingEpicValueText	The text displayed when the epic value is used for the post processing setting	FText	
GraphicsPostProcessingCinematicValueText	The text displayed when the cinematic value is used for the post processing setting	FText	
GraphicsPostProcessingDefault	The default value of the post processing setting	int	2

GraphicsShadowQualityImplementation	The implementation for the graphics shadow quality settings item	TSubclassOf<USettingsItem>	USettingsItemGraphicsShadowQuality::StaticClas
GraphicsShadowQualityText	The text displayed in the graphics shadow quality settings item	FText	
GraphicsShadowQualityLowValueText	The text displayed when the low value is used for the shadow quality setting	FText	
GraphicsShadowQualityMediumValueText	The text displayed when the medium value is used for the shadow quality setting	FText	
GraphicsShadowQualityHighValueText	The text displayed when the high value is used for the shadow quality setting	FText	
GraphicsShadowQualityEpicValueText	The text displayed when the epic value is used for the shadow quality setting	FText	
GraphicsShadowQualityCinematicValueText	The text displayed when the cinematic value is used for the shadow quality setting	FText	
GraphicsShadowQualityDefault	The default value of the shadow quality setting	int	2
GraphicsTextureQualityImplementation	The implementation for the graphics texture quality settings item	TSubclassOf<USettingsItem>	USettingsItemGraphicsTextureQuality::StaticClas
GraphicsTextureQualityText	The text displayed in the graphics	FText	

	texture quality settings item		
GraphicsTextureQualityLowValueText	The text displayed when the low value is used for the texture quality setting	FText	
GraphicsTextureQualityMediumValueText	The text displayed when the medium value is used for the texture quality setting	FText	
GraphicsTextureQualityHighValueText	The text displayed when the high value is used for the texture quality setting	FText	
GraphicsTextureQualityEpicValueText	The text displayed when the epic value is used for the texture quality setting	FText	
GraphicsTextureQualityCinematicValueText	The text displayed when the cinematic value is used for the texture quality setting	FText	
GraphicsTextureQualityDefault	The default value of the texture quality setting	int	2
GraphicsVisualEffectsImplementation	The implementation for the graphics visual effects settings item	TSubclassOf<USettingsItem>	USettingsItemGraphicsVisualEffects::StaticClas
GraphicsVisualEffectsText	The text displayed in the graphics visual effects settings item	FText	
GraphicsVisualEffectsLowValueText	The text displayed when the low value is used for the visual effects setting	FText	

GraphicsVisualEffectsMediumValueText	The text displayed when the medium value is used for the visual effects setting	FText	
GraphicsVisualEffectsHighValueText	The text displayed when the high value is used for the visual effects setting	FText	
GraphicsVisualEffectsEpicValueText	The text displayed when the epic value is used for the visual effects setting	FText	
GraphicsVisualEffectsCinematicValueText	The text displayed when the cinematic value is used for the visual effects setting	FText	
GraphicsVisualEffectsDefault	The default value of the visual effects setting	int	2
GraphicsShadingQualityImplementation	The implementation for the graphics shading quality settings item	TSubclassOf<USettingsItem>	USettingsItemGraphicsShadingQuality::StaticClas
GraphicsShadingQualityText	The text displayed in the graphics shading quality settings item	FText	
GraphicsShadingQualityLowValueText	The text displayed when the low value is used for the shading quality setting	FText	
GraphicsShadingQualityMediumValueText	The text displayed when the medium value is used for the shading quality setting	FText	
GraphicsShadingQualityHighValueText	The text displayed when the high value	FText	

	is used for the shading quality setting		
GraphicsShadingQualityEpicValueText	The text displayed when the epic value is used for the shading quality setting	FText	
GraphicsShadingQualityCinematicValueText	The text displayed when the cinematic value is used for the shading quality setting	FText	
GraphicsShadingQualityDefault	The default value of the shading quality setting	int	2
GraphicsVsyncImplementation	The implementation for the graphics vsync settings item	TSubclassOf<USettingsItem>	USettingsItemGraphicsVsync::StaticClas
GraphicsVsyncText	The text displayed in the graphics vsync settings item	FText	
GraphicsVsyncEnabledText	The text displayed when the vsync setting is enabled	FText	
GraphicsVsyncDisabledText	The text displayed when the vsync setting is disabled	FText	
GraphicsVsyncDefault	The default value of the vsync setting	bool	false
GraphicsFpsIndicatorImplementation	The implementation for the graphics FPS indicator settings item	TSubclassOf<USettingsItem>	USettingsItemGraphicsFpsIndicator::StaticClas
GraphicsFpsIndicatorText	The text displayed in the graphics FPS indicator settings item	FText	

GraphicsFpsIndicatorEnabledText	The text displayed when the FPS indicator setting is enabled	FText	
GraphicsFpsIndicatorDisabledText	The text displayed when the FPS indicator setting is disabled	FText	
GraphicsFpsIndicatorDefault	The default value of the FPS indicator setting	bool	false
AccessibilityColorBlindModeImplementation	The implementation for the accessibility color blind mode settings item	TSubclassOf<USettingsItem>	USettingsItemAccessibilityColorBlindMode::StaticC
AccessibilityColorBlindModeText	The text displayed in the accessibility color blind mode settings item	FText	
AccessibilityColorBlindModeNormalVisionText	The text displayed when the normal vision value is used for the accessibility color blind mode setting	FText	
AccessibilityColorBlindModeDeuteranopiaText	The text displayed when the deuteranopia value is used for the accessibility color blind mode setting	FText	
AccessibilityColorBlindModeDeuteranomalyText	The text displayed when the deuteranomaly value is used for the accessibility	FText	

	color blind mode setting		
AccessibilityColorBlindModeProtanopiaText	The text displayed when the protanopia value is used for the accessibility color blind mode setting	FText	
AccessibilityColorBlindModeProtanomalyText	The text displayed when the protanomaly value is used for the accessibility color blind mode setting	FText	
AccessibilityColorBlindModeTritanopiaText	The text displayed when the tritanopia value is used for the accessibility color blind mode setting	FText	
AccessibilityColorBlindModeTritanomalyText	The text displayed when the tritanomaly value is used for the accessibility color blind mode setting	FText	
AccessibilityColorBlindModeSeverityImplementation	The implementation for the accessibility color blind mode severity settings item	TSubclassOf<USettingsItem>	USettingsItemAccessibilityColorBlindModeSeverity
AccessibilityColorBlindModeSeverityText	The text displayed in the accessibility color blind mode severity settings item	FText	
ControlsRemapImplementation	The implementation for the controls remap settings item	TSubclassOf<USettingsItem>	USettingsItemControlsRemap::StaticClas

| ControlsWaitingForKeyPressText | The text displayed in the menu item while waiting for the user to press a new key | FText | |

17.3 Items

17.3.1 Types

Introduction

An enum of all supported settings items

Values

Value	Description
None	A setting not managed by the game instance
GameplayDifficulty	The game difficulty setting
AudioMaster	The master audio volume setting
AudioMusic	The music audio volume setting
AudioEffects	The effects audio volume setting
AudioUi	The UI audio volume setting
AudioVoice	The voice audio volume setting
GraphicsResolution	The graphics resolution setting
GraphicsFullscreen	The graphics fullscreen setting
GraphicsViewDistance	The graphics view distance setting
GraphicsAntiAliasing	The graphics anti-aliasing setting
GraphicsPostProcessing	The graphics post processing setting
GraphicsShadowQuality	The graphics shadow quality setting
GraphicsTextureQuality	The graphics texture quality setting
GraphicsVisualEffects	The graphics visual effects setting
GraphicsShadingQuality	The graphics shading quality setting
GraphicsVSync	The graphics vsync setting
GraphicsFpsIndicator	The graphics FPS indicator setting
AccessibilityColorBlindMode	The accessibility color blind mode setting
AccessibilityColorBlindModeSeverity	The accessibility color blind mode severity setting
ControlsRemap	Remap the controls

17.3.2 Logic

Introduction

An implementation for a settings item controlling how the setting is configured, saved and applied

Dependencies

The `SettingsItem` relies on other components of this plugin to work:

• Game Instance: Used to monitor for input device changes and handle saving/loading game data

Implementations

There is already implementations for all settings items. But you can expand this if needed

Category	Name	Description
Audio	SettingsItemAudioMaster	An implementation for the audio master settings item
Audio	SettingsItemAudioMusic	An implementation for the audio music settings item
Audio	SettingsItemAudioEffects	An implementation for the audio effects settings item
Audio	SettingsItemAudioUi	An implementation for the audio UI settings item
Audio	SettingsItemAudioVoice	An implementation for the audio voice settings item
Graphics	SettingsItemGraphicsResolution	An implementation for the graphics resolution settings item
Graphics	SettingsItemGraphicsFullscreen	An implementation for the graphics fullscreen settings item
Graphics	SettingsItemGraphicsViewDistance	An implementation for the graphics view distance settings item
Graphics	SettingsItemGraphicsAntiAliasing	An implementation for the graphics anti-aliasing settings item
Graphics	SettingsItemGraphicsPostProcessing	An implementation for the graphics post processing settings item
Graphics	SettingsItemGraphicsShadowQuality	An implementation for the graphics shadow quality settings item
Graphics	SettingsItemGraphicsTextureQuality	An implementation for the graphics texture quality settings item
Graphics	SettingsItemGraphicsVisualEffects	An implementation for the graphics visual effects settings item
Graphics	SettingsItemGraphicsShadingQuality	An implementation for the graphics shading quality settings item
Graphics	SettingsItemGraphicsVsync	An implementation for the graphics vsync settings item
Graphics	SettingsItemGraphicsFpsIndicator	An implementation for the graphics FPS indicator settings item
Accessibility	SettingsItemAccessibilityColorBlindMode	An implementation for the accessibility color blind mode settings item
Accessibility	SettingsItemAccessibilityColorBlindModeSeverity	An implementation for the accessibility color blind mode severity settings item
Controls	SettingsItemControlsRemap	An implementation for the controls settings item

API Reference

FUNCTIONS

Name	Description	Params	Return
ConfigureMenuItem	Configure the menu item	**Config (void)** The settings config specified in the game instance **Settings (USettingsData*)** The current settings data **MenuItem (UMenuItem*)** The menu item to configure	
SaveSettings	Save the settings managed by a menu item	**Settings (USettingsData*)** The current settings data **MenuItem (UMenuItem*)** The menu item containing the updated settings	**USettingsData*** The updated settings data
ApplySettings	Apply the settings	**World (UObject*)** The world context **Config (USettingsConfig*)** The settings config specified in the game instance **Settings (USettingsData*)** The current settings data	

Blueprint Usage

You can use the `SettingsItem` using Blueprints by adding one of the following nodes:

• Ultimate Starter Kit > Settings > Configure Menu Item

• Ultimate Starter Kit > Settings > Save Settings

• Ultimate Starter Kit > Settings > Apply Settings

C++ Usage

Before you can use the plugin, you first need to enable the plugin in your `Build.cs` file:

```
PublicDependencyModuleNames.Add("USK");
```

The `SettingsItem` can now be used in any of your C++ files:

```
#include "USK/Settings/SettingsItem.h"

void ATestActor::Test()
{
    // SettingsItem is a pointer to the USettingsItem
    SettingsItem->ConfigureMenuItem(Config, Settings, MenuItem);
    USettingsData* SaveSettingsValue = SettingsItem->SaveSettings(Settings, MenuItem);
    SettingsItem->ApplySettings(World, Config, Settings);
}
```

17.4 Utils

17.4.1 Introduction

A Blueprint Function Library class used to load, save and apply all settings

17.4.2 Dependencies

The `SettingsUtils` relies on other components of this plugin to work:

- Logger: Used to log useful information to help you debug any issues you might experience

- Game Instance: Used to monitor for input device changes and handle saving/loading game data

17.4.3 Requirements

It's important that you configure the settings in the Game Instance before you can use the settings feature

17.4.4 Controls Settings

Before you can use the plugin to automatically handle the controls settings, you need to configure your Input Mapping Context. You are required to specify the `Name` for each key that can be changed through the plugin. This `Name` should match the value you specify in your `Menu Item`

NB: This feature is only available on Unreal Engine 5 and newer (UE5.3 support coming soon)

17.4.5 API Reference

Functions

Name	Description	Params	Return
Initialize	Initialize the settings	**GameInstance (UUSKGameInstance*)** A reference to the game instance	
LoadSettings	Load the settings		**USettingsData*** The loaded settings data
SaveSettings	Save the settings	**Settings (USettingsData*)** The updated settings data	
ApplySettingsInWorld	Apply the settings	**World (UObject*)** The world context **Settings (USettingsData*)** The settings data to apply	
ApplySettings	Apply the settings	**GameInstance (UUSKGameInstance*)** A reference to the game instance **Settings (USettingsData*)** The settings data to apply	
ConfigureMenuItem	Configure the menu item to manage the specified settings item	**MenuItem (UMenuItem*)** The menu item to configure	
SaveMenuItemSettings	Save the settings managed by the menu item	**MenuItem (UMenuItem*)** The menu item containing the updated settings **ApplySettings (bool)** Should the settings also be applied?	
ApplyMenuItemSettings	Apply the settings managed by the menu item	**MenuItem (UMenuItem*)** The menu item containing the updated settings	
GetSettingsItemForMenuItem	Get the settings item for the specified menu item	**MenuItem (UMenuItem*)** The menu item to get the settings item for **Config (USettingsConfig*)** The settings config specified in the game instance	**USettingsItem*** The settings item
GetSettingsItem	Get the settings item for the specified settings item type	**SettingsItemType (ESettingsItemType)** The menu item to get the settings item for **Config (USettingsConfig*)** The settings config specified in the game instance	**USettingsItem*** The settings item

17.4.6 Blueprint Usage

You can use the `SettingsUtils` using Blueprints by adding one of the following nodes:

- Ultimate Starter Kit > Settings > Initialize

- Ultimate Starter Kit > Settings > Load Settings

- Ultimate Starter Kit > Settings > Save Settings

- Ultimate Starter Kit > Settings > Apply Settings In World

- Ultimate Starter Kit > Settings > Apply Settings

- Ultimate Starter Kit > Settings > Configure Menu Item

- Ultimate Starter Kit > Settings > Save Menu Item Settings

- Ultimate Starter Kit > Settings > Apply Menu Item Settings

- Ultimate Starter Kit > Settings > Get Settings Item For Menu Item

- Ultimate Starter Kit > Settings > Get Settings Item

17.4.7 C++ Usage

Before you can use the plugin, you first need to enable the plugin in your `Build.cs` file:

```
PublicDependencyModuleNames.Add("USK");
```

The `SettingsUtils` can now be used in any of your C++ files:

```cpp
#include "USK/Settings/SettingsUtils.h"

void ATestActor::Test()
{
    USettingsUtils::Initialize(GameInstance);
    USettingsData* LoadSettingsValue = USettingsUtils::LoadSettings();
    USettingsUtils::SaveSettings(Settings);
    USettingsUtils::ApplySettingsInWorld(World, Settings);
    USettingsUtils::ApplySettings(GameInstance, Settings);
    USettingsUtils::ConfigureMenuItem(MenuItem);
    USettingsUtils::SaveMenuItemSettings(MenuItem, ApplySettings);
    USettingsUtils::ApplyMenuItemSettings(MenuItem);
    USettingsItem* SettingsItemForMenuItem = USettingsUtils::GetSettingsItemForMenuItem(MenuItem, Config);
    USettingsItem* SettingsItem = USettingsUtils::GetSettingsItem(SettingsItemType, Config);
}
```

18. Utils

18.1 Config Utils

18.1.1 Introduction

A Blueprint Function Library class used to extract config values

18.1.2 Dependencies

The `ConfigUtils` relies on other components of this plugin to work:

• Logger: Used to log useful information to help you debug any issues you might experience

18.1.3 API Reference

Functions

Name	Description	Params	Return
GetConfigValue	Extract a config value from a given config file	**Filename (FString)** The name of the config file **Section (FString)** The section in the config file **Key (FString)** The key in the config file **DefaultValue (FString)** The default value to return if the config file can't be read	**FString** The value extracted from the config file
GetGameConfigValue	Extract a config value from the default game config file	**Section (FString)** The section in the config file **Key (FString)** The key in the config file **DefaultValue (FString)** The default value to return if the config file can't be read	**FString** The value extracted from the config file

18.1.4 Blueprint Usage

You can use the `ConfigUtils` using Blueprints by adding one of the following nodes:

• Ultimate Starter Kit > Utils > Config > Get Config Value

• Ultimate Starter Kit > Utils > Config > Get Game Config Value

18.1.5 C++ Usage

Before you can use the plugin, you first need to enable the plugin in your `Build.cs` file:

```
PublicDependencyModuleNames.Add("USK");
```

The `ConfigUtils` can now be used in any of your C++ files:

```
#include "USK/Utils/ConfigUtils.h"

void ATestActor::Test()
{
    FString ConfigValue = UConfigUtils::GetConfigValue(Filename, Section, Key, DefaultValue);
    FString GameConfigValue = UConfigUtils::GetGameConfigValue(Section, Key, DefaultValue);
}
```

```
#include "USK/Utils/ConfigUtils.h"
```

18.2 Platform

18.2.1 Platform Type

Introduction

The types of supported platform types

Values

Value	Description
Unknown	An unknown or unsupported platform
Windows	Windows (any architecture)
MacOS	MacOS (any architecture)
Linux	Linux (any architecture)
ConsoleMx	Console MX
ConsoleSp	Console SP
ConsoleNs	Console NS
Android	Android (any architecture)
IOS	iOS

18.2.2 Platform Utils

Introduction

A Blueprint Function Library class used for platform detection

API Reference

FUNCTIONS

Name	Description	Params	Return
GetPlatform	Get the current platform		**EPlatform** The current platform
IsInEditor	Is the build running inside the editor?		**bool** A boolean value indicating if the build is running inside the editor
IsDesktop	Is the build running on a desktop platform?		**bool** A boolean value indicating if the build is running on a desktop platform
IsWindows	Is the build running on Windows?		**bool** A boolean value indicating if the build is running on Windows
IsMacOS	Is the build running on MacOS?		**bool** A boolean value indicating if the build is running on MacOS
IsMacOSx86	Is the build running on MacOS (x86)?		**bool** A boolean value indicating if the build is running on MacOS (x86)
IsMacOSArm	Is the build running on MacOS (ARM)?		**bool** A boolean value indicating if the build is running on MacOS (ARM)
IsLinux	Is the build running on Linux?		**bool** A boolean value indicating if the build is running on Linux
IsConsole	Is the build running on a console platform?		**bool** A boolean value indicating if the build is running on a console platform
IsConsoleMx	Is the build running on Console MX?		**bool** A boolean value indicating if the build is running on Console MX
IsConsoleSp	Is the build running on Console SP?		**bool** A boolean value indicating if the build is running on Console SP
IsConsoleNs	Is the build running on Console NS?		**bool** A boolean value indicating if the build is running on Console NS
IsMobile	Is the build running on a mobile platform?		**bool** A boolean value indicating if the build is running on a mobile platform
IsAndroid	Is the build running on Android?		**bool** A boolean value indicating if the build is running on Android
IsAndroidx86	Is the build running on Android (x86)?		**bool** A boolean value indicating if the build is running on Android (x86)
IsAndroidx64	Is the build running on Android (x64)?		**bool** A boolean value indicating if the build is running on Android (x64)

IsAndroidArm	Is the build running on Android (ARM)?		**bool** A boolean value indicating if the build is running on Android (ARM)
IsAndroidArm64	Is the build running on Android (ARM64)?		**bool** A boolean value indicating if the build is running on Android (ARM64)
IsIOS	Is the build running on iOS?		**bool** A boolean value indicating if the build is running on iOS

Blueprint Usage

You can use the `PlatformUtils` using Blueprints by adding one of the following nodes:

- Ultimate Starter Kit > Utils > Platform > Get Platform

- Ultimate Starter Kit > Utils > Platform > Is In Editor

- Ultimate Starter Kit > Utils > Platform > Is Desktop

- Ultimate Starter Kit > Utils > Platform > Is Windows

- Ultimate Starter Kit > Utils > Platform > Is MacOS

- Ultimate Starter Kit > Utils > Platform > Is MacOS (x86)

- Ultimate Starter Kit > Utils > Platform > Is MacOS (ARM)

- Ultimate Starter Kit > Utils > Platform > Is Linux

- Ultimate Starter Kit > Utils > Platform > Is Console

- Ultimate Starter Kit > Utils > Platform > Is Console MX

- Ultimate Starter Kit > Utils > Platform > Is Console SP

- Ultimate Starter Kit > Utils > Platform > Is Console NS

- Ultimate Starter Kit > Utils > Platform > Is Mobile

- Ultimate Starter Kit > Utils > Platform > Is Android

- Ultimate Starter Kit > Utils > Platform > Is Android (x86)

- Ultimate Starter Kit > Utils > Platform > Is Android (x64)

- Ultimate Starter Kit > Utils > Platform > Is Android (ARM)

- Ultimate Starter Kit > Utils > Platform > Is Android (ARM64)

- Ultimate Starter Kit > Utils > Platform > Is iOS

C++ Usage

Before you can use the plugin, you first need to enable the plugin in your `Build.cs` file:

```
PublicDependencyModuleNames.Add("USK");
```

The `PlatformUtils` can now be used in any of your C++ files:

```
#include "USK/Utils/PlatformUtils.h"

void ATestActor::Test()
{
    EPlatform Platform = UPlatformUtils::GetPlatform();
    bool IsInEditorValue = UPlatformUtils::IsInEditor();
    bool IsDesktopValue = UPlatformUtils::IsDesktop();
    bool IsWindowsValue = UPlatformUtils::IsWindows();
    bool IsMacOSValue = UPlatformUtils::IsMacOS();
    bool IsMacOSx86Value = UPlatformUtils::IsMacOSx86();
    bool IsMacOSArmValue = UPlatformUtils::IsMacOSArm();
    bool IsLinuxValue = UPlatformUtils::IsLinux();
    bool IsConsoleValue = UPlatformUtils::IsConsole();
    bool IsConsoleMxValue = UPlatformUtils::IsConsoleMx();
    bool IsConsoleSpValue = UPlatformUtils::IsConsoleSp();
    bool IsConsoleNsValue = UPlatformUtils::IsConsoleNs();
```

```
    bool IsMobileValue = UPlatformUtils::IsMobile();
    bool IsAndroidValue = UPlatformUtils::IsAndroid();
    bool IsAndroidx86Value = UPlatformUtils::IsAndroidx86();
    bool IsAndroidx64Value = UPlatformUtils::IsAndroidx64();
    bool IsAndroidArmValue = UPlatformUtils::IsAndroidArm();
    bool IsAndroidArm64Value = UPlatformUtils::IsAndroidArm64();
    bool IsIOSValue = UPlatformUtils::IsIOS();
}
```

18.3 Project Utils

18.3.1 Introduction

A Blueprint Function Library class used to extract project values

18.3.2 API Reference

Functions

Name	Description	Params	Return
GetProjectId	Get the project ID from the game config file		**FString** The project ID
GetProjectName	Get the project name from the game config file		**FString** The project name
GetProjectDescription	Get the project description from the game config file		**FString** The project description
GetProjectVersion	Get the project version from the game config file		**FString** The project version
GetProjectCompanyName	Get the project company name from the game config file		**FString** The project company name
GetProjectCopyrightNotice	Get the project copyright notice from the game config file		**FString** The project copyright notice
GetProjectLicensingTerms	Get the project licensing terms from the game config file		**FString** The project licensing terms
GetProjectHomepage	Get the project homepage from the game config file		**FString** The project homepage

18.3.3 Blueprint Usage

You can use the `ProjectUtils` using Blueprints by adding one of the following nodes:

- Ultimate Starter Kit > Utils > Project > Get Project Id
- Ultimate Starter Kit > Utils > Project > Get Project Name
- Ultimate Starter Kit > Utils > Project > Get Project Description
- Ultimate Starter Kit > Utils > Project > Get Project Version
- Ultimate Starter Kit > Utils > Project > Get Project Company Name
- Ultimate Starter Kit > Utils > Project > Get Project Copyright Notice
- Ultimate Starter Kit > Utils > Project > Get Project Licensing Terms
- Ultimate Starter Kit > Utils > Project > Get Project Homepage

18.3.4 C++ Usage

Before you can use the plugin, you first need to enable the plugin in your `Build.cs` file:

```
PublicDependencyModuleNames.Add("USK");
```

The `ProjectUtils` can now be used in any of your C++ files:

```cpp
#include "USK/Utils/ProjectUtils.h"

void ATestActor::Test()
{
    FString ProjectId = UProjectUtils::GetProjectId();
    FString ProjectName = UProjectUtils::GetProjectName();
    FString ProjectDescription = UProjectUtils::GetProjectDescription();
    FString ProjectVersion = UProjectUtils::GetProjectVersion();
    FString ProjectCompanyName = UProjectUtils::GetProjectCompanyName();
    FString ProjectCopyrightNotice = UProjectUtils::GetProjectCopyrightNotice();
    FString ProjectLicensingTerms = UProjectUtils::GetProjectLicensingTerms();
    FString ProjectHomepage = UProjectUtils::GetProjectHomepage();
}
```