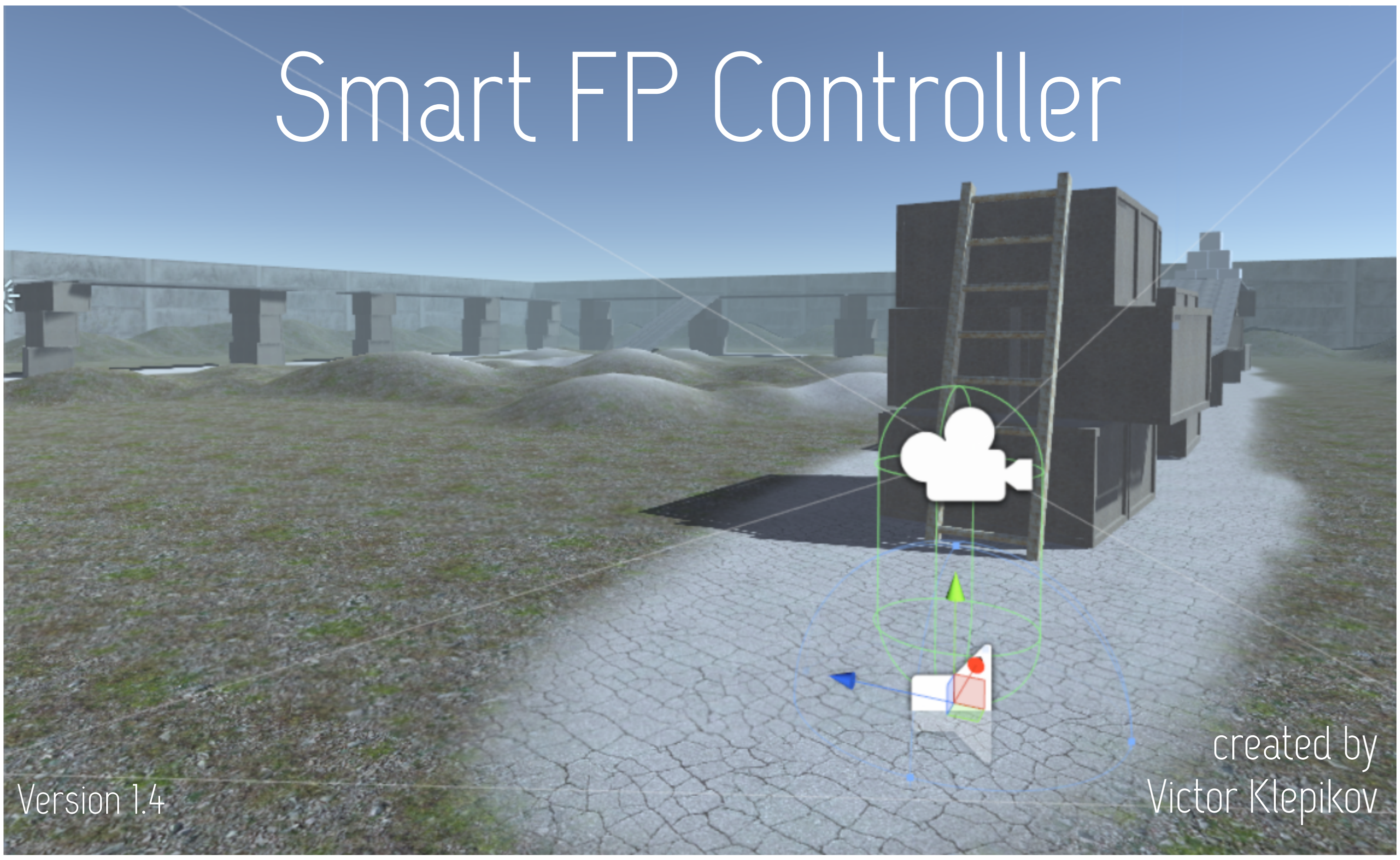


Thanks for your purchase!
Your support is greatly appreciated!

Introduction

Asset **Smart First Person Controller** is a powerful, flexible and intuitive to use first person controller system, allows you to quickly and easily setup your player controller for any first person game, and have them working in minutes! And many other features.

Included features are:

- ✓ All asset files, for free or commercial use (re-selling prohibited).
- ✓ States: Idle. Walking. Running. Jumping. Crouching. Climbing.
- ✓ Ladders and climbing system.
- ✓ Smart surface detection.
- ✓ Smart head bobbing system.
- ✓ Falling Damage Calculation.
- ✓ Advanced input system.
- ✓ Optimized for mobile.
- ✓ Intuitive and easy to modify source code for any of your needs.

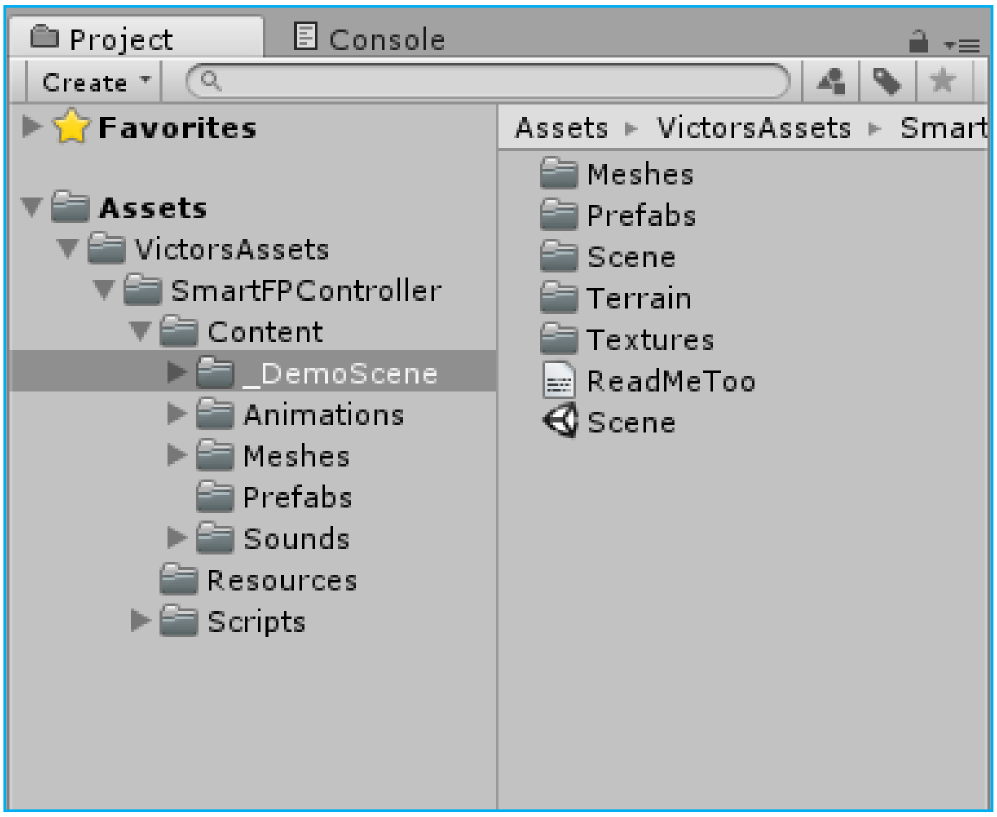
These features should cover the most requirements for a shooting games. However, please note that a this project can't suit all game cases. You likely want to modify it to fit your needs and implement your own unique game and user interface mechanics. In the following chapters, this manual explains all components involved in this kit, so you can see where you might want to start.

First Steps

WARNING: If you are new to Unity, please take a quick break and get dirty with its main functionalities first, because this documentation will assume you have some basic knowledge regarding the interface and its editor tools.

Import the "Smart FP Controller" unitypackage into an empty project.

Once the import has finished, you'll see all project files listed in the Project panel.

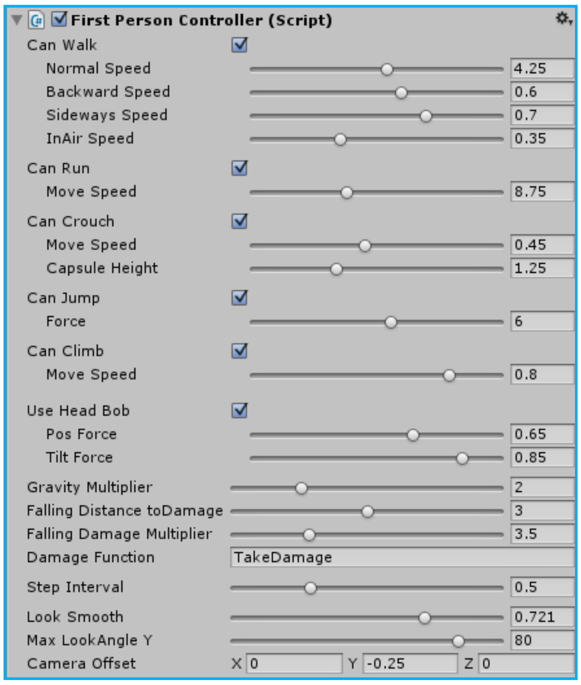


This asset contains one demo scene:

This scene - Demonstrated the all features of this first person controller.

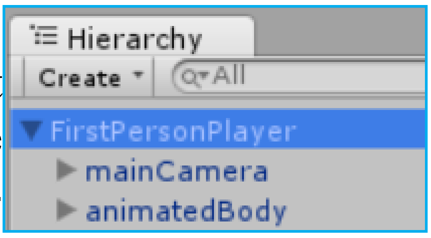
So, you probably have already seen how it works and you already want to understand the principles of operation, as well as set up all by your project. Well, let's start, the following pages are devoted to this.

Player



First Person Controller (Script) ✓

Can Walk	✓
Normal Speed	4.25
Backward Speed	0.6
Sideways Speed	0.7
InAir Speed	0.35
Can Run	✓
Move Speed	8.75
Can Crouch	✓
Move Speed	0.45
Capsule Height	1.25
Can Jump	✓
Force	6
Can Climb	✓
Move Speed	0.8
Use Head Bob	✓
Pos Force	0.65
Tilt Force	0.85
Gravity Multiplier	2
Falling Distance toDamage	3
Falling Damage Multiplier	3.5
Damage Function	TakeDamage
Step Interval	0.5
Look Smooth	0.721
Max LookAngle Y	80
Camera Offset	X 0 Y -0.25 Z 0

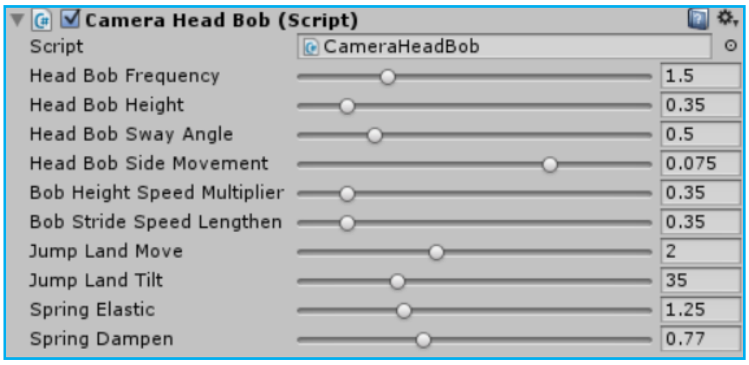


Hierarchy
Create All
FirstPersonPlayer
mainCamera
animatedBody

This script is responsible for player moving on scene.

Sets the movement speed and another parameters for all states.

Also, play sounds for currrent surface. And responsible for main camera look controlling.

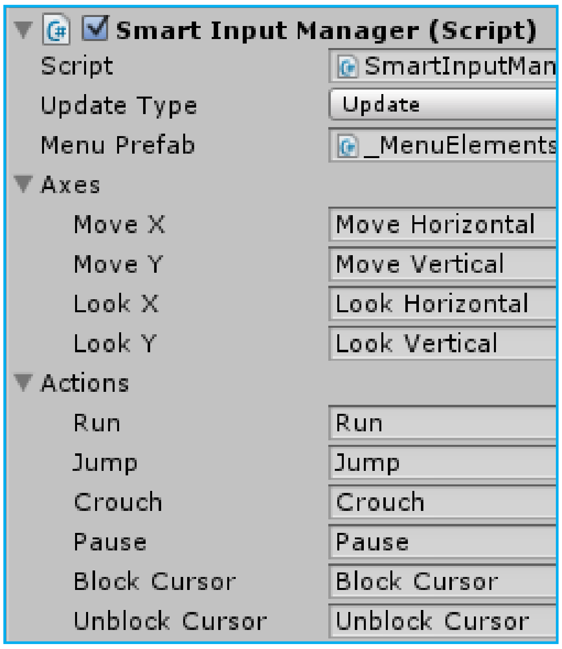


Camera Head Bob (Script) ✓

Script	CameraHeadBob
Head Bob Frequency	1.5
Head Bob Height	0.35
Head Bob Sway Angle	0.5
Head Bob Side Movement	0.075
Bob Height Speed Multiplier	0.35
Bob Stride Speed Lengthen	0.35
Jump Land Move	2
Jump Land Tilt	35
Spring Elastic	1.25
Spring Dampen	0.77

This script will need to calculate the shaking effect of the head and weapons when player movement.

Player
input settings



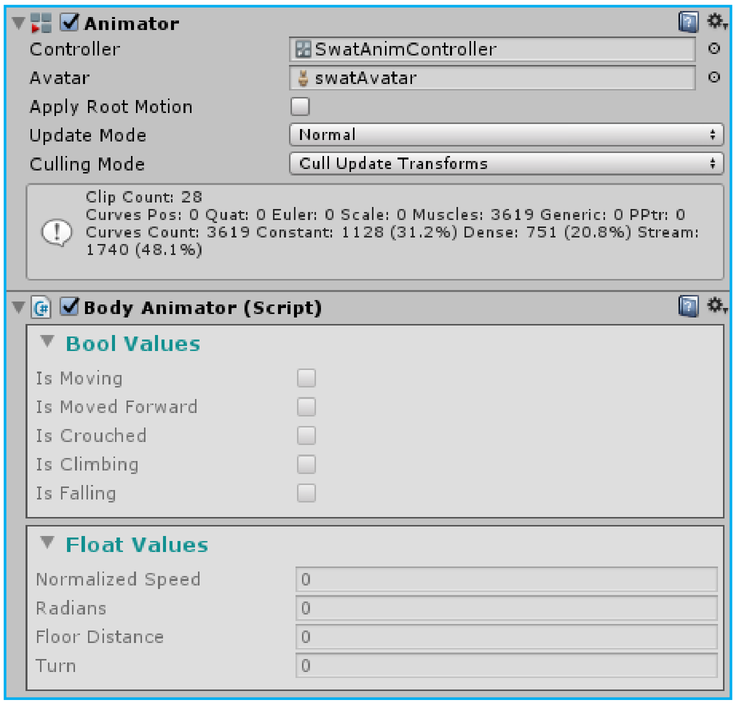
Smart Input Manager (Script) ✓

Script	SmartInputMan
Update Type	Update
Menu Prefab	_MenuElements
Axes	
Move X	Move Horizontal
Move Y	Move Vertical
Look X	Look Horizontal
Look Y	Look Vertical
Actions	
Run	Run
Jump	Jump
Crouch	Crouch
Pause	Pause
Block Cursor	Block Cursor
Unblock Cursor	Unblock Cursor

This script is responsible for playback animations of player body.

Since version 1.4 all animation systems of this asset has been moved to mecanim.

All fields values are read-only. Because mainly they are used for transfer values to the animator controller.


Animator

Controller	SwatAnimController
Avatar	swatAvatar
Apply Root Motion	☐
Update Mode	Normal
Culling Mode	Cull Update Transforms

```
Clip Count: 28
Curves Pos: 0 Quat: 0 Euler: 0 Scale: 0 Muscles: 3619 Generic: 0 PPtr: 0
Curves Count: 3619 Constant: 1128 (31.2%) Dense: 751 (20.8%) Stream:
1740 (48.1%)
```

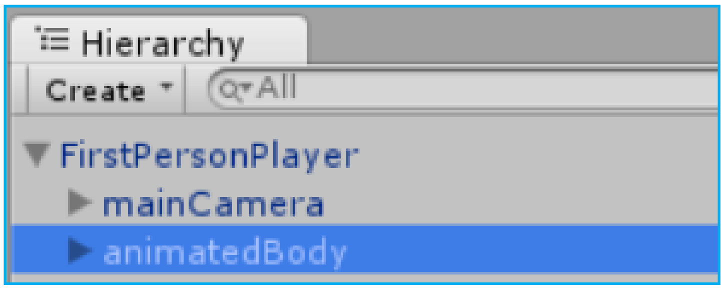
Body Animator (Script)

Bool Values

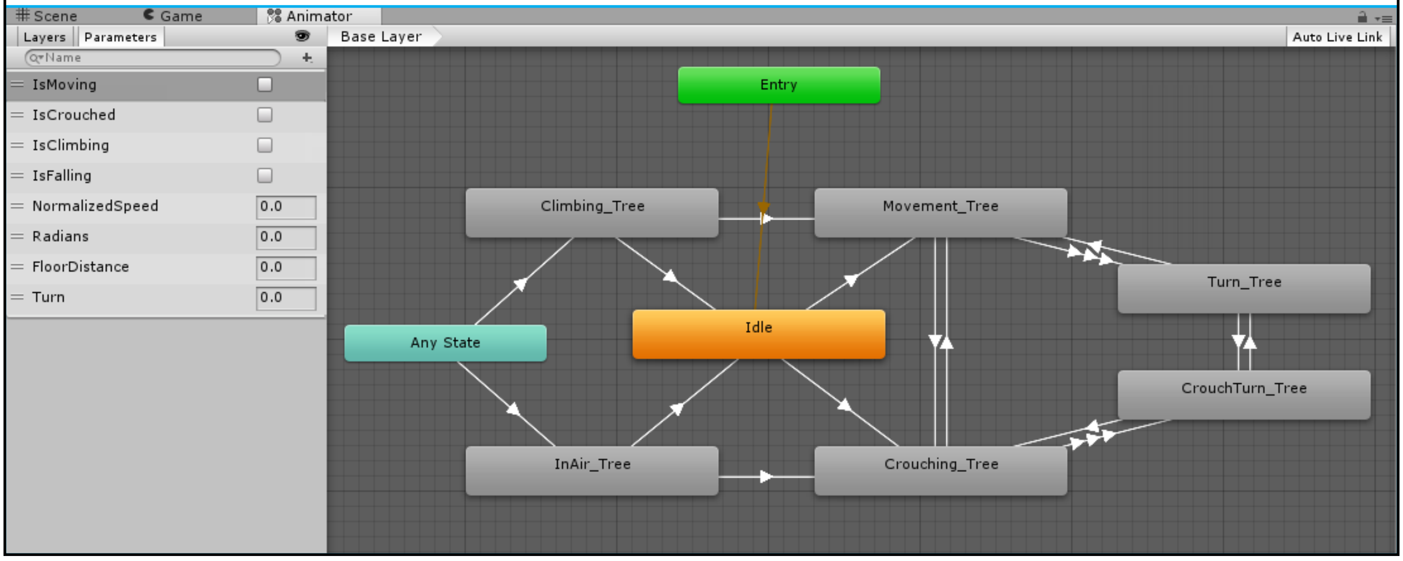
Is Moving	☐
Is Moved Forward	☐
Is Crouched	☐
Is Climbing	☐
Is Falling	☐

Float Values

Normalized Speed	0
Radians	0
Floor Distance	0
Turn	0



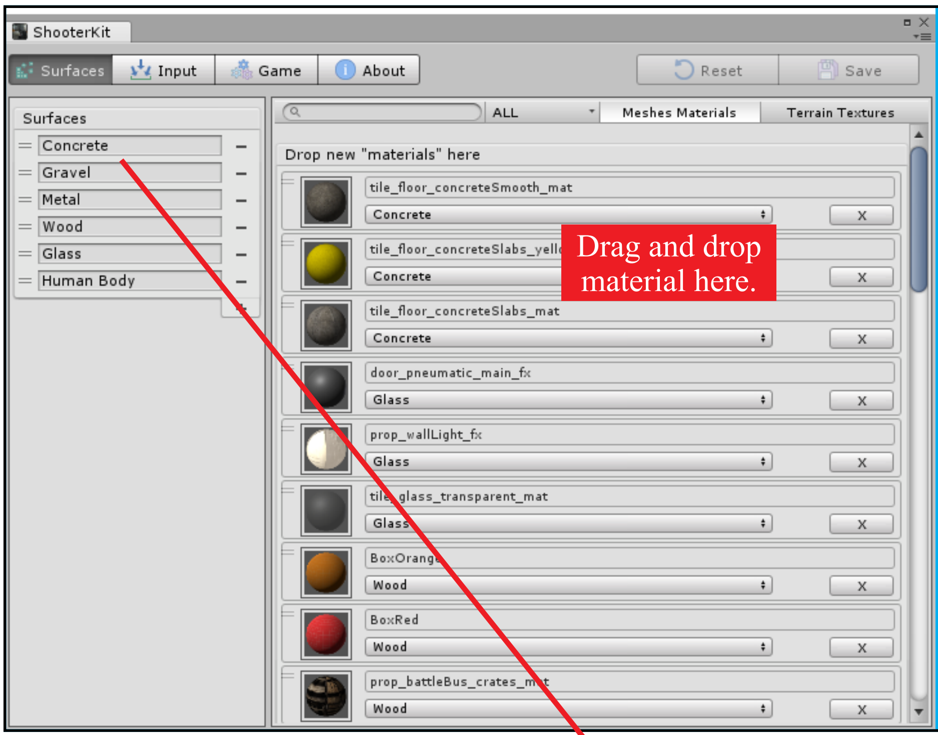
Assets/VictorsAssets/SmartFPController/Content/Animations



IsMoving	☐
IsCrouched	☐
IsClimbing	☐
IsFalling	☐
NormalizedSpeed	0.0
Radians	0.0
FloorDistance	0.0
Turn	0.0

Surfaces

Tools -> Victors Assets -> Smart FP Controller->Settings



Surfaces
- Concrete
- Gravel
- Metal
- Wood
- Glass
- Human Body

Drop new "materials" here

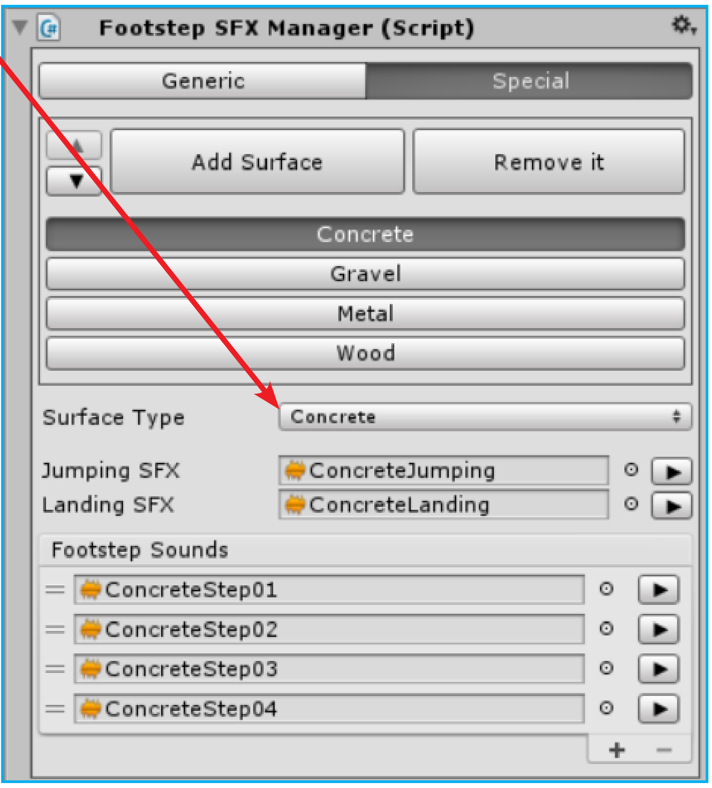
tile_floor_concreteSmooth_mat	Concrete
tile_floor_concreteSlabs_yell	Concrete
tile_floor_concreteSlabs_mat	Concrete
door_pneumatic_main_fx	Glass
prop_wallLight_fx	Glass
tile_glass_transparent_mat	Glass
BoxOrange	Wood
BoxRed	Wood
prop_battleBus_crates_mat	Wood

Surfases window tab need for configuration surface detection system.

Add (register) material to associate it for custom surface.

This component configurate all sounds of character movement, it`s a footsteps, jumping and landing sounds.

**Generic for unknown surface.*



Surface Type: Concrete

Jumping SFX: ConcreteJumping

Landing SFX: ConcreteLanding

Footstep Sounds
- ConcreteStep01
- ConcreteStep02
- ConcreteStep03
- ConcreteStep04

Input

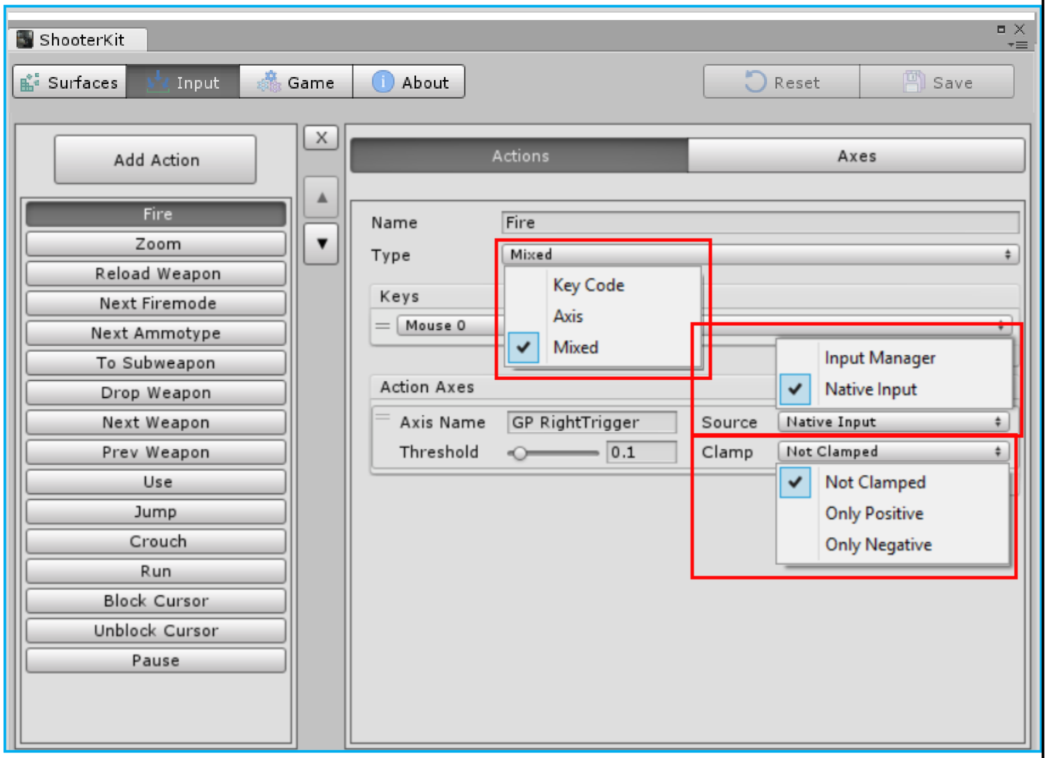
<u>Tools -> Victors Assets -> Smart FP Controller->Settings</u>

Actions sub tab need for register the actions (buttons and axes) to perform actions. Use this only when implementing events that trigger an action.

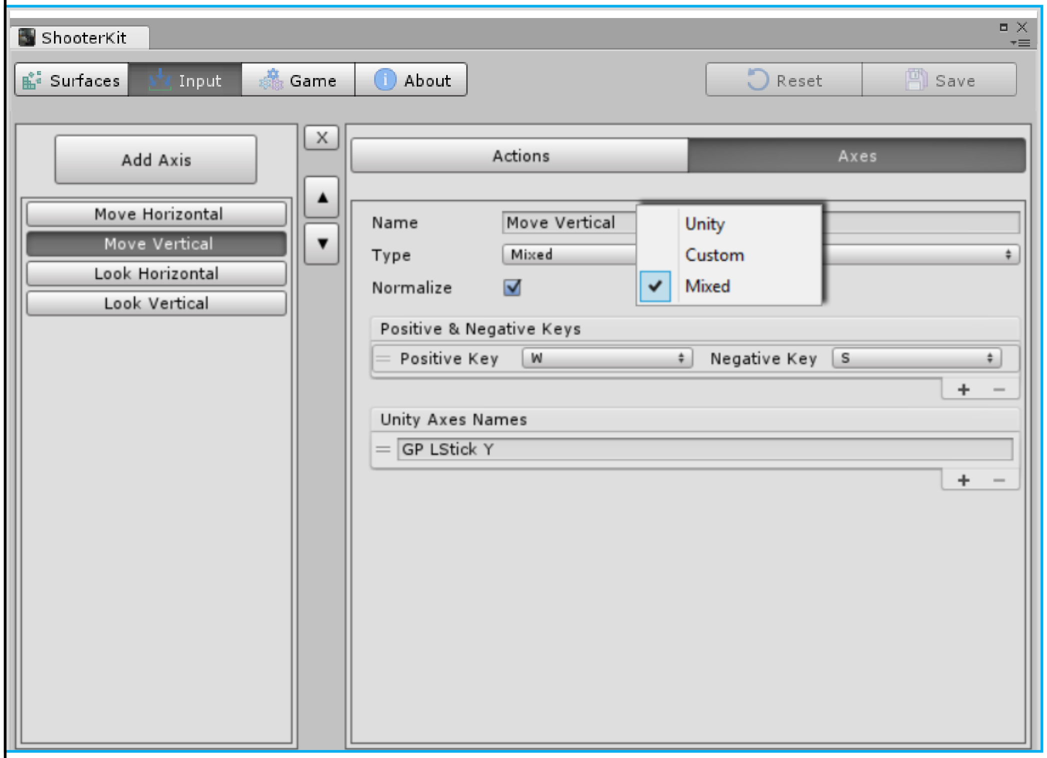
Input Manager can be handle ***Events*** *based on* ***Axis*** *values*.

Source - used for select source of get axis value. *<u>Input Manager</u>* - get axis registered in this manager. *<u>Native Input</u>* used for get axis directly from the Unity.Input.

Clamp used to clamp the events of only positive or only negative values.



Name: Fire
Type: Mixed [✓ Mixed]
Keys: Mouse 0
Action Axes
Axis Name: GP RightTrigger
Threshold: 0.1
Source: Native Input [✓ Native Input]
Clamp: Not Clamped [✓ Not Clamped] Only Positive Only Negative



Name: Move Vertical
Type: Mixed [✓ Mixed] Unity Custom
Normalize: ✓
Positive & Negative Keys
Positive Key: W Negative Key: S
Unity Axes Names
GP LStick Y

Actions sub tab need for register axis to get float value.

Custom - The value will be in the range -1...1 for keyboard and joystick input. Since input is not smoothed, keyboard input will always be either -1, 0 or 1. This is useful if you want to do all smoothing of keyboard input processing yourself.

Unity - The value will be in the range -1...1 for keyboard and joystick input. If the axis is setup to be delta mouse movement, the mouse delta is multiplied by the axis sensitivity and the range is not -1...1.

Normalize - if true the value will be in the range -1...1.

API

namespace: **SmartFPController**

class SmartInputManager

static void BindAction(string m_Name, EActionEvent m_Event, ActionHandler m_Handler);	Bind your void to named action.
static void UnbindAction(string m_Name, EActionEvent m_Event, ActionHandler m_Handler);	Unbind your void to named action.
static void BindActionByAxis(string actionName, AxisState axisState, ActionHandler m_Handler);	Bind your void to named axis works as action.
static void UnbindActionAxis(string actionName, AxisState axisState, ActionHandler m_Handler);	Unbind your void to named axis works as action.
static void BindAxis (string m_Name, AxisHandler m_Handler);	Bind your void to named axis.
static void UnbindAxis (string m_Name, AxisHandler m_Handler);	Unbind your void to named axis.
static bool GetAction (string m_Name, EActionEvent m_Event);	Use this only when implementing events that trigger an action.
static bool GetActionByAxis (string actionName, EAxisState axisState);	Use this only when implementing events based only axis value that trigger an action.
static float GetAxis(string m_Name);	Returns the value of the virtual axis identified by name.
void BlockCursor();	Locking mouse cursor.
void UnblockCursor();	Unlocking mouse cursor.
void Pause();	Pause/Unpause game.

delegate void ActionHandler();
delegate void AxisHandler(float value);

enum EActionEvent

Down - during the frame the user pressed down the virtual button.
Pressed - while the virtual button is held down.
Up - first frame the user releases the virtual button.

enum EAxisState **-** Event of axis state works as EActionEvent, but based only axis value.

PositiveDown, **PositivePress**, **PositiveUp**, **-** Events based Positive axis values.
NegativeDown, **NegativePress**, **NegativeUp -** Events based Negative axis values.

class FirstPersonController	
bool isGrounded { get; }	Was the player touching the ground during the last move?
bool isClimbing { get; }	Was the player climbs?
bool isMoving { get; }	Was the player movement?
bool isRunning { get; }	Was the player runs?
bool isCrouched { get; }	Was the player crouching?
bool isJumping { get; }	Was the player jumping?
bool isFalling { get; }	Was the player falling?
RaycastHit floorHit { get; }	Used to get information back from a floor sphere cast.

class FootstepSFXManager	
void PlayJumpingSound(RaycastHit hit);	Play sound on jumping start.
void PlayLandingSound(RaycastHit hit);	Play sound when player fell the ground.
void PlayFootStepSound(RaycastHit hit);	Play one foorstep sound from array.

class SurfaceDetector	
static string GetSurface(RaycastHit hit);	Returns name of surface by RaycastHit.
static int count { get; }	Returns count of surfaces.
static string[] allNames { get; }	Returns string array of surfaces names.

extensions of RaycastHit	
static string GetSurface();	Returns surface name by RaycastHit.
static Material GetMaterial();	Returns mesh material by RaycastHit.
static Texture GetTerrainTexture();	Returns terrain texture by RaycastHit.

class GameSettings	
static bool InvertLookX { get; set; }	Invert horizontal look axis.
static bool InvertLookY { get; set; }	Invert vertical look axis.
static float LookSensitivity { get; set; }	Get/Set of look sensitivity.
static float MasterVolume { get; set; }	The total volume for all Audio Sources (AS).
static float MusicVolume { get; set; }	Volume level for AS whose output is set to music.
static float SFXVolume { get; set; }	Volume level for AS whose output is set to SFX.
static float VoiceVolume { get; set; }	Volume level for AS whose output is set to voice.
static AudioMixer MasterMixer { get; }	Get master mixer.
static AudioMixerGroup MusicOutput { get; }	Get music output Mixer Group.
static AudioMixerGroup SFXOutput { get; }	Get sfx output Mixer Group.
static AudioMixerGroup VoiceOutput { get; }	Get voice output Mixer Group.

Contacts

All the source code is made so that it is easy to understand,
feel free to take a look at the scripts
and to modify them to fit your needs.

If you have any questions, comments, suggestions or find errors
in this documentation, don`t hesitate to contact me.

Support:	http://bit.ly/vk-SupportNew
Forum:	http://forum.unity3d.com/threads/333931
More Assets:	http://u3d.as/5Fb

Thank you for choosing
Smart FP Controller!

If you've bought this asset on the Unity Asset Store,
please write a short review
so other users can form an opinion!

Again, thanks for your support,
and good luck with your projects!

Kindest regards, **Victor Klepikov**