uPlatformer v0.21 12/11/2017

Introduction

Thanks for the purchase and support! We are a community of game developers, working together to create games, experiences, development tools, and tutorials to empower indie developers worldwide. Join us here: https://www.youtube.com/nurfacegames/

Video Tutorials

Intro v0.2: https://www.youtube.com/watch?v=V3bPKFm5U-g
Tutorial v0.2: https://www.youtube.com/watch?v=kB3foWzHAXU

What is uPlatformer?

This pack aims to be the industry standard for building a Unity Platform Game. We are still in development and currently are releasing as an early beta, giving access to new features as they become available. Your feedback is helpful and valuable during beta.

This pack attempts to use as many native Unity features as possible, such as using Unity's CharacterController class, Mecanim Humanoid Animations, and using Unity's Input Manager to handle input from keyboard/mouse or controller. We officially support other input managers such as Rewired.

uPlatfomer Includes:

- Scripts, full source included & well commented
- 15 Animations, Humanoid Mecanim
- Animation Controllers for Mecanim
- Demo Scenes & assets
- Documentation & Video Tutorials

uPlatfomer Features:

- 2.5D / 3D Movement
- Walking & Running
- Slope Handling
- Jumping, Double Jump
- Crouching
- Edge Grab & Climbing Up
- Ladder Climbing
- Camera Follows Player
- Compatible with Unity 5.4.2 - 2017
- Compatible with Unity Input Manager & Rewired
- Compatible with Heroic Traversal animations

Planned Features:

- Multiplayer
- Wall Jumping
- Inventory System
- Enemies & Combat
- Water / Swimming System
- Hang from rope & swinging
- Moving platforms, hazards, etc
- Better animations, more animations

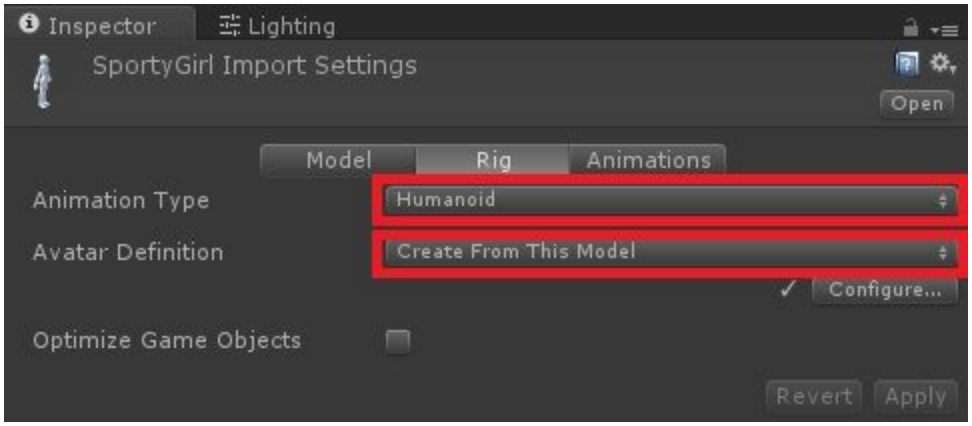
Your support and feedback helps us bring new features and more frequent updates! All planned features will be implemented by version 1.0.

How To Use

Initial Setup

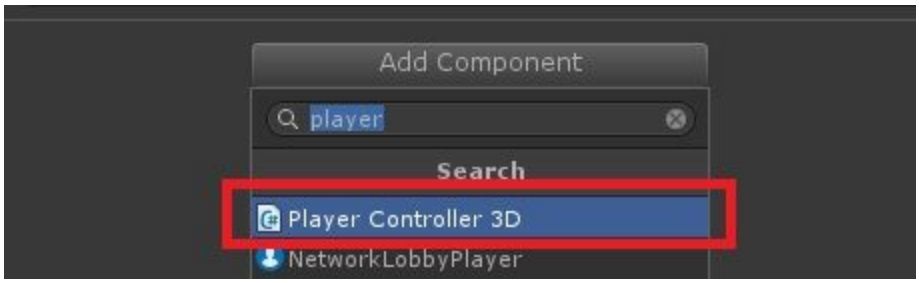
Import the Character Video Tutorial

Your project will require a 3D Character that has a mecanim compatible skeleton. Make sure the Animation Type is set to **Humanoid** and that an avatar has been defined:



Add the PlayerController Script Video Tutorial

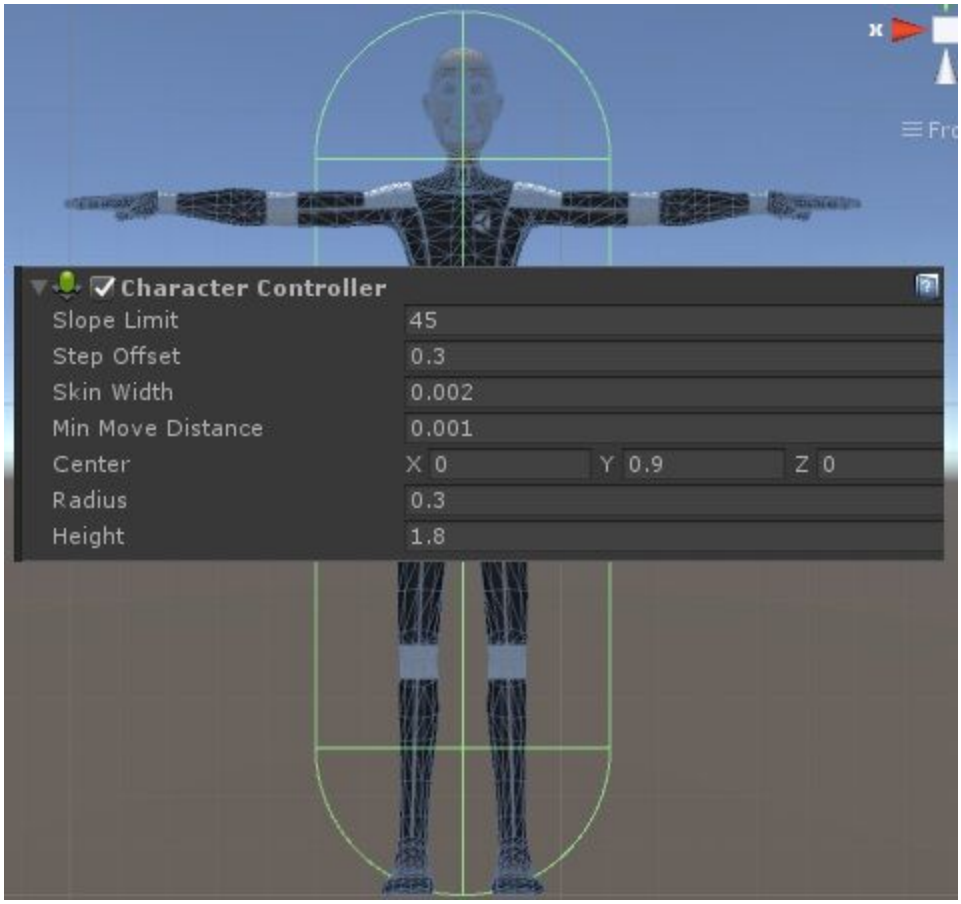
Add the character into your scene, select the character's root gameobject, and add the *PlayerController3D.cs* script as a new component.



This will add the script, a Unity Character Controller component, and an Animator if required.

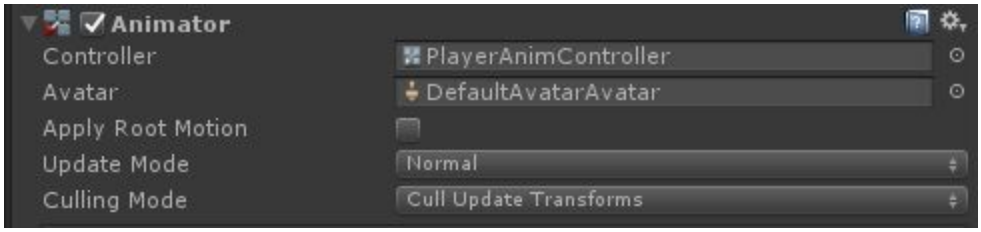
Configure Character Controller [Video Tutorial]

Skin Width: 0.002
Center: 0, [Height / 2], 0
Radius: Match the width of character's body
Height: Match the height of character's body



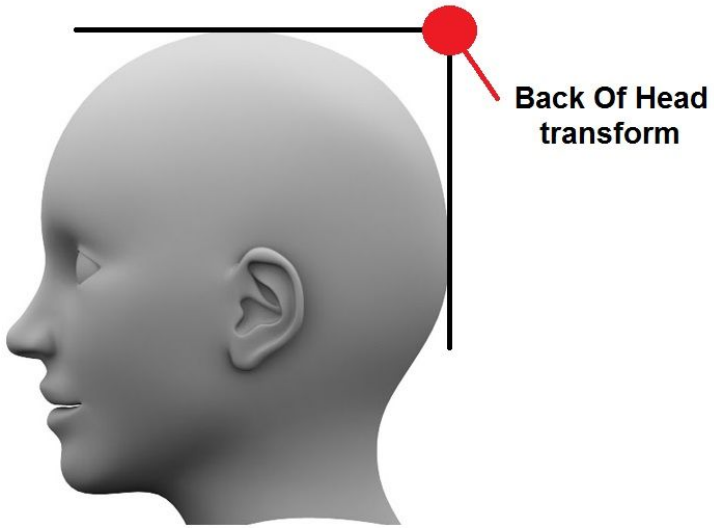
Configure Animator [Video Tutorial]

Controller: PlayerAnimController
Avatar: The avatar that is defined on the 'Rig' tab of your character's 3d model, when setting the Animation Type to Mecanim.
Apply Root Motion: False

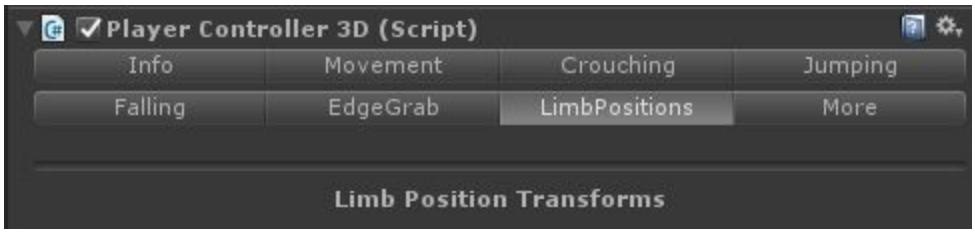


Configure Bone Transforms [Video Tutorial]

The *PlayerController3D* script needs to know where the player's hands are, and the position which is at the **back & top** of the head. The 'Back Of Head' position is marked by the red dot in the following image:

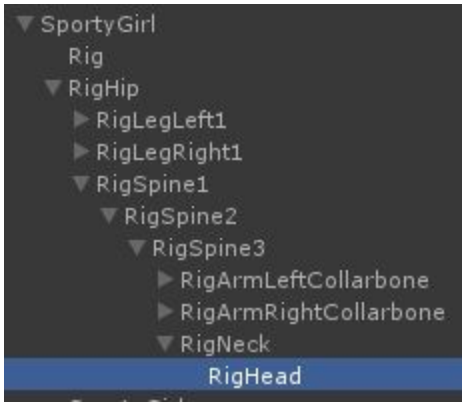


Select the PlayerControler3D script and click on **Limb Positions**:

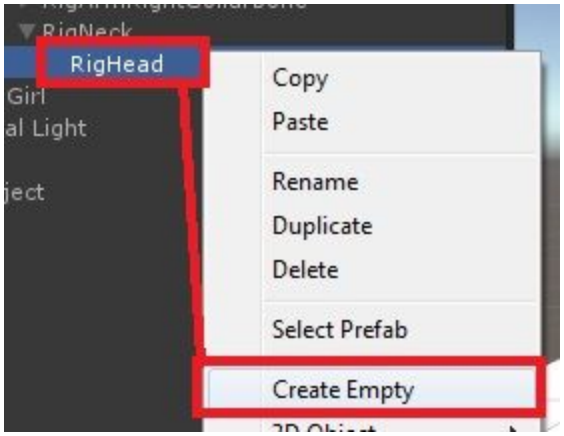


Locate the Hand Transforms. They should be located at
Player Root > *Hips* > *Spine* > *Chest* > *Shoulder* > *Arm* > *Lower Arm* > ***Hand***
An example is shown here:

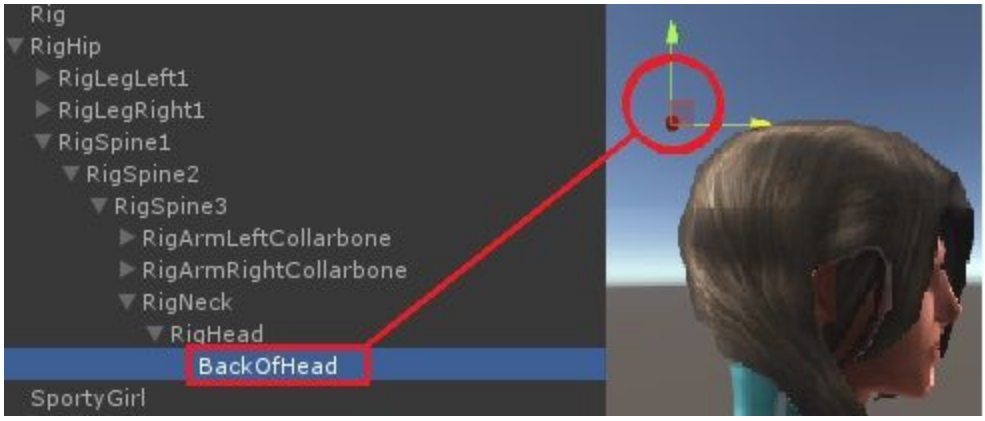
Locate the Head Transform. It should be located at
Player Root > *Hips* > *Spine* > *Chest* > *Neck* > ***Head***
A different example is shown here:



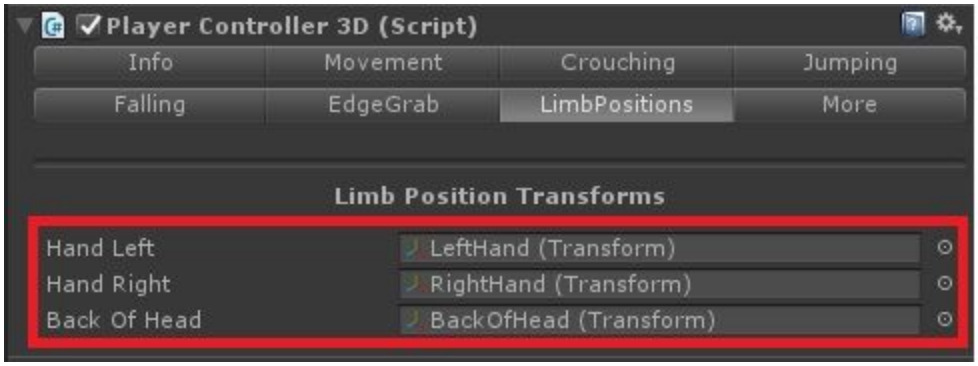
Select the Head Transform and create a new empty gameobject as a child.



Rename this GameObject to “BackOfHead”, and we only need to use the Transform component. As a child of the head, move the GameObject into the position which was previously explained at **back & top** of the head:

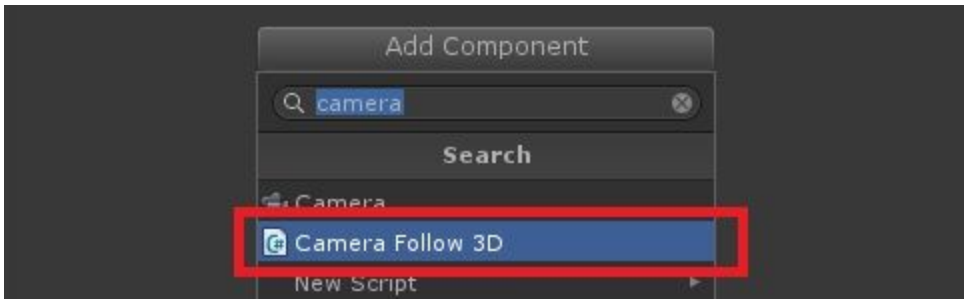


Populate the Left Hand, Right Hand, and Back Of Head transforms on the *PlayerController3D* script:

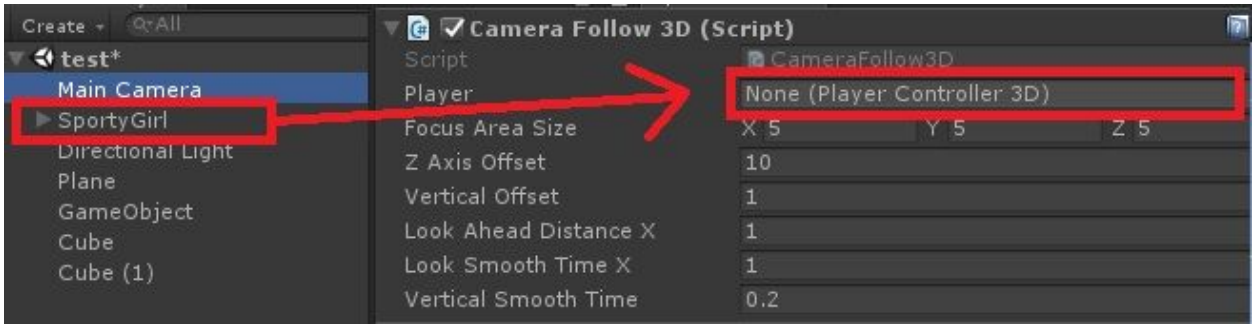


Configure Camera to Follow Player [Video Tutorial]

Select the Main Camera and add the *CameraFollow3D.cs* script to it.



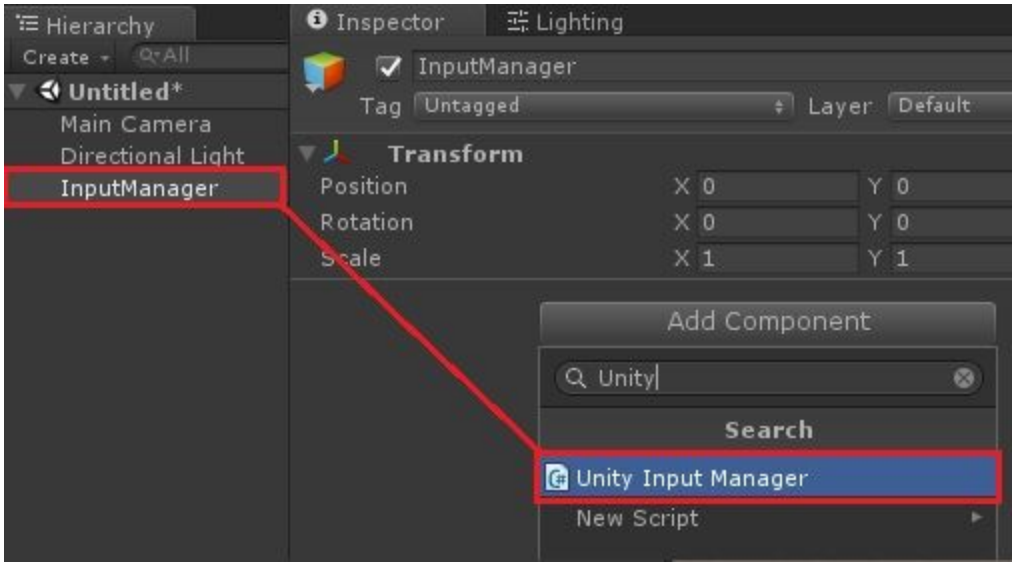
The Camera Follow script will require a PlayerController3D script which it should follow. Add the player into this field.



Add the Input System [Video Tutorial]

Create a new empty GameObject at the scene root:

Reset position to 0,0,0, and name the GameObject “InputManager”. Now add the script UnityInputManager.cs:



The default inputs are set here, you can customize them now:

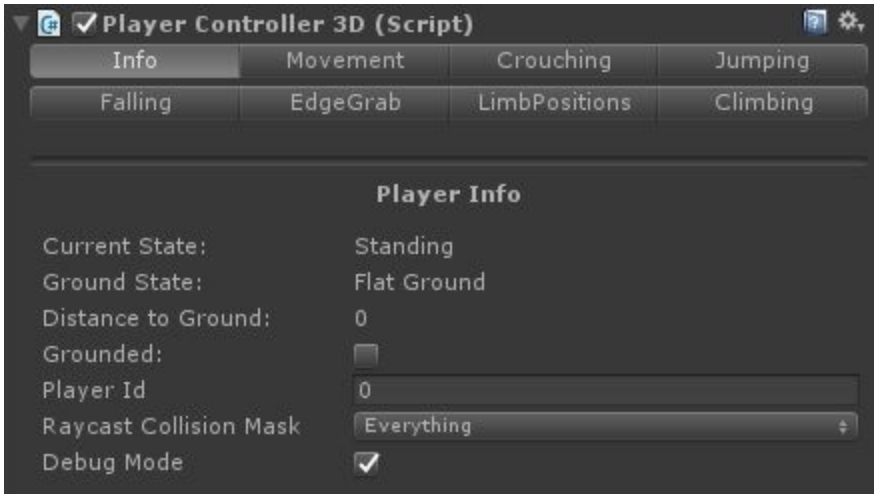
\

Now your character is ready to play!

PlayerController3D Class

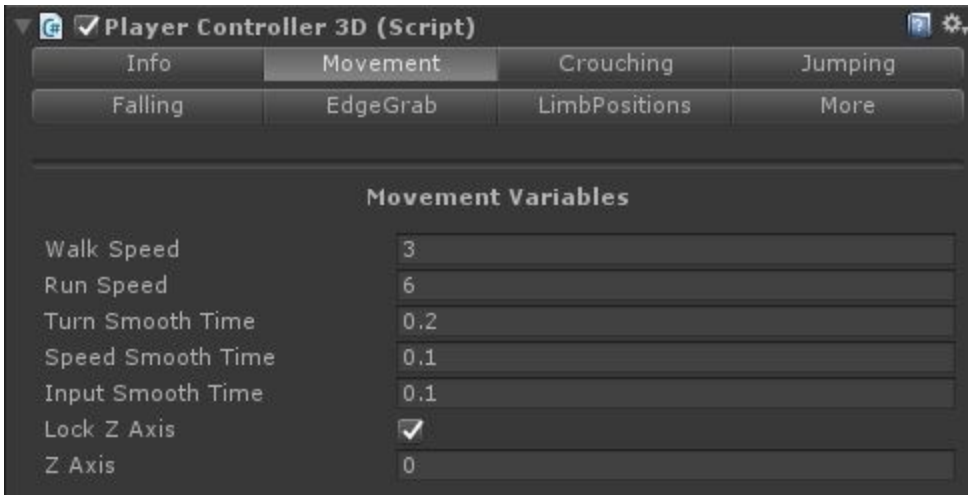
This class is responsible for all player movement for both 2.5D and 3D and should be added to the player's root GameObject.

Player Info [Video Tutorial]



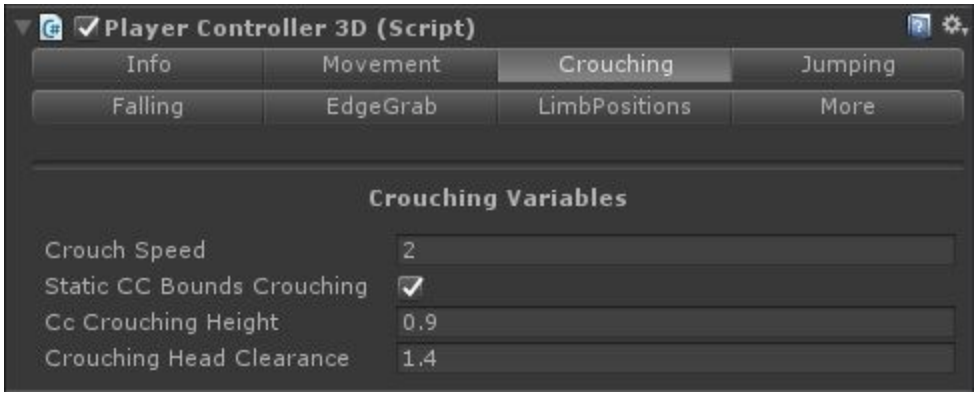
Current State: Shows what state the player is in, values can be: *Standing, Jumping, Falling, Crouching, EdgeGrab, EdgeGrabClimbUp, EdgeGrabDrop, ClimbingLadder.*
Ground State: Shows information about the ground below the player, values can be: *Flat Ground, Uphill, Downhill*
Distance to Ground: How far the player is vertically from the ground below them.
Grounded: A true/false value to show whether the player is grounded or not.
Player Id: Player identification integer. Will be used for future multiplayer. When using Rewired Input Manager, will correspond to Rewired Player Id.
Raycast Collision Mask: All raycasts performed by the character for movement will apply this mask.
Debug Mode: This option will show helpful debug information in the editor window. For example, you can view the bezier curve used for climbing up an edge.

Movement [Video Tutorial]



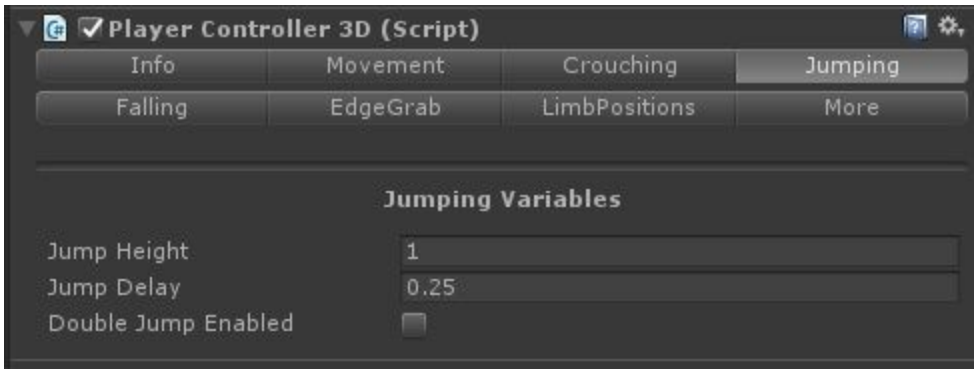
Walk Speed: How fast the player moves when walking.
Run Speed: How fast the player moves when running.
Turn Smooth Time: A smoothing amount applied to turning (rotation) movements.
Speed Smooth Time: A smoothing amount applied to speed changes.
Input Smooth Time: A smoothing amount applied to input axes.
Lock Z Axis: Should movement be locked in the Z axis?
Z Axis: The Z axis the player will be locked into.

Crouching [Video Tutorial]



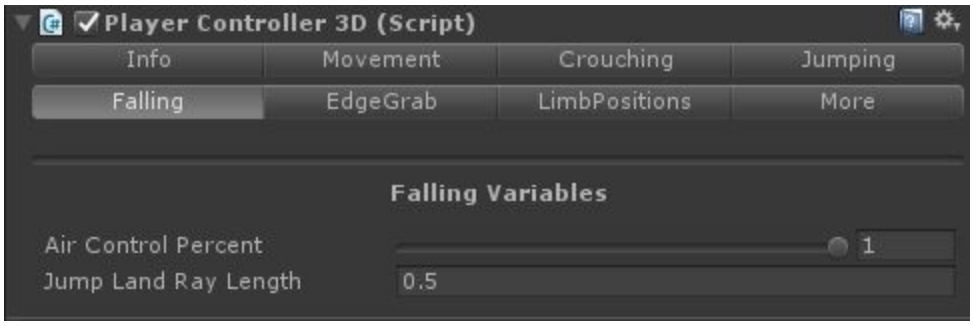
Crouch Speed: How fast the player moves when crouching.
Static CC Bounds Crouching: Static or dynamic CharacterController bounds while crouching.
Cc Crouching Height: The height of the player's CharacterController bounds while crouching.
Crouching Head Clearance: The distance from the ground going up, that it's safe for the player to stand up.

Jumping [Video Tutorial]



Jump Height: How high in Units the player will jump.
Jump Delay: A small delay after jump button has been pressed, before moving CharacterController up, to allow the character's jump animation to 'bend down' in anticipation of a jump.
Double Jump Enabled: If the player can perform a double jump.

Falling [Video Tutorial]

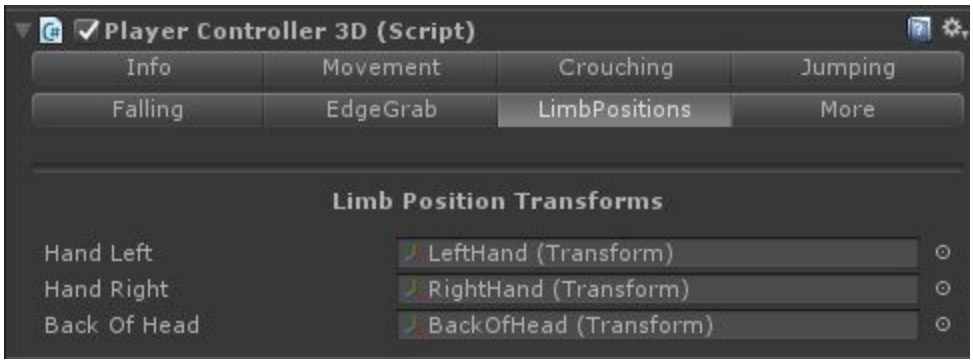


Air Control Percent: How much control, between 0 and 100 percent, should the player have to control their movements while in the air.
Jump Land Ray Length: How far to raycast below player's feet to detect the ground for landing a jump.

Edge Grab [Video Tutorial]

Edge Grab Climb Speed: How fast the player climb up and edge. This is a very important setting for your character to successfully climb up an edge.
Edge Grab Climb Forward: How far to move the player forward past an edge when climbing.
Edge Grab Climb Up: How far upwards above the edge the player will move when climbing up.
Edge Grab Bounds Bottom: From the feet going up, where the bounds area begins.
Edge Grab Bounds Top: From the feet going up, where the bounds area ends.
Edge Grab Bounds Radius: The radius of the bounds area.
Edge Grab Min Edge Size: The minimum size of an edge that player can grab onto.
Edge Grab Ray Count: How many raycasts should be used when detecting an edge.
Static CC Bounds Climbing: Static or dynamic CharacterController bounds while climbing.
Cc Climbing Height: The height of the player's CharacterController bounds while climbing.

Limb Positions [Video Tutorial]

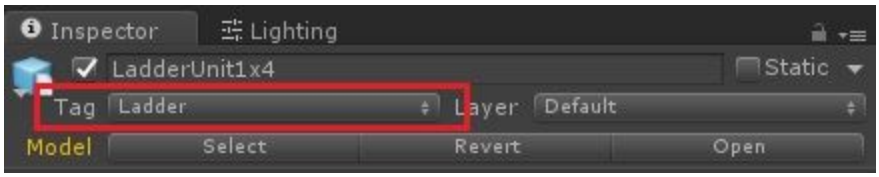


Hand Left: The Transform of the player's left hand.
Hand Right: The Transform of the player's right hand.

Back Of Head: The custom made Transform at the back and top of the head.
See Configure Bone Transforms for more information.

Climbing (Ladder) Video Tutorial

The player can climb up objects which have a collider are are tagged with “Ladder”.

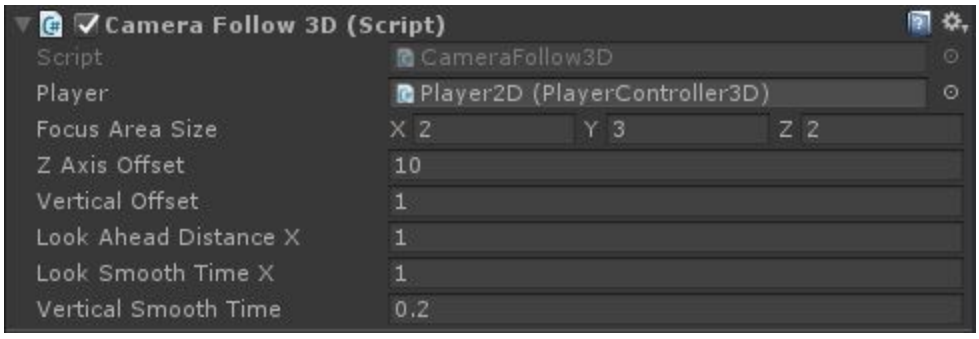


To configure climbing options, select the *Climbing* tab:

Allow Climbing Ladder: Allow the player to climb ladders.
Ladder Cast Length: How far the player will raycast in front of them to check for a ladder.
Climb Ladder Speed: How fast the player will move when climbing up/down ladders.

CameraFollow3D Class Video Tutorial

This Class should be added to the Main Camera.



Player: The Player which the camera should follow
Focus Area Size: The bounds of a box which surrounds the player, the camera moves only when the box moves.
Z Axis Offset: How far away from the player the camera should be on the Z axis.
Vertical Offset: How far above or below the player the camera should be on the Y axis.
Look Ahead Distance X: How far in front of the player the camera should look on the X axis.
Look Smooth Time X: Smooth camera movements on the X axis.
Vertical Smooth Time: Smooth camera movements on the Y axis.

Input System Video Tutorial

uPlatformer uses a modular input system that will work with any input manager of your choice.

- All of the game's actions are defined in *InputAction.cs.*
- The base class *InputManager.cs* is only a partial class and must be implemented by an input manager class of your choice. Unity and Rewired Input classes are included. See *UnityInputManager.cs* as an example.

To use the Input System:

1. First make an IInputManager (InputManager Interface) variable and name it input:

private IInputManager input;

2. Next, in Start() populate the input varible:

input = InputManager.instance;

3. Now you can access the any actions defined in InputAction.cs:

input.GetAxis(playerId, InputAction.MoveHorizontal);
input.GetButton(playerId, InputAction.Run);

A tutorial series covering the creation of this input system can be found here: https://www.youtube.com/watch?v=k1LkeW_od8o&list=PLHQUlKGE3pvy_20H6u1p0lMQJcKwQu_p5&index=1

Unity Input Manager Video Tutorial

Each action defined in *InputAction.cs* is mapped to an axis in the Unity Input Manager. To change which buttons move the player or run, crouch, jump, etc simply enter the name of the Unity Axis which you want to use.

Rewired Video Tutorial

More details coming in Version 0.2 about the Rewired Integration. It's already working and more details can be found in the previously mentioned tutorial series.

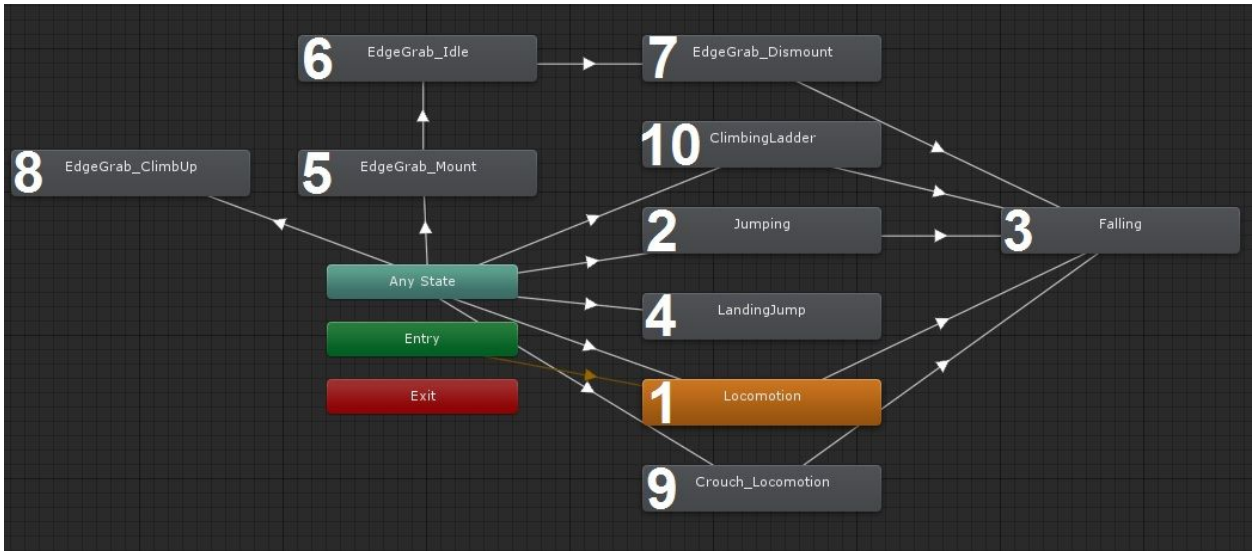
Animation Video Tutorial

There are currently 17 animations which can be used, and 15 are included by default.

Animation Name	**Description**
Standing_Idle	Idle while standing.
Standing_Walk	Walking while standing.
Standing_Run	Running while standing.
Crouch_Idle	Idle while crouching.
Crouch_Walk	Walking while crouching.
Jump_Idle	Jumping while standing idle.
Jump_Walk	Jumping while standing walking.
Jump_Run	Jumping while standing running.
Jump_Land_Idle	Landing a jump while idle.
Jump_Land_Walk	Landing a jump while walking (not included).
Jump_Land_Run	Landing a jump while running (not included).
Falling_Loop	Falling in the air loop.
EdgeGrab_Idle	Idle loop while holding onto an edge.
EdgeGrab_Mount	Grabbing onto an edge from the falling position.
EdgeGrab_Dismount	Dismounting from an edge into the falling position.
EdgeGrab_ClimbUp	Climbing up from an edge.
LadderClimb	Climbing Up/Down a ladder.

Animation Controller [Video Tutorial]

Here you can see where each animation is located within the Animation Controller:



1. **Locomotion:** A blend tree with all standing animations, based on speed of player:
   - Animations used: *Standing_Idle*, *Standing_Walk*, *Standing_Run*
2. **Jumping:** A blend tree with all jumping animations, based on speed of player:
   - Animations used: *Jump_Idle*, *Jump_Walk*, *Jump_Run*
3. **Falling:** This is the 'falling through the air' animation. Animation: *Falling_Loop*
4. **LandingJump:** A blend tree with all 'Landing a jump' animations based on player speed.
   - Animations used: *Jump_Land_Idle*, *Jump_Land_Walk*, *Jump_Land_Run*
5. **EdgeGrab_Mount:** Grabbing onto an edge from the falling position.
6. **EdgeGrab_Idle:** Idle loop while holding onto an edge.
7. **EdgeGrab_Dismount:** Dismounting from an edge into the falling position.
8. **EdgeGrab_ClimbUp:** Climbing up from an edge.
9. **Crouch_Locomotion:** A blend tree with crouching animations, based on player speed.
   - Animations used: *Crouch_Idle*, *Crouch_Walk*
10. **ClimbingLadder:** Climbing up and down a ladder.

- Any state can transition to crouching, locomotion, jumping, landing a jump, climbing a ladder, edge grab mount, or edge grab climb up.

- Only crouching, locomotion, jumping, climbing a ladder, and edge grab dismount can transition into the falling state.

Heroic Traversal Integration Video Tutorial

More details about using Heroic Traversal animations coming in version 0.2. If you have imported the Heroic Traversal pack, you can use the included Animation Controller at: *uPlatformer/Animations/Controllers/PlayerAnimController_HeroicTraversal*

Help And Support

For a video tutorial related to this asset, please click here:
Intro v0.2: https://www.youtube.com/watch?v=V3bPKFm5U-g
Tutorial v0.2: https://www.youtube.com/watch?v=kB3foWzHAXU

Join our VR community here for VR tutorials and videos:
https://www.youtube.com/nurfacegames/

For any questions or support, please email:
nurfacegames@gmail.com