FasTPS

Advanced Locomotion Controller + Climbing System

Table Of Contents:

Youtube Videos:

Showcase: https://youtu.be/5Cv9t-N-xhw
Showcase V2: https://youtu.be/Ttm1KsKqj7E
Setup Character: https://youtu.be/l_1zwyfnYrU

Support:

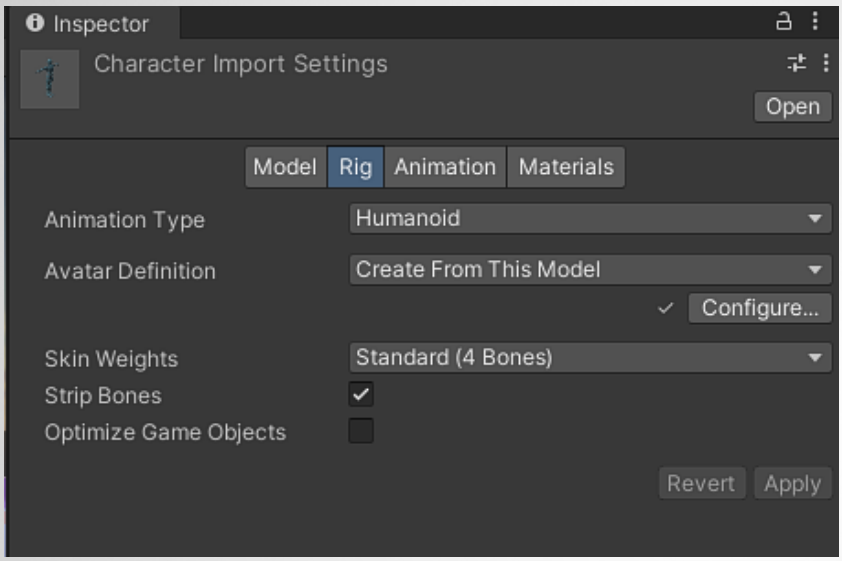
Discord: https://discord.gg/PazMUeeT

SETUP

In the Advanced Locomotion Controller, you can easily create your own controller with your own model using the FasTPS Editor Character Creation System. To use this, your model **MUST** be a Humanoid Rig.

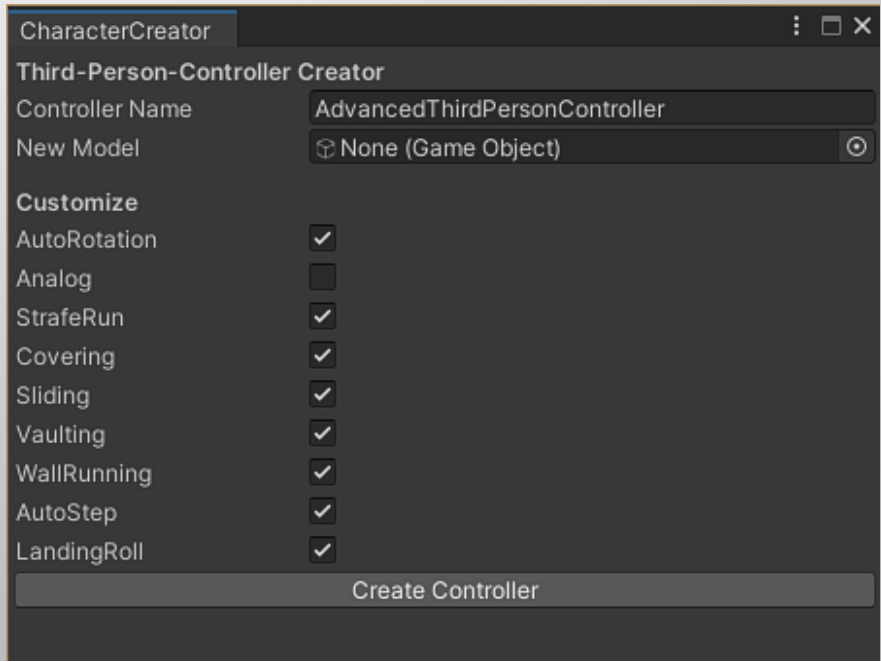
Make your Model Humanoid:

1. Locate your model in the Project Panel
2. In the Inspector locate to the Rig Panel
3. In "**ANIMATION TYPE**" change the drop-down to "Humanoid" and Click, **APPLY.**



Create your Advanced Character Controller:

1. Locate to the following editor window: **FasTPS/Character Creator**
2. In the Controller Name change the Controller to a name, you would prefer.
3. In the **New Model,** Object Field drag and drop the new Humanoid Model that you created.
4. Head over to Customize and change the settings to as you wish. The Features are explained on the features page.
5. Then click "Create Controller" and your controller will show up in the scene with the name you chose.



Tags/Layers

In the Advanced Locomotion Controller, we utilize tags and layers to make development easier, but these tags must be used in the necessary places.

Ground Layer:

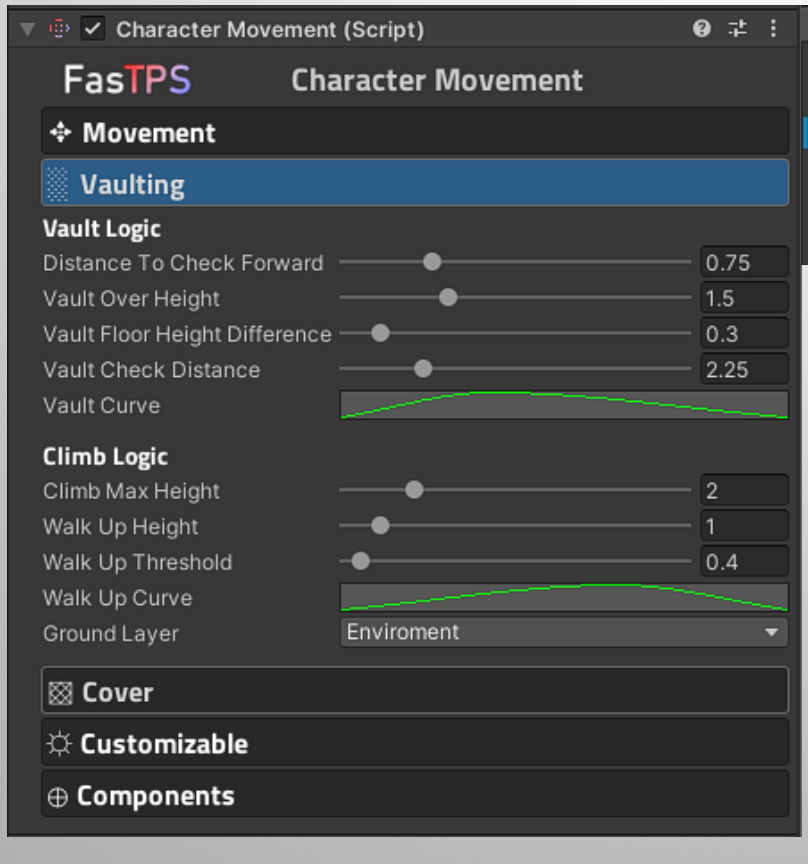
In the controller, by default, the Ground Layer is set as "Environment".

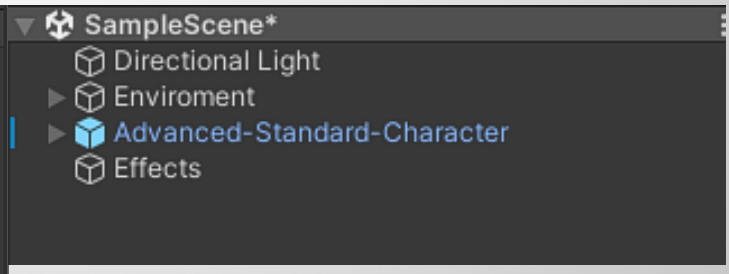
Change Ground Layer:

1. Locate the Environment object in the scene and head to the inspector.
2. Under Layers, create your own layer and click the select this layer for the Environment Object.
3. Click **YES** to change the layer for all child objects as well and then you have successfully changed the layer.

Use New Ground Layer:

1. Locate your TPS Controller which is under your New Character Models Parent.
2. Locate the **Character Movement** script on the controller and head over to **VAULTING.**
3. Click Vaulting and under vaulting scroll down and change the **GROUND** layer to the new layer you have created, then save the prefab.





Tags/Layers

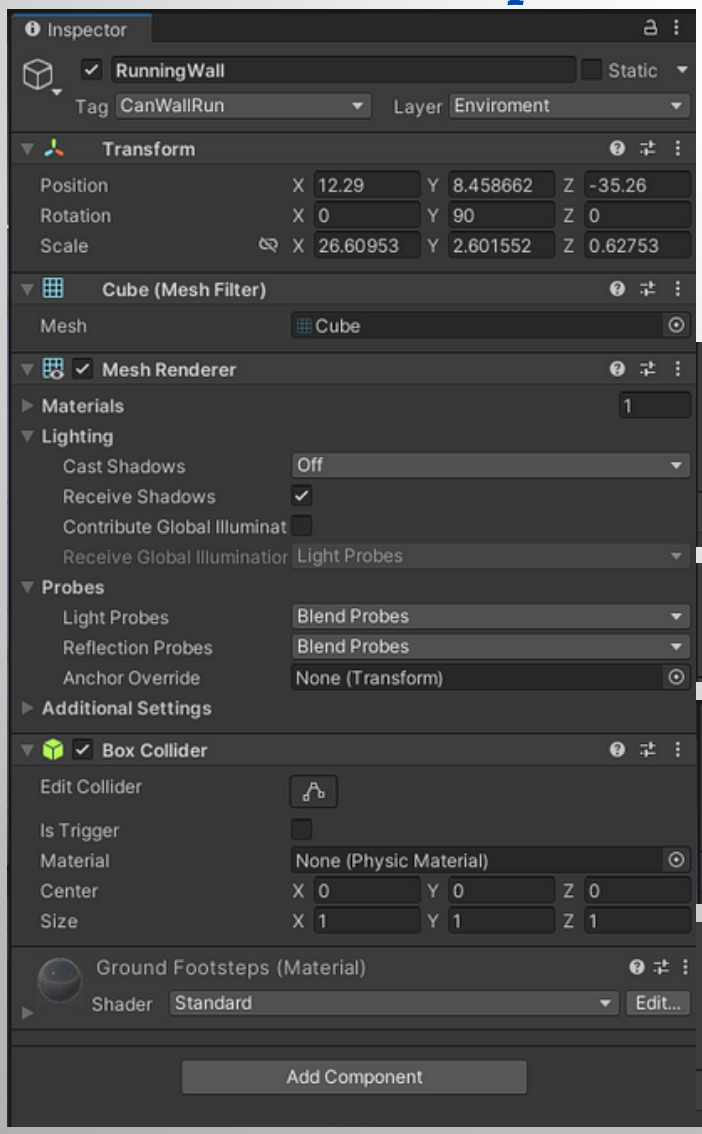
Climbing Layers:

In the Controller, every wall that is climbable requires a collision component on the wall. This specific component has to be set with the layer **"Climb Points",** our the controller will not start climbing if it does not recognise a layer by this name.

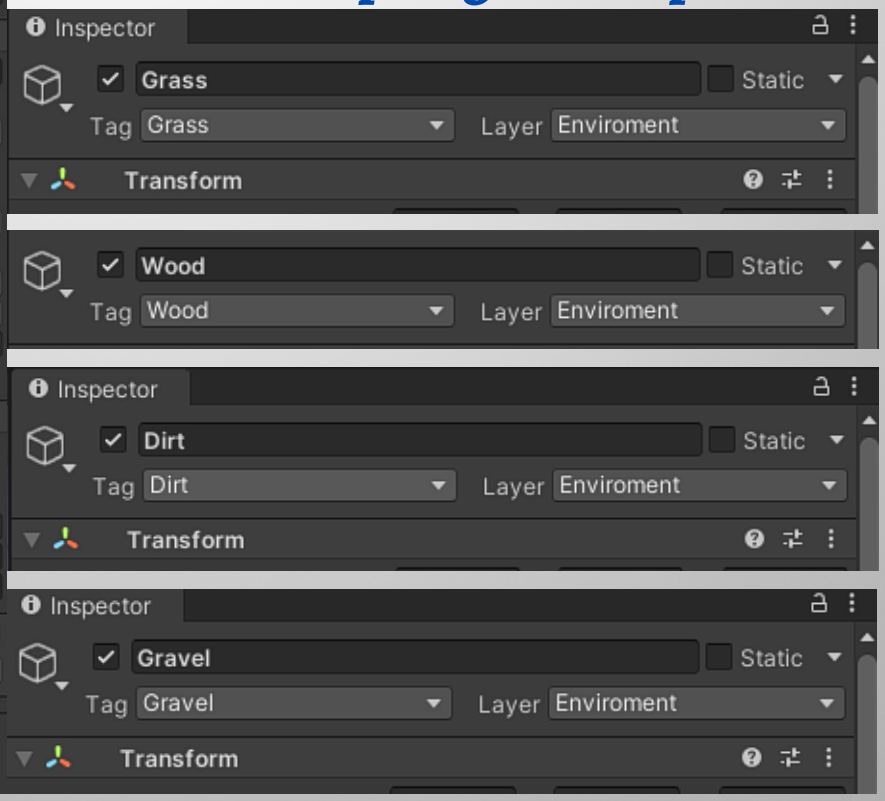
Footstep Tags:

In the controller, the footstep system drives from tags. To allow the footstep system to work on tags, you must make sure each object has a collider that is set to **"IS TRIGGER"** and each object has one of the respective footstep tags: **Dirt, Wood, Grass, Gravel.** If the controller does not detect a tag, it defaults the tag to **"Concrete".**

Wall Run Example



Footstep Tags Examples



Features

In the Advanced Locomotion Controller, there is an abundance of features to allow individuals to customize their gameplay to the feeling that they want. The main features can be found in the **Character Movement** script under the **CUSTOMIZABLE** Toggle which can be found in the TPS Controller under the parent of your player.

Feature Run-Down:

In this controller, there is a multitude of features which include:

- **Auto Rotation:** This feature is utilized when the character rotates in one specific direction, with their camera, the player model will lerp at the same speed in this direction.
  On: The player will rotate in the direction the camera faces with the speed located under the **Movement** Toggle in Character Movement
  Off: When the player presses the input the character will **THEN** move to the direction the camera is facing, instead of moving automatically.

- **Analog:** This feature is utilized so the character will not auto rotate but will face in the direction that they are moving. This is used so that only one animation is played in every direction which is the "**Forward**" animation and the other ones are not played.
  On: The player will rotate to the direction that the input is so if they are going left, the player will turn left and player the forward animation while moving left.
  Off: This will ignore the analog procedures and if the player presses left it will just play the strafe animation.

- **Strafe Run:** This Feature will play a strafe running animation when strafing is toggled.
  On: If Sprinting and going left/right, the player will play a strafe run anim.
  Off: If Sprinting and going left/right, the player won't sprint and will stay strafing at the constant walking speed.

- **Covering:** This feature is utilized when the character is facing toward a wall and toggles using the Covering Keybind, they will enter the cover state if the wall is at the correct height.
  On: The player will cover if there is a wall in front of them and they are pressing the correct input to activate the covering mechanic.
  Off: If the player tries to toggle the cover button and there is a wall in front it will ignore the input and keep it at the same state that they are on.

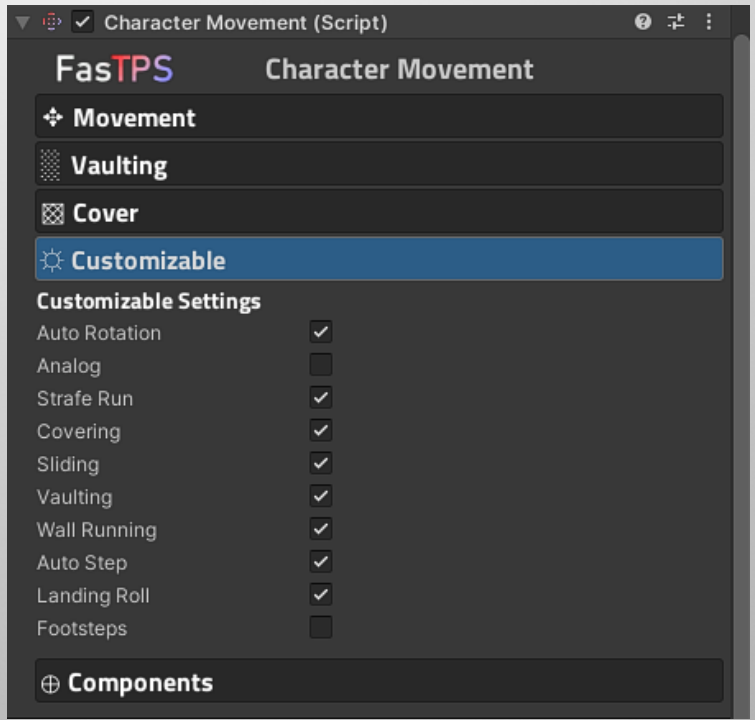
- **Sliding:** This feature is a common mechanic in many fast-paced games. This allows the player to slide at a specific speed that you specify for a specific amount of time that you specify.
  On: When the player presses the corresponding slide input they will perform the slide movement mechanic.
  Off: When the player presses the corresponding slide input it will ignore the input and stay in the same state as the last tick.

- **Vaulting:** This feature allows the user to use the automatic vaulting feature when toggled on.
  On: When the player moves in front of a barrier that has clear ground on the other side, they will vault over this barrier.
  Off: When the player moves in front of a barrier they will stay at the same state as the last tick and not vault over the barrier.

- **Auto-Step:** This feature is utilized when the character is facing toward a step that **DOESNT** have clear ground on the other side but is elevated to the same height of the step.
  On: The player will automatically step when there is a step in the way and they cannot vault on this step.
  Off: If the player turns off this toggle, when they are facing a step it will ignore the step and stay in the same state as the last tick.

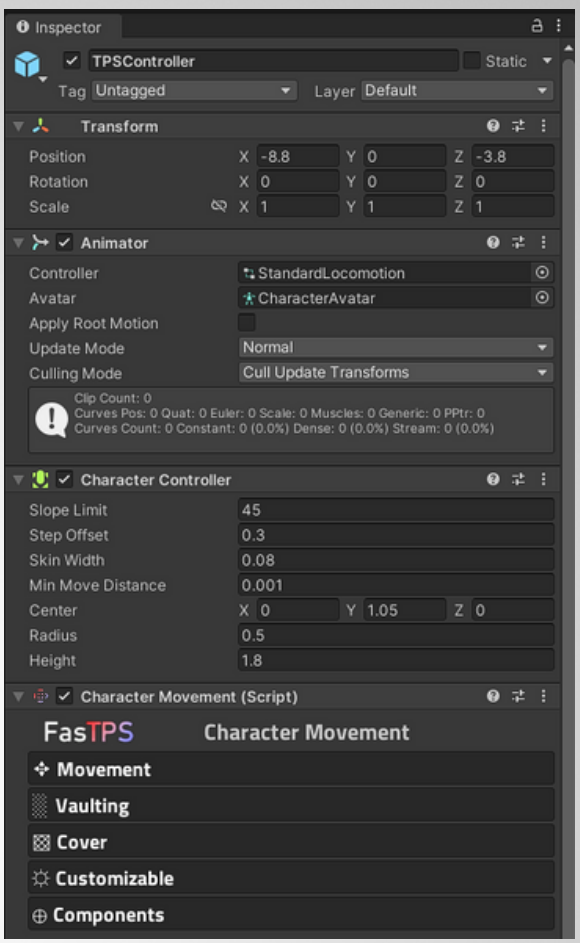
- **Landing Roll:** This feature checks the velocity of the player when in the air and if it exceeds a specific number it will perform a **LANDING Roll** which is a roll that replaces the original land animation.
  On: When the player exceeds this velocity and hits the ground it will play the landing roll animation.
  Off: When the player exceeds this velocity and hits the ground it will play the standard landing animation.

- **Footsteps:** This feature is driven by tags and allows the user to play footsteps sound when walking on dynamic ground layers. This footstep sound will be triggered with this toggle.
  On: The player will play these dynamic footstep sounds if the player **HAS** this feature toggled.
  Off: The player will not play any footstep sounds if this feature is not toggled.



Character Movement

In the Advanced Locomotion Controller, the state-of-the-art movement allows for dynamic gameplay for the user. Using custom editors the process of customizing has never been easier. The run down of each feature is below.



Movement:

Move Speeds:

Move Speeds allows the user to customize 5 different speed types. These include Walk, Crouch, Sprint and Slide Speeds. The Slider for each maxes at 10 but can be easily changed. The speeds are decided based on the current state that the character is in and the features selected.

Move Options:

Move Options allows the user to vary 5 different move options.

- **Jump Height** is the force applied to the character when jumping.
- **Backward Speed Drop-Off** is the decrease in speed when the character is moving in a backward direction.
- **Gravity** is the downward force applied to the character to keep it on the ground.
- **Ledge Height Fall Roll** is the minimum Height the character needs to fall from for the falling roll animation to play.
- **Ground Distance** is the radius of the physics sphere to check whether the character is on the ground.

Rotation Speeds:

Rotation Speeds allow the player to decide what speed they would like the player to lerp at depending on which direction the camera is facing. This feature can also be changed depending on the options you toggle in the customizable area.

Vaulting:

Vault Logic:

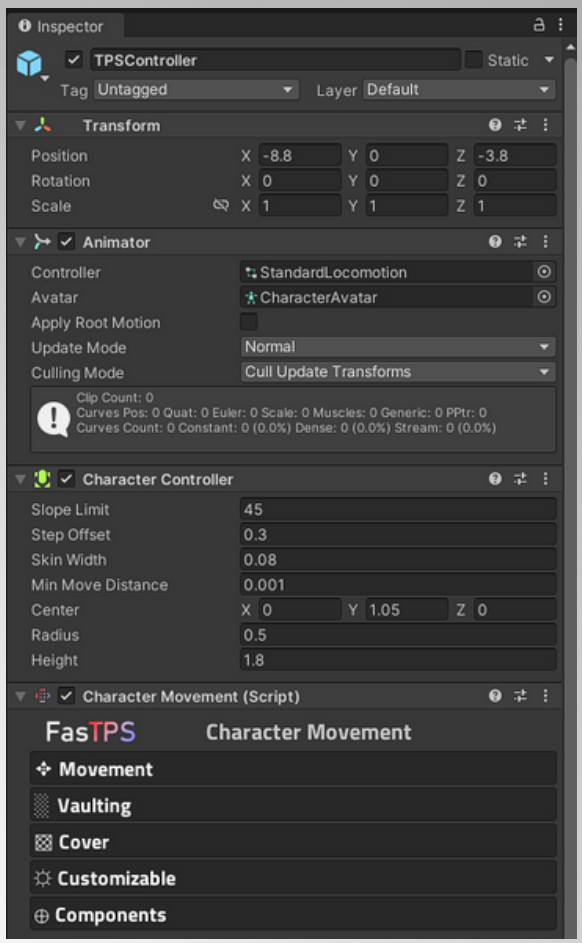
The vault logic allows the user to change a few features of the vaulting system.

- **Distance To Check Forward** the Maximum Distance from the player to the wall.
- **Vault Over Height** is the maximum height of the wall
- The **Vault Floor Height Difference** is between the player and the landing spot.
- **Vault Check Distance** is the distance to the other side of the wall to check that the wall isn't a block and you can vault.

Climb Logic:

The climb logic allows the user to change values like the vault system to customize the movement.

- **Climb Max Height** is the maximum height allowed from the ground to the object.
- **Walk Up Height** is the Minimum height for a player to be able to climb.
- **Walk Up Threshold** is the Maximum angle slope that the player can walk upon.



Components:

These are the non-negotiable components that don't necessarily need to be touched unless important as they are constant even after character creation

Covering:

Cover Logic:

The cover logic allows the user to change a few features of the covering system.

- **Wall Distance** is the Minimum distance from the player to the wall when covering

Wall-Running Logic:

The wall running logic allows the user to change the values of the wall running capabilities to allow for clean and dynamic wall-running features.

- **Minimum Jump Height** is the height from the ground that the player needs to be able to wall run on a specific wall.
- **Wall Run Gravity** is the force applied to the player towards the direction of the wall to keep the player on the wall.
- **Wall Run Jump Force** is the force applied when the player tries to jump on the wall.

Animation Manager

The animation manager plays a vital part in the growth of this asset. Though it is a simple script all the background work is done for you and the animation manager only leaves you with two variables to customize, which highlights the ease of using this.

Animation Speeds:

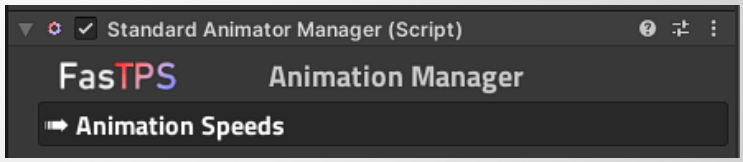
- **Acceleration** is the speed or force that is applied for the character to go from one state to another in the blend tree. Hence the character will blend from walking to running faster if the acceleration is higher number. Usually, the preset numbers are perfect and allow for smooth transitions but this can be easily changed.

- **Deceleration** is the speed of force that is applied to the character as they leave a state. This may be from running to walking the blend speed is the deceleration value. Usually, the preset numbers are perfect and allow for smooth transitions but this can be easily changed.

Animations:

The animations are easy to find and can be found in the following path: Assets/Setup/Player/Animations.

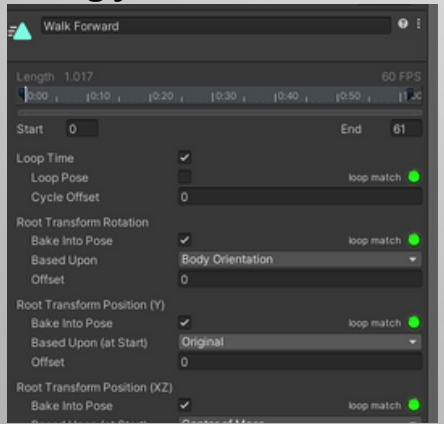
This path includes all the animations included in the asset and some states such as crouching and jumping have a few variants for you to test around and change depending on what animation you like.

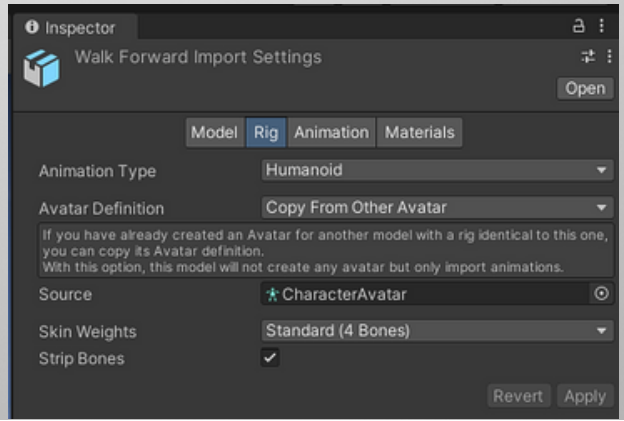


Add your Own Animations:

To add your own animation is quite simple.

1. Drag and drop your animation into the animations folder.
2. On your animation click on it and in the Rig, Panel Set it to Humanoid but unlike the characters in the avatar, the definition changes it to **COPY FROM OTHER AVATAR.**
3. Then Change the avatar to **CHARACTER AVATAR** and click apply.
4. In the Animation Panel on all three ROOT TRANSFORMS change them to **BAKE INTO POSE** and **ORIGINAL** Based Upon.
5. Then drag your animation into the correct state on the Animator.





Advanced Foot IK

The Advanced Foot IK system allows for Dynamic Foot IK Positioning and Body Positioning with an abundance of features.

IK Values:

- **Height Raycast** is the Raycast from the sky which is checking that there is an object below
- **Raycast Down Distance** is the Raycast length that is checking the maximum height allowed for this object before it is "Too High"
- **Pelvis Offset** is the offset of the pelvis from the body positioning values but it is usually set to zero unless not preferred.
- **Pelvis Move Speed** is the blend speed of the pelvis from one position to another.
- **Foot IK Move Speed** is the blend speed of the foot from one position to another.

Advanced Curves:

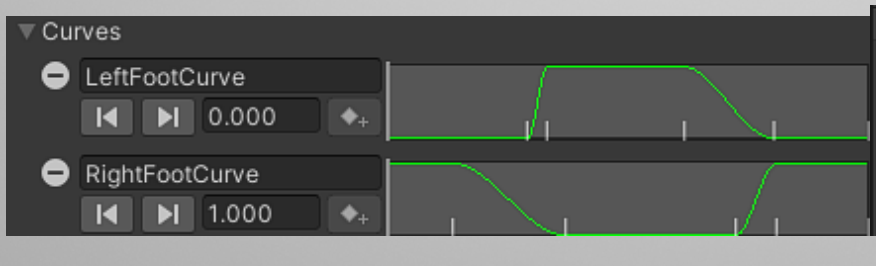
- **Left Foot Curve** is the name of the Left Foot Curve in your animations.
- **Right Foot Curve** is the name of the Right Foot Curve in your animations.

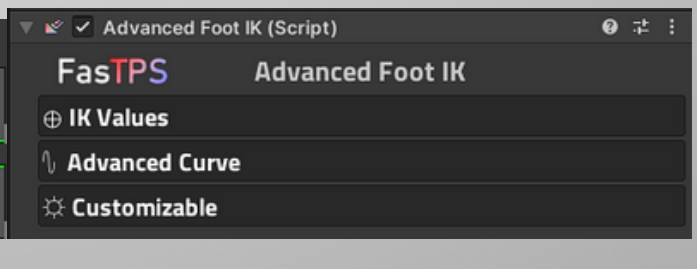
Customizable:

- **Enable Foot IK** is the basis for when you would like to Enable the Foot IK System.
- **Enable Pro IK Feature** allows for rotations of the foot when on slopes and curved objects like spheres and domes.
- **Show Solver Debug** shows the gizmos for the IK and what is happening behind the scenes.

Setup Animation With Foot IK Curves:

1. **Locate the animation in the project panel and locate in the Animation Tab. Scroll down to the bottom and locate curves and create two curves "Left Foot Curve" and "Right Foot Curve". Watch the animation and when each foot is on the ground set the curve to 1 and when they are in the air set the curve to 0. The Curves should look like polar opposites if it is a moving animation.**





Footsteps

The footstep system allows for dynamic footsteps using tags. It can be found on the FootStepChecker Object which is located in the following path: **Model > TPSController > Other > Items > FootStepChecker.** This object holds a Box Collider and the Standard Footstep Manager Script.

How Does It Work?

The Box Collider on the Footstep Checker interacts with an **IS TRIGGER** collider on another object may it be the ground, and this is then made into a string that holds the current ground. Each Animation on the character has an **ANIMATION EVENT** which is relayed to the main character who calls the play function on this object.

Footsteps:

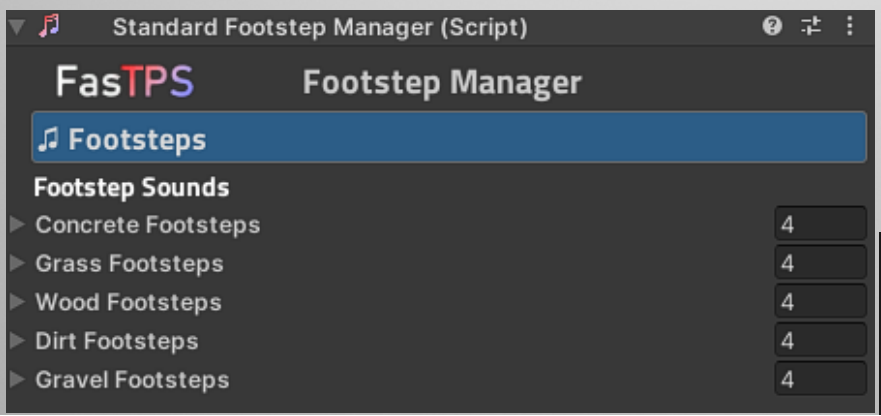
Each ground has 4 sounds which allow for 2 Walking Sounds and 2 Running Sounds. The sounds are chosen at random using an integer and are played at random to add diversity. Each sound is on an Audio Source Object and this object can be found in the following path: **Model > TPS Controller > Other > Footsteps.**

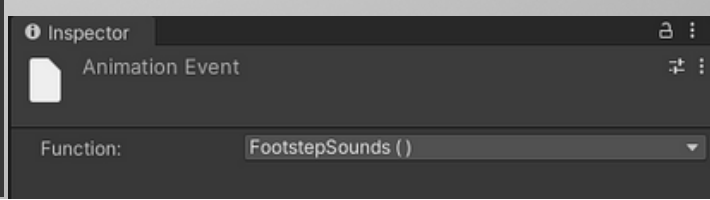
Add Your Own Footsteps:

1. When you find your footstep sound, locate the Audio Source Footstep Objects in the following path: **Model > TPS Controller > Other > Footsteps.**
2. Find the ground type you want to replace. Each Object should have four footsteps underneath. 2 are called **Audio Source Walk** and 2 are called **Audio Source Run**.
3. Find the one you want to replace and drag your audio clip into the **Audio Source's Audio Clip** and you are done.

Setup Animation Clip With Footsteps.

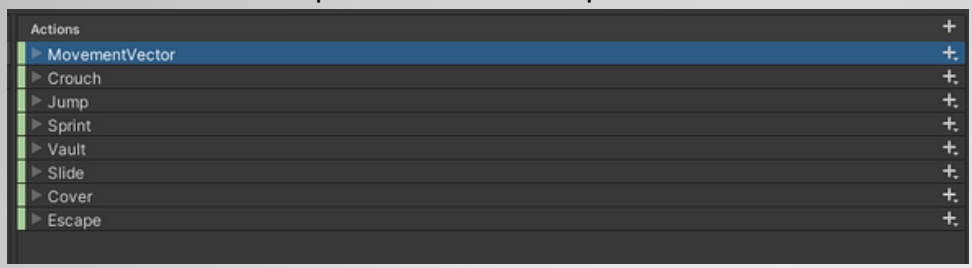
1. Locate your TPS Controller in the scene and open the animation panel. It can be found in **Windows/Animation/Animation** or Ctrl + 6.
2. Find the animation that you added and open it up.
3. In the inspector watch, the animation clip and every time the character puts its foot on the ground make an animation event.
4. In the event set the function to **FootstepSounds** on every event in the animation and you are done.





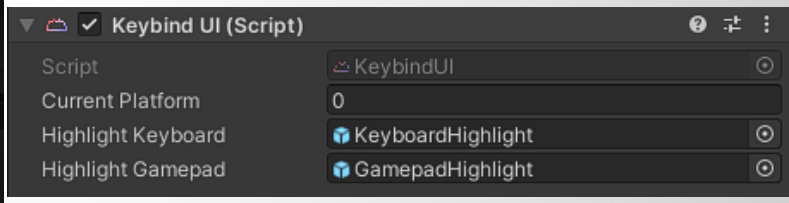
Player Input

Player Input is the basic and simple script that relays input that is down by the player to the character controller. It builds on the New Input System and uses the Input Action Map which can be found in:**Assets/Setup/Controls/PlayerControls.inputactions.** Bindings can always be added to the action map but you must set them up in the Player Input Script. This can be done similarly to the other inputs but inside the function, you want to change the process. There are **NO VARIABLES** require for this script.



Keybind-UI

Keybind-UI is a simple but necessary script to allow for the panel to open and close succinctly while checking what UI we are on. In the bottom left the Gamepad and Keyboard buttons from the Rebinding panel change what input we are changing. This Keybind UI script just makes sure that what we are changing is that HIGHLIGHT that is on. The two variables are Gameobjects.



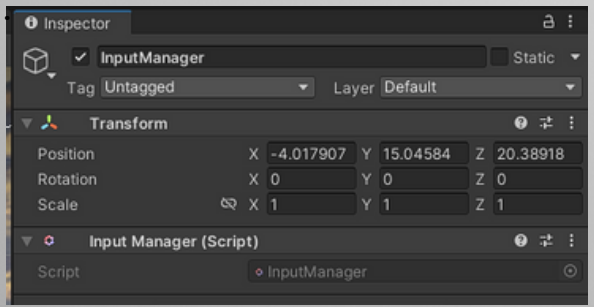
Input Manager

The Input Manager is a static object in the scene that saves and loads the persistent keybinds with no variables required. It is called upon by the Rebind Scripts when it wants to rebind and it performs the rebound on this specific script. The input manager doesn't require any altercations but if you want to exclude specific inputs from the rebind script:

1 Locate the Do Rebind Function.

2. At the bottom write rebind.WithControlsExclding(Input That you want to exclude).

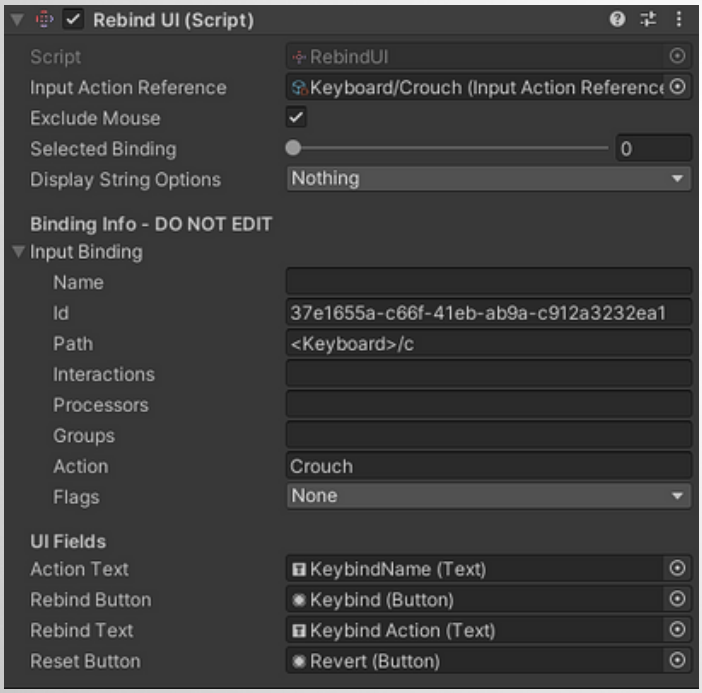
3. The inputs that you can exclude include, Gamepad, Mouse, and Keyboard.



Rebind UI

The Rebind UI is the script that handles all the changing of the UI during the gameplay. Each Keybind is located in the following path: Model > TPS Controller > Other > Canvas > Keybind Panel and one of the vertical layouts has a Rebind UI Script on it.

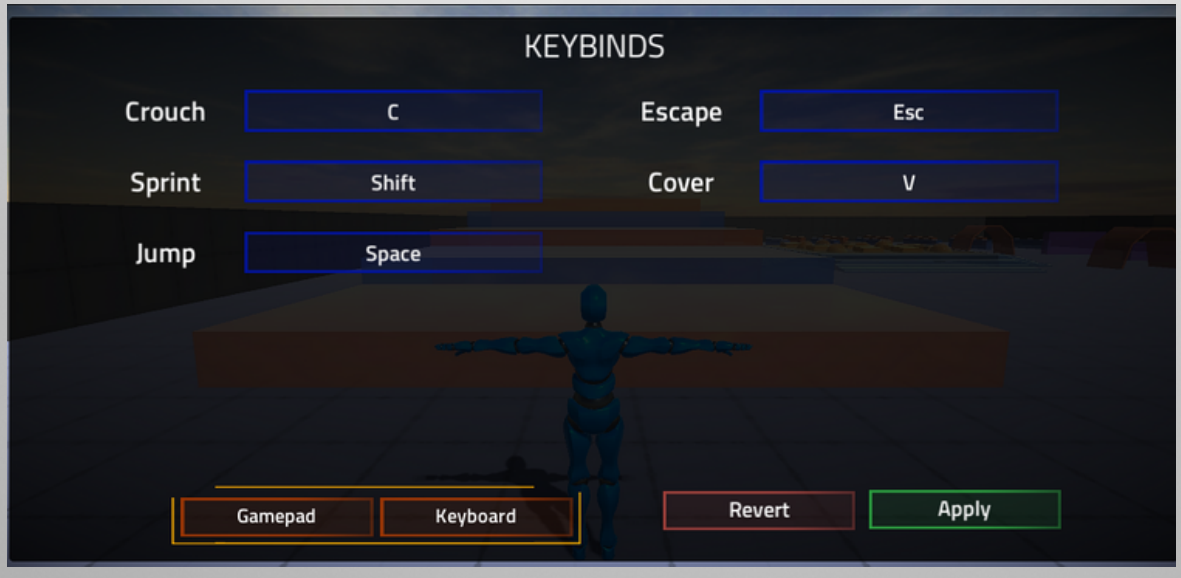
- **Input Action Reference** is the Input action may it be Crouch, Jump, or Sprint from the input action map.
- **Exclude mouse** is a bool which means that in the specific action, the mouse input won't be registered as a binding.
- **Selected binding** shows all the bindings for this binding so the Gamepad and the Keyboard binding which is how you switch between bindings in the ui.
- **Display String Options** is an optional variable that can be tinkered with.
- **Binding Info** is the info about the action binding that you chose and this doesn't have to be changed and you MUST not change it.
- The **UI Fields** at the bottom identify which UI is responsible for which procedure of the binding and this doesn't have to be changed as well.



How to add new binding

Adding a new binding is easier than ever possible.

1. First, you must create your binding in the input action map.
2. Then duplicate one of the existing Keybind UI's and locate this binding under one of the layout groups.
3. Change the Input Action Reference to the new binding that you create and then you have successfully created a new binding.

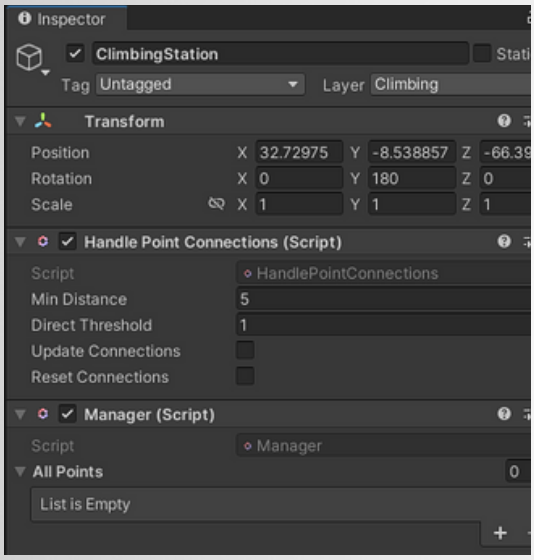


Climb System - Setup

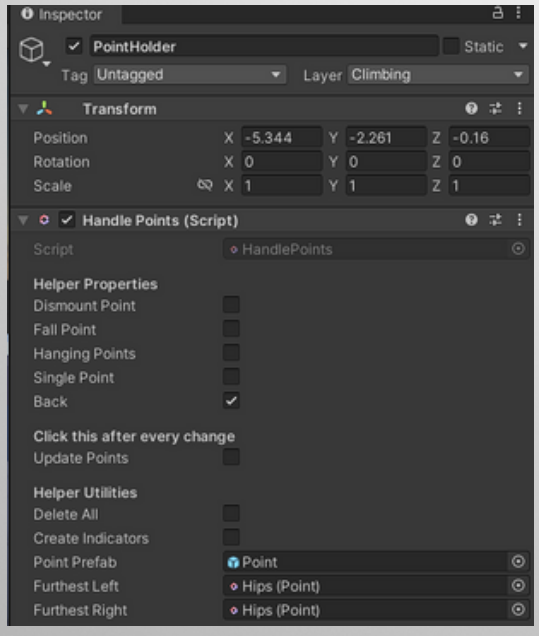
In the Advanced Locomotion Controller, you can easily use and modify the dynamic climb system that is included in the asset.

Create a Level:

1. Create a Wall Or Box to start your CLIMBING level, and change the layer type to **"CLIMB POINTS".**
2. Add a Few Beams around the wall in your own configuration, and make an empty game object and drag all your CLIMB SYSTEM components under this object
3. To this object add the **Handle Point Connections** and **Manager** Scripts



4. Then Create an empty Child Gameobject called **"Points"**
5. Create an empty Object under the Points object and name it **Points Holder** and add the **Handle Points** script to the object.
6. Click the Boolean **Create Indicators** on the Handle Points script in the inspector, and drag the furthest left point to the furthest left part of the beam, and the furthest right point to the furthest right part of the beam.
7. Now, Click Update Points on the script, and all the points in between will automatically generate.
8. If you would like your points to be Dismount Points click the **Dismount Points** bool. And Vise Versa for **Hanging Points** and **Fall Points**.
9. Duplicate this method for the rest of the wall and create multiple ledges. You do not have to use the automatic point generation system and you can manually place points. Although it is included for your ease.



Climbing Movement

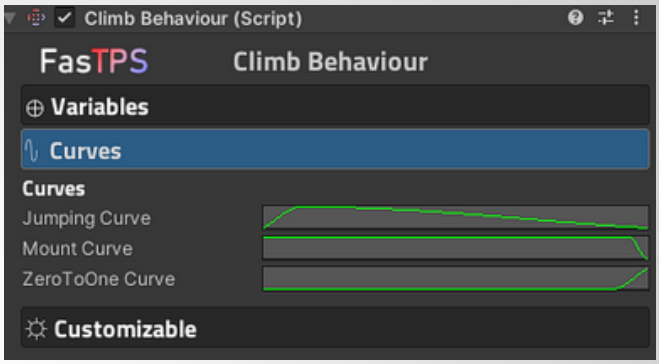
In the Advanced Locomotion Controller, there are many movement types when climbing as it is an 8-Directional Climb System.

Inputs - Climb System

WASD - Move Around Linearly
Space + W - Swing Forward
Shift + S - Jump Backwards
Shift + W - Turn Around Hanging

Types Of Movement

Direct - Jumping Between Points (Threshold)
Linear - Linearly Between Points
Split - Between Two Points Linearly
Jump Back - Jump To Point Behind
Dismount - Go to the top of the wall
Drop Ledge - Fall from wall to climbing
In/Out Corners - Climb In/Outward Corners
Swing Forward - Swing to Point in Infront
Turn Around - Turn around while Hanging
Diagonal Move - Move on an angle (ie. NE|NW)



Curves

Jumping Curve - Jumping Back and Forth Between Points
Mount Curve - Degressive curve when Mounting
Zero To One Curve- Curve For Dismount and Dropping Ledges

Curves Holder

The curves holder holds a series of curves that allow for the Climbing system to move in unique ways following a sequential curve. These curves allow for the climbing system to not clip through walls when performing simple tasks such as jumping from ledges and jumping through different hanging points.

Climbing IK

In the Advanced Locomotion Controller, includes an intuitive IK solution which allows for the character to move swiftly and dynamically through different points.

Climb IK

In the climb IK system the character uses four Points on each Point Prefab to choose where the Left Hand, Right Hand, Left Foot and Right Foot should be positioned throughout the climbing phase of the character.

Customize positions

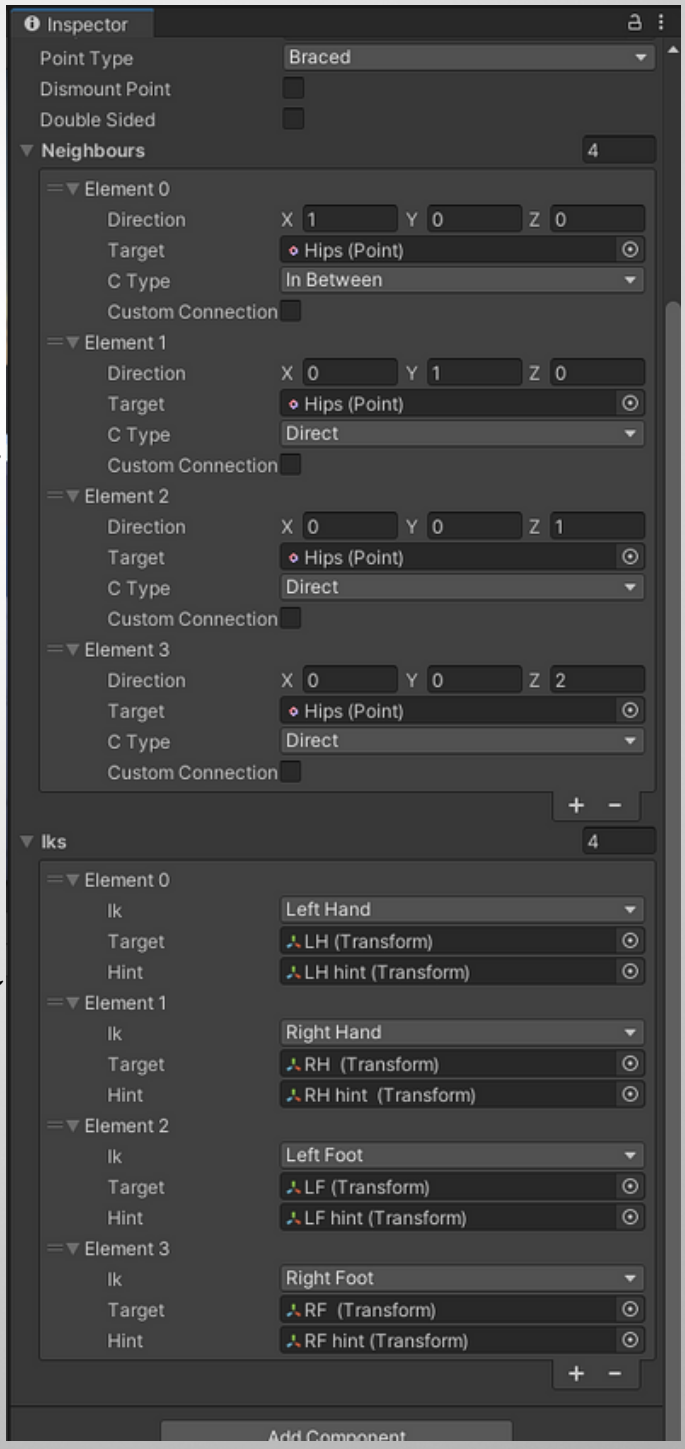
If you are unsatisfied with the positioning of the characters limbs using the Ik system there are a few steps that you can follow to change these positions.

1.Take the current point the character is on and find the ik positions in the point script in the inspector

2.With this point now take the IK target you want to modify and move it around to your ideal position.

3.Now go the position and copy it then find the point prefab in the project and paste the values in.

4. Every point now that is created will contain the updated point positions and ik positions for the character.



Climb Behaviour

The Climb Behaviour script is the power house of the climb system and this will run through what all the variables of the script mean/do.

Variables

Root Offset - The offset of the root from the Point's original Position
Linear Speed- The Speed at which the character moves when moving through points linearly
Direct Speed- The Speed at which the character moves when jumping directly between points
Drop Ledge Speed - The speed in which the character moves when dropping from the top of the wall back to the nearest ledge.

Curves

Jumping Curve- The Curve in which the character jumps from point to point
Mount Curve - The Curve in which the character follows when it is first mounting onto the wall
ZeroToOne Curve - The Curve in which the character follows when dropping from the wall to the ledge.

Customizable

EnableRootMovement- Required if you want your characters movement to be lerped else you will require your own movement system,

Point

The Point Script is the script that assigns, IK Targets and Neighbours to each point automatically.

Componenets

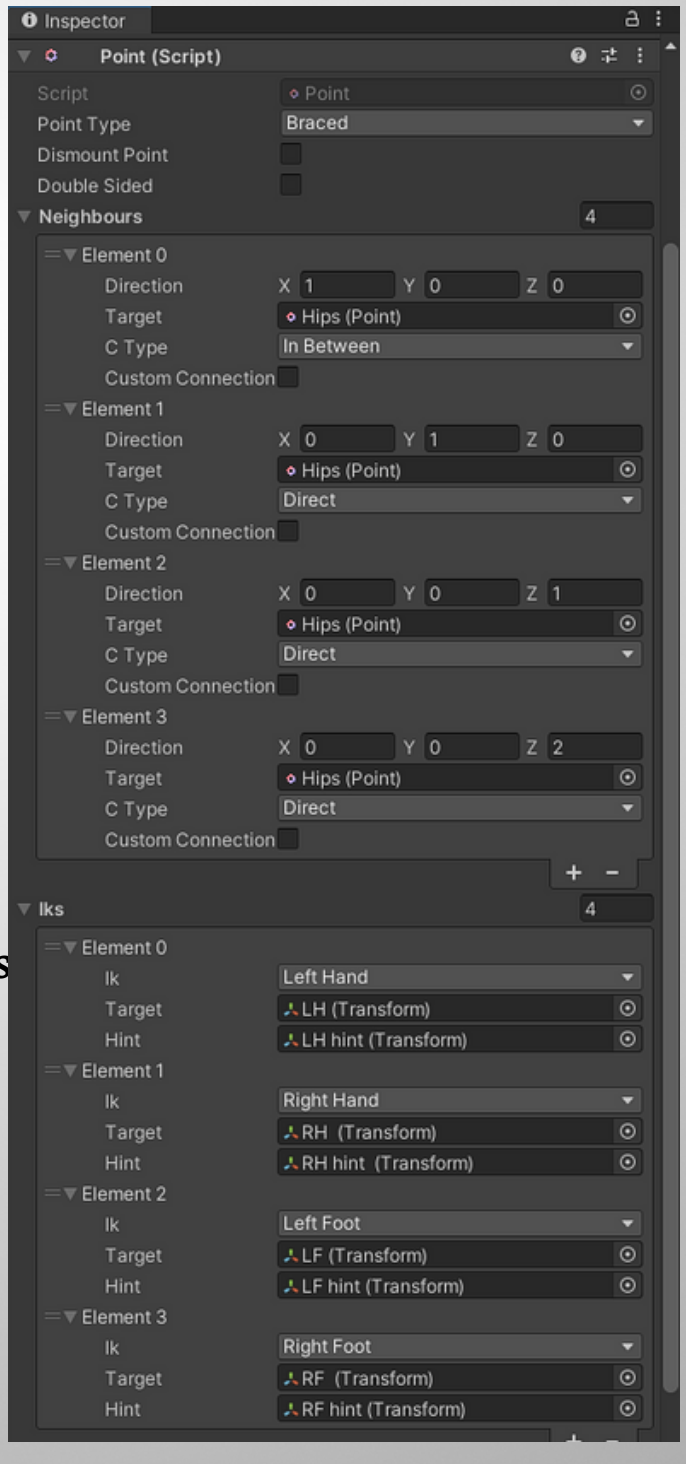
Point Type - A Point is either Hanging(Free Climb) or Braced
Dismount Point - This is whether a point is dismountable and is set automatic
Double-Sided - This only applies to hanging points which you can turn around

Neighbours

Each element has:
Direction - This is a direction Vector 3 which tells the climb behaviour script which direction to move to reach a point
Target - The desired Point
CType- Is the connection type between two points whether direct or linear
Custom Connection - Custom Connects

IKs

Each element has:
IK - Which body part is the Ik Controlling
Target- What is the target that this body part is following or leading towards.
Hint - Is there an elbow or a knee that allows for a more natural readjusting of the body parts such as bending the elbows or knees.



Handle Connections

The Handle Connections script automatically connects all the points and sets all the neighbours and directions for your use.

Variables-

Min Distance - The maximum distance between one point and another that is allowed when automatically making the connections and neighbours between two points.

Direct Threshold - The number at which a point stops being a linear or normal point and becomes direct of jumping point. This means if the distance between two points is 0.5 then the character will move normally in a linear type movement. Although if the distance between the two points goes above 1 then the type of movement between the two points will then become jumping and not be linear.

Update Connections - Takes all the points in the scene and makes automatic connections between each one and sets their corresponding neighbours for each direction possible.

Reset Connections - Resets all the current connections on all points so that you can update the connections again.

Note:
Reset the connections before you start to update the connections, and modify the variables.