CUSTOMIZABLE PIXEL CHARACTER

Thank you for purchasing this asset pack. For any question, please email to *support@cainos.net*

QUICK TUTORIAL

Drag and drop one of the character presets in **[Cainos\Customizable Pixel Character\Prefab\Character Preset]** into the scene and there you go. Character customizable is done in the **[Customization]** foldout group in **[Pixel Character]** script.

NOTES

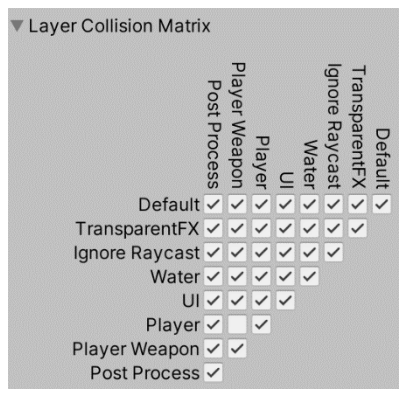
Layer Collision Settings

To avoid the character colliding with its own weapon. Put the character and weapon into different layer and in the project's Physics 2D settings, make sure these two layers do not collide.

You need to setup this by hand as project settings will not be imported with the asset pack

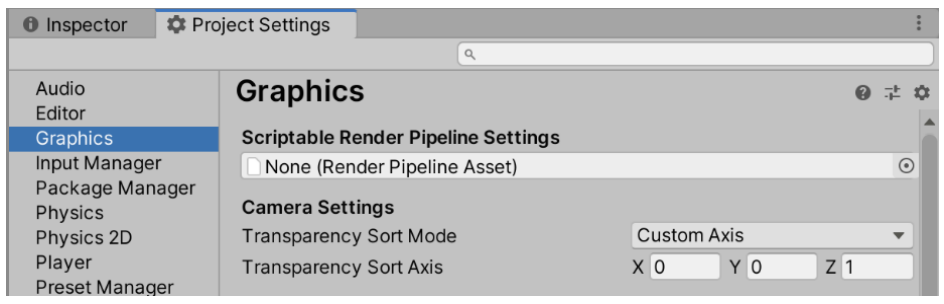
In this case we put characters into **[Player]** layer and weapons into **[Player Weapon]** layer. Of course, this may vary based on your need.

All the character presets are variation prefabs of **[Cainos\Customizable Pixel Character\Prefab\PF Pixel Character]**. So, for overall changes you only need to modify this one.



Custom Sort Axis

It is recommended that you set **[Transparency Sort Mode]** to **[Custom Axis]** and **[Transparency Sort Axis]** to **[0,0,1]** at the **[Graphics]** settings.



Sorting Order Glitch When Using Multiple Character

You may see some glitch when two characters are too close. It happens when the two characters take up the same z space. Give them different z position value will solve the problem.

You can set the character's z scale to a smaller value like 0.1 (but avoid setting it to 0), so that it takes up less z space.

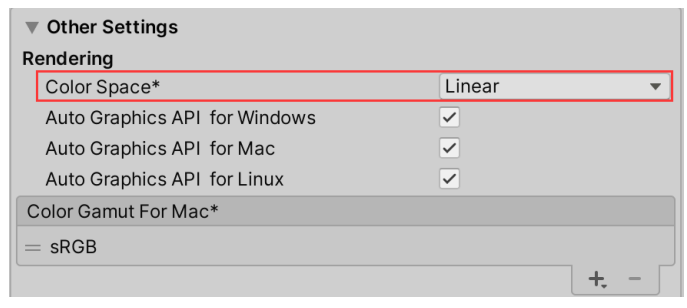
Skin Weights

For best animation quality, in **[Project Settings/Quality]**, **[Skin Weights]** should be set to at least **[2 Bones]**.

Color Space

It is recommended to use **[Linear]** color space with this asset.

You can find it in **[Project Settings/Player/Other Settings]**.



CHARACTER SCRIPTS

Overview

There are 3 scripts attached to the character object.

Pixel Character	Storing object reference inside the character object and character customization.
Pixel Character Controller	Controlling movement of the character.
Pixel Character Input Mouse and Keyboard	Reading mouse and keyboard input and feed into the [Pixel Character Controller] script

There are also 2 scripts attached to the [Animator] object inside character
Generally, you don't need to pay attention to these two scripts.

Animation Event Receiver	For receiving animation events.
Root Motion Receiver	For receiving root motion from the animation.

Pixel Character

For storing object reference inside the character object and character customization.

Reference Foldout

Contains reference to objects inside the character object.

Customization Foldout

Parameters here is mainly for tweaking the character appearance.
The customization is mainly done by changing the materials here.
Can be changed both in editor and runtime.

Blink Interval	The interval range for the character to play an eye blink animation.
Hat Material	Material of the character's hat
Hair Material	Material of the character's hair
Eye Material	Material of the character's eye
Eye Base Material	Material of the character's eye shape. You only need to change this when changing the character's gender.
Facewear Material	Material of the character's facewear.
Cloth Material	Material of the character's cloth.
Pants Material	Material of the character's pants.
Socks Material	Material of the character's socks.
Shoes Material	Material of the character's shoes. This will be displayed behind the character's pants.
Back Material	Material of the object the character carries on the back.
Body Material	Material of the character's body. You need to change this when changing the character's gender. Also, special character presets would use their specific body material, like Vampire or Zombie.
Clip Hair	Hide part of the character's hair. When wearing curtain hats, you need to enable this.
Hide Hair	Completely hide the character's hair.
Shoes in Front	Whether to display shoes in front of the character's pants.

Runtime Foldout

Parameters here should only be changed at runtime.

Facing	The character's facing. 1: Facing right -1: Facing left
Is Dead	Whether the character is dead. Turning this on will only make the character appear dead on the graphic side. For making the character truly dead, turn on the [Is Dead] on the [Pixel Character Controller] script.
Expression	The character's expression.
Injured Front	Play [Injured Front] animation.
Injured Back	Play [Injured Back] animation.

Detach Weapon	Detach weapon to a separated object from the character. Returns the detached weapon game object.
Clear Weapon	Clear out everything in the weapon slot.

Pixel Character Controller

Script for controlling the character's movement.
It will modify some of the **[Facing]** parameters in the **[Pixel Character]** script and parameters in the **Animator** to control animation.

Movement Foldout

Parameters for the character's movement.

Ground Check Layer Mask	Objects under this layer mask will be consider ground the character can stand on.
Walks Speed Max	Max walking speed.
Walks Acc	Walking Acceleration.
Run Speed Max	Max running speed
Run Acc	Running Acceleration
Crouch Speed Max	Max move speed while crouching.
Crouch Acc	Crouching acceleration.
Crawl Speed Max	Max move speed while crawling.
Crawl Acc	Crawling acceleration.
Air Speed Max	Max move speed while in air
Air Acc	Air acceleration
Ground Drag	Braking acceleration (from movement to still) while on ground
Air Drag	Braking acceleration (from movement to still) while in air
Jump Enabled	Whether the character can jump.
Jump Speed	Speed applied to the character when jump.
Jump Cooldown	Time needed to be able to jump again after landing
Jump Tolerance	When character is off ground and inside this time, the character is still able to jump.
Jump Gravity Multiplier	Gravity multiplier when character is jumping. Should be within [0.0,1.0], set it to lower value so that the longer you press the jump button, the higher the character can jump.
Fall Gravity Multiplier	Gravity multiplier when character is falling. Should be equal or greater than 1.0
Dash Enabled	Whether the character can dash.
Dash Speed Start	Speed when dash starts.
Dash Speed Max	Max move speed while dashing.
Dash Acc	Dash acceleration.
Dash Time	Time the dash state will last.
Dash Cooldown	Time it takes for the character to be able to dash again after exiting dash state.
Dodge Enabled	Whether the character can dodge.
Dodge Speed Mul	Dodge speed multiplier
Dodge Cooldown	Time it takes for the character to be able to dodge again after exiting dodge state.

Ladder Climb Enabled	Whether the character can climb ladder.
Ladder Climb Speed	Ladder climbing speed.
Ladder Climb Speed Fast	Ladder climbing speed when the run key (by default is SHIFT) is pressed.
Ledge Climb Enabled	Whether the character can climb ledge.

Attack Foldout

Parameters for the character's attack action.

Attack Action	Attack action to perform whether primary attack key (by default is LEFT MOUSE BUTTON) is pressed.
Attack Action Melee	Attack action to perform whether melee attack key (by default is V) is pressed, or when the character is in a state when primary attack action is not available, like when crawling or climbing ladder.
Throw Force	When using **[Throw]** attack action, the character will throw out the weapon it is holding. This defines the force applied to the thrown weapon.
Throw Angular Speed	The angular speed applied to the thrown weapon.
Arrow Speed	The speed applied the arrow when using **[Archery]** attack action.
Arrow Prefab	The prefab of the arrow to be instantiate when using **[Archery]** attack action

Event Foldout

This foldout contains some Unity events that you can use to trigger your own function.
A typical use case is to play sound effects when curtain event happens.

On Footstep	When the character's foot touches the ground in animation like walk or run.
On Jump	When the character jump away from ground.
On Land	When the character land on the ground from air.
On Dodge Start	When dodge action starts.
On Dodge End	When dodge action ends.
On Attack Start	When attack action starts.
On Attack Hit	When attack action is supposed to hit the target. Damage to the target can be applied at this event.
On Attack End	When attack action ends
On Bow Pull	When using [Archery] attack action, the character start pulling the bow string.
On Bow Shoot	When using [Archery] attack action, the character release the bow string and the arrow is shot.
On Throw	When using [Throw] attack action, the character throws out the holding weapon.

Input Foldout

Parameters for receiving inputs from input scripts.
If you are going to use your own input scripts, you need to modify these parameters in your scripts.
Only available at runtime.

Input Move	movement input, x for horizontal, y for vertical, x and y should be in [-1.0, 1.0]
Input Run	Run input, when enabled the character will run instead of walk.
Input Dash	Dash input, when triggered the character will enter dash state.
Input Dodge	Dodge input, when triggered the character will enter dodge state.
Input Crouch	Crouch input, when enabled the character will be crouching.
Input Crawl	Crawl input, when enabled the character will be crawling. Will override crouch input.
Input jump	Jump input, when triggered the character will start to jump. When continuously enabled the character will jump higher.

Input Attack	Primary attack input, will trigger the attack action when available.
Input Melee	Melee attack input, will trigger the melee attack action.
Input Look	Look input, when enabled the character will look at the position defined by **Input Target**
Input Target	This is a world space Vector2 position that defines the target the character should look at or point at when performing curtain attack action.

Debug Foldout

Parameters for switching debug gizmos display.
You don't need to be bothered with this.

Runtime Foldout

Parameters here should only be changed at runtime.
Most parameters here is read only and just for exposing states of the controller.

Is Dead	Is the character dead? if yes, plays dead animation and disable control.

Pixel Character Input Mouse and Keyboard

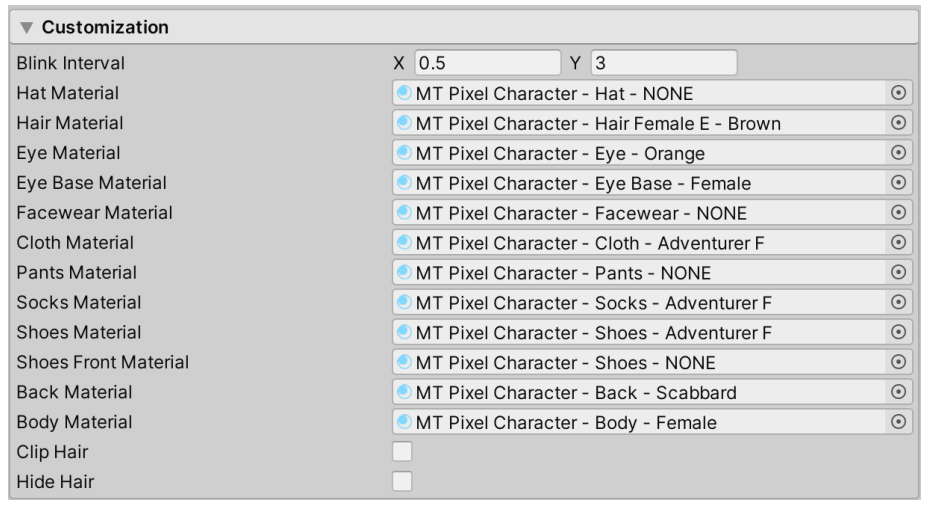
This script will read mouse and keyboard input and modify parameters in the **[Input]** foldout of the **[Pixel Character Controller]** script.
If you are going to use your own input script or let an AI to control the character, just remove this script.
It is also a good example for showing how to feed input to the **[Pixel Character Controller]** script.

CUSTOMIZATION

In this part we will go through the ways to customize different parts of the character.

Most of the customization is done by changing material in [**Customization]** foldout of the **[Pixel Character]** script.

You can check the character presets to see how their customizations are set up.



Gender

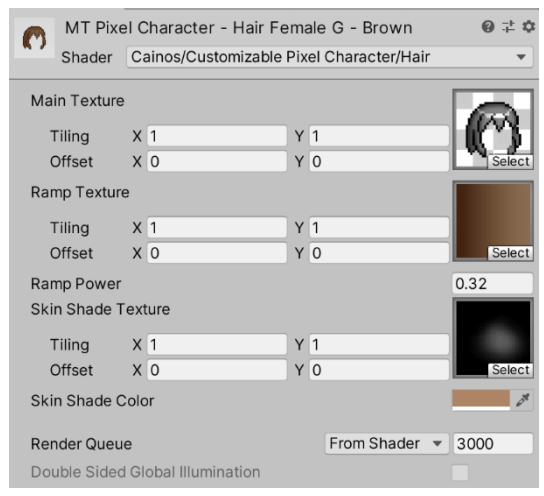
By changing the **[Body Material]** and **[Eye Base Material]** to corresponding material
Also, pick a hair material to match the gender.

Skin Tone

By changing the **[Skin Tint]** parameter in the **[Body Material]** of the character.

Hairstyle & Hair Color

By changing the **[Hair Material]**.
Notes that not every hairstyle & hair color combination has a material created in advance, but you can easily create your own.
Duplicate a hair material and change the “Main Texture” for hairstyle, “Ramp Texture” for hair color.



Hat

By changing the **[Hat Material]**.
When using curtain hat, you also need to enable the **[Clip Hair]** or **[Hide Hair]** to hide part of the hair outside the hat.

Facewear

By changing the **[Facewear Material]**.

Cloth

By changing the **[Cloth Material]**.

Pants

By changing the **[Pants Material]**.

Socks

By changing the **[Socks Material]**.

Shoes

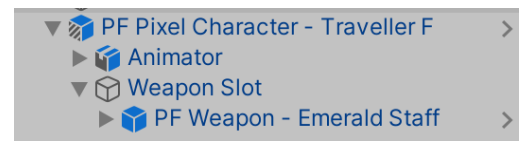
By changing the **[Shoes Material]** or **[Shoes Front Material]**.
Shoes in **[Shoes Material]** will be displayed behind the pants while shoes in **[Shoes Front Material]** will be displayed in front of the pants.

Back

By changing the **[Back Material]**.

Weapon

By dragging one of the weapon prefabs in [**Cainos\Customizable Pixel Character\Prefab\Weapon]** into the **[Weapon Slot]**.



INTERACTABLE OBJECTS

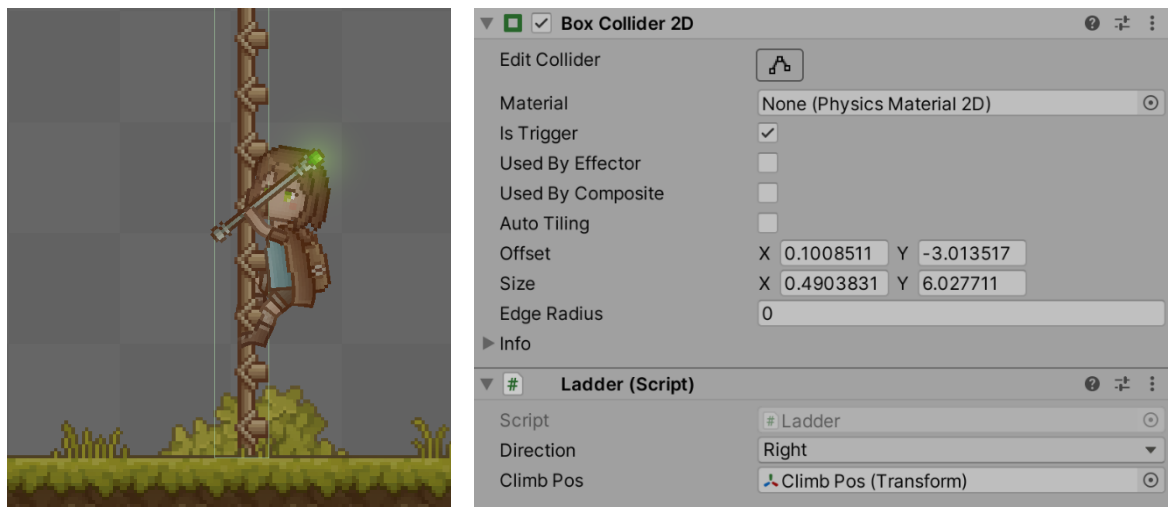
Ladder

A ladder the character can climb onto.

It need a **[Box Collider]** with **[Is Trigger]** enabled which defines the area of the ladder.

And the **[Ladder]** script with the **[Direction]** set to the ladder's direction (**Left** or **Right**).

The **[Climb Pos]** is a transform inside the ladder object which defines the character's position when climbing the ladder.



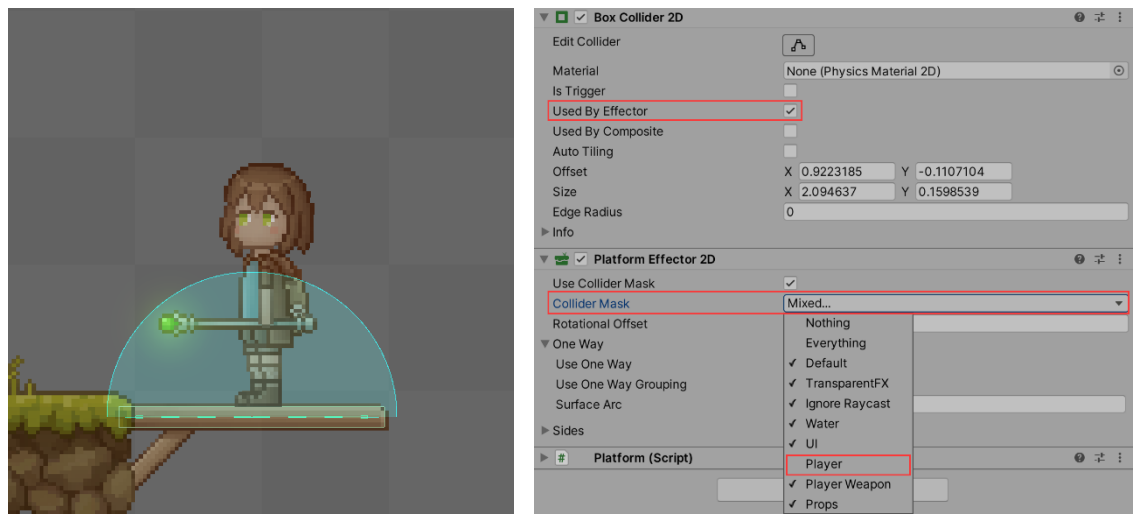
Platform

One-way platform that the character can jump onto and get down from it.

When standing on a platform and keep pressing down key, the character can directly get down from the platform.

It needs a collider with **[Used By Effector]** enabled, a **[Platform Effector 2D]** with **[Collider Mask]** set to exclude the player layer.

And a **[Platform]** script.

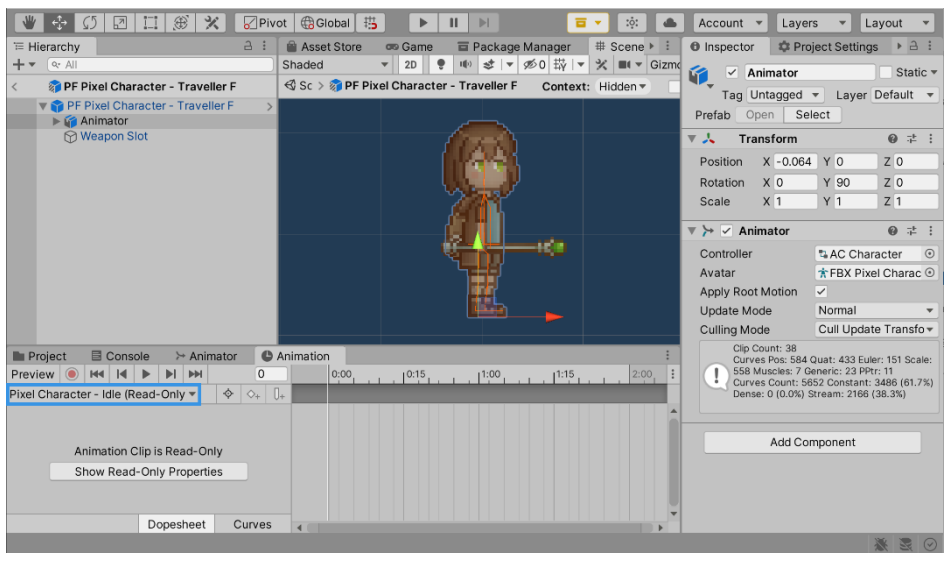


MAKING YOUR OWN ANIMATIONS

Below is a short tutorial for making new animation for characters inside Unity.

Basic

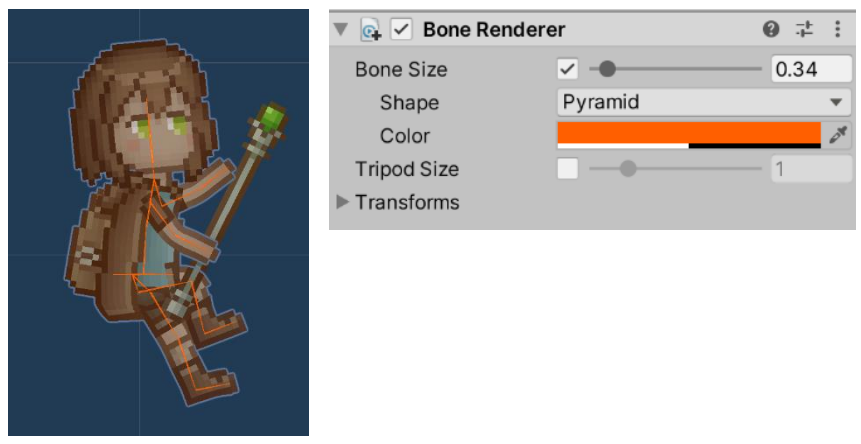
Select any one of the character prefabs and go into Prefab Editing mode.
Select **[Animator]** in the **[Hierarchy]**.
Create a new clip in the Animation window.



Then you can start keying your animation.
You should only key on objects under the **[Animator]** object.
All the character's bones are in the **[Rig]** hierarchy.

Bone Display

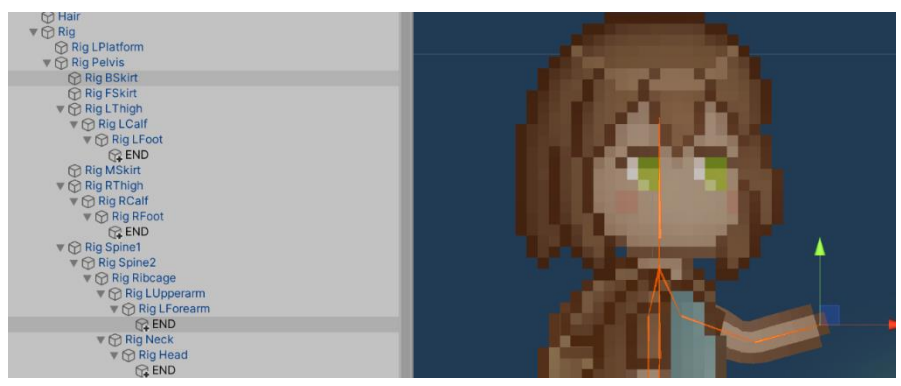
You can use the [**Bone Renderer]** script of Unity's **[Animation Rigging]** package (available in **Package Manager**) for displaying and easy selecting bones in the scene.



Add a **[Bone Renderer]** script to the character and click on the lock button on the top-right corner of the Inspector window.
Right click on **[Rig]** object inside the character hierarchy and select **[Select Children]**.
Drag all the objects selected to the **[Transforms]** list of the **[Bone Renderer]** script.

Some of the Rig objects are just helper objects (for example [**Rig]**, **[Rig LPlatform]**, **[Rig RPlatform]**). You can delete them from the "Transforms" list to stop them from being displayed.

As the **[Bone Renderer]** cannot properly display end bone's length correctly. You can manually and a new object to the end bone and place it at the end position, then drag it to the **[Transforms]** list of the **[Bone Renderer]** to force the end bone to be displayed with a proper length.

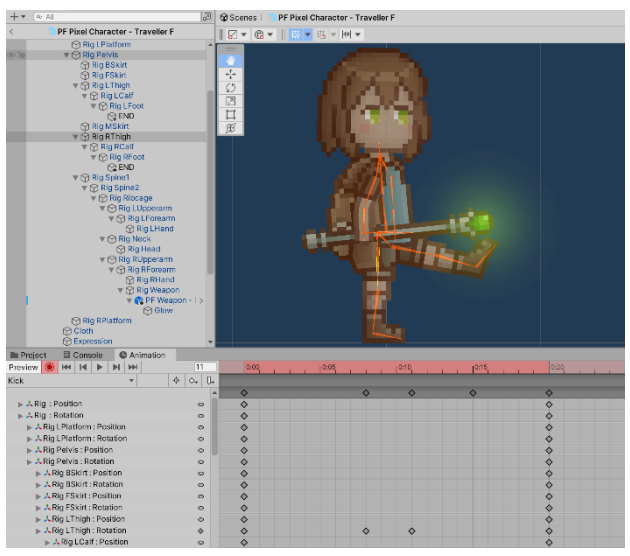


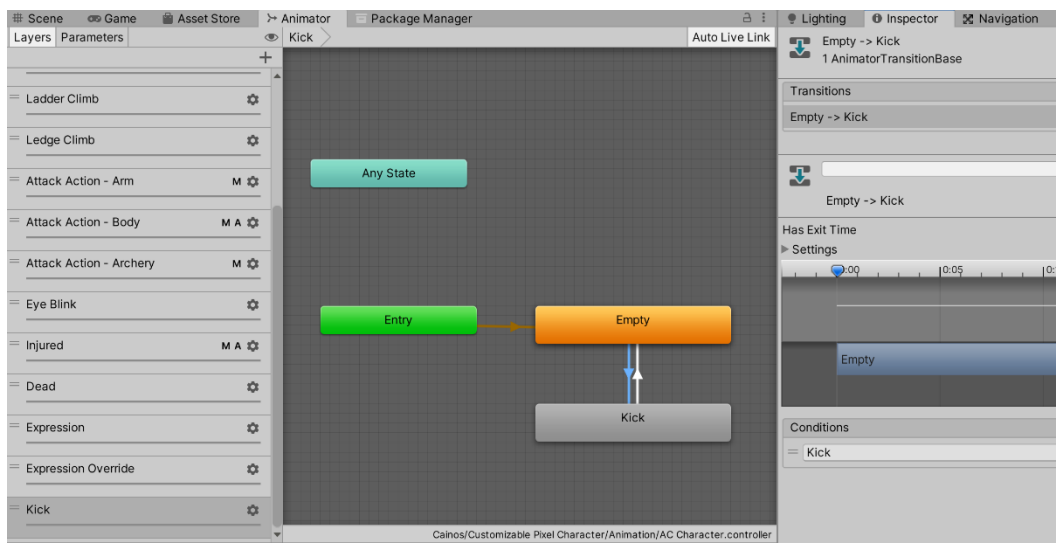
Animator Setup

When your animation is done, you need to put the clip into the character's Animator Controller.

In this case we made a kick animation, so we add a new layer in the Animator called **[Kick]**, a new trigger parameter call **[Kick]** and setup the animation state as the picture below.

We use an empty state (with no motion clip) as the entry state. When **[Kick]** is triggered, it transits to the **[Kick]** state with the animation clip we just made. When the **[Kick]** animation finish playing, it goes back to the **[Empty]** state.



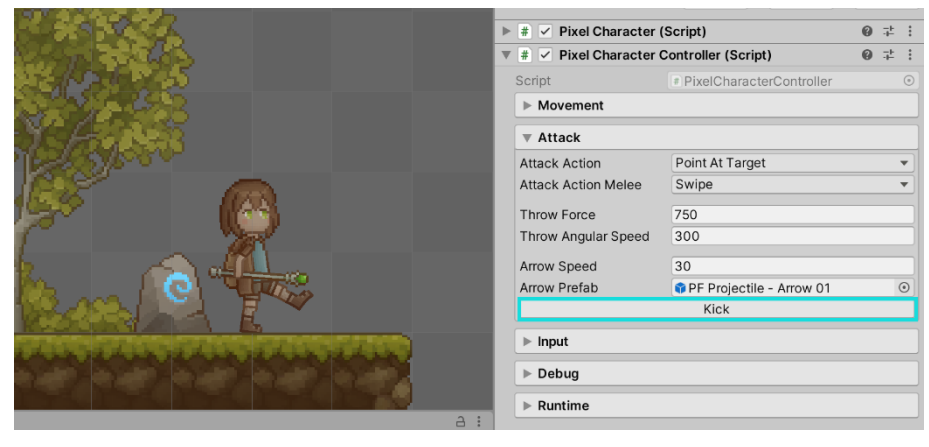


Scripting

Then we need to trigger the **[Kick]** with our code. You can put the code anywhere you like. In this case, we put it in the **[Pixel Character Controller]** script.

Then, we are able to trigger the kick animation by the Kick function or the Kick button in the Inspector.

```
[FoldoutGroup("Attack"), Button("Kick")]
0 references
public void Kick()
{
    animator.SetTrigger("Kick");
}
```



Create a Specific Prefab for Making Animation

The weapon object place inside **[Weapon Slot]** is syncing its position with **[Rig Weapon]** bone through script. So, in the editor, the weapon object will not move with the **[Rig Weapon]** bone. This brings inconvenience when making animation.

To solve this, simply drag the weapon object into the **[Rig Weapon]**.

As mentioned above, there are many modifications we may need to do to the character prefab to prepare it for making animation.

So, my suggestion is that you copy the character prefab you want to make animation for and make your modifications to the clone instead of directly to the prefab you are going to use in game.

Animations created in any one of the character prefabs are also compatible in other characters as they share the same rig structure.

LIGHTING SUPPOSRT

By default, the character shaders behave like an unlit sprite shader, and can be used in any render pipeline.
But if you need lighting on the character, you need to install additional shaders to make them support lighting.
Currently lighting on character only supports Universal Render Pipeline.

Universal Render Pipeline 2D Lighting

Import files from **[Shader Patch - URP 2D Lit]**. It will replace current character shaders with URP 2D Lighting supported version.

Make sure your 2D lighting is properly set up so the character can be displayed correctly.

Universal Render Pipeline 3D Lighting

Import files from **[Shader Patch - URP Lit]**. It will replace current character shaders with URP 3D Lit supported version.