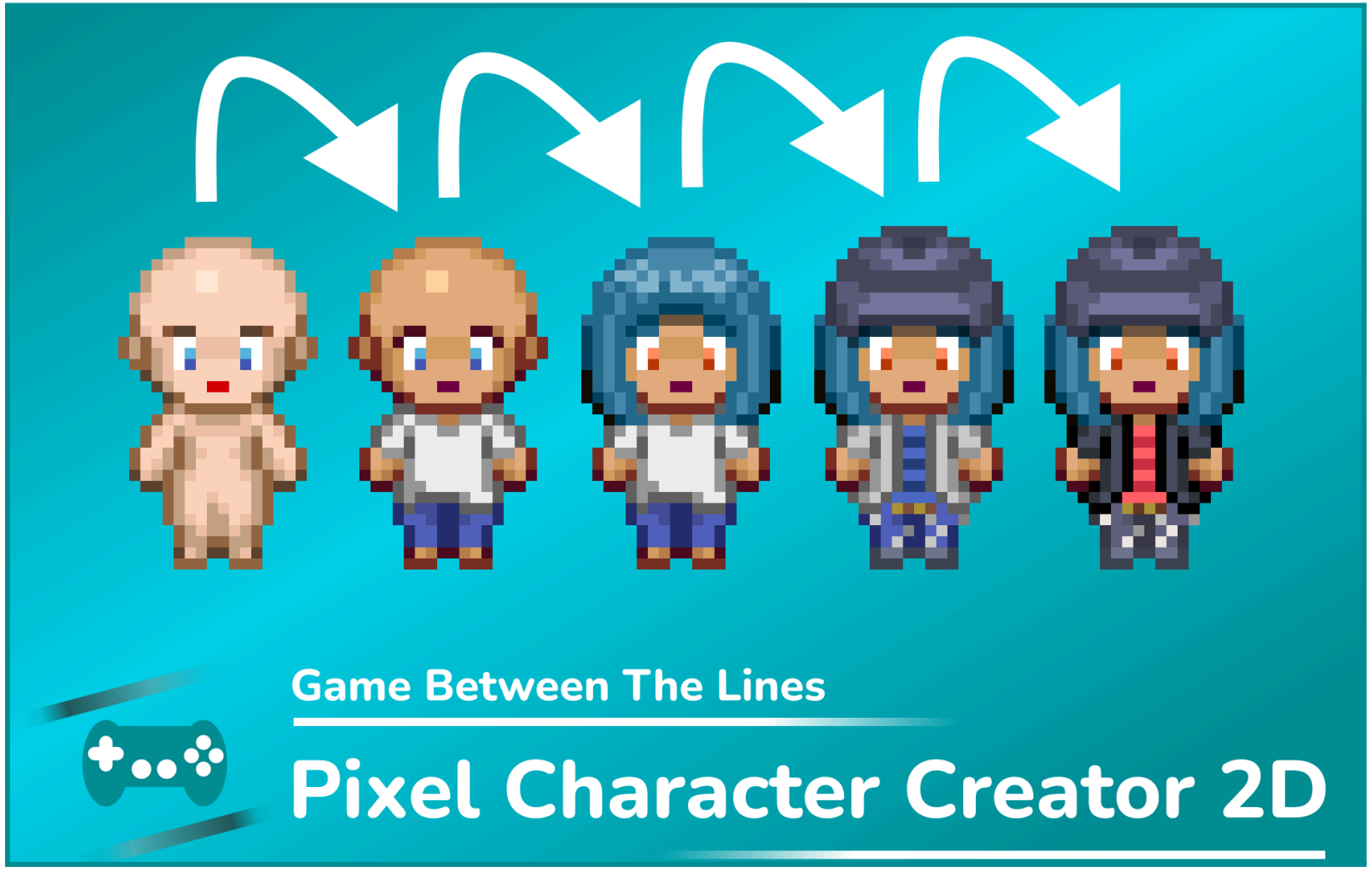
# Pixel Character Creator 2D For Unity Documentation

**Version 1.0.1**



# Table of contents

# 1. Quick Setup & Introduction

These are the necessary steps to get the asset up and running as quickly as possible. In case you wish to view the online documentation for this chapter, you can do so here. If you've become familiar with how this asset works and want to learn how to set up this asset entirely from scratch, I recommend you read Chapter 7.

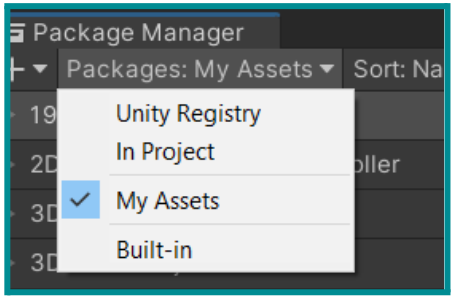
## 1.1 Importing the package

For demonstration purposes and to avoid confusion, I recommend creating a new project and importing the asset there so you can learn how the tool works without breaking something in your game.

**STEP 1.**
Open your (newly created) Unity project.

**STEP 2. (Unity Package Manager)**
Open the Package Manager. At the top of the window, click "Packages:", then select "My Assets".

**STEP 2. (Manually)**
Navigate to the directory where you stored your download.




**STEP 3. (Unity Package Manager)**
Look for the "Pixel Character Creator 2D" asset in this list, then click import.

**STEP 3. (Manually)**
Drag the .unitypackage file from your directory into the project tab. Alternatively, you can right-click in the project tab, select Import Package > Custom Package…, and search for the .unitypackage file.

**STEP 4.**
Decide what you want to import. I recommend importing everything if you created a new Unity project so you can learn the tool with the help of demo files. If you're importing this asset into your own game, you may want to exclude the demo files.

# 1.2 Texture Importers in the Preset Manager

The Preset Manager is a great tool to automatically set import settings for certain file types. For this asset I used it specifically with Texture Importers, These Texture Importers ensure that all the .png files we create for the character have the correct import settings upon creation. Follow the steps below to set the manager up properly. Failing to set up the manager correctly will result in errors upon exporting your character.

I won't go into detail about how the Preset Manager works, but I do recommend you read the official Unity Documentation about this topic if you are interested.

## Setup method 1: Selecting a pre-made .preset-file

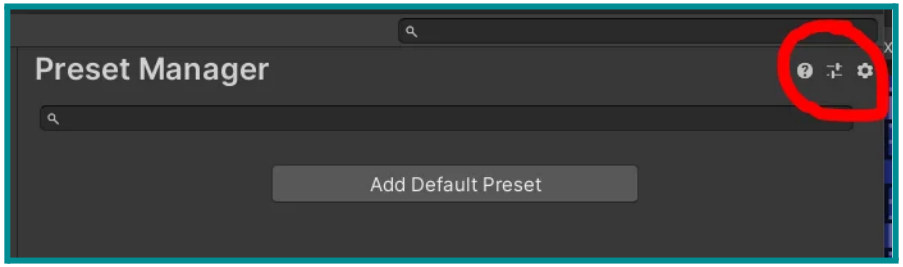
Setup method 1 is the fastest way to set up the Preset Manager. Using this method may be beneficial if you are not using the Preset Manager for any other purpose. However, if you are using the Preset Manager for other purposes, you may want to go for the 2nd setup method to avoid accidentally resetting your own settings.

**STEP 1.**

In the top bar: go to File > Project Settings > Preset Manager.

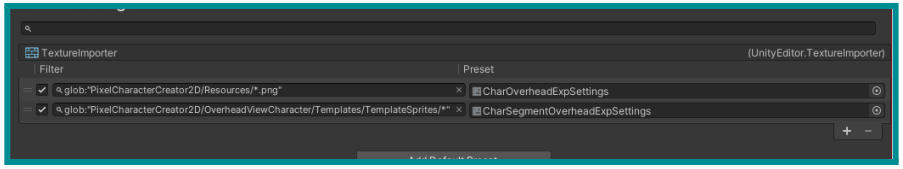
**STEP 2.**

In the Preset Manager window, click the "Select Preset" button in the top-right corner. This is what the button looks like:  .




**STEP 3.**

With the "Select Preset" pop-up window opened, select the already existing PresetManager file. Your Preset Manager Window should now match the example below. We're now done with the Preset Manager.



# Setup method 2: Manually adding Texture Importer .preset-files

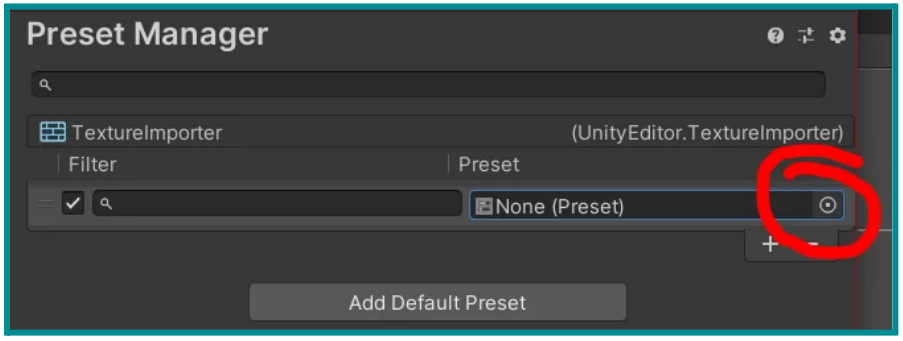
This setup method is safer if you wish to keep any presets you already placed in this manager. Setting up the Preset Manager via this method will likely cost a few minutes.

**STEP 1.**

In the top bar: go to Edit > Project Settings > Preset Manager.

**STEP 2.**

In the Preset Manager window, click “Add Default Preset”. Look for “Texture Importer” in the search bar and select it. Your Preset Manager should now look like the screenshot below.




**STEP 3.**

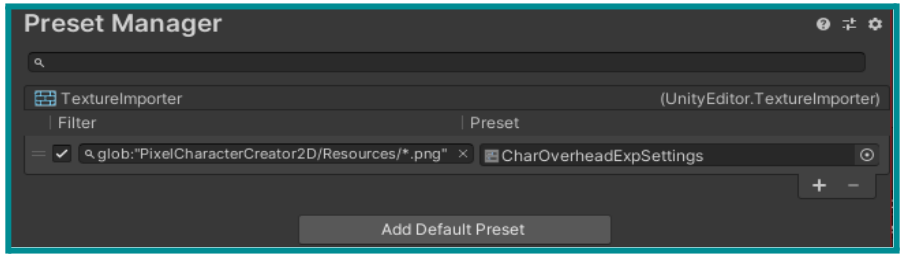
Click on the white dot I marked with the red circle in the screenshot. Then select “CharOverheadExpSettings”.

**STEP 4.**

Paste the following path into the Filter field next to the Preset field:

`glob:"PixelCharacterCreator2D/Resources/*.png"`

Your Preset Manager Window should now look like this:




**STEP 5.**

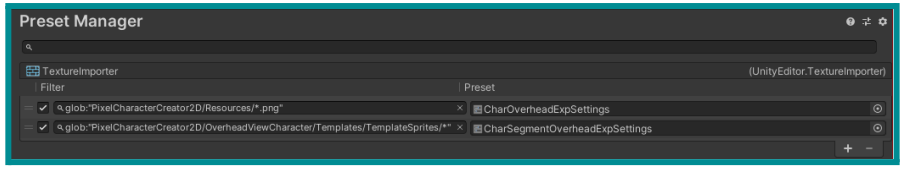
Click the + icon to create another row. Then click on the white dot again. This time, select “CharSegmentOverheadExpSettings”.

**STEP 6.**

Paste the following path into the Filter field next to the Preset field:

`glob:"PixelCharacterCreator2D/OverheadViewCharacter/Templates/TemplateSprites/*"`

Your Preset Manager Window should match the example below. We’re now done with the Preset Manager.



# 1.3 Opening the Character Creator

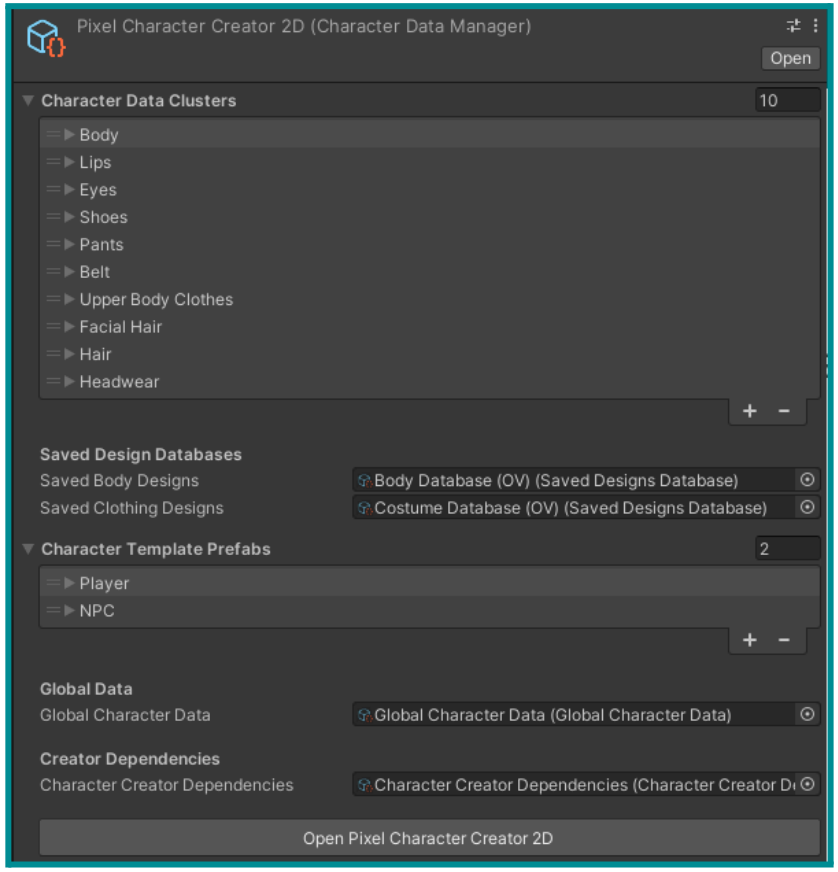
That does it for the quick setup. Now, I recommend we take a first look at the Character Creator itself. I want you to learn where to find the Character Creator, how to open it, and what the individual character editing windows are. Later in the documentation, we'll go more in-depth.

**STEP 1.**

In the Project tab: navigate to *Assets / PixelCharacterCreator2D / OverheadCharacter*. In this folder, you will find 3 additional folders, some of which I will explain later. For now only the scriptable object "PixelCharacterCreator2D" matters. That is a Scriptable Object that functions as a Character Data Manager. Think of this Data Manager as the main place where all data regarding the characters is gathered. Click on the Scriptable Object once to view it in the inspector.

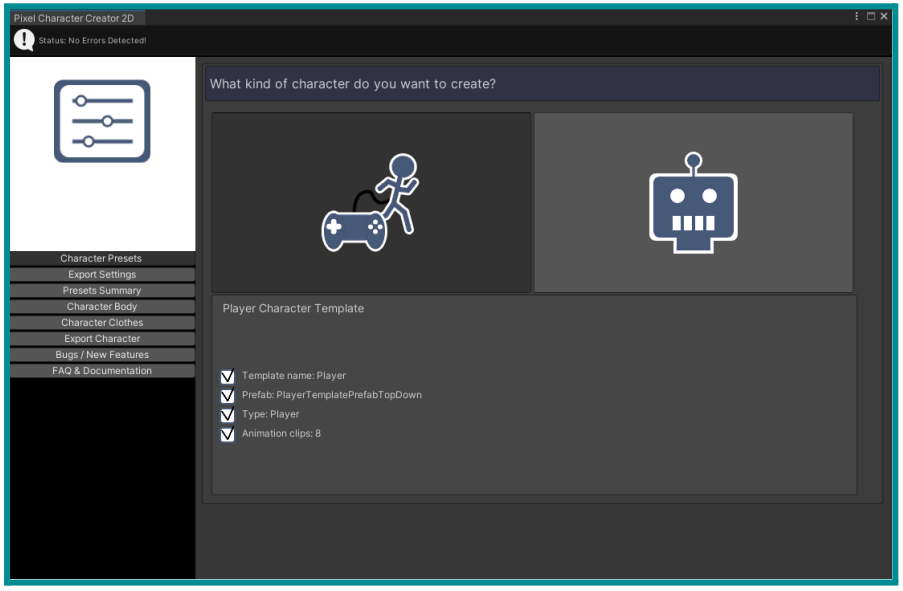
**STEP 2.**

If you imported the demo files, your inspector window will look similar to the one in the screenshot on the right. Everything in this Character Data Manager will be explained later, but feel free to look around in the inspector. For now, we need to press the button at the bottom of the inspector called "Open Pixel Character Creator 2D". Double-clicking the Scriptable Object itself will also open up the Creator Tool.



**Info**

Upon opening the Creator, you will see a menu with buttons on the left, an error message bar at the top, and a bigger section with more information on the right. I will quickly mention the different creator windows and what their purposes are.




## Character Presets

Here you can select if you want to create a Player character or an NPC. You can also preview some information about the Player or NPC which is drawn from a Character Template. More on that in a later chapter.

## Export Settings

There are three main exportable parts; the sprite sheet, the prefab, and the animations. Here you can customize the character export process for all three parts. This is where we'll spend most of our setup time. If you've imported the demo, this window will be preset with different values and variables, most of which will be covered in Chapter 1.4.

## Presets Summary

As the name implies, this summarizes your chosen settings. This is also where you get to name the character and initiate the creation process.

## Character Body

Upon pressing the 'Create' button, this window becomes available. Here you get to edit as little or as many body segments and colors as you want. More on this in Chapter 3.

## Character Clothes

Upon pressing the 'Create' button, this window becomes available. Here you can edit as little or as many clothing designs and colors as you want. More on this in Chapter 3 as well.

## Export Character

If you're satisfied with your design choices, you can export your character here. This window will also remind you where your designs will be exported to.

## Bugs / New Features

In case you find a bug or opportunity for a new feature, you can use this window to send in your feature request or bug report.

## FAQ & Documentation

This window will direct you to the FAQ and documentation on this website.

# 1.4 The Export Settings explained

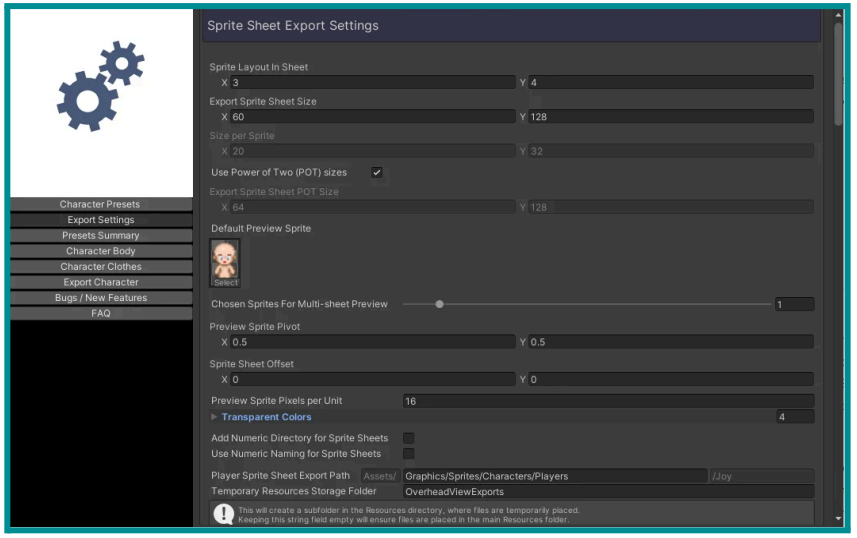
The Export Settings window displays all settings regarding the output of the Character Creator. Understanding the significance of these settings will help you reach the full potential of this asset.
This section will be divided into three parts:

- Sprite Sheet Export Settings
- Prefab Export Settings
- Animation Export Settings

## Sprite Sheet Export Settings

The Sprite Sheet Export Settings section contains variables that control the way sprite sheets are handled. Here you can find variables to alter character preview sprites as well as the actual sprite sheet export at the end of the creation process.

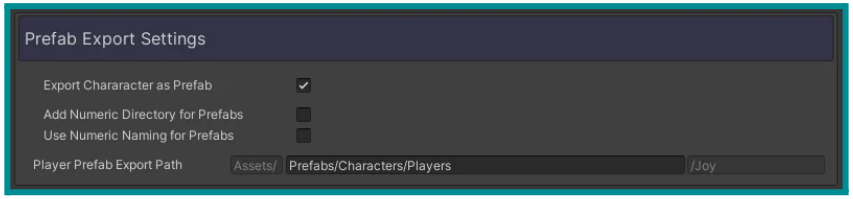
The table below showcases all the variables that have to do with the character sprite preview and the export sprite sheets. These variables can be adjusted to tailor the sprite output to your liking. Review the table below if you find yourself wondering what the variables do.



| VARIABLE NAME | TYPE | VARIABLE USAGE |
|---|---|---|
| Sprite Layout In Sheet | Vector2 | The X-value represents the amount of sprites in a horizontal row. In the demo, the sprite sheets have 3 sprites in a horizontal line. The Y-value represents the amount of sprites in a vertical column. In the demo, the sprite sheets have 4 sprites in a vertical line. |
| Export Sprite Sheet Size | Vector2 | The X-value represents the width of the sprite sheet in pixels. In the demo, the width of the sprite sheets is 60 pixels. The Y-value represents the height of the sprite sheet in pixels. In the demo, the height of the sprite sheets is 128 pixels. An error appears if the size doesn't match the sprite sheet size in the database. |
| Size Per Sprite | Vector2 | The size per sprite is automatically calculated based on the Sprite Layout In Sheet and the Export Sprite Sheet Size. This is calculated as follows:<br>sizePerSprite.x = exportSpriteSheetSize.x / spriteLayoutInSheet.x<br>sizePerSprite.y = exportSpriteSheetSize.y / spriteLayoutInSheet.y |
| Use Power of Two (POT) Sizes | Bool | If this boolean is enabled, the Export Sprite Sheet Size is converted to the nearest POT (Power Of Two)-size. This is useful for creating sprite sheets that are optimized for compression. |
| Export Sprite Sheet POT Size | Vector2 | This is the calculated nearest POT-size of the Export Sprite Sheet Size. If the bool in the row above is enabled, this sheet size is used instead of the Export Sprite Sheet Size. If the size in this field doesn't match the actual sprite sheet size in the database, an error message will pop up. |
| Default Preview Sprite | Sprite | A sprite that shows up as a default character before any adjustments are made to the body or clothes. The sprite is replaced if the character in the Character Creator Window is updated. |
| Chosen Sprite For Multi-sheet Preview | Int | This number determines which sprite (pose) in the sprite sheet is used to preview changes you make to the character. In the demo, the default value here is 1, which shows the idle front sprite of the character. Should you, for example, change this number to a 10, the sprite (pose) changes to the idle back sprite of the character. With this, you can view the character from different poses during the design process. |

| | | |
|---|---|---|
| Preview Sprite Pivot | Vector2 | If your character sprite has a custom pivot point, this is where you can ensure that the preview sprite has the same pivot point. This is mostly meant to help avoid the sprite's position suddenly changing during the creation process. |
| Sprite Sheet Offset | Vector2 | If the 1st sprite in your sprite sheet has an offset and doesn't immediately appear in the top-left corner of your sprite sheet, use this. The X-value represents the amount of horizontal pixels that need to be bridged before the first sprite in the sheet appears. The Y-value represents the amount of vertical pixels that need to be bridged before the first sprite in the sheet appears. |
| Preview Sprite Pixels per Unit | Int | The Pixels per Unit amount to set the newly created sprite preview at. Ensure this matches the Pixels per Unit of your character segment sprites in the database to avoid unwanted behavior. |
| Transparent Colors | List | This List consists of custom serialized class instances. In code, it looks like this: List<TransparantColor>. The TransparentColor class consists of a string, Color, and two float fields. In this List, you can include default colors with an opacity below alpha 1. This is particularly useful for transparent shadows and see-through clothing pieces such as glasses. |
| Add Numeric Directory for Sprite Sheets | Bool | If this boolean is enabled, an additional numeric directory will be added at the end of the export path. Consider using this option if you prefer to give each character their own directory. |
| Add Numeric Naming for Sprite Sheets | Bool | If this boolean is enabled, a number will be added at the start of the character's name. Consider using this option if you prefer to have multiple characters in one directory. |
| Player Sprite Sheet Export Path | String | This export path determines where the exported character sheet will be stored after the creation of the character. If this string field is left empty, the character sprite sheet will be placed in the root Assets folder. |
| Temporary Resources Storage Folder | String | If this string isn't empty, A temporary subfolder will be created in the Resources folder to store the player sprite sheet. That new subfolder will use this string as its name. This is useful for keeping the permanent files you have stored in the Resources folder separate from files created by this asset. |

# Prefab Export Settings

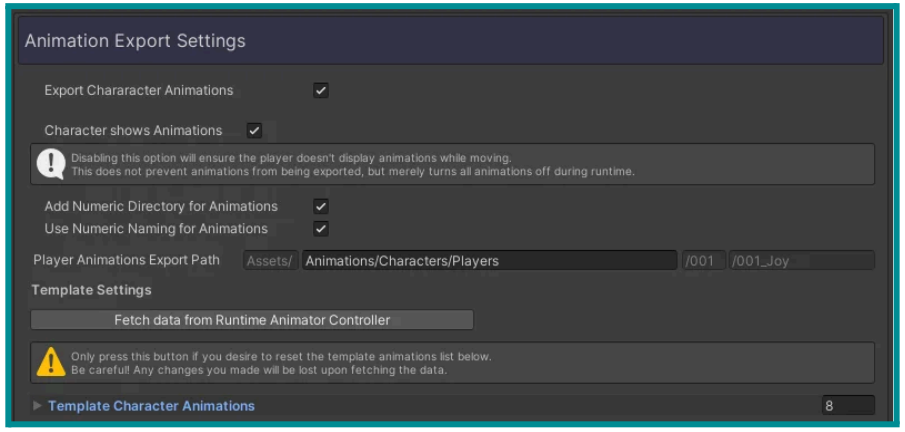



This is the shortest section of the Export Settings window. The Prefab Export Settings section contains variables that control the way Prefabs are handled. These variables are mostly meant to alter the export path.

| VARIABLE NAME | TYPE | VARIABLE USAGE |
|---|---|---|
| Export Character as Prefab | Bool | If this boolean is enabled, a Prefab will be exported at the end of the creation process. Otherwise, this step will be skipped. |
| Add Numeric Directory for Prefabs | Bool | If this boolean is enabled, an additional numeric directory will be added at the end of the export path. Consider using this option if you prefer to give each character their own directory. |
| Add Numeric Naming for Prefabs | Bool | If this boolean is enabled, a number will be added at the start of the character's name. Consider using this option if you prefer to have multiple characters in one directory. |
| Prefab Export Path | String | This export path determines where the exported Prefab will be stored after the creation of the character. If this string field is left empty, the Prefab will be placed in the root Assets folder. |

# Animation Export Settings

The Animation Export Settings section contains variables that control the way Animations are handled. In addition to the variables that are mostly meant to alter the export path, there are also Template Character Animation variables that can be adjusted.




| VARIABLE NAME | TYPE | VARIABLE USAGE |
|---|---|---|
| Export Character Animations | Bool | If this boolean is enabled, an Animator Controller and Animations will be exported at the end of the creation process. Otherwise, this step will be skipped. |
| Character Shows Animations | Bool | If this boolean is disabled, the character won't be animated while performing actions such as walking. This does not prevent the Animations and Animator Controller from being exported but merely turns all animations off during runtime. (This can be changed retroactively if a prefab was also exported.) |
| Add Numeric Directory for Animations | Bool | If this boolean is enabled, an additional numeric directory will be added at the end of the export path. Consider using this option if you prefer to give each character their own directory. |
| Add Numeric Naming for Animations | Bool | If this boolean is enabled, a number will be added at the start of the character's name. Consider using this option if you prefer to have multiple characters in one directory. |

| | | |
|---|---|---|
| Player Animations Export Path | String | This path determines where the exported character animations are stored after creating the character. If the string field is kept empty, the animations and animator controller are placed in the root Assets folder. |
| Template Character Animations | List | This List consists of custom serialized class instances. In code, it looks like this: List<TemplateCharacterAnimation>. The TemplateCharacterAnimation class consists of many fields that manipulate how animations function. The List is automatically filled with Template Animation Data from the Template Character's Runtime Animator Controller. In this List, you can change variables such as the keyframe amount, loop time, and frames per second. |

# 2. Demo

In this chapter, we explore the demo scene and all the demo files included in this asset. This way, you'll get to see an example of what can be achieved with this character creator.

## 2.1 Dependencies for the demo

### Unity's New Input System

Unity's New Input System is used to handle the player's inputs in the demo scene. If you've never worked with the New Input System before, don't worry. For this demo, you won't have to configure anything. All you have to do is download and import the New Input System from the package manager. Everything should work immediately afterward.

### Free assets made by Game Between The Lines

A handful of assets from the Game Between The Lines library are used in this demo to give you a good impression of what the Pixel Character Creator 2D is capable of:

- Basic Dialogue System (included with the demo)
- Basic Top-Down NPC System (included with the demo)
- Basic Top-Down Player System (included with the demo)
- Top-Down Pixel Art Casino Tileset (included with the demo)

Important note: the previously mentioned dependencies are only needed for the demo and aren't required when using the Pixel Character Creator 2D tool itself.

### Layers and Tags

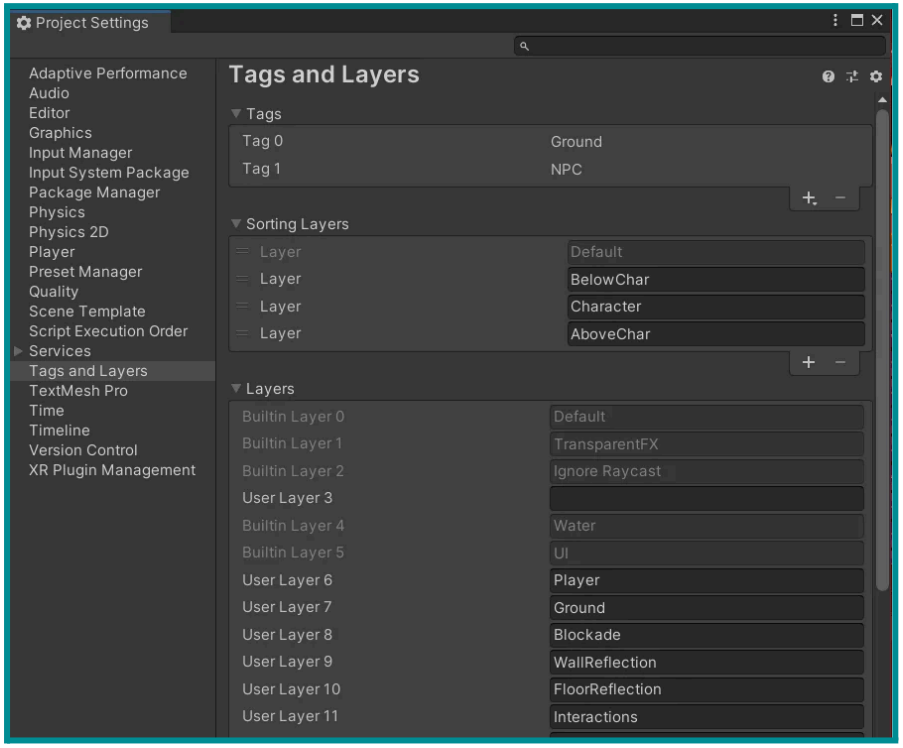
**Automatic setup (recommended)**

You can quickly set up the required layers, sorting layers, and tags by following the instructions below.
WARNING, this method will replace the current layers and tags you already set up. Only use this method if you're planning to use this demo in a clean Unity Project:

1. Go to *Edit > Project Settings > Tags and Layers*.
2. With the Tags and Layers window opened, click on the "Select Preset" icon in the top right corner (between the question mark and the cogwheel). The button looks like this: 
3. Select "TagAndLayerManager". You should now have all the required layers and tags.

**Manual setup (may require fixing scene objects and prefabs)**

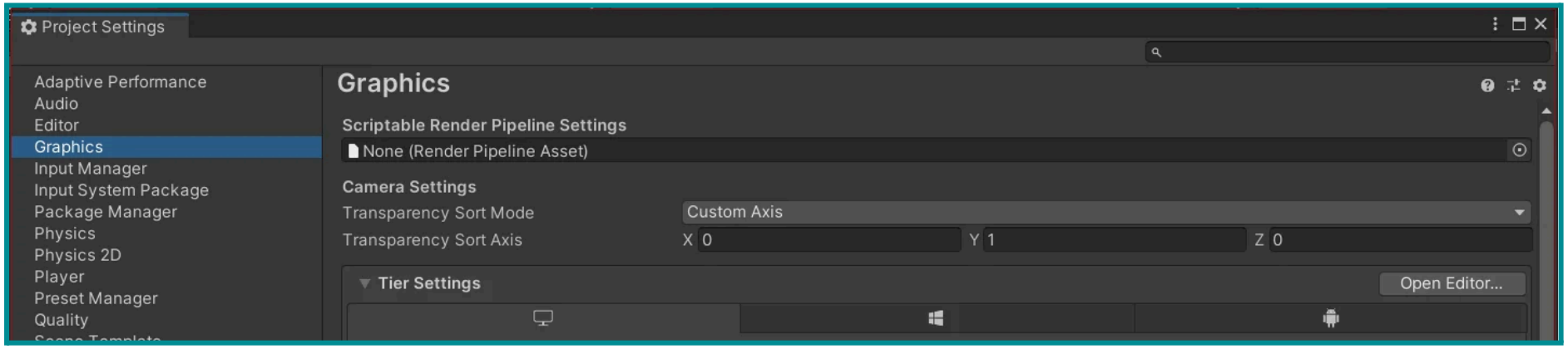
Alternatively, you can view the image on the next page and manually add the tags and layers. Please keep in mind that if you decide to add them manually, Sorting Layers for prefabs and scene objects may not be recognized instantly. This will require manual fixing, so it's best to avoid using this method if possible.




## Transparency Sort Mode

In order to render the 2D Tilemap Layers properly a quick adjustment needs to be made to the sort mode:

1. Go to *Edit > Project Settings*. Then select the Graphics tab. Under the Camera Settings, there are two variables we are going to adjust.
2. Change the Transparency Sort Mode to Custom Axis.
3. Then, set the Transparency Sort Axis to match the one in the screenshot at the bottom of the page.



## 2.2 Playing the demo scene

The demo scene can be found by navigating to *PixelCharacterCreator2D / Demo / Demo_Scenes / TopDownDemoScene.unity*.
With all the previously mentioned dependencies taken care of, this scene is now ready for playtesting.

### Demo Scene Controls

Please view all the controls below so you can playtest the demo scene as fluently as possible.

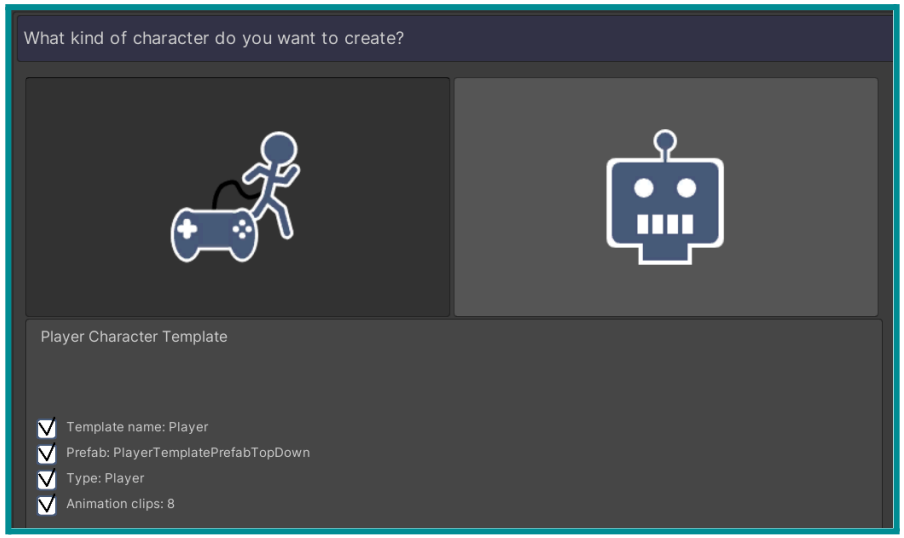
| DEFAULT KEY BINDING | USAGE | ADDITIONAL INFORMATION |
|---|---|---|
| WASD | Walk / Run | The player can walk and run in 4 directions. |
| E | Interact | The player can interact with NPCs and objects like trash cans, slot machines, and closed doors. Interaction input will only be handled if the player is standing next to an 'interactable entity'. |
| R | Toggle Run | The player can use a toggle to walk or run. This choice was made to prevent having to hold a key every time you want to quickly test something. Switch to walk or run by just pressing R once. |

# 2.3 Practicing with the Pixel Character Creator 2D

Now that we've tested the demo scene and seen the characters in action, let's create a new character.

**STEP 1.**

Open the *PixelCharacterCreator2D.asset*. In the Character Presets tab, select either the Player Character Template or the NPC Character Template. This will determine whether we're creating a player or NPC. You'll learn how to set this up yourself in Chapter 7.

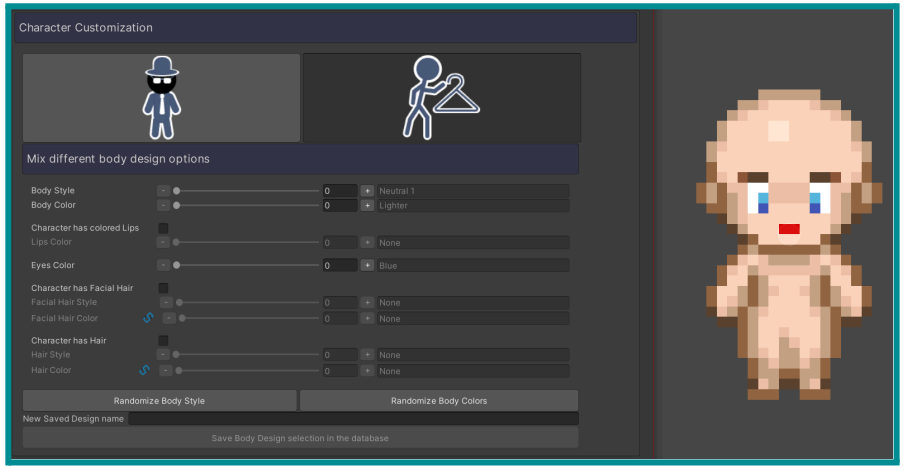



**STEP 2.**

I recommend keeping the Export Settings as they are unless you know what you're doing. You'll learn more about setting up all of these settings yourself in Chapter 7. If you want, you could toggle certain settings such as "Export Character as Prefab" and "Export Character Animations" on or off in case you, for instance, only want to export Sprite Sheets.

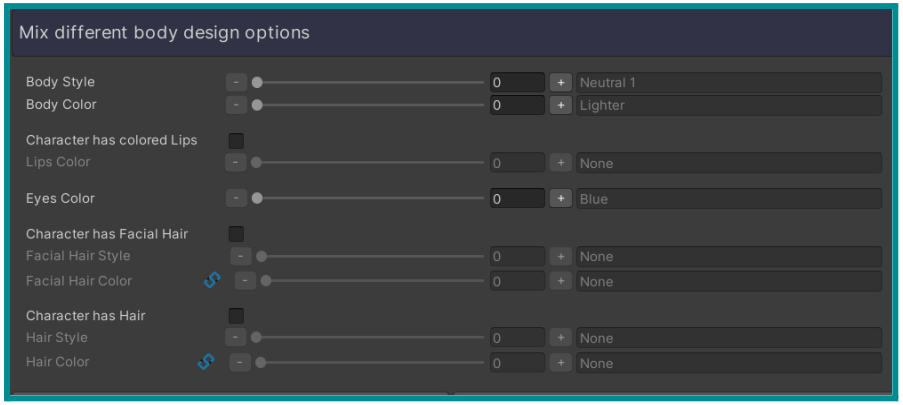
**STEP 3.**

Navigate to the Presets Summary to review your choices. If you're satisfied, name your character and click "Create new Character". A character with your chosen name has now been created in the scene. Zoom in on the default character sprite so you can easily view the changes you'll make.



**STEP 4.**

Use the sliders in the Character Body tab to change the different body segments and their colors. In Chapter 7 you'll learn how to add a custom body segment and colors to this tab.

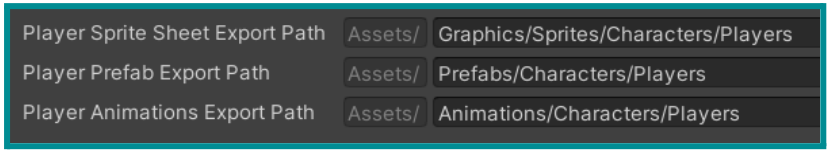



**STEP 5.**

Now do the same in the Character Clothes tab to change the clothing of your new character. Feel free to also tinker with the Randomize buttons near the bottom of the window.

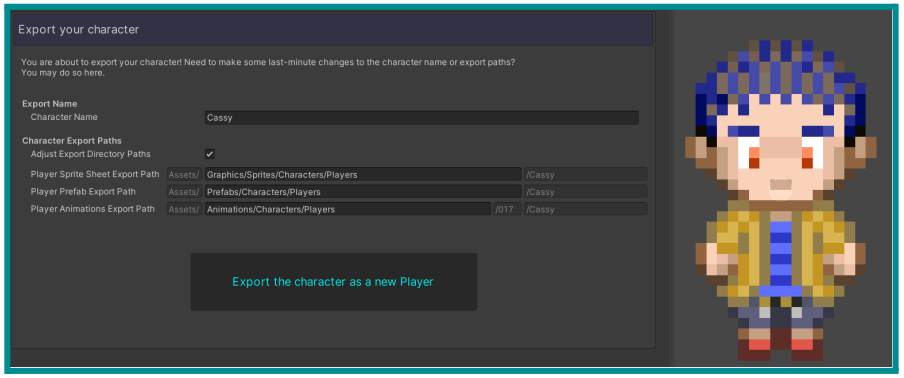
**STEP 6.**

Once you're satisfied, navigate to the Export Character tab. If you find yourself wanting to change the name, you can do so here. You can also change the export path for the sprites, prefab, and animations if needed.




**STEP 7.**

Click "Export the character as a new Player". You've created a character!




**STEP 8.**

Test your character in the demo scene and tinker with their settings in the Inspector Window.

## 2.4 Cross-asset features in the demo

The demo scene requires some of the free-to-use assets such as the Basic Dialogue System, Basic NPC system, and the Basic Player system. The compatibility between the Character Creator and those other assets provides you with a near-seamless demo. None of the other assets are required for using the Character Creator 2D itself, but since all of the assets mentioned in section 2.1 are free, I do recommend using them for testing purposes in this demo.


### Refer to the body and clothes in dialogue

The Basic Dialogue System can be used together with the Basic NPC System and the Basic Player System to retrieve the character design data of either character, provided by the Character Creator 2D. This is how it works:

If a character has the Character Data script attached to it, the Dialogue System will be able to detect it. Every time the design of the character is updated with the Character Creator 2D, all design information for that character is stored in this script, including all chosen segments and colors. A reference to a stored character segment or color can be made by placing the variable name in the Dialogue Initiator in a specific format. View these examples below:

- *"Whoa, your hair is totally awesome! I believe that's a {p_hairstyle}, isn't it?" (p_hairstyle refers to the hairstyle of the player)*
- *"How'd you dye your hair {p_haircolor} like that?" (p_haircolor refers to the hair color of the player)*
- *"What do you think of my {npc_facialhaircolor} beard?" (npc_facialhaircolor refers to the facial hair color of the NPC the player is talking to)*
- *"I always request a {npc_facialhairstyle} style when I'm at my local barber." (npc_facialhairstyle refers to the facial hairstyle of the NPC the player is talking to)*


Here, you can view all default Dialogue Reference Tags that are included in this demo. These can be used to refer to the design of the player character or the NPC whom you initiated the conversation with. The references are recognized by the initial letters “p” (short for player) or “npc”. Do keep in mind that the tags below are case-sensitive. Open up the foldable sections to view all Dialogue Reference Tags.

| PLAYER REFERENCE TAGS | REFERS TO: | EXAMPLE IN TEXT |
|---|---|---|
| {Player} | Player name | “Oh hi, {player}. Long time no see!” |
| {p_bodystyle} | Player body style | “Does your body feel {p_bodystyle}?” |
| {p_bodycolor} | Player body color | “Are you doin’ okay there buddy? You look a bit {p_bodycolor}…” |
| {p_lipscolor} | Player lips color | “Is that a new color of lip gloss? Tell me how I can get {p_lipscolor} lips as well!” |
| {p_eyescolor} | Player eyes color | “Is {p_eyescolor} your natural eye color or are you wearing contacts?” |
| {p_shoesstyle} | Player shoes style | “Nice shoes! What are those? {p_shoesstyle}?” |
| {p_shoescolor} | Player shoes color | “Not sure I like that {p_shoescolor} color, but I do like the shoes’ design.” |

| | | |
|---|---|---|
| {p_pantsstyle} | Player pants style | “Why are you wearing {p_pantsstyle} to the party? Is there a dress code?” |
| {p_pantscolor} | Player pants color | “The witness stated they saw a figure with {p_pantscolor} jeans. It just so happens you are wearing that color.” |
| {p_beltstyle} | Player belt style | “Oh hey, did you buy that {p_beltstyle} at the clothing store?” |
| {p_beltcolor} | Player belt color | “Told ya! A {p_beltcolor} belt works fine with the rest of your outfit!” |
| {p_upperclothingstyle} | Player upper clothing style | “A {p_upperclothingstyle}, eh? Looks like someone is going somewhere special tonight.” |
| {p_upperclothingcolor} | Player upper clothing color | “{p_upperclothingcolor} just so happens to be my favorite color, which means you’re in luck.” |
| {p_facialhairstyle} | Player facial hair style | “I’m not sure whether I like you better with or without that {p_facialhairstyle}…” |
| {p_facialhaircolor} | Player facial hair color | “That {p_facialhaircolor} facial hair of yours does make you look a bit older.” |
| {p_hairstyle} | Player hair style | “Oh my, your hair is lovely! What’s that style called? {p_hairstyle}, am I right?” |
| {p_haircolor} | Player hair color | “That’s a nice hair color! {p_haircolor}, suits you a lot.” |
| {p_headwearstyle} | Player headwear style | “That’s a cool {p_headwearstyle}. where can I buy one?!” |
| {p_headwearcolor} | Player headwear color | “That {p_headwearcolor} color complements the rest of your outfit very well!” |

| NPC REFERENCE TAGS | REFERS TO: | EXAMPLE IN TEXT |
|---|---|---|
| {npc} | NPC name | “My name is {npc}. Nice to meet you!” |
| {npc_bodystyle} | NPC body style | “My body feels {npc_bodystyle}…” |
| {npc_bodycolor} | NPC body color | “I was hoping to get a tan today, but now I’m all {npc_bodycolor} instead…” |
| {npc_lipscolor} | NPC lips color | “Read my {npc_lipscolor} lips … I won’t do it!” |
| {npc_eyescolor} | NPC eyes color | “Do you think the {npc_eyescolor} contacts I’m wearing are… too much? ” |
| {npc_shoesstyle} | NPC shoes style | “We require you to wear special shoes, like the {npc_shoesstyle} I’m wearing.” |
| {npc_shoescolor} | NPC shoes color | “Now look what you’ve done! My beautiful {npc_shoescolor} shoes… RUINED!” |
| {npc_pantsstyle} | NPC pants style | “I used to hate wearing {npc_pantsstyle}, but I’ve grown accustomed to them.” |
| {npc_pantscolor} | NPC pants color | “But I am wearing {npc_pantscolor} jeans as well. Maybe the witness was talking about me…” |
| {npc_beltstyle} | NPC belt style | “I am wearing a {npc_beltstyle} to avoid losing my pants during my daily stroll.” |
| {npc_beltcolor} | NPC belt color | “When it comes to belts, I usually only wear {npc_beltcolor} ones. Don’t ask me why!” |

| | | |
|---|---|---|
| {npc_upperclothingstyle} | NPC upper clothing style | “Do you think this {npc_upperclothingstyle} makes me look fat?” |
| {npc_upperclothingcolor} | NPC upper clothing color | “I always wear the same {npc_upperclothingcolor} clothes. I can’t help myself. It’s just too comfy…” |
| {npc_facialhairstyle} | NPC facial hair style | “I just can’t seem to grow a proper {npc_facialhairstyle}… Thanks, genes…” |
| {npc_facialhaircolor} | NPC facial hair color | “Yarr landlubber, you may call me Captain {npc_facialhaircolor} beard! You’d better compliment me, lest ye be prepared to walk the plank.” |
| {npc_hairstyle} | NPC hair style | “Do you like my hair like this? It’s called a {npc_hairstyle}!” |
| {npc_haircolor} | NPC hair color | “I like to dye my hair {npc_haircolor} every once in a while.” |
| {npc_headwearstyle} | NPC headwear style | “How do you like my {npc_headwearstyle}? I just bought it the other day.” |
| {npc_headwearcolor} | NPC headwear color | “What do you think of this hat? Is it too {npc_headwearcolor}? Be honest!” |

## Check a character for a specific design feature

The Character Data script exposes a lot of information in the inspector. All of the data in this inspector can be accessed by other scripts. This makes it easy to check if a character meets certain criteria. A simple example of this could be that you’re only allowed to enter a lab room with clothing that’s suited for your safety, such as Safety Goggles and a Lab Coat. You could make an NPC check if the player is wearing these items and allow passage if this variable returns true.

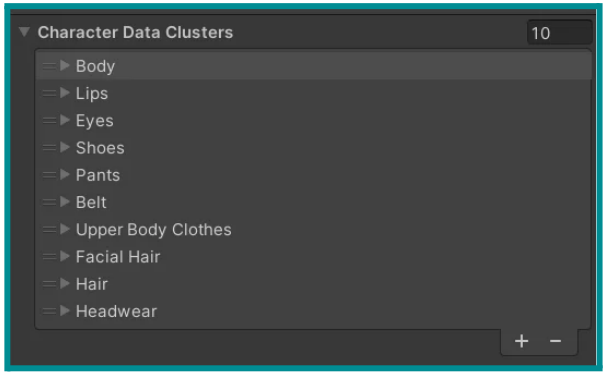
# 3. Data Manager Part 1: Clusters & Databases

In this chapter, you'll learn everything you need to know about Character Data Clusters and Databases. In addition to explaining how a Character Data Cluster works, I'm going to give you instructions on how to create your own data clusters and databases.

## 3.1 What is a Character Data Cluster?

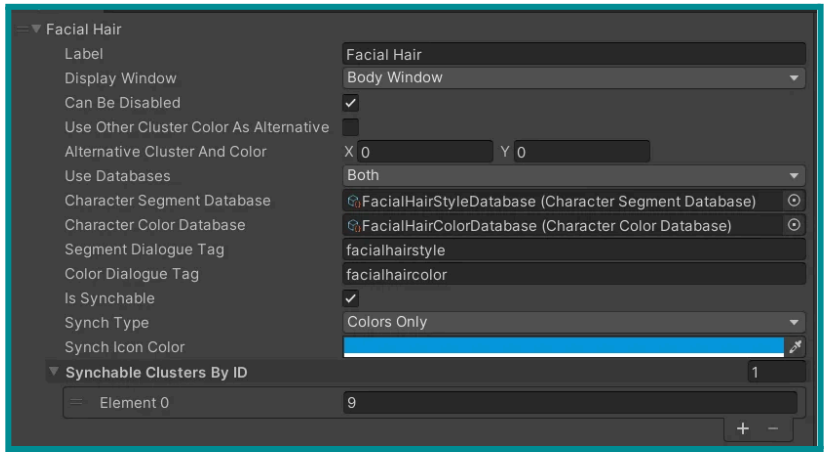
### The purpose of Data Clusters

Data Clusters can be found within a Character Data Manager. They contain information about a specific character part. For instance, if you want to customize the hair of a character, creating a Data Cluster for the character's hair allows you to do so. A Data Cluster should be created for each part of the character that you want to be able to change. This way you can make a very simple character composed of a few Data Clusters (I.E. for a body and face) or very detailed characters with many different parts that can be changed independently from each other. Data Clusters give you more control at the cost of a bit more setup time. I recommend thinking the designs of your characters through a bit. How many parts of the character do you want to be able to change independently from another? If you want to change the 'hair', 'eyes', 'mouth', and 'skin' then the amount of Data Clusters will most likely be four.




### The Variables in a Data Cluster

On the next page, you can view the different variables that allow you to define certain aspects of the Character Data Cluster. This Data Cluster is then read by the Pixel Character Creator 2D and taken into account during the creation process of the character.



| VARIABLE NAME | TYPE | VARIABLE USAGE |
|---|---|---|
| Label | String | The name of the Character Data Cluster. This name will be shown as the label inside of the Character Creator 2D editor. |
| Display Window | Enum | This enum decides whether this Data Cluster will be shown in the ‘body’ section of the Character Creator or the ‘clothing’ section. |
| ID | Int | The ID of this Character Data Cluster. This ID is used for recognition purposes by other clusters or other scripts. Typically, you’d want to keep the ID variable the same as the Character Data Clusters index number. |
| Can Be Disabled | Bool | If this boolean is enabled, a tick box will appear next to the label of this cluster, inside the Character Creator. This way you can make certain character parts, such as hair, entirely optional. |
| Replace Color From Other Cluster | Bool | This allows you to look inside another cluster for a specific color to replace with the chosen color from this cluster. You’ll typically only use this option if you have a Data Cluster that’s only in charge of replacing a specific color, without adding a new sprite. In the demo, this option is used to replace the color of the character’s lips with a different color, without changing the sprite of the character’s face. |
| Replaceable Cluster And Color | Vector2 | This variable only works if the boolean “Replace Color From Other Cluster” is set to true. The X-variable represents the cluster you want to replace a specific color from. The Y-variable represents the specific color inside of the cluster you want to replace. In the demo, this value is set to Vector2(0,3). This means that the 3rd color (#EBC1A8) inside of cluster 0 (Body) will be replaced by the chosen lip color. |

| | | |
|---|---|---|
| Use Databases | Enum | This enum decides whether this Data Cluster will make use of Character Segments (sprites), Character Color Groups, or both. This can be handy if you want to create a cluster that's only meant to add a sprite or color instead of both. |
| Character Segment Database | Scriptable Object | The Segment Database for this specific Data Cluster should be placed here. Segment Databases contain data for a specific sprite sheet that belongs to this cluster. If this cluster is supposed to be meant for the character's hair, the Segment Database should contain all the different hairstyles that the character can wear. |
| Character Color Database | Scriptable Object | The Color Database for this specific Data Cluster should be placed here. Color Databases contain data for the colors that belong to this cluster. If this cluster is supposed to be meant for the character's hair, the Color Database should contain all the different hair color options. |
| Segment Dialogue Tag | String | This tag will allow for the segment in this Data Cluster to be recognizable in dialogue. This only works when combined with the Basic Dialogue System. In this field, you should add the tag with which you want to refer to this cluster's segment (sprite). For the hairstyle of the character, this could be hairstyle. |
| Color Dialogue Tag | String | This tag will allow for the colors in this Data Cluster to be recognizable in dialogue. This only works in conjunction with the Basic Dialogue System. In this field, you should add the tag with which you want to refer to this cluster's colors. For the hairstyle of the character, this could be haircolor. |
| Is Syncable | Bool | If this boolean is set to true, this Data Cluster can be synced to another Data Cluster. This can be useful if you have matching sprites or matching colors that you want to always equip simultaneously. In the demo, for example, the character can have facial hair and hair on their head. If both of these options are enabled and you try to change the hair color of either cluster, both clusters end up with the same color. This ensures you won't have to manually set the hair and facial hair colors separately, which might save some time during the creation process. |

| | | |
|---|---|---|
| Sync Type | Enum | If the boolean “Is Syncable” is enabled, this enum allows you to determine if segments (sprites), colors, or both should be synced with another Data Cluster. This can allow for scenarios where, for example, hairstyles (on head and face) don’t have to be synced, but hair colors do. |
| Sync Icon Color | Color | This field determines the color of the Sync Icon in the Creator Window. This only serves an aesthetic purpose. The reason why you’d want to change this color is to keep your synced clusters organized if you have multiple Data Clusters that are synced to one another. |
| Syncable Clusters By ID | List<int> | This list can be used to define which specific Data Clusters should be synced to this one. In the demo, the ‘Facial Hair’ cluster with ID 7 links to ID 8 in this field. This means that cluster 8 will be changed if cluster 7 changes. If you link back from cluster 8 to cluster 7 as well, both clusters are in control of the other cluster. |

# 3.2 Add a custom Data Cluster

Whether you are starting with an empty Character Data Manager or adding Data Clusters to an existing Character Data Manager, making custom Data Clusters appear in the Character Creator can greatly enhance your experience with this asset. Here, we'll go over adding a new Data Cluster to the pre-existing clusters you're given with the download of the asset.

**STEP 1.**

Open the Scriptable Object called "PixelCharacterCreator2D" in the inspector. This will give you an overview of all the Character Data Clusters at the top.

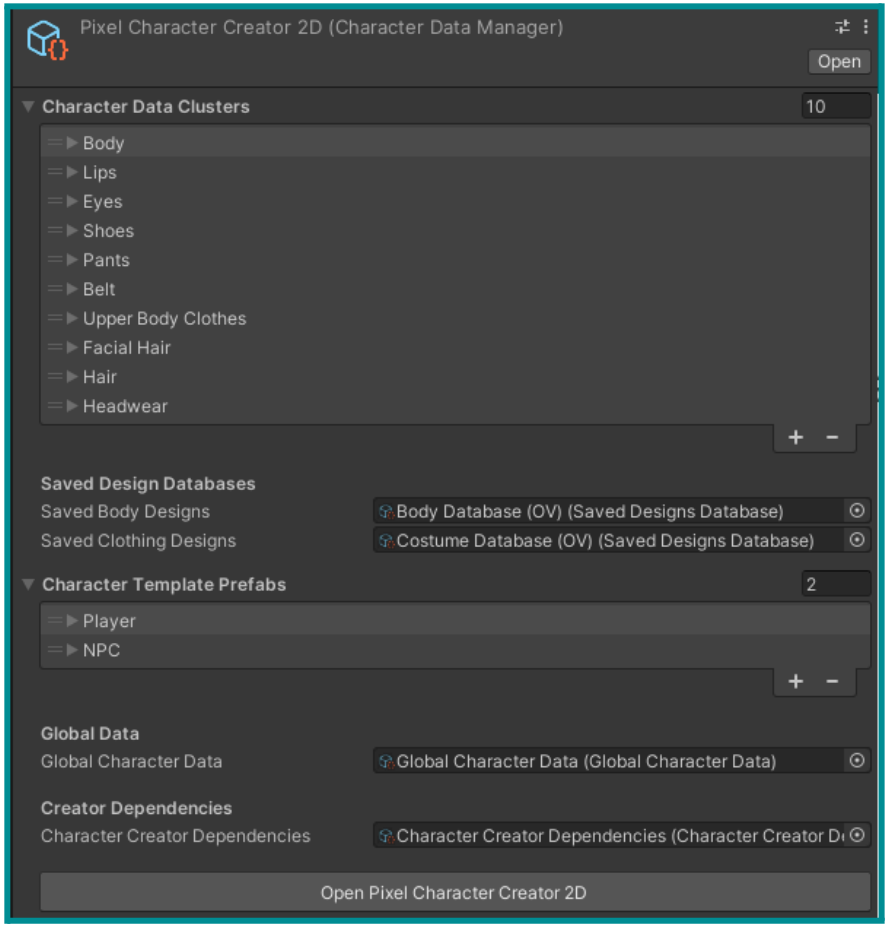
**STEP 2.**

Click the + icon to add a new Character Data Cluster. This creates an exact copy of the Data Cluster at the bottom. Let's change the 'Label' variable of the new Data Cluster to "Gloves".

**STEP 3.**

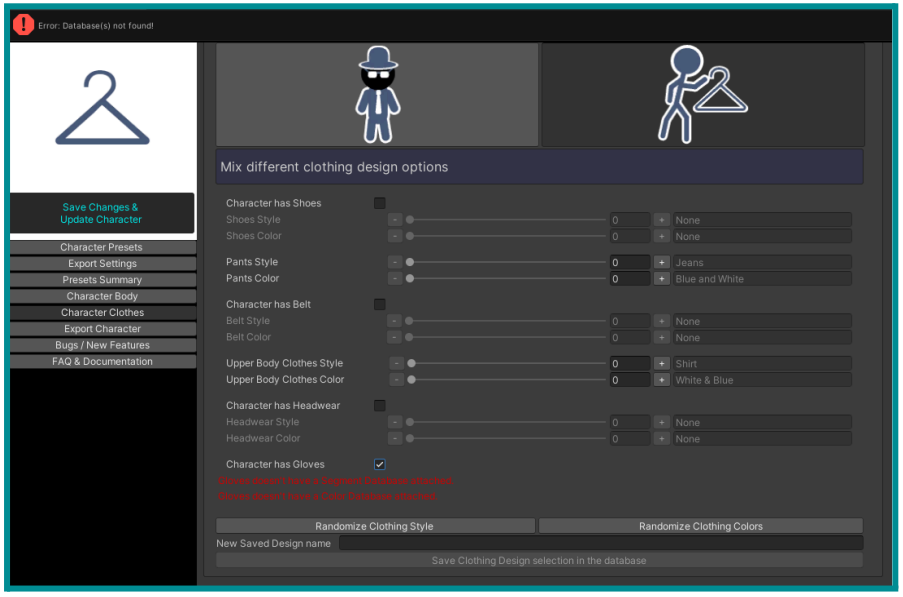
Let's keep most of the variables the same as the "Headwear" data cluster. I'll note the other variables that I want you to change below:

- Remove the Scriptable Objects in the "Character Segment Database" and "Character Color Database" sections. We'll add new databases in a bit.
- If you so choose you can add a dialogue tag for the new Data Cluster for both the segment and color. This doesn't impact the process of the Character Creator itself, so in this tutorial, I will keep these string fields empty.



**STEP 4.**

Now open the Character Creator by clicking on the button “Open Pixel Character Creator 2D” or double-clicking on the Scriptable Object itself. Upon opening the Creator Tool, you will be greeted with a red text at the top of the window: “Status Database(s) not found!“. This message is meant as a reminder that the Character Creator won’t function properly because we just added a new Data Cluster without any databases attached. For now, we can choose to ignore this message and try to create a character nonetheless. Click on the “Presets Summary” tab on the left, choose a name for your character, and press “Create new Character”.




**STEP 5.**

If we go to the “Character Clothes” tab, you should be able to see a new section with a tick box next to it called “Character has Gloves”. You can enable and disable this option but we aren’t able to select any gear. We now only see two messages in red:
“Gloves doesn’t have a Segment Database attached”
“Gloves doesn’t have a Color Database attached”

These messages will go away if we add databases to our new Data Cluster. Let’s start by adding a Segment Database. You can read about that in section 3.3.

# 3.3 Create a Segment Database

A Segment Database contains Character Segments which the user can populate the Character Creator with. A Character Segment is a custom class that consists of a string field (the name of the segment) and a Sprite field (for the sprite sheet associated with this Character Segment). In this section, you'll find a step-by-step example of how to create a Segment Database, add a segment to the database, and lastly, connect this new Segment Database to the Character Data Manager.

**STEP 1.**

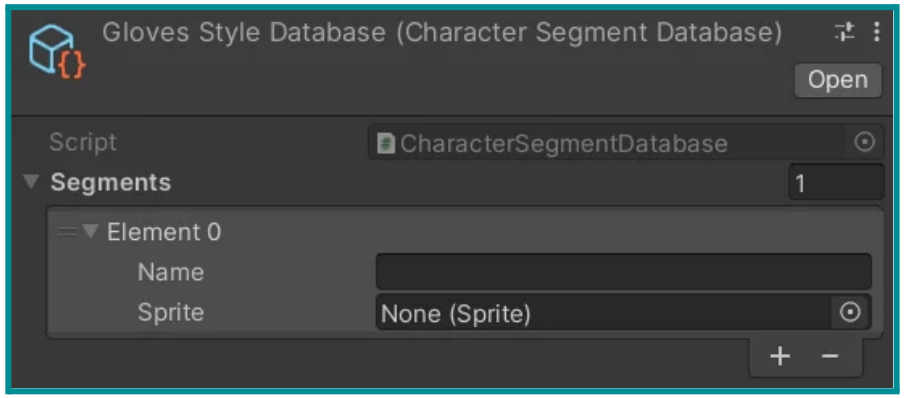
The first thing to do is decide where you want to store your new Segment Database. For this tutorial, I recommend placing this new database in the same location where the other Segment Databases are kept. Here is the path to finding them: *Assets / PixelCharacterCreator2D / OverheadCharacter / Databases / SegmentDatabases*

Once in this folder, click the right mouse button and choose:
*Create > Database > Character > Segment Database*

I will name it "GlovesStyleDatabase" to stay consistent with the rest of the database names, but you can name the database whatever you like.

**STEP 2.**

Upon inspecting the new Scriptable Object you just created you will see that it doesn't contain any "segments". If you add a segment to the list your view will match the image on the right. Now we have to add a name and a sprite sheet to this segment.




- For simplicity's sake, I will name this segment "Default Glove".
- For the sprite, we'll want to add a new sprite sheet that shows the design of this segment in all of the angles in which the character can stand and walk. I have prepared a sprite sheet for this tutorial. Feel free to download this sprite sheet so you can use it to follow along. You can download it by clicking the button below.

Download the Glove Sprite Sheet

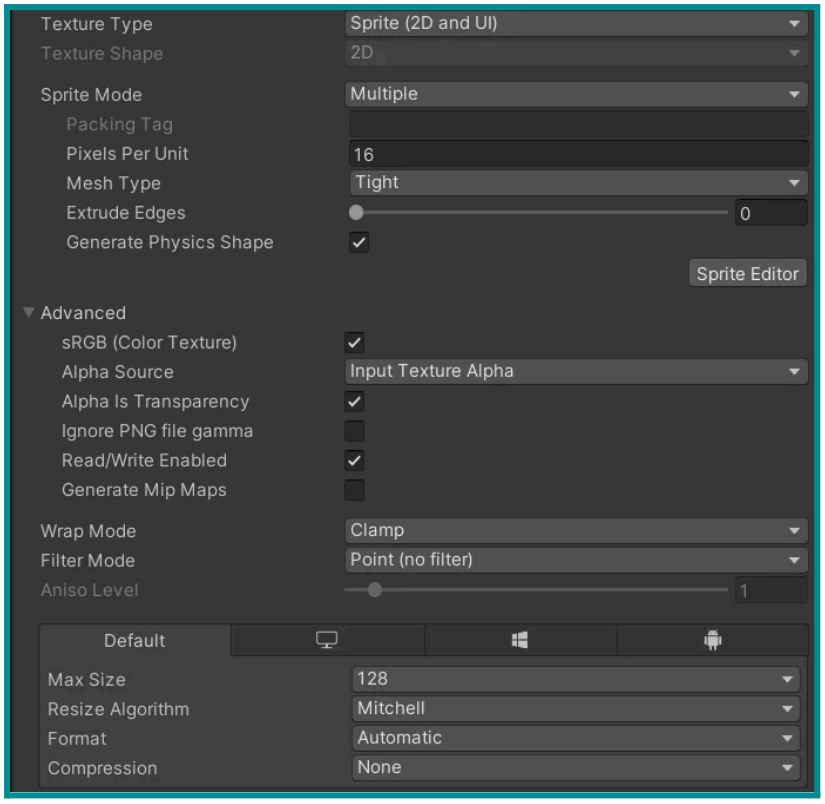
**STEP 3.**

With the sprite sheet downloaded, add it to the Unity Project and place it in a folder that makes sense to you. I recommend placing it near the other Character Segment Sprite Sheets. You can find those at the following directory path: *Assets / PixelCharacterCreator2D / OverheadCharacter / Templates / TemplateSprites*

**STEP 4.**

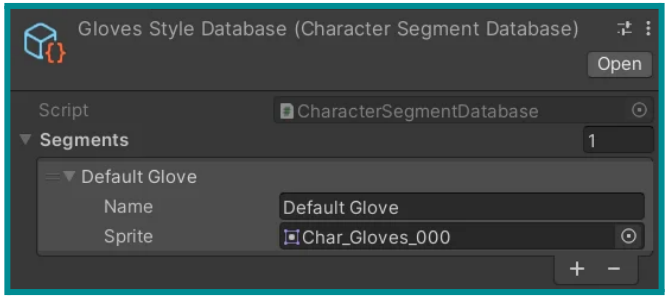
Let's check if the Texture Import Settings match the settings that we need. Click the sprite sheet you just added to the project. If your export settings aren't automatically set to the settings in the example screenshot below, you can fix the settings quite easily by doing the following:

1. Click  in the top-right corner (above the "Open" button).
2. Select "CharSegmentOverheadExpSettings".
3. Click "Apply" in the bottom right of the Import Settings windows.




**STEP 5.**

Drag the newly added sprite sheet into the empty sprite field of the Default Glove. Your GlovesStyleDatabase should now match the one below. You've now created an equipable glove. All that's left to do is ensure we can see this glove inside of the Character Creator.




**STEP 6.**

Return to the PixelCharacterCreator2D Scriptable Object and go to the Gloves Data Cluster we created. We have to add our newly created GlovesStyleDatabase into the open slot called "Character Segment Database". You can add it by dragging the database into the field or by clicking  next to the empty field and selecting GlovesStyleDatabase.

You have successfully added a Segment Database to the new Character Data Cluster. If you open the Character Creator, you'll see that there is still an error. However, if we try to create a character, we can now see that the Gloves Style was added to the Character Creator. We aren't able to select other gloves, because we only added a single one. This means that we'll always equip the Default Glove if the "Character has Gloves" checkbox is set to true. If you want to add more gloves to choose from, you can do so by repeating the previous steps, 2 through 5.

# 3.4 Create a Color Database

A Color Database contains Character Color Groups which the user can populate the Character Creator with. A Color Group is a custom class that consists of a string field (the name of the Color Group) and a List (with an infinite amount of color fields). In this section, you are going to read about creating a Color Database, adding two color groups to the database, and connecting this new Color Database to the Character Data Manager. If you read section 3.3 before this one, you'll find that a lot of steps are roughly the same. That should help expedite this process a little.

**STEP 1.**

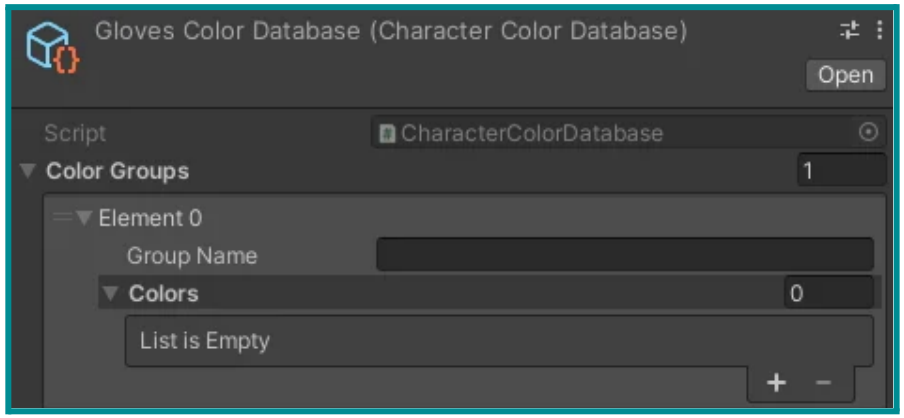
Firstly, you'll have to pick a folder where you want to store your new Color Database. For this tutorial, I will be placing this new database in the same folder where the other Color Databases are kept. Here is the path to finding them: *Assets / PixelCharacterCreator2D / OverheadCharacter / Databases / ColorDatabases*

Once in this folder, click the right mouse button and select:
*Create > Database > Character > Color Database*

I will name it GlovesColorDatabase, but you can name the database whatever you prefer.

**STEP 2.**

Upon inspecting the new Scriptable Object you'll notice that it doesn't contain any Color Groups yet. If you populate the list with a Color Group, your inspector window should match the preview below.



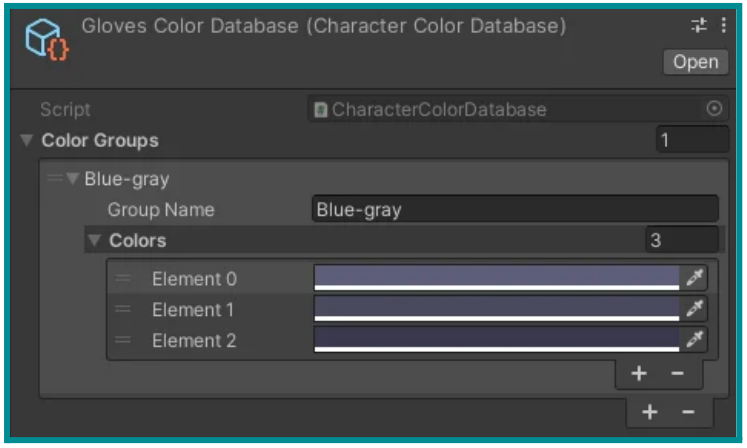

We can now add a name and define the colors for this Color Group:

- I'll name this Color Group "Blue-gray". This is the default color because the gloves were created with this color.
- Now we have to add the hex codes for every color that we used in the gloves sprite sheet. I recommend using software, like Aseprite, to extract all hex codes from an image file so you don't have to find all hex codes yourself. If you load in your image in Aseprite you will be able to see every hex code that was used.

- Now it's just a matter of copying those hex codes inside of our Color Group. To help move the tutorial further along I saved you the trouble of finding the hex codes of the gloves. You can find them in the three squares below. Copy those hex codes into the color fields.

| #5D5D79 | #49495E | #39384C |
|---|---|---|

With the colors added, your Color Group should now look like the one in the example below.




**STEP 3.**

You may recall me stating that we need to add two Color Groups. That's because the first Color Group holds all the default colors. Those default colors always have to match the colors of the sprite sheet. For the second Color Group, we are free to add whatever colors we want. You can perform the same steps you made for the first group, but this time either add your own hex codes or borrow a few more hex codes from me. I have put a few hex codes below, just in case you need them. When adding colors, don't forget to set the alpha to 1. Otherwise, a warning message will pop up in the Character Creator.

| #A7FA9D | #63DD42 | #1B9A24 |
|---|---|---|
| #DD4242 | #B42A2A | #841919 |
| #FFD956 | #DCB325 | #AB923A |
| #6EE1FA | #3CB1CA | #1F7B8E |
| #FBFBFB | #D2CCE2 | #ACA7BC |

**STEP 4.**

After adding more Color Groups, return to the PixelCharacterCreator2D Scriptable Object. Go to the Gloves Data Cluster and add our new GlovesColorDatabase into the open slot called “Character Color Database”. As a reminder: you can add it by dragging the Color Database into the field or by clicking  next to the empty field and selecting the GlovesColorDatabase.

You have successfully added a Color Database to the Gloves Character Data Cluster. If you open the Character Creator window now you will see that there are no more errors. You can now select both the Gloves Style and Gloves Color in the Character Creator. You’ll notice that you can switch between the default color and the other color(s) you added. The more Color Groups you add to the Color Database, the more choices you’ll have in this window.

# 3.5 Syncable Data Clusters

Syncing Data Clusters will ensure that certain Character Segments and/or Colors will always match accordingly. An example of this could be the color of hair on a character's head and facial hair color.

## How to sync up two or more Data Clusters?

**STEP 1.**

Determine which Data Clusters should be synced. Additionally, consider whether you want both the segments and colors to sync up or if you just want segments or colors to be synced.

**STEP 2.**

Navigate to *Assets / PixelCharacterCreator2D / OverheadCharacter*, then select the scriptable object "PixelCharacterCreator2D" and look at it in the inspector. Then navigate to the Data Clusters you want to sync up.

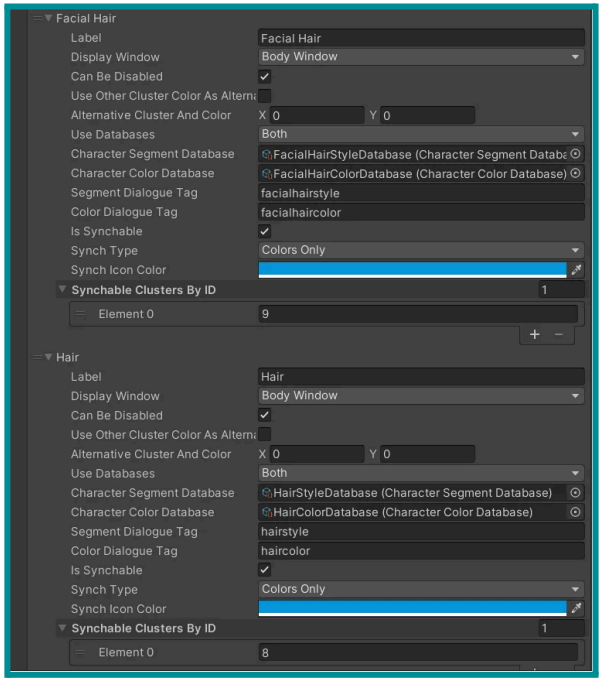
**STEP 3.**

Set the "Sync Type" for both Data Clusters to any of the following options that match your intention the best:

- Segments Only (if only sprites should be synced. For example, if you have a matching headwear style for every upper clothing option, and if said headwear should always be equipped automatically when selecting upper body clothes.)
- Colors Only (if only colors should be synced. For example, if you want to match hair color with facial hair color.)
- Both (if both sprites and colors should be synced.)

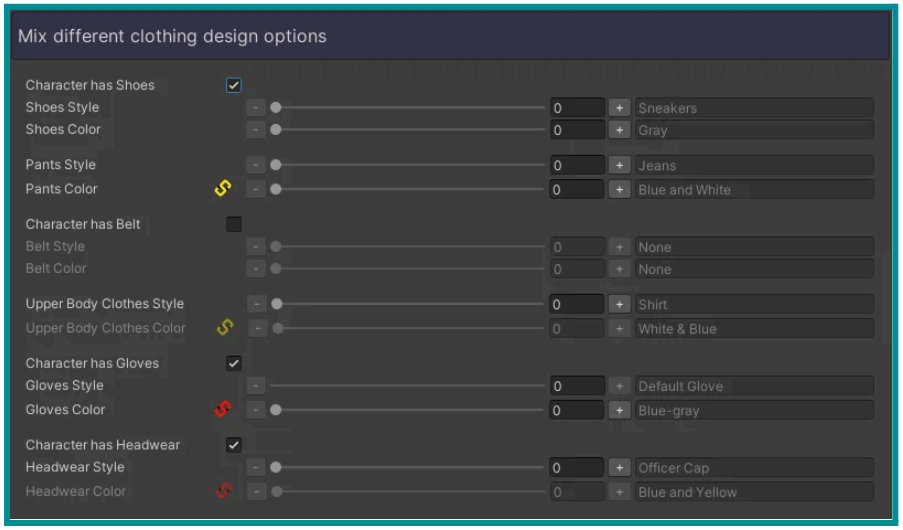
**STEP 4.**

Set the "Sync Icon Color" for both Data Clusters to a color that will help you recognize that these clusters are synced. The Icon looks like this: This icon matches the color you choose. This way you can set up multiple data clusters that sync with each other. By color-coding your Syncable Data Clusters, you create a more understandable workflow for yourself. Don't forget to set the Alpha to 1. Otherwise, your colors will be opaque.



**STEP 5.**

We now have Data Clusters that are ready to be synced. However, we're not there yet. To achieve the result in the screenshot below, we have to tell the Data Cluster which other Data Clusters they need to sync up with.




To reproduce the contents of the screenshot we have to add one or more integers to the "Syncable Clusters By ID". The ID we need to reference is the same as the indexing position in the Character Data Clusters List. For example, the Body Data Cluster has ID 0, because it is at the top of the list. Consequently, the Headwear Data Cluster's ID is 9 if the list contains 10 Data Clusters in total because it is at the bottom of the list. By counting the index position of the cluster inside of the list, you'll know what the ID is and what ID you'll need to refer to.

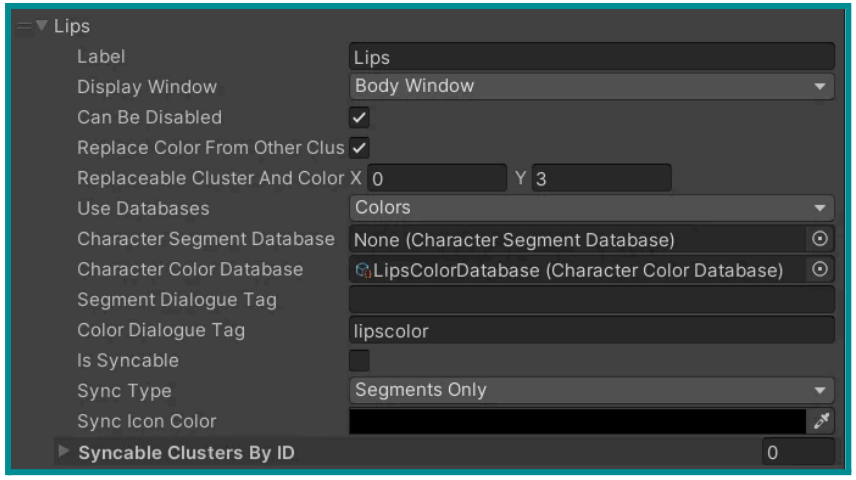
Keep in mind that if you eventually decide to rearrange the order of the Data Clusters, you'll have to adjust the Syncable Clusters By ID segment as well, as this doesn't change automatically. Since I've added a Gloves Data Cluster to the list, my order has changed, making the Headwear Data Cluster's ID 10 instead of 9.

# 3.6 Replacing colors from another cluster

In this section I'll give step-by-step instructions on how to use a Data Cluster to replace an existing color from another cluster.

## Why and when to replace colors from other clusters?

Before I show you how to implement this properly, it's important to know why and when to use this feature. The core reason for this function existing is to add or remove a color at will, without affecting the base design of any of the Character Segments. In the screenshot below, I show a Data Cluster that implements this functionality. This is an optional Data Cluster that allows you to add or remove a lip color.




It may sound simple, but there is more going on. The Data Cluster in the screenshot:

- overwrites one specific color from one specific cluster;
- can be turned off, without the character losing a mouth (as the mouth's base design is part of the body cluster);
- can overwrite the lip color, no matter the body's skin tone. This is possible because we're not looking for a specific hex code, but rather an index number;
- doesn't have an associated Segment Database, because it will be using a Segment Database from another Data Cluster.

An alternative use case for this feature could be the addition of optional highlights to the character's hair, by replacing a specific color from the hair. This wouldn't change the design of the hair itself, but just one specific color from that hair.

## Setting this up yourself

**STEP 1.**

Navigate to Assets / PixelCharacterCreator2D / OverheadCharacter, then select the scriptable object "PixelCharacterCreator2D" and look at it in the inspector. Then navigate to the Data Clusters and add a new one with the + icon.

**STEP 2.**

Decide what color from which Data Cluster you want to replace. In the previous example, we wanted to overwrite the default lip color of the character with an extra lip color. The default lip color is found inside the Color Database linked to the “Body” data cluster. Since the “Body” Data Cluster is at index 0, this will be the X-value in the Vector2 variable “Replaceable Cluster And Color”. For the Y-value, we need the index number of the color we want to replace. The lip color of the character is the fourth color (index 3) inside of the “Body” Color Database.
In short, this means that if we set “Replaceable Cluster And Color” to Vector2(0,3), we are replacing the fourth color inside the first Character Data Cluster’s Color Database.

**STEP 3.**

Set the following variables:

- set “Can Be Disabled” to true;
- set “Replace Color From Other Cluster” to true;
- now decide which color from which Data Cluster you want to override and set the Vector2 field “Replace Cluster And Color” to that respective value;
- set “Use Databases” to “Colors”;
- ensure the Character Segment Database field is left empty.
- add a Character Color Database for this specific cluster. Inside that database:
    - add at least one Color Group.
    - add just 1 color per Color Group.

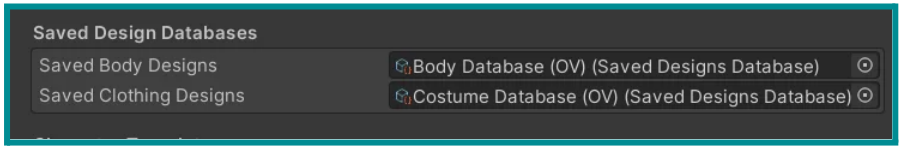
You are free to set the remaining variables to your liking.

**STEP 4.**

You should now be able to select a new color inside of the Creator window. If you update the look of the character, this new color should overwrite the color you selected. As you’ve probably noticed, this is a very situationally useful feature and may be tweaked in the future depending on user feedback.

# 3.7 Saving your designs for later use

There might come a time when you'd want to have multiple characters wearing the same clothes or having multiple characters of the same race but with different clothes. While it is possible to recreate the clothes and body types over and over again, why not make this a bit easier for yourself? What if you could save a design you created for later and fast-select that design for other characters? Well… you can!




The screenshot shows another section within the Character Data Manager. This section is in charge of holding the "Saved Body Designs" and "Saved Clothing Designs" for you. Let's see how it works and how you can easily save and fast-select your designs.

## Create Databases for Saved Designs

If your Data Manager's Saved Designs section doesn't match the screenshot, chances are you haven't set up Saved Design Databases yet. Before we can save our designs we have to create two Saved Design Databases. One for the body section and one for the clothes section. You can do so by selecting *Create > Game Between The Lines > Character Creator 2D > Database > Saved Designs Database*.
After you've created those databases, give them proper names so you know which database holds which types of data. After that, just drag these databases into their corresponding fields in the Character Data Manager. You're now done with setting up the databases.

Note: if you forget to create these databases and/or forget to drag them into their corresponding fields, you won't be able to save your designs and select them for new characters. Error messages will pop up in the Console instead.

## Mix clothes or choose an existing design

Upon navigating to either the Character Body or Character Clothes window, you will find two buttons at the top of the page, both with an icon. The icon on the right displays a character hand picking their clothes. This is the section for handpicking every little piece of the character's design. The icon on the left displays a character in a costume. In this section, you can scroll through your saved designs and fast-select any design you've previously created.

# Mixing different body/clothing design options

If you are in the section that matches this icon, you'll be able to hand-pick different body or clothing options. This is also the place where we get to save our designs. We can do so by doing the following:

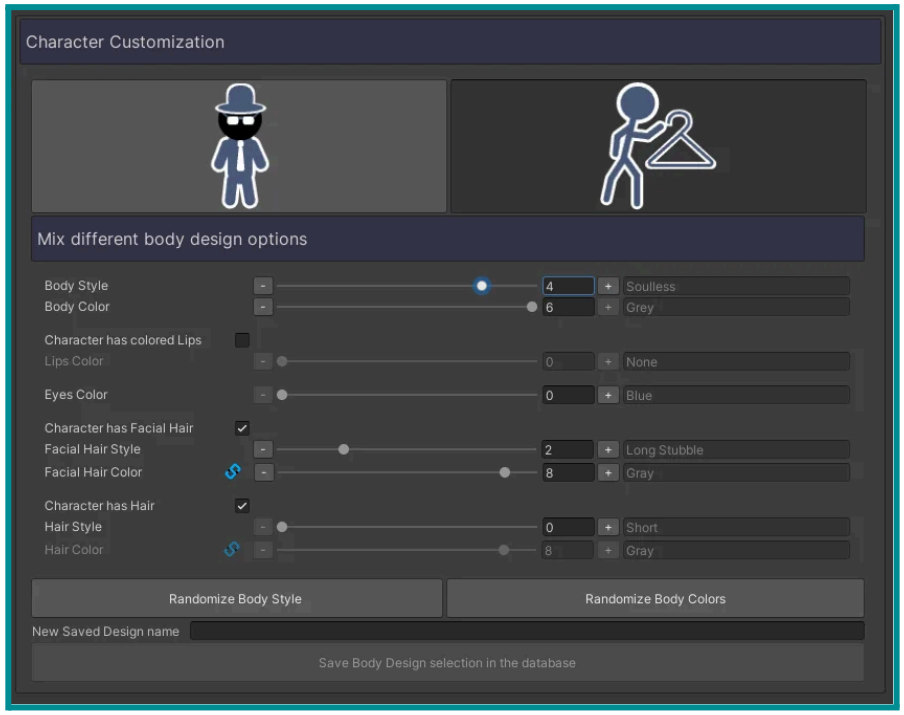
**STEP 1.**

Select the body or clothing you style you design to save. Be sure to hit the save button as well so you know what the design looks like.

**STEP 2.**

Navigate to the bottom of the window to the greyed-out button that says "Save Body/Clothing Design selection in the database". We can enable this button by giving our currently equipped body or clothes a name. Fill in the name for the saved design in the "New Saved Design Name" field, then press the button below to save the design. Congratulations, you have just saved your first body/clothing design to a Saved Designs Database.

Note: the button will remain greyed out if no Saved Design Databases have been defined.



## Pick from existing body/clothing designs

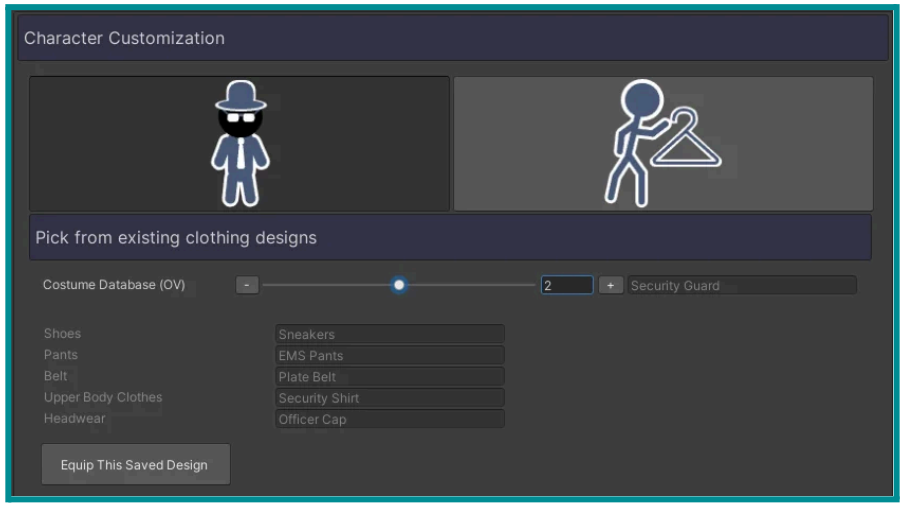
Now that we have saved at least one body or clothing design, we should try to equip it with another character. To equip your newly saved design, follow the steps below:

**STEP 1.**

If the design you just saved was a Body Design, navigate to the Character Body selection and then click the button that matches this section's icon. If the design you just saved was a Clothing Design, navigate to the Character Clothes selection and then click the button that matches this section's icon.

**STEP 2.**

Your view should now resemble the screenshot. Your view will differ depending on the amount of saved designs you have stored in the database. Select the design you want by inputting the index number you want or by dragging the circle on the bar left or right. This will change the text on the right. The greyed-out text will display the name of the saved designs in the database. The name you just saved should (also) be in there. Below you can view all of the specific segments your character has equipped. At the bottom, there is a button called "Equip This Saved Design". If you click that button, the design you selected will be equipped. Please note that if you don't see any changes with your character, you need to click the "Saved Changes & Update Character button".



# 4. Data Manager Part 2: Character Templates

In this chapter, you're going to learn what Character Templates are and what their significance is regarding the creation of your new characters. We'll take a look at how Character Templates can be set up and we'll cover tips and suggestions for an improved workflow.

## 4.1 What are Character Templates?

Character Templates function as a blueprint for what your Character Prefab output is supposed to look like. In chapter 1 you learned that you have the option to export a Character Prefab. That's where the Character Template comes in. The character that will be exported at the end of the creation process will be a mirror image of the Template Character you define, except for the sprite. This means if you give your Template Character the option to walk with a Character Controller, the newly created character's prefab will be able to do the same! You may be able to imagine what kind of a time-saver this feature can be, provided you have a lot of characters that need to behave in the same way. Without further ado, let's take a look at how you can set this up for yourself!

## 4.2 Setting up your Character Prefab

Think of a Character Template as a blueprint that all new characters should listen to. Each newly generated character will use the Template as its base and take from it all the input needed to make the new Character Prefab. Below you can follow step-by-step instructions to set this up.

**STEP 1.**

The setup of your Character Template Prefab starts with setting up the character you want to use as a base. So start with a simple game object and add everything the character needs to function. What do you want your character to look like in terms of component layout, child objects, and scripts? If you're creating a player template you will most likely add scripts or components that have to do with movement and controls. If it's a template of an NPC character, think of all the behaviors that the NPC will need to function properly.

**STEP 2.**

Now that you've decided what you want out of this character, we need to check if it fits the requirements. We do need to follow a few guidelines. Check if your Character Template GameObject or one of its children contains the following components:

- Sprite Renderer;
- Animator.

This is the bare minimum that's needed for a Character Template. If the GameObject or a child of that object contains both components, you're good to go for now.

**STEP 3.**

If you haven't done so already, Make a Prefab of your Template Character GameObject. Then navigate to the Character Data Manager for the final steps.

**STEP 4.**

Inside the Character Data Manager, add a Character Template Prefab by clicking the  icon. You can create as many templates as you want, but I recommend only adding the amount you actually need.

**STEP 5.**

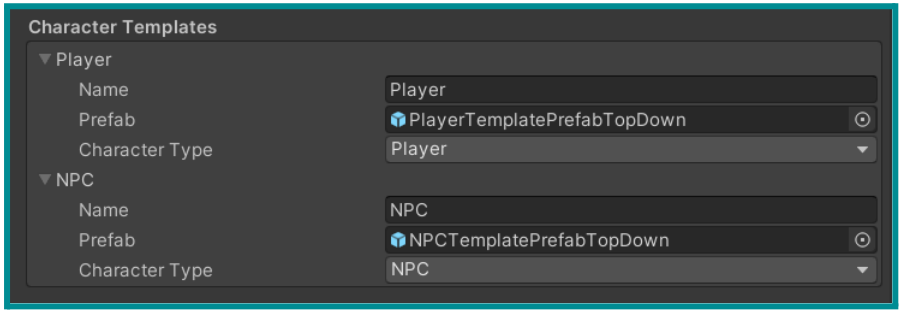
Give the newly added template a name that's descriptive enough for you to know what kind of Character Template it is. If you don't give the template a name this will cause a warning message to appear, but you can continue the creation process nonetheless.

**STEP 6.**

Drag the Template Character Prefab you recently created into the empty "Prefab" slot. Keeping this slot empty will cause errors during the creation process.

**STEP 7.**

Decide whether this character is a player or an NPC character. This is mostly meant for the sake of keeping your templates organized. After you've performed these steps, your character template should now somewhat resemble the screenshot on the right.




## 4.3 Multiple templates

It is recommended that you use multiple character templates so that you can easily switch between different templates and thus potentially streamline your process better. For example, suppose you are creating a game where you need multiple players and multiple NPSs. In this case, you can easily switch between creating players and NPCs by creating at least two character templates. Then you only need to select which template you want to use with one press of a button.

An example of this can be seen in the default Character Data Manager that comes with the download of the Pixel Character Creator 2D asset. It is by no means necessary to use the same setup and you could use this tool with just one template for all characters or however many templates you fancy.

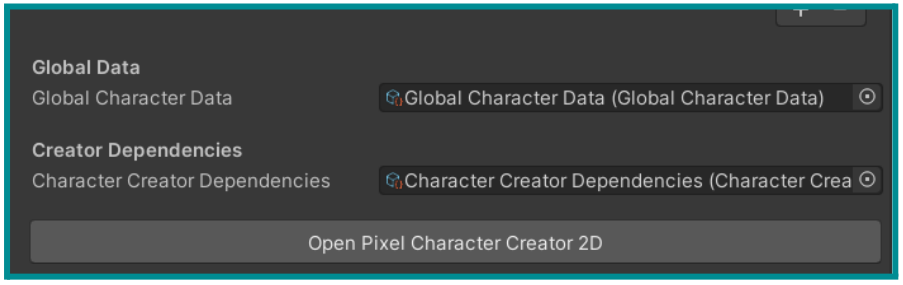
# 4.4 Remaining Data Manager Fields

## Global Character Data

The Scriptable Object named “Global Character Data” is an object that holds data that needs to be accessible for multiple systems, hence the term ‘global’. The data inside of this Scriptable Object is also used by the Basic Dialogue System. This data mainly consists of information about what the character looks like and what kind of clothes the character wears. You can use that information to reference design choices such as clothing in dialogue or to use certain logic such as an event that triggers only when a character meets a certain design choice.

## Character Creator Dependencies

The scriptable object called Character Creator Dependencies is an object that mainly contains information about visual elements belonging to the Character Creator itself. This includes icons that are used and other GUI elements. With this Scriptable Object, you as the end user do not have to do anything substantive. The only important thing is that this Scriptable Object is added to the Character Data Manager. If not, certain visual elements will be missing from the Character Creator.



# 5. Animations

The Pixel Character Creator 2D asset allows you to automatically animate characters. This is done with the help of both sprite sheets and a template character prefab. In this section, we'll go over how the animating process works and how you can add custom animations.

## 5.1 Automatically animated characters

If you are using animations in 2D Unity games, odds are you're quite familiar with having to manually slice sprite sheets and create animations frame by frame. If you have many different characters this process alone can take hours of your time. That's why this is one of the core features the Pixel Character Creator 2D provides. This asset can automatically slice the sprite sheets for you and animate your characters frame-by-frame with just one click of a button. So how does it work, and more importantly, how do you set it up so you can use it with your characters? Let's find out!

## 5.2 Animation Templates

If you've read the previous chapter, you'll have heard about Character Templates. Animation Templates are similar in principle but are as the name implies a template that's purely meant as a blueprint for the Animator and Animation Clips. In this subsection, I'll go over what Animation Templates are and how you can set them up properly.

### What are Animation Templates?

Animation Templates are meant to be a blueprint for your custom character's animations. This blueprint determines:

- which sprites from a sprite sheet are used in the animation;
- the duration of the animation;
- the position of the animation frames;
- whether or not the animation contains an Animation Event.

Defining your Animation Templates properly will allow you to automatically animate your characters at a fast pace. Do take into account that characters created with this animation template will be animated accordingly. If we want to add new animations to our characters and automate the creation process for these animations as well, we'll have to add those animations to the template as well.

### Setting up Animation Templates

If you are using the standard characters that were included in the package, you may skip this subsection. However, if you decide to use your own characters and custom animations you can learn how to set that up here. For this tutorial, I will presume that you already have a Template Character Prefab prepared to animate. If you don't have a Character Prefab yet, I recommend preparing one first by following the steps in Chapter 4.2.

**STEP 1.**

Animate your Template Character as you normally would. This is the only time when we'll need to manually slice the sprite sheet and set up the animation frames ourselves. In this tutorial, I won't go over the manual process of cutting sprites and creating animation frames. In this video, Brackeys explains how to create your animations with sprite sheets.

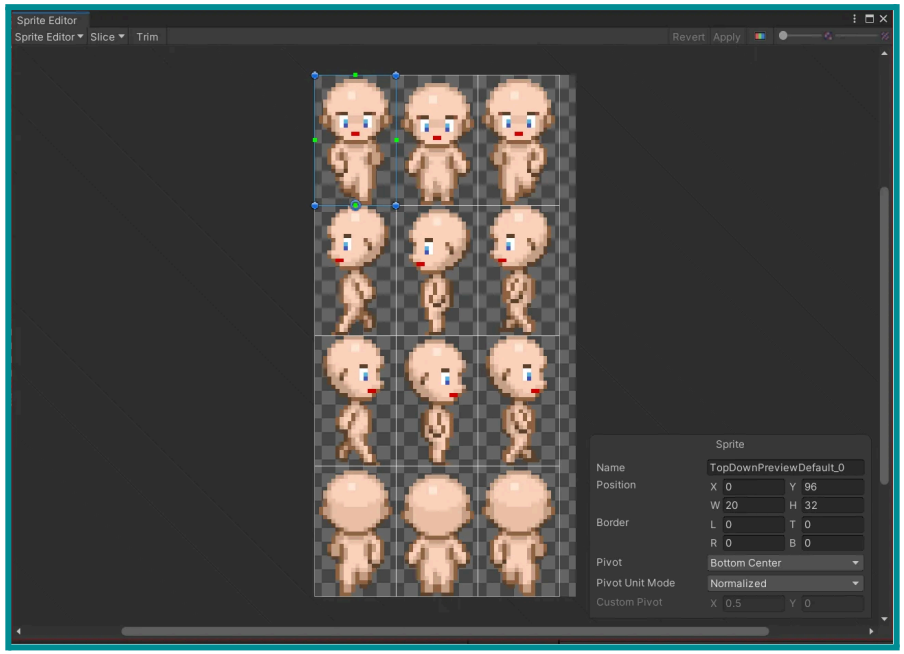
An important thing to keep in mind is that all individual sprites should have the same dimensions. You'll also have to update the Sprite Export Settings to match the sprite sheets you provide. You can learn more about how to do so in Chapter 7.3.

**STEP 2.**

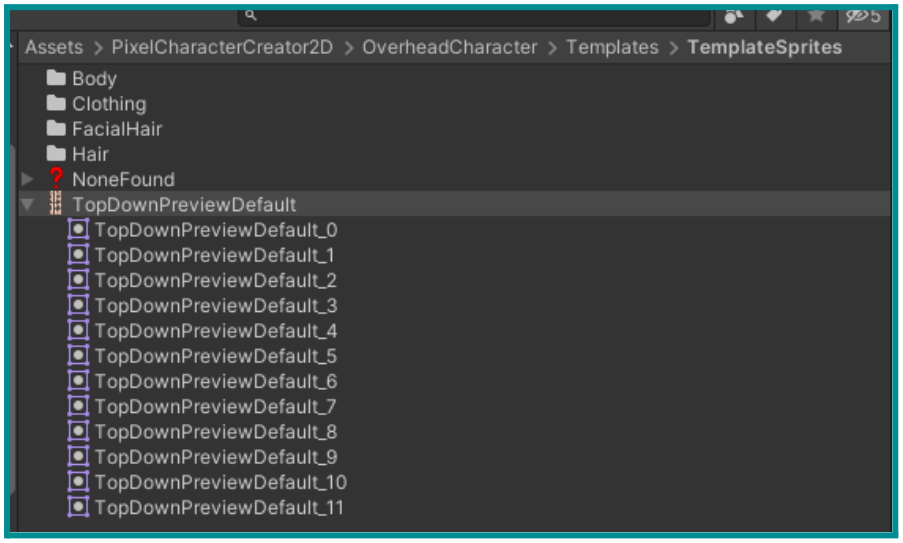
Now that you've set up the animations you want, let's start with organizing. Create a folder where you can keep all of your template Animation Clips and your Template Animator Controller. If you need an example of how I've set this up, you can find that by navigating to *Assets > PixelCharacterCreator2D > OverheadCharacter > Templates > TemplateAnimations*.

**STEP 3.**

Next up, go to the sprite sheet you sliced. It should now consist of multiple individual sprites. Navigate to the Texture 2D Import Settings of your sprite sheet and click on the "Sprite Editor" button. With the Sprite Editor opened you can click any of the sprites to reveal bits of information such as the Name, Position, Border, and Pivot.




The only info we are currently interested in is the "Name". For the Character Creator 2D to animate the sprite correctly, we'll need to tell the Character Creator in which order the sprites are laid out. You can do so by naming them in alphabetical order or, as I like to do, give these sprites the same name but with a number attached at the end of the string. If you need an example of this, you can navigate to *Assets > PixelCharacterCreator2D > OverheadCharacter > Templates > TemplateSprites*. There you'll find a sprite sheet called "TopDownPreviewDefault". If you click the arrow next to its name you'll see the same view as seen in the image on the next page. That is how I prefer to name the individual sprites.




As for the order in which the sprites should be numbered, the image below shows the exact order you should follow. Of course, don't forget to click "Apply".


**STEP 4.**

Navigate to *Assets > PixelCharacterCreator2D > OverheadCharacter* and select the Data Manager called "PixelCharacterCreator2D". Then open the Character Creator 2D and select the Character Template you added. Ensure this character is correct and that it has the animations you want.

**STEP 5.**

Open the Export Settings and scroll down to the "Animation Export Settings". Enable the option "Export Character Animations". If you see an error message pop up, your selected Character Template likely doesn't support animation exports. View the debug section to find out what the problem is. If there are no error messages, we're good to continue.

**STEP 6.**

Near the bottom of the "Animation Export Settings" section, you'll find a button called "Fetch data from Runtime Animator Controller". This extracts all relevant data from the Animator Controller that's attached to the Character Template you selected. Upon opening the "Template Character Animations" List, you'll see the following data:

- the animation's name;
- ID;
- loop time (if it's enabled or not);
- the number of keyframes the animation has;
- the frames per second;
- the amount of Animation Events;
- a list of Animation Events found in the Animation Clip with the following data:
- the frame where the event is supposed to be triggered;
- the name of the function we want to call via the event;
- the keyframes of the sprite sheets that are used in the animation.

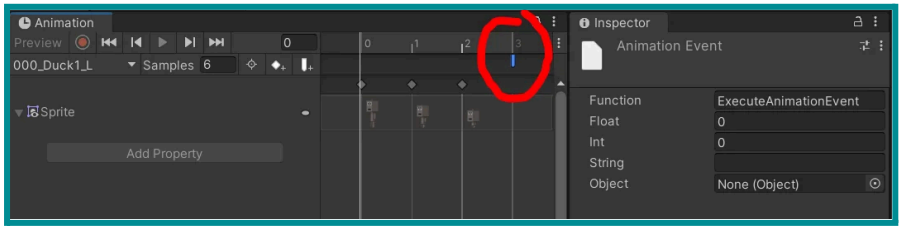
**STEP 7.**

Now that you've extracted all this data, review the data and check if everything was extracted properly. Is the animation using the correct keyframes? Does it include the animation events you need? Is Loop Time enabled when it needs to be? After you've confirmed that the data was correctly extracted, you're good to go. It is possible to manually edit these settings after extraction. Your changes will, however, be overwritten if you decide to fetch the data from the Runtime Animator Controller again. For this reason, if you find yourself wanting to adjust certain settings, it's best to do so inside the Animation Clips themselves.

# 5.3 Animation Events

## What are Animation Events?

As you may have noticed in the previous section, Animation Events can be carried over from a Template Animation to the other Animations you want to create. Animation Events are special events that are triggered upon a certain animation frame being reached. You can view an example of what this looks like in the screenshot below. Animation events are particularly useful when you want to run a specific piece of code at a specific point in time during the animation.

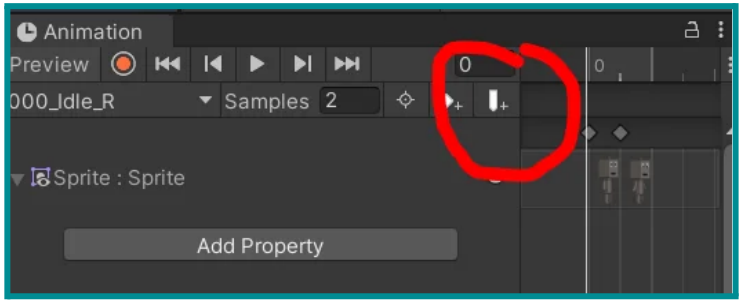



In the example above, I created an Animation Clip of a character that can duck down. Upon Ducking down to the ground I want to trigger an Animation Event as soon as the animation has ended. This way I can, for example, signal to the Character Controller that we need to change a few things such as shrinking down the character's hitbox or decreasing the character's movement speed.

## How to set up Animation Events?

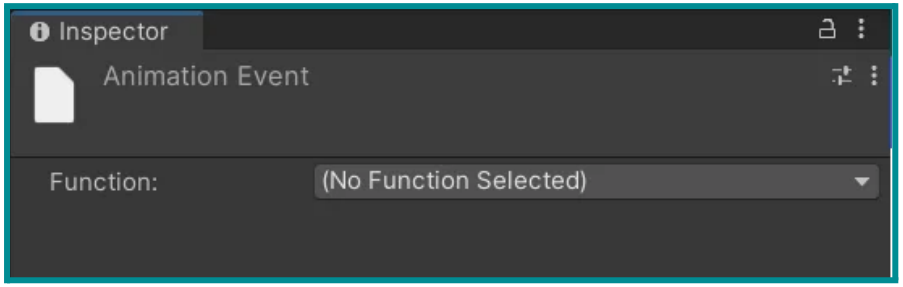
**STEP 1.**

To set up animation events you have to select a frame in the Animation Clip where you need your logic to trigger. Then click the  icon. Look at the screenshot to see where you can find it if you don't know where it is.



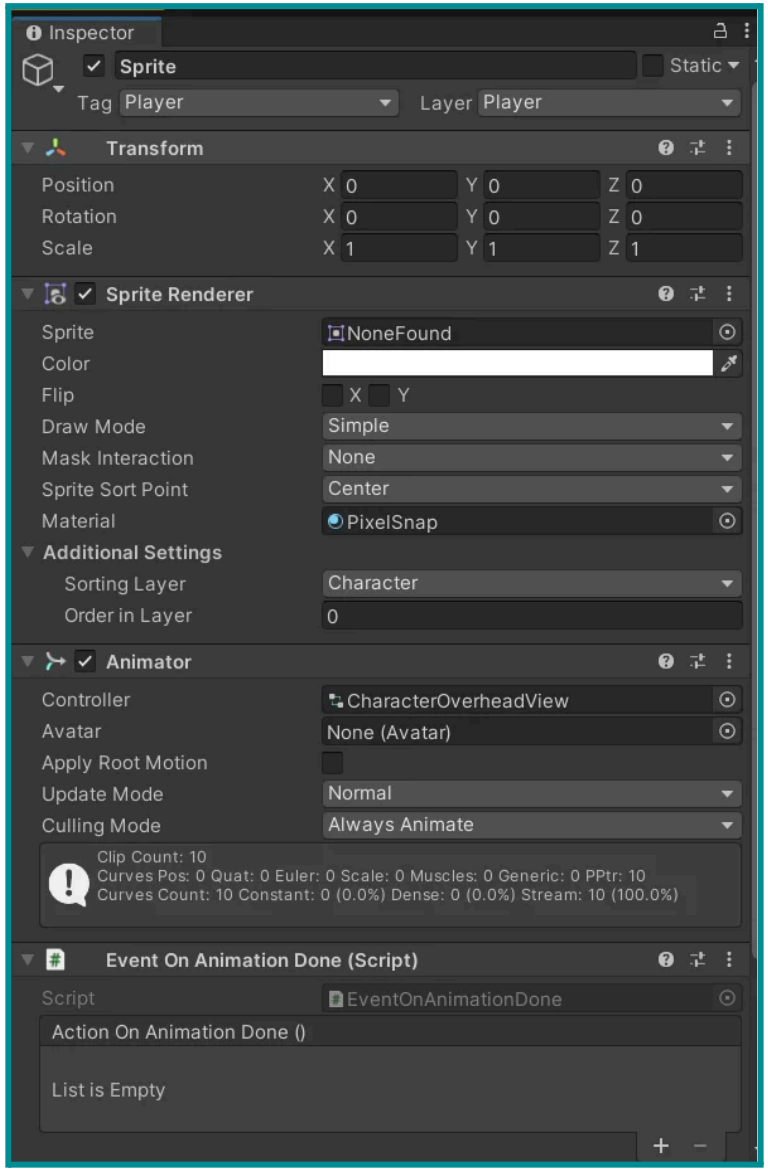

**STEP 2.**

After clicking the icon, you'll now be able to click the Animation Event inside of the Animation Clip's timeline. You will then see a similar view to the screenshot below in your inspector window.




**STEP 3.**

In order to run logic through this Animation Event, we need a script with at least one public function to reference here. This script needs to be attached to the same object that the Animator Controller is attached to. If you downloaded the demo files, you will find that one of the children of the Template Character Prefab contains a script called "EventOnAnimationDone". The sole purpose of this script is to pass along a signal from the Animation Event in the Animation Clip to another script.



**STEP 4.**

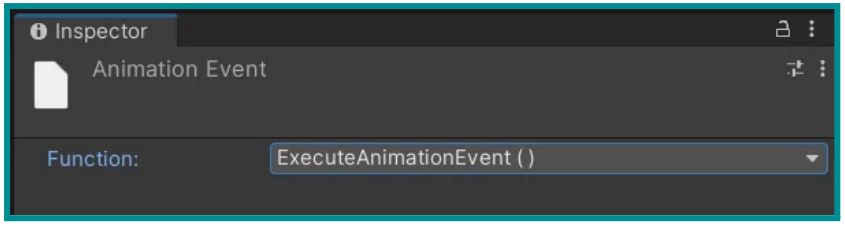
If you haven't downloaded the script along with the demo files, here's the C# code so you can create the script yourself and attach it to the object that holds the Animator Controller:

```
using UnityEngine;
using UnityEngine.Events;

public class EventOnAnimationDone : MonoBehaviour
{
    public UnityEvent actionOnAnimationDone;

    public void ExecuteAnimationEvent()
    {
        actionOnAnimationDone.Invoke();
    }
}
```

You may notice that the class, UnityEvent, and function are all public. That's because these have to be public for them to become accessible for the Animation Event. If you have attached the script to the same object as the one that holds the Animator Controller, you can now access the public void "ExecuteAnimationEvent" from the Animation Event.




**STEP 5.**

Now all that's left to do is return to the "Event On Animation Done" script and create a UnityEvent. You can do so by pressing the + icon, dragging a GameObject into the empty slot, and then selecting the script and function that needs to be triggered.

For a more in-depth explanation of how UnityEvents work, I recommend watching this video by Contraband.

## 5.4 Animation Overrides

The moment you export a character with "Export Character Animations" enabled, you will create something called an Animator Override Controller. This Controller will look at the original Animator Controller that's held onto by the Template Character Prefab. The Override Controller will then replace all animations in the original controller with the correct animations needed for the new character. This way, other settings such as animation transitions will remain intact, while the Animation Clips themselves can be changed easily.

# 6. Debug Messages

Here you'll learn about the debug messages you might receive while creating your character. Some debug messages are harmless reminders that a specific action, such as saving your changes, needs to be taken. Other debug messages may indicate there's an underlying problem that needs to be taken care of before the creation process is continued.

## 6.1 Regular notifications

Upon opening the Pixel Character Creator 2D window you may notice a bar at the top of the window. Inside that bar, you'll find a white text with an icon next to it. This is a debug message that will notify you if a certain action is required, if something is wrong, or if certain settings aren't set correctly. Below you can view all the possible debug messages as of version 1.0.0. You can expand the sections for more information about when these messages appear and how to resolve them.
Before we cover the warning and error messages, I'll briefly show you the regular notifications that may pop up. These notifications are purely meant to inform you of the current status of your workflow with the Character Creator 2D and don't require any action.

**No errors detected**

This is the default message that's shown if no errors or problems have been detected.

## 6.2 Warning messages

Warning messages don't require a lot of urgency but are usually recommended to address. It is possible to ignore these warnings and continue your workflow. Error messages will still pop up because those messages are treated with a higher priority. Those notifications overwrite warning notifications temporarily until the errors are dealt with.

**Unsaved changes detected!**

If you change the character's design without updating the character, this message pops up. This isn't indicative of a problem but is rather meant as a reminder that you should update the design of your character by pressing the "Save Changes & Update Character" button.

**The selected Template Character is unnamed. For organizational purposes, giving your template a name is recommended.**

If you see this message, you added and selected a Template Character Prefab without giving it a name. While this doesn't break anything, I always recommend properly naming your templates to avoid confusion during your workflow. This warning message will disappear as soon as the selected Template Character has been given a name.

**Scriptable Object 'Saved Body Designs' is missing. This prevents saving and quick-selecting your designs from the Body Window!**

This warning appears if you didn't add a Saved Designs Database for the body to the Character Data Manager. To fix this, click the white dot next to the empty Saved Designs Database field and select the proper database. If this database doesn't exist yet, you can create one by selecting *Create > Game Between The Lines > Character Creator 2D > Database > Saved Designs Database*. Give it a descriptive name so you know what its

purpose is, and drag it into the empty Saved Designs Database field in the Character Data Manager. The warning message should now disappear.

**Scriptable Object 'Saved Clothing Designs' is missing. This prevents saving and quick-selecting designs from the Clothing Window!**

This warning message shows up if you didn't add a Saved Designs Database for clothing to the Character Data Manager. To fix this, click the white dot next to the empty Saved Designs Database field and select the correct database. If it doesn't exist yet, create a new one by selecting *Create > Game Between The Lines > Character Creator 2D > Database > Saved Designs Database*. Give it a descriptive name to avoid confusion about its purpose, and drag it into the empty slot in the Character Data Manager. This warning notification should have disappeared.

**Scriptable Object 'Global Character Data' is missing. This causes unwanted behavior if you're using the Basic Dialogue System!**

This message indicates that the 'Global Character Data' Scriptable Object inside of the Character Data Manager is missing. If you're not using this Character Creator in conjunction with the Basic Dialogue System, this warning can be ignored.

If you are using the Dialogue System, we need to fix this issue by adding a new 'Global Character Data' Scriptable Object. You can do so by selecting C*reate > Game Between The Lines > Character Creator 2D > Database > Global Character Data*. Drag the newly created Scriptable Object into the empty slot and the warning message should go away.

# 6.3 Error messages

Error messages have a high priority and usually indicate that something is missing or broken. It is almost always required to deal with these errors before you can continue with the creation process of your character. There are a lot of different errors that may pop up. Understanding what these messages mean will go a long way in helping you solve them and continue with your workflow.

Below you will find the most common error messages that you may encounter while working with the Pixel Character Creator 2D. If you found an error message and you are unsure of how to solve it, please view the list below first to see if your specific case is covered here. If you encountered an error that isn't covered on this page, read section 6.4.

**Database not found!**

This message will appear in the following scenarios:

- One or more of your Character Data Clusters have "Use Databases" set to "Both" but are missing a Character Segment Database and/or Character Color Database.
- One or more of your Character Data Clusters have "Use Databases" set to "Segments" but are missing a Character Segment Database.
- One or more of your Character Data Clusters have "Use Databases" set to "Colors" but are missing a Character Color Database.

This message can be resolved by inspecting all your Character Data Clusters. Check if you are missing a database or if the "Use Databases" field should be adjusted.

**At least 1 database appears to be empty!**

This message appears if one or more Character Data Clusters contain an empty database. This can be an empty Segment Database or an empty Color Database. Double-check your databases to ensure that they include at least one option to pick from the Character Creator window.

**At least 1 database contains a Segment or Color Group without a name!**

This message appears if any of the Segment Databases or Color Databases contain a segment or color group without a name. Double-check your databases and ensure all selectable segments and colors have a name.

**At least 1 invisible color was detected in a Color Group Database!**

If this message pops up, you may have recently changed one or more color groups. The message tells you that a transparent color (with alpha 0) was found in one of the Color Groups. This is an easy “mistake” to make because Unity gives color fields a default alpha of 0. Check if your recently added colors have an alpha value of 1. This problem may be patched in the future so the alpha is set to 1 by default upon adding a new color.

**ScriptableObject 'Character Creator Dependencies' is missing. This will cause unwanted behavior!**

A Scriptable Object called “Character Creator Dependencies” is needed for the Character Creator 2D to work properly. This Scriptable Object holds data for elements like icons and other GUI-related settings. To fix this error, navigate to the bottom of the Character Data Manager and to the field called “Character Creator Dependencies”. Click the white dot at the far right of that field; then select a Scriptable Object called “Character Creator Dependencies”. This is given to you by default upon downloading the project. If no such object exists, I recommend reimporting the project.

**Export settings currently include prefabs, but no template characters are assigned yet!**

This message pops up if you enable the “Export Character as Prefab” option in the Export Settings, but leave the list of Character Templates in the Character Data Manager empty.

If you don’t want to export a prefab for your characters, you may ignore this error message and set “Export Character as Prefab” to false. If you do want to export prefabs, you’ll have to add at least one Character Template for your prefabs to use as a blueprint. In Chapter 4, you can read more about why this is necessary.

**Export settings currently include prefabs, but at least one template character misses a prefab!**

This message shows up when you create a Template Character without adding a prefab to it, whilst having the “Export Character as Prefab” variable in the Export Settings set to true.

If you have no intention of exporting a prefab for your characters, feel free to ignore this error message and keep “Export Character as Prefab” set to false. If you do want to export prefabs, I recommend double-checking your Template Characters to find out which of those characters is causing the error to appear. If you add a prefab to all of your templates, the error should disappear.

**Animation exports are enabled, but the selected Character Template Prefab lacks an Animator Component with clips!**

If this message is shown, you have enabled the “Export Character Animations” option in the Export Settings while selecting a character template that doesn’t have an Animator Component. If you want to use a non-animated character as a template you can ignore this error message by disabling the “Export Character Animations” boolean in the Export Setting window. If you do want to export animations for your character, check your Character Template Prefab again to ensure it or one of its children contains an Animator Component with at least one animation clip.

## 6.4 Found an undetected error?

If you found a problem that the Debug Section didn’t detect, you may have encountered a bug or triggered unintended behavior that’s yet to be covered by the Debug Section. In this case, I recommend reporting the problem on the contact page. There you can select this asset and specify the exact nature of your problem. If possible, I also encourage you to visualize the problem with a screenshot.

**Report a Problem**

# 7. Setup From Scratch

Now we'll go over the entire setup process when starting a project with the Pixel Character Creator 2D from scratch. We won't be using any demo files and we won't be using the default character templates and files that come with the project. Instead, we'll add different pixel art characters, and color palettes and create every database and data cluster from the ground up. This section will be useful for people who want to add custom pixel art and use this tool in conjunction with player and NPC characters you've already created for your game project.

## 7.1 Quick recap

### Before we continue…

If you haven't read the previous chapters of the documentation yet, I highly recommend that you do so now. Knowing how this asset functions generally will go a long way in understanding what to do in this section. If you have finished the previous sections, I wish you lots of fun with the customization process!

### What you'll learn in this chapter

Now that you already have a general idea of how the Pixel Character Creator 2D works, you might wonder: "Well, how do I use this tool with my art to create my very own custom characters?". That's what you're going to learn in this chapter. After reading this part of the documentation, you are able to set up the Character Creator 2D entirely from scratch and customize it to your liking.

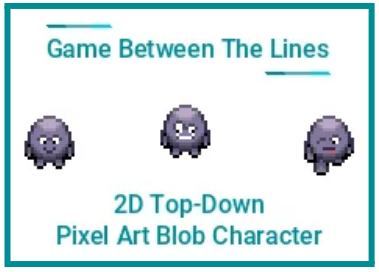
### The first step after importing the project

Before we start, navigate to *Assets / PixelCharacterCreator2D / OverheadCharacter*. In this folder, you should find a scriptable object called "PixelCharacterCreator2D". Since we are starting entirely from scratch, feel free to delete that one, and let's create a brand new Data Manager by right-clicking in the folder and then selecting *Create > Game Between The Lines > Character Creator 2D > Data Manager*. Now, let's add a character to the Unity Project.

## 7.2 A free asset to get you started

### In case you need a sprite sheet

If you want to try out this 'setup from scratch' method, but don't have a pixel art character to use, I am gifting you a free pixel art character sprite sheet asset. This asset is free for commercial and non-commercial use and can be downloaded on Itch. Click the button below to go to the informational page for this Asset. You'll also find the link to the Itch page.




**Get this sprite sheet for free**

### If you use your own artwork

If you have pixel art of your own that you want to use with the Character Creator, please keep in mind that for this tutorial I will be referencing the

sample sprite sheet that can be downloaded from Itch. In your case, settings like the size of your sprite sheets and individual sprites may differ from mine. I will mention this whenever necessary so you can adjust your settings to the correct values.

# 7.3 Prepare your character sprite sheets

## Setting up the character sprite parts

**STEP 1.**

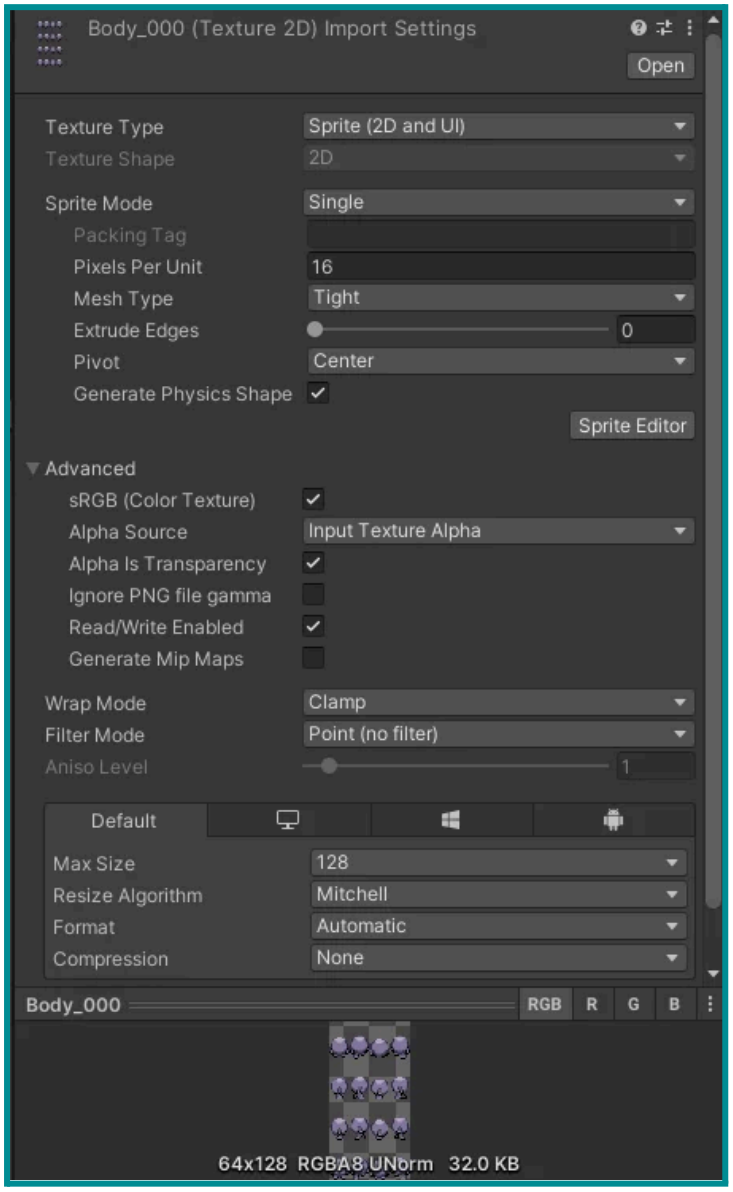
Let's import the character sprite sheets you created (or downloaded) into the Unity Project and place them in a folder that makes sense to you. You can see an example of how I stored the sprites by navigating to *Assets > PixelCharacterCreator2D > OverheadCharacter > Templates > TemplateSprites*.

**STEP 2.**

Pick either of the character sprite pieces and look at the Texture Import Settings in the inspector. We're going to edit one of these sprites and then copy our settings onto the other sprites. For the sprite parts we only need to adjust a few settings.

- Firstly, I recommend changing the pixels per unit to match the amount of pixels one unit should represent. You can tinker with this if you like, but I prefer to go with 16 pixels per unit for this character.
- Secondly, I prefer to set "Extrude Edges" to 0, but if you're consistent you can keep this the default value or change it to something else.
- Next up, go down until you see Read/Write Enabled. Make sure that this boolean is set to true. This is necessary for the Character Creator 2D to read the pixels.
- Since we're using Pixel Art, set the Filter Mode to "Point (no filter)".
- Lastly, set Compression to "None" and set the Max Size to be equal to or greater than the largest axis of the sprite sheet. In this case, the largest axis is the y-axis with 128 pixels, so I can select 128 to be the Max Size. These settings will help decrease the size of the sprite whilst maintaining quality.

You can check the image on the next page to see if your inspector window resembles mine. If you're satisfied, click "Apply".




**STEP 3.**

Now that we've set an example for what our Texture Import Settings for the Sprite pieces should look like, apply these settings to the rest of the sprites we just imported into Unity. We can do this with the help of a Texture Importer Preset. Navigate to the top right on the inspector window, with the Texture Import Settings still opened. Click the  icon and a pop-up window should open. Press the button in the bottom left corner of the window that says "Save current to…". Save it with a descriptive name so you know what this Texture Importer Preset is meant to represent.

**STEP 4.**

Now open your other Character Sprite Pieces in the inspector and apply the Texture Import Settings we just saved by clicking the  icon and selecting the Texture Imported Preset you just saved. Don't forget to click "Apply" afterward. All of the Character Sprite Pieces should now have the same settings applied.

Alternatively, you could select all of the Character Sprite Pieces at once and edit them all at the same time. However, the reason why I'm doing it this way is because I may add new Character Sprite Pieces later down the line. I would then have to go to the other Sprite Pieces to look at how those are set up, and then change the new sprite's Import Settings manually. With this method, you can easily skip all of that and just select the Textured Import Settings that you already set up earlier.

**STEP 5.**

Let's connect the Character Sprite Pieces with the Character Creator 2D. As you've read in Chapter 3.3, you can do so with a Character Segment Database. Do visit that chapter again if needed. Enact the following steps:

1. Create a new Segment Database by selecting *Create > Game Between The Lines > Character Creator 2D > Database > Character Segment Database.* Do this for the amount of Character Sprite Pieces you imported into the project. For the Blob Character I provided, I'll create 3 Segment Databases:
    - one for the body;
    - one for the eyes;
    - one for the mouth.
2. Add the Sprite Pieces to their respective database and give these "Segments" a recognizable name to make them stand out.

## Setting up the Character Template Sprite

**STEP 1.**

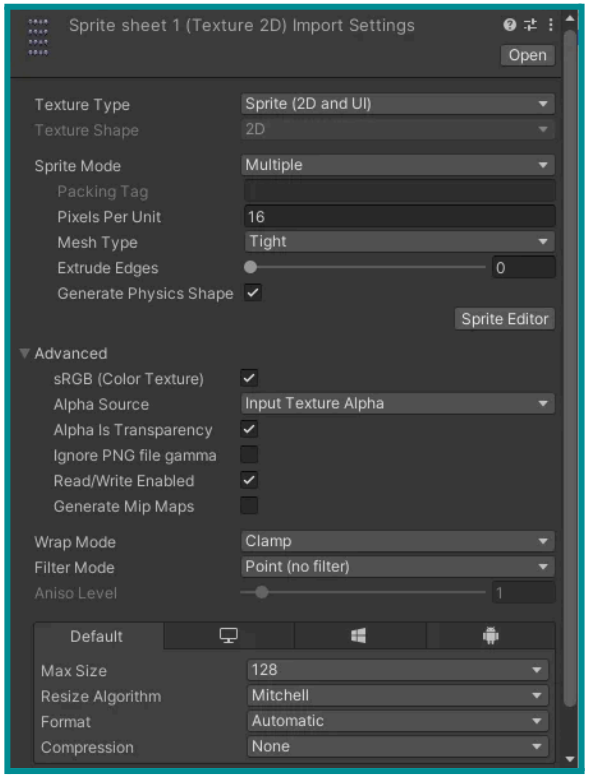
Import an example sprite sheet you created (or downloaded) into the Unity Project and place it in a folder that makes sense to you. I've put mine in *Assets > PixelCharacterCreator2D > OverheadCharacter > Templates > TemplateSprites*.

**STEP 2.**

With the example sprite sheet selected, look at the Texture Import Settings in the inspector. We need to adjust the following settings:

- This time, we'll set the Sprite Mode to "Multiple". We're going to slice this sprite sheet into different sprites.
- Change the pixels per unit to match the amount of pixels you set earlier. I'll go for 16 pixels per unit again.
- Make sure the "Extrude Edges" field matches the value you set before. If you set it at 0, like I did, set that value here as well.
- Set Read/Write Enabled to true again.
- Set Compression to "None" and set the Max Size to the same value as the other sprite sheet you set up earlier.

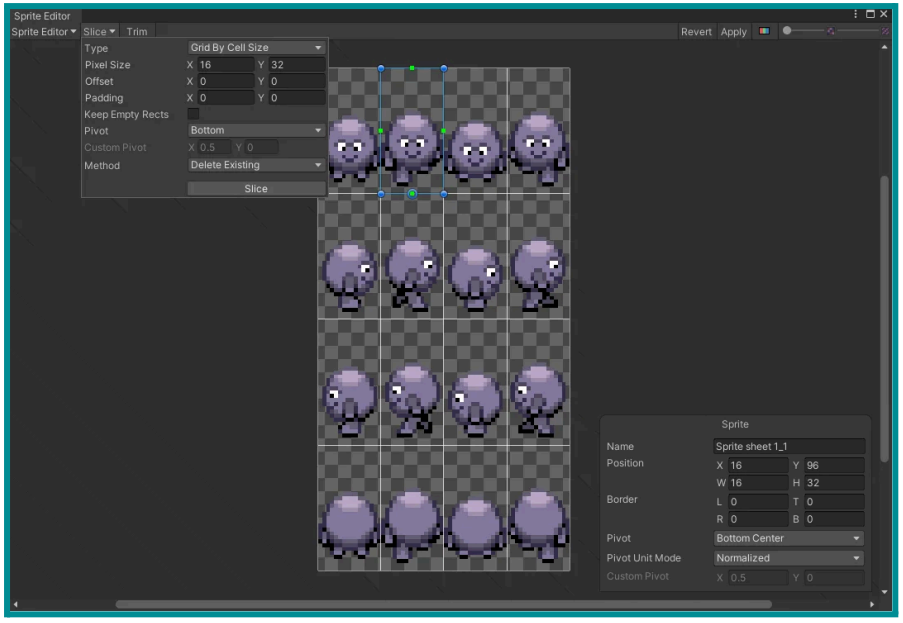
Check the screenshot below to see if it matches your inspector window. Press "Apply" as soon as you're done.



**STEP 3.**

Now we'll have to slice the sprite sheet. Click the "Sprite Editor" button and slice the sprite sheet into equal parts. There are multiple ways to achieve this, but my preferred method is the following:

- Click "Slice" in the top left corner and select "Grid By Cell Size".
- Set the Pixel Size to the size of the individual sprites for your character. For the example character I provided earlier, the width is 16 and the height is 32.
- I prefer setting the Pivot to "Bottom", but feel free to keep this as the default or change it however you like.
- You can now click the "Slice" Button and click "Apply" in the top right corner.




**STEP 4.**

If you aim to upload other Example Character Sprite Sheets I recommend creating a Texture Importer Preset for this sprite sheet as well. Follow the same steps as before, clicking the  icon and saving these settings as a new Texture Importer Preset.

## Setting the Sprite Sheet Export Settings

Open the Character Creator 2D and navigate to the Export Settings window in the left menu. Here we will set certain settings to correspond well with our character's Texture Import Settings.

If you need a refresher about the Sprite Sheet Export Settings and what they do, you can read a more detailed explanation of all the values in Chapter 1.4. Without further ado, let's set all of the settings to match our character:

**Sprite Layout In Sheet**

- Set the X-value of the Sprite Layout in Sheet (Vector2) to the number of sprites your character's sprite sheet contains in a horizontal line (row). For the blob character I provided, this amount should be 4, since there are four sprites per row.
- Set the Y-value of the Sprite Layout in Sheet (Vector2) to the number of sprites your character's sprite sheet has in a vertical line (column). For the blob character, this amount should also be 4, since there are four sprites per row.

**Export Sprite Sheet Size**

- Set the X-value of the Export Sprite Sheet Size (Vector2) to the width of your sprite sheet. For the blob character I provided, this amount should be 64. This is because there is no offset or padding between the sprites and there are four sprites per row, with each sprite having a width of 16.
- Set the Y-value of the Export Sprite Sheet Size (Vector2) to the height of your sprite sheet. For the blob character I provided, this amount should be 64 as well. This is because there is no offset or padding between the sprites and there are four sprites per column, with each sprite having a height of 16.

**Use Power of Two (POT) sizes**

- Whether you want to enable this is entirely up to you, but if you want to ensure that sprite sheet outputs are optimized for reduced file sizes, I recommend setting this boolean to true.

**Default Sprite Preview**

- Set the Default Sprite Preview to the front sprite of the example character you created in the previous subsection. This is not a requirement, but this will ensure a better user experience.

**Chosen Sprites For Multi-sheet Preview**

- This option affects the pose of the character that will be shown during the creation process. I often choose the idle front sprite of the character but you can change this to your liking if you prefer viewing your character from different angles or different poses.

**Preview Sprite Pivot**

- To prevent unintended behavior I recommend setting this Vector2 to match the pivots you set for the example character in the previous subsection. I will set this value to 0.5 on the X-axis and 0 on the Y-axis because I previously set the pivot of the Blob Character to “Bottom Center” in the Sprite Editor window.

**Sprite Sheet Offset**

- If your individual character sprites aren’t aligned to the top-left of your character’s sprite sheet, you should set the offset on the X-axis and Y-axis here. Since the Blob Character doesn’t use any offset within the sprite sheet, I will leave both the X and Y-values as 0.

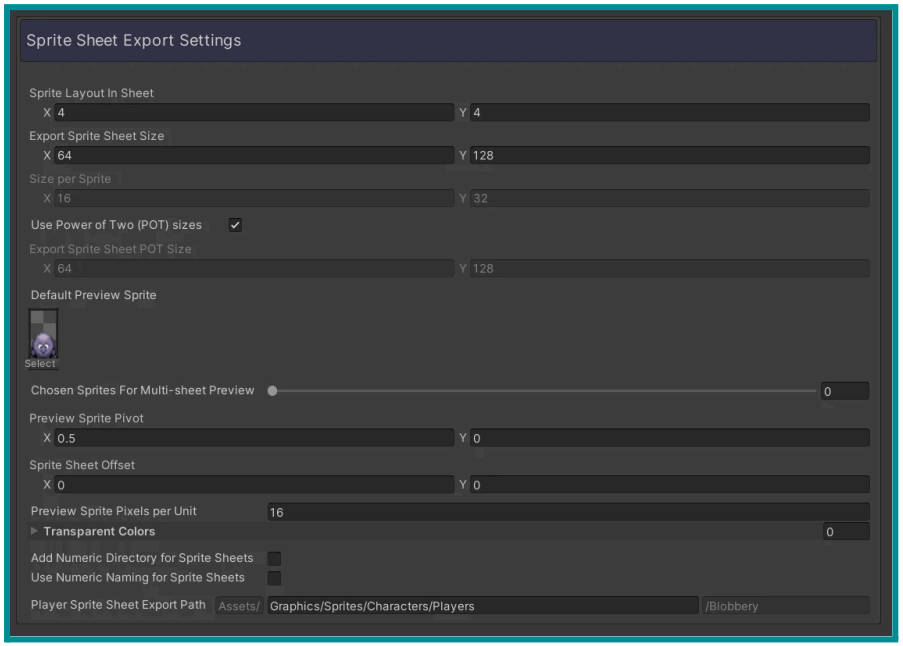
**Preview Sprite Pixels per unit**

- Set this value to the same amount of Pixels Per Unit you set for your example character’s Texture Import Settings in the previous subchapter.

**Transparent Colors**

- If your character uses transparent colors in any of its Character Sprite Pieces, those colors should be added here. This will ensure those colors are handled properly with a consistent opacity. Since the Blob Character doesn’t contain transparent colors, I will leave this list empty.

Your Sprite Sheet Export Settings should now be good to go. I created a screenshot of my settings for you to see on the next page, in case you’d like to compare your window to mine.

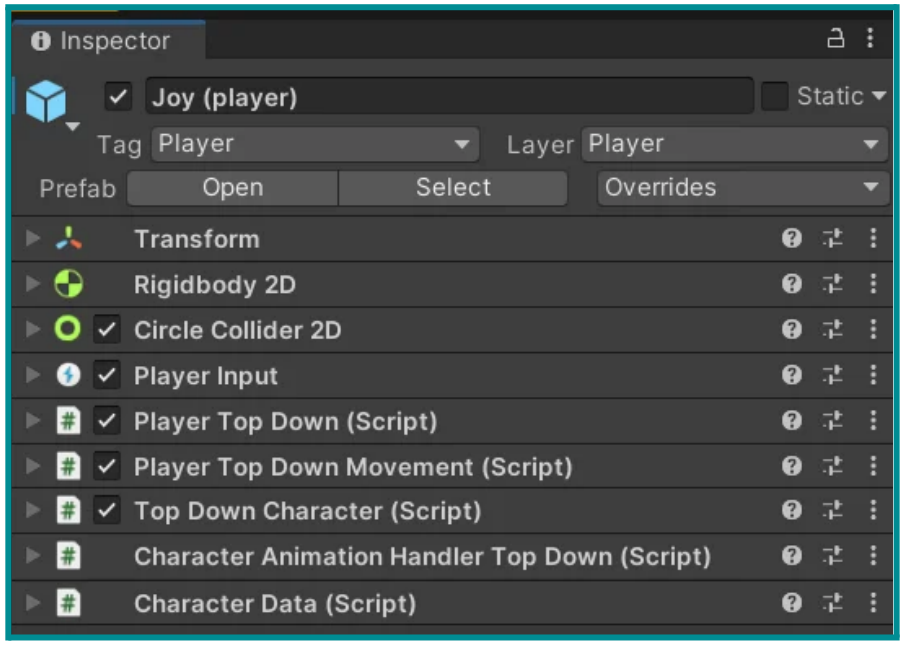



# 7.4 Prepare a Template Character

## Create a Character Prefab

Set up your character as you normally would and create a prefab for it when you're done. Add anything to this character that you would want to carry over to other characters. If you're setting up a Player Prefab, add components to it that other players would require, such as a script that handles physics or input.

If you're creating a Prefab meant for NPC characters, use components that an NPC would need to function, such as a Box Collider 2D and a script for triggering dialogue. All components and scripts you add to the Character Template are carried over.



# Character Data Script

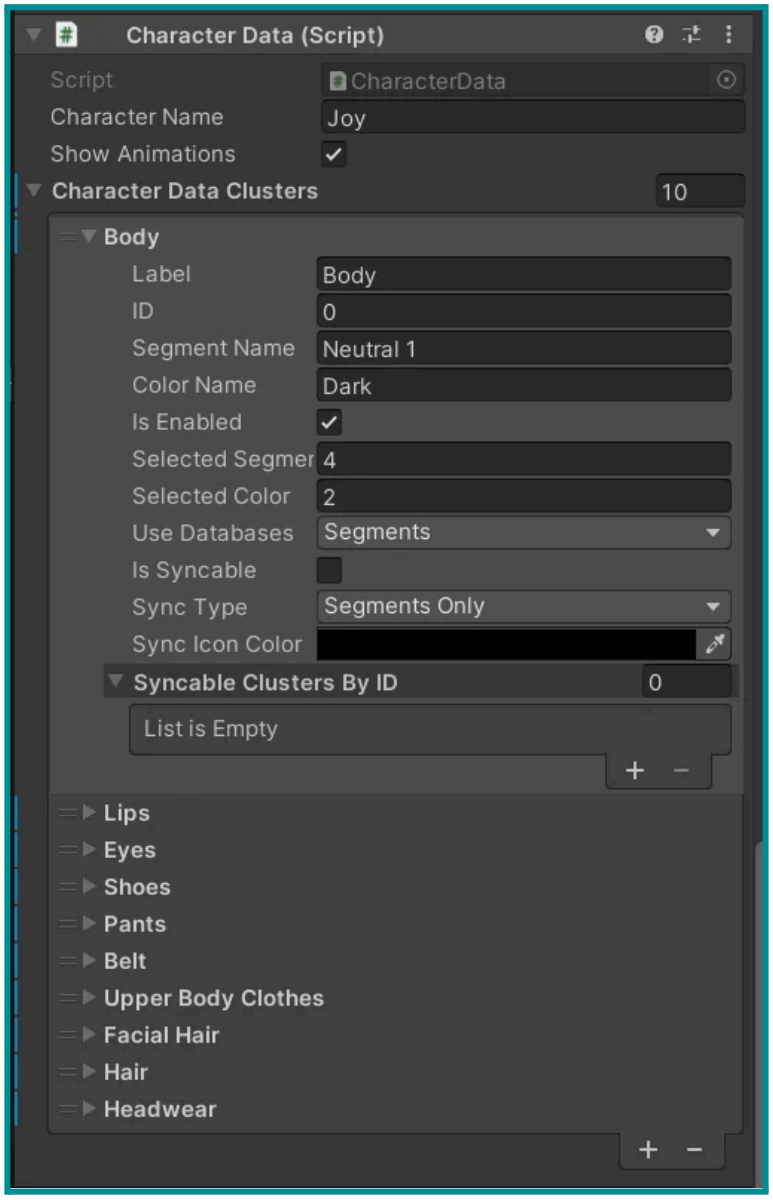
**Usage**

The Character Data Script holds any and all significant data regarding the Character you created with the Pixel Character Creator 2D. The purpose of this script is to hold this data in case it is needed for certain in-game logic. Here are a few examples you could use this script for:

- check if the character is wearing a specific type of clothing by retrieving the data from that data cluster;
- retrieve the name of the character;
- turn character animations on or off when needed.

**Setup**

The Character Data Script can be added manually as a component in the inspector. I recommend adding this script to the Prefab itself and not to any of its children. If you forget to add this script to the Template Character Prefab, the Character Creator 2D will add the script for you automatically. This means setting this script up manually is not required.



# Sprite Renderer and Animator Components

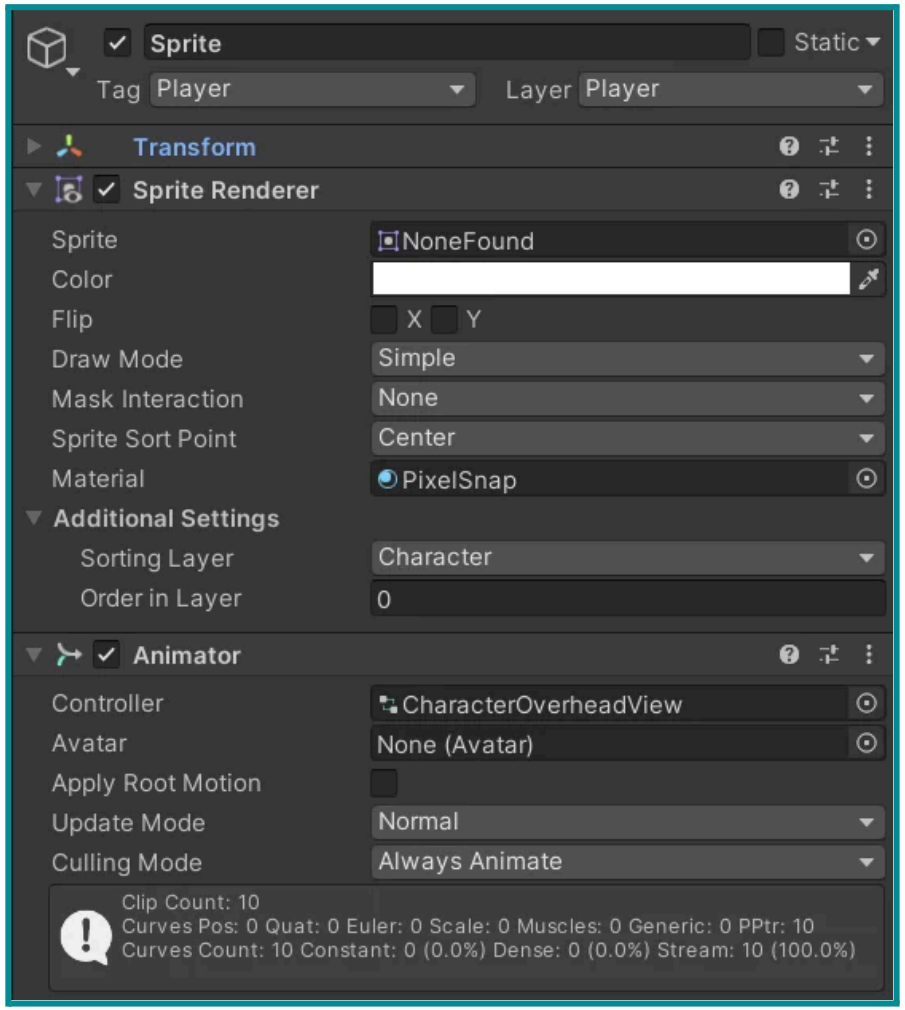
**Sprite Renderer Component**

Add a Sprite Renderer to the Character Prefab or one of its children. The Sprite Renderer requires a sprite to be added. It doesn't matter what sprite this is, as this sprite will be overwritten anyway. Just think of the sprite you add here as a placeholder. This sprite may only show up if an error occurs during the creation process of the character.

Since we're using Pixel Art, I also recommend adding a new material to the Sprite Renderer with the Shader set to Sprites/Default and "Pixel snap" enabled. In the demo, one such material is also included and added to the characters by default.

**Animator Component**

Add an Animator Component to the Character Prefab or either of its children. Make sure that the Animator Component is added to the same GameObject as the Sprite Renderer. The Animator Component will also need an attached Animator Controller with Animation Clips. If any of these requirements are forgotten, the Character Creator 2D will throw an Error message to remind you that these steps cannot be skipped.



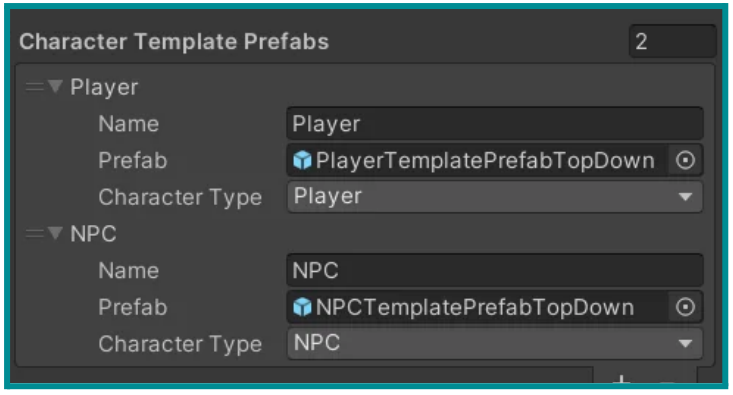
## Add the template character to the Data Manager Scriptable Object

Firstly, go to the Character Data Manager. Go to the section called “Character Template Prefabs” and add a new Character Template Prefab by clicking the  icon. While it’s possible to generate multiple templates, I recommend exercising a bit of restraint and only adding templates that are essential for creating your characters.

Subsequently, assign a descriptive name to the newly added template so its purpose is easily understandable. If you don’t name the template, this will prompt a warning message. However, you can proceed with the creation process despite this.

Next, drag the Template Character Prefab you’ve recently created into the designated “Prefab” slot. If this step is forgotten, this will result in errors throughout the creation process.

Lastly, decide whether the character is intended to function as a player or an NPC. This helps in maintaining organizational coherence with your templates.



# 7.5 Determining the Data Clusters

## Plan ahead

This may sound like a no-brainer, but I assure you it's beneficial to know how many Data Clusters you will need in advance. In the case of the sample character I provided, I know that I'll need three Data Clusters:

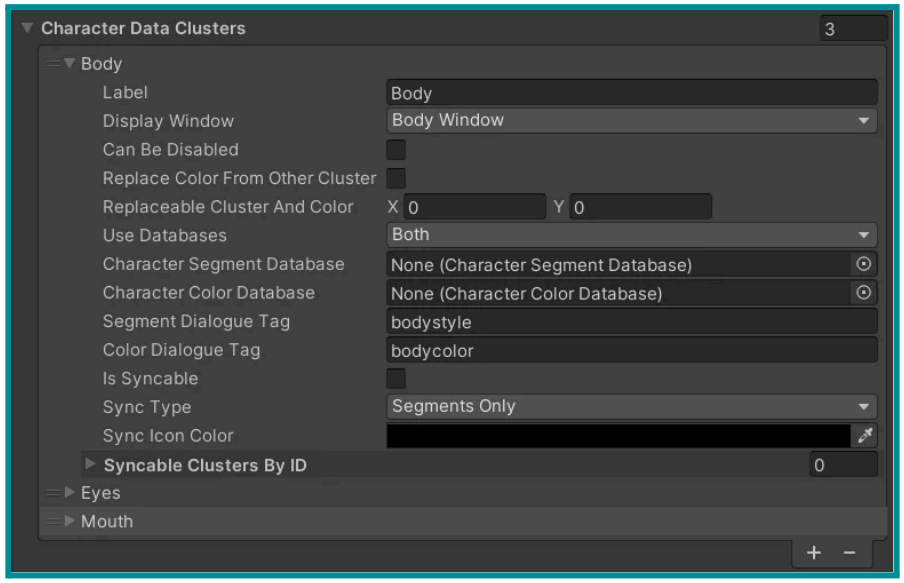
1. a Body Cluster;
2. an Eyes Cluster;
3. a Mouth Cluster.

If you haven't created character sprite pieces yet, consider how much control you need over your character's design and how much customizability you want for that character. You can make this as simple or as complex as you'd like, so take a moment to plan things out.

## Add clusters for every customizable part

Access the Character Data Manager Scriptable Object you created a while back and view it in the inspector window. Here, you'll find an overview of all the Character Data Clusters which should be at 0. Let's add some!

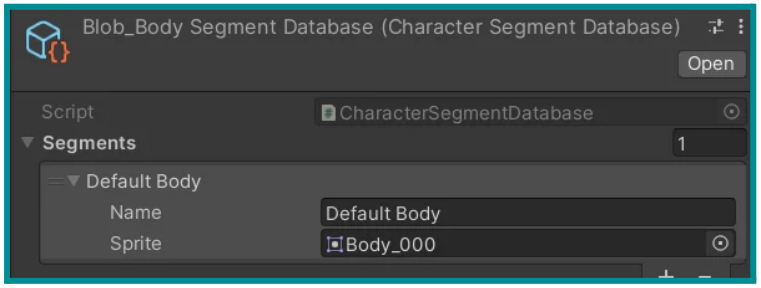
Select the + icon to create 3 new Character Data Clusters. Let's give these Data Clusters a name while we're at it. I'll be naming the clusters "Body", "Eyes", and "Mouth" respectively. Feel free to tinker with the other settings to achieve your desired outcome. I will leave all three as they are for now and keep the default values. Now, we'll need to add a few databases to gain control over the character segments and colors.



# 7.6 Setting up the Databases

## Segment Databases

First, determine the location to store your new Segment Database. Then, right-click and choose: *Create > Database > Character > Segment Database*. Create as many Segment Databases as there are Character Data Clusters. In my case, I'll create three and give them descriptive names to match their purpose in the Character Data Manager. After creating the databases, add segments to the lists and drag the Character Sprite Piece sheets into their corresponding Databases. In the case of the Blob Character, I will show you how I've set these Character Segment Databases up.




*Segment database for the body*


*Segment database for the eyes*


*Segment database for the mouth*

Finally, add the Segment Databases into their corresponding Data Clusters inside of the Character Data Manager.

## Color Databases

Choose a folder to store your new Color Databases. Navigate to the folder of your choosing and right-click and choose: *Create> Database> Character > Color Database*. Create as many Color Databases as there are Segment Databases. In my case, that'll be three. Give the databases names that are descriptive enough for you to recognize them with relative ease.

**Add a default Color Group**

Now, let's populate the newly created Scriptable Objects with the default Color Groups. Give them a name that describes the default color schemes well enough. Next, We need to add the hex codes for every color used in the sprite sheets with the Character Sprite Pieces. I recommend using Aseprite or a similar tool to extract hex codes from the image files.

If you're using the Blob Character, like me, I've provided the hex codes for the body, eyes, and mouth on the next page for your convenience. Simply copy and paste these hex codes into the color fields and you're done with the default colors. Don't forget to set the alpha to 1 for each color.

**Hex codes for the Blob Character Body Sprite Pieces**

| #BBA7C6 | #A28FAB | #83789E | #666077 | #4D475C | #352D4B | #000000 |
|---|---|---|---|---|---|---|

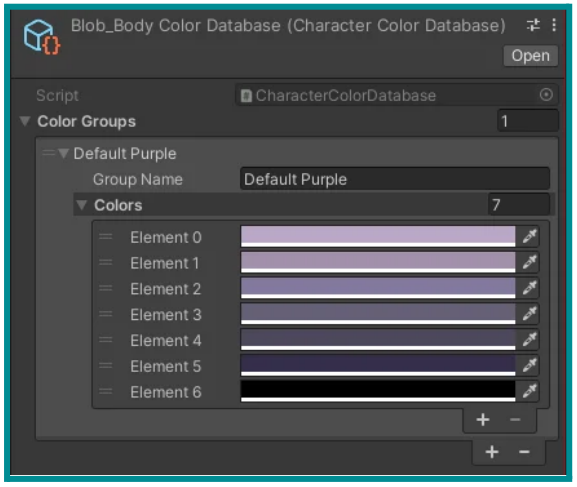
**Hex codes for the Blob Character Eyes Sprite Pieces**

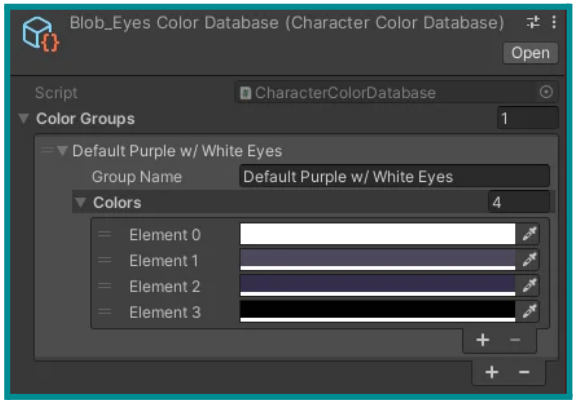
| #FFFFFF | #4D475C | #352D4B | #000000 |
|---|---|---|---|

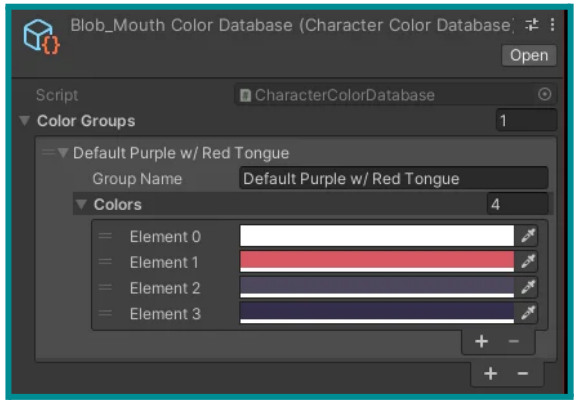
**Hex codes for the Blob Character Mouth Sprite Pieces**

| #FFFFFF | #D95763 | #4D475C | #352D4B |
|---|---|---|---|

If you've completed this part, this might be a good time to take a quick breather and compare our results. You can view the screenshots below as a reference to see if you're on the right track.



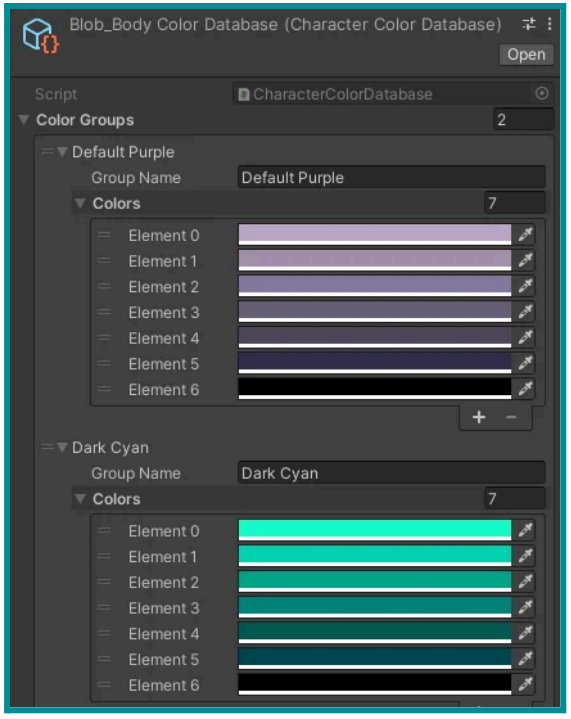



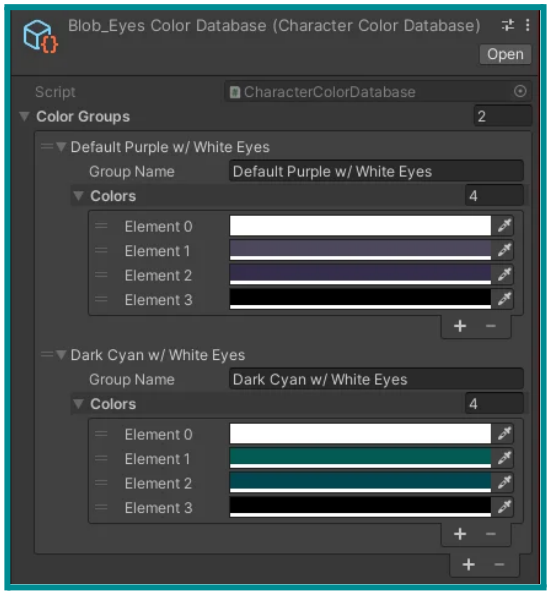



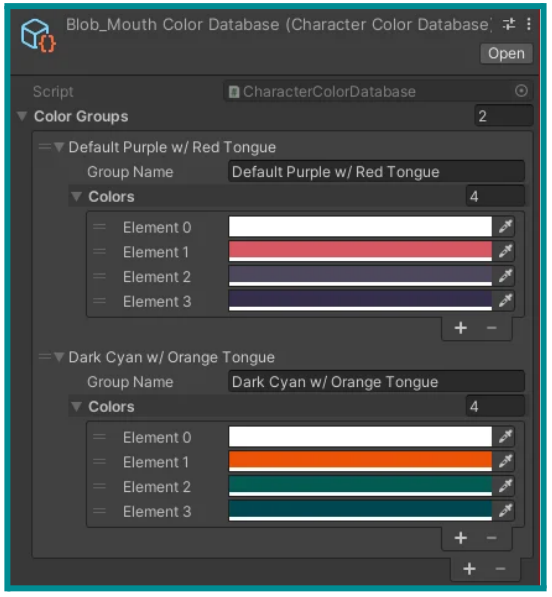
**Add more Color Groups**

Now that we're done with the default colors, let's add some other color groups so that we can actually change the color of our character. I will add one Color Group per Color Database, just to get the point across, but feel free to add as many as you like. If you're having trouble coming up with good color palettes, numerous online tools can help you with that. I usually use either of the following tools for this purpose:

- iColorpalette
- ColorHexa
- Adobe Color



**Extra hex codes for the Blob Character Body Sprite Pieces**

| | | | | | | |
|---|---|---|---|---|---|---|
| #14FFC9 | #00D6B2 | #00A689 | #00847D | #035B55 | #004851 | #000000 |

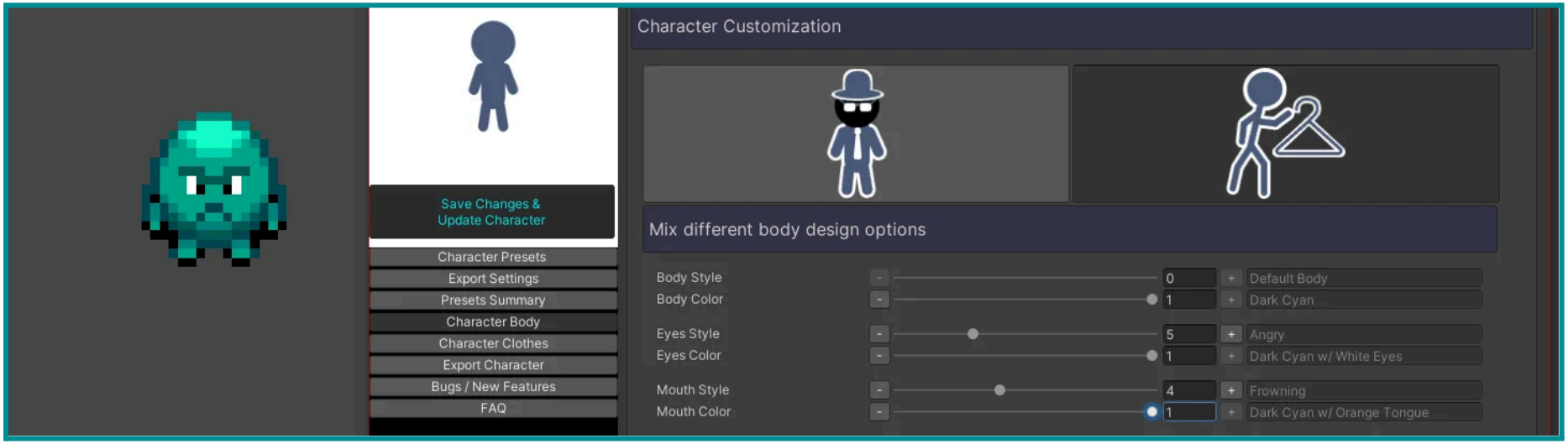
**Extra hex codes for the Blob Character Eyes Sprite Pieces**

| | | | |
|---|---|---|---|
| #FFFFFF | #035B55 | #004851 | #000000 |

**Extra hex codes for the Blob Character Mouth Sprite Pieces**

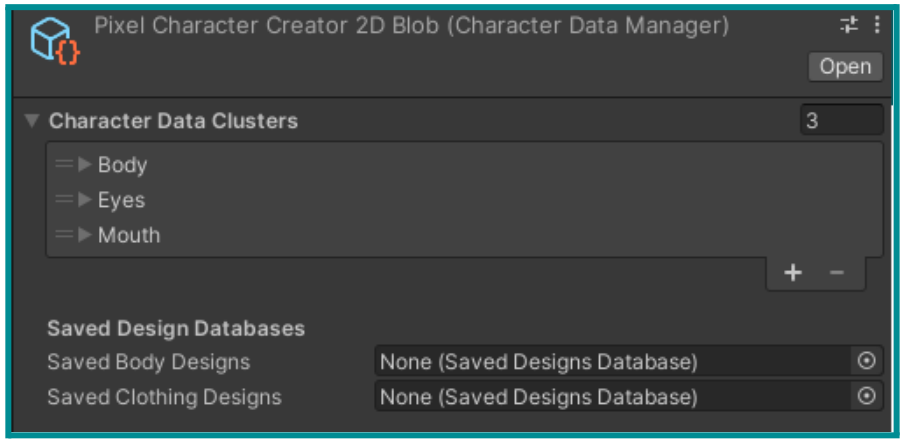
| | | | |
|---|---|---|---|
| #FFFFFF | #EC5206 | #035B55 | #004851 |

After adding your new Color Groups, you should be able to open up the Character Creator 2D and tinker with the character design. View my result below.



## Saved Designs Databases

If you opened the Character Creator 2D in the previous subsection to test your segments and colors, I assume you saw the same warning message as I did: *"Warning: Scriptable Object 'Saved Body Designs' is missing. This prevents saving and quick-selecting your designs from the Body Window!"*




If you would like to read the entire explanation of this warning message, you can do so in Chapter 6.2. The gist of it is that we need a Database to store Saved Designs of our character. While solving this warning message and adding Saved Design Databases is optional, I recommend trying this feature out at least once in case you may want to use it in the future.

Let's add a Saved Designs Database for our character. If you need to freshen up on this feature, you can read the full explanation of Saved Design Databases in Chapter 3.7. You can create a Saved Designs Database by selecting *Create > Game Between The Lines > Character Creator 2D > Database > Saved Designs Database*. Again, be sure to choose a folder and name for this database that makes sense to you.

Now add the Saved Designs Database you just created to the Character Data Manager by dragging it into the correct empty slot. Since we've only added segments and colors for the Body Window, we only need to add a Saved Designs Database to the Saved Body Designs slot. The other slot can remain empty since the Clothing Window is currently unused. After adding the Saved Body Designs Database, the warning message we previously received should now have disappeared.

# 7.7 Preparing Character Animations

## Template Animations & Animator Controller

If you've followed along with the steps in Chapter 7.4, you should already have a template character with an Animator Controller Component ready for use. Now's the time to animate your character if you haven't done so already. Be sure to save the Animation Clips and Animator Controller of your Template Character in a place that makes sense to you.

Give the Template Character as many animations as you desire. If you're using the Blob Character I provided, you should be able to add an Idle Animation and Walking Animation in four directions.

## Fetch the Animation Data from the Template Character

Open the Character Creator 2D and navigate to the Export Settings window in the left menu. In this window, we need to scroll down to the "Animation Export Settings" section. There, we need to click the button that says "Fetch data from Runtime Animator Controller". If you need a reminder of what this does, you can read more about it in Chapter 5.2.

After pressing the button, the Template Animation Data will have been extracted from the Template Character's Runtime Animator Controller. Go through the list of Template Character Animations to check if everything was carried over correctly. You can edit some of these values as well, but I'd rather recommend changing settings in your Template Character's Animation Clips themselves if needed.

# 7.8 Try it out!

Have you followed along with the entire instruction? If you did, let's try it out! Open the Character Creator 2D and try to create a Character of your design. I have created a few Blob Characters below to give you an idea of the output you can expect.

Are you encountering unresolved warnings or bugs? In that case, I recommend visiting Chapter 6 again and checking whether your problem is covered there. If it isn't, feel free to send me a message via my Email or the designated contact form.

# 8. Changelog

## Version 1.0.1

### Bug fixes

- If the user selected a Character Template and then deleted said template, the Character Creator Window would fail to load. The selected Character Template will now default to the last available character in the template list.
- If the user disabled the exporting of animations and then re-enabled this option, the Preset Summary window would still show that animation exports were disabled. This toggle function is now fixed.
- Template Characters had missing sprites in their animation clips. This caused errors if the user wanted to fetch Animator Template Data from the Character Templates. This is now fixed.
- If the user wanted to create a character without adding Character Text Tags to the global character data, the character export would fail. This is now resolved and completely optional.
- Sometimes character names would be forgotten upon exporting the character, leaving blank names that prevent animations from being exported. This is now resolved.

### Changes / Additions

- A new debug message is added to inform the user of missing sprites in Template Character animation clips.
- Small tweaks were made to comments and guiding texts in order to prevent contradictory instructions.
- The template characters that are provided with the download of this asset now have their default Walk Speed set to ‘Regular’ and their Run Speed set to ‘Very fast’.