So you want to create a new character for your game? Here's how to do that!

I chose Moguera as an example for this tutorial.

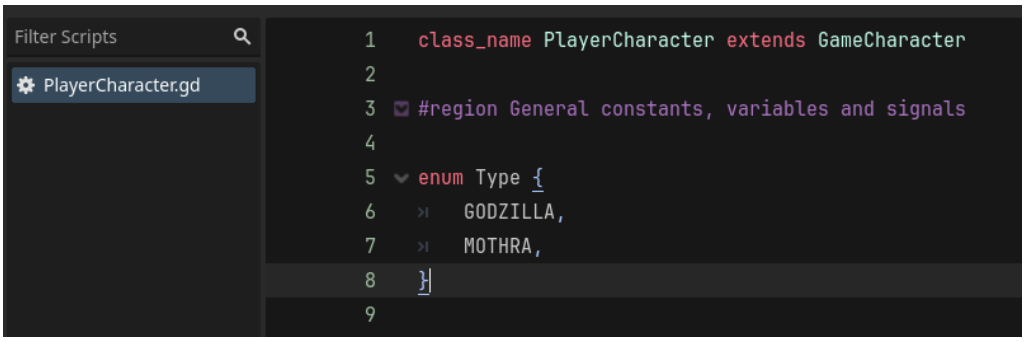
(Please note that there is a part 2 of this tutorial)

Contents

New character slot

First, let's add a slot for our new character. The character types are stored in the characters' script, "PlayerCharacter.gd", it's contained in "Scripts/Objects/Characters" folder.

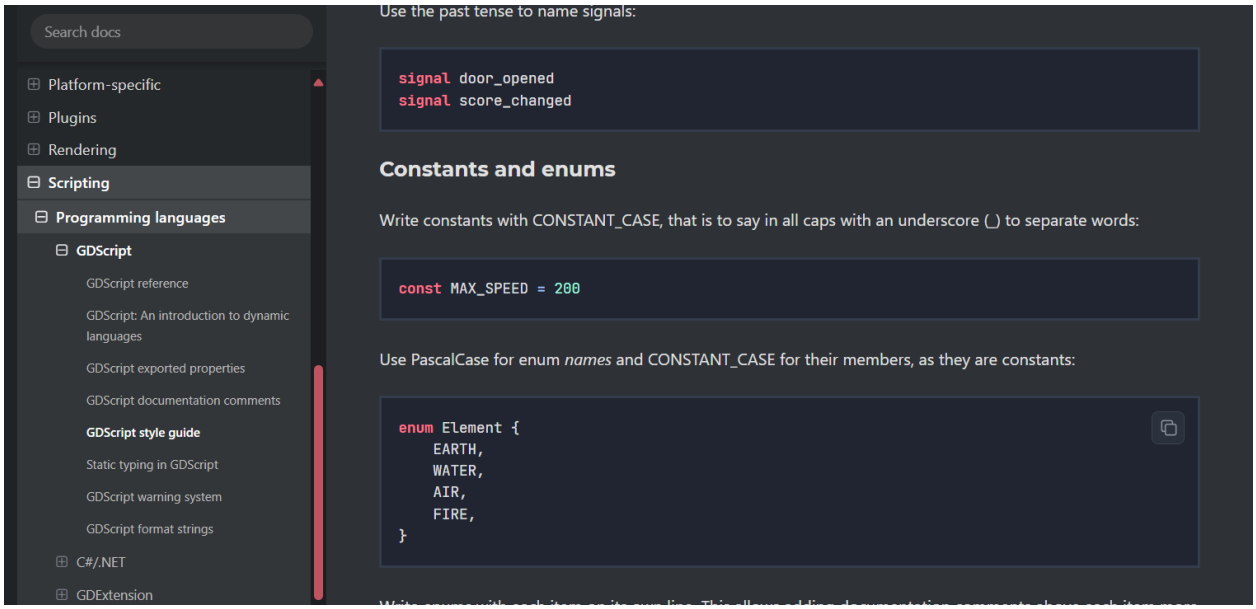
If you open the script in the Godot script editor and scroll to the very top you will immediately notice the enumeration that contains every character type:



Let's add a new character type! Create a new line after the last character ("MOTHRA," in this case) and put in similar text but for your character's name (if your character's name has multiple words, like "Mecha Godzilla", it's recommended to use underscores, like "MECHA_GODZILLA", don't use spaces or hyphens because Godot would be confused), in our case it's "MOGUERA,", like so:

```
4  #region General constants
5
6  enum Type {
7      GODZILLA,
8      MOTHRA,
9      MOGUERA,
10     }
11
```

You're probably wondering why we are using uppercase letters for character names, well, it's just a convention: if we declare something constant (in our case, the character type will be the same throughout the entire game, so it's constant, in other constants can be something like the value of pi, or how much damage should an attack do) we use uppercase letters and underscores ("_") if we want to separate several words. This convention is applicable to many languages, including GDScript (screenshot from Godot documentation):



And also Java, for example:

2. Basics

A constant is a variable whose value won't change after it's been defined.

Let's look at the basics for defining a constant:

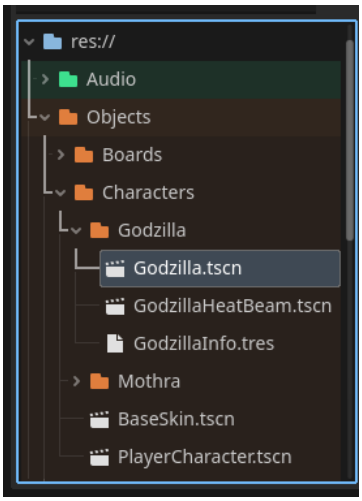
```
private static final int OUR_CONSTANT = 1;
```

Some of the patterns we'll look at will address the *public* or *private* access modifier decision. We make our constants *static* and *final* and give them an appropriate type, whether that's a Java primitive, a class, or an *enum*. **The name should be all capital letters with the words separated by underscores**, sometimes known as screaming snake case. Finally, we provide the value itself.

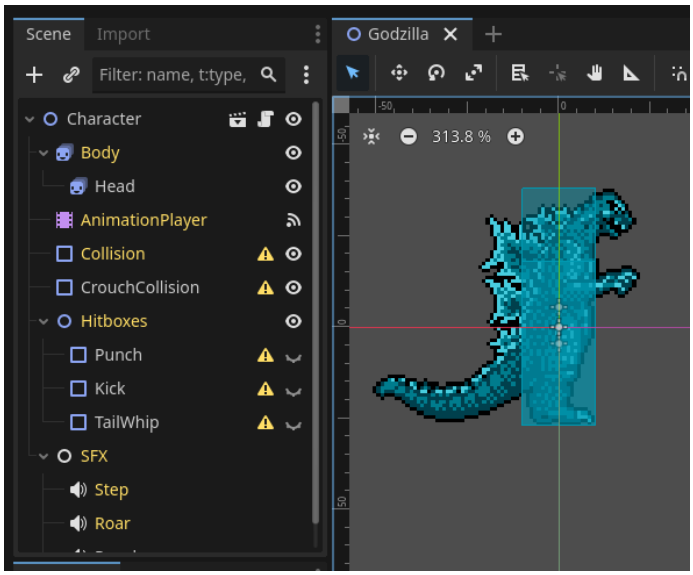
Creating new character skin

Alright, we made our new slot. What now? Let's create a new character skin for our character! How are we going to do that? Easy, but first let's find the existing skins for Godzilla and Mothra and see how they work.

The character skins are in "Objects/Characters" folder:



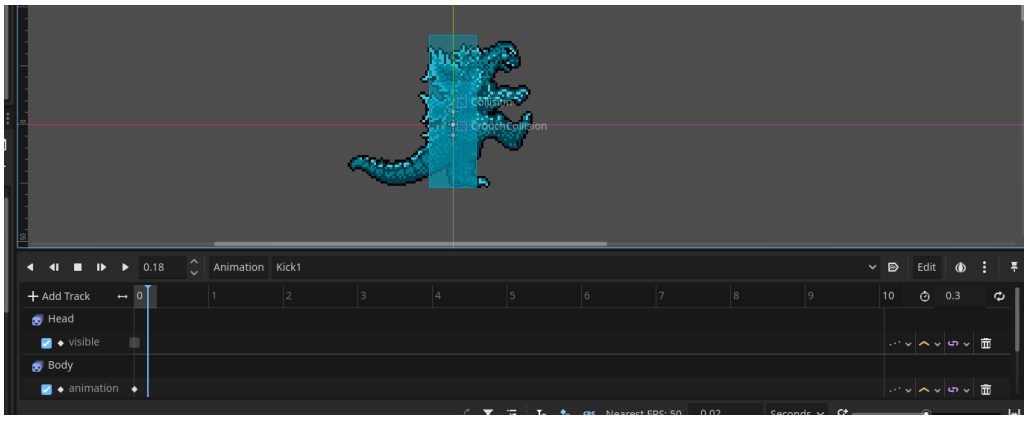
Now let's open "Godzilla.tscn"



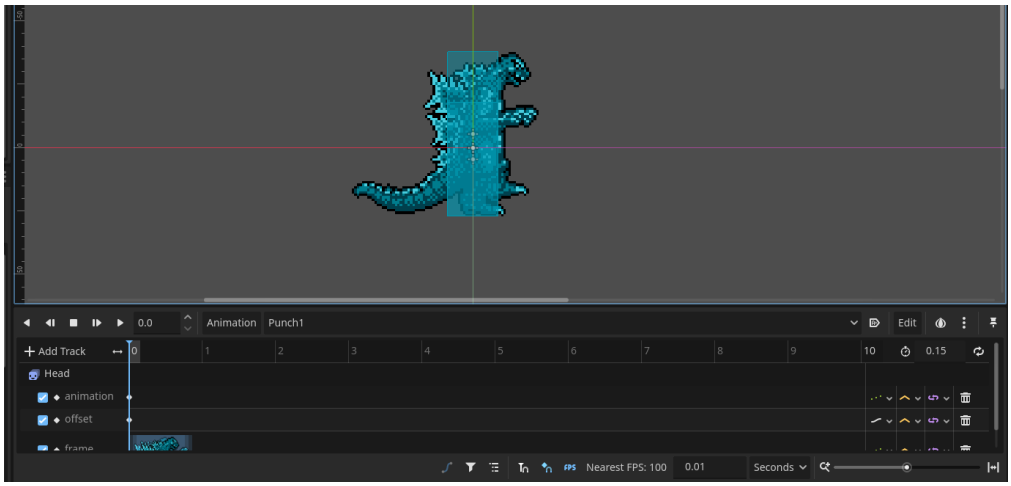
We can see that there's a root node called "Character", several sprite nodes, an AnimationPlayer node, 2 collision boxes for the character itself, 3 collision boxes for attacks and 3 AudioStreamPlayers for character-specific sound effects (2 of which, marked yellow, should be present in every character).

There are 2 sprite nodes because they need to be separate (think about attacking while walking or while standing, in that case the sprites would've contained a combination of every top part of Godzilla's sprites attacking and the bottom one walking if they weren't split, that sounds like a mess, so we separate them into 2

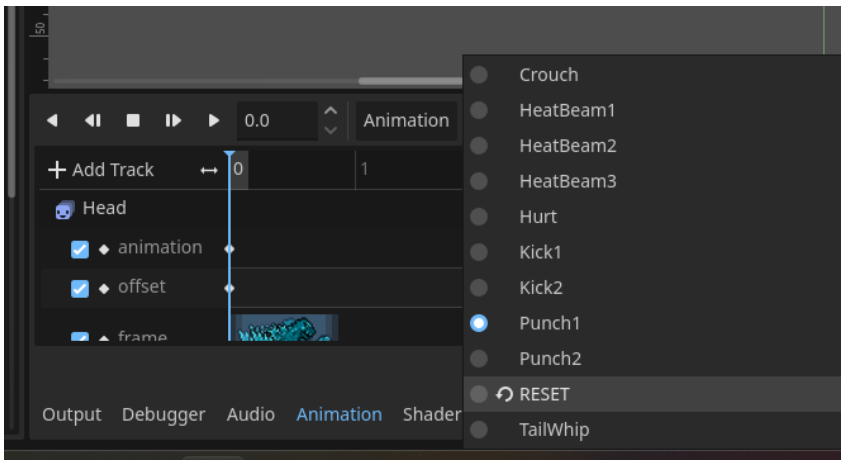
nodes). The animation player, as you may guess, is responsible for Godzilla's animations:



Sometimes if you keep playing with that by selecting various animations you may notice that the top part of the sprites becomes invisible or the animation doesn't change

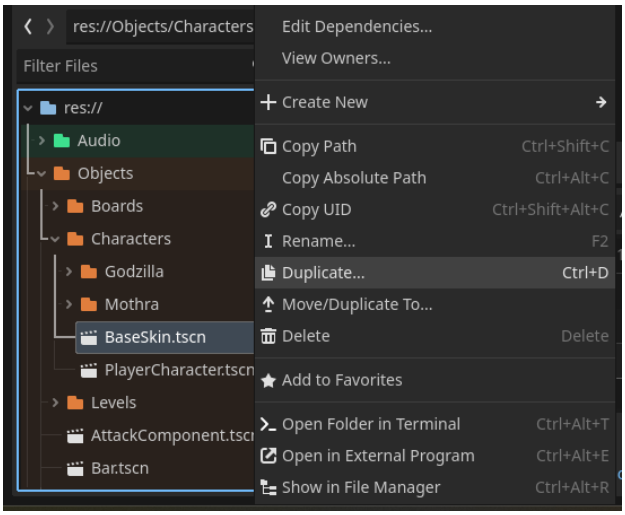


But don't worry, you can reset everything by selecting "RESET" animation:

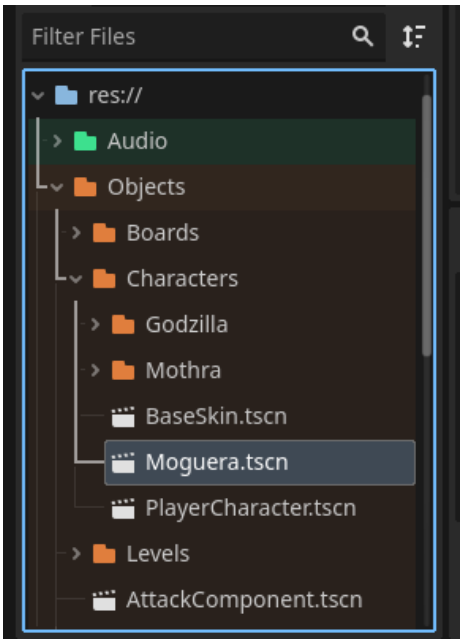


You can also notice that there's no walking animation. That's intentional, we change the walking frame using code, not with animations (if you're curious about how it works, you can check out the "move" function in "Scripts/Objects/Characters/Walk.gd").

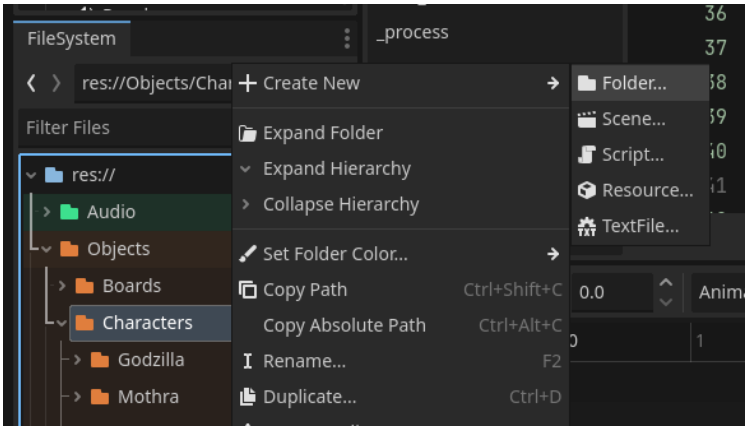
So, how do we actually create a new skin? Well, we should duplicate the base skin scene and start from there:



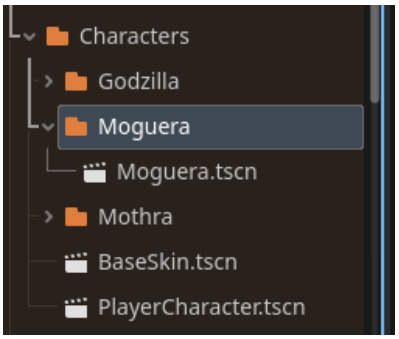
There we go:



And we should probably create a folder for our character just like what Godzilla and Mothra have. Right click on “Characters” folder and create a new folder:



Now drag and drop your new character’s skin onto this folder:



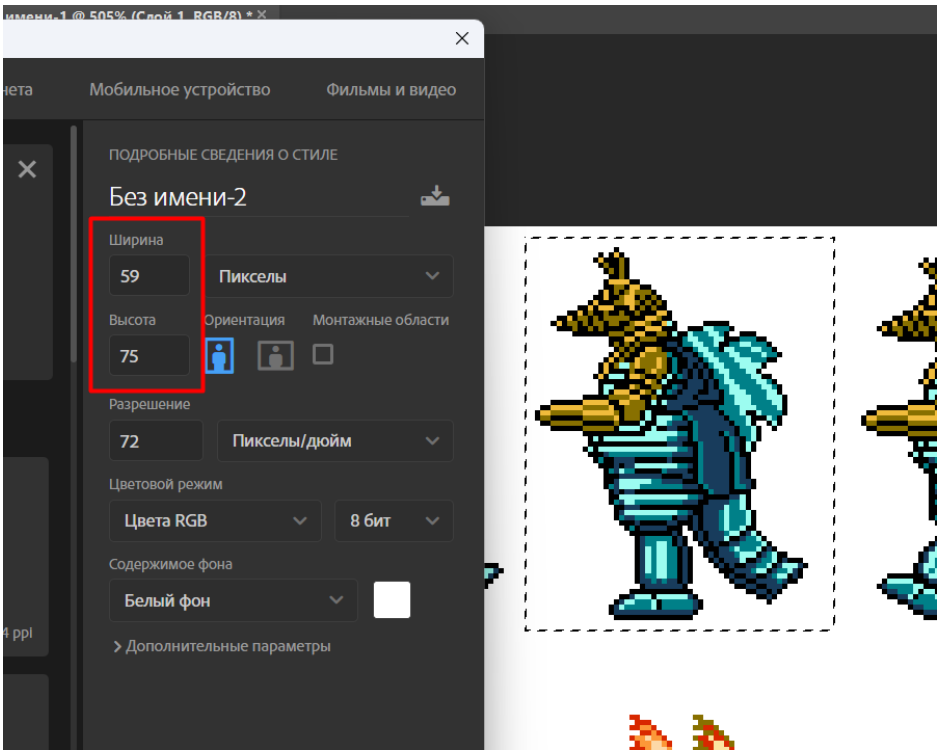
Great, now we can prepare the sprites for our character!

Preparing sprites

Now we should prepare the sprites for our character. I will be using Photoshop for that, and thanks to FreKay Planet for ripping the Moguera sprites from Godzilla: Monster of Monsters.

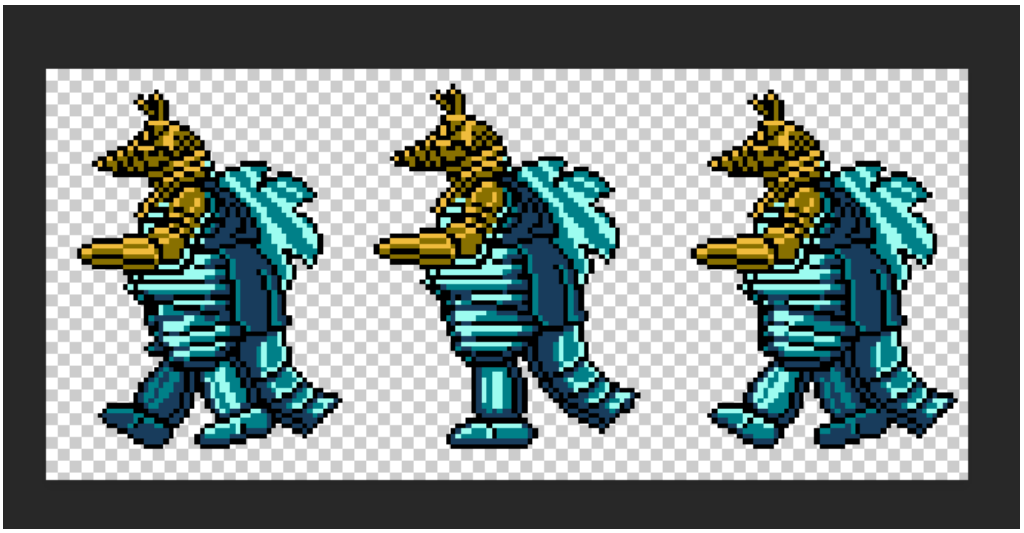
Since Moguera doesn't have any special animations with its top part of the sprites we can omit the step of preparing separate sprites for top and bottom parts of the sprites.

Now let's figure out the bounding box for the sprites. I roughly made a selection over a sprite and copied it.



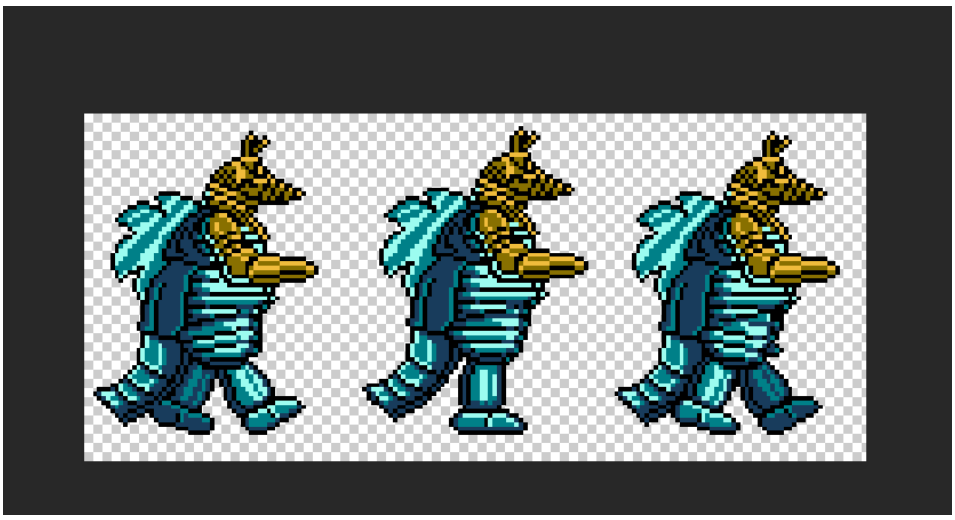
As we can see, the bounding box of 59 x 75 is fine, but I like pretty numbers so I will go with 60 x 80, but since there are 3 sprites we will make a new spritesheet with the size of 180 x 80 (180 = 60 * 3).

Now let’s copy our sprites and delete the white background



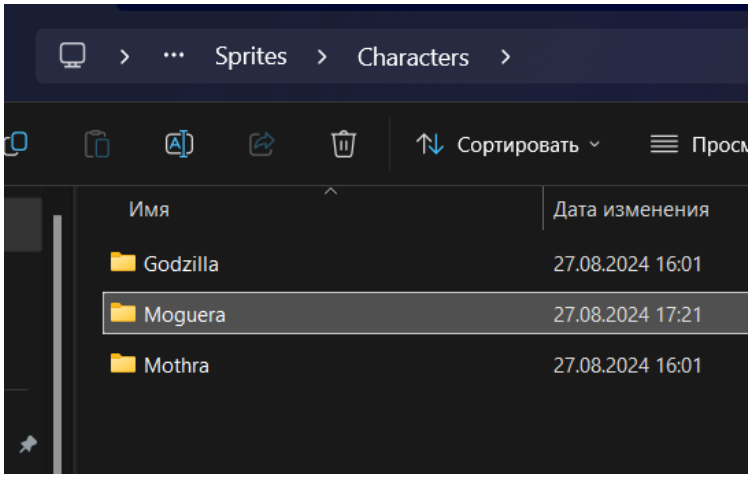
If we use the sprites in this state the animations will be a bit shaky (we haven’t aligned each sprite correctly in their respective bounding boxes yet, you will see what I mean later) but we can fix that later.

Now let’s mirror them since the player is facing right by default

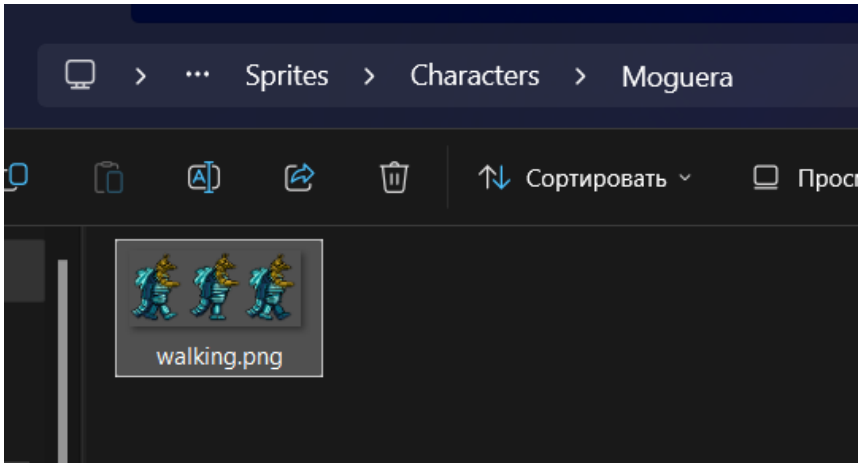


Great! Now let’s save it. The folder for character sprites is “Sprites/Character”, and you will notice that the sprites for each character are separated into folders. If you look into both folders, you will know why: it would be messy and hard to manage if they were in the same folder.

So now let’s create our new folder for our character:



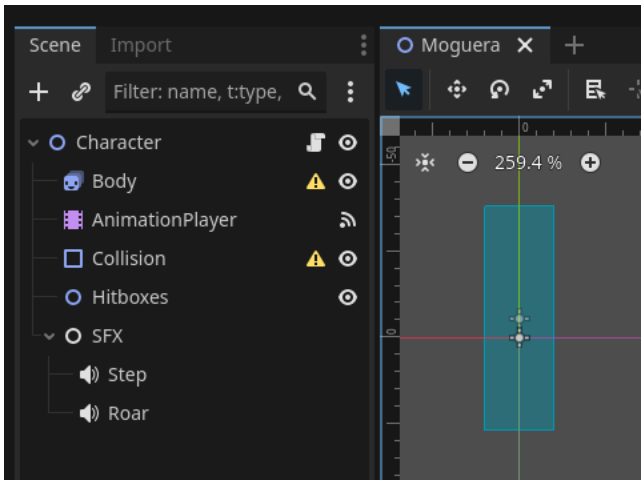
And now let’s save our sprites there:



Great, now we’re ready to import them into Godot!

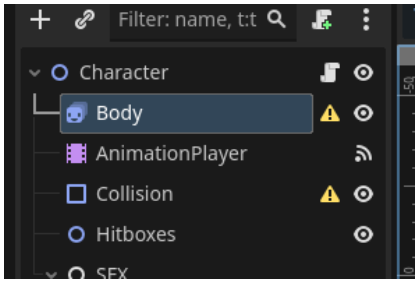
Importing sprites into Godot

Open the skin scene you made earlier.

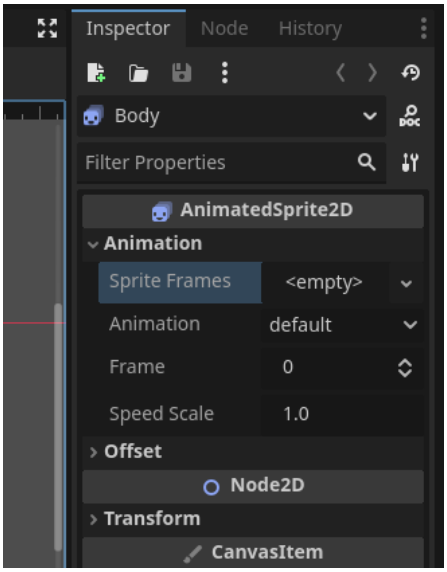


In the base skin there's only one sprite node, unlike what Godzilla has. That's because not every character should use 2 sprite nodes, so the base skin only has one. I will describe how to separate the character sprites into 2 nodes later in this tutorial.

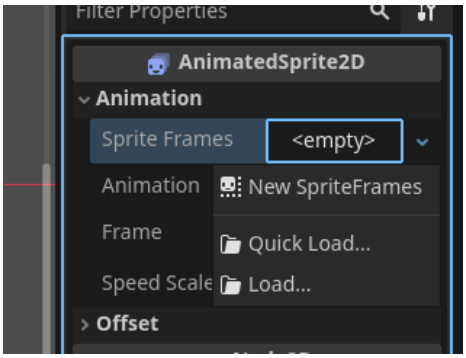
And since we duplicated the base skin scene there are no animations. Let's add one. Click on the "Body" node:



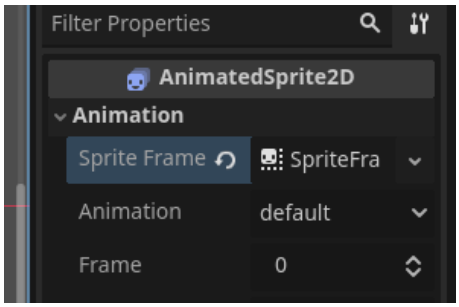
Go to its properties on the right side of the Godot editor and open the "Animation" section:



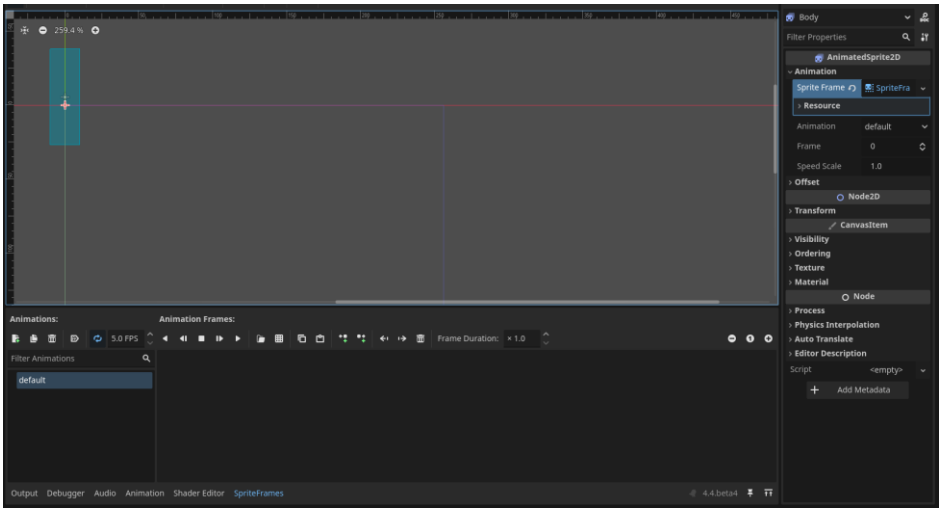
Click on “<empty>” next to the Sprite Frames property:



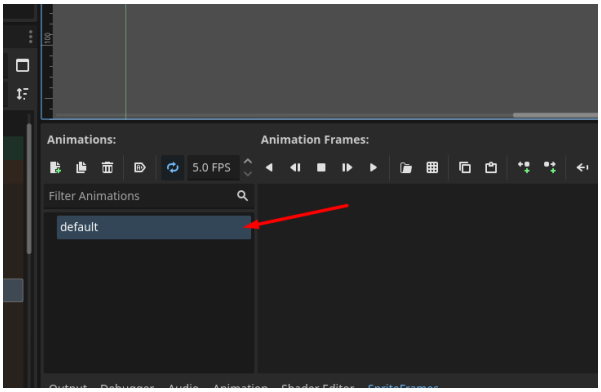
Now click on “New SpriteFrames”.



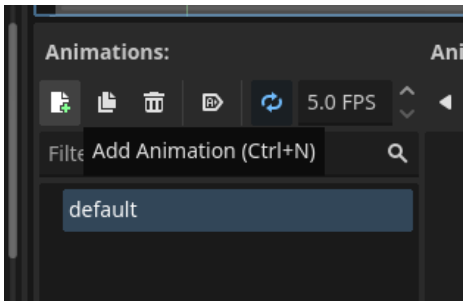
Click on the newly created SpriteFrames resource, a new menu will appear at the bottom of the editor:



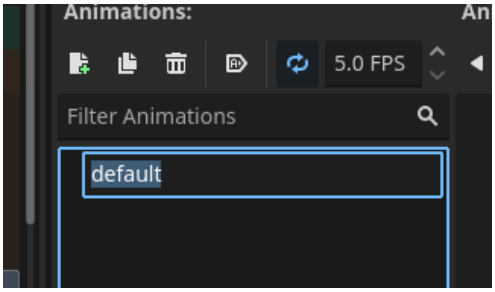
Now we can either rename the default animation



Or create a new one

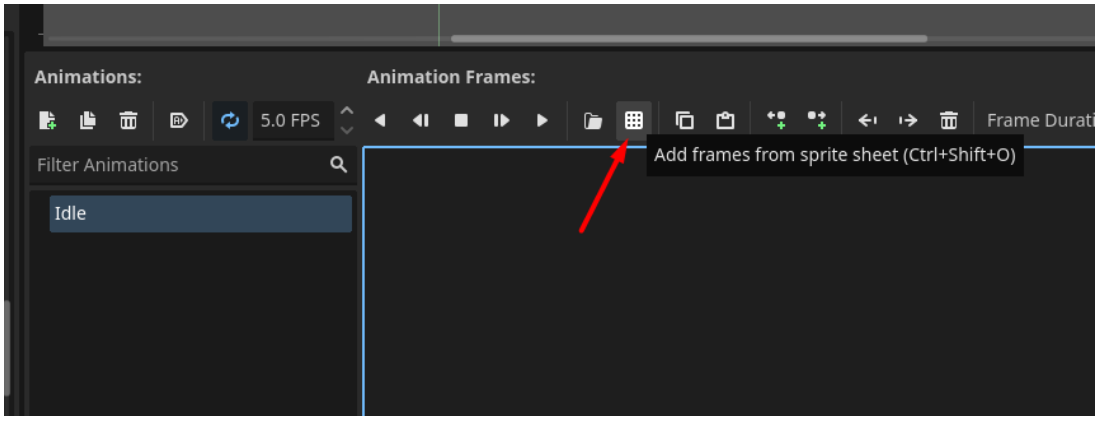


You can rename the existing default animation since it's already been created for you. Click on it while it's selected, and it should be ready for renaming:

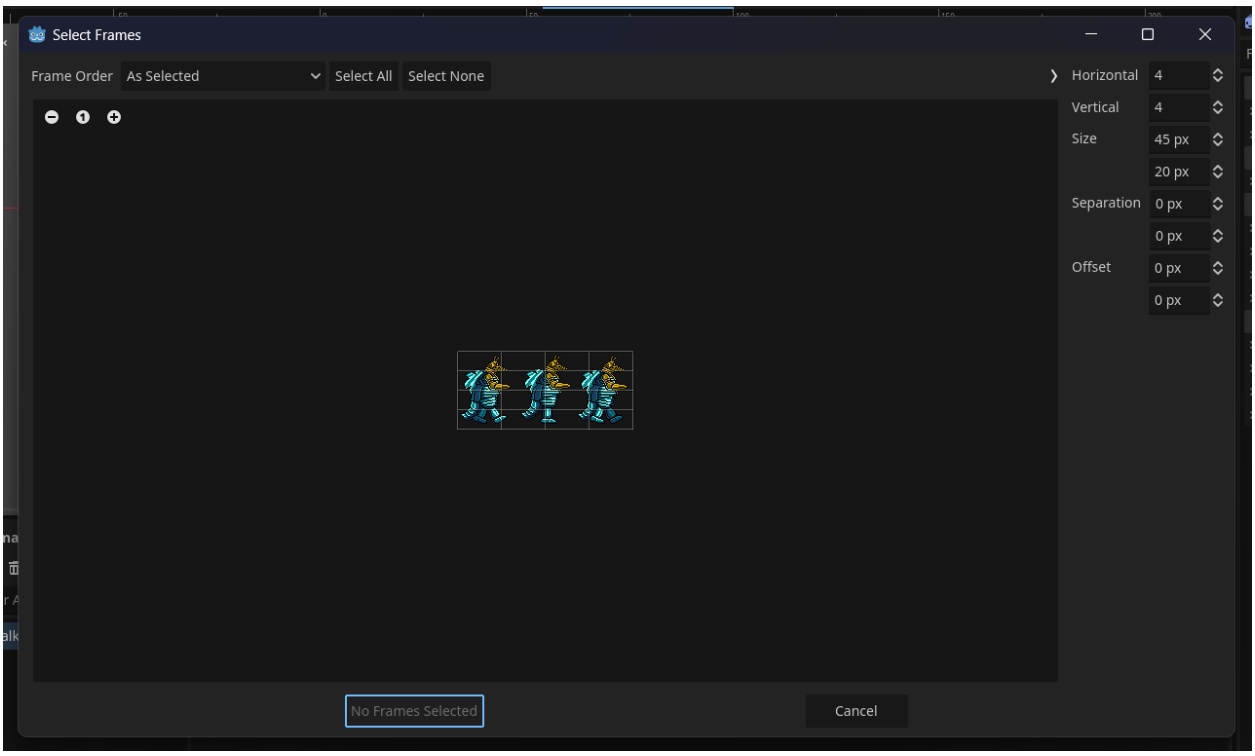


Rename it to "Idle".

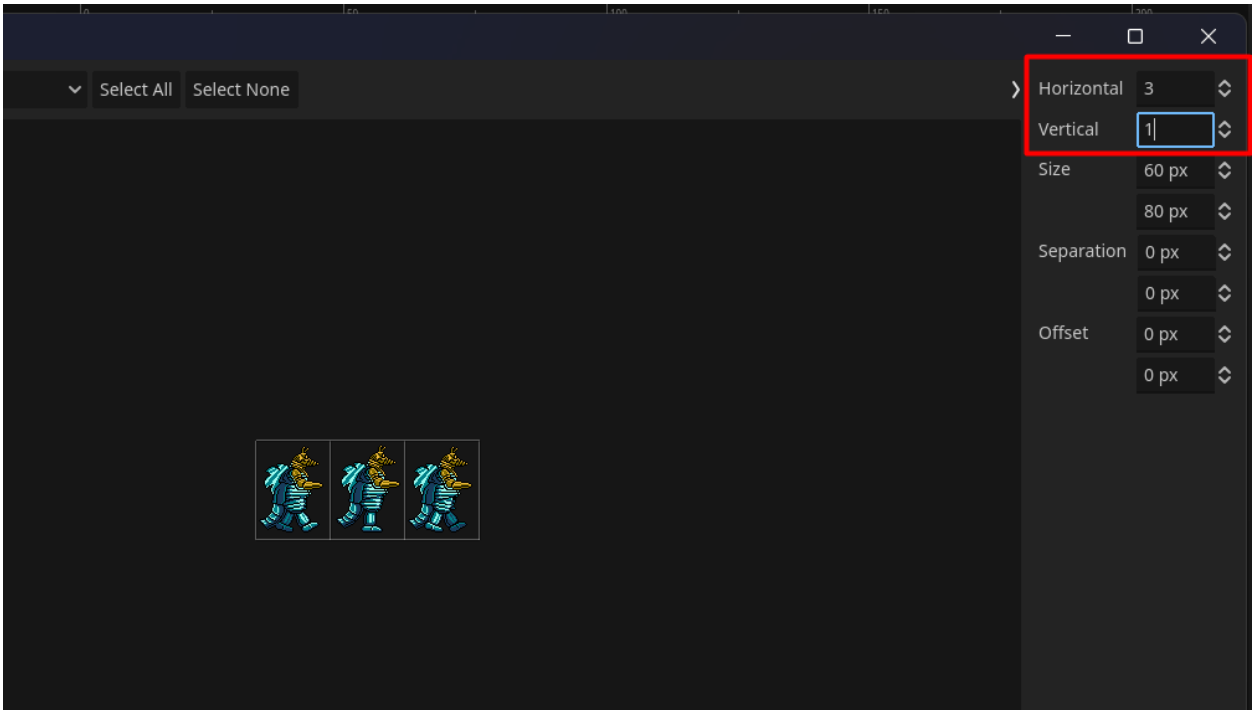
Now let's add our sprites:



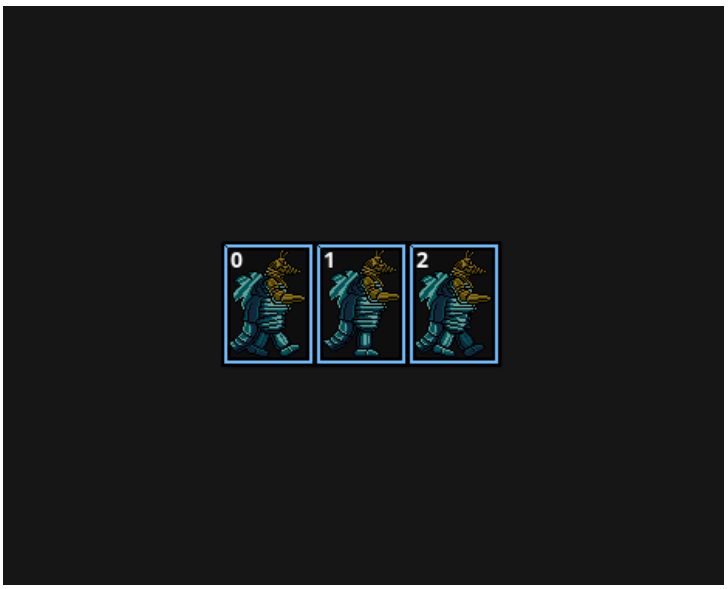
A file dialog will appear, select your sprites file in "Sprites/Characters/Moguera" folder. A new menu will open:



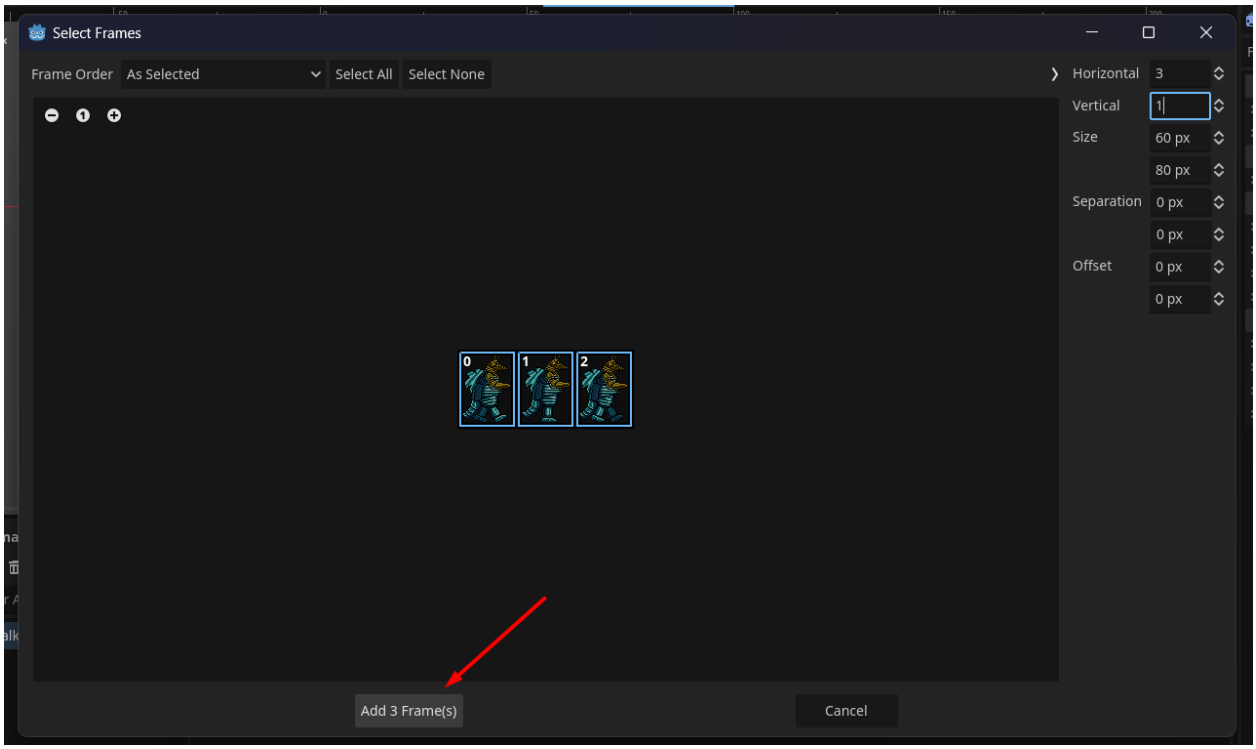
This is our sprite importer. Since there are 3 sprites arranged horizontally in just 1 vertical row, put 3 and 1 in “Horizontal” and “Vertical” options respectively:



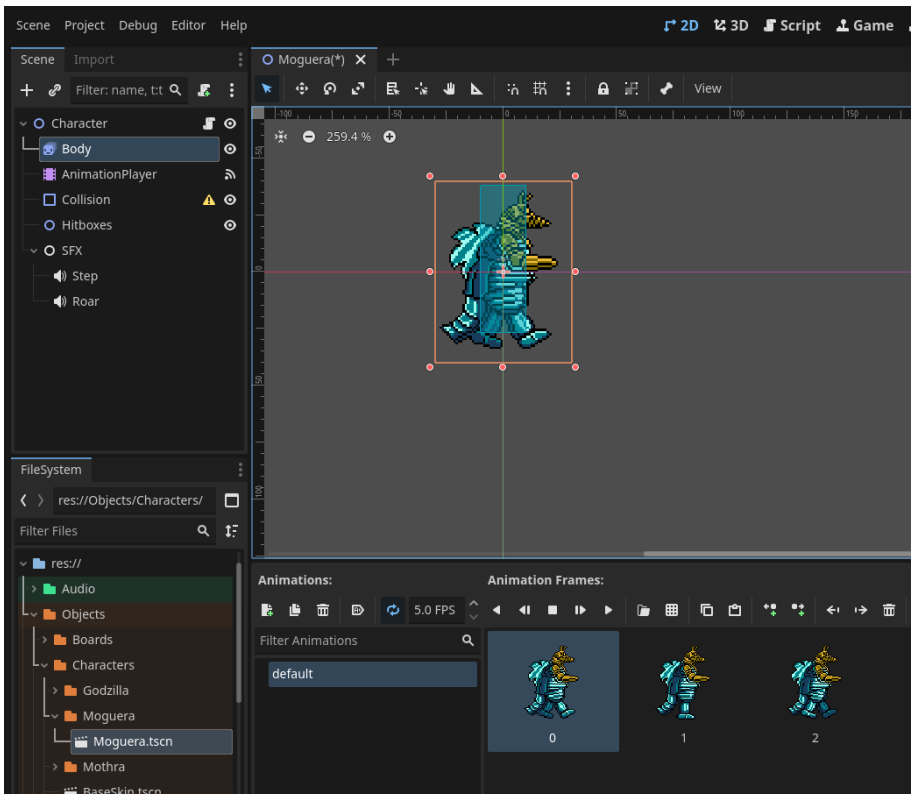
Now you can select every sprite



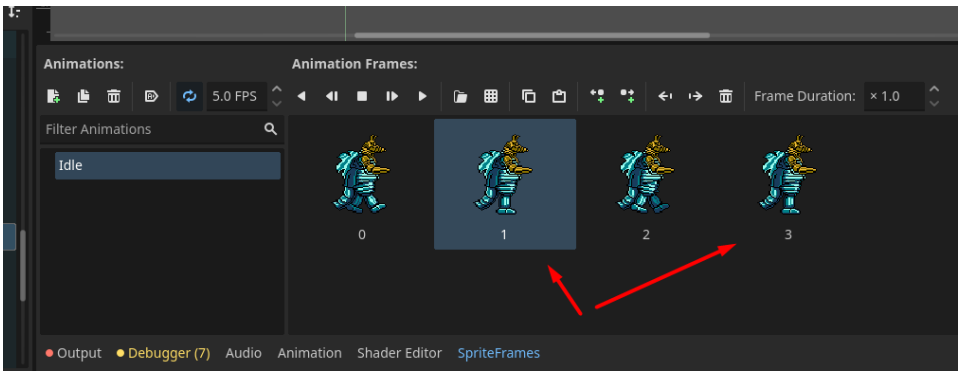
Press “Add 3 Frame(s)” to add the sprite frames.



Wow, we can now see our new character in Godot editor!



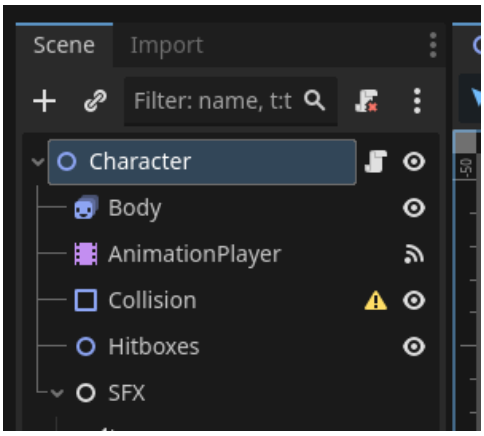
Now you may need to copy some animations frames to make the animation loop properly from the last frame to the first one, so for that you can just select a frame, press Ctrl+C and then Ctrl+V:



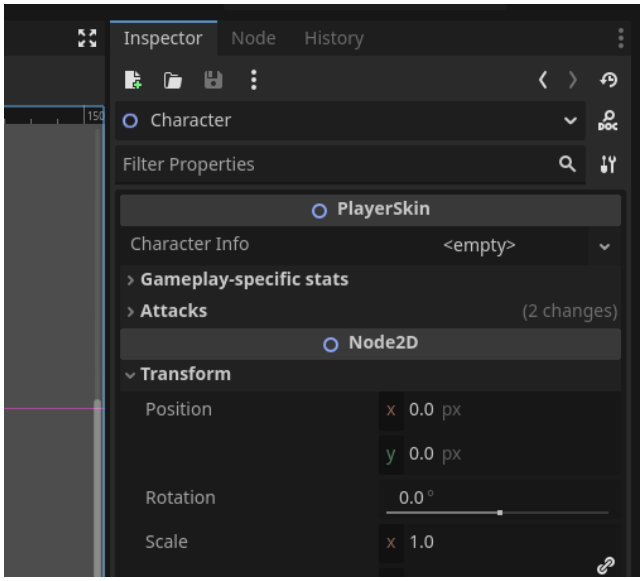
Now the sprites are imported!

Changing character's properties

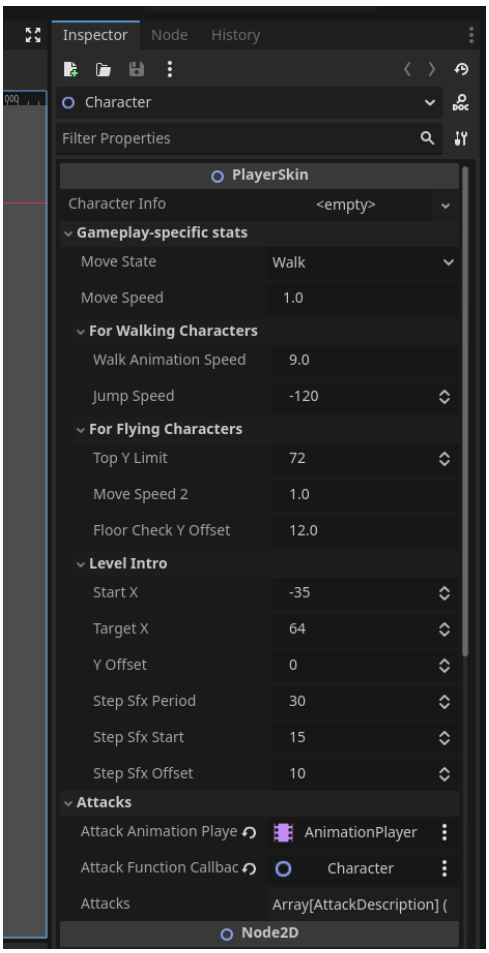
Click on the top node named "Character":



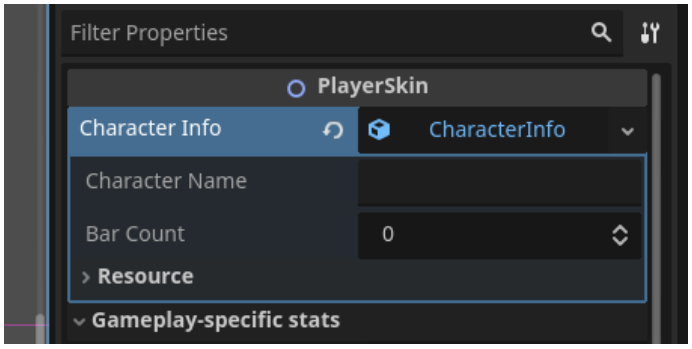
Now look at its properties:



There's a Character Info property and 2 groups. If you open all the groups, you will see all the character-specific properties the skin has:



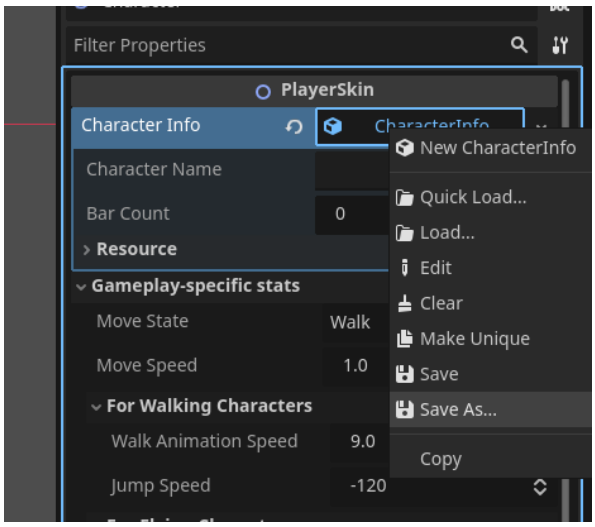
That maybe looks like a lot, but it's really not, I will explain each one in a minute, but first we'll have to deal with the Character Info property. Click on "<empty>" next to it, and select "New CharacterInfo" button that will appear after you click on it.



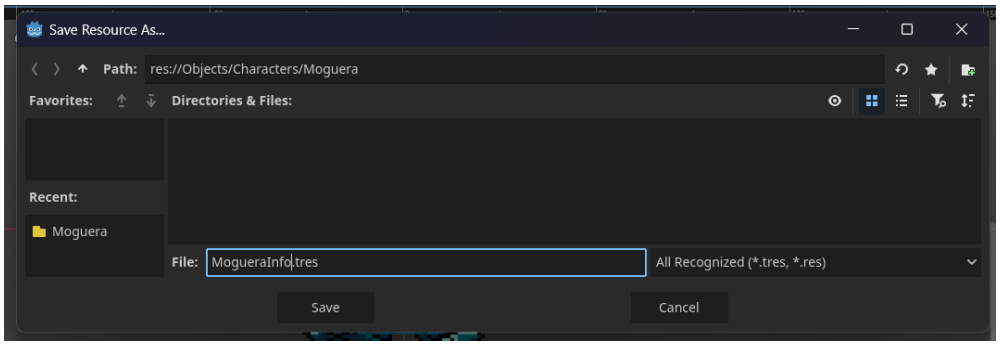
Now you can see 2 additional character-specific properties: name and "bar count".

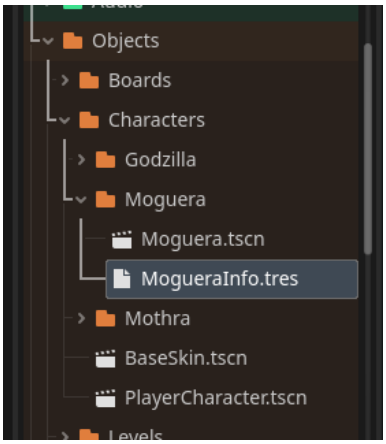
"Bar count" is the number of bars of life and power the character should have.

Now right click on "CharacterInfo" (without the space) text and select "Save As":



Save it inside the Moguera skin folder and name it “MogueraInfo.tres”:



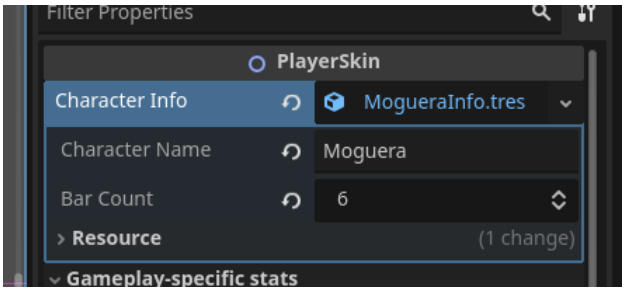


Nice!

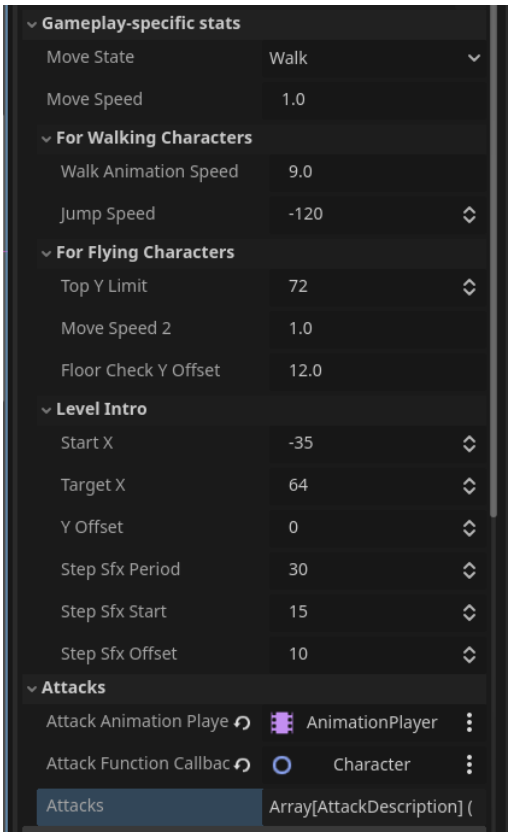
Why are they inside of a “CharacterInfo” stuff and not like inside the other 2 groups like the rest of the properties? You see, there are character properties that should also be accessed outside the level stuff (we’re currently dealing with the character skin that will be inside of levels), like a board. So, if those properties were just a part of the skin like the rest, to access these properties in a board the code would have to create a temporary object of the level skin, get the properties, and then destroy the skin because a board is not a level. Sounds clunky, right?

So instead of doing all that, we move these 2 properties into a separate file so we can access them anywhere without creating unnecessary objects.

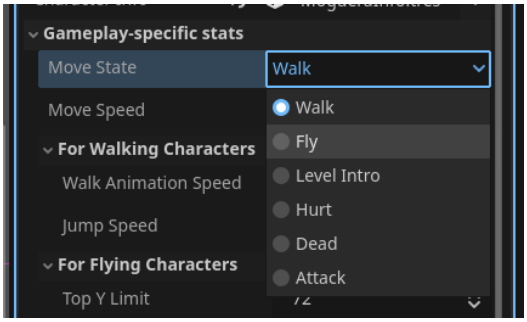
Now you can change these 2 properties and save the scene (by pressing Ctrl+S on your keyboard), it should save both the scene and the MogueraInfo file:



Now I will explain every property one-by-one.



1. **Move State** is the default state for the character, whether it's walking or flying.



   If your character is a walking character, it should be "Walk", if it's a flying character, it should be "Fly".

The game will return to this state whenever the character needs to change into its default state, like after getting hurt or after the level intro finishes.

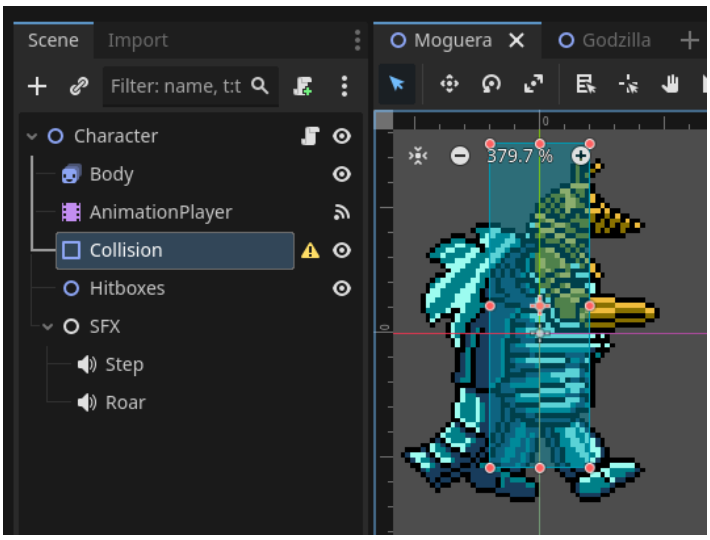
2. **Move Speed** is the default moving speed for the character when it moves.
3. **For Walking Characters - Walk Animation Speed** is the speed of the walking animation. The bigger the value, the faster the animation.
4. **For Walking Characters - Jump Speed** is the vertical jump speed when a walking character presses Up button.
5. **For Flying Characters – Top Y Limit** is, well, the top vertical limit where the character won't move any higher.
6. **For Flying Characters – Move Speed 2:** When the camera is following the player when they're moving right, in this case this move speed for horizontal axis is used and not the regular move speed. You probably haven't noticed that while playing the original games, but when the camera moves horizontally along with Mothra her horizontal speed is decreased.
7. **For Flying Characters – Floor Check Y Offset** is the height where the flying character should stay if they're moving down and touching the floor, it's useful if the character's collision box should be of reasonable size but the skin still shouldn't clip into the floor.
8. **Level Intro - Start X** is the X position the character starts in when the level intro starts. If you're unsure why you would need to change that you can leave it as-is.
9. **Level Intro – Target X** is the X position of the character when the level intro should stop.
10. **Level Intro - Y Offset** is the offset from the default character position in a level on the vertical axis, it's used by Mothra to make her start in the air when the level starts while Godzilla would be on the ground.
11. **Level Intro – Step Sfx Period** is the number of physics frames (or just game frames if the game's FPS is 60) between 2 step sounds. 60 frames is 1 second, 30 frames is half a second and so on.
12. **Level Intro – Step Sfx Start** is when the step sound effects should start (in physics frames), so if it's 60 then the steps won't play in the first second of the level intro.
13. **Level Intro – Step Sfx Offset** is the offset of when the steps sfx should play, it's needed to more precisely sync the sounds with the character walking animation.
14. **Attacks – Attack Animation Player:** The player's attack component attacks can change the player's animation, so that's where we specify the skin's animation player that will be used when an attack starts.
15. **Attacks – Attack Function Callback**: Sometimes attack should have their specific code, outside of what the attack component's attack properties can provide, and inside Godot custom function can only be inside classes, which can be instanced by themselves or used as a node in a scene, and since we

want the player-specific code to be inside the player's skin that's where you specify the player's node which can contain custom attack functions.

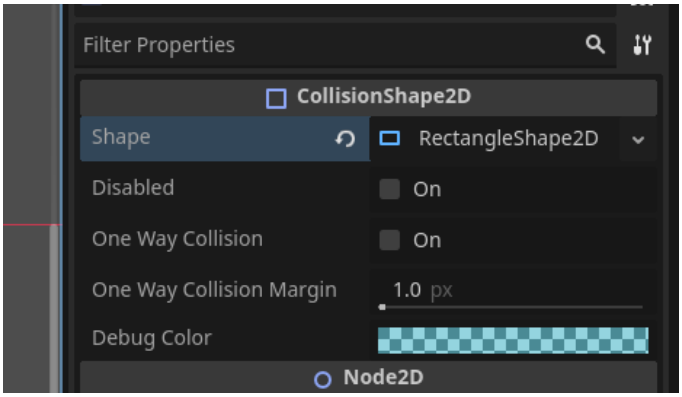
16. **Attacks – Attacks**: That's where we specify the attacks the character should have, more on that in a different tutorial.

Now you can change the properties however you want and however you see fit!

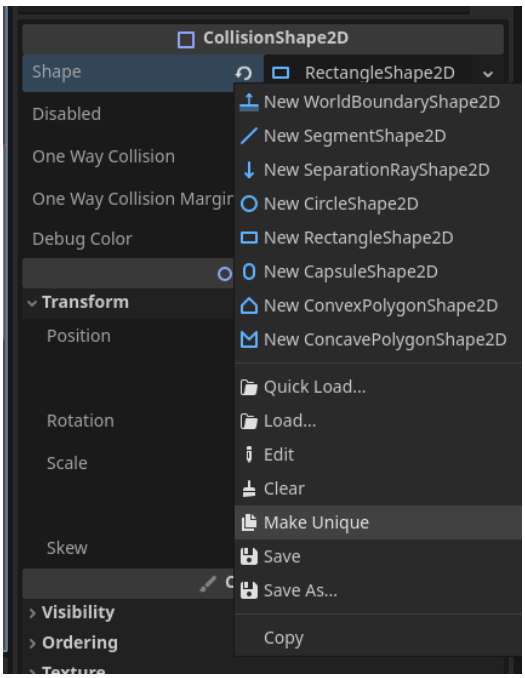
You can also, of course, change the player's collision box using the "Collision" node



But first you need to open its properties:

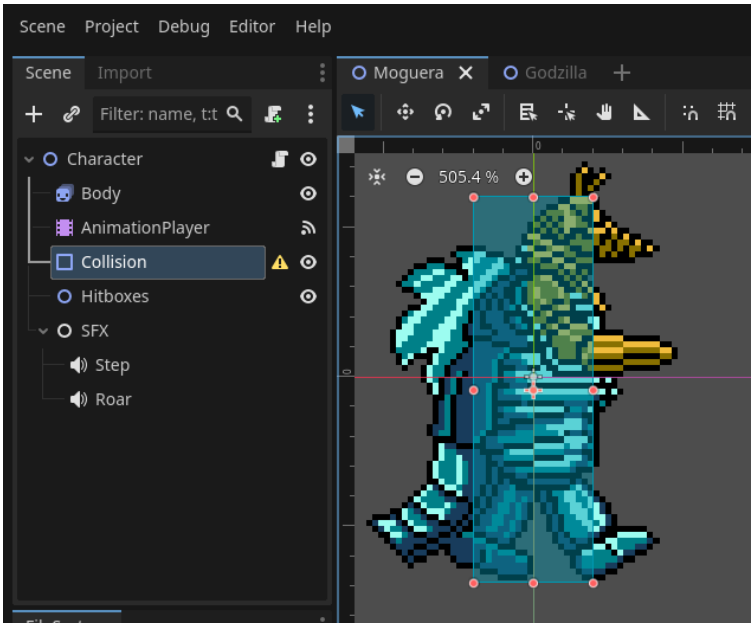


Right click on "RectangleShape2D" and then on "Make Unique":



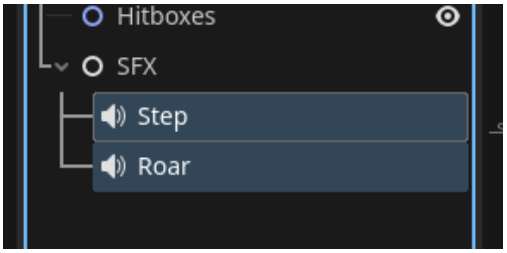
This will ensure that if you change the shape the base skin's collision (and possibly some other characters that use the same shape resource) will stay the same.

Now you can change and reposition the collision shape:

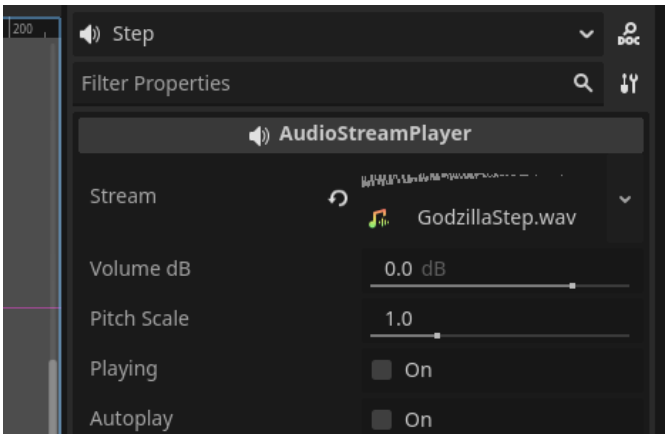


(Well technically you already could reposition it but still)

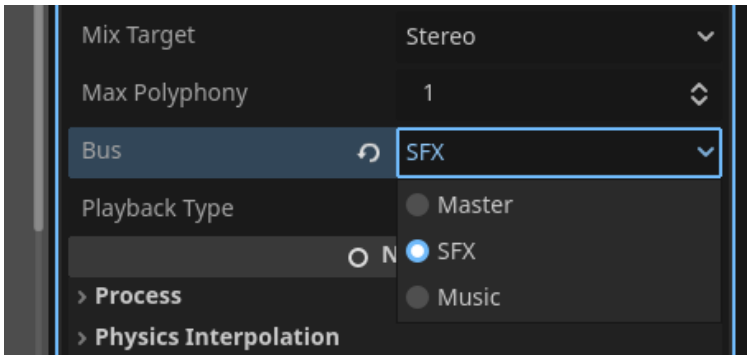
You can also change the character-specific sound effects inside the "SFX" node:



Just select a sound effect and then drag and drop a sound effect from the file system dock onto the stream property of the sound effect:



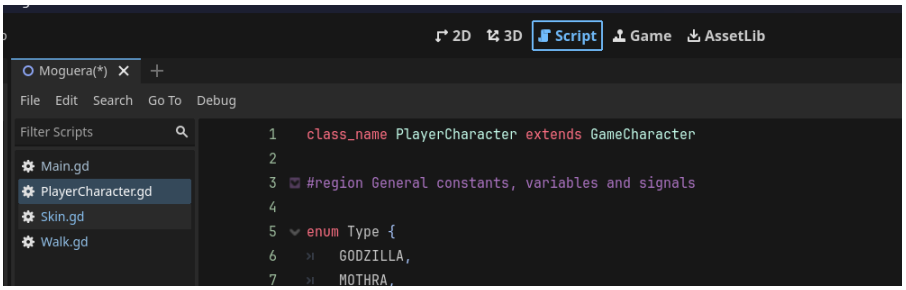
And if you want to add new character-specific sound effects, don't forget to change the bus of the AudioStreamPlayer node to "SFX":



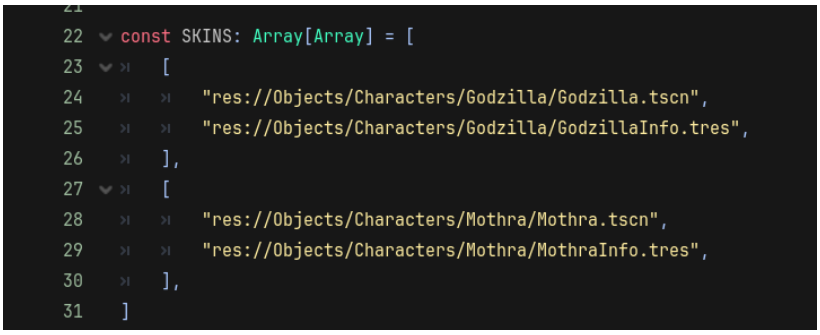
The audio buses are used to change the volume of the individual audio group in the game in the settings menu. And also the "Master" bus just means there's no custom bus assigned to the node, it's the default bus present in any Godot project.

Make the skin load

Our character skin should be ready now! Let's make the game load it. Open the "Script" tab:

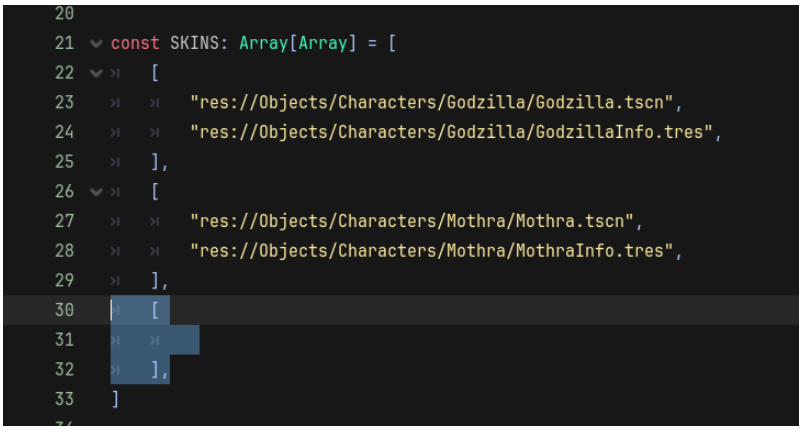


If you haven't closed any script files in the tab you should still have the PlayerCharacter.gd file opened. Select that file and scroll a little bit down until you find the code that looks like this:



```
const SKINS: Array[Array] = [
    [
        "res://Objects/Characters/Godzilla/Godzilla.tscn",
        "res://Objects/Characters/Godzilla/GodzillaInfo.tres",
    ],
    [
        "res://Objects/Characters/Mothra/Mothra.tscn",
        "res://Objects/Characters/Mothra/MothraInfo.tres",
    ],
]
```

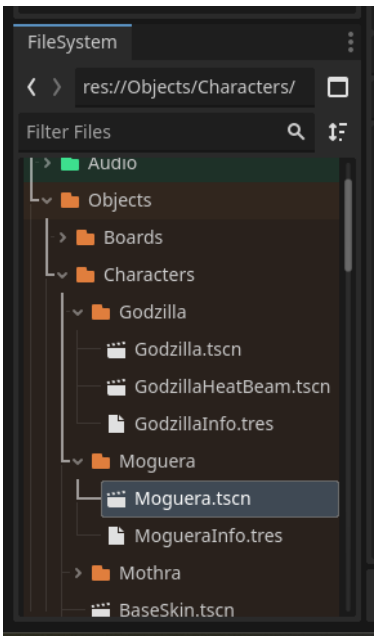
That's where we tell the game what files to load for the character. Create a new entry like this:



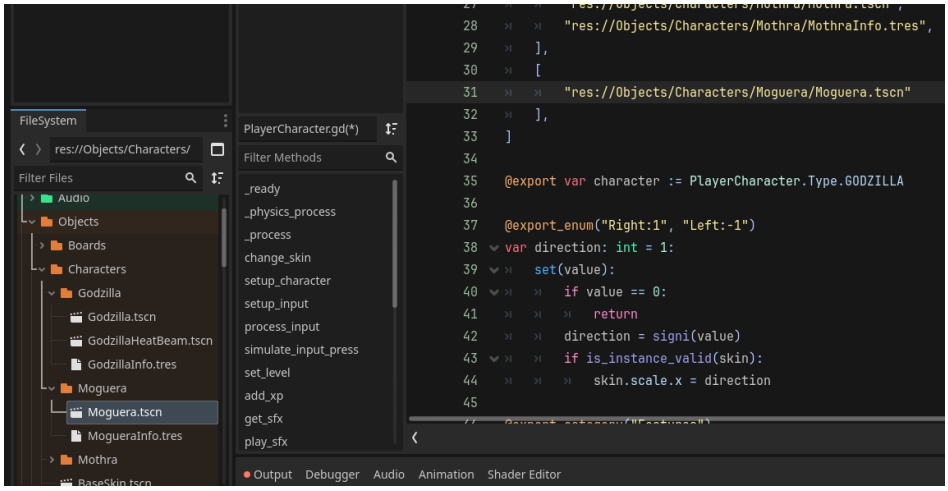
```
const SKINS: Array[Array] = [
    [
        "res://Objects/Characters/Godzilla/Godzilla.tscn",
        "res://Objects/Characters/Godzilla/GodzillaInfo.tres",
    ],
    [
        "res://Objects/Characters/Mothra/Mothra.tscn",
        "res://Objects/Characters/Mothra/MothraInfo.tres",
    ],
    [

    ],
]
```

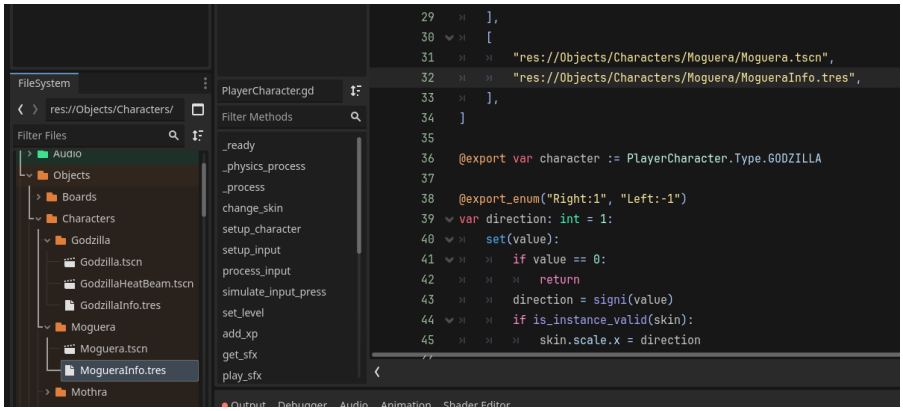
Now find the files created for your new characters in the file system dock:



Now you can drag and drop the skin onto the new entry like this:



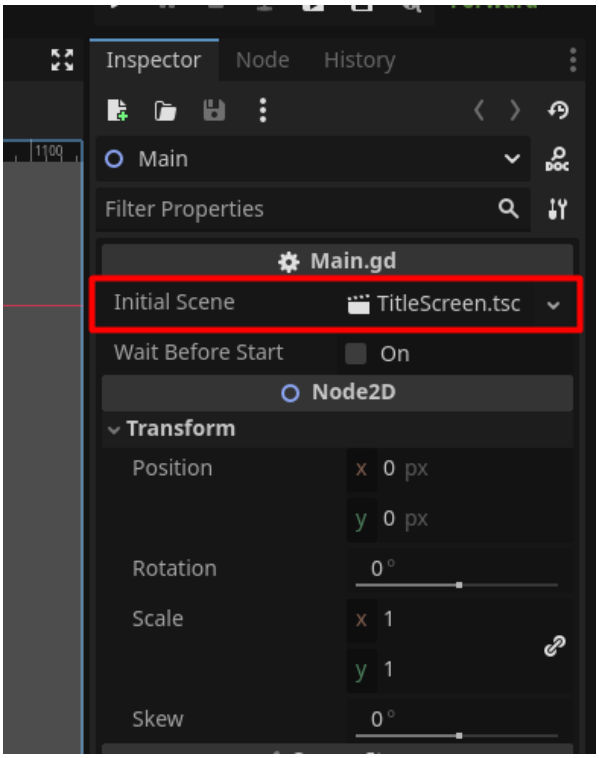
Now we just need to add a comma after this text, add a new line and do the same for the info file:



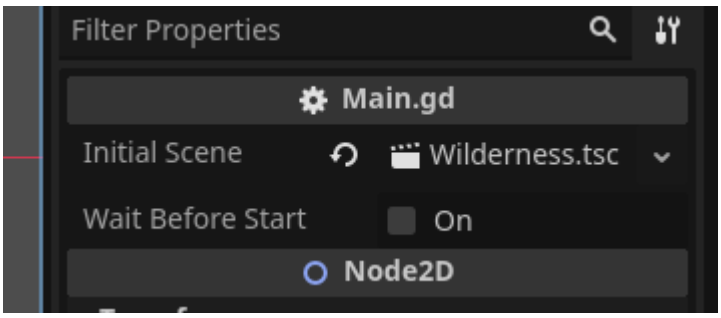
The comma just before the end of a list (we created one using square brackets) is not really required but it’s there for consistency, so all entries have a comma at the end.

Testing our new character

Now let's look at our character in-game! First, let's make the game to start on a level. In the Godot file system (bottom-left corner of the screen), find the file "Scenes/Main.tscn", now double-click it and look at the properties in the right part of the screen:



Now we know where we can change the initial scene the game starts at. Now where's the level scene? Let's choose the wilderness level. In the Godot file system, find "Scenes/Levels/Wilderness.tscn", now click and hold it with left mouse button, drag it onto the "Initial Scene" property value (where it says "TitleScreen.tscn" like on the picture above), now stop holding the left mouse button. You will notice that the initial scene property was changed:



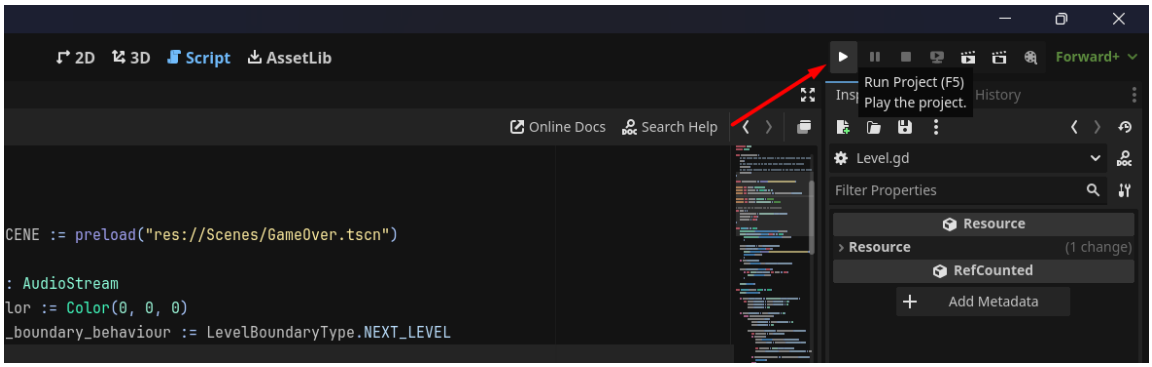
But if you run the game, Godzilla will be the default character, so let's change that. In the Godot file system, find "Scripts/Levels/Level.gd", that's the level scene code, and in the GameplayData class you will see the default value for the character ID:

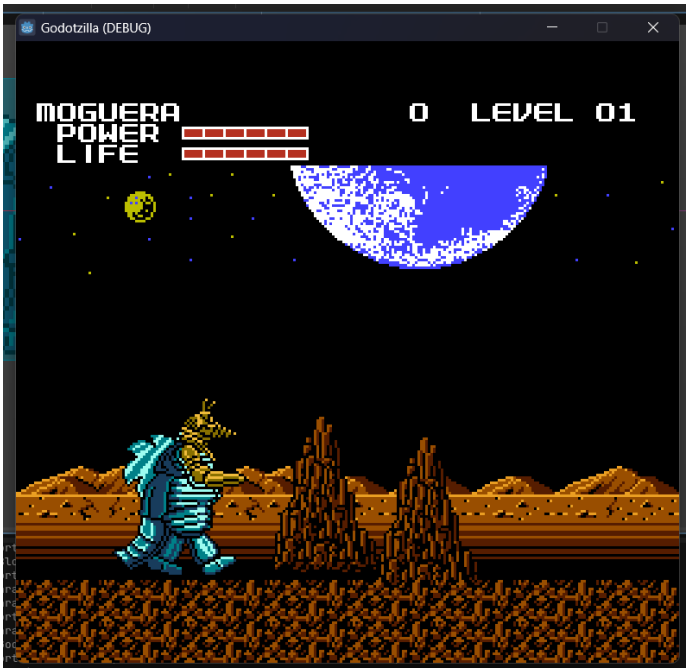
```
18
19  ## Important gameplay data that should be passed between levels (and from the boards)
20  class GameplayData:
21      var current_character := PlayerCharacter.Type.GODZILLA
22      var board_piece: BoardPiece = null
23      var boss_piece: BoardPiece = null
24
```

That's our type enum that we added a new entry in! You can now change "GODZILLA" to "MOGUERA":

```
18
19  ## Important gameplay data that should be passed between levels (and from the boards)
20  class GameplayData:
21      var current_character := PlayerCharacter.Type.MOGUERA
22      var board_piece: BoardPiece = null
23      var boss_piece: BoardPiece = null
24
```

Now you can start the game by either pressing F5 or pressing this button in the top-right corner:





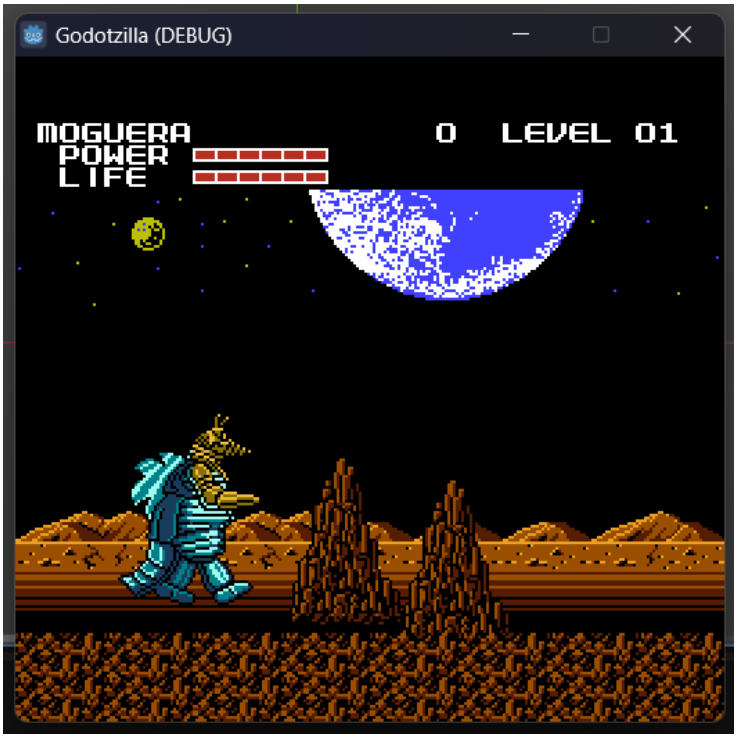
We can now see our character on the screen!

If you think the walking animation is too fast, you know what values to change! (Check the explanations from 2 sections earlier)

Fixing skin position and animation

If your character appears to be inside the floor or in the air, you can reposition the character's skin. Do you remember where's your skin file? If not, it's in "Objects/Characters/<your character name here>" folder, double click on "Moguera.tscn". Now you can move the sprite upper, you can just drag it.

Now let's test again



Keep moving the sprite until it looks alright:

If your characters looks a bit shaky while walking and the animation looks weird, it’s almost the same thing as the skin position, you can change the position of each sprite on the spritesheet by separating them and moving them individually.

Board object for the character (a “board piece”)

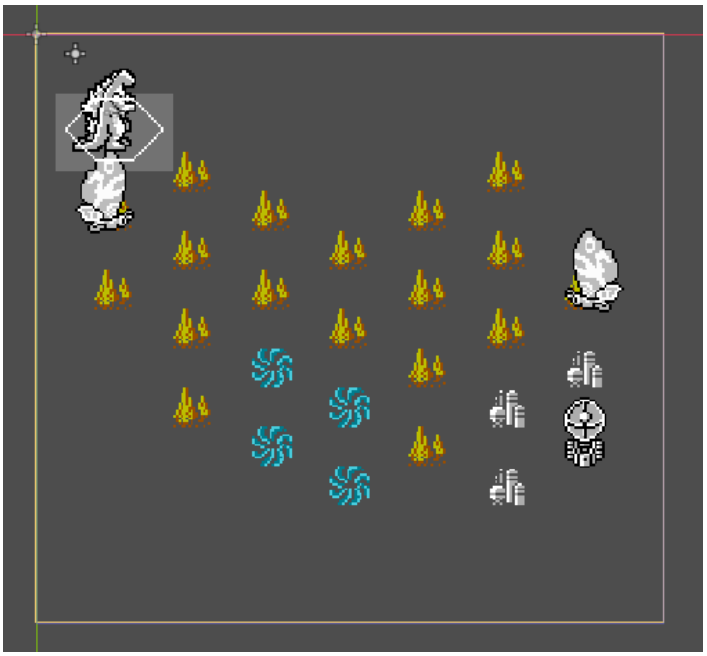
I think I should specify what I mean by a “board piece”. In the project code I refer to the character objects (both the player ones and the boss ones) on the boards as “pieces”, so keep that in mind.

Don’t forget that there also should be a specific board piece that can be used to control where the character goes on the board! First, let’s find out where are the board pieces’ sprites are located: it’s “Sprites/BoardSprites.png”, and you will notice that there are sprites for both Godzilla and Mothra pieces as well as other board sprites, such as the selector and message window.

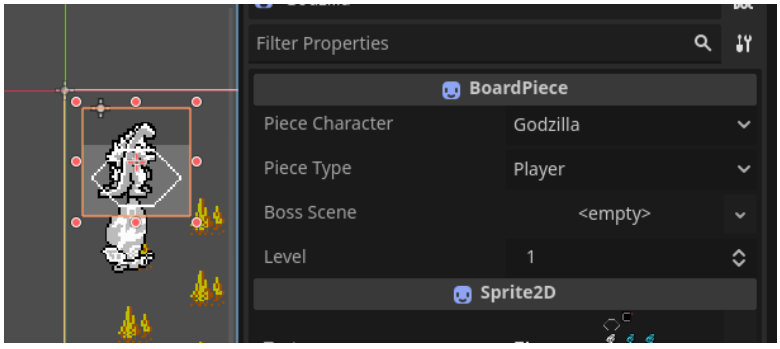
Let’s make some space for our new character. Every piece sprite here is 48x48 in size, so let’s add a new 48-pixel-high row.

Nice, now let’s add our sprites.

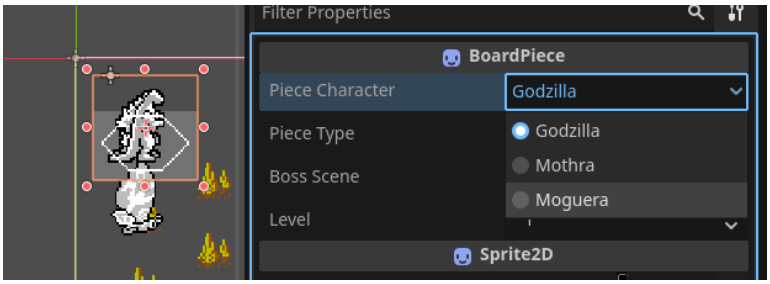
Great! Now let’s actually make a new board piece. First, open a board scene, for example, “Scenes/Boards/TheEarth/TheEarth.tscn”.



Now click on any character piece and look at its properties.



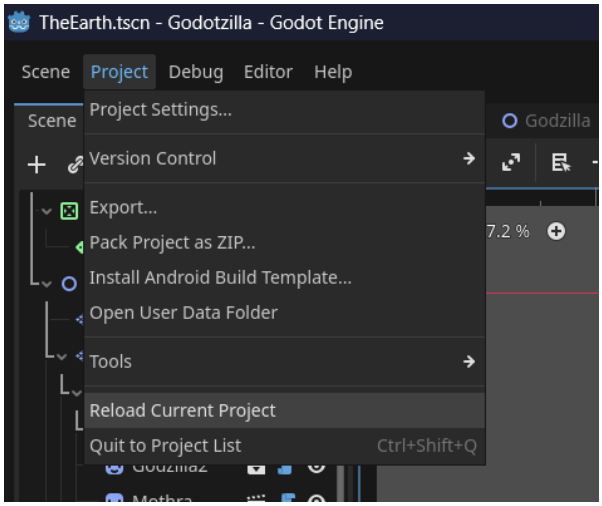
We can set its character with “Piece Character” property, click on the dropdown box next to it.



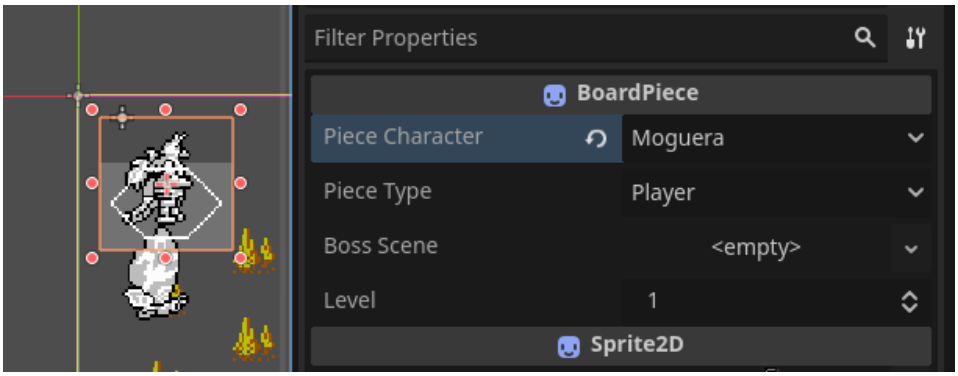
Wait, what? Moguera is already there? I mean, we didn’t even have to add any texts ourselves to make it work in the editor.

Yes! Godot is smart. See, the piece character property actually has the type of our character type enum that we added a new entry in in the first chapter. Godot automatically gets every enum entry and changes its name to something more readable. Amazing!

(If you don’t see your character’s name here, just restart the Godot editor)

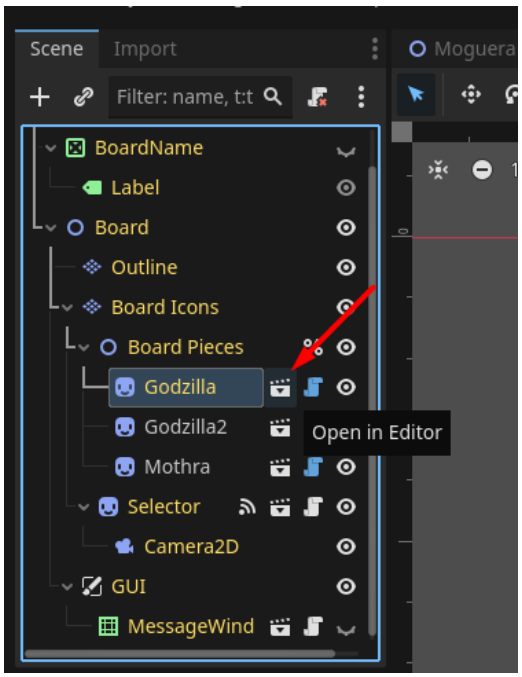


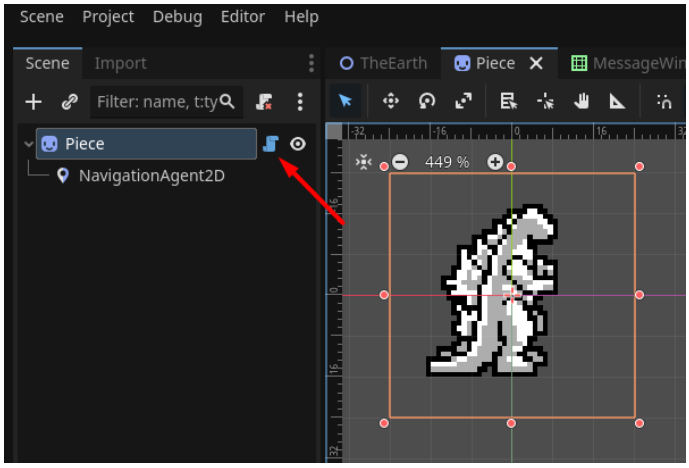
Now click on our new character’s name in that dropdown box.



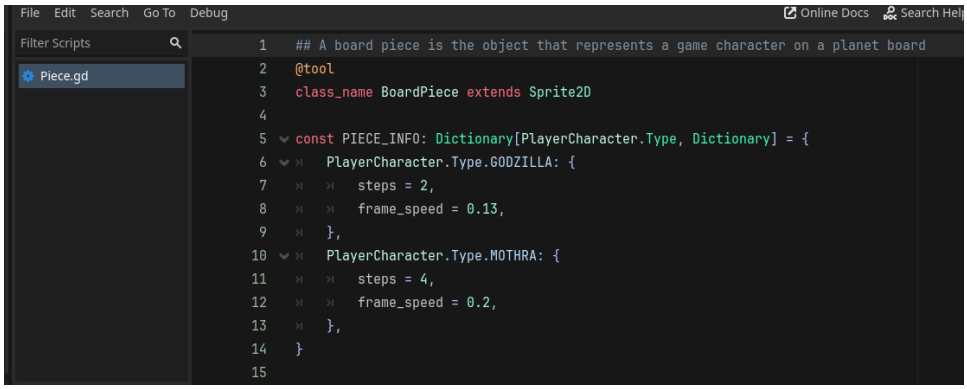
The piece immediately changed its sprite to show our new character, neat!

There are still some piece properties we haven’t changed yet, so open the piece scene:





Click on this button to open up the piece script, and scroll up to the very top of the script.



Here are some piece parameters: piece steps and frame speed. “Piece steps” means how much levels on the board can the character move through on the player’s turn, and frame speed is actually how much time (in seconds) is required for a piece to proceed to its next frame while moving. I’m going to use the same values as Godzilla for this tutorial.

```gdscript

const PIECE_INFO: Dictionary[PlayerCharacter.Type, Dictionary] = {
    PlayerCharacter.Type.GODZILLA: {
        steps = 2,
        frame_speed = 0.13,
    },
    PlayerCharacter.Type.MOTHRA: {
        steps = 4,
        frame_speed = 0.2,
    },
    PlayerCharacter.Type.MOGUERA: {
        steps = 2,
        frame_speed = 0.13,
    },
}
```

Now, remember the chapter about testing our character, where we set the initial scene to be a level? You can do the same thing but change the level to the board we're currently on in the editor, "Scenes/Boards/TheEarth.tscn".

Wow, it works! We now have our character on the board!

If you select the character and it looked like the sprite moved, you can change the sprite positions on the board sprites spritesheet.