

# Animation Lab



What are we gonna build? Ghostbusters! Over the course of the next 2 labs we will build a simple Ghostbusters game. This first lab will be focused on animation, and we will move on to

## Objective

- To get familiar using Godot
- To learn the 2 methods for animating movement in Godot
- To understand how to write basic scripts in GDScript

## Partnerships

- None. This lab is individual work only

## Godot

Godot is an open source game engine. Instead of using objects, everything is built out of nodes. A cluster of nodes is called a scene.

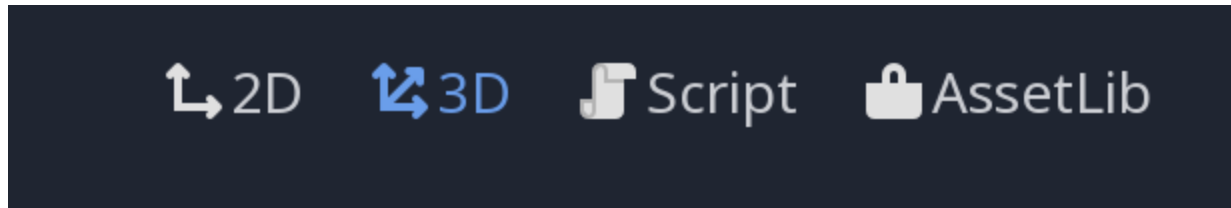
## Installation

Download the engine off of: <https://godotengine.org/download/osx>. We recommend the standard version as opposed to the mono version. This lab and the next rely on GDScript, so you won't need the C# capabilities offered in mono. Our other motivation to use standard is the ui is slightly different than monos, and there's more support available online for projects built with GDScript in standard than anything with mono.

## Project Set Up

Begin by creating a new project. Before Godot creates a project for you it will ask you for an empty folder to put the project in. Give it a name like lab<whatever lab number this is>. This

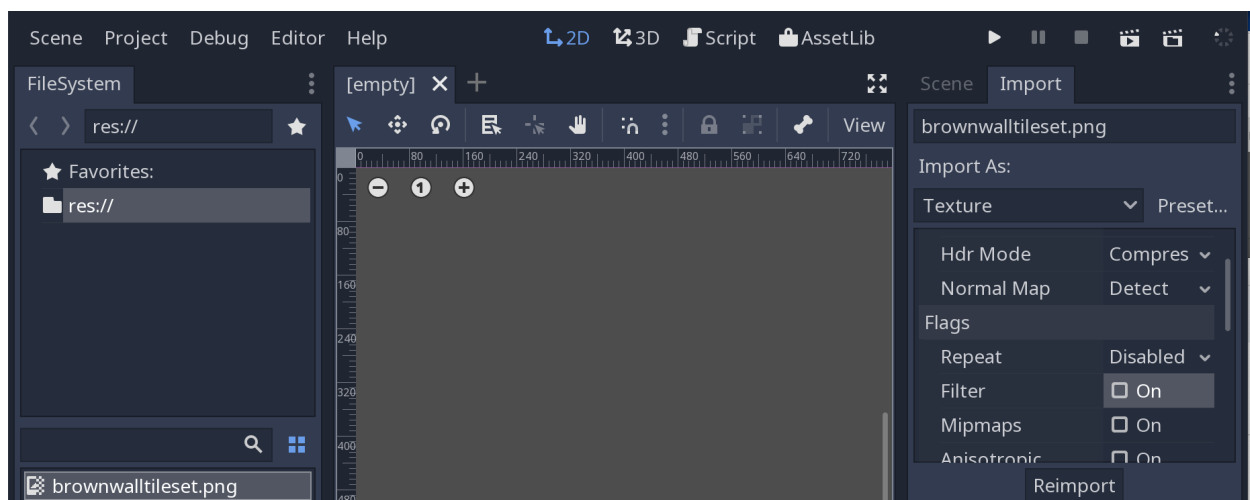
folder will store all your scripts, images, scenes etc. When the project opens the UI will automatically assume you want 3d. Switch it to 2D.



Download this file and copy the photos into your project folder:

<https://drive.google.com/open?id=1L6f49Az4MphkABnYJZop8NB5YUUah6m3>

Next in Godot click on the tiles image, then the import tab. Deselect filter and then reimport. Its important to turn off filter so then the tile borders turn out better.



Next, click scene, add, and then select Tile Map. To see details about your TileMap node select the inspector tab. Now you're all set for part 1.

## Part 1 - Building Your Map

To see how your sprites are interacting with the world, you're going to want to build a map for them. Please build a basic top down tile map using this tutorial:

[https://docs.godotengine.org/en/3.0/tutorials/2d/using\\_tilemaps.html](https://docs.godotengine.org/en/3.0/tutorials/2d/using_tilemaps.html)

*Once this is done save it as your map or course scene.*

## Part 2 - Animating Your Ghost

First open a new scene, and set its root node as a rigid player. Why a rigid player over a kinematic player you may ask? Well here is a handy chart of when to use which:


## Physic Bodies

### Rigid Body

Physics provided for you

Use when

- You're a beginner
- You need simple physics




### Kinematic Body

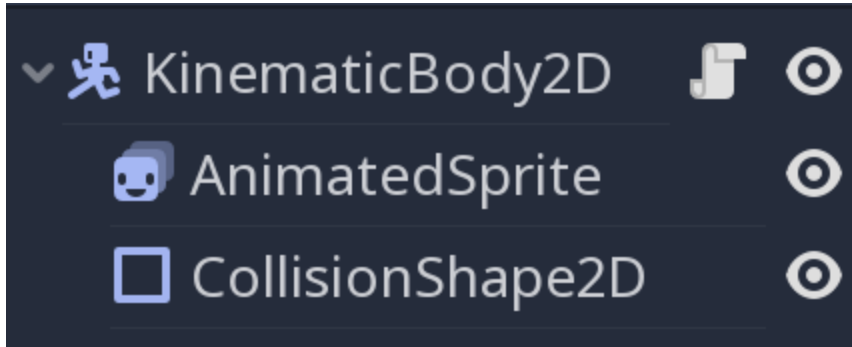
Write your own physics

Use when

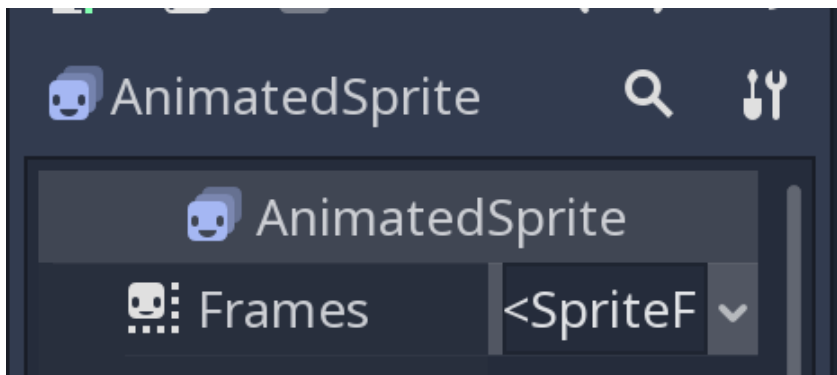
- You have a custom or changing env
  - Move slow in quicksand
  - Alternate between moon and space gravity
- You have power-ups that change your physics
  - Eat a berry that reduces gravity temporarily



Your final ghost node-tree should look similar to this, except using rigid body instead of kinematic body. General note on this lab, if you see kinematic body written anywhere replace it with rigid body. Please create this node setup in the scene tab:



Go to animated sprite, click the <Sprite Frames>



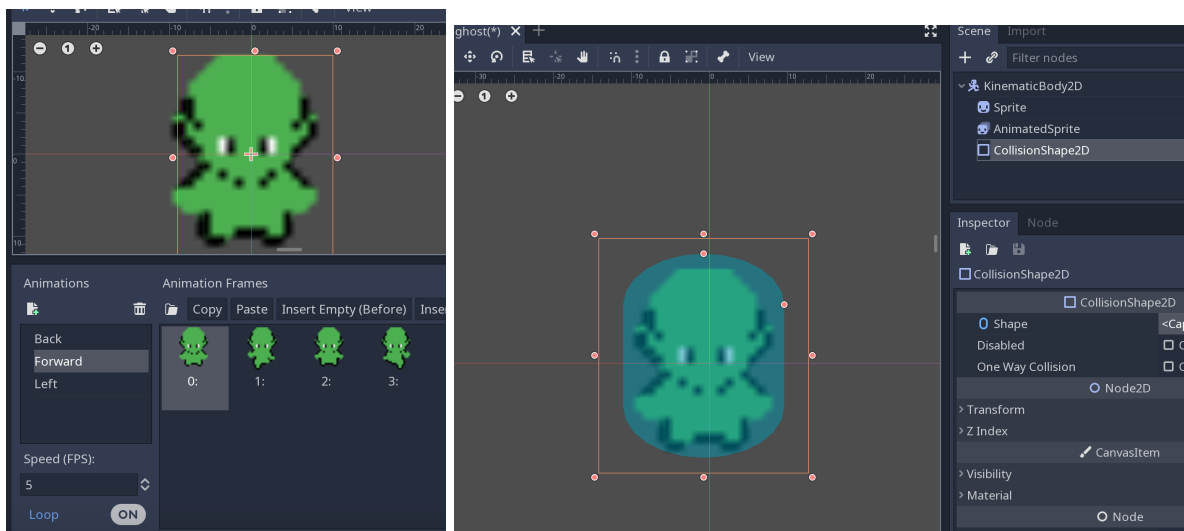
This should pop up a SpriteFrames interface. Lets build 3 animations left, up and down (you don't need right!) We'll explain why later in script writing.



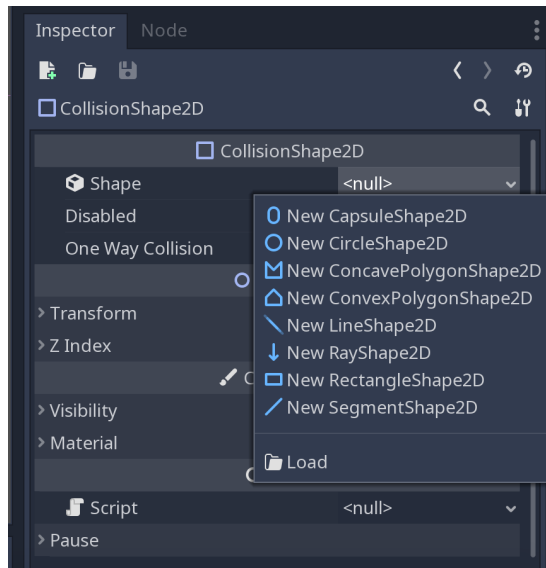
This kind of animation technique works consistently and is fairly easy to use, you just pick the picture you want to use and they become frames. The 2 major drawbacks of this style are

- 1) if you have a sprite sheet you need to make multiple copies of it and then crop each one to be of a single frame
- 2) Each frame is the same speed

There is an alternative way to animate your sprite, but we'll get to it in part 3. The result should look like:



Next add the node CollisionShape 2D to your kinematic character. Set a shape and then cover your character with it. Remember to adjust the size from the orange border, not the blue shape. This reduces the risk of Transform scale issues.



Now onto script writing.

Select your Rigidbody2D node, and then click the attach script button. The default script it gives you comes with a `_ready` function and a `_processdelta` function. Ready sets up the scene, and `_processdelta` updates the scene each animation frame.

Put this at the top of your code to set the constants and vars you'll need later:

```
export var speed = 400 # How fast the player will move (pixels/sec).
var screen_size # Size of the game window.
```

Use this given code snippet for ready -

```
func _ready():
    screen_size = get_viewport_rect().size
    pass
```

Lastly, this is the start of the process function for animating your sprite, some of the code has been provided, see if you can reason in the blanks. Make sure you watch for capitalization when calling the animation names

```
func _process(delta):
    var velocity = Vector2() # The player's movement vector.
    if Input.is_action_pressed("ui_right"):
        velocity.x += 1
    if Input.is_action_pressed("ui_left"):
        #fill in the blank
    if Input.is_action_pressed("ui_down"):
        #fill in the blank
    if Input.is_action_pressed("ui_up"):
```

```

velocity.y -= 1
if velocity.length() > 0:
    velocity = velocity.normalized() * speed
    $AnimatedSprite.play()
else:
    $AnimatedSprite.stop()
position += velocity * delta
position.x = clamp(position.x, 0, screen_size.x)
position.y = clamp(position.y, 0, screen_size.y)
if velocity.x != 0:
    $AnimatedSprite.animation = "Left" #capitalization matters
    $AnimatedSprite.flip_v = false
    $AnimatedSprite.flip_h = velocity.x > 0
elif velocity.y != 0:
    if velocity.y > 0:
        $AnimatedSprite.animation = "Forward" #or whatever you named the animation
    else:
        #fill in the blank

```

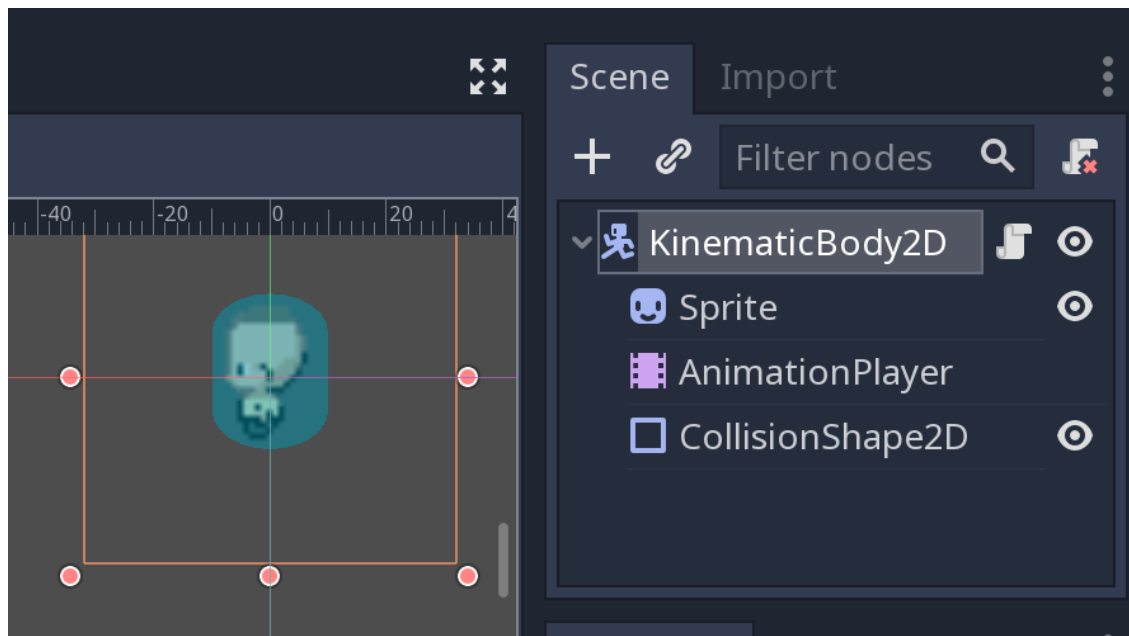
Now run the scene by clicking on the movie clapper thing with a play symbol on it.

In lab 2 we will modify our ghost so he can move on his own.

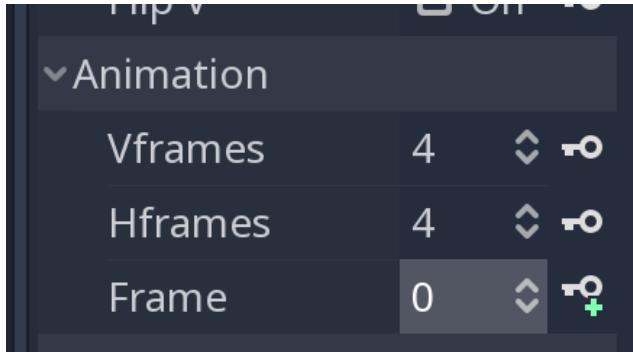
*Save your ghost scene.*

### Part 3 - Animating your sprite.

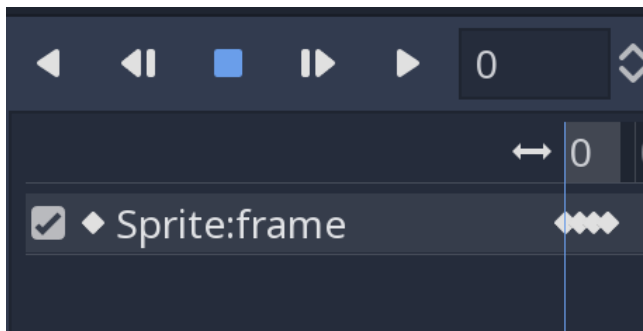
Create a new scene, and set it up as follows. We are using a sprite and AnimationPlayer instead of an animated player so we try animating the other way.



Set your sprite image as the ghost busters. Set the animation settings to match the dimensions of the sprite sheet. Then go into animation player and set up a new animation. Adjust the length to the amount of time you want. Then click on sprite again, and pick the frame number you want by clicking on the key+ symbol.



It should create something like:



The script will be a bit different. You're using `$AnimationPlayer.play("Right")` etc now, not `$AnimationSprite`. If you named your `AnimationPlayer` something else, call it by typing `$<name>`.

Also you need both right and left now! There is no direction flipping for animation player like there was for animation sprite.

#### Part 4 - Putting it all together

Create a new scene. Drag on your map and some of your characters And press play. Volia! You're done. Don't worry about your characters running through walls, they're ghosts so it is okay.

