02 - Godot, GDScript, Publishing

Jumping in

David G. Cooper, PhD

Designer Creates Experience

Illustration: Zachary D. Coe

To use this lens, stop thinking about your game, and start thinking about the experience of the player. Ask yourself these questions:

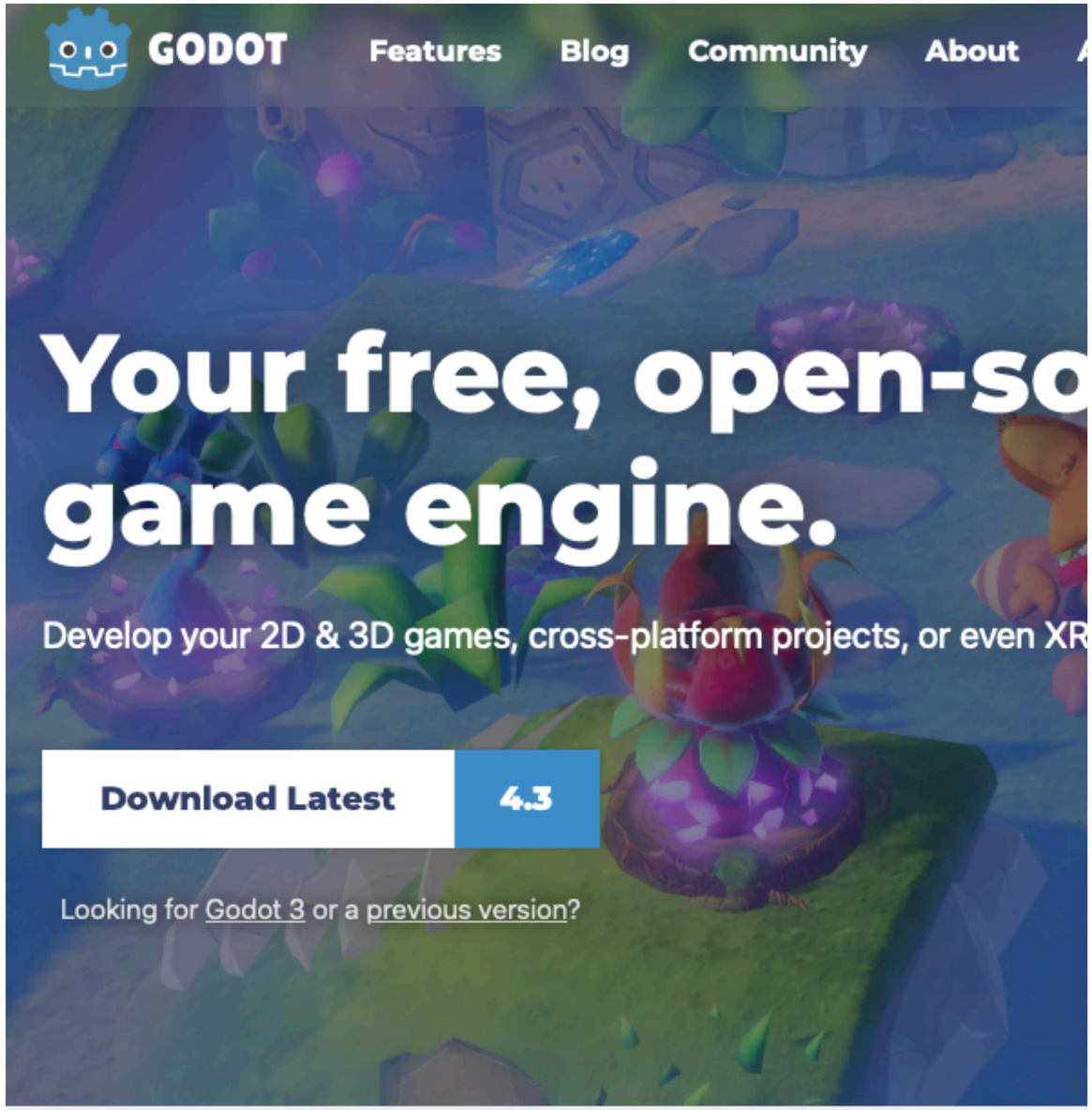
- *What experience do I want the player to have?*
- *What is essential to the experience?*
- *How can my game capture that essence?*

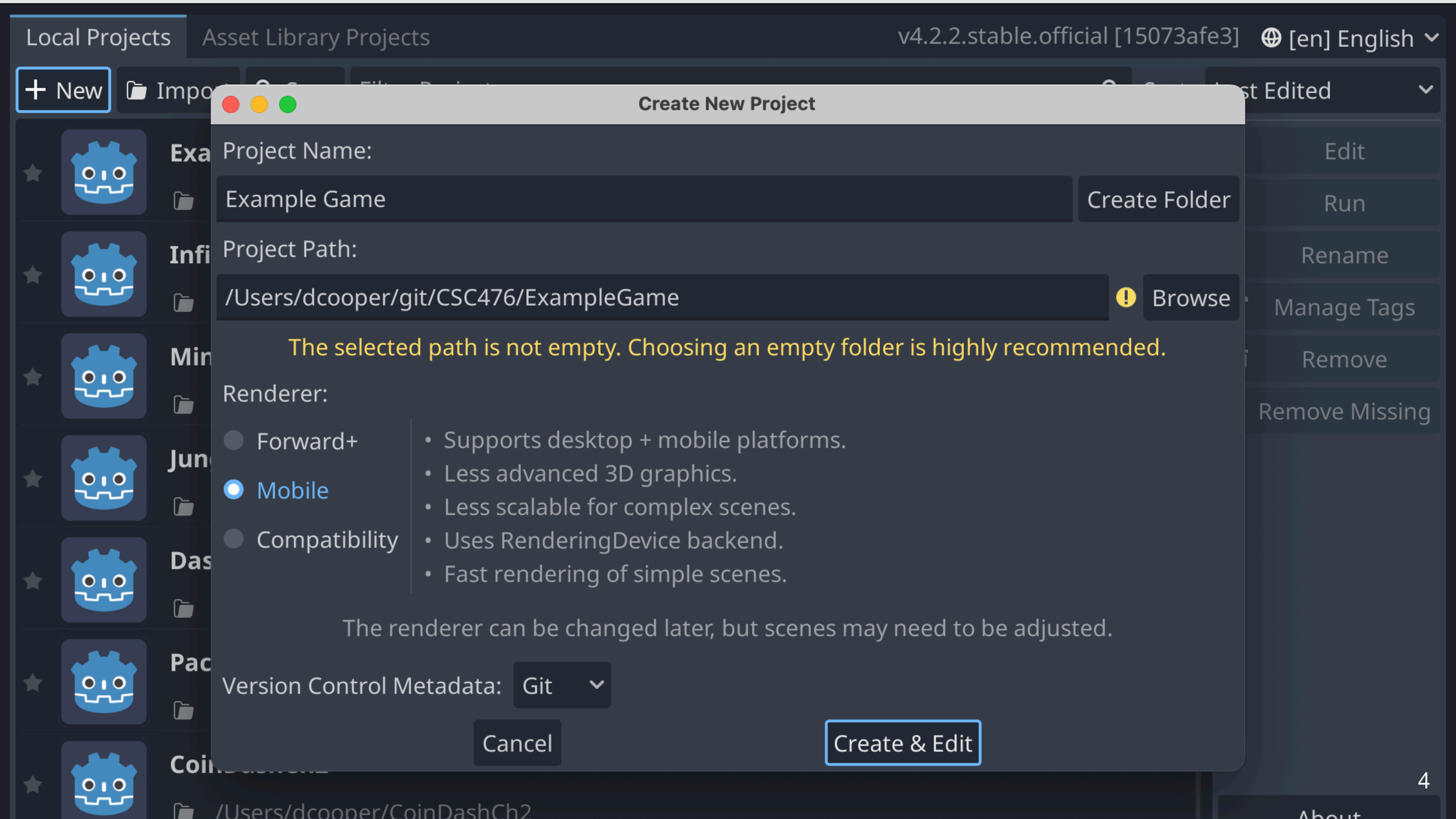
Illustration: Rachel Dorrett

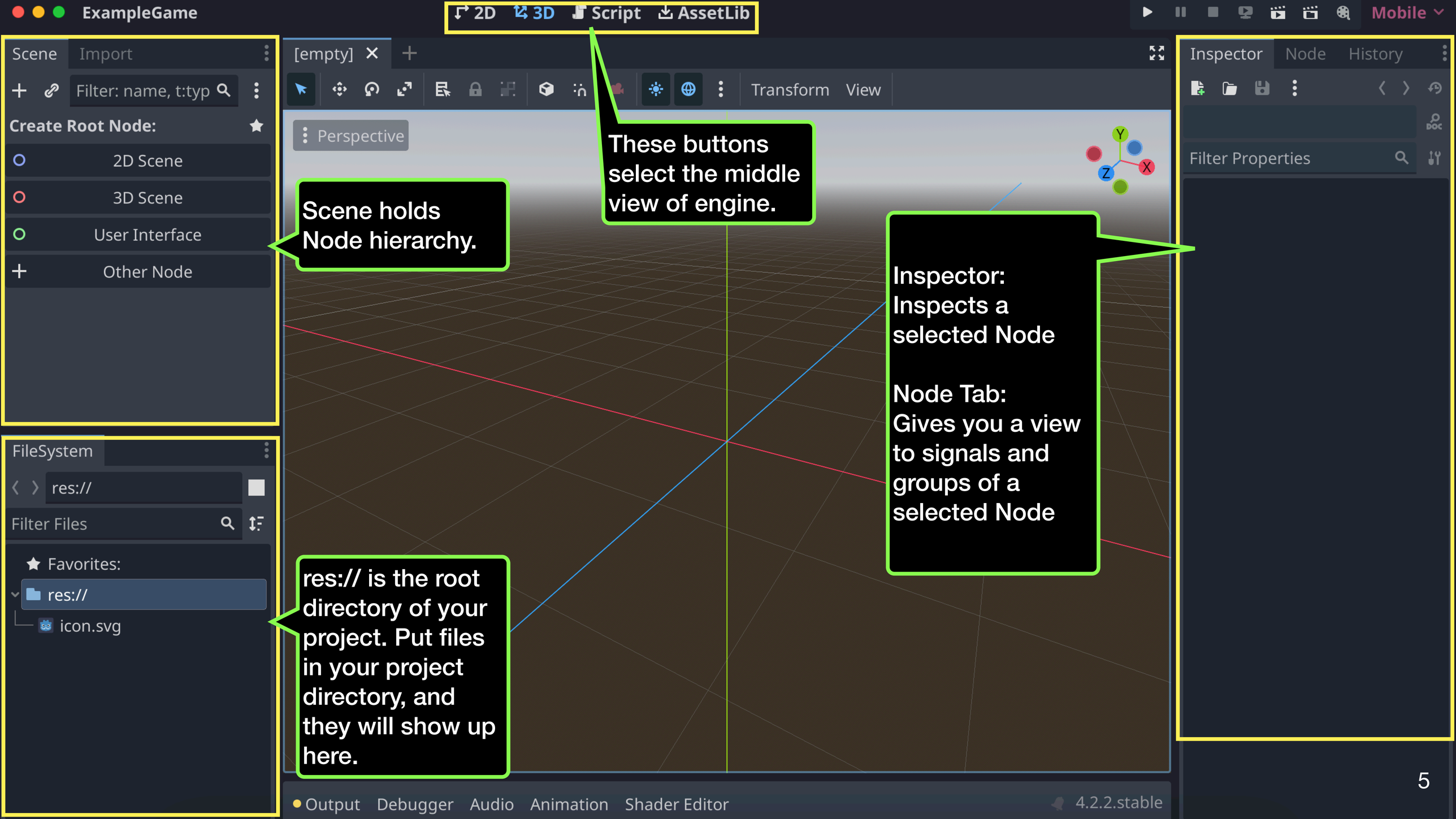
~People may forget what you said, but they'll never forget how you made them feel. -Maya Angelou~ To make sure the emotions you create are the right ones, ask yourself these questions:

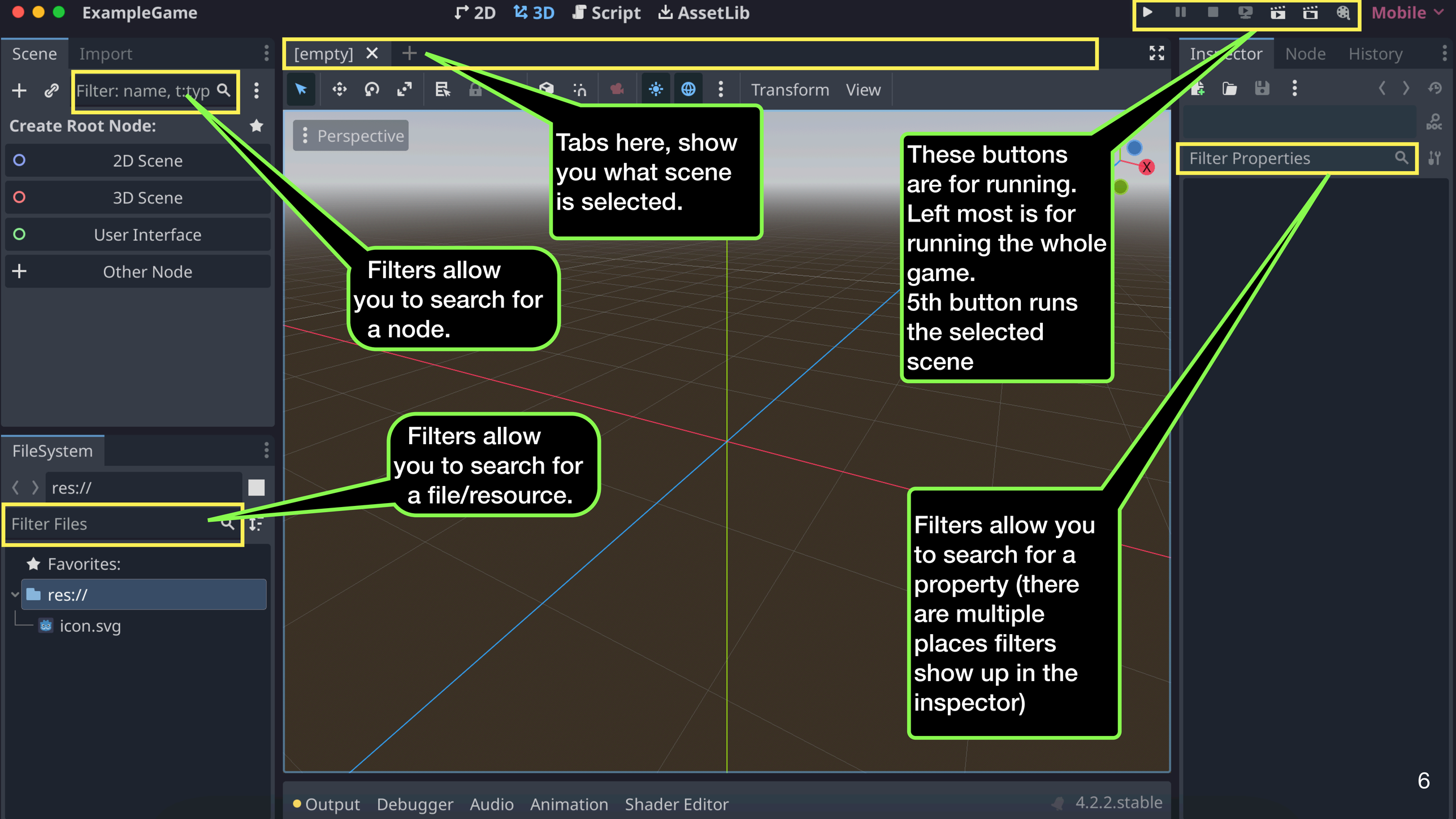
- *What emotions would I like my player to experience? Why?*
- *What emotions are players (including me) having when they play now? Why?*
- *How can I bridge the gap between the emotions players are having and the emotions I'd like them to have?*

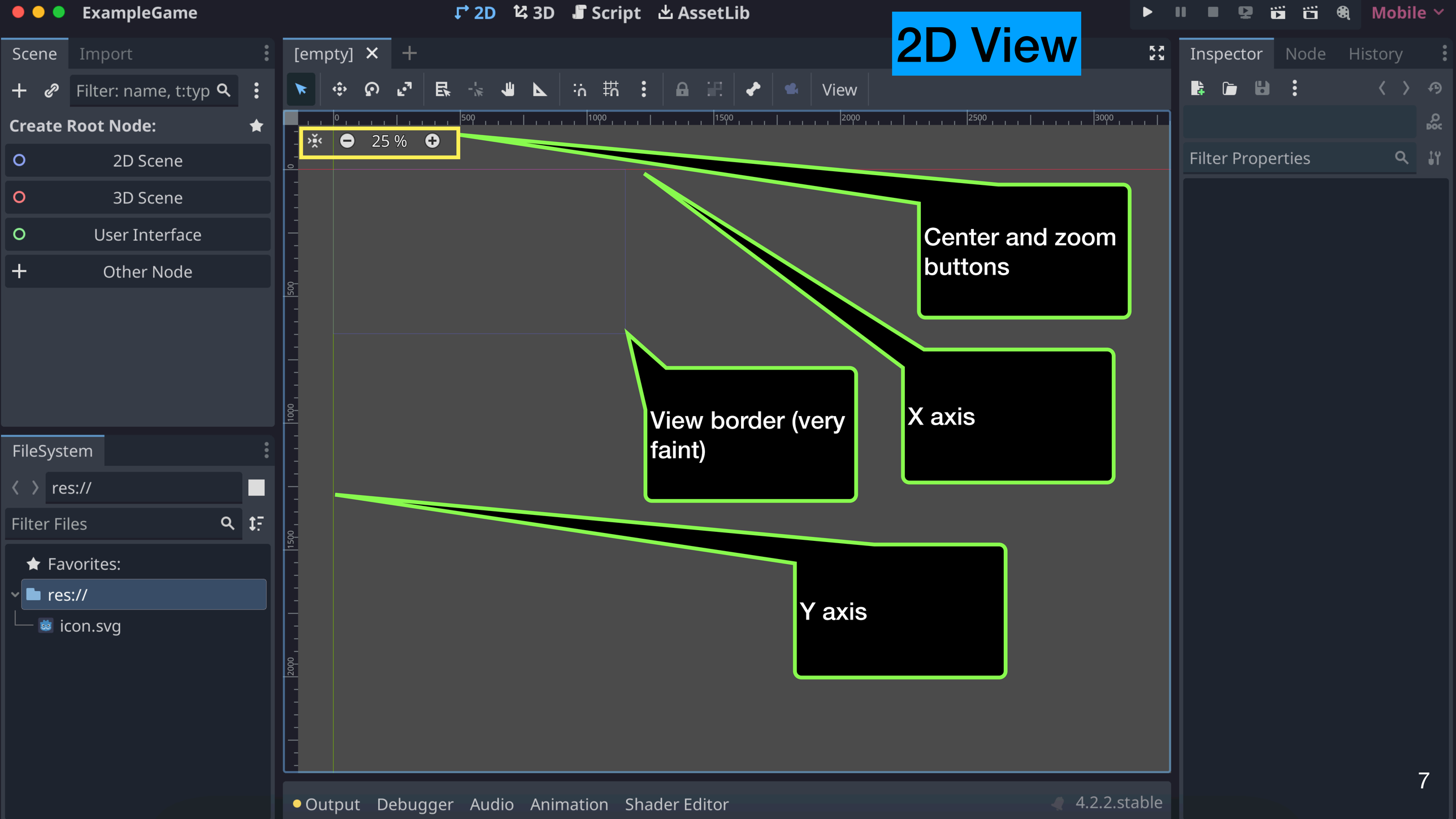
Godot Flythrough




Local Projects
Asset Library Projects
v4.2.2.stable.official [15073afe3]
[en] English
New
Edit
Run
Rename
Manage Tags
Remove
Remove Missing
Create New Project
Project Name:
Example Game
Create Folder
Project Path:
/Users/dcooper/git/CSC476/ExampleGame
Browse
The selected path is not empty. Choosing an empty folder is highly recommended.
Renderer:
Forward+
Mobile
Compatibility
• Supports desktop + mobile platforms.
• Less advanced 3D graphics.
• Less scalable for complex scenes.
• Uses RenderingDevice backend.
• Fast rendering of simple scenes.
The renderer can be changed later, but scenes may need to be adjusted.
Version Control Metadata:
Git
Cancel
Create & Edit
/Users/dcooper/CoinDashCh2
About


ExampleGame
2D
3D
Script
AssetLib
Mobile
Scene
Import
Filter: name, t:typ
Create Root Node:
2D Scene
3D Scene
User Interface
Other Node
[empty]
Transform
View
Perspective
Inspector
Node
History
Filter Properties
FileSystem
res://
Filter Files
Favorites:
res://
icon.svg
Output
Debugger
Audio
Animation
Shader Editor
4.2.2.stable
These buttons select the middle view of engine.
Scene holds Node hierarchy.
Inspector: Inspects a selected Node
Node Tab: Gives you a view to signals and groups of a selected Node
res:// is the root directory of your project. Put files in your project directory, and they will show up here.


ExampleGame
2D
3D
Script
AssetLib
Mobile
Scene
Import
Filter: name, t:typ
Create Root Node:
2D Scene
3D Scene
User Interface
Other Node
[empty]
Transform
View
Perspective
Inspector
Node
History
Filter Properties
Tabs here, show you what scene is selected.
Filters allow you to search for a node.
These buttons are for running. Left most is for running the whole game.
5th button runs the selected scene
FileSystem
res://
Filter Files
Favorites:
res://
icon.svg
Filters allow you to search for a file/resource.
Filters allow you to search for a property (there are multiple places filters show up in the inspector)
Output
Debugger
Audio
Animation
Shader Editor
4.2.2.stable


ExampleGame
2D
3D
Script
AssetLib
Mobile
2D View
Scene
Import
Filter: name, t:typ
Create Root Node:
2D Scene
3D Scene
User Interface
Other Node
[empty]
View
25 %
Inspector
Node
History
Filter Properties
Center and zoom buttons
View border (very faint)
X axis
Y axis
FileSystem
res://
Filter Files
Favorites:
res://
icon.svg
Output
Debugger
Audio
Animation
Shader Editor
4.2.2.stable

GDScript Intro

GDScript

Static vs. Dynamic Typing

```gdscript
# Everything after "#" is a comment.
# A file is a class!

# (optional) icon to show in the editor dialogs:
@icon("res://path/to/optional/icon.svg")

# (optional) class definition:
class_name Test

# Inheritance:
extends Node

# Statically typed variables.
var a := 5
var s : String = "Hello"
var arr := [1, 2, 3]
var dict := {"key": "value", 2: 3}
var other_dict :Dictionary = {key = "value", other_key = 2}
var typed_var: int
var inferred_type := "String"

# Constants.
const ANSWER = 42
const THE_NAME = "Charly"
```

GDScript

Static vs. Dynamic Typing

```
25
26  # Named Enums.
27  enum Named {THING_1, THING_2, ANOTHER_THING = -1}
28  # there are also unnamed enums, but they are not statically typed.
29
30  # Built-in vector types.
31  var  v2 : Vector2 = Vector2(1, 2)
32  var v3 := Vector3(1, 2, 3)
33
34  # Functions are first class.
35  func first_class(x: int, y: int) -> int:
36      return x * y
37
38  func g(f:Callable) -> int :
39      var x : int = f.call(4,5)
40      return x
41
42  func _ready() -> void:
43      var y = g(first_class)
44      print(y)
45
46
```

GDScript

Static vs. Dynamic Typing

```gdscript
# more Functions.
func some_function(param1: int, param2: int, param3:int) -> int:
    const local_const := 5
    if param1 < local_const:
        print(param1)
    elif param2 > 5:
        print(param2)
    else:
        print("Fail!")

    for i in range(20):
        print(i)

    while param2 != 0:
        param2 -= 1

    match param3:
        3:
            print("param3 is 3!")
        _:
            print("param3 is not 3!")

    var local_var := param1 + 3
    return local_var
```

GDScript

Static vs. Dynamic Typing

```
74  # Functions override functions with the same name on the base/super class.
75  # If you still want to call them, use "super":
76  func something(p1, p2):
77      #super(p1, p2)
78      pass
79
80
81  # It's also possible to call another function in the super class:
82  func other_something(p1, p2):
83      #super.something(p1, p2)
84      pass
85
86
87  # Inner class
88  class Something:
89      var a := 10
90
91
92  # Constructor
93  func _init():
94      print("Constructed!")
95      var lv = Something.new()
96
97      print(lv.a)
```

GDScript

Similarities to Python

- Indentation has meaning
- static typing in GDScript is an enforced version of type hints in python 3
- self is a keyword in GDScript instead of a parameter

GDScript

Differences from Python

- func instead of def for functions
- variable declaration in class instead of variable assignment in _init()
- self is a keyword in GDScript instead of a parameter
- GDScript has specific keywords related to Godot and the Godot editor
- Every file is a class in Godot
- Here's a GDScript tutor system: https://gdquest.github.io/learn-gdscript/

Player Scene

Fly through (then walk at your pace)

Viewport

Display Window

- Set to 480 wide by 720 high

Vectors

You can also think of the `(4, 3)` position as an *offset* from the `(0, 0)` point, or *origin*. Imagine an arrow pointing from the origin to the point:

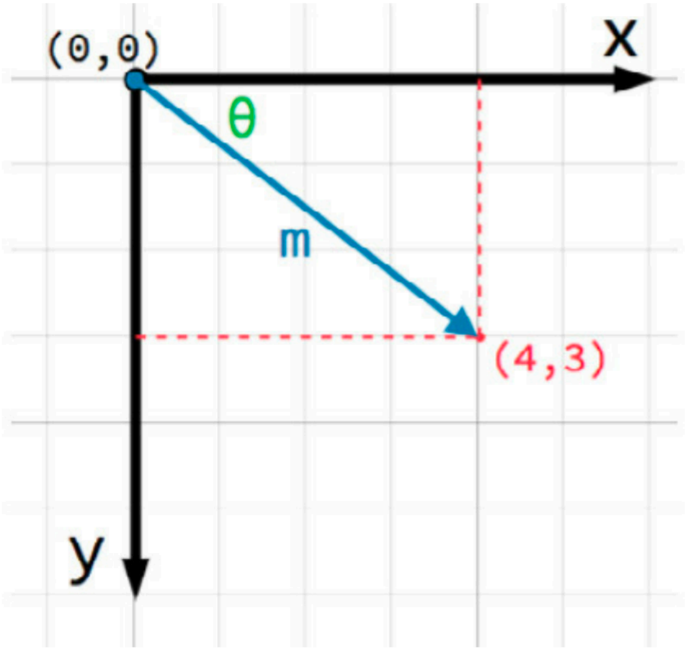


Figure 2.6: A 2D vector

First Scene

Player.tscn

- \+ adds a new node
  - Select Area2D
  - Rename to Player
  - Save
  - Group Selected nodes

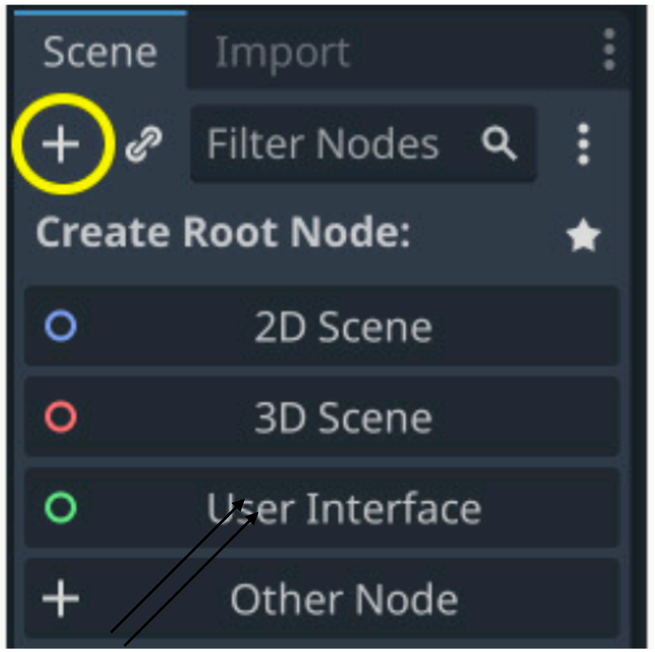
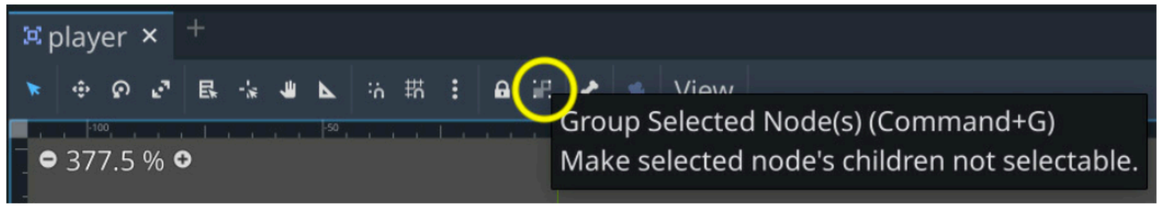
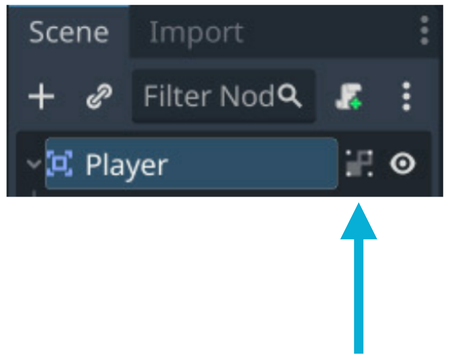


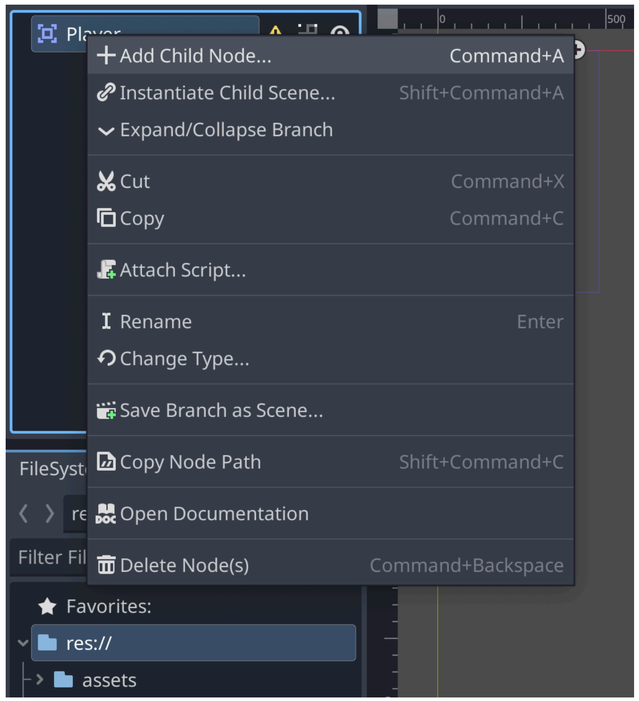
Figure 2.7: Adding a node

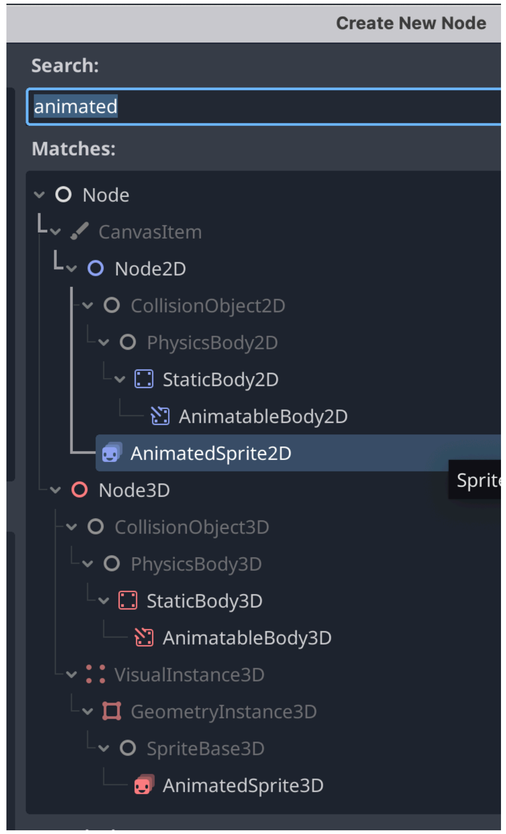


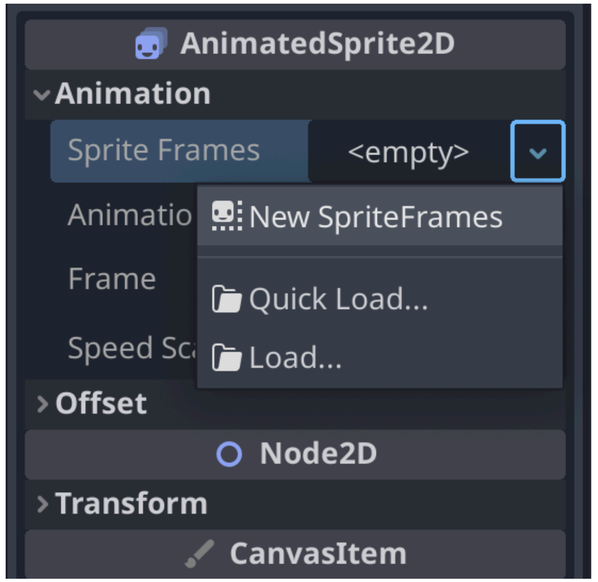


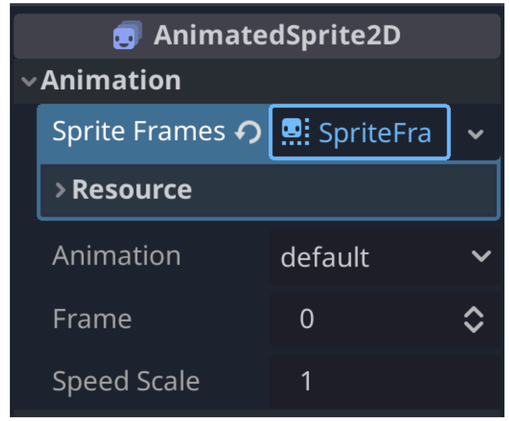
AnimatedSprite2D

Part of player









Setup Animations

Copy assets from FileSystem

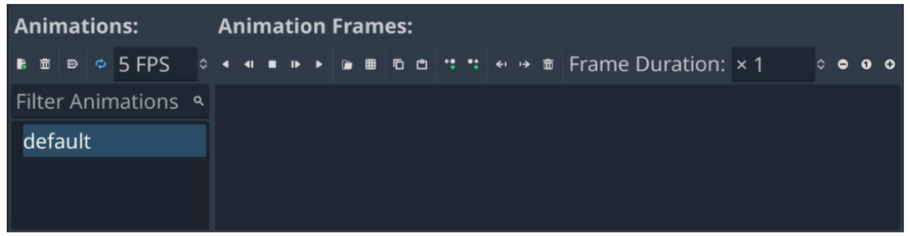
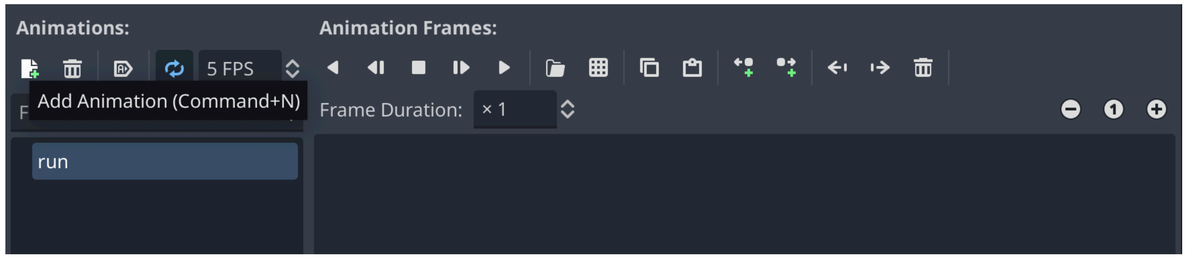
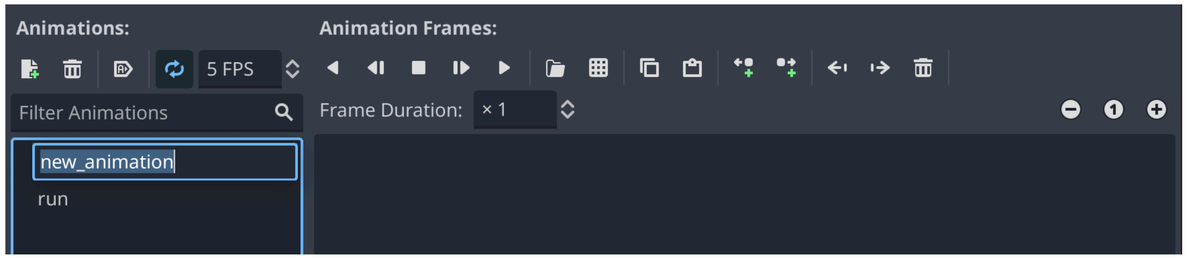
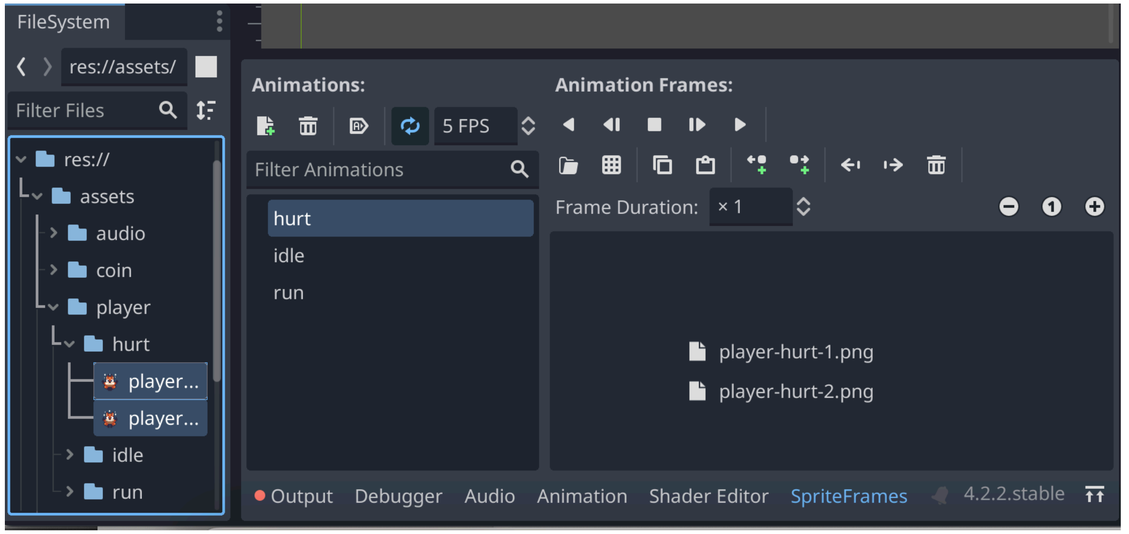
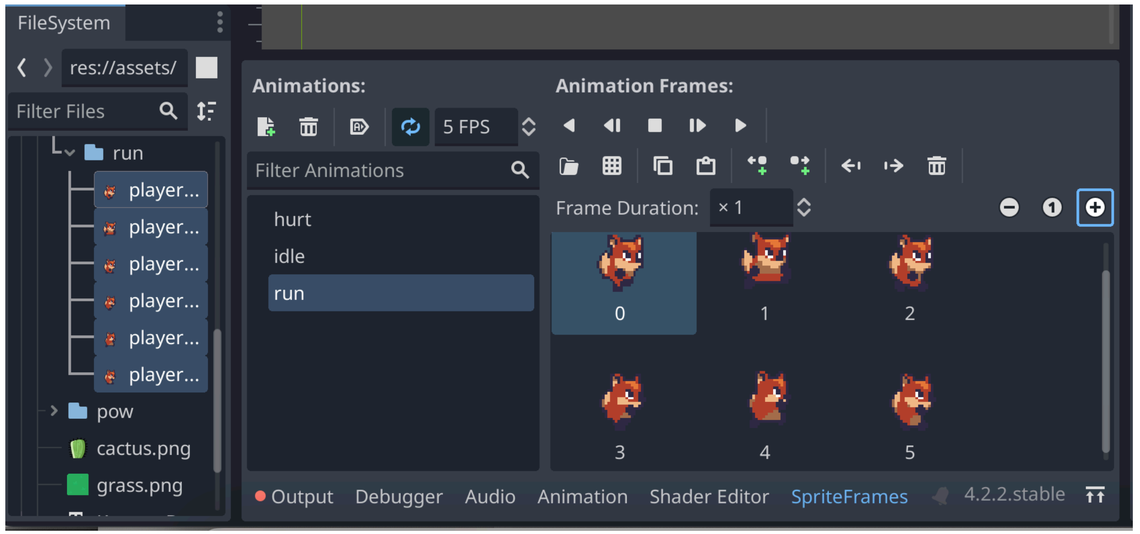


Figure 2.11: The SpriteFrames panel





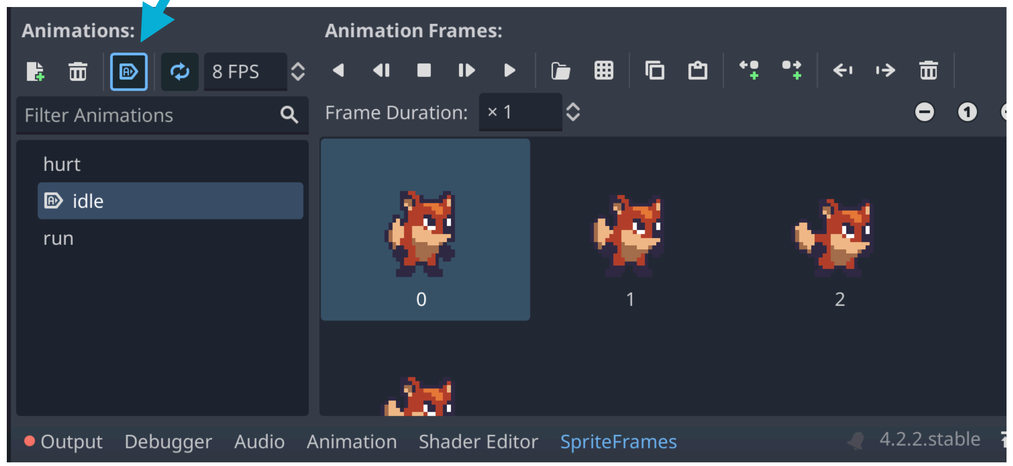




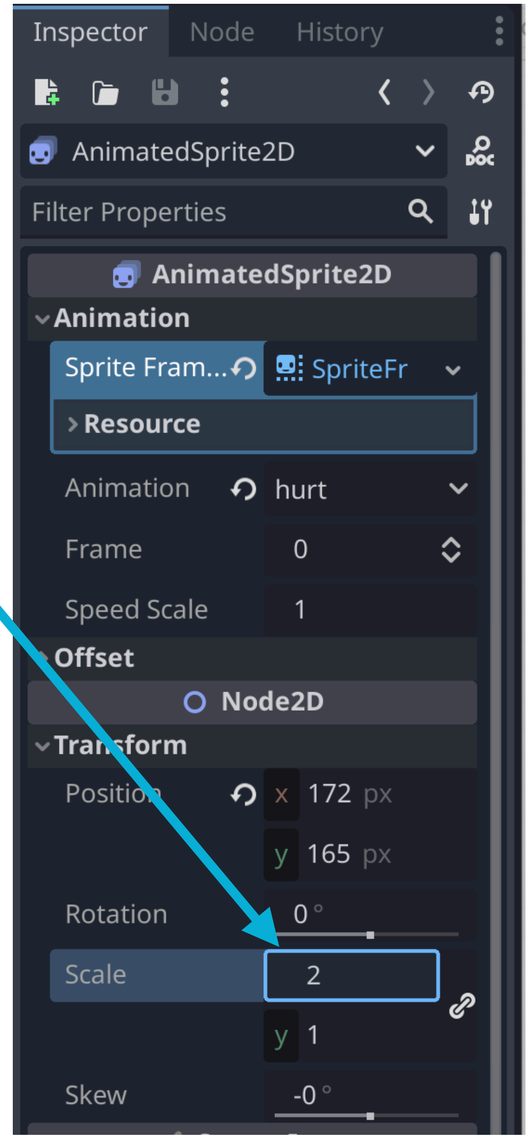
Extra Changes

Animated Sprite2D

Autoplay on Load

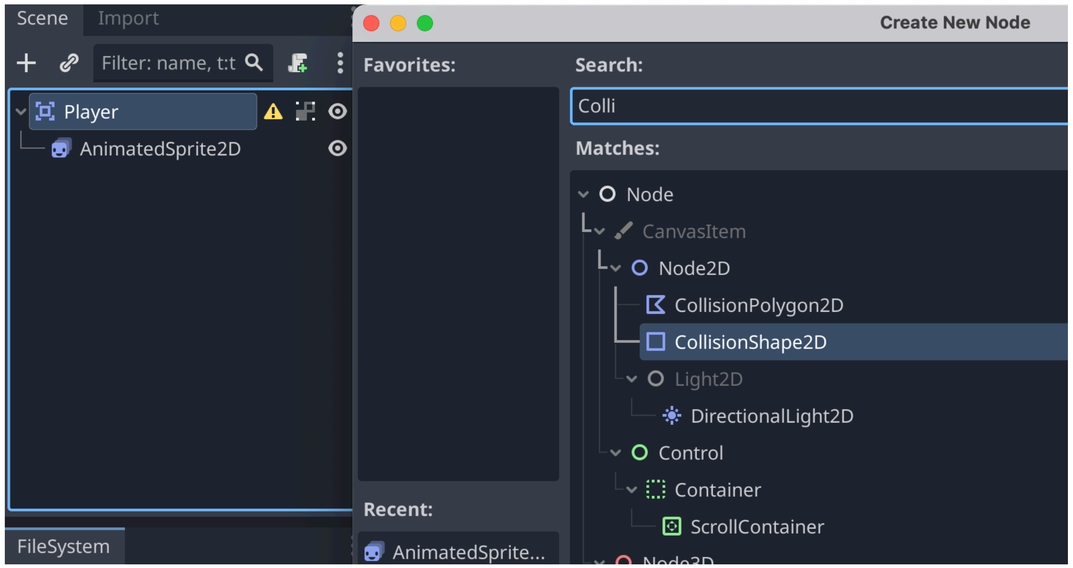


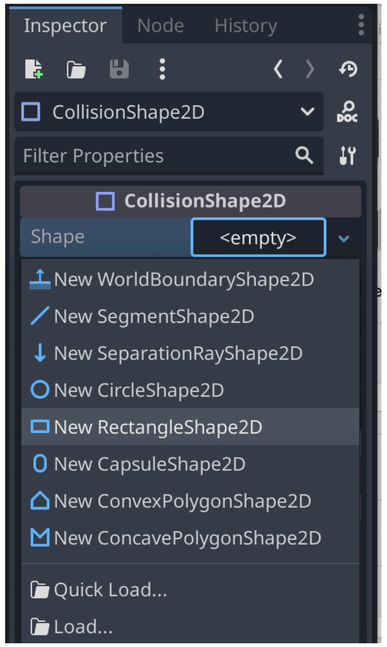
Scale up to 2



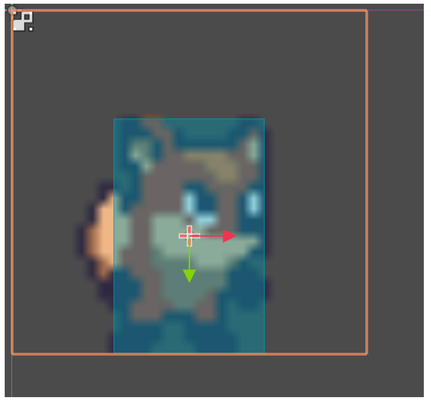
Collision Shape

Child node of Player

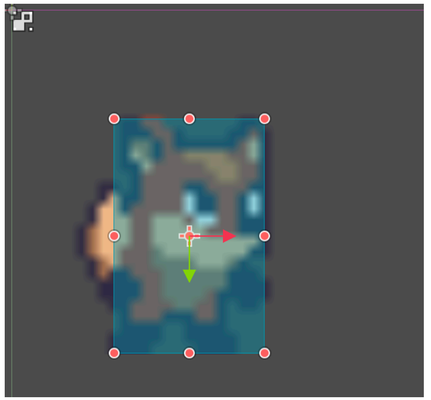




Animated Sprite Shape

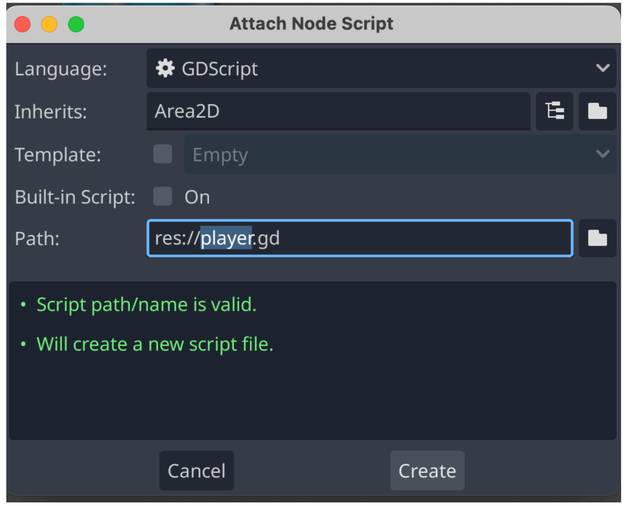


Collision Shape



Player Script

player.gd - basic motion

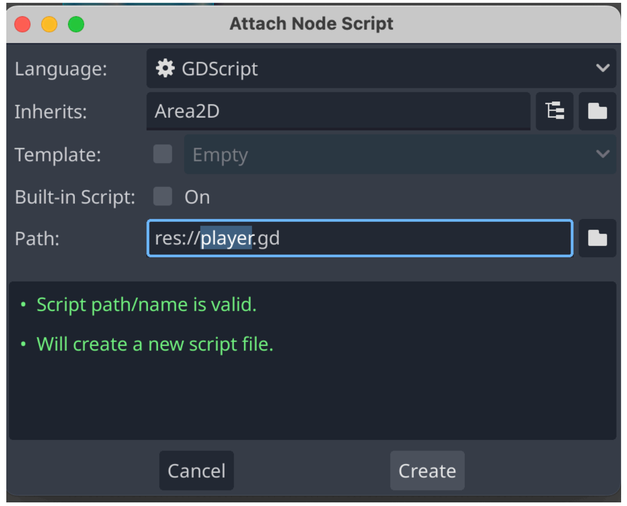



```gdscript
extends Area2D
@export var speed = 350
var velocity = Vector2.ZERO
var screensize = Vector2(480,720)

func _process(delta: float):
    velocity = Input.get_vector("ui_left","ui_right",
                                "ui_up","ui_down")
    position += velocity * speed * delta
```

Player Script

player.gd - select animations




```gdscript
func _process(delta: float):
    velocity = Input.get_vector("ui_left","ui_right",
                                "ui_up","ui_down")
    position += velocity * speed * delta

    if velocity.length() > 0:
        $AnimatedSprite2D.animation = "run"
    else:
        $AnimatedSprite2D.animation = "idle"
    if velocity.x != 0:
        $AnimatedSprite2D.flip_h = velocity.x < 0
```

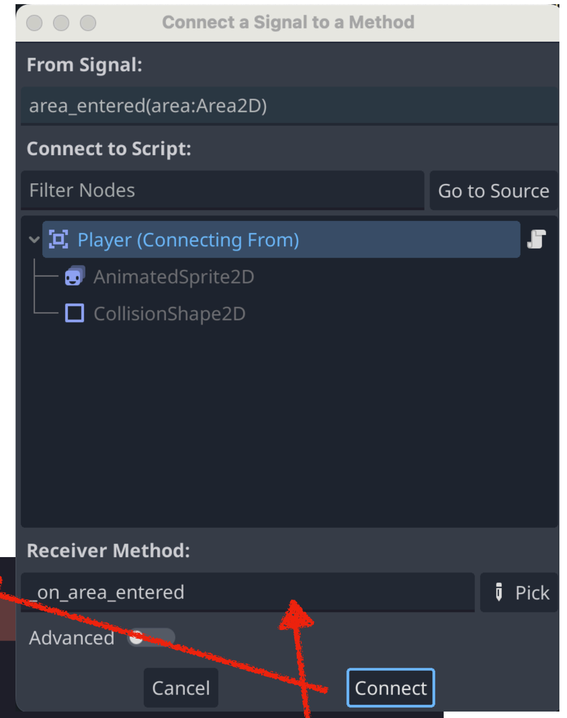
Player Script

player.gd - begin and end -> collision signals

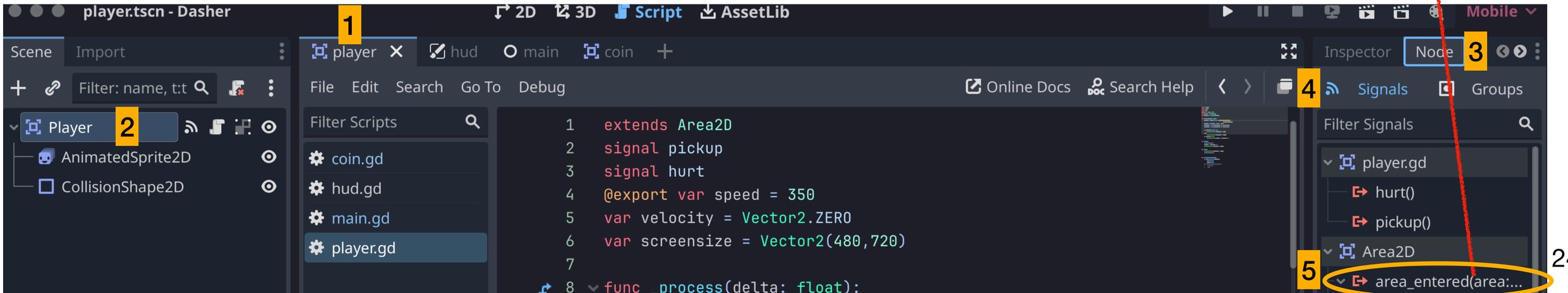
```gdscript
func start():
    set_process(true)
    position = screensize / 2
    $AnimatedSprite2D.animation = "idle"

fun die():
    $AnimatedSprite2D.animation = "hurt"
    set_process(false)
```

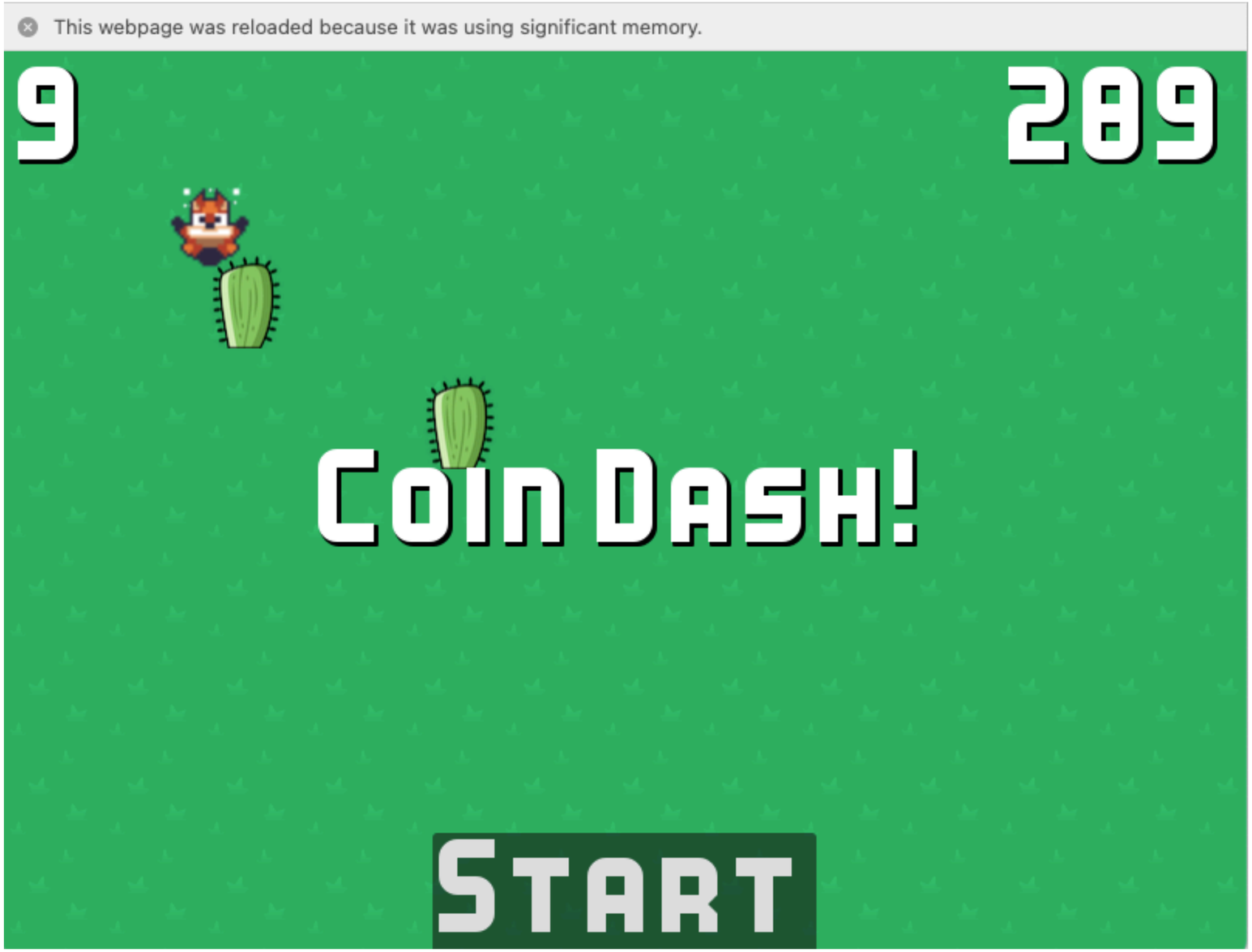
```gdscript
func _on_area_entered(area):
    if area.is_in_group(coins):
        area.pickup()
        pickup.emit()
    if area.is_in_group(obstacles):
        hurt.emit()
        die()
```



Why are these boxes red?



Exporting


This webpage was reloaded because it was using significant memory.
9
289
Coin Dash!
START

Exporting to Web

In Godot

1. Make sure the player scene is selected
2. Click the play button
3. When it asks if it should use the current node as default say yes.
4. Go to the export window and click the web preset
5. Click the export project… button
6. Make a new folder called player_scene to save to
7. Unclick export with debug
8. Click save

Exporting to Web

In your File System

1. Go to your workbench repository
2. Copy the player_scene folder to your Game Demo repository
3. Copy the coi-serviceworker.js file from the helper files directory to the player_scene directory
4. Copy the CoinDash.html file to index.html

Exporting to Web

Last Publishing steps

1. Edit the index.html file to add the line
   <script src="coi-serviceworker.js"></script>
   right before the </head> closing tag
2. Save the file, commit to your game demo repository and push
3. Wait for the push to be published
4. Play your game at the player_scene directory of your game demo page
5. update your README.md file to link to your player scene