Sign in here!

DevLUp FSU GBM #3

Intro to 3D Game Dev

September 19th, 2024

#👀showoff

Our Meeting Schedule:

Date	Week #	GBM Title	Secondary Event	Presenter
29 Aug	1	(No Meeting)	Involvement Fair	
5 Sep	2	Intro to Club and New Club Project		Club
12 Sep	3	Intro to Game Design		Chris
19 Sep	4	Intro to 3D Game Dev in Godot		Dion
26 Sep	5	Intro to 3D Modelling in Blender		Jake, Parker, Emma
3 Oct	6	Blender Animations		Ares
10 Oct	7	Blender Materials		Parker, Jake
17 Oct	8	Pixel Art		Ares, Emma
24 Oct	9	Tile Maps		Jake, Ares
31 Oct	10	Spooky Game Night Social	CANDY FOR ALL (No Candy)	Jack Skellington
7 Nov	11	Writing for Games		Emma, Chris
14 Nov	12	UI Design		Emma, Jake
21 Nov	13	3D Math		Dion
28 Nov	14	Thanksgiving Break		
5 Dec	15	Goodbye Chris Social		Chris
12 Dec	16	Finals		

New Club Project

- With the Seminole Innovators

- A Mario Party-like game
- Make your own minigames
- Made in Godot

- No experience is required
- Need programmers, artists, musicians, designers, etc.

- Meeting at 6pm!!!

DevLUp War Games - Game Jam

- DevLUp-wide Game Jam
  - FSU, UF, FIT, FAU, and more

- November 1st — 3rd
  - 48 hours
  - Starts Friday afternoon/evening
  - Ends Sunday afternoon/evening

- We will likely have access to the Innovation Hub (still needs to be reserved)

Next Week's Workshop: Intro to 3D Modelling

You will learn to make your very own 3D model in blender!

- Blender basics
- Spooky bones! What's that? You'll find out!
- A look at Animating your models
- Implementing your model into Godot

GBM 3: Intro to 3D Game Dev in Godot

Making a First-person Shooter

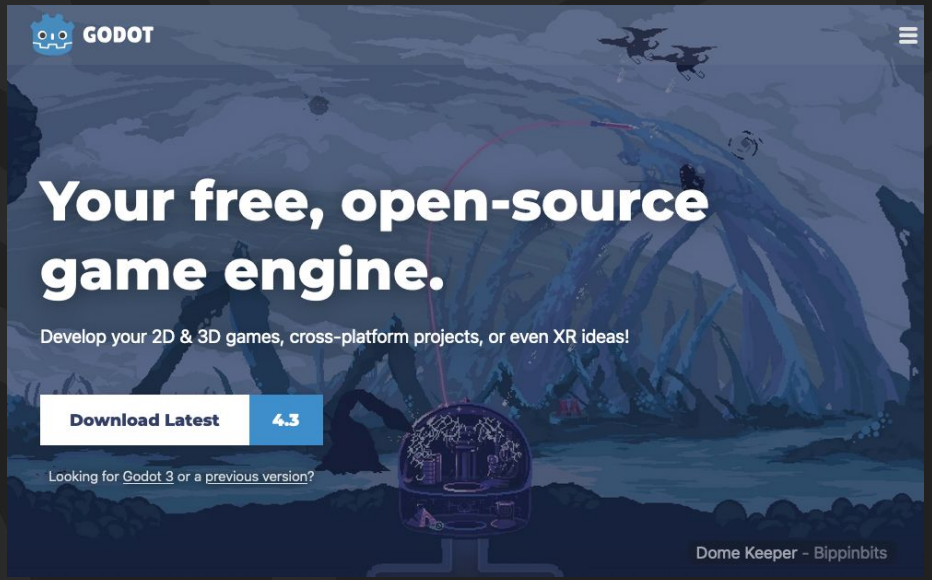
Lesson Plan

- Download Godot
- What are Nodes?
- Creating the Project
- The Player
- Shooting
- Simple Enemy
- Extras (if we have time)

Download Godot

https://godotengine.org/

Download Latest (v4.3)



Download Assets and Copy Paste Code

https://fsu.devlup.org

Downloads

- Godot
- assets.zip

Code to Copy and Paste

- Google CoLab

What are Nodes?

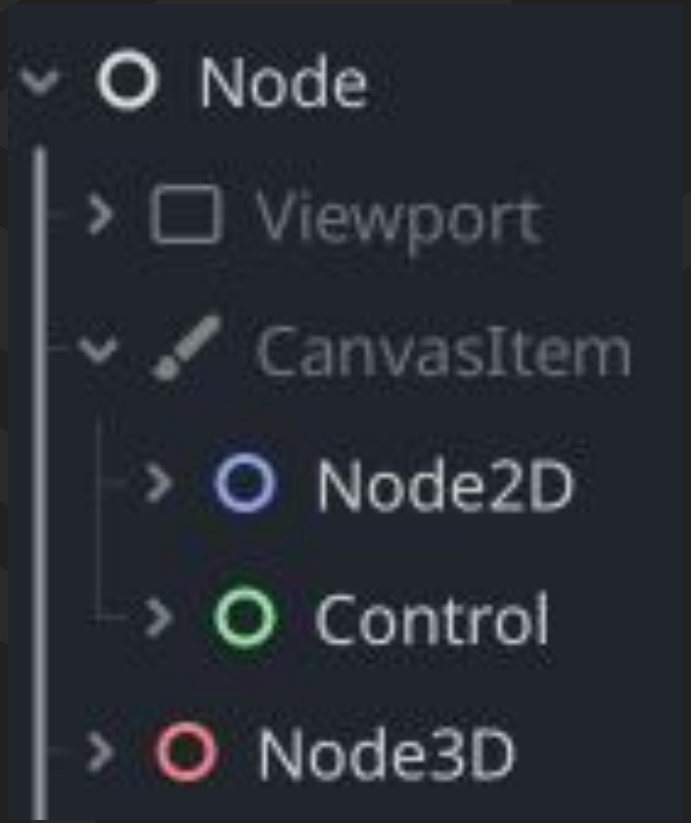
What are Nodes?

A “Node” is the basic building block of everything in Godot.

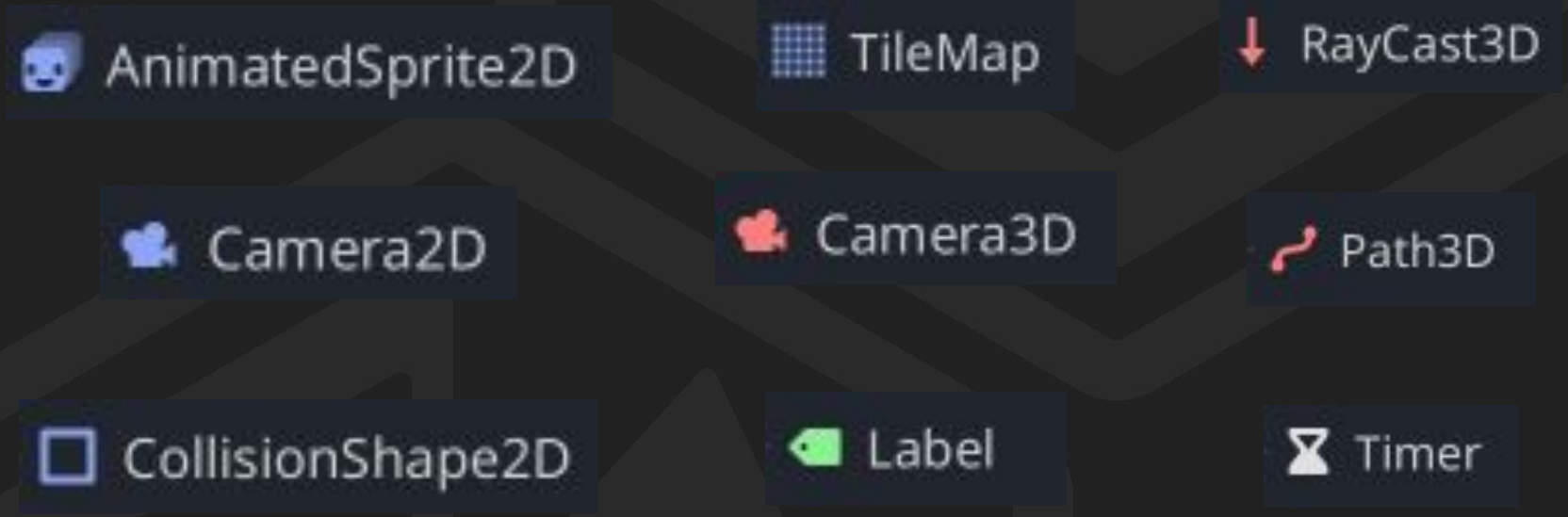
Nodes provide you with features built-in to the engine (e.g., sprites, animations, timers, physics bodies, UI, audio, tile maps, etc.)

Most Nodes “inherit” from either

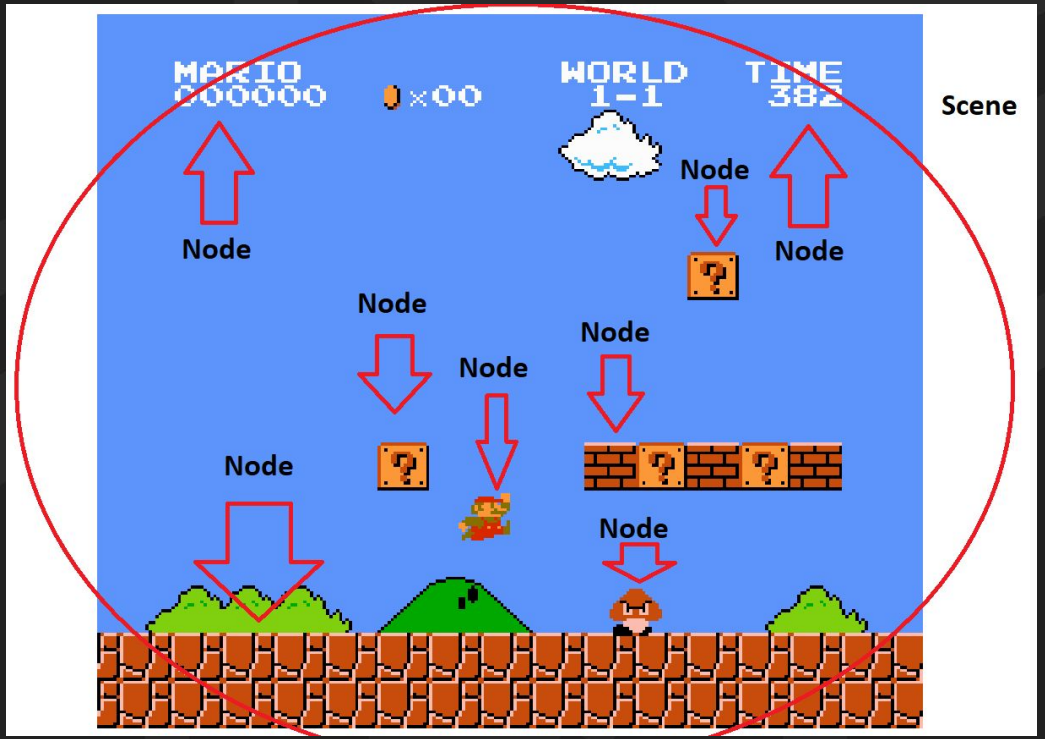
- Node2D (Nodes with a 2D position)
- Node3D (Nodes with a 3D position)
- Control (Nodes with an “anchor” position)



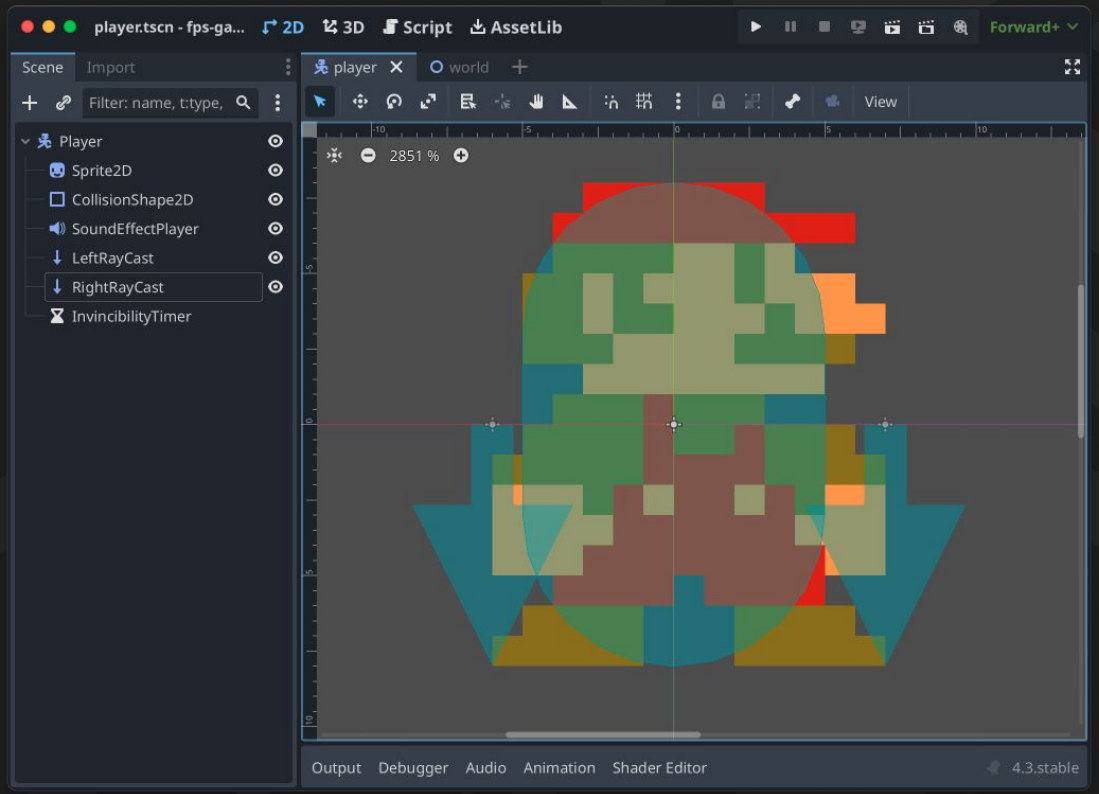
Some Examples of Nodes



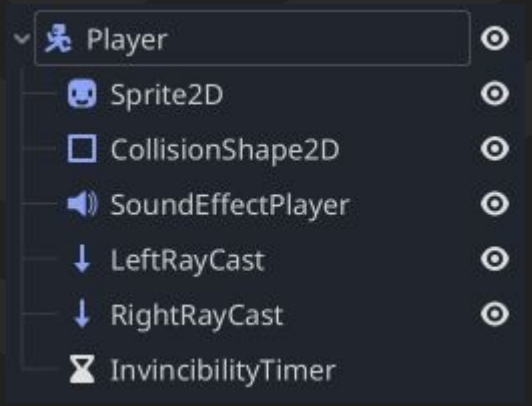
Everything is Made of Nodes!



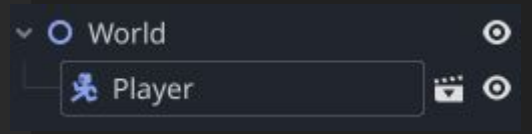
Make Your own Nodes: Scenes



Scenes are made up of nodes!



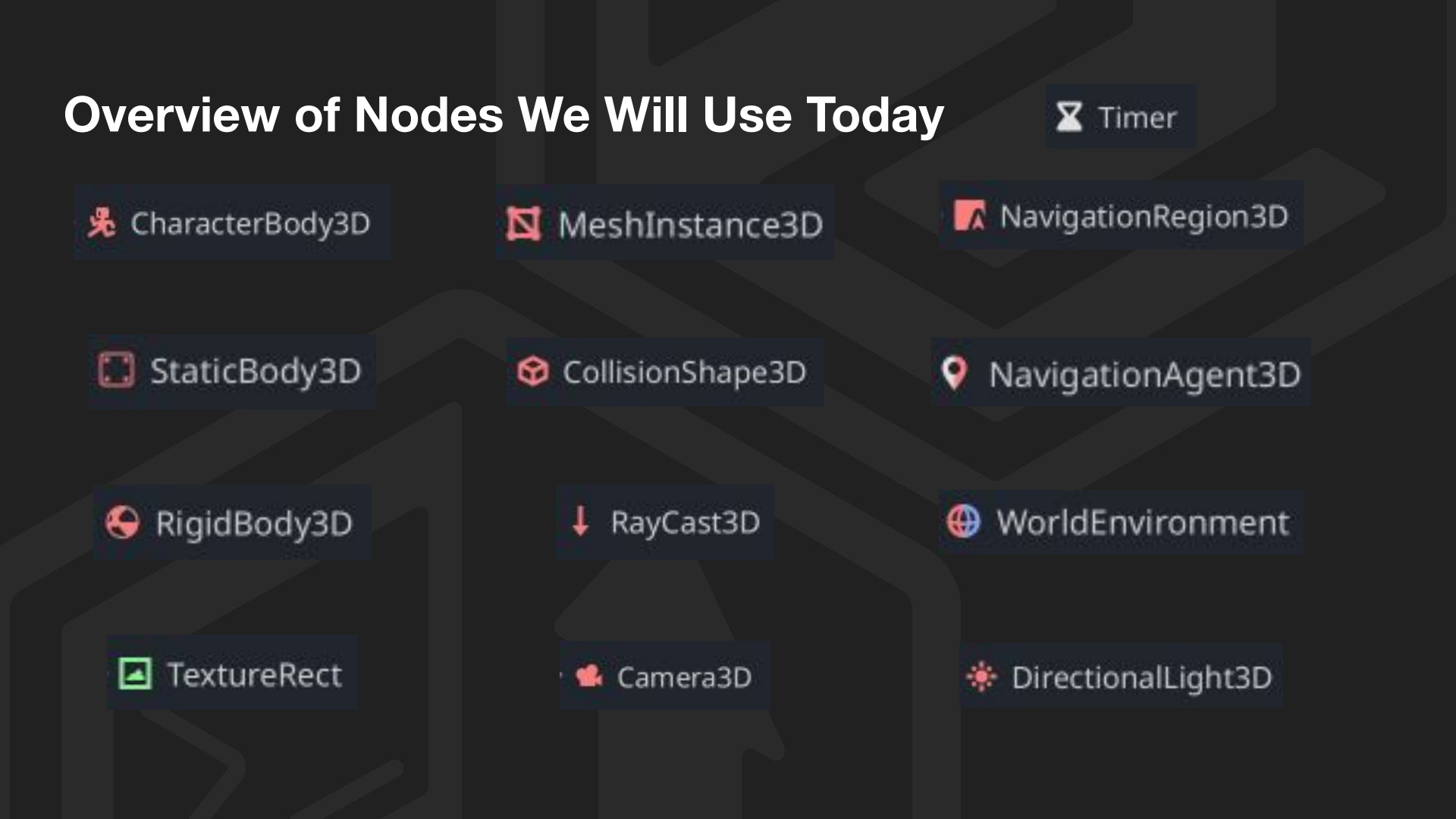
Scenes *are* nodes!



Make Your own Nodes: Scripts

```
1   class_name Player
2   extends CharacterBody2D
3
4
5   # Called when the node enters the scene tree for the first time.
6   func _ready() -> void:
7       pass # Replace with function body.
8
9
10  # Called every frame. 'delta' is the elapsed time since the previous frame.
11  func _process(delta: float) -> void:
12      pass
```

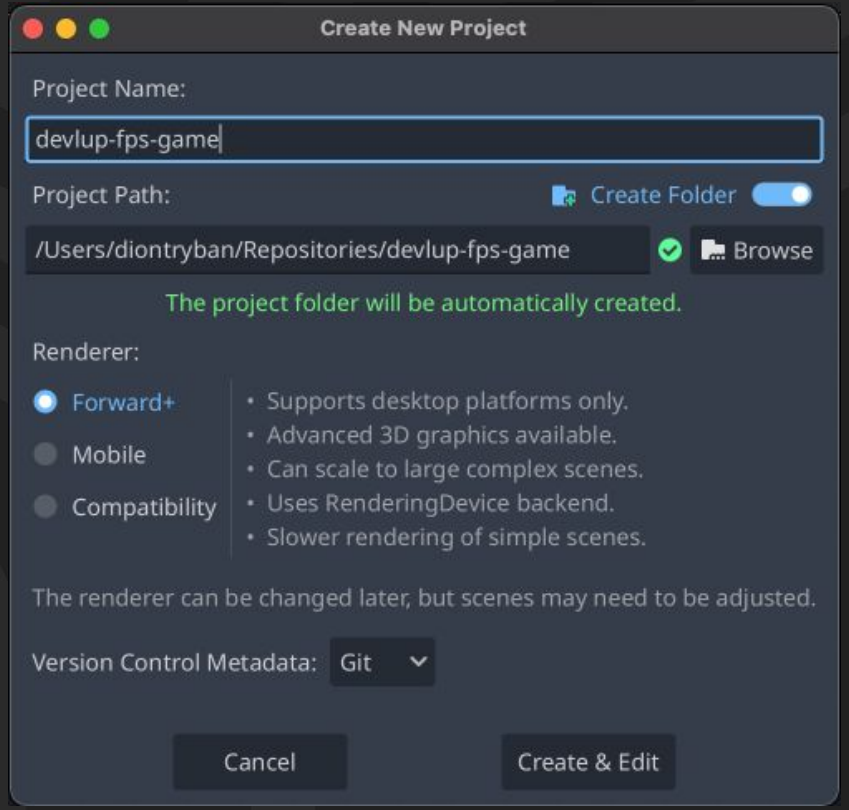
Overview of Nodes We Will Use Today



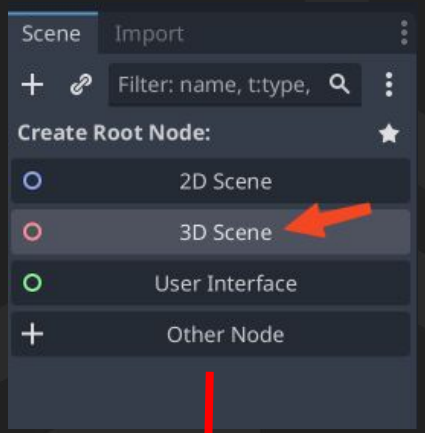
What Nodes Could Describe This Scene?

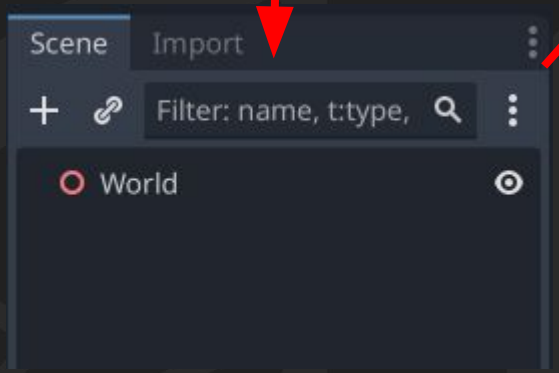
Creating the Project

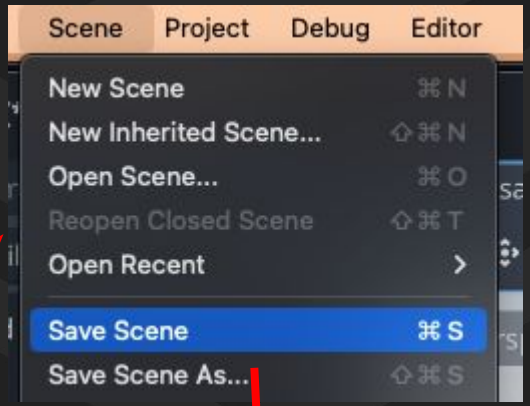
Create New Project

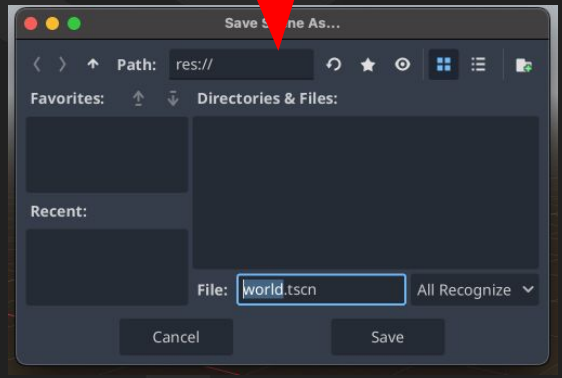


Create the “World” Scene

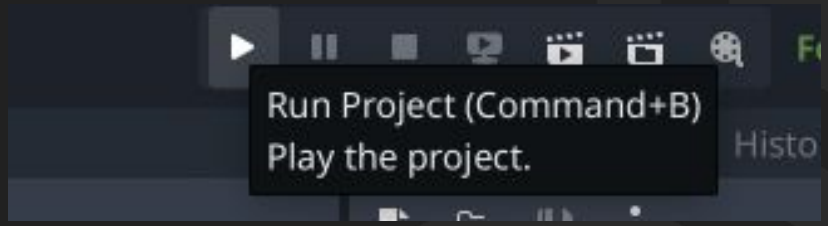








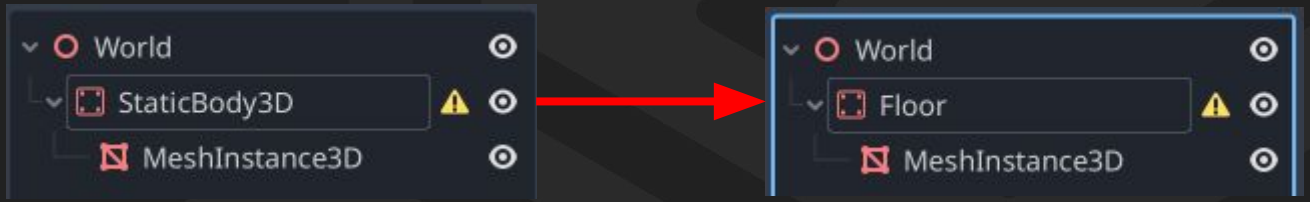
Run the Game!

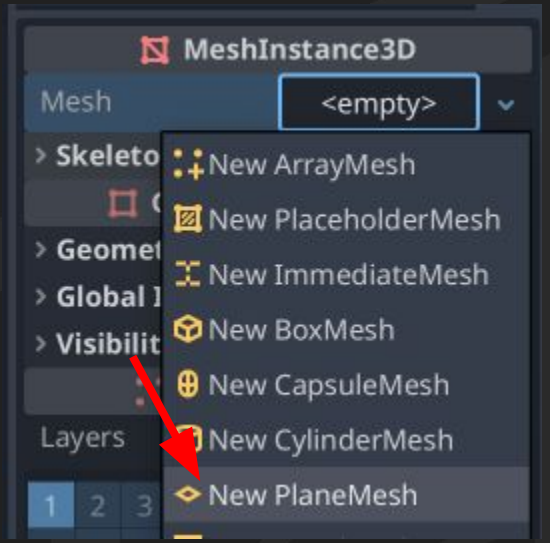

Run Project (Command+B)
Play the project.

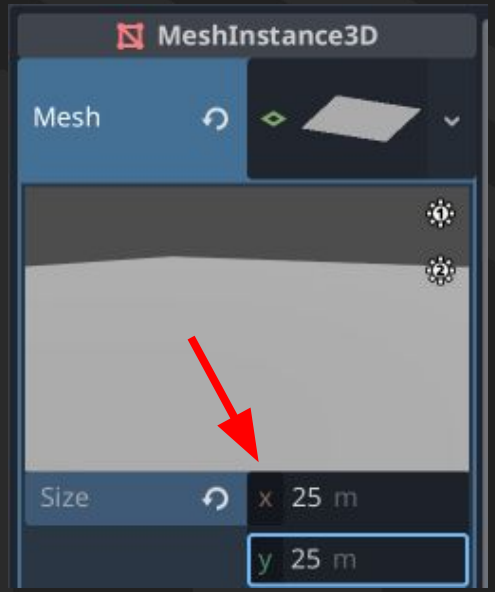

Please Confirm...
No main scene has ever been defined, select one?
You can change it later in "Project Settings" under the 'application' category.
Cancel
Select
Select Current

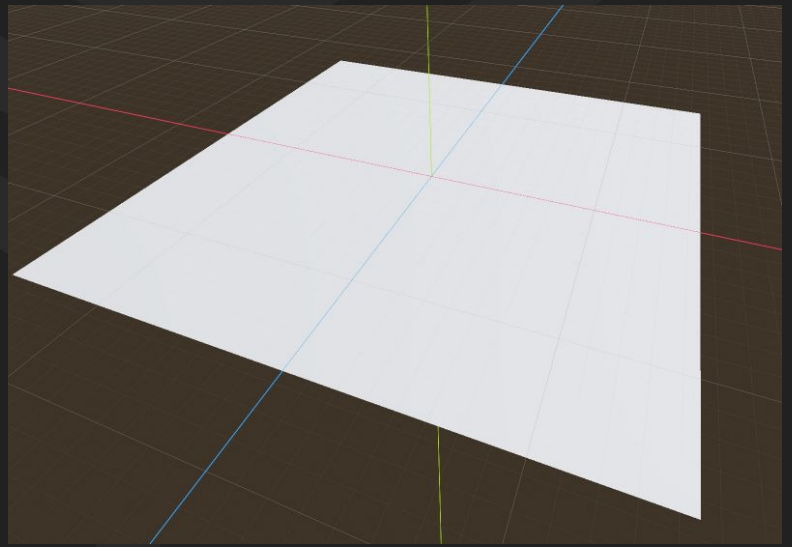

devlup-fps-game (DEBUG)

Create the Floor Mesh

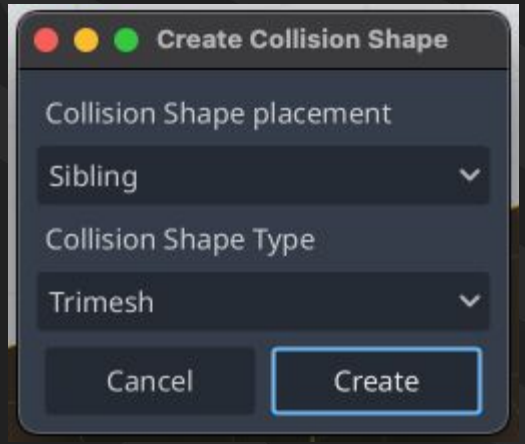






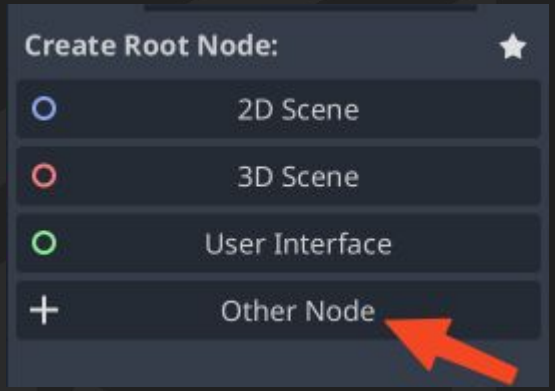


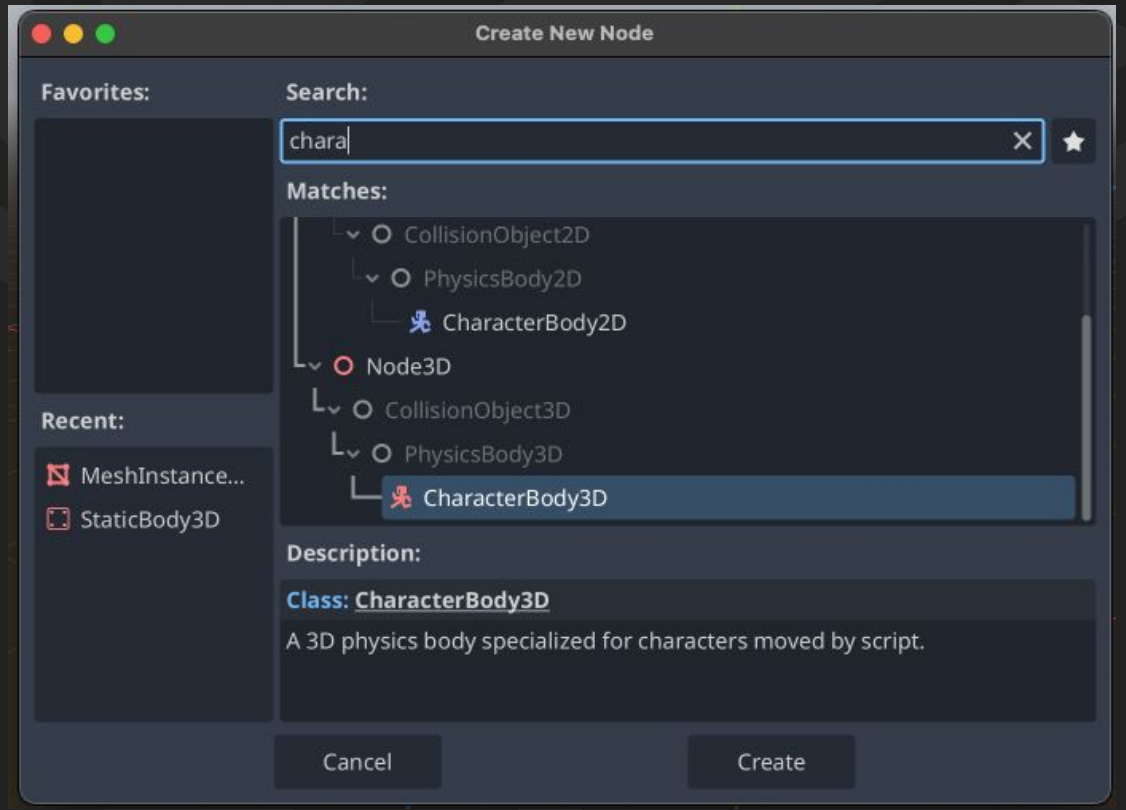
Make the Floor have “Collision”



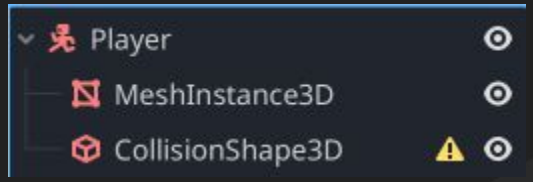
The Player

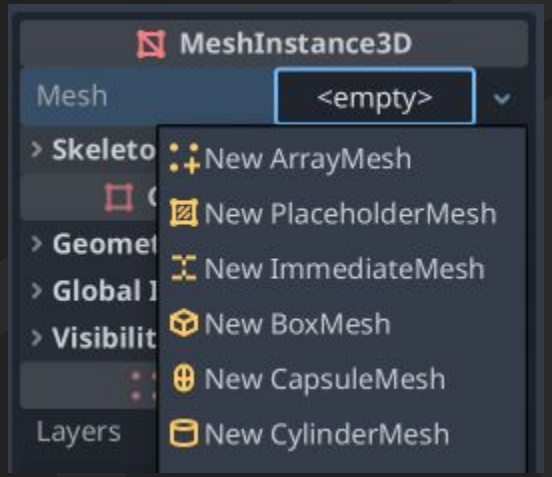
Make the Player Scene

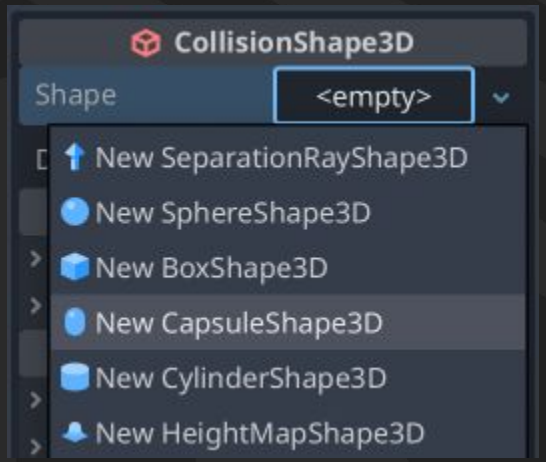


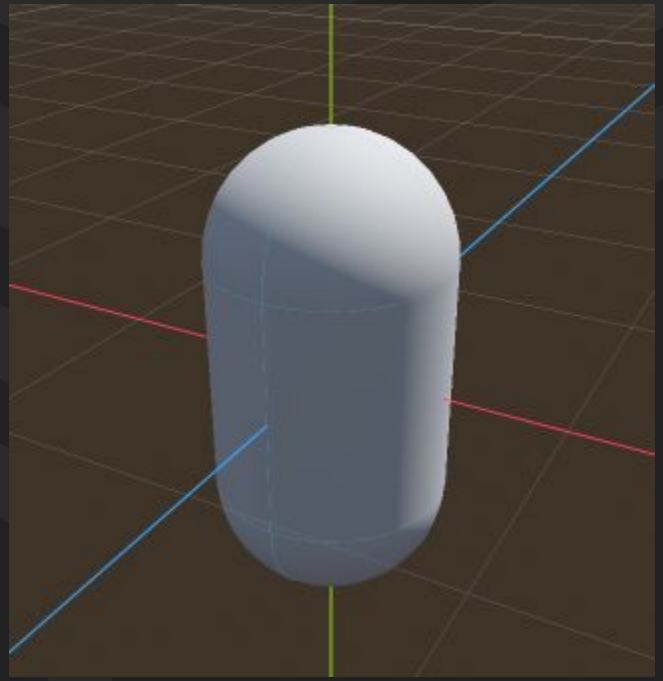


Make the Player Scene



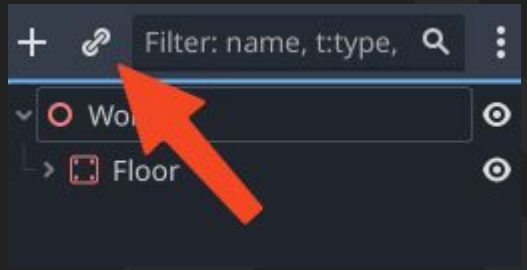


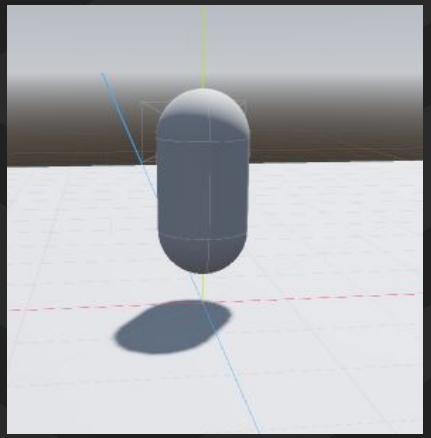


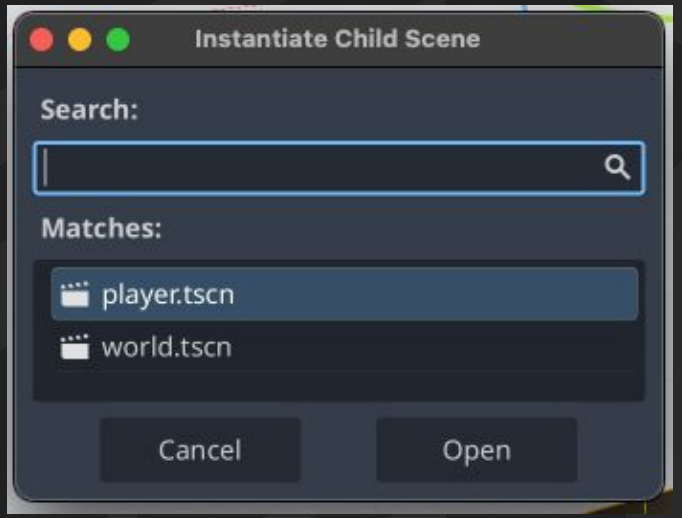


Add a Camera to the Player

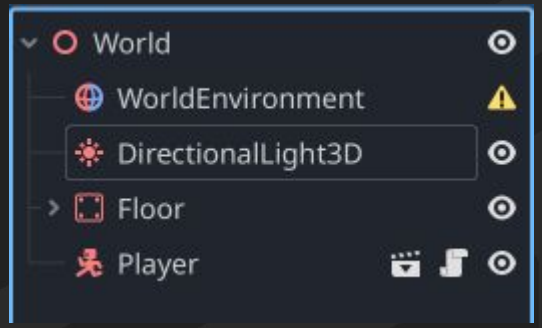
Add Player to Main Scene

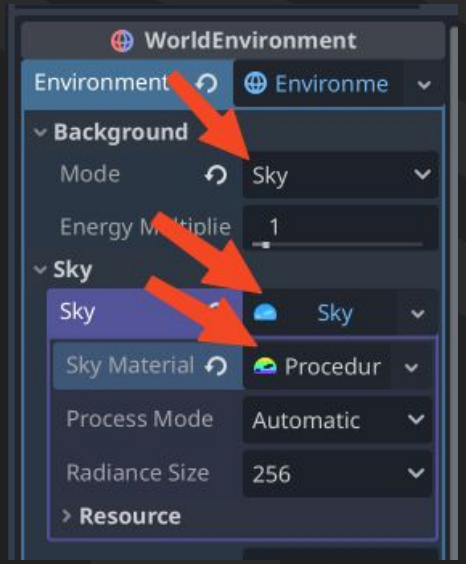


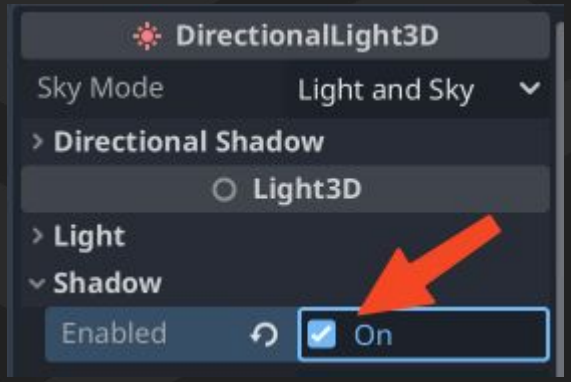


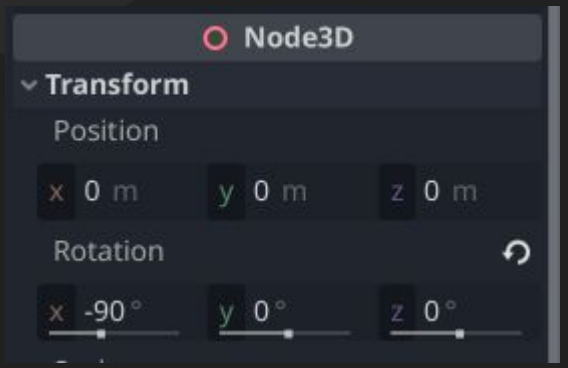


Let there be light! (Add Environment to World)

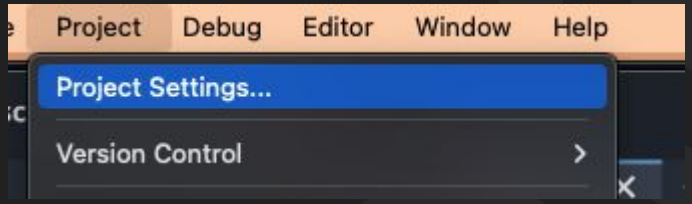




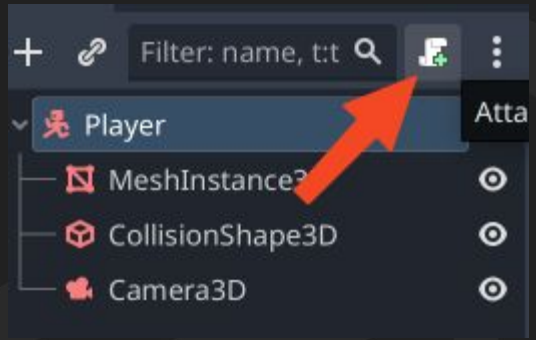




Add Input Mappings



Add Script to Player

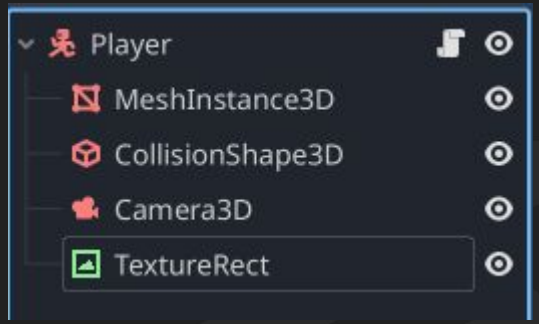


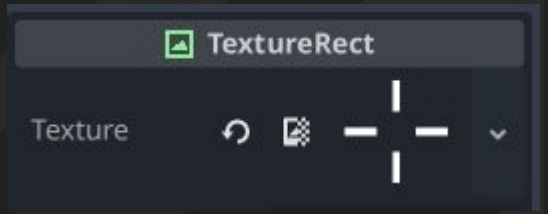
player.gd

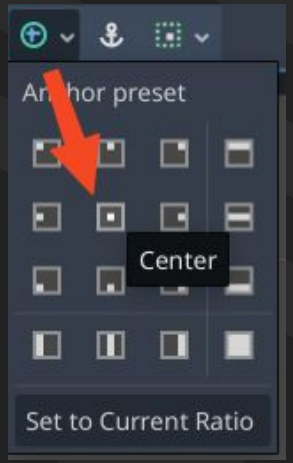
```gdscript
extends CharacterBody3D


const SPEED = 5.0
const JUMP_VELOCITY = 4.5
const MOUSE_SENSITIVITY := 0.1


func _ready() -> void:
    # Capture the mouse when the game starts.
    Input.mouse_mode = Input.MOUSE_MODE_CAPTURED


func _process(_delta: float) -> void:
    # Toggle whether the mouse is captured when "pause" is pressed.
    if Input.is_action_just_pressed("pause"):
        if Input.mouse_mode == Input.MOUSE_MODE_CAPTURED:
            Input.mouse_mode = Input.MOUSE_MODE_VISIBLE
        else:
            Input.mouse_mode = Input.MOUSE_MODE_CAPTURED


func _input(event: InputEvent) -> void:
    if Input.mouse_mode == Input.MOUSE_MODE_CAPTURED:
        if event is InputEventMouseMotion:
            rotation_degrees.y -= event.relative.x * MOUSE_SENSITIVITY
            rotation_degrees.x -= event.relative.y * MOUSE_SENSITIVITY
            rotation_degrees.x = clamp(rotation_degrees.x, -90.0, 90.0)


func _physics_process(delta: float) -> void:
    # Add the gravity.
    if not is_on_floor():
        velocity += get_gravity() * delta

    # Handle jump.
    if Input.is_action_just_pressed("player_jump") and is_on_floor():
        velocity.y = JUMP_VELOCITY

    # Get the input direction and handle the movement/deceleration.
    var input_dir := Input.get_vector("player_left", "player_right", "player_forward", "player_back")
    var direction := (transform.basis * Vector3(input_dir.x, 0, input_dir.y)).normalized()
    if direction:
        velocity.x = direction.x * SPEED
        velocity.z = direction.z * SPEED
    else:
        velocity.x = move_toward(velocity.x, 0, SPEED)
        velocity.z = move_toward(velocity.z, 0, SPEED)

    move_and_slide()

```

Shooting

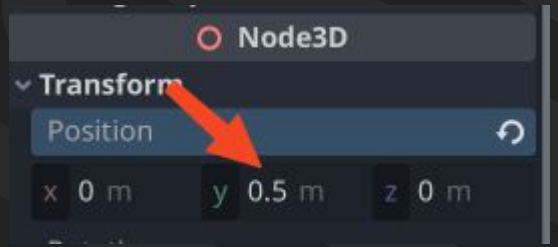
Add Crosshair

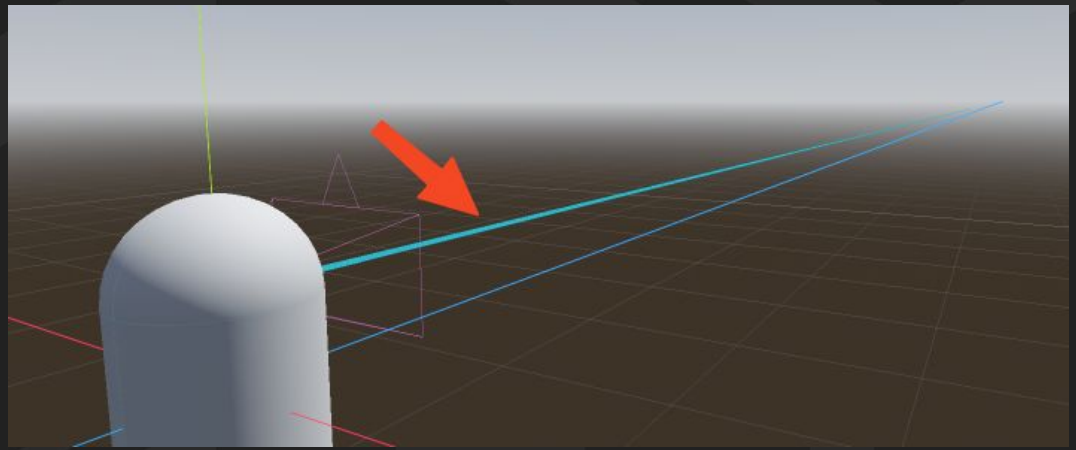






Add RayCast for “HitScan” Weapon



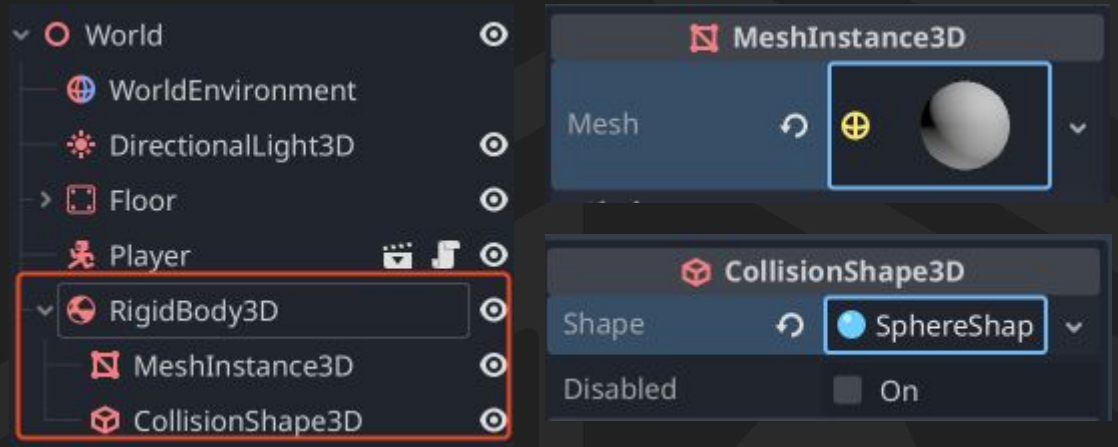


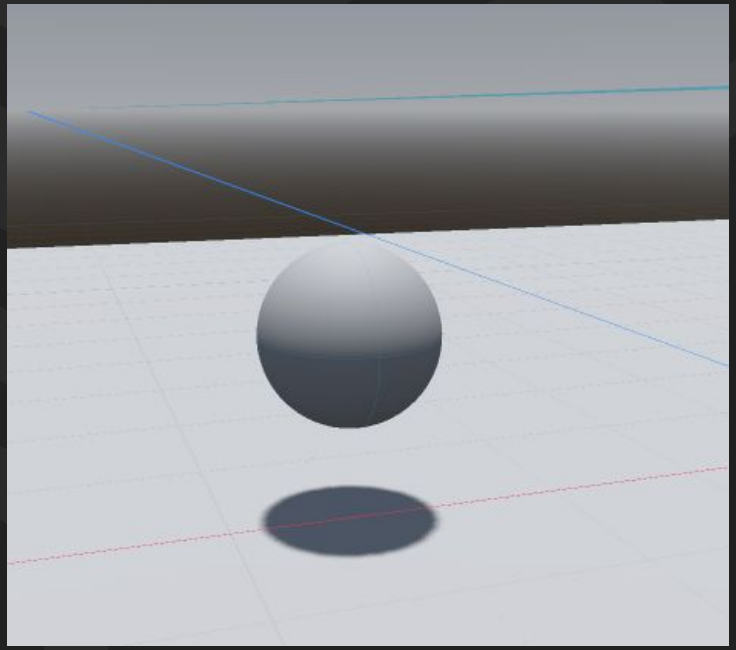
Add Shooting Code to Player Script

player.gd

```gdscript
extends CharacterBody3D


const SPEED = 5.0
const JUMP_VELOCITY = 4.5
const MOUSE_SENSITIVITY := 0.1


# Get the "RayCast3D" node that is a child of the player node.
@onready var ray_cast: RayCast3D = $RayCast3D


func _ready() -> void:
    # Capture the mouse when the game starts.
    Input.mouse_mode = Input.MOUSE_MODE_CAPTURED


func _process(_delta: float) -> void:
    # Toggle whether the mouse is captured when "pause" is pressed.
    if Input.is_action_just_pressed("pause"):
        if Input.mouse_mode == Input.MOUSE_MODE_CAPTURED:
            Input.mouse_mode = Input.MOUSE_MODE_VISIBLE
        else:
            Input.mouse_mode = Input.MOUSE_MODE_CAPTURED

    # Handle shooting using the RayCast3D to determine what we hit.
    if Input.is_action_just_pressed("shoot"):
        var collider = ray_cast.get_collider()
        if collider is RigidBody3D:
            print("Hit RigidBody3D")
            collider.apply_impulse((transform.basis * Vector3(0, 0, -1)).normalized() * 3)
        else:
            print("Missed")
```

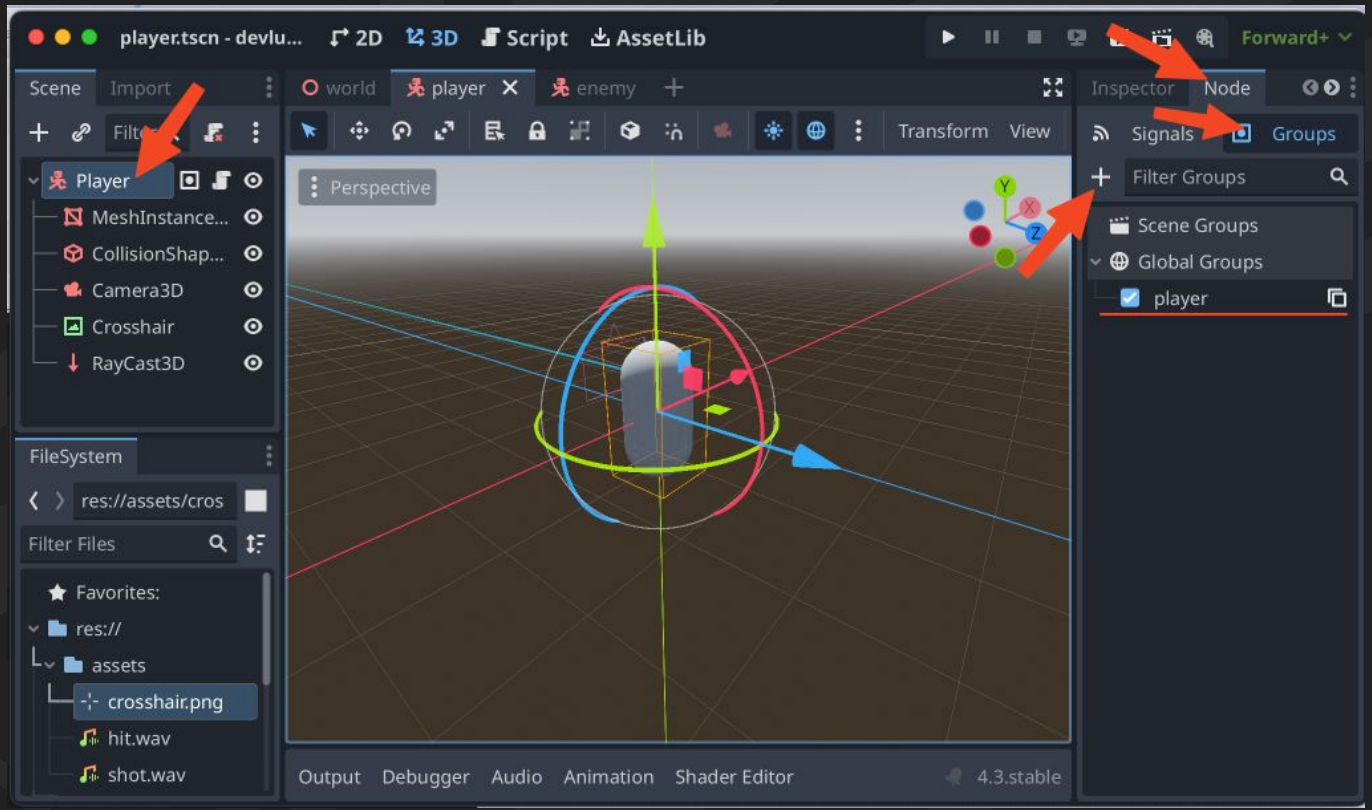
Add Thing to Shoot At to World



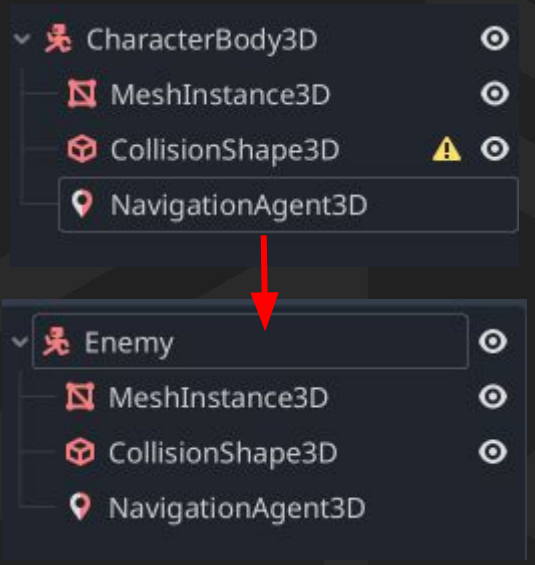


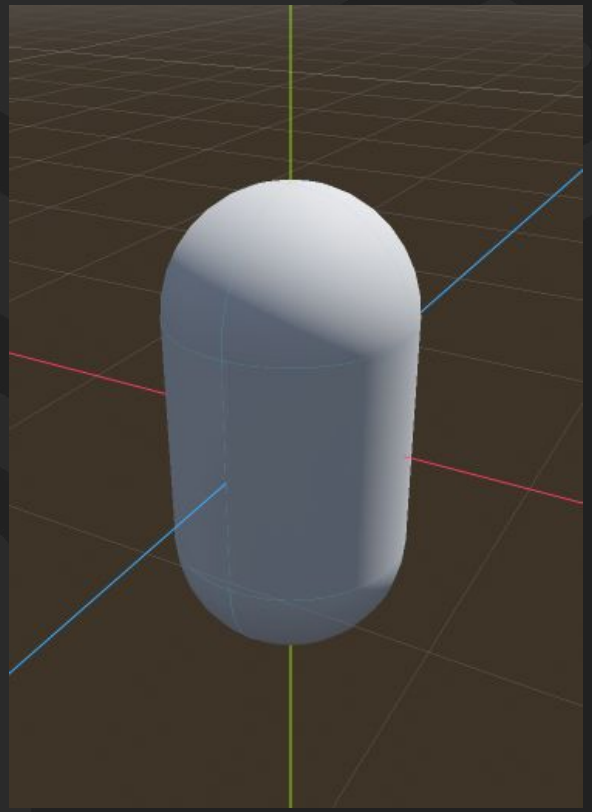
Our Enemy

Add “player” Tag to Player

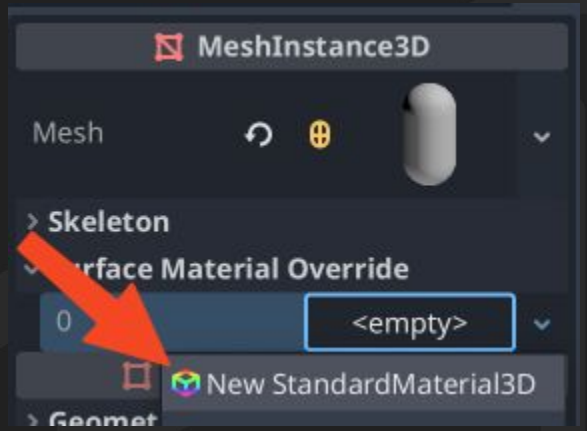


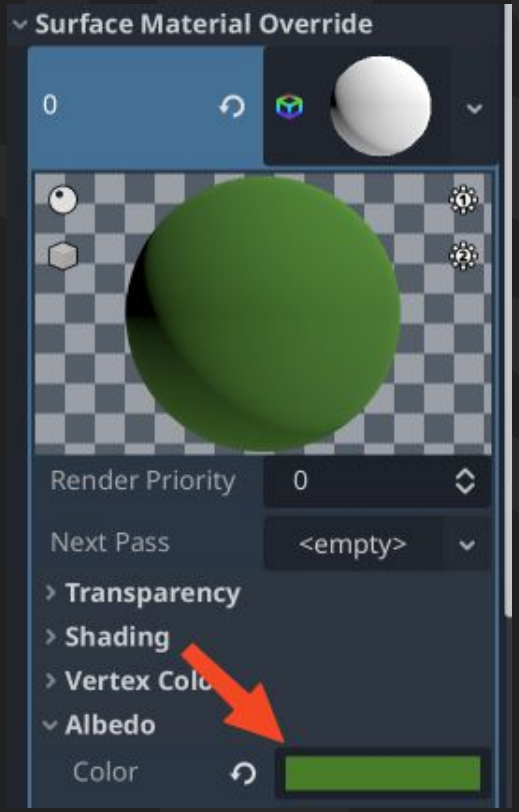
Create the Enemy Scene

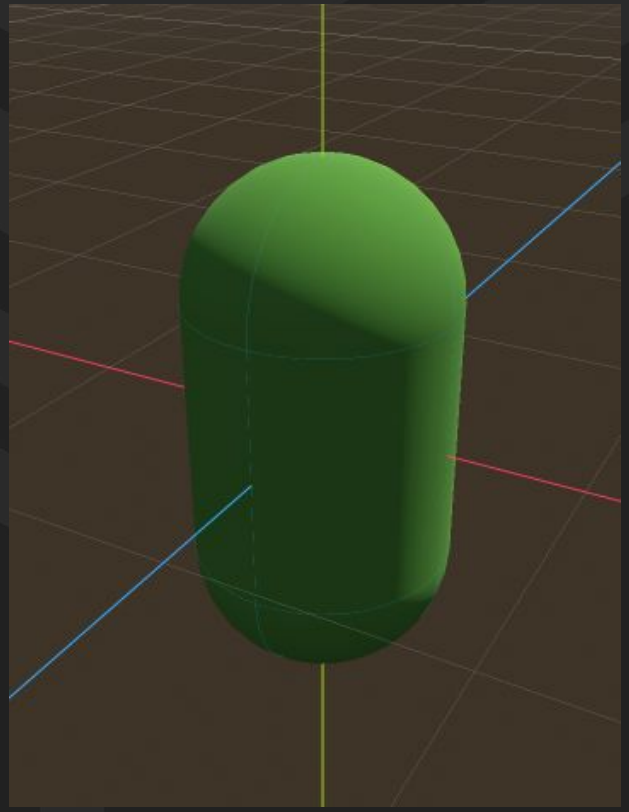




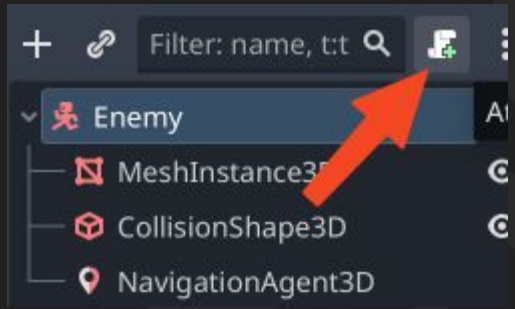
Change the Enemy's Color







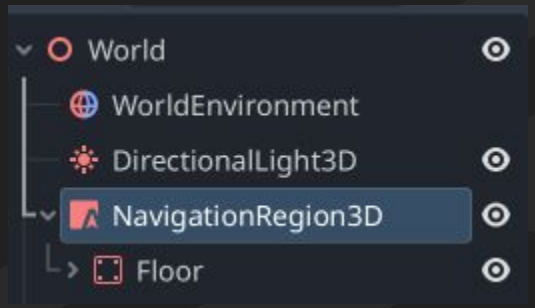
Add Script to Enemy

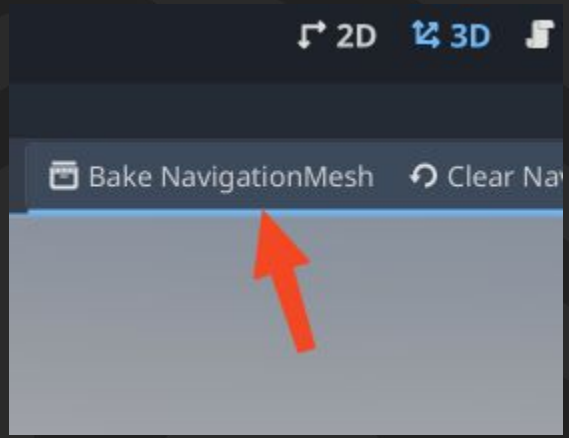


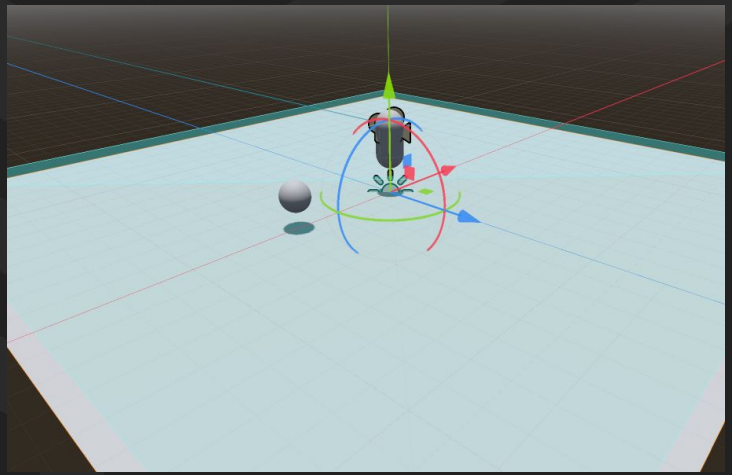
enemy.gd

```gdscript
class_name Enemy
extends CharacterBody3D


const SPEED = 5.0


@onready var nav_agent: NavigationAgent3D = $NavigationAgent3D
@onready var player: CharacterBody3D = get_tree().get_first_node_in_group("player")


func _ready() -> void:
    nav_agent.avoidance_enabled = true
    nav_agent.velocity_computed.connect(_on_nav_agent_velocity_computed)


func _physics_process(delta: float) -> void:
    if not is_on_floor():
        velocity += get_gravity() * delta
        move_and_slide()

    # Do not query when the map has never synchronized and is empty.
    if NavigationServer3D.map_get_iteration_id(nav_agent.get_navigation_map()) == 0:
        return

    if player and is_instance_valid(player):
        nav_agent.target_position = player.global_position

    var next_path_pos = nav_agent.get_next_path_position()
    var new_velocity = global_position.direction_to(next_path_pos) * SPEED

    nav_agent.velocity = new_velocity


func _on_nav_agent_velocity_computed(safe_velocity: Vector3) -> void:
    velocity = safe_velocity
    move_and_slide()
```

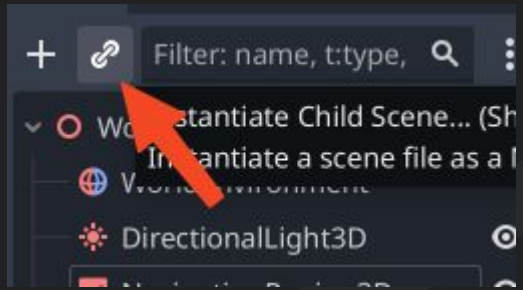
Add NavigationRegion3D to World

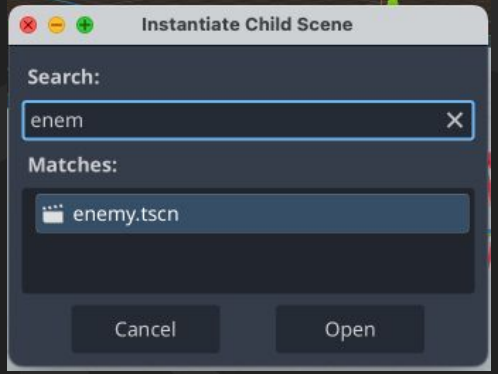


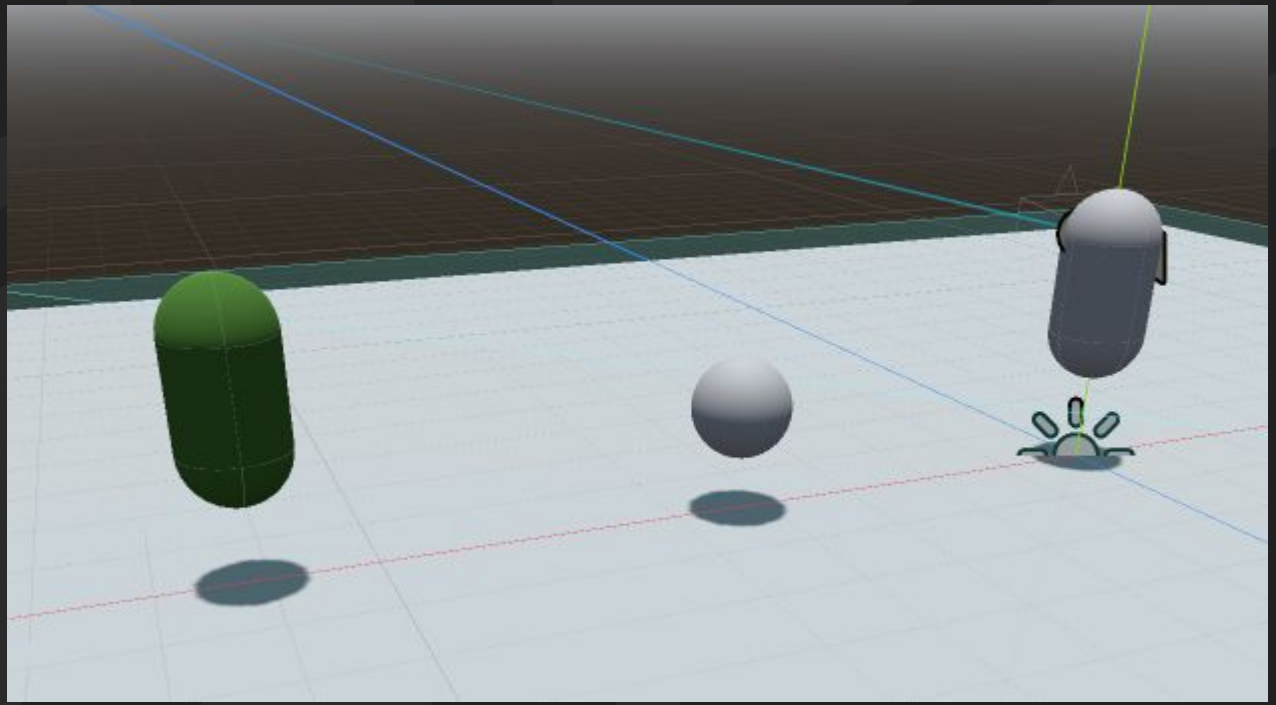




Add Enemy to World







Shooting the Enemy

player.gd

```gdscript
func _process(_delta: float) -> void:
    # Toggle whether the mouse is captured when "pause" is pressed.
    if Input.is_action_just_pressed("pause"):
        if Input.mouse_mode == Input.MOUSE_MODE_CAPTURED:
            Input.mouse_mode = Input.MOUSE_MODE_VISIBLE
        else:
            Input.mouse_mode = Input.MOUSE_MODE_CAPTURED

    # Handle shooting using the RayCast3D to determine what we hit.
    if Input.is_action_just_pressed("shoot"):
        var collider = ray_cast.get_collider()
        if collider is RigidBody3D:
            print("Hit RigidBody3D")
            collider.apply_impulse((transform.basis * Vector3(0, 0, -1)).normalized() * 3)
        elif collider is Enemy:
            print("Hit Enemy")
            collider.queue_free()
        else:
            print("Missed")
```

Taking Damage from the Enemy

player.gd

```
# Get the "RayCast3D" node that is a child
@onready var ray_cast: RayCast3D = $RayCast

var health: int = 5

func _ready() -> void:
    # Capture the mouse when the game starts
    Input.mouse_mode = Input.MOUSE_MODE_CAP
```

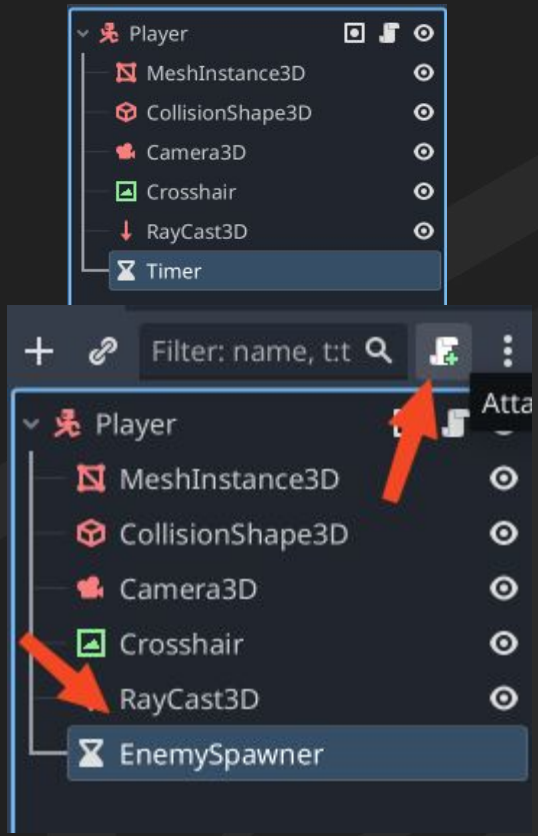
player.gd

```
func damage() -> void:
    health -= 1
    print("%s health remaining" % health)

    if health <= 0:
        print("You Lose!")
        get_tree().reload_current_scene()
```

enemy.gd

```
func _on_nav_agent_velocity_computed(safe_velocity: Vector3) -> void:
    velocity = safe_velocity
    move_and_slide()

    for i in range(get_slide_collision_count()):
        var collision = get_slide_collision(i)
        var collider = collision.get_collider()
        if collider == player:
            player.damage()
            queue_free()
```

Create the “EnemySpawner” Node from a Timer Node



enemy_spawner.gd

```gdscript
extends Timer

@onready var enemy_scene: PackedScene = load("res://enemy.tscn")
@onready var player: CharacterBody3D = get_parent()


func _ready() -> void:
    start()

    timeout.connect(_on_timeout)


func _on_timeout() -> void:
    # Picks a random position on a unit circle.
    var theta: float = randf() * 2 * PI
    var spawn_position = Vector3(cos(theta), 0, sin(theta))
    # Grow the circle by a random number between 10 and 30 meters.
    spawn_position *= randi_range(10, 30)
    # Offset the circle to be around the player.
    spawn_position += player.global_position

    # If spawn position is off the platform, try again.
    # This is a pretty bad practice because it could run "forever"... but it works
    if spawn_position.x > 25 or spawn_position.x < -25 or spawn_position.z > 25 or spawn_position.z < -25:
        _on_timeout()
        return

    # Spawn an enemy at the position.
    var enemy: Enemy = enemy_scene.instantiate()
    player.add_sibling(enemy)
    enemy.global_position = spawn_position
```

Extras (if we have time)

Extras

- Basic Level Design
- Scoring
- Sound Effects
- Second Enemy
- Basic UI

Recommended Further Learning

Exit Survey

Fig. 1: *Homer dislikes exit surveys.*