

Sign in here!



DevLUp FSU

GBM #3



Intro to 3D Game Dev

September 19th, 2024

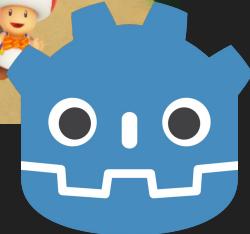
# showoff

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

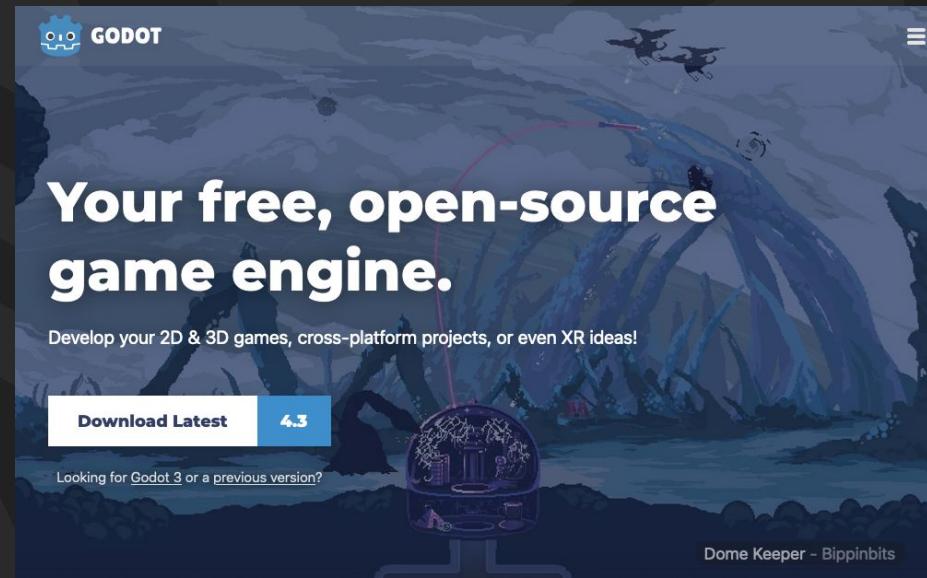
Lesson Plan

- 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?

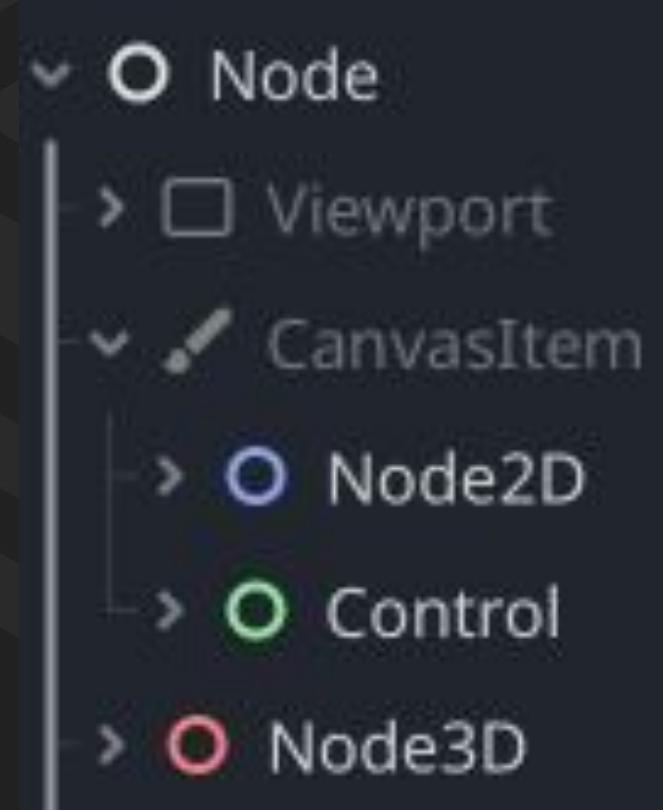
What are Nodes?

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



AnimatedSprite2D



TileMap



RayCast3D



Camera2D



Camera3D



Path3D



CollisionShape2D

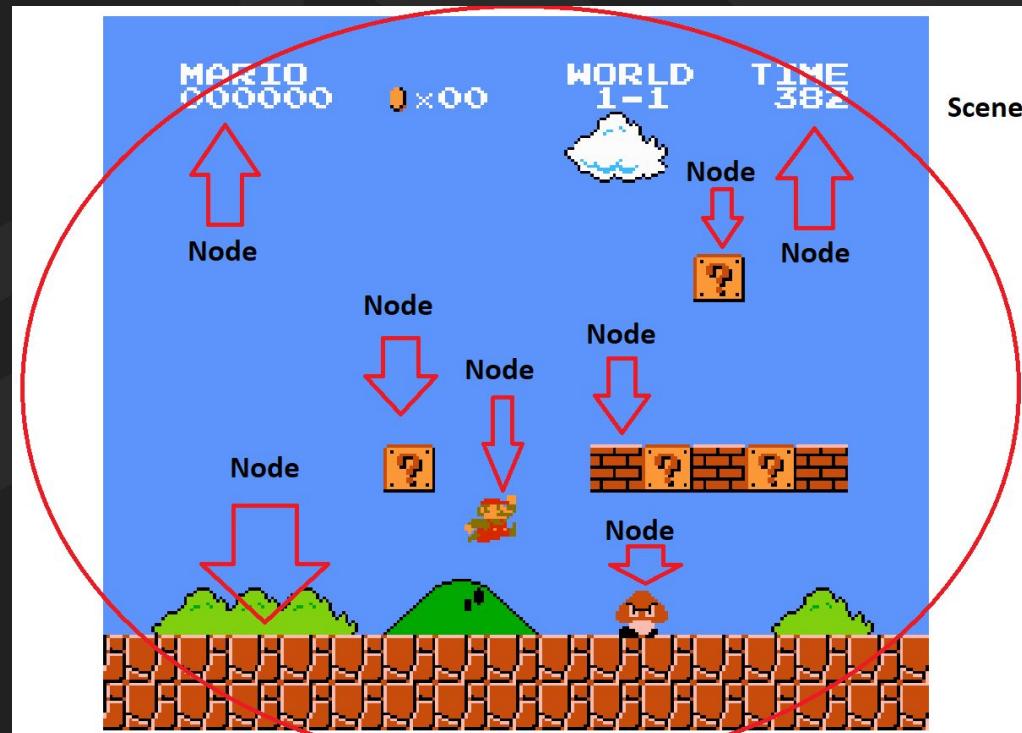


Label

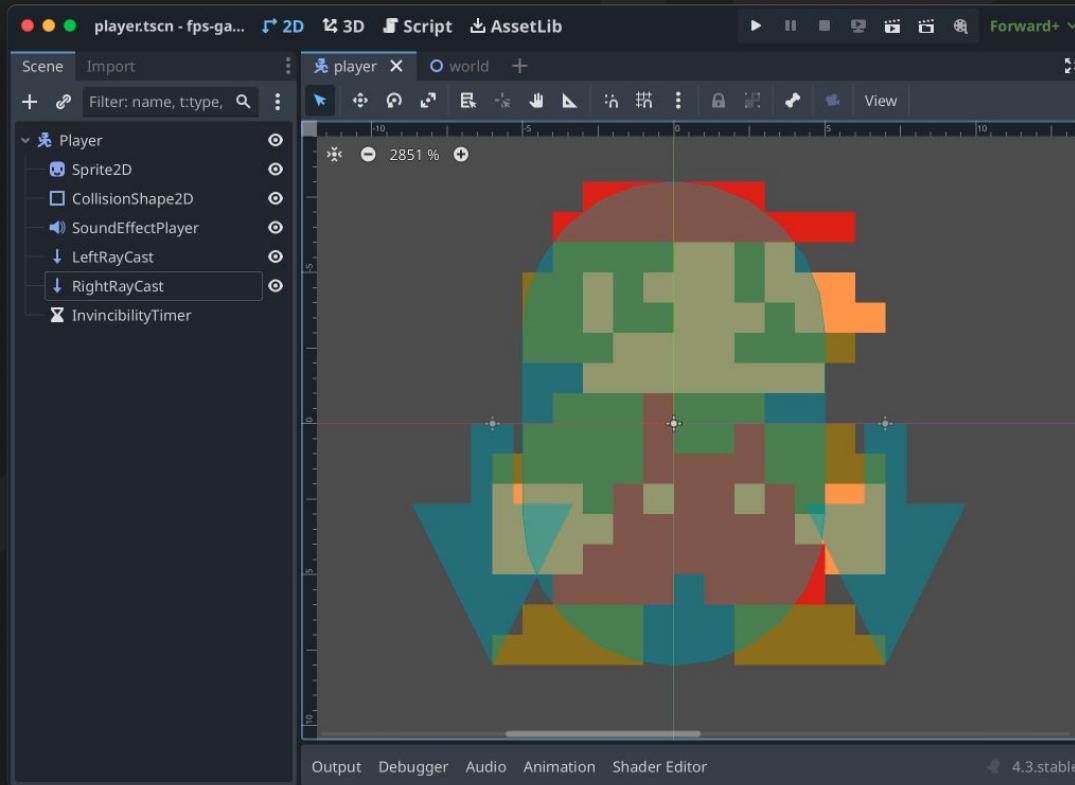


Timer

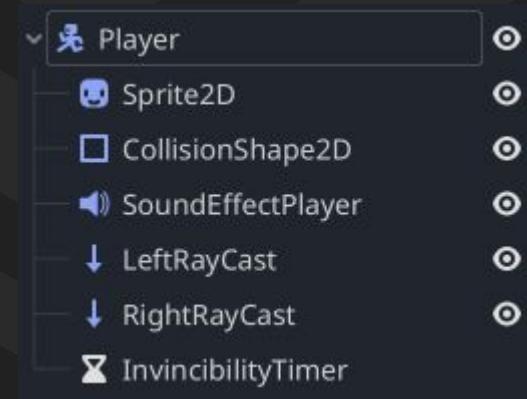
Everything is Made of Nodes!



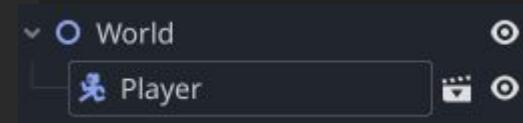
Make Your own Nodes: Scenes



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

Timer

CharacterBody3D

MeshInstance3D

NavigationRegion3D

StaticBody3D

CollisionShape3D

NavigationAgent3D

RigidBody3D

RayCast3D

WorldEnvironment

TextureRect

Camera3D

DirectionalLight3D

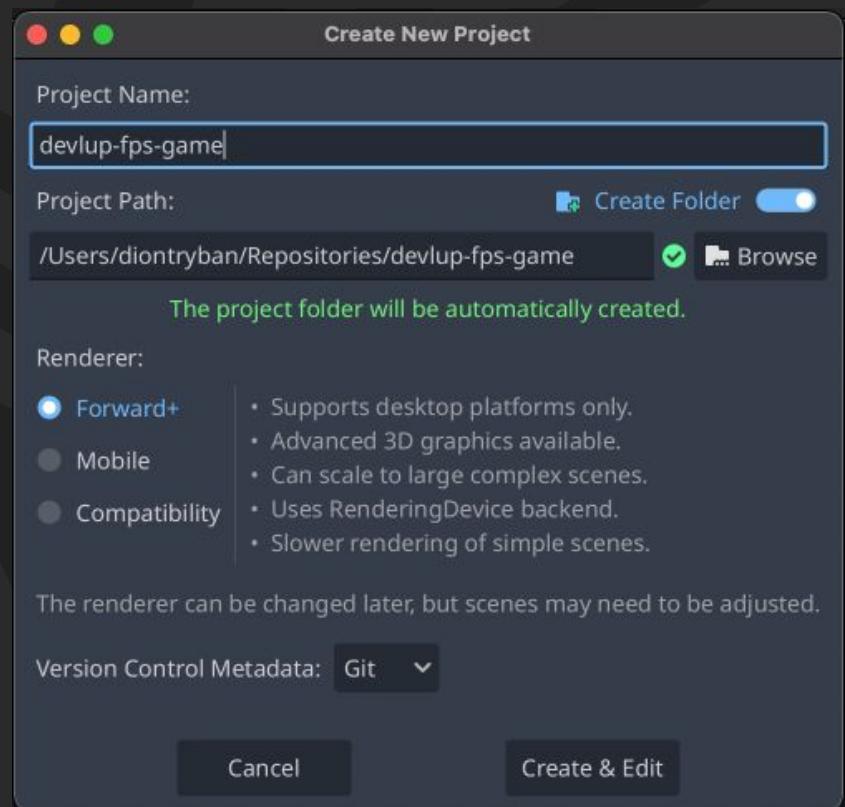
What Nodes Could Describe This Scene?



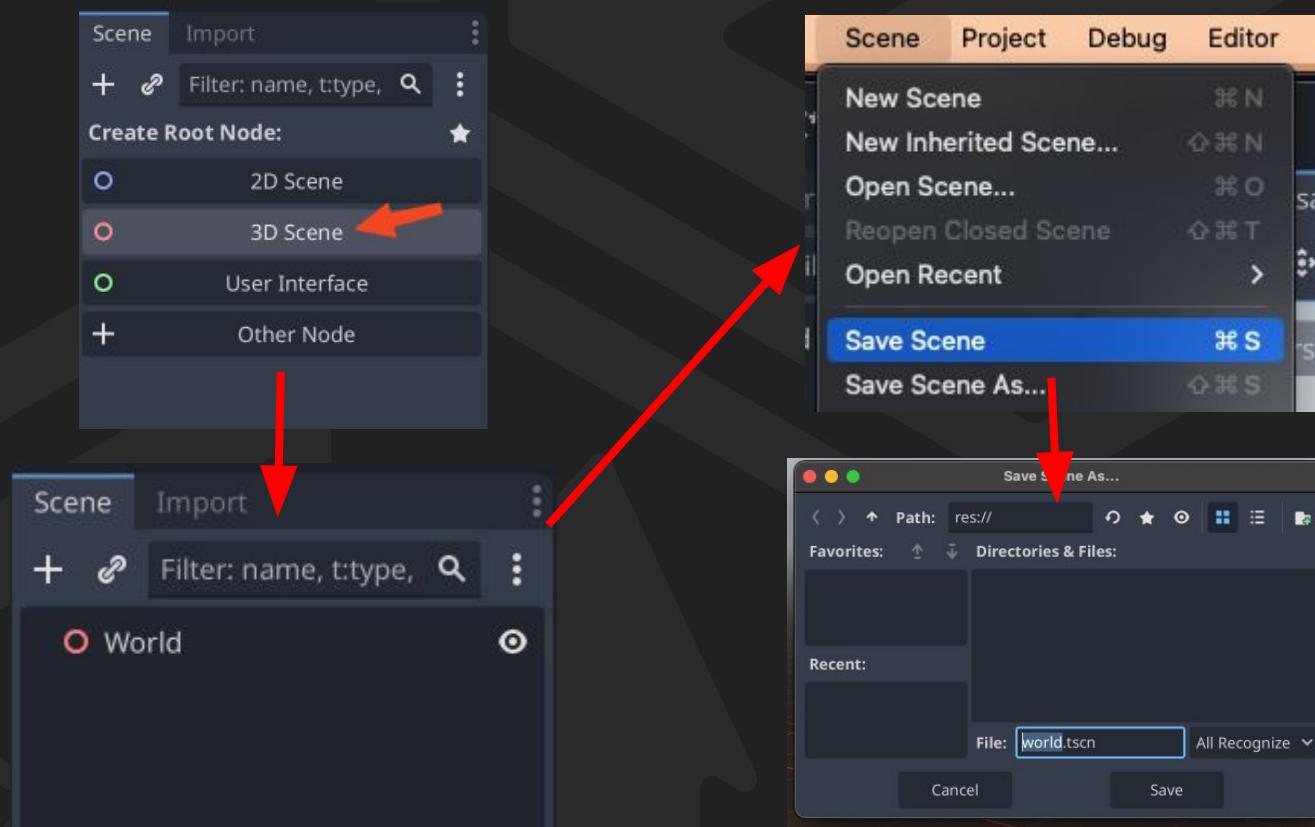
Creating the Project



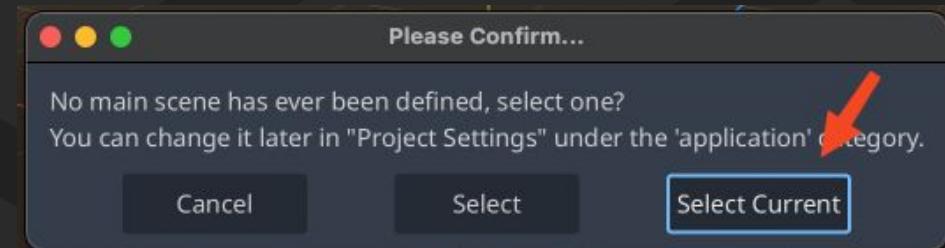
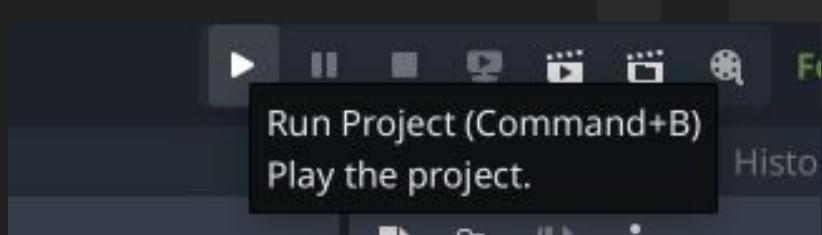
Create New Project



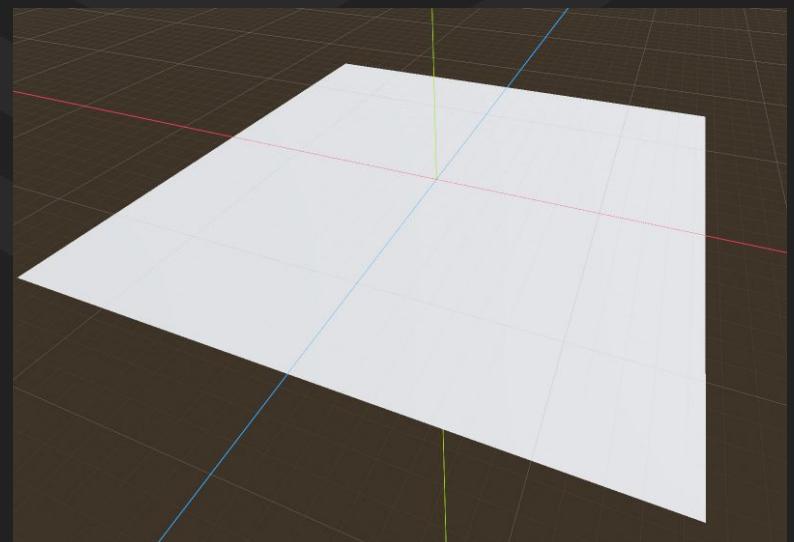
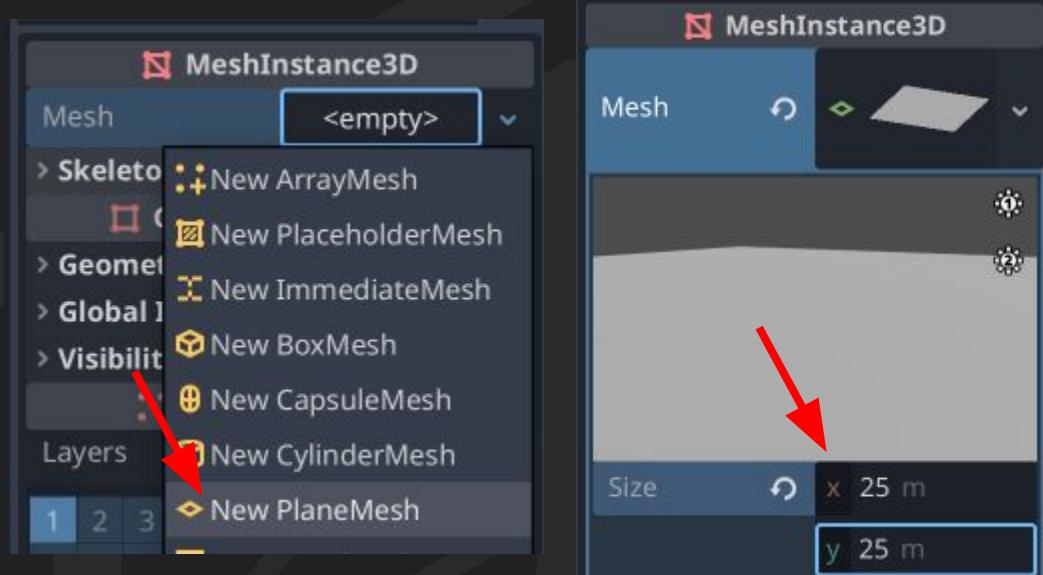
Create the “World” Scene



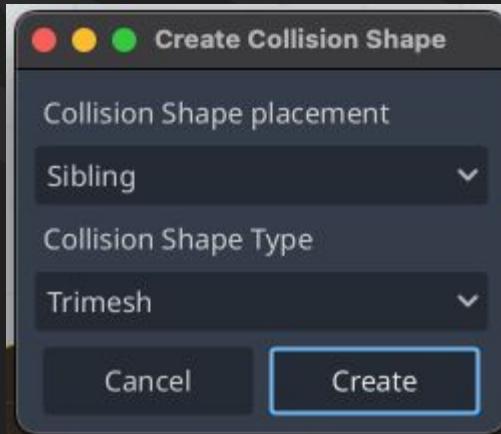
Run the Game!



Create the Floor Mesh



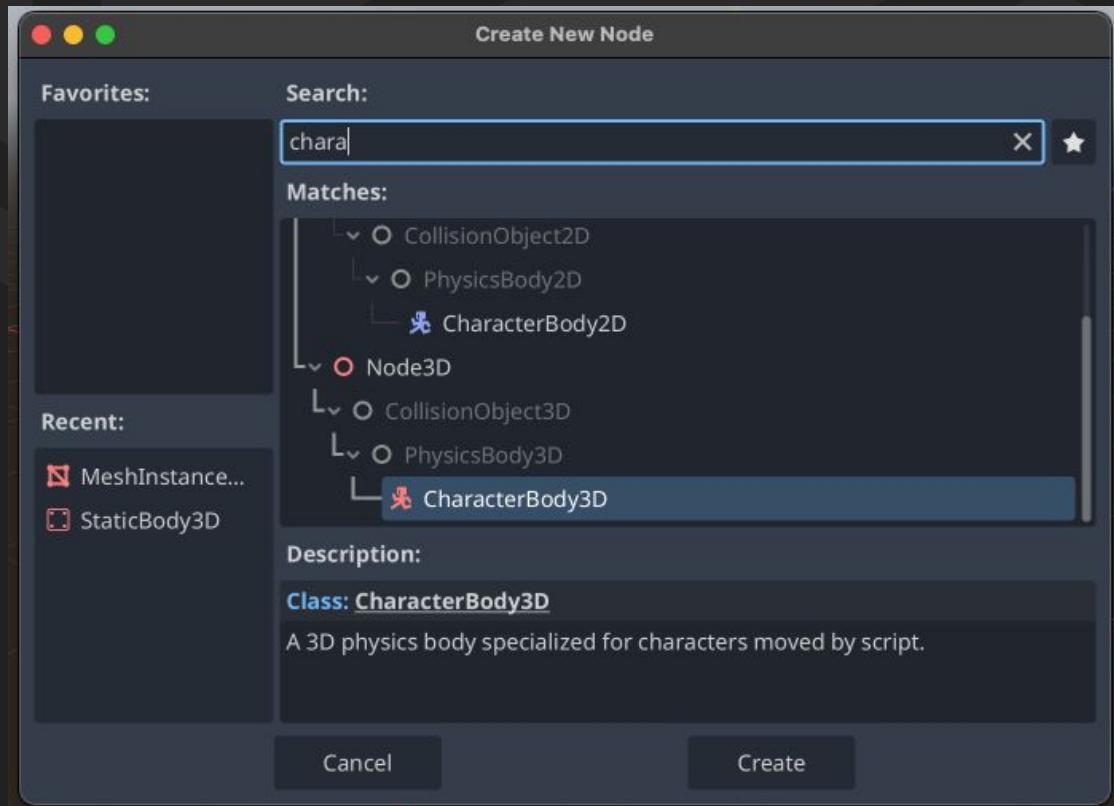
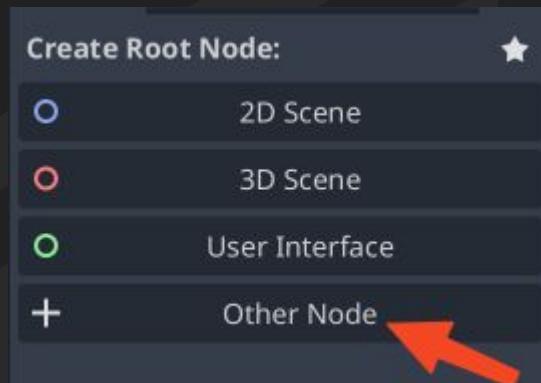
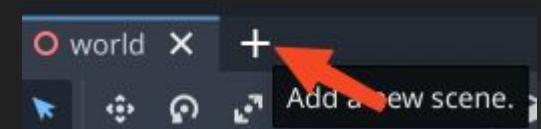
Make the Floor have “Collision”



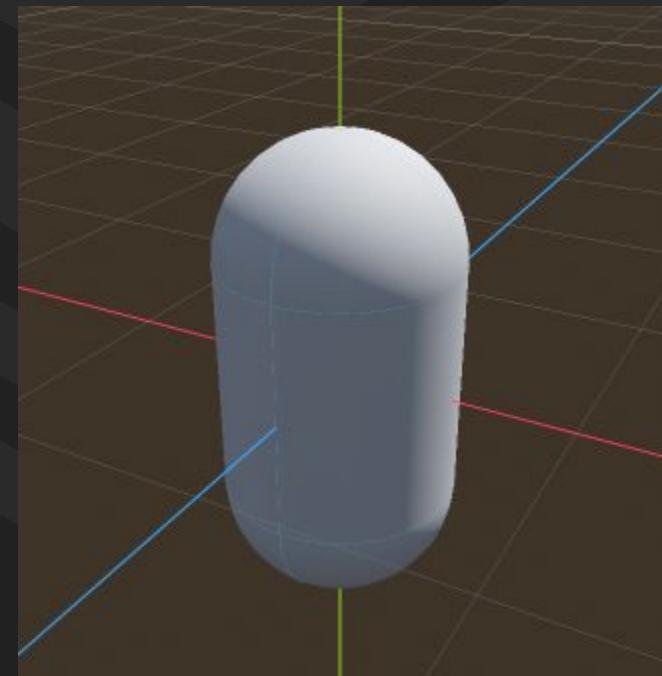
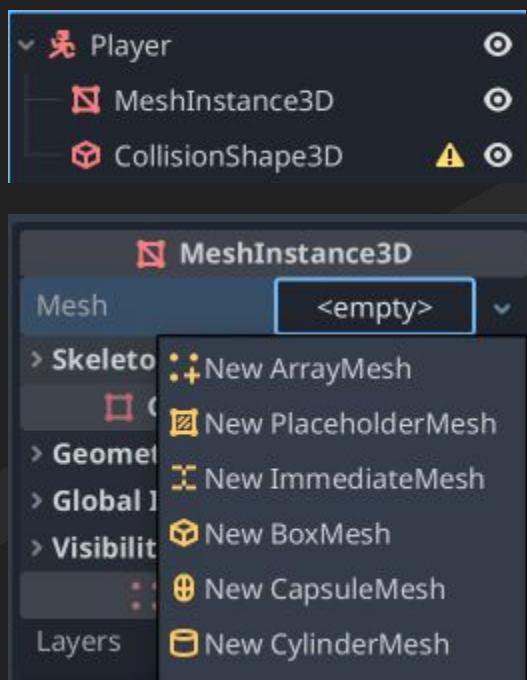
The Player



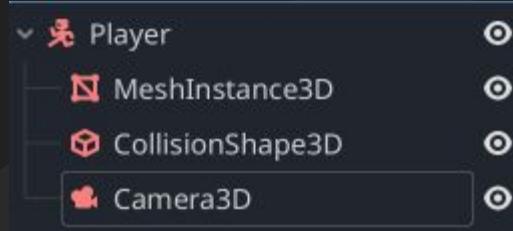
Make the Player Scene



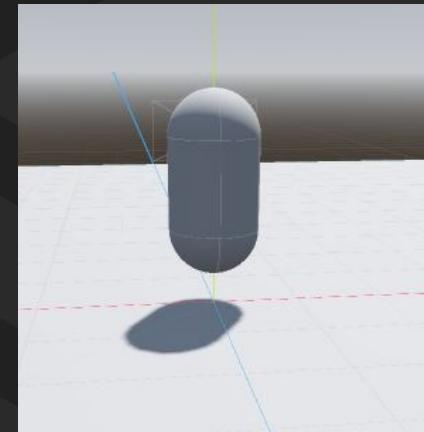
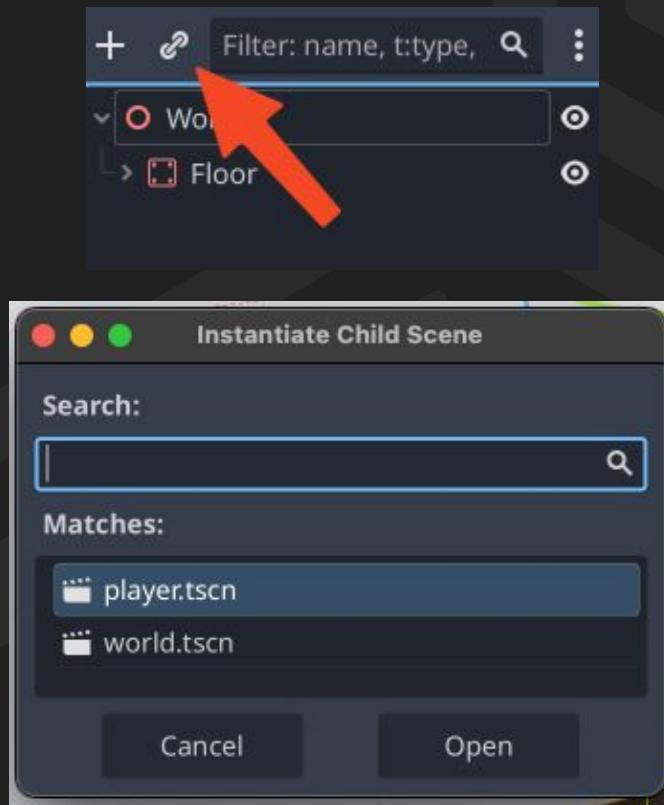
Make the Player Scene



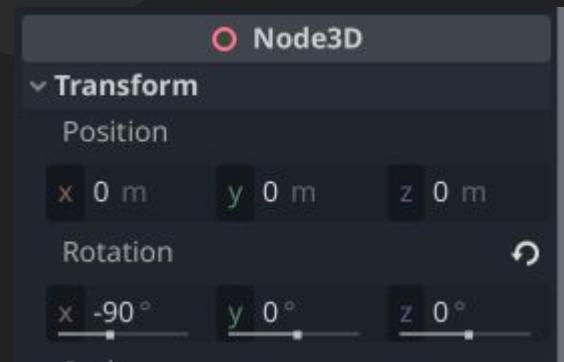
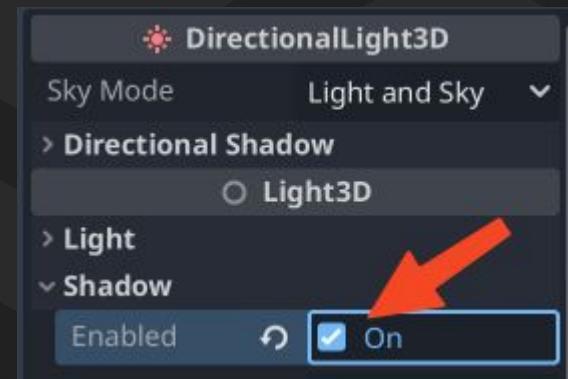
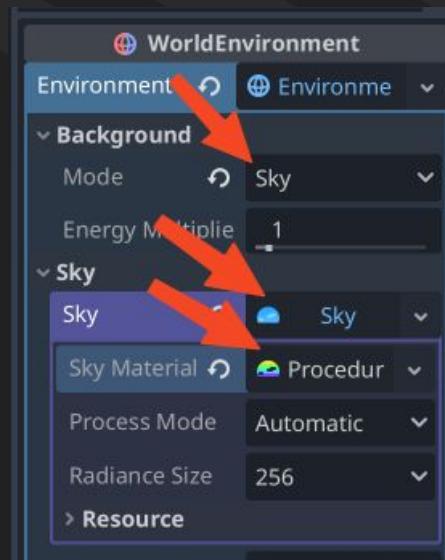
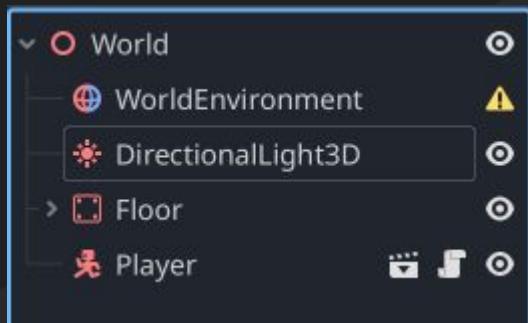
Add a Camera to the Player



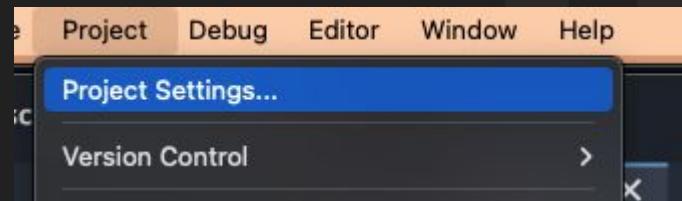
Add Player to Main Scene



Let there be light! (Add Environment to World)



Add Input Mappings



Project Settings (project.godot)

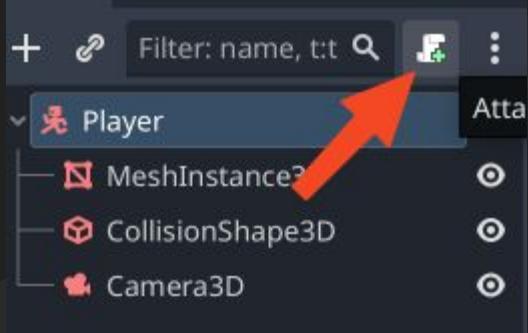
General Input Map Localization Globals Plugins

Filter by Name Filter by Event Clear All

Add New Action Show Built-in Actions

Action	Deadzone	
player_forward	0.5	<input type="button" value="+"/> <input type="button" value="Delete"/>
W (Physical)		<input type="button" value="Delete"/>
player_back	0.5	<input type="button" value="+"/> <input type="button" value="Delete"/>
S (Physical)		<input type="button" value="Delete"/>
player_left	0.5	<input type="button" value="+"/> <input type="button" value="Delete"/>
A (Physical)		<input type="button" value="Delete"/>
player_right	0.5	<input type="button" value="+"/> <input type="button" value="Delete"/>
D (Physical)		<input type="button" value="Delete"/>
player_jump	0.5	<input type="button" value="+"/> <input type="button" value="Delete"/>
Space (Physical)		<input type="button" value="Delete"/>
pause	0.5	<input type="button" value="+"/> <input type="button" value="Delete"/>
Escape (Physical)		<input type="button" value="Delete"/>
shoot	0.5	<input type="button" value="+"/> <input type="button" value="Delete"/>
Left Mouse Button - All Devices		<input type="button" value="Delete"/>

Add Script to Player



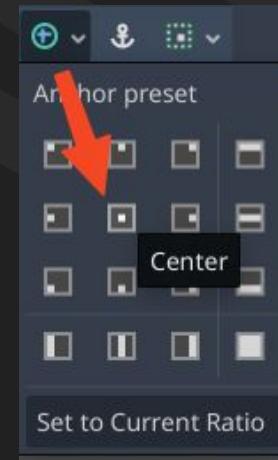
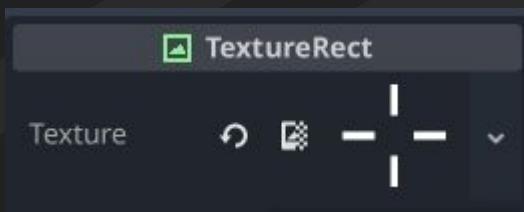
player.gd

```
1  extends CharacterBody3D
2
3
4  const SPEED = 5.0
5  const JUMP_VELOCITY = 4.5
6  const MOUSE_SENSITIVITY := 0.1
7
8
9  func _ready() -> void:
10     # Capture the mouse when the game starts.
11     Input.mouse_mode = Input.MOUSE_MODE_CAPTURED
12
13
14  func _process(_delta: float) -> void:
15     # Toggle whether the mouse is captured when "pause" is pressed.
16     if Input.is_action_just_pressed("pause"):
17         if Input.mouse_mode == Input.MOUSE_MODE_CAPTURED:
18             Input.mouse_mode = Input.MOUSE_MODE_VISIBLE
19         else:
20             Input.mouse_mode = Input.MOUSE_MODE_CAPTURED
21
22
23  func _input(event: InputEvent) -> void:
24     if Input.mouse_mode == Input.MOUSE_MODE_CAPTURED:
25         if event is InputEventMouseMotion:
26             rotation_degrees.y -= event.relative.x * MOUSE_SENSITIVITY
27             rotation_degrees.x -= event.relative.y * MOUSE_SENSITIVITY
28             rotation_degrees.x = clamp(rotation_degrees.x, -90.0, 90.0)
29
30
31  func _physics_process(delta: float) -> void:
32     # Add the gravity.
33     if not is_on_floor():
34         velocity += get_gravity() * delta
35
36     # Handle jump.
37     if Input.is_action_just_pressed("player_jump") and is_on_floor():
38         velocity.y = JUMP_VELOCITY
39
40     # Get the input direction and handle the movement/deceleration.
41     var input_dir := Input.get_vector("player_left", "player_right", "player_forward", "player_back")
42     var direction := (transform.basis * Vector3(input_dir.x, 0, input_dir.y)).normalized()
43
44     if direction:
45         velocity.x = direction.x * SPEED
46         velocity.z = direction.z * SPEED
47     else:
48         velocity.x = move_toward(velocity.x, 0, SPEED)
49         velocity.z = move_toward(velocity.z, 0, SPEED)
50
51     move_and_slide()
```

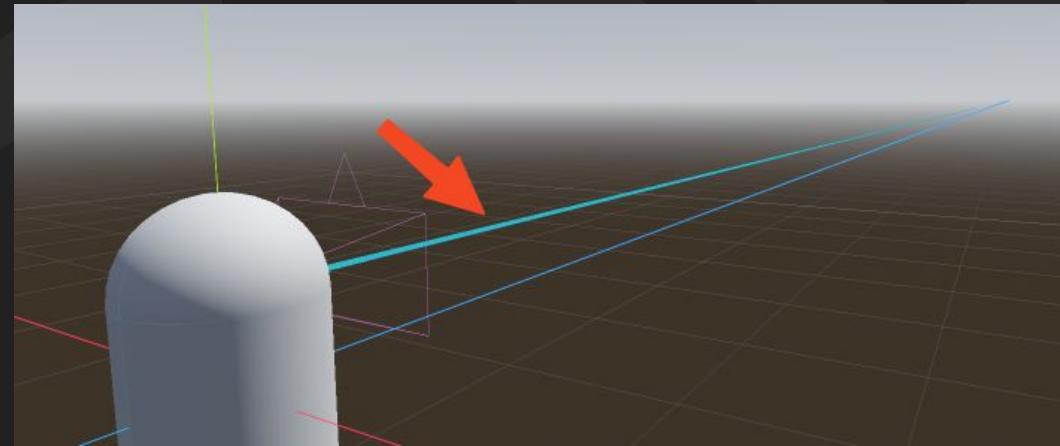
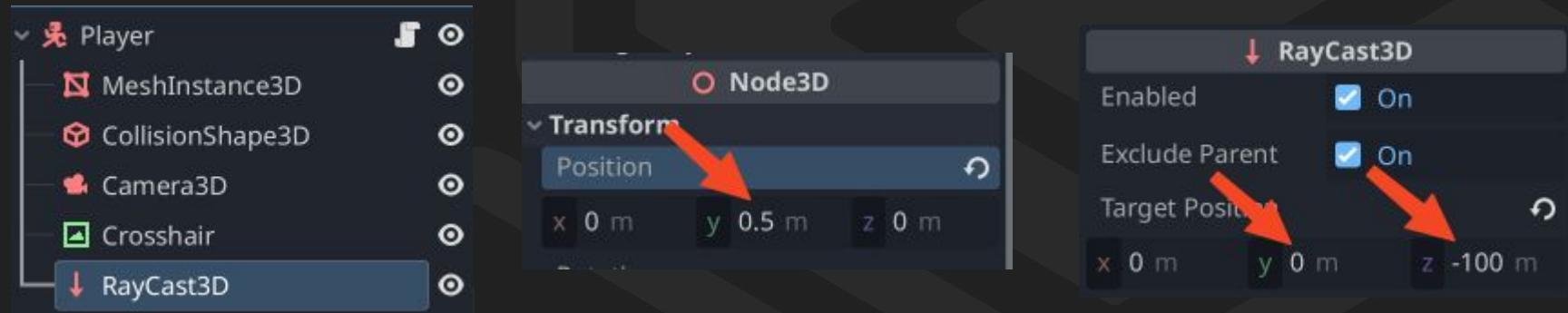
Shooting



Add Crosshair



Add RayCast for “HitScan” Weapon

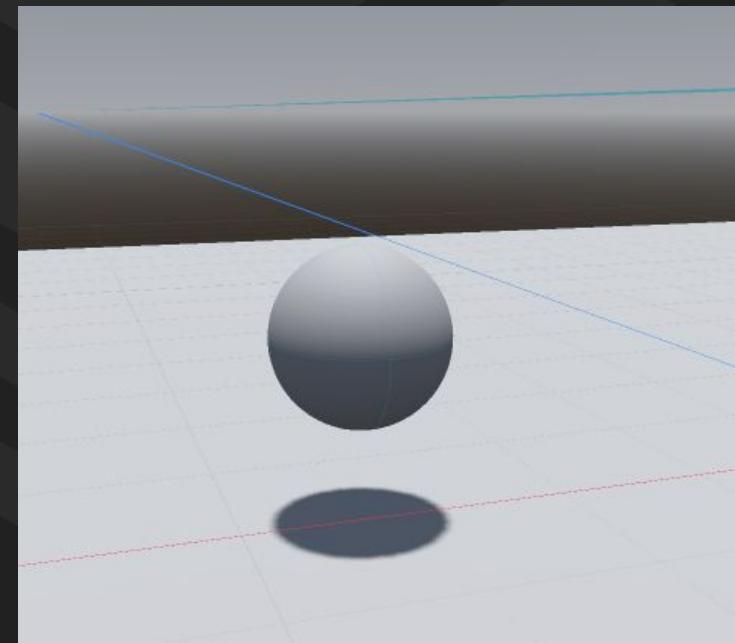


Add Shooting Code to Player Script

```
1  extends CharacterBody3D
2
3
4  const SPEED = 5.0
5  const JUMP_VELOCITY = 4.5
6  const MOUSE_SENSITIVITY := 0.1
7
8
9  # Get the "RayCast3D" node that is a child of the player node.
10 @onready var ray_cast: RayCast3D = $RayCast3D
11
12
13 func _ready() -> void:
14     # Capture the mouse when the game starts.
15     Input.mouse_mode = Input.MOUSE_MODE_CAPTURED
16
17
18 func _process(_delta: float) -> void:
19     # Toggle whether the mouse is captured when "pause" is pressed.
20     if Input.is_action_just_pressed("pause"):
21         if Input.mouse_mode == Input.MOUSE_MODE_CAPTURED:
22             Input.mouse_mode = Input.MOUSE_MODE_VISIBLE
23         else:
24             Input.mouse_mode = Input.MOUSE_MODE_CAPTURED
25
26     # Handle shooting using the RayCast3D to determine what we hit.
27     if Input.is_action_just_pressed("shoot"):
28         var collider = ray_cast.get_collider()
29         if collider is RigidBody3D:
30             print("Hit RigidBody3D")
31             collider.apply_impulse((transform.basis * Vector3(0, 0, -1)).normalized() * 3)
32         else:
33             print("Missed")
```

player.gd

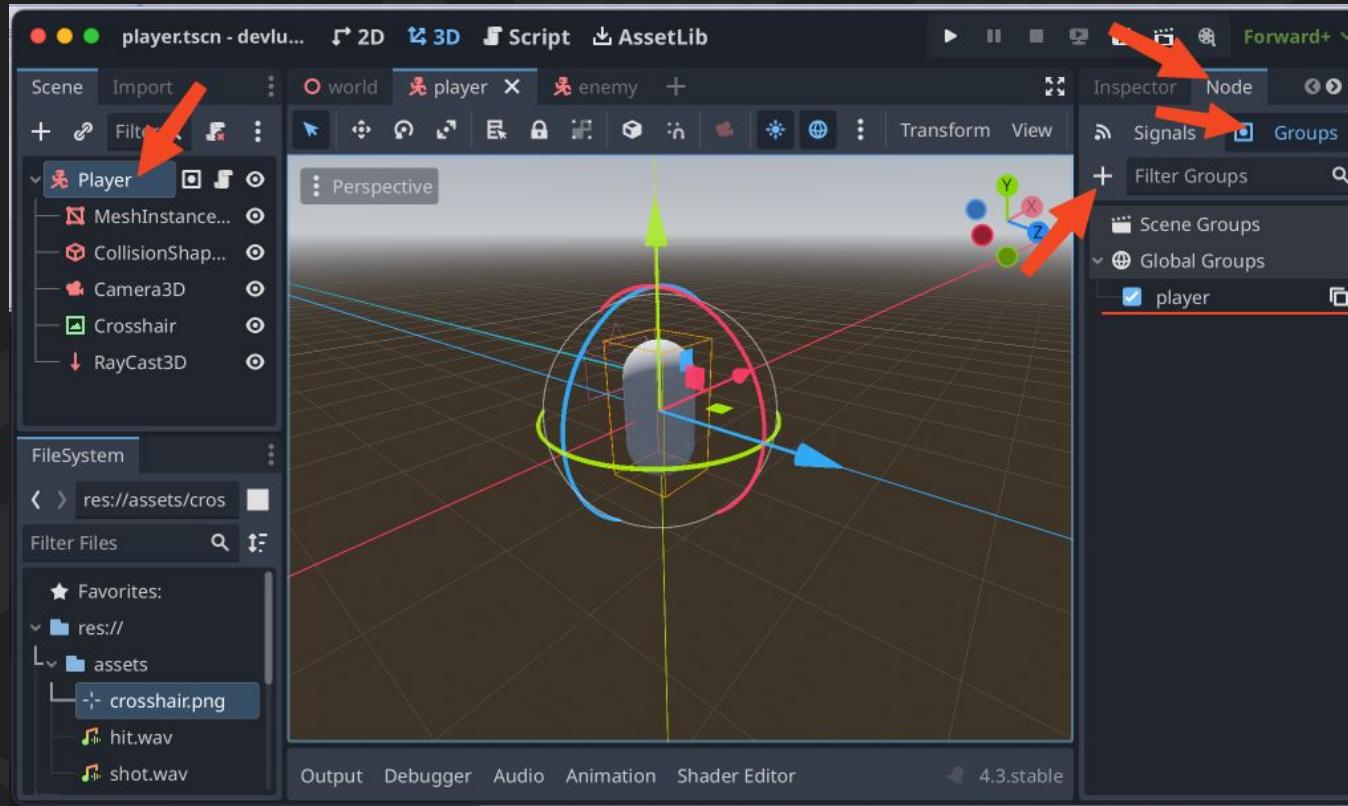
Add Thing to Shoot At to World



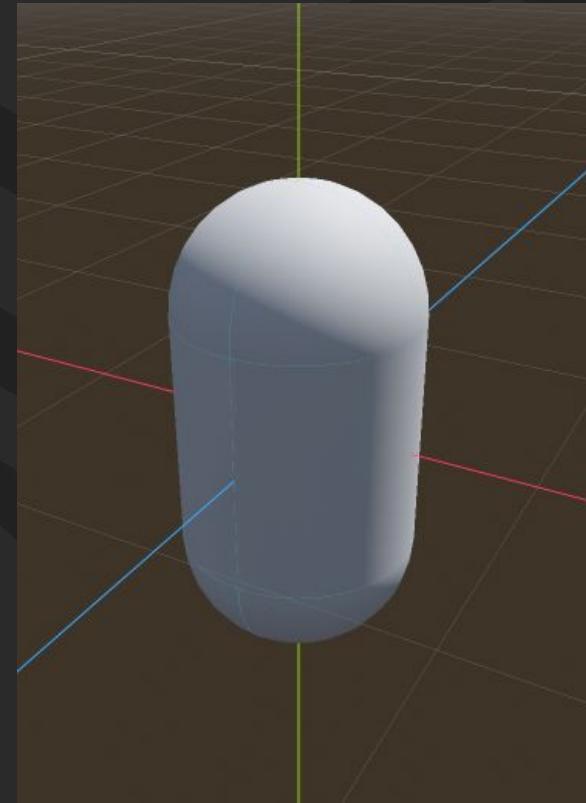
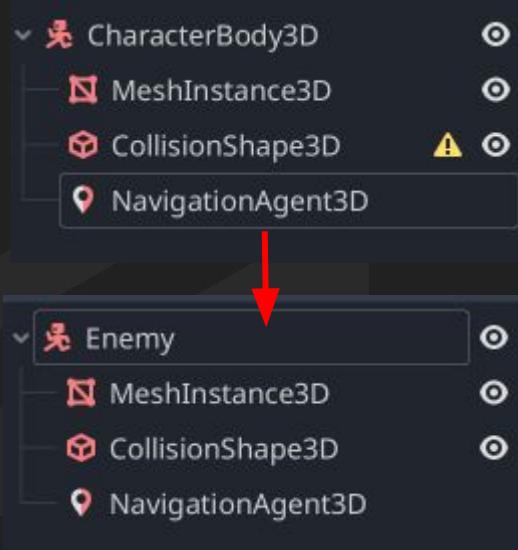
Our Enemy



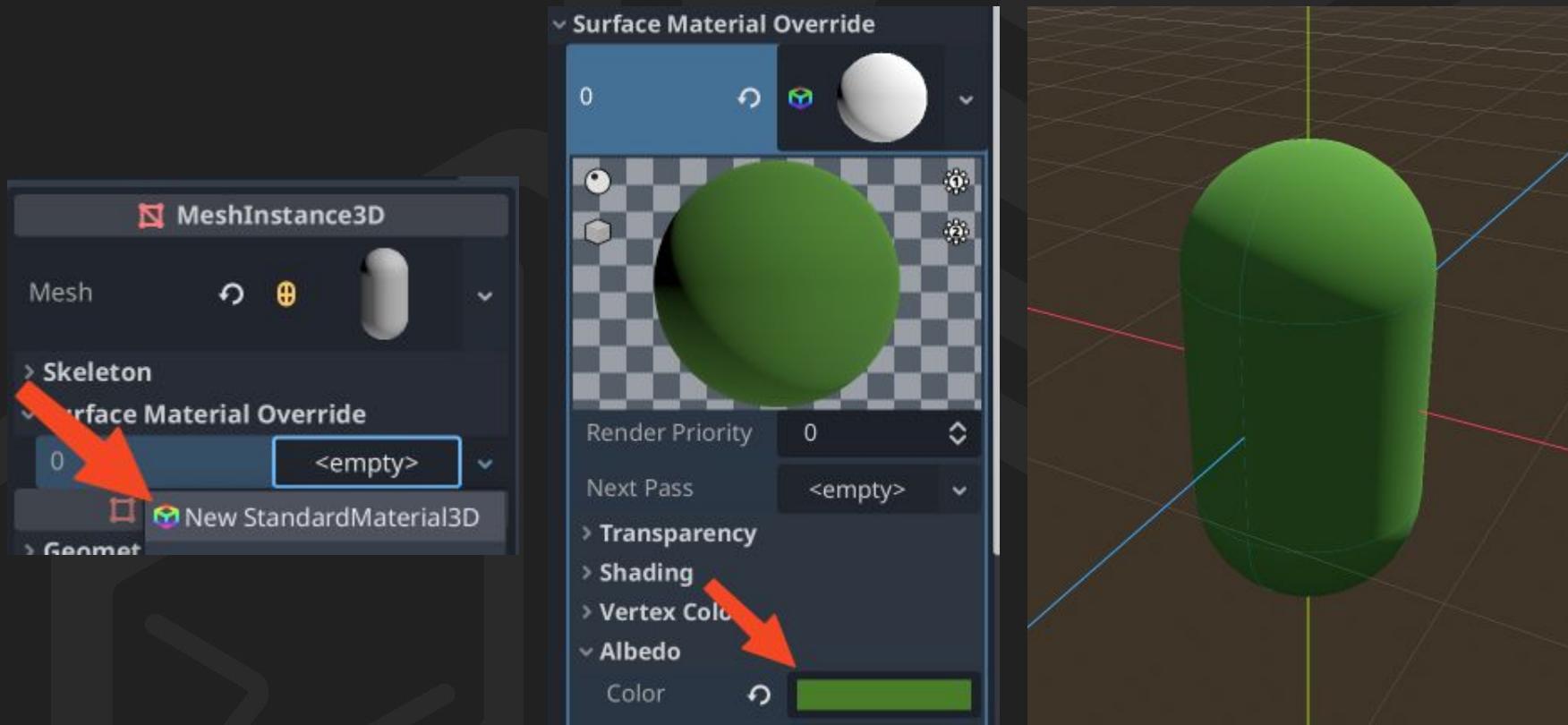
Add “player” Tag to Player



Create the Enemy Scene



Change the Enemy's Color



The image illustrates the process of changing the color of an enemy model in a 3D engine. It consists of three main panels: the Inspector, the Surface Material Override settings, and the 3D Viewport.

Inspector (Left): Shows the `MeshInstance3D` component. The `Surface Material Override` slot is currently empty, indicated by the placeholder text `<empty>`. A red arrow points to this slot.

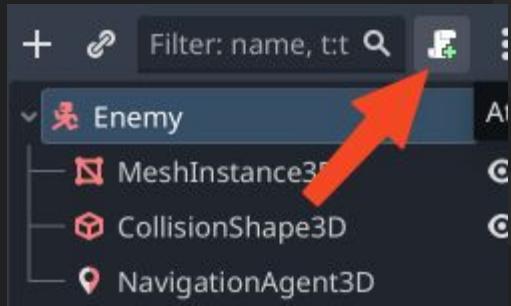
Surface Material Override Settings (Center): This panel shows the configuration for the material override. It includes:

- Preview:** A 3D preview of a green sphere.
- Render Priority:** Set to 0.
- Next Pass:** Placeholder text `<empty>`.
- Transparency:** Placeholder text `<empty>`.
- Shading:** Placeholder text `<empty>`.
- Vertex Color:** Placeholder text `<empty>`.
- Albedo:** A color swatch set to green. A red arrow points to this section.

3D Viewport (Right): Shows a green capsule-shaped model in a 3D coordinate system. The model is centered at the origin (0,0,0) and is oriented vertically. A bounding box is visible around the model.

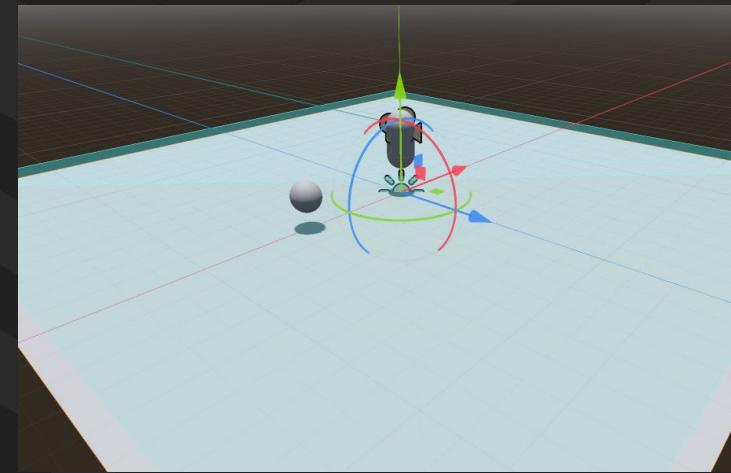
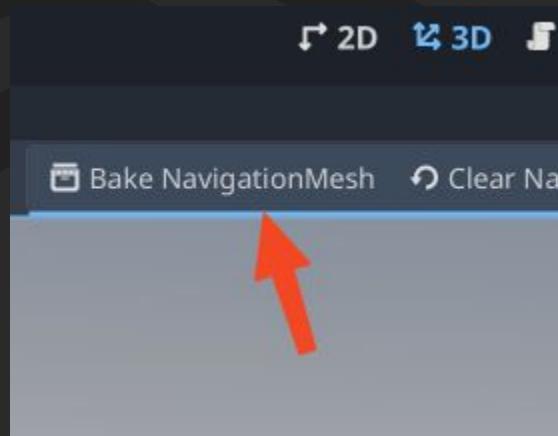
Add Script to Enemy

enemy.gd

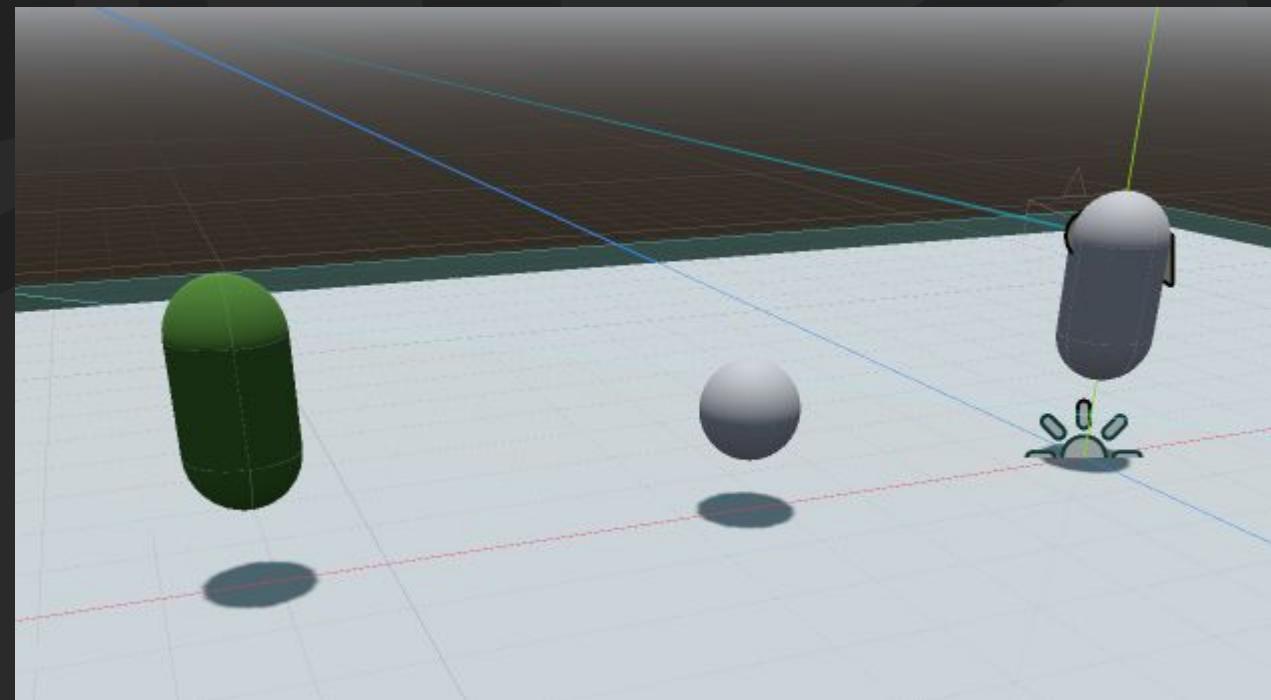
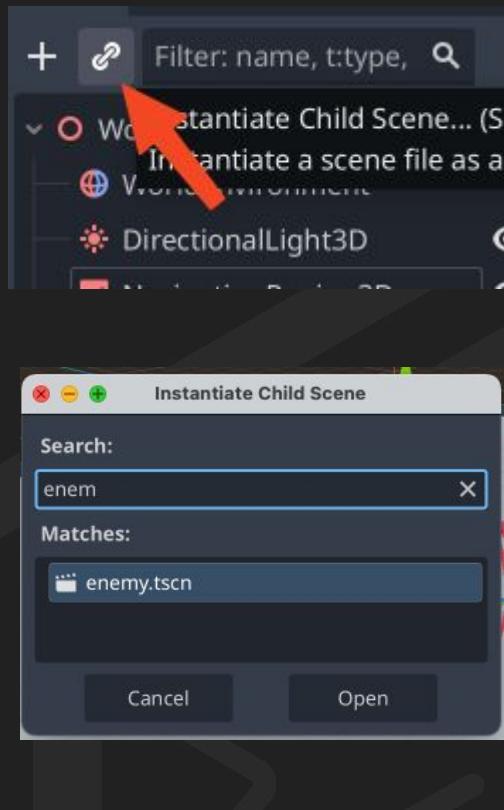


```
1  class_name Enemy
2  extends CharacterBody3D
3
4
5  const SPEED = 5.0
6
7
8  @onready var nav_agent: NavigationAgent3D = $NavigationAgent3D
9  @onready var player: CharacterBody3D = get_tree().get_first_node_in_group("player")
10
11
12  func _ready() -> void:
13      nav_agent.avoidance_enabled = true
14      nav_agent.velocity_computed.connect(_on_nav_agent_velocity_computed)
15
16
17  func _physics_process(delta: float) -> void:
18      if not is_on_floor():
19          velocity += get_gravity() * delta
20          move_and_slide()
21
22      # Do not query when the map has never synchronized and is empty.
23      if NavigationServer3D.map_get_iteration_id(nav_agent.get_navigation_map()) == 0:
24          return
25
26      if player and is_instance_valid(player):
27          nav_agent.target_position = player.global_position
28
29      var next_path_pos = nav_agent.get_next_path_position()
30      var new_velocity = global_position.direction_to(next_path_pos) * SPEED
31
32      nav_agent.velocity = new_velocity
33
34
35  func _on_nav_agent_velocity_computed(safe_velocity: Vector3) -> void:
36      velocity = safe_velocity
37      move_and_slide()
```

Add NavigationRegion3D to World



Add Enemy to World



Shooting the Enemy

```
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")
```

player.gd

Taking Damage from the Enemy

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

```
var health: int = 5
```

```
func _ready() -> void:
```

```
    # Capture the mouse when the game starts
    Input.mouse_mode = Input.MOUSE_MODE_CAPTURED
```

```
func damage() -> void:
```

player.gd

```
    health -= 1
    print("%s health remaining" % health)
```

```
    if health <= 0:
```

```
        print("You Lose!")
```

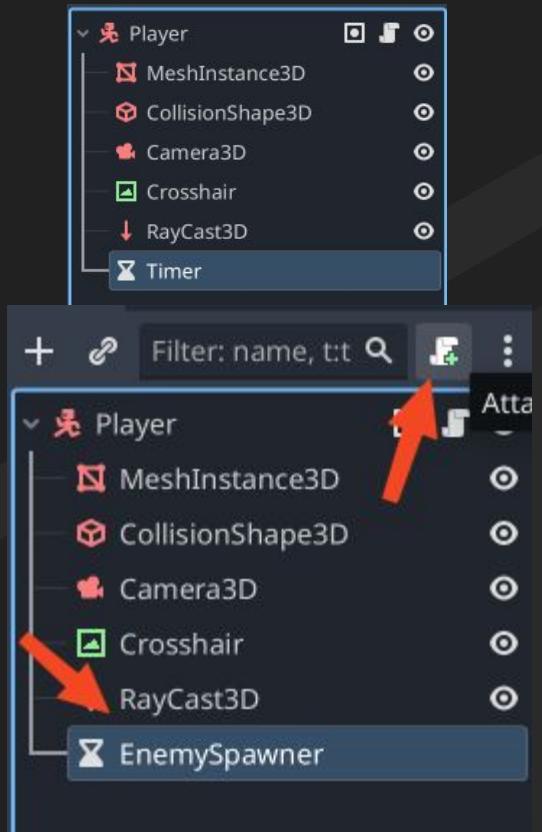
```
        get_tree().reload_current_scene()
```

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

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

enemy.gd

Create the “EnemySpawner” Node from a Timer Node



```
extends Timer

@ready var enemy_scene: PackedScene = load("res://enemy.tscn")
@ready 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
```

enemy_spawner.gd

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.*

