




Video Game: Punchman

By: Isaiah Dorsey, Nasir Hill, Demario James, Joseph Johnson

++++

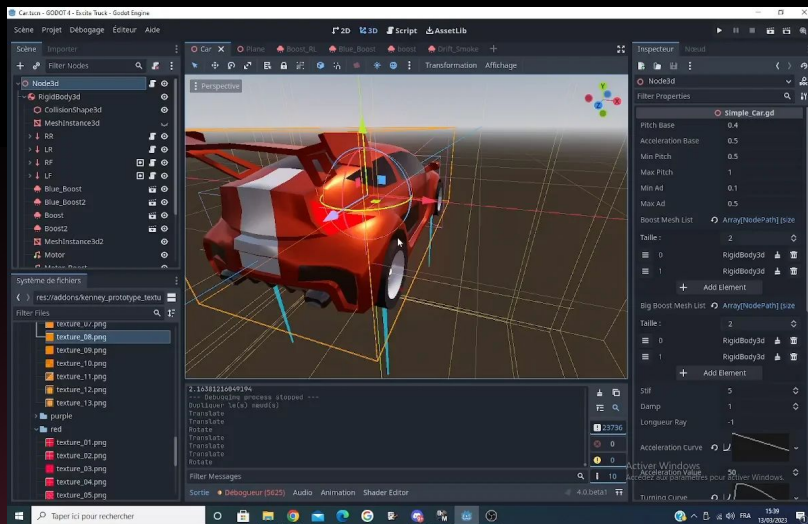


Introduction

- Purpose of the Project:
 - Try something different from what we have learned
 - We wanted to do a small 2D Video Game to have some experience in game development.
 - To understand how Game Development process looks like

Software

GoDot 4



Last Semester

- Gave an idea of what our video game was going to look like
- Plans about what to add to our project
 - Character & enemies movement/animations
 - Database
 - Score system
 - Different UI screens

What didn't get added

- The Database
 - Software for this: Silent Wolf
 - It couldn't get added because the leaderboard it was supposed to come with wasn't working together with Godot 4. It was not allowing the user to save their score and name to the database properly.
- Leaderboard
 - Software for this: Silent Wolf
 - It wasn't able to be added due to compatibility issues with this and Godot 4. It wasn't able to get the scores from the database.
- Pause Menu
 - Wasn't able to connect the pause menu with the main game

Character & enemies movements

- Since character and enemies are vital to make a video game, we had to make sure that they were interaction accordingly.



Character & Enemies Code

```
3 @onready var animation_tree : AnimationTree = $AnimationTree
4
5 @export var Starting_move_directions : Vector2 = Vector2.LEFT
6 @export var Movement_Speed : float = 30.0
7 @export var hit_state : State
8
9 @onready var state_machine : CharacterStateMachine = $CharacterStateMachine
10
11 var gravity = ProjectSettings.get_setting("physics/2d/default_gravity")
12
13 func _ready():
14     animation_tree.active = true
15
16 func _physics_process(delta):
17     if not is_on_floor():
18         velocity.y += gravity * delta
19
20     var direction : Vector2 = Starting_move_directions
21     if direction && state_machine.check_if_can_move():
22         velocity.x = direction.x * Movement_Speed
23     elif state_machine.current_state != hit_state:
24         velocity.x = move_toward(velocity.x, 0, Movement_Speed)
25
26     move_and_slide()
```

```
1 extends CharacterBody2D
2
3 class_name Player
4
5 @export var Speed : float = 280.0
6 @onready var sprite : Sprite2D = $Sprite2D
7 @onready var animation_tree : AnimationTree = $AnimationTree
8 @onready var state_machine : CharacterStateMachine = $CharacterStateMachine
9
10
11 # Get the gravity from the project settings to be synced with RigidBody nodes.
12 var gravity = ProjectSettings.get_setting("physics/2d/default_gravity")
13 var direction : Vector2 = Vector2.ZERO
14
15 signal facing_direction_changed(facing_right : bool)
16
17 func _ready():
18     animation_tree.active = true
19
20
21 func _physics_process(delta):
22     # Add the gravity.
23     if not is_on_floor():
24         velocity.y += gravity * delta
```

```
28     direction = Input.get_vector("left", "right", "up", "down")
29     if direction.x != 0 && state_machine.check_if_can_move():
30         velocity.x = direction.x * Speed
31     else:
32         velocity.x = move_toward(velocity.x, 0, Speed)
33
34     move_and_slide()
35     update_animation_parameters()
36     update_facing_direction()
37
38
39
40 func update_animation_parameters():
41     animation_tree.set("parameters/Move/blend_position", direction.x)
42
43 func update_facing_direction():
44     if direction.x > 0:
45         sprite.flip_h = false
46     elif direction.x < 0:
47         sprite.flip_h = true
48
49     emit_signal("facing_direction_changed", !sprite.flip_h)
50
```

Different UI Screens

Game over Menu



Main Menu



Code for the UI screens

```
1  extends Control
2
3  ▸ func _process(delta: float) -> void:
4  ▸ ▸ if Input.is_action_just_pressed("ui_accept"):
5  ▸ ▸     get_tree().change_scene_to_file("res://scene/Menu.tscn")
6  ▸     pass
7
```

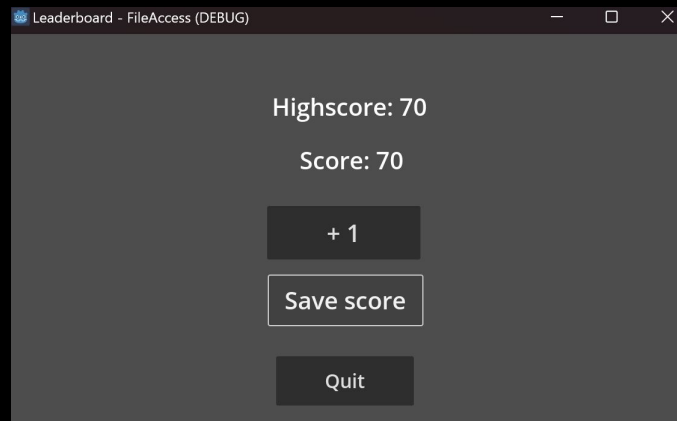
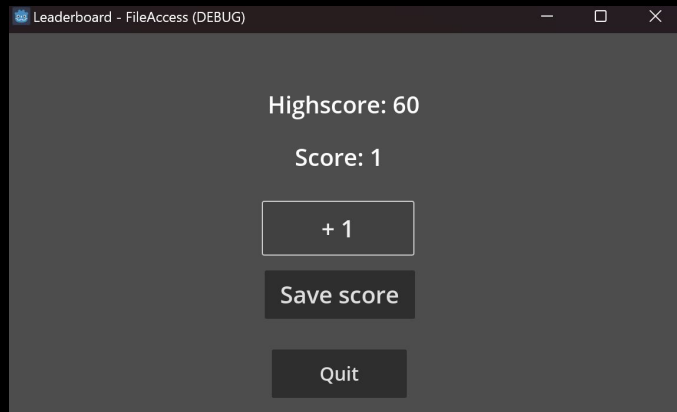
Code for the UI screens

Main menu Code

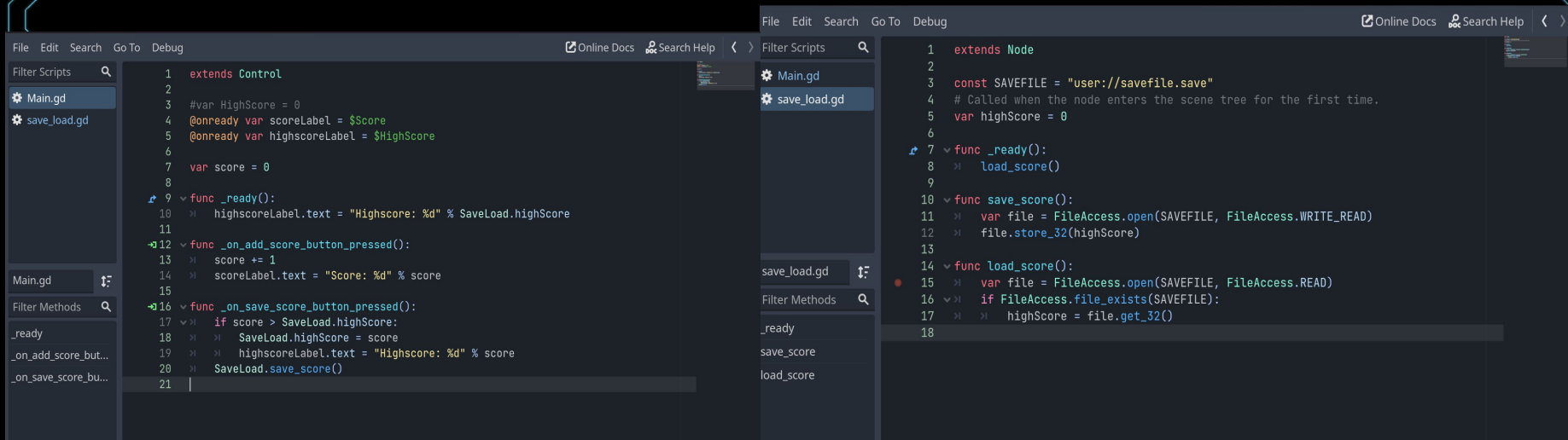
```
1  extends Control
2
3  ▾ func _on_play_pressed():
4      >| get_tree().change_scene_to_file("res://scene/punchman.tscn")
5
6
7
8  ▾ func _on_options_pressed():
9      >| get_tree().change_scene_to_file("res://Options Menu.tscn")
10
11
12 ▾ func _on_quit_pressed():
13     >| get_tree().quit()
14
```

Score System

- Due to the Database and Leaderboard not working, decided to use a feature on Godot called FileAccess.
- FileAccess
 - It allows use to be able to keep track of the high score on the system.
 - It will also show the players current score as they are playing the game.
 - If the user's score is more than the system's highscore, it will update the high score when the player loses.



Score System Code



Onto the Demo