Универзитет „Св. Кирил и Методиј“ во Скопје
ФАКУЛТЕТ ЗА ИНФОРМАТИЧКИ НАУКИ И КОМПЈУТЕРСКО ИНЖЕНЕРСТВО

Additional Homework in Video Game Programming Course

Escape the Fall

Professor: Катарина Тројачанец

Assistant: Славе Темков

Student: Blerona Muladauti

Student ID: 221541

February 2025

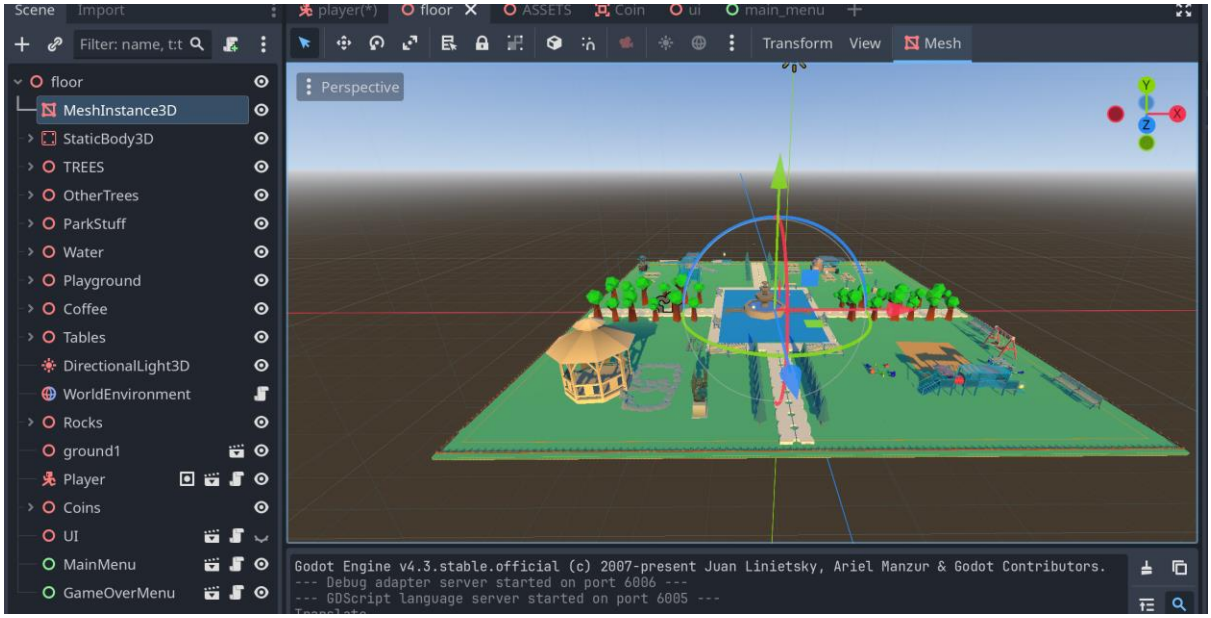
Contents

Introduction

This document provides an overview of the functionalities implemented in the game and explains how they are integrated using the Godot Engine. Escape the fall game is a 3D adventure game where the player controls a fox exploring a park and its assets to find hidden coins before time runs out. The challenge lies in locating all the coins within the time limit.

As the timer approaches the final 5 seconds, the player is made aware of time running out by a blinking effect. If the time runs out before collecting all coins, the player experiences falling through the floor, signalling the game over state. The game utilizes smooth movement mechanics, park assets, and animations to enhance player's movements.

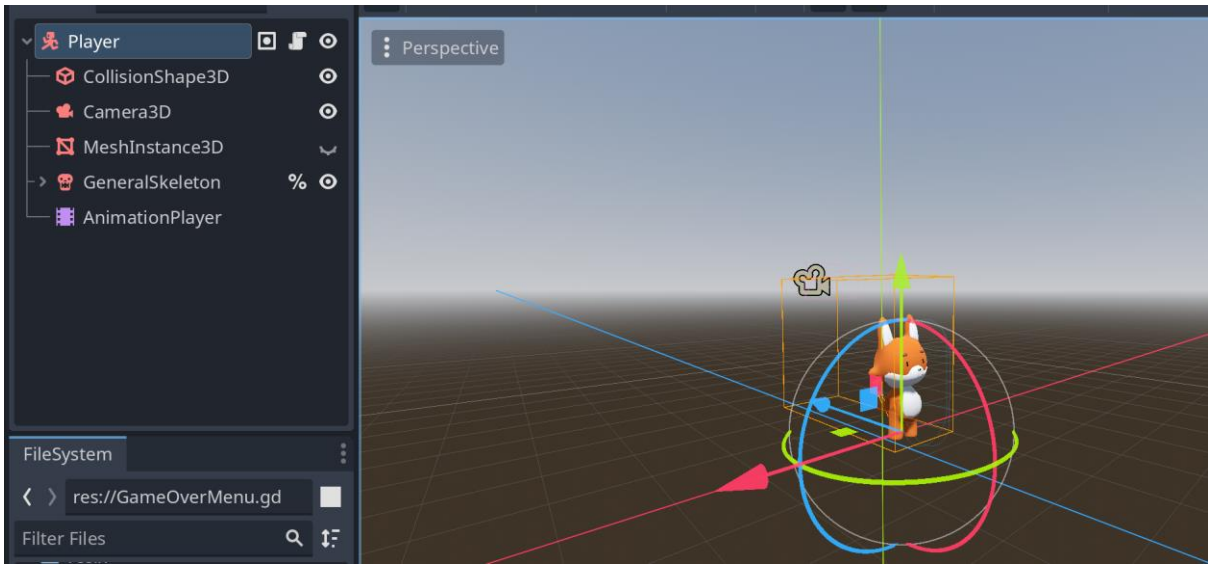
The main scene of the whole game, the root, is the floor node. In the floor node everything else is contained.



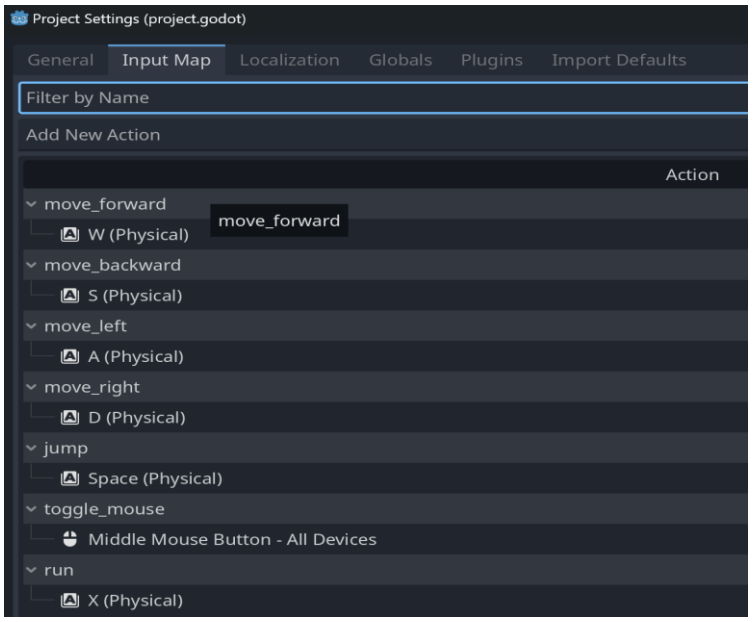
In order for the floor to be borderized, the StaticBody3D handles that. There are 5 instances of collisions blocking fall on the sides and below.

Main Player

The main Player in the game Is a fox. The player can move in 4 directions, forward, backward, left and right, can jump, and sprint.



To do the previous movements, the following actions are taken:
In the Project input map the actions are added along with the Keys needed for the user to execute those actions. Additionally for better game friendly experience, ‘walk’, ‘run’ and ‘jump’ animations are added to the player, handled in the AnimationPlayer node.



The actual movements are handled in the Player script more specifically in the following function:

```
func _physics_process(delta: float) -> void:
```

For the player to be movable, the root node needs to be a CharacterBody3D.

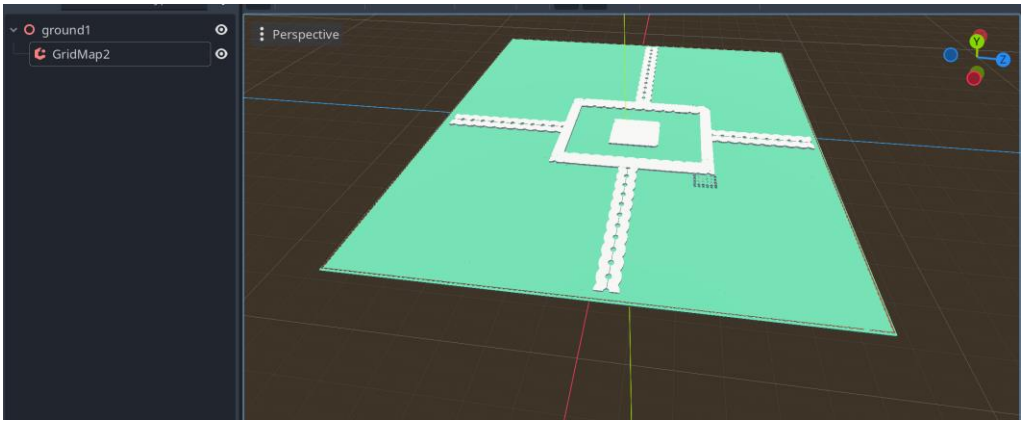
Apart from the options of moving around, user can also move the camera around using the middle mouse button, but this arose different problems having to handle the mouse cursor visibility during game play, so its fixed that in order to view the cursor, user has to click the middle mouse button once, and in order to move the camera around the same button needs to be held down.
For the camera to follow the player and not be static, the Camera3D node has to be a child of the Player root node, hence in the player scene and not the floor scene.

```
extends CharacterBody3D

const BASE_SPEED = 7.0
const SPRINT_SPEED = 12.0
const JUMP_VELOCITY = 4.5
const DOUBLE_JUMP_VELOCITY = 7.0
const MOUSE_SENSITIVITY = 0.2

var gravity = ProjectSettings.get_setting("physics/3d/default_gravi
var pitch: float = 0.0
var can_double_jump: bool = false
var speed: float = BASE_SPEED
```

In this part, the initial values of the player movement parameters are defined.

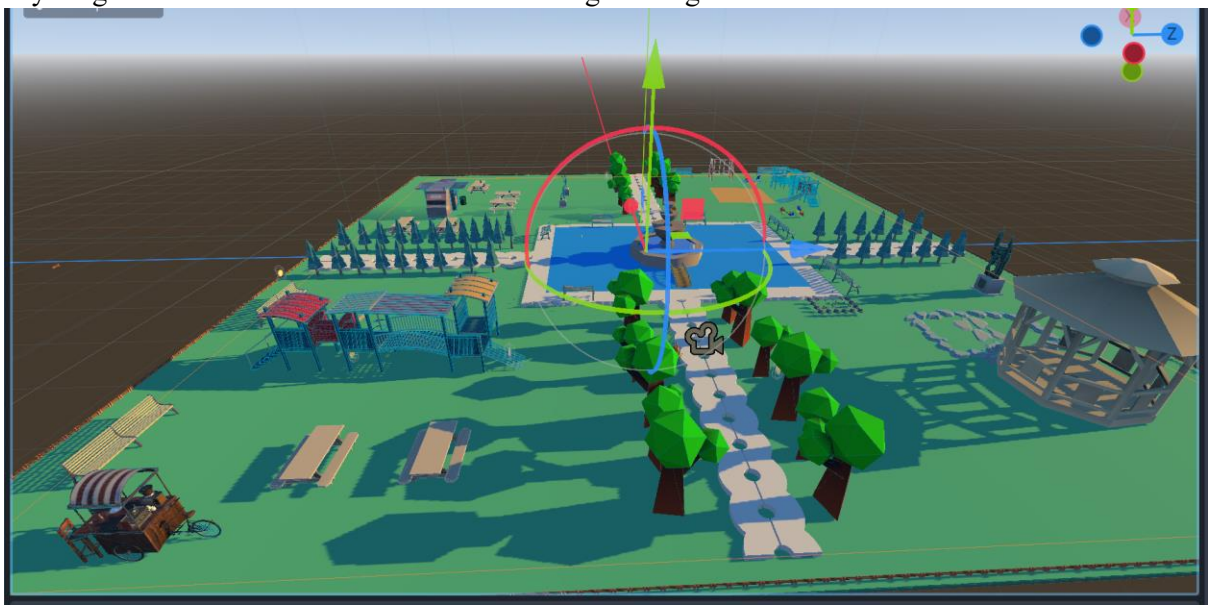
Park

To draw the basis of the floor, (the grass and tiles) Godot's Grid Map is used, and it looks like this:



This is made as a separate scene and then attached to the main floor scene I previously mentioned in the Introduction part.

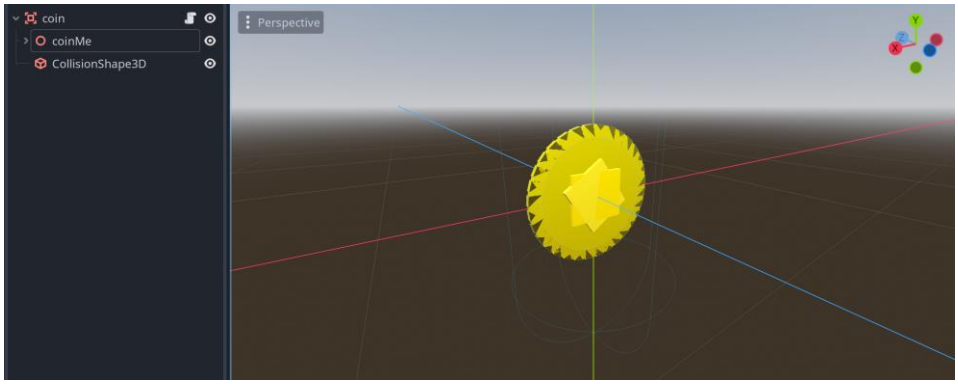
In the root floor node I organized the park assets in different nodes of their own. Important in most assets was that they needed to be reimported with the physiscs button enabled so that player or anything can collide with them but not be able to go through.



This is the final look of the floor.

Coin

I designed the coin using Blender and then imported the scene in godot res folder. In the floor node for better organization there is a coin node, with 10 children each a coin scene. For the coin collision to work with the player, all coins were added in a group, and it must have a collision shape.



This is the coin design I made using Blender

For the collision to work with the player, player needs to be added in the group called Player.

```
61  func _on_body_entered(body: Node) -> void:
62      if body.is_in_group("Player"):
63          print("Coin collected!")  #  Debugging print
64
65
66
67          emit_signal("coin_collected")  #  Emit signal for GameManage
68          queue_free()  # Remove coin from scene after collection
69
```

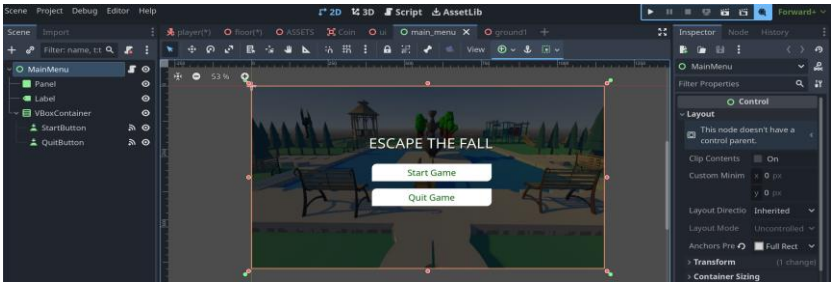
In the Coin script there is a signal defined to handle the coin collection for the coin count that is needed to win or lose the game

```
func connect_coins():
    var coins = get_tree().get_nodes_in_group("Coins")  # ✅ Find a

    for coin in coins:
        if coin.has_signal("coin_collected"):  # ✅ Check if signal
            if not coin.is_connected("coin_collected", self.add_tre
                coin.connect("coin_collected", self.add_treasure)
            print("✅ Connected coin at", coin.global_transform.ori
        else:
            print("⚠ Error: Coin at", coin.global_transform.origin,

    print("Connected", len(coins), "coins to GameManager.")  # ✅ D
```

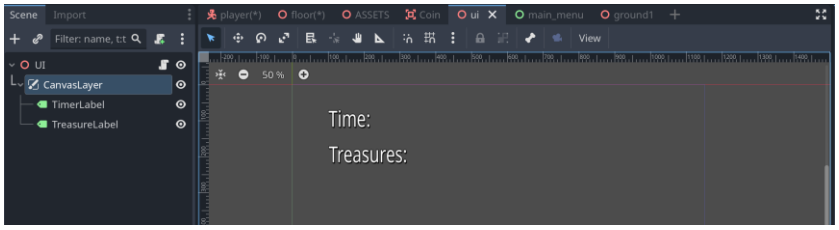
This is in the Game Manager script where the coin connection is handled with the signal emition previously defined in the coin script.

Menu

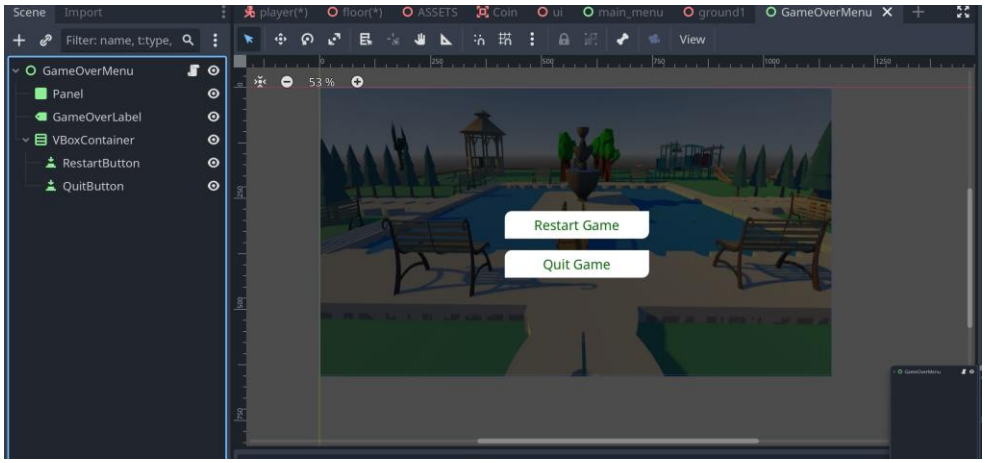
The menus are of type Canvas Layer and they contain Labels and Buttons and a panel for the background image. In Escape the fall there are two menus, start menu and end menu.
The start menu contains two buttons, Start Game and Quit Game and a label with the game title on top. The end menu is very similar to the start menu except there is the restart button and instead of the game title it either says You won or you lost based on the game outcome. The menus design is done in the 2D panel in Godot using all the options in the inspector panel.
The logic of the menus appearance and disappearance and the button functions is handled in the main_menu.gd and GameOverMenu.gd scripts.
When the game is run initially the start menu is visible and the rest are all invisible. Then when player clicks the start button, start menu disappears and the game ui appears in my case the floor and the timer and treasure labels, and mouse cursor disappears until player clicks the middle mouse button. Lastly, when player wins or loses, the game ui disappears and the game over menu appears with its options. The following images are the menus themselves in the godot engine:



Start menu



Time and Treasure labels in ui



End Game Menu

Timer and Treasure count

Game logic functionalities are handled in the GameManager script. The _process function in the script handles the timer logic and treasure count.

```gdscript
func _process(delta: float) -> void:
    if game_ui.visible:
        time_left -= delta

        # Start blinking in the last 5 seconds
        if time_left <= 5 and time_left > 1 and player:
            var blink_rate = 5  # Blink every 0.5 seconds
            player.visible = int(time_left * blink_rate) % 2 == 0

        # Disable collision in the last second
        if time_left <= 1 and player:
            player.visible = true  # Ensure visibility before falling
            var collider = player.get_node("CollisionShape3D")
            if collider:
                collider.disabled = true  # Turn off collision

        if time_left <= 0:
            time_left = 0
            game_over("Time's up! Game Over.")

        update_labels()
```

After the start menu, when player starts game the gameui appears and timer countdown starts. In the last 5 seconds as an alarm to the user about the time the player starts blinking and when time reaches <=1, the collision of the player is disabled, hence he falls through the floor and game lost menu appears.

```gdscript
func update_labels() -> void:
    if treasure_label:
        treasure_label.text = "Treasures: " + str(collected) + " / " + str(requir
    if timer_label:
        timer_label.text = "Time Left: " + str(round(time_left))
        if time_left <= 10:
            timer_label.add_theme_color_override("font_color", Color.RED)
        else:
            timer_label.add_theme_color_override("font_color", Color.WHITE)

func add_treasure() -> void:
    collected += 1
    update_labels()
    print("Treasure collected! New count:", collected)  # ✅ Debugging print
    if collected >= required_treasure:
        game_over("Congratulations, you won!")
```

Here the treasure count is handled. Where uppon collision with coin the label is updated and count incremented. When collected equals required coins, game is won. And Game won menu appears.

Game Play

Player Movement

The player can move in four directions (forward, backward, left, and right) using keyboard input. (AWDS) The movement speed varies based on whether the player is walking or sprinting.

Jumping & Double Jump

The player can jump while on the ground and can perform a second jump (double jump) in mid-air. Gravity is applied to simulate realistic movement. (SPACE / DOUBLE SPACE)

Sprinting

The player can sprint by holding the designated sprint button, increasing their movement speed. (X)

Camera Control

The camera follows the player and allows for free look using mouse movement. The pitch (vertical rotation) is clamped to prevent excessive movement. (MIDDLE MOUSE BUTTON)

Coin Collection and Timer System

Players must find hidden coins scattered throughout the park before the timer runs out. The game provides visual feedback in the last 5 seconds by **blinking effects** to alert the player.

Game Over Mechanism

If the time runs out, the player experiences a **fall-through-floor effect**, ending the game.

Initially the start menu appears, when user starts game timer starts countdown and player movement is enabled. Player has to follow the park and find the scattered coins, in the last 10 seconds as a prewarning, the timer label turns red, and in the last 5 seconds the player starts blinking as a final warning. When timer ends player falls through the ground, and game is lost. User can either quit or restart the game.

References

https://www.mixamo.com/

https://sketchfab.com/

https://kenney.nl/

https://www.blender.org/

Github Link:

https://github.com/mblerona/Video-Games-Programming/tree/main/BonusHomework

Screen Recording is also included in the github repo