An-Najah National University

Computer Science Apprenticeship Program

Faculty of Engineering and AI

Race-Game Project Report

Prepared by:

- Ahmad Namrouti
- Faris Sahili
- Hala Khalifeh
- Mayar Basheer

Course: Game & Artificial Intelligence

Supervised by: Dr. Zaina Saadeddin

Summer Semester, Monday, 25 August 2025

I. Introduction

The **Race Game** is a 2D endless runner built with **Godot Engine 4.x** and **GDScript** as part of the *Game & Artificial Intelligence* course.

The player guides a character through dynamically generated obstacles (stumps, barrels, rocks, and birds) while the game gradually increases in speed and difficulty.

The game consists of **two levels**:

- **Level 1** introduces the mechanics and ends after reaching a set finish score, after which the player can transition to Level 2.
- **Level 2** is more challenging, with faster movement, additional obstacles (including flying birds), and no finish score; the goal is to survive as long as possible.

A **scoring and HUD system** tracks the player's score, remaining distance in Level 1, and high scores in Level 2.

In addition, the project integrates an **AI Coach (NPC)** powered by a local LLM model, which provides short motivational or instructional words during gameplay. This feature transforms the experience from a simple runner into an interactive and supportive challenge.

The project development was divided into three modules:

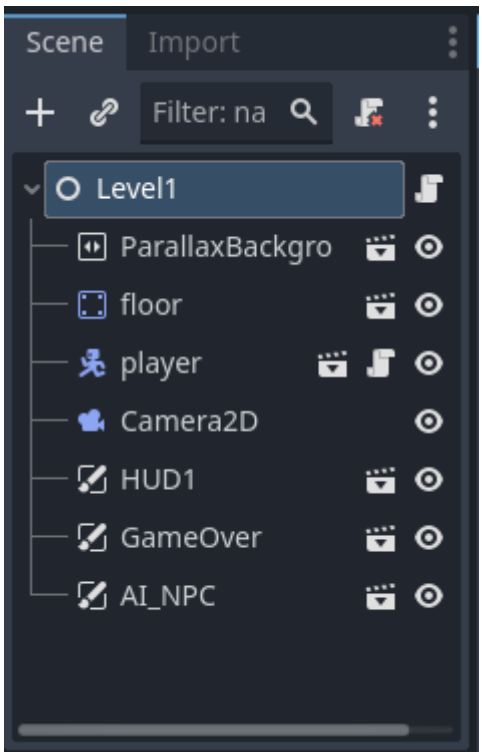
1. **2D Game Design:** designing levels, environments, and HUD.
2. **Game Development (GDScript):** player physics, obstacle spawning, scoring, and level transition.
3. **AI Integration:** adding a conversational NPC that reacts to gameplay events and encourages the player.

II. Scripts Used

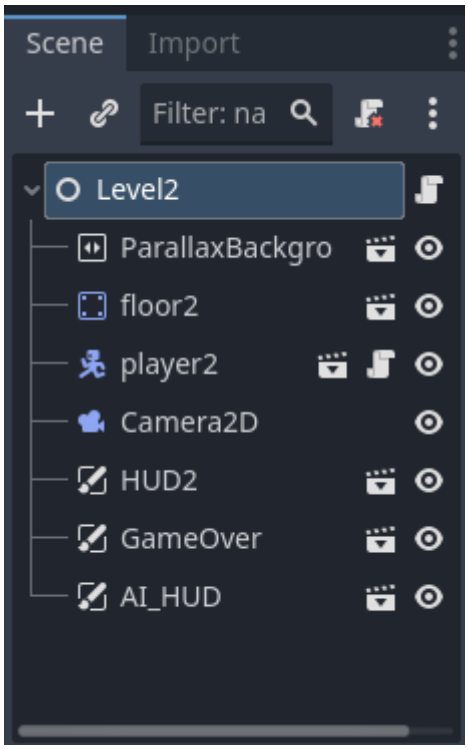
- **level1.gd**—Controls Level 1 gameplay: obstacle spawning, scoring, difficulty scaling, and transition to Level 2 after reaching the finish score.
- **level2.gd**—Controls Level 2 gameplay: faster pace, harder obstacles, flying birds, high score tracking, and endless survival mode.
- **player.gd**—Handles Level 1 player movement and physics (running, jumping, crouching, collisions).
- **player2.gd**—Handles Level 2 player movement and physics with similar logic adapted for higher difficulty.
- **ai_npc.gd**—Manages the AI Coach (NPC) that periodically or contextually gives short motivational words.
- **bird.gd**—Defines bird movement for flying obstacles that appear in Level 2.

III.Module 1: 2D Game Design

- **Concept:** A 2D endless runner with two progressively harder levels.
- **Level 1:** Beginner-friendly with ground obstacles (stumps, barrels). Ends at a fixed score (20,000).
- **Level 2:** Faster speed, more frequent obstacles & adding rocks, and flying birds at varied heights. No finish score—survival is the goal.
- **Scenes:**
  - Level 1: `player`, `floor`, `hud1`, `parallax_background`, `game_over`, `ai_npc`, `music`, plus obstacle scenes (`stump`, `barrel`, `rock`).
  - Level 2: `player2`, `floor2`, `hud2`, `bird`, and the same base elements with increased difficulty.
- **HUDs:**
  - **HUD1:** Shows score and remaining distance.
  - **HUD2:** Shows score and tracks high scores.



Level 1



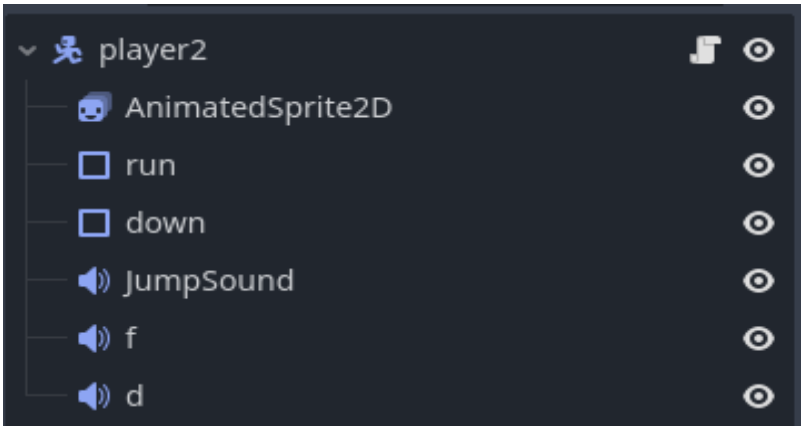
Level 2

Ground, background, and environment.

Scoring system

IV.Module 2: Game Development

- Implemented **player physics** (gravity, jump, crouch, and animations).




```gdscript
func _physics_process(delta: float) -> void:
    if not started or finished:
        velocity = Vector2.ZERO
        $run.disabled = false
        $AnimatedSprite2D.play("idle")
        move_and_slide()
        return
    # Normal gameplay
    velocity.y += GRAVITY * delta
    if is_on_floor():
        $run.disabled = false
        if Input.is_action_pressed("ui_accept"):
            velocity.y = JUMP_SPEED
            $JumpSound.play()
        elif Input.is_action_pressed("ui_down"):
            $AnimatedSprite2D.play("down")
            $run.disabled = true
        else:
            $AnimatedSprite2D.play("run")
    else:
        $AnimatedSprite2D.play("jump")
    move_and_slide()
```

- Added **dynamic obstacle spawning** with randomness in spacing and type.


```
unc generate_obs() -> void:
    if score >= 18500:
        return

    if obstacles.is_empty() or last_obs.position.x < score + randi_range(300, 500):
        var obs_type = obstacle_types[randi() % obstacle_types.size()]
        var obs
        var max_obs = int(difficulty) + 1
        for i in range(randi() % max_obs + 1):
            obs = obs_type.instantiate()
            var obs_height = obs.get_node("Sprite2D").texture.get_height()
            var obs_scale = obs.get_node("Sprite2D").scale
            var obs_x = screen_size.x + score + 100 + (i * 100)
            var obs_y : int = screen_size.y - ground_height - (obs_height * obs_scale.y / 2) + 5
            last_obs = obs
            add_obs(obs, obs_x, obs_y)
```

- Built a **scoring system**:
    - Level 1: score increases with distance; HUD shows remaining distance to finish.
    - Level 2: faster scoring with high score tracking.

- Implemented **level transition**:
  - Level 1 ends at 20,000 points.
  - The player presses **Enter** to move to Level 2.

```
unc level_finished() -> void:
    game_running = false
    level_completed = true
    $player.get_node("AnimatedSprite2D").play("idle")  # set idle
    $HUD1/S.visible = true
    $HUD1/S.text = "Level Complete!"
    print("Level 1 finished! Ready to go to Level 2?")

unc load_level2() -> void:
    var level2_scene = preload("res://scenes/level2/level2.tscn")
    get_tree().change_scene_to_packed(level2_scene)
```

- Developed **difficulty scaling** by increasing speed and obstacle count as the score rises.

```
if game_running:
    speed = START_SPEED + score / SPEED_MODIFIER
    if speed > MAX_SPEED:
        speed = MAX_SPEED
    adjust_difficulty()
```

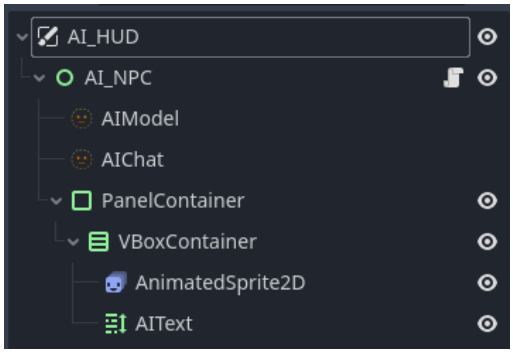
- Added **a Game Over screen** with a restart button.
- Cleaned up off-screen obstacles to optimize performance.

V. Module 3: AI Integration (AI Coach)

Goal
To make gameplay more interactive by providing motivational or instructional words through an AI coach.

Implementation

- Integrated **ai_npc.gd**, which uses a local **Gemma-2** LLM model by using:
    - NobodyWhoModel
    - NobodyWhoChat
    - Signals (response_updated, response_finished)
- Periodically sends very short motivational messages (“Go!”, “Watch out!”, “Nice jump!”).
- Event-based reactions:
    - **On scoring milestones** → encouragement.
    - **On death** → supportive message.
    - **On level completion** → congratulatory message.



VI.Lessons Learned

- **Godot 4.x** provided powerful tools for building 2D physics and scene management.
- Using **signals** simplified communication between the AI NPC and the game logic.
- Organizing scenes per level kept the project structured.
- Balancing speed, difficulty, and obstacle frequency required repeated playtesting.
- AI integration added personality and made the game more engaging.

VII.Future Enhancements

- Add more levels with new environments (city, desert, space).
- Expand AI Coach to give adaptive advice (e.g., after repeated failures at the same obstacle).
- Improve graphics, animations, and sound for better immersion.
- Implement persistent high score saving across sessions.
- Add **a multiplayer mode** with AI encouragement for each player.
- Publish builds on more platforms (Android, iOS).

VIII.Conclusion

The **Race Game** project successfully combines 2D game design, physics-based gameplay, and AI-driven encouragement.
Level 1 introduces mechanics with a clear completion goal, while Level 2 offers endless challenge and score competition.
The AI Coach transforms the game into an interactive experience by motivating and guiding players in real time.

IX.Team Contribution

All team members (Ahmad Namrouti, Faris Sahili, Hala Khalifeh, and Mayar Basheer) contributed to design, coding, and testing. Collaboration ensured smooth progress and integration across game logic, AI, and level design.