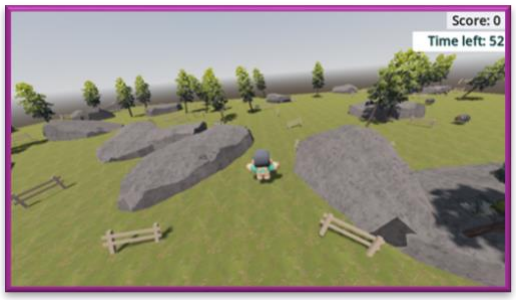
Дополнителна домашна – 3D игра во Godot

Опис на играта

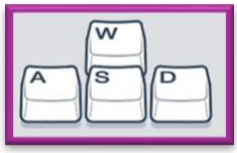
Оваа игра е 3D авантура каде играчот контролира карактер кој се движи низ сцената, собира предмети и достигнува крајна цел за да победи. Играта вклучува различни елементи како терен, пречки, анимации, временско ограничување и кориснички интерфејс (UI) кој ја прикажува моменталната состојба на играчот.

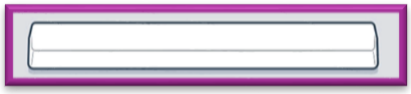
Како се игра играта

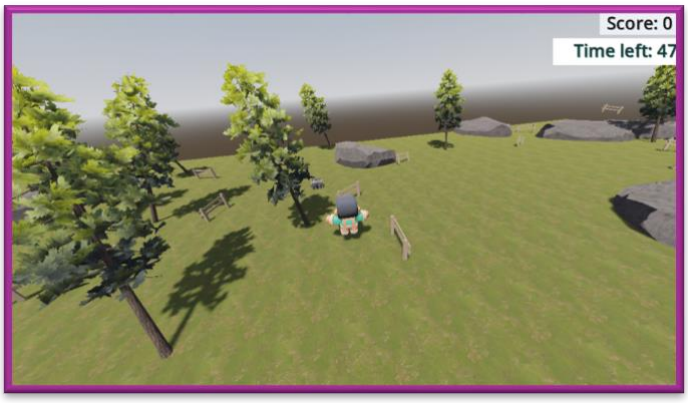
1. Играчот почнува на стартна позиција. Играчот има една минута да ги собере сите богатства. Неговиот моментален Score е 0, бидејќи сеуште нема собрано ниту едно богатство.



2. Со користење на копчињата за движење (WASD), играчот се движи низ сцената.
    1. W – играчот се движи **нанапред**
    2. A – играчот се движи **лево**
    3. S - играчот се движи **наназад**
    4. D – играчот се движи **десно**
    5. Space – играчот **скока**







3. Целта е да се соберат сите предмети(вкупно 6 предмети) пред да истече времето (60 секунди).

4. Ако играчот ги собере сите предмети, се прикажува порака "You Won!".

5. Ако времето истече, се прикажува "Game Over".

Главни функционалности

1. **Движење на играчот**

   ⇒ Карактерот може да се движи напред, назад, лево и десно со користење на копчињата W, A, S, D или стрелките.
   ⇒ Поддржано е скокање и гравитација за реалистично движење.
   ⇒ Имплементирана е `move_and_slide()` за движењето да е smooth и колизии.

2. **Колизии**

   ⇒ Пречките како карпи и огради се дефинирани со `StaticBody3D` и `CollisionShape3D` за да спречат играчот да поминува низ нив.
   ⇒ Играчот има `CharacterBody3D` за да детектира судири со околината.

3. **Собирање предмети**

⇒ Предметите што играчот треба да ги собере се дефинирани како **`Area3D`** со сигнал **`body_entered`**.
⇒ Кога играчот ќе собере предмет, предметот се отстранува од сцената и резултатот (**`score`**) се зголемува.

4. **UI елементи**

⇒ **`Label`** за прикажување на резултатот и преостанатото време.
⇒ **`Control`** node за приказ на пораки како **"Game Over"** и **"You Won!"**.

5. **Тајмер**

⇒ **`Timer`** ја следи времетраењето на играта и одредува кога истекува времето.
⇒ По истекот на времето, ако не се соберат сите предмети, се прикажува **"Game Over"**.

Објаснување на кодот

Движење на играчот

```gdscript
func _physics_process(delta):
    # Update timer label
    if game_timer and timer_label:
        timer_label.text = "Time left: " + str(int(game_timer.time_left))

    # Apply gravity
    if not is_on_floor():
        velocity.y -= gravity * delta

    # Handle movement
    var input_dir = Vector3.ZERO
    if Input.is_action_pressed("move_forward"):
        input_dir += transform.basis.z
    if Input.is_action_pressed("move_backward"):
        input_dir -= transform.basis.z

    # Handle rotation
    if Input.is_action_pressed("move_left"):
        rotation_degrees.y += rotation_speed * delta
    if Input.is_action_pressed("move_right"):
        rotation_degrees.y -= rotation_speed * delta

    # Apply movement
    input_dir = input_dir.normalized()
    velocity.x = input_dir.x * speed
    velocity.z = input_dir.z * speed

    if Input.is_action_just_pressed("jump") and is_on_floor():
        velocity.y = jump_force

    move_and_slide()

    # Check if player fell off
    if global_transform.origin.y < fall_limit:
        game_over_func()
```

Оваа функција управува со физиката на играчот и се извршува во секој кадар. Прво, го ажурира текстот на тајмерот за да го прикаже преостанатото време во секунди. Потоа, проверува дали играчот е на подот и ако не е, применува гравитација за да го турне надолу. Движењето на играчот се контролира со проверка на притиснатите копчиња (напред, назад, лево, десно) и ротација околу оската Y. Ако играчот скокне, тогаш се применува сила на вертикалната брзина (`velocity.y`). На крај, `move_and_slide()` се повикува за да го помести играчот и да се избегнат судири. Доколку играчот падне под одредено ниво, се повикува `game_over_func()`.

Детекција на судири

```
func _on_treasure_collected(body, treasure):
    if body == self:
        treasure.queue_free()
        await get_tree().create_timer(0.1).timeout
        score += 1
        collected_treasures += 1
        score_label.text = "Score: " + str(score)

        # If all treasures are collected, win the g
        if collected_treasures == total_treasures:
            win_game()
```

Кога играчот ќе допре предмет, функцијата _on_treasure_collected() проверува дали телото што го допрело предметот е играчот. Ако е така, предметот се брише од сцената со queue_free(), а по кратко чекање, резултатот (score) се зголемува за еден, а бројот на собрани предмети (collected_treasures) се ажурира. Потоа, резултатот се прикажува на екранот преку score_label. Доколку бројот на собрани предмети стане еднаков со вкупниот број предмети во играта, функцијата win_game() се повикува за да се означи крајот на играта со победа.

Управување со тајмерот

```
func _on_time_up():
    # If time runs out before collecting all treasures, game over
    if collected_treasures < total_treasures:
        game_over_func()
    else:
        win_game()
```

Функцијата `_on_time_up()` се активира кога ќе истече времето зададено од тајмерот. Таа проверува дали играчот успеал да ги собере сите предмети пред крајот на времето. Доколку сите предмети се собрани, тогаш играчот победува преку функцијата `win_game()`. Во спротивно, се повикува `game_over_func()`, што означува пораз на играчот.

Победа на играчот

```
func win_game():
    win_ui.visible = true
    timer_label.visible = false  # Hide timer
    if game_timer:
        game_timer.stop()  # Stop the timer

    # Show Restart Button
    var restart_button = get_tree().get_root().find_child("RestartButton", true, false)
    if restart_button:
        restart_button.visible = true  # Make it visible
        restart_button.connect("pressed", _on_restart_button_pressed, CONNECT_DEFERRED)

        get_tree().paused = true  # Pause the game
```

Кога играчот ќе победи, функцијата `win_game()` го прави UI-то за победа (`win_ui`) видливо и го скрива тајмерот (`timer_label`) за да не продолжи да се брои времето. Ако тајмерот работи, тој се стопира за да спречи дополнителни промени. Дополнително, во сцената се бара копчето за рестарт и се прави видливо. За да може копчето да работи правилно, се поврзува со функцијата `_on_restart_button_pressed()`, што ќе овозможи рестартирање на играта. Конечно, играта се паузира со `get_tree().paused = true`, спречувајќи понатамошни движења на играчот.

Главната функција – ready()

```gdscript
func _ready():
    var restart_button = get_tree().get_root().find_child("RestartButton", true, false)
    if restart_button:
        restart_button.visible = false
    # Ensure GameTimer is using the correct time
    if game_timer:
        game_timer.stop()  # Stop any existing timer
        game_timer.wait_time = 60  # Ensure it uses the correct time
        game_timer.start()
        print("Game Timer Set to:", game_timer.wait_time)  # Debugging

        if not game_timer.is_connected("timeout", _on_time_up):
            game_timer.connect("timeout", _on_time_up)

    # Hide UI elements initially
    game_over.visible = false
    win_ui.visible = false


    # Find all treasures
    var treasures = get_tree().get_nodes_in_group("Treasures")
    total_treasures = treasures.size()

    # Connect treasure collision detection
    for treasure in treasures:
        if treasure.has_node("TreasureArea"):
            var area = treasure.get_node("TreasureArea")
            if area.has_signal("body_entered") and not area.is_connected("body_entered", _on_treasure_collected):
                area.connect("body_entered", _on_treasure_collected.bind(treasure))
```

Функцијата `_ready()` се повикува веднаш кога сцената ќе се вчита. Во оваа функција, најпрво се проверува дали постои копчето за рестарт (`RestartButton`) и ако постои, се прави невидливо на почетокот на играта. Потоа, се иницијализира тајмерот (`GameTimer`), кој се ресетира, се поставува на 60 секунди и се стартува. Се додава и сигнал кој ќе ја повика функцијата `_on_time_up()` кога ќе истече времето. Исто така, сите UI елементи поврзани со победа или пораз се скриени за да не бидат видливи од самиот почеток. Конечно, се бараат сите предмети за собирање (кои припаѓаат на групата `"Treasures"`) и се зачувува нивниот вкупен број. Секој предмет има `TreasureArea`, кој кога ќе биде допрен од играчот, ја повикува функцијата `_on_treasure_collected()`.

Пораз на играчот

```gdscript
func game_over_func():
    game_over.visible = true
    #restart_button.visible = true
    get_tree().paused = true
```

Кога играчот ќе изгуби, функцијата `game_over_func()` го прави UI-то за пораз (`game_over`) видливо. Потоа, ја паузира играта со `get_tree().paused = true` за да спречи понатамошно движење на играчот. Ова овозможува играчот да види дека изгубил и да одлучи дали ќе ја рестартира играта.

Проектна структура

```
game_project/
├── assets/                 # Текстури, модели и други ресурси
├── scenes/                 # Godot сцени
```

```
│   ├── main.tscn          # Главна сцена на играта
│   ├── player.tscn        # Играчки карактер
│   ├── environment.tscn   # Дрвја, камења и терен
│   └── UI.tscn            # UI елементи (резултат, тајмер, пораки)
├── scripts/               # GDScript датотеки
│   ├── player.gd          # Логика за движење на играчот
│   ├── game_manager.gd    # Управување со состојбата на играта
│   └── ui_manager.gd      # UI ажурирања
└── project.godot          # Godot проект датотека
```

Референци

https://youtu.be/OoVAtGHEgjA?si=BWWSx60-QRoPhFnh

https://youtu.be/AoGOIiBo4Eg?si=1wJXEEOlAoCCI06h

https://youtu.be/ke5KpqcoiIU?si=VVPVlH_h-Cuytxd3

https://youtu.be/2W4JP48oZ8U?si=YZfAgU7UvT9jubX2