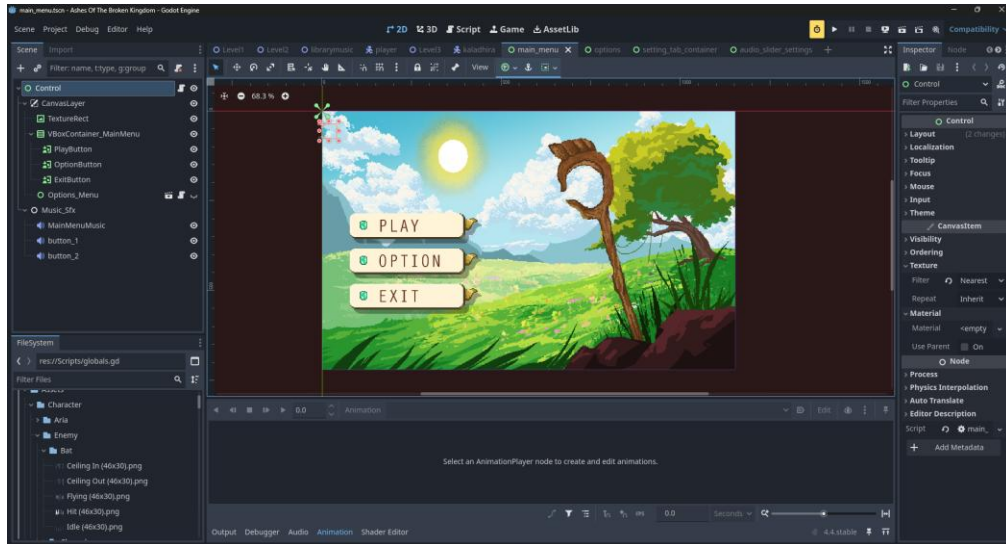


LAMPIRAN

1 Main Menu



```
class_name MainMenu
extends Control

@onready var play_button: TextureButton =
$CanvasLayer/VBoxContainer_MainMenu/PlayButton
@onready var option_button: TextureButton =
$CanvasLayer/VBoxContainer_MainMenu/OptionButton
@onready var exit_button: TextureButton =
$CanvasLayer/VBoxContainer_MainMenu/ExitButton

@onready var options_menu: OptionMenu = $CanvasLayer/Options_Menu as
OptionMenu
@onready var v_box_container_main_menu: VBoxContainer =
$CanvasLayer/VBoxContainer_MainMenu

@onready var star_level = preload("res://Scenes/Level1.tscn") as
PackedScene

func _ready():
    handle_connecting_signal()
    Librarymusic.MainMenuMusic()

func on_play_pressed() -> void:
    Librarymusic.button1()
    Gamestate.reset()
    await get_tree().create_timer(0.3).timeout
    Globals.load_start_time = Time.get_ticks_msec()
    get_tree().change_scene_to_packed(star_level)
    var load_duration = Time.get_ticks_msec() - Globals.load_start_time
    print("Waktu load dari menu ke level 1: %d ms" % load_duration)

func on_option_pressed() -> void:
    Librarymusic.button1()
    v_box_container_main_menu.visible = false
    options_menu.set_process(true)
    options_menu.visible = true
```

```

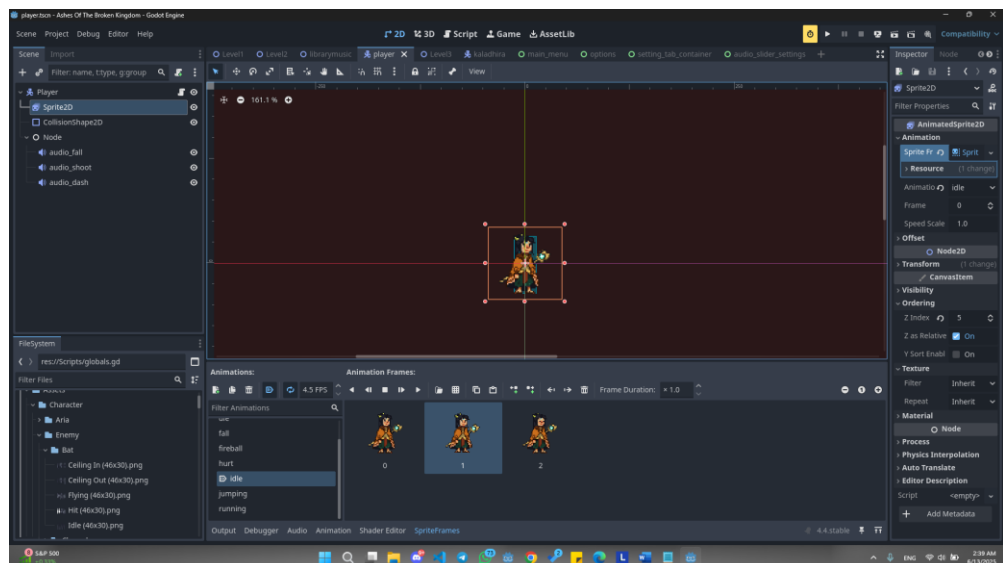
func on_exit_pressed() -> void:
    Librarymusic.button2()
    await get_tree().create_timer(0.5).timeout
    get_tree().quit()

func on_exit_option_menu() -> void:
    Librarymusic.button2()
    v_box_container_main_menu.visible = true
    options_menu.visible = false

func handle_connecting_signal() -> void:
    play_button.button_down.connect(on_play_pressed)
    option_button.button_down.connect(on_option_pressed)
    exit_button.button_down.connect(on_exit_pressed)
    options_menu.exit_options_menu.connect(on_exit_option_menu)

```

2 Kontrol Karakter



extends CharacterBody2D

```

@export var run_speed = 250.0
@export_range(0, 1) var acceleration = 0.1
@export_range(0, 1) var deceleration = 0.1
@export var jump_force = -550.0
@export_range(0, 1) var decelerate_on_jump_release = 0.5

@export var dash_speed = 1000.0
@export var dash_max_distance = 300.0
@export var dash_curve : Curve
@export var dash_cooldown = 1.0

@export var wall_jump_force_x = 300.0
@export var wall_jump_force_y = -500.0

var took_damage = false
var can_move = true

var gravity = ProjectSettings.get_setting("physics/2d/default_gravity")

```

```

var is_dashing = false
var dash_start_position = 0.0
var dash_direction = 0.0
var dash_timer = 0.0
var can_air_dash = true

const MAGIC_PROJECTILE = preload("res://Scenes/MagicProjectile.tscn")

var jump_count = 0
var max_jumps = 2
var is_jumping = false

signal shoot(is_flip)

var is_shooting = false

@onready var sprite_2d: AnimatedSprite2D = $Sprite2D
@onready var game_manager: Node = $"../GameManager"

func _ready() -> void:
    sprite_2d.animation_finished.connect(on_animation_finish)
    sprite_2d.animation_changed.connect(on_animation_changed)

func on_animation_finish():
    if sprite_2d.animation == "fireball":
        is_shooting = false

func on_animation_changed():
    if sprite_2d.animation != "fireball":
        is_shooting = false

func hurt():
    #self.visible = false
    can_move = false
    Librarymusic.AudioHurt()
    sprite_2d.play("hurt")
    await get_tree().create_timer(0.5).timeout #waktu respawn

    can_move = true

    await get_tree().create_timer(0.5).timeout #waktu respawn
    took_damage = false

func take_damage():
    if took_damage == false:
        took_damage = true #set damage
        game_manager.update_player_health(-1)
        if Gamestate.player_health > 0:
            hurt() #respawn Player
        else:
            can_move = false
            Librarymusic.AudioHurt()
            Librarymusic.AudioGameOver()
            sprite_2d.play("die")
            await
    get_tree().create_timer(2.5).timeout
    game_manager.game_over()

func _physics_process(delta: float) -> void:
    if is_on_floor():
        can_air_dash = true

```

```

        jump_count = 0
        is_jumping = false

    if not is_on_floor() and not is_dashing:
        velocity.y += gravity * delta

    # cek collision jika mengenai duri/spike
    for i in get_slide_collision_count():
        var collision = get_slide_collision(i)

        var collider = collision.get_collider()
        if collider is Node and collider.name == "TileMapLayer_Spike":
            take_damage()

    # =====
    # WALL JUMP & NORMAL JUMP
    # =====
    if Input.is_action_just_pressed("jump"):
        if is_on_wall() and not is_on_floor():
            # WALL JUMP
            var wall_normal = Vector2.ZERO
            if get_slide_collision_count() > 0:
                var col = get_slide_collision(0)
                wall_normal = col.get_normal()

            velocity.y = wall_jump_force_y
            velocity.x = wall_jump_force_x * wall_normal.x
            sprite_2d.play("jumping")
            Librarymusic.AudioJump()
            jump_count += 1
        elif jump_count < max_jumps:
            # NORMAL / DOUBLE JUMP
            velocity.y = jump_force
            sprite_2d.play("jumping")
            Librarymusic.AudioJump()
            is_jumping = true
            jump_count += 1

    if Input.is_action_just_released("jump") and velocity.y < 0:
        velocity.y *= decelerate_on_jump_release

    # =====
    # DASH
    # =====
    var direction := Input.get_axis("left", "right")
    if not is_dashing:
        if direction:
            velocity.x = move_toward(velocity.x, direction *
run_speed, run_speed * acceleration)
        else:
            velocity.x = move_toward(velocity.x, 0, run_speed *
deceleration)

    if Input.is_action_just_pressed("dash") and direction and dash_timer
<= 0 and (is_on_floor() or can_air_dash):
        is_dashing = true
        dash_start_position = position.x
        dash_direction = direction
        dash_timer = dash_cooldown
        if not is_on_floor():
            can_air_dash = false

```

```

        if is_dashing:
            var current_distance = abs(position.x - dash_start_position)
            if current_distance >= dash_max_distance or is_on_wall():
                is_dashing = false
            else:
                velocity.x = dash_direction * dash_speed *
dash_curve.sample(current_distance / dash_max_distance)
                velocity.y = 0
                #audio_dash.play()
                sprite_2d.play("dashing")

        if dash_timer > 0:
            dash_timer -= delta

        # =====
        # ANIMASI
        # =====
        if not is_dashing:
            if not is_on_floor():
                if is_jumping and velocity.y > 0:
                    sprite_2d.play("fall")
                    is_jumping = false
            elif abs(velocity.x) > 5 and can_move:
                #
                tambahkanmusic disini
                sprite_2d.play("running")
            else:
                if not is_shooting and can_move:
                    sprite_2d.play("idle")

        # Flip karakter
        if direction != 0:
            sprite_2d.flip_h = direction < 0

        #shoot

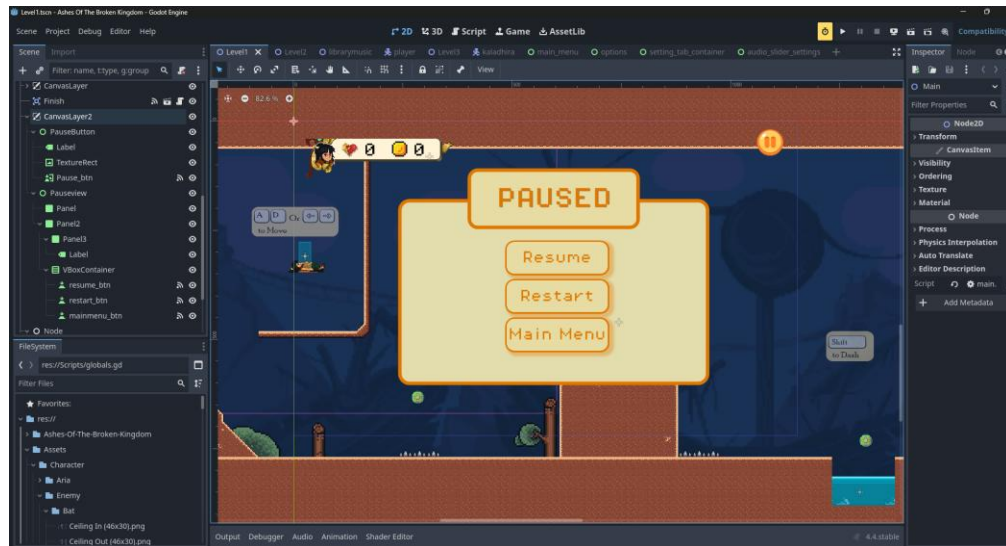
        if Input.is_action_just_pressed("shoot"):
            is_shooting = true
            Librarymusic.AudioShoot()
            sprite_2d.play("fireball")
            shoot.emit(sprite_2d.flip_h)

        if can_move:
            move_and_slide()

#death area duri dll

```

3 Pause Menu



```
extends Node2D
```

```
# Called when the node enters the scene tree for the first time.
@onready var player: CharacterBody2D = $Player
```

```
@onready var GAME_UI = $CanvasLayer2/PauseButton
@onready var PAUSE_UI = $CanvasLayer2/Pauseview
@onready var HUD_UI = $CanvasLayer
```

```
#@onready var audio_stream_player_music: AudioStreamPlayer =
$Node2D/AudioStreamPlayer_music
```

```
const MAGIC_PROJECTILE = preload("res://Scenes/MagicProjectile.tscn")
```

```
func _ready() -> void:
    Librarymusic.StopMainMenuMusic()
    Librarymusic.LevelMusic()
    GAME_UI.show()
    PAUSE_UI.hide()
    HUD_UI.show()
    player.shoot.connect(_on_shoot)
```

```
func _on_shoot(is_flip):
    var projectile = MAGIC_PROJECTILE.instantiate()
    self.add_child(projectile)
    projectile.position = player.position
    projectile.change_direction(is_flip)
```

```
func _on_pause_btn_pressed() -> void:
    Librarymusic.button2()
    await get_tree().create_timer(0.3).timeout
    pause_game()
```

```
func pause_game():
    GAME_UI.hide()
    PAUSE_UI.show()
    HUD_UI.hide()
```

```

get_tree().paused = true

func _on_resume_btn_pressed() -> void:
    Librarymusic.button1()
    await get_tree().create_timer(0.3).timeout
    resume_game()

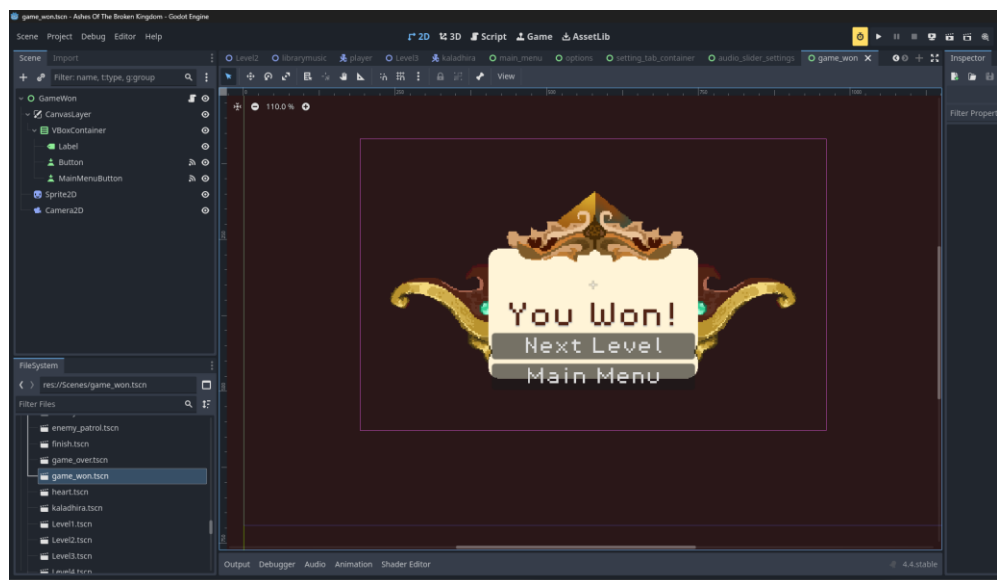
func resume_game():
    GAME_UI.show()
    PAUSE_UI.hide()
    HUD_UI.show()
    get_tree().paused = false

func _on_restart_btn_pressed() -> void:
    Librarymusic.button1()
    resume_game()
    get_tree().reload_current_scene()
    Gamestate.reset()

func _on_mainmenu_btn_pressed() -> void:
    Librarymusic.button1()
    Librarymusic.StopLevelMusic()
    await get_tree().create_timer(0.3).timeout
    resume_game()
    get_tree().change_scene_to_file("res://Scenes/main_menu.tscn")

```

4 Win Condition



```

extends Control

func _ready() -> void:
    pass

# Ganti Level dan ukur waktu loading
func _on_button_pressed() -> void:
    var start_time = Time.get_ticks_msec() # Gunakan Time

    var level_path := ""

```

```

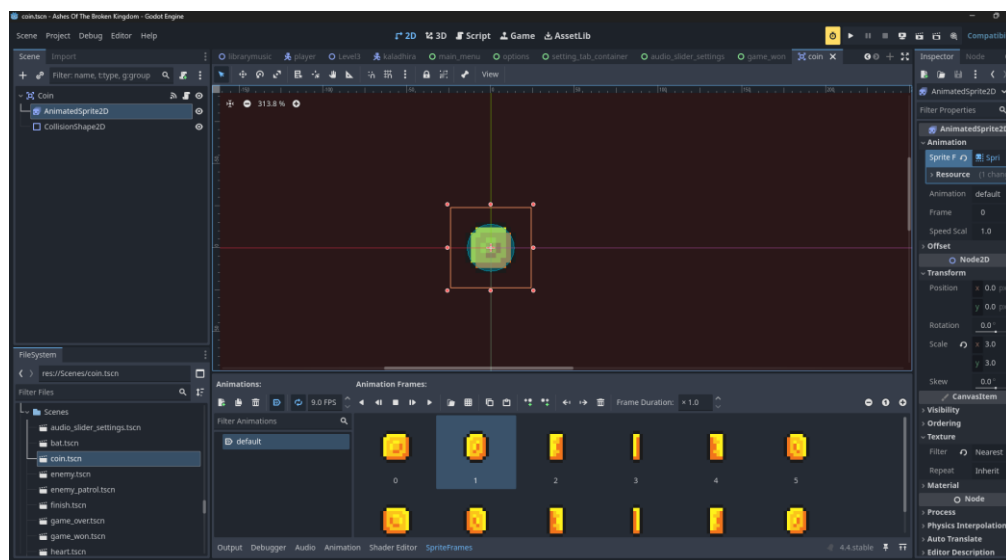
if GameState.level == 2:
    level_path = "res://Scenes/Level2.tscn"
elif GameState.level == 3:
    level_path = "res://Scenes/Level3.tscn"

if level_path != "":
    get_tree().change_scene_to_file(level_path)
    var end_time = Time.get_ticks_msec()
    var load_duration = end_time - start_time
    print("Waktu Loading level (ms): ", load_duration)

func _on_main_menu_button_pressed() -> void:
    get_tree().change_scene_to_file("res://Scenes/main_menu.tscn")

```

5 Collectible coin



```

extends Area2D

#@onready var audio_stream_player_2d: AudioStreamPlayer2D =
$AudioStreamPlayer2D
@onready var animated_sprite_2d: AnimatedSprite2D = $AnimatedSprite2D
@onready var game_manager: Node = $"../..//GameManager"

var picked_up = false

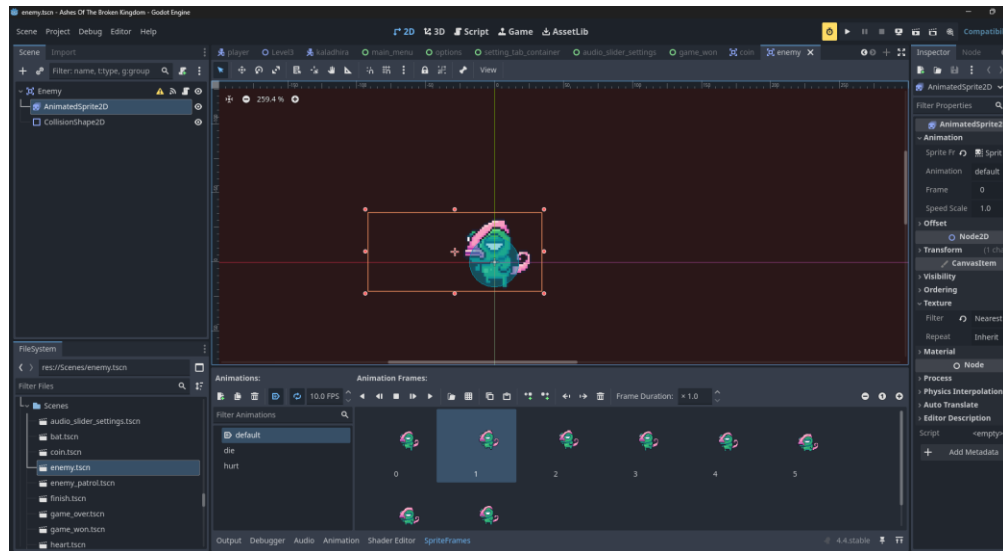
func _on_body_entered(body: Node2D) -> void:

    if body.name == "Player" and picked_up == false:

        picked_up = true
        game_manager.add_score()
        animated_sprite_2d.visible = false
        Librarymusic.AudioCoin()
        await get_tree().create_timer(0.3).timeout
        queue_free()

```

6 Enemy



```
extends Area2D

@onready var game_manager: Node = $"../../../../../GameManager"

@onready var animated_sprite_2d: AnimatedSprite2D = $AnimatedSprite2D

var last_position: Vector2 = Vector2.ZERO
var enemy_health = 3
# Called when the node enters the scene tree for the first time.
func _ready() -> void:
    pass # Replace with function body.

# Called every frame. 'delta' is the elapsed time since the previous
frame.
func _process(_delta: float) -> void:
    # Calculate the movement vector based on the difference in global
position
    var movement: Vector2 = global_position - last_position # 200 - 201 =
-1

    # Check the x component of the movement vector to determine direction
    # If moving left
    if movement.x < 0:
        scale.x = 1
    # If moving right
    elif movement.x > 0:
        scale.x = -1

    # Update last_position to the current position for the next frame
    last_position = global_position

#keno damage
func take_damage(damage):
    enemy_health -= damage

    if enemy_health <= 0:
        #animated_sprite_2d.play("hurt")
        #await get_tree().create_timer(0.3).timeout
        animated_sprite_2d.play("die")
        queue_free()
```

```
        else:
            animated_sprite_2d.play("hurt")
            await get_tree().create_timer(0.5).timeout
            animated_sprite_2d.play("default")

func _on_body_entered(body: Node2D) -> void:
    pass # Replace with function body.

    if body.name == "Player":
        body.take_damage()
```