Práctica 3

Juego de Plataformas 3D con Colección de Monedas

Arturo Daniel Agredano Gutiérrez

222966871

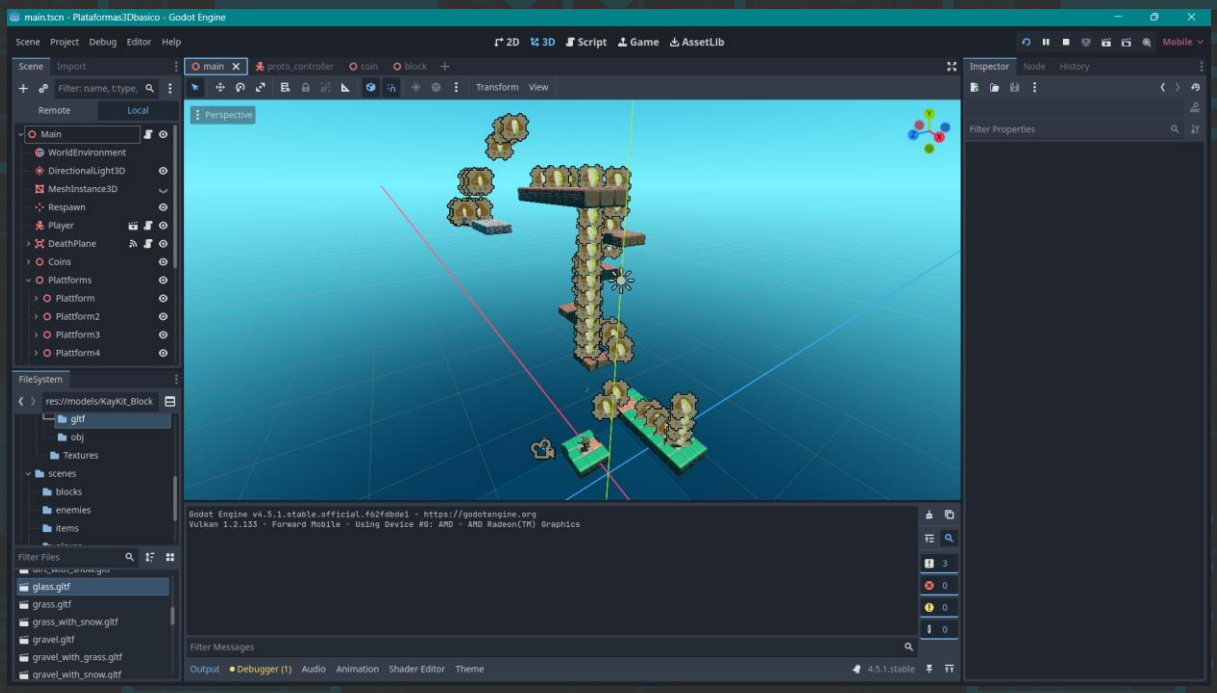
Programacion de gráficos 3D

D02

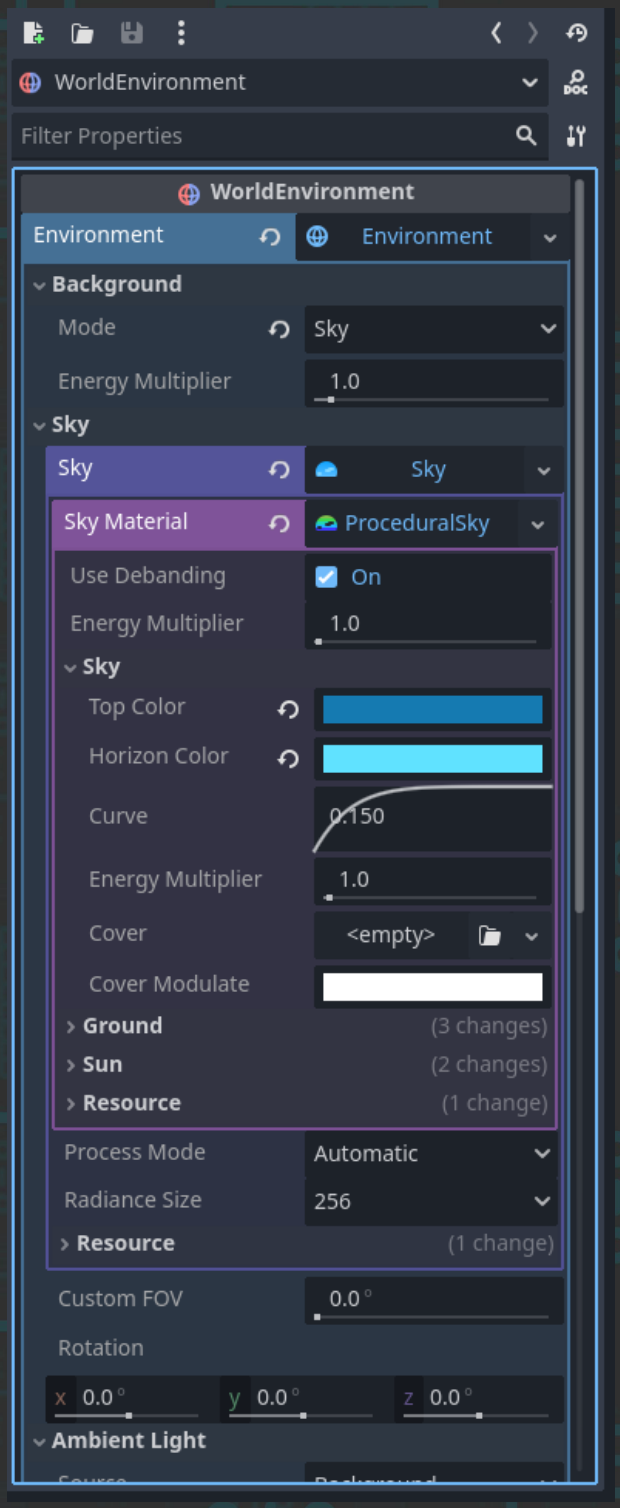
Jose Luis David Bonilla Carranza

01/11/2025

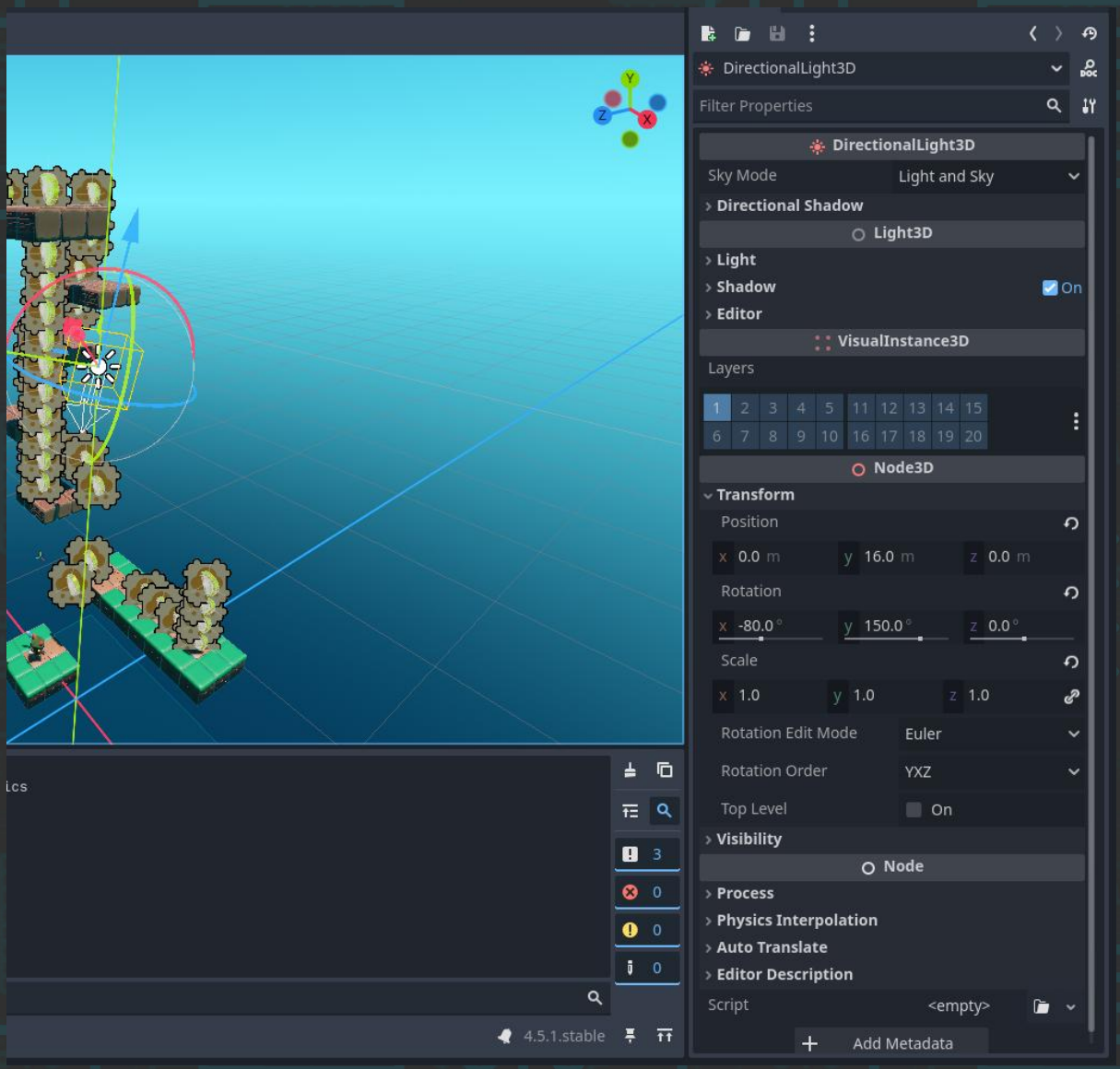
Capturas del desarrollo



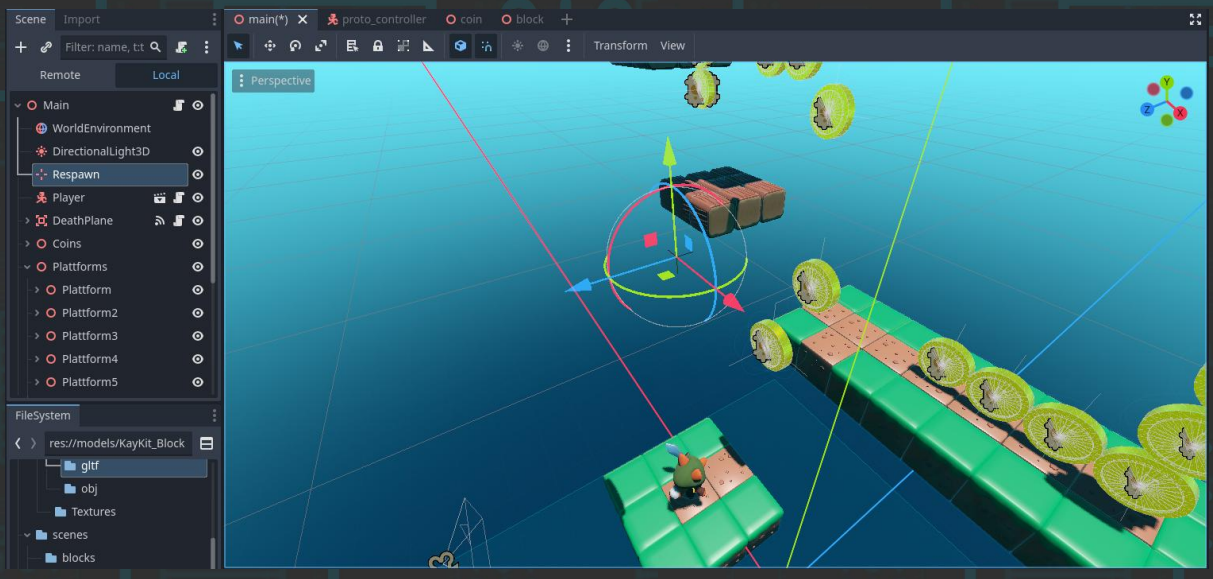
Escena del nivel



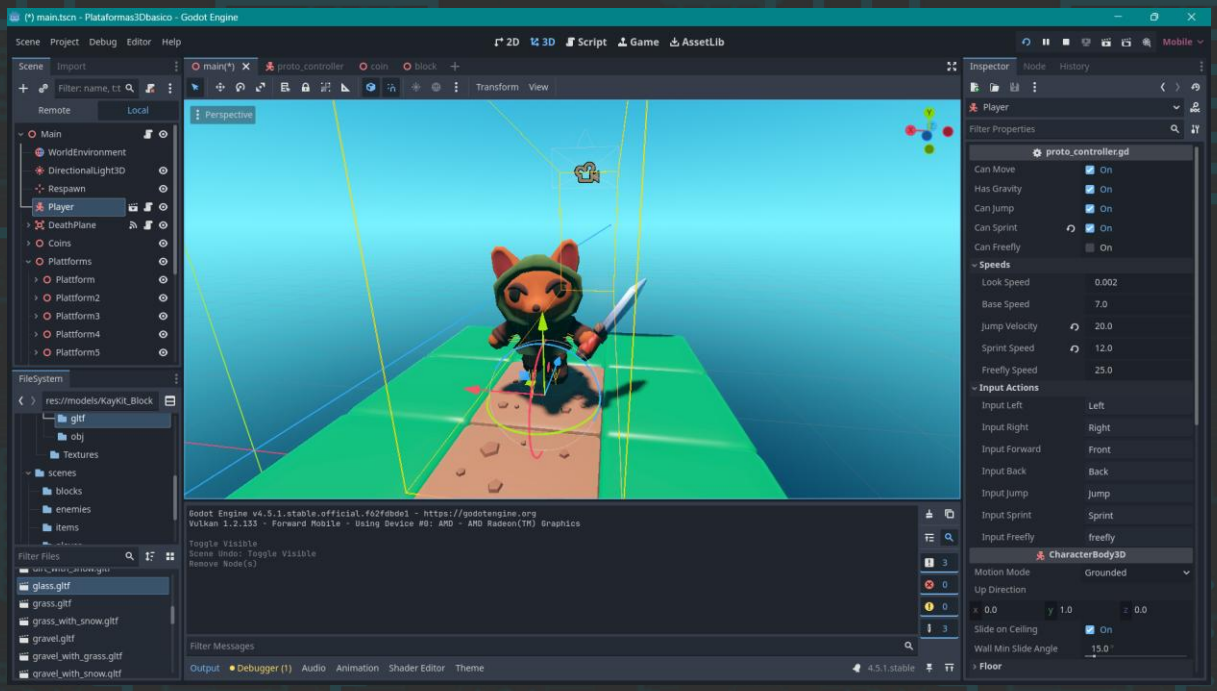
Configuración del ambiente



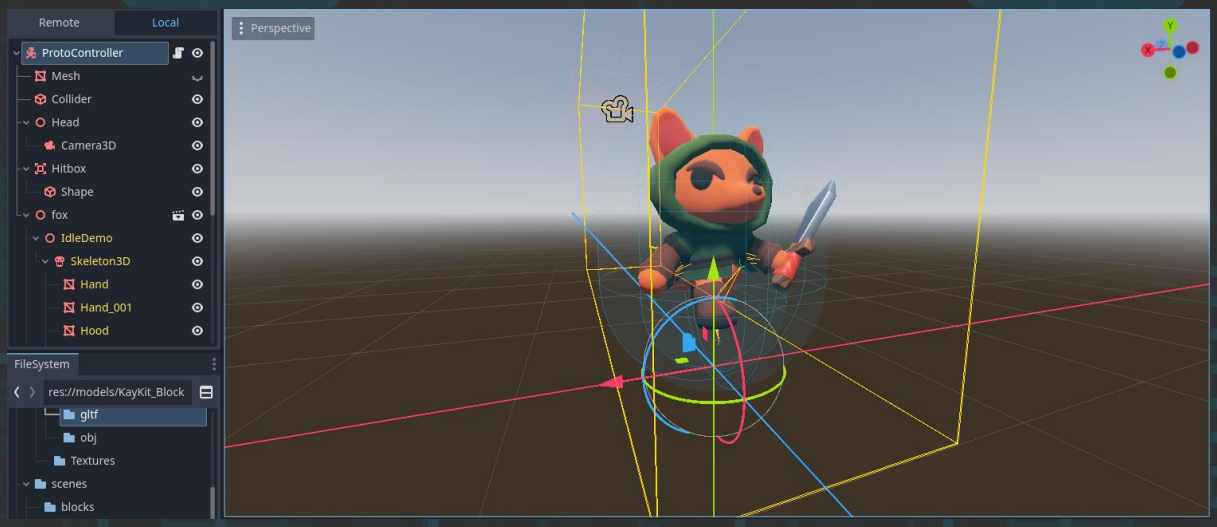
Configuración de la luz global



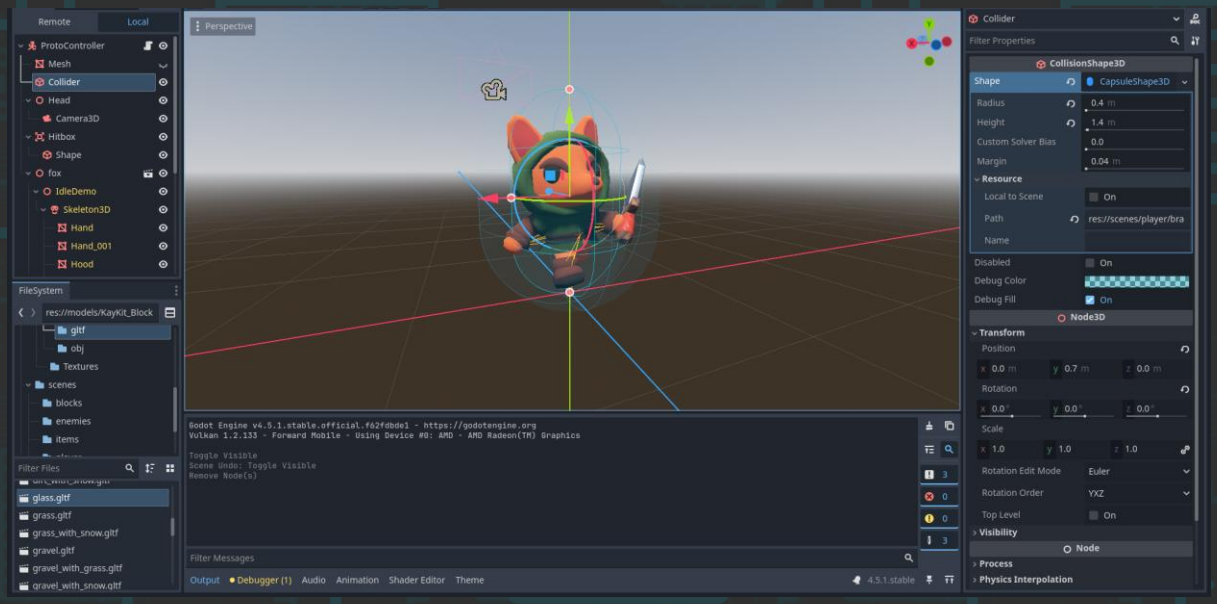
Punto de respawn



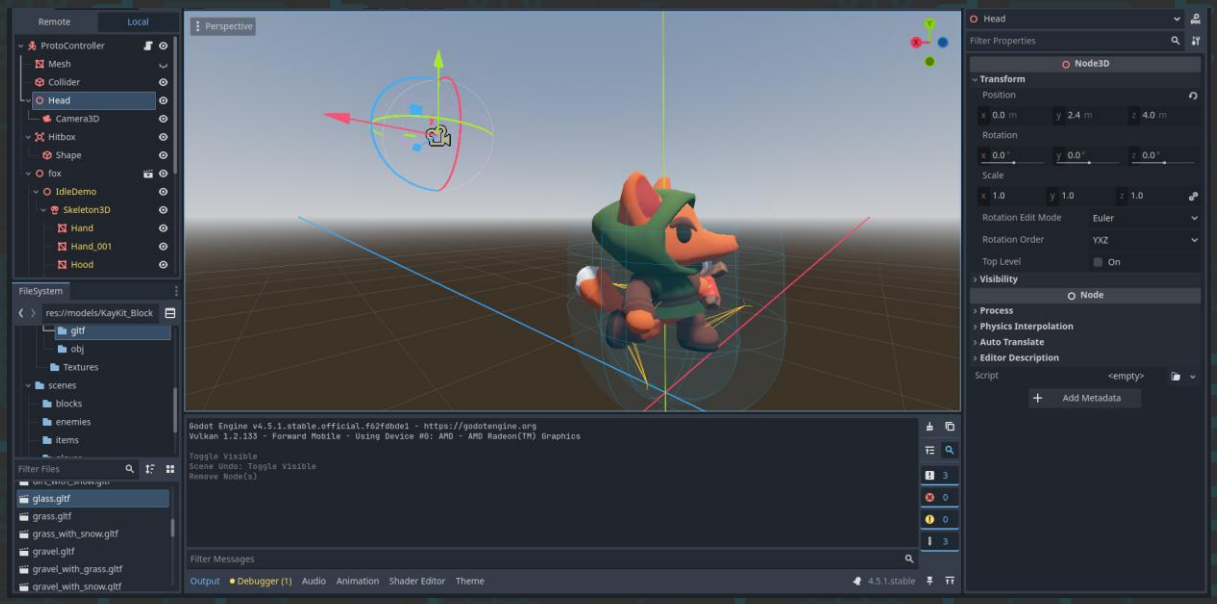
Jugador



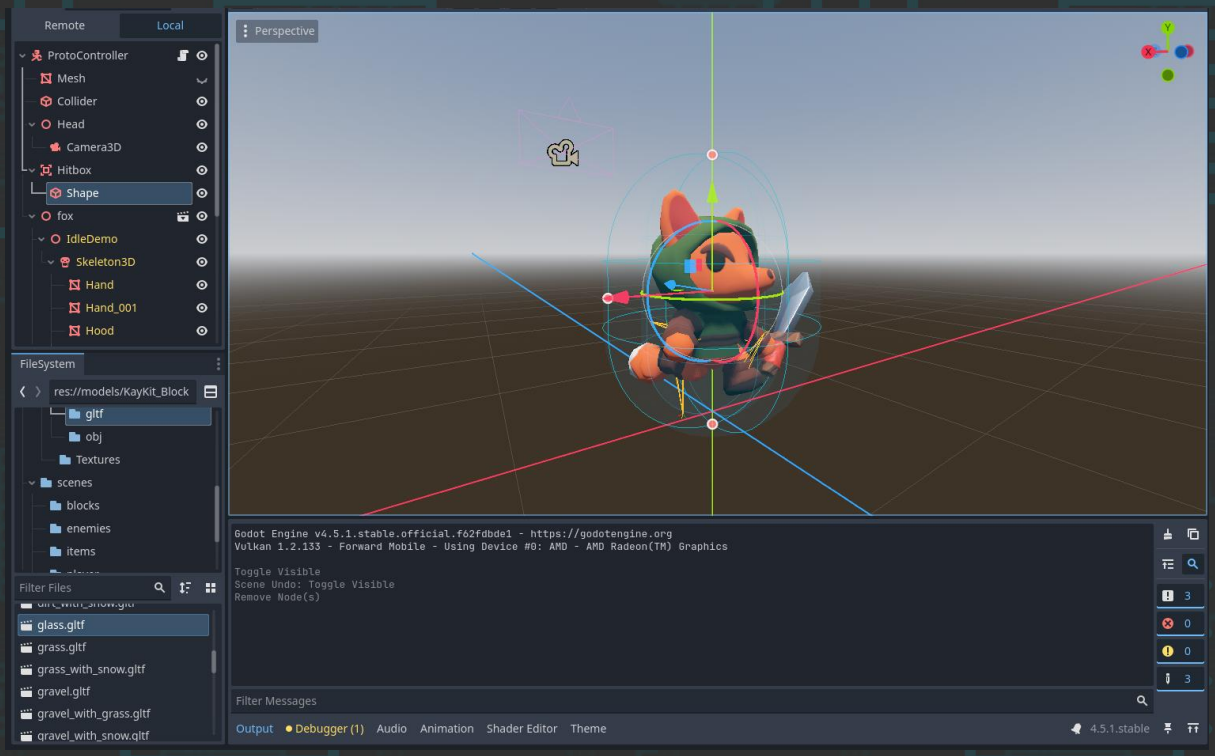
Escena del jugador



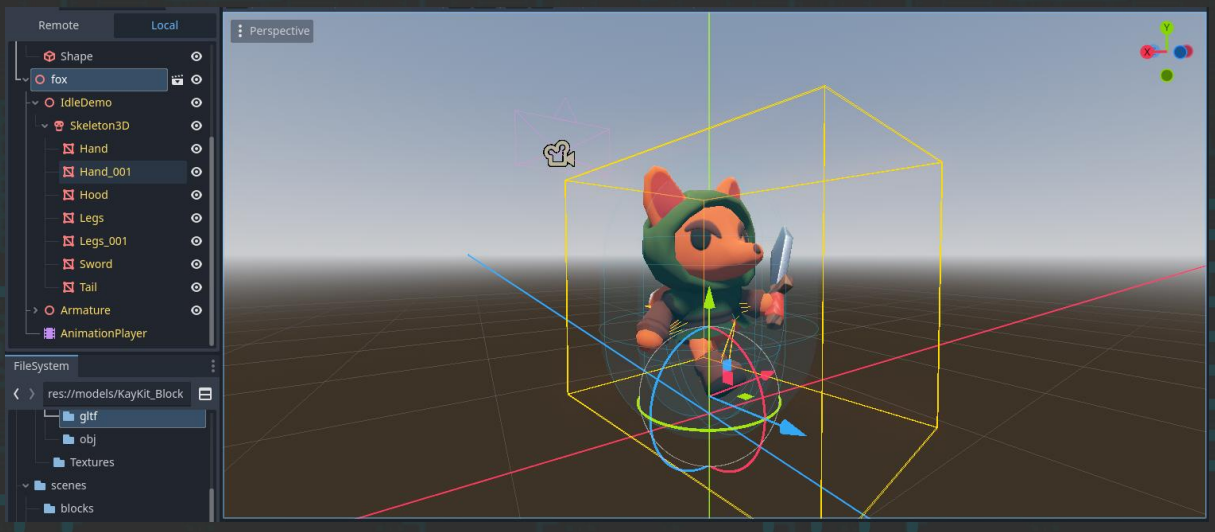
Colisión para terreno



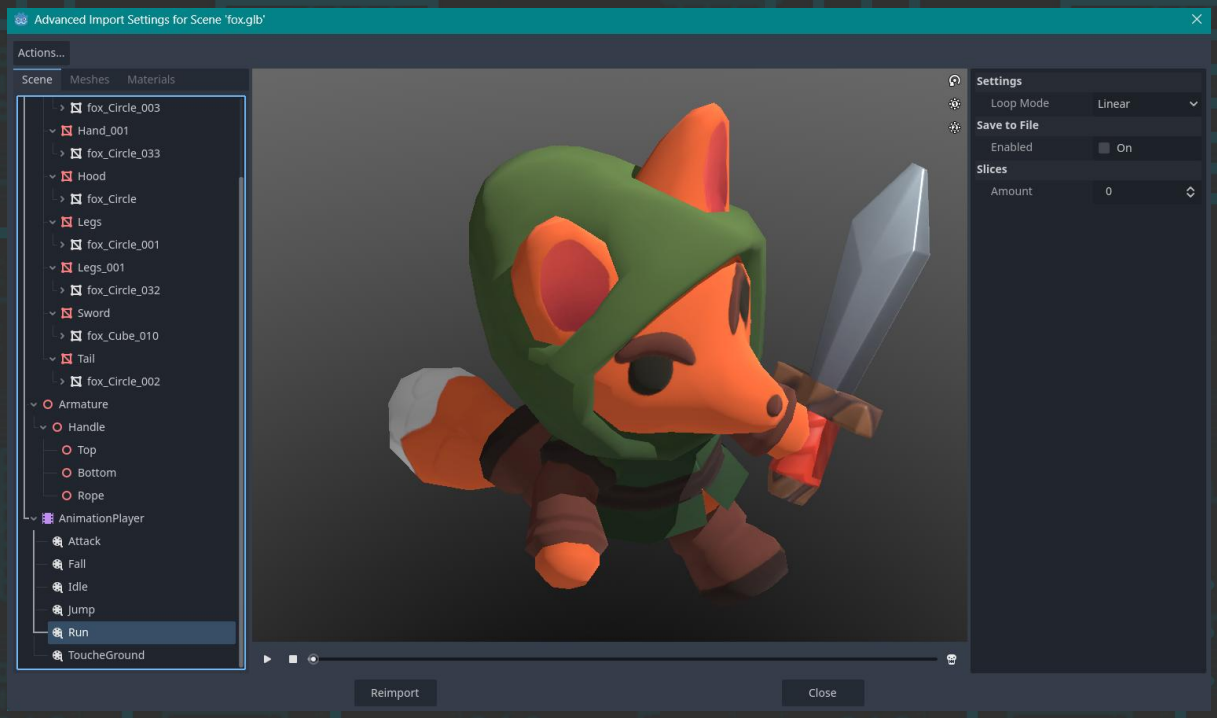
Cámara ajustada a cierta distancia del jugador



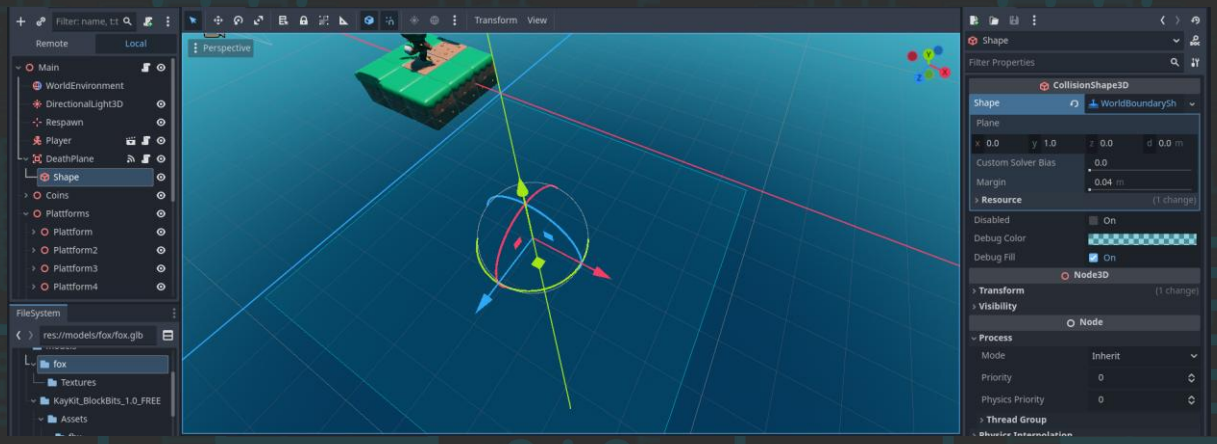
Colisión para monedas



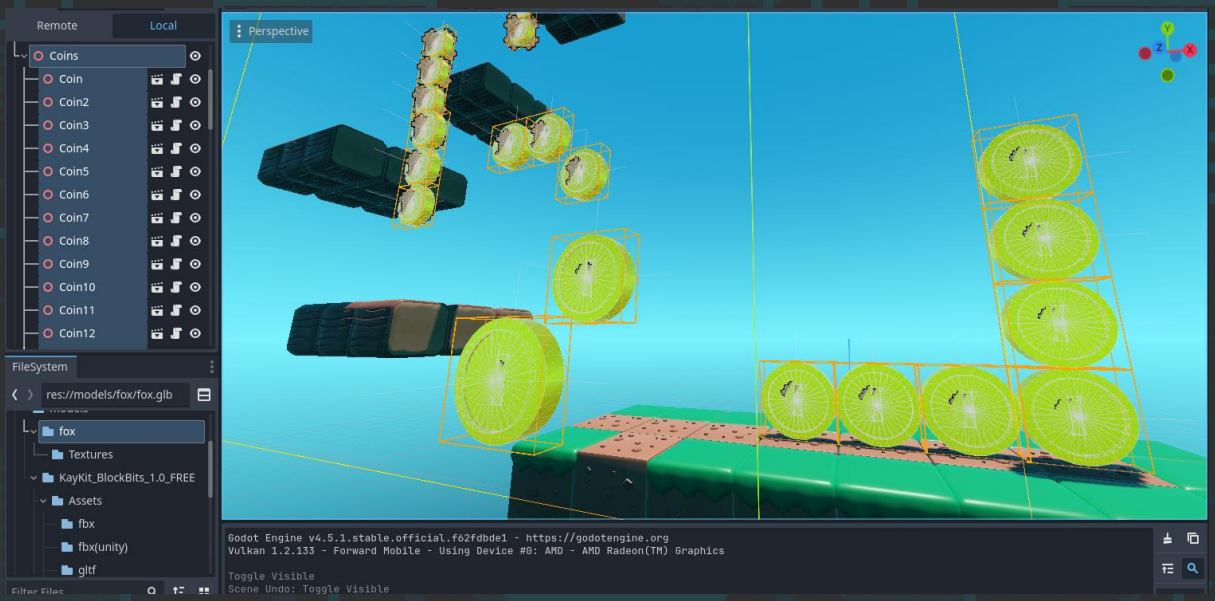
Modelo importado con meshes, texturas y animaciones



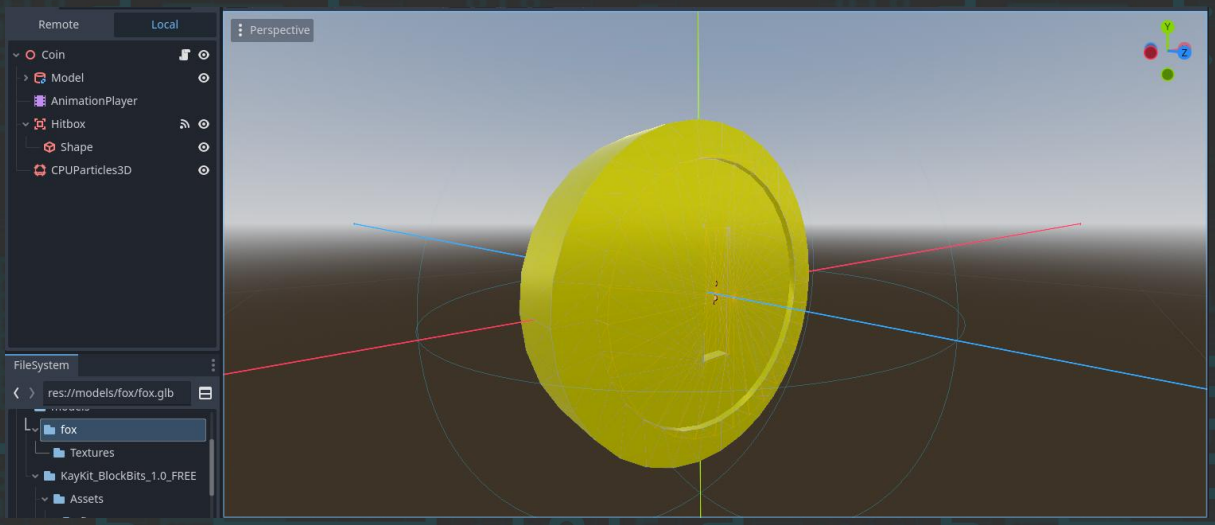
Animaciones adaptadas para loopear



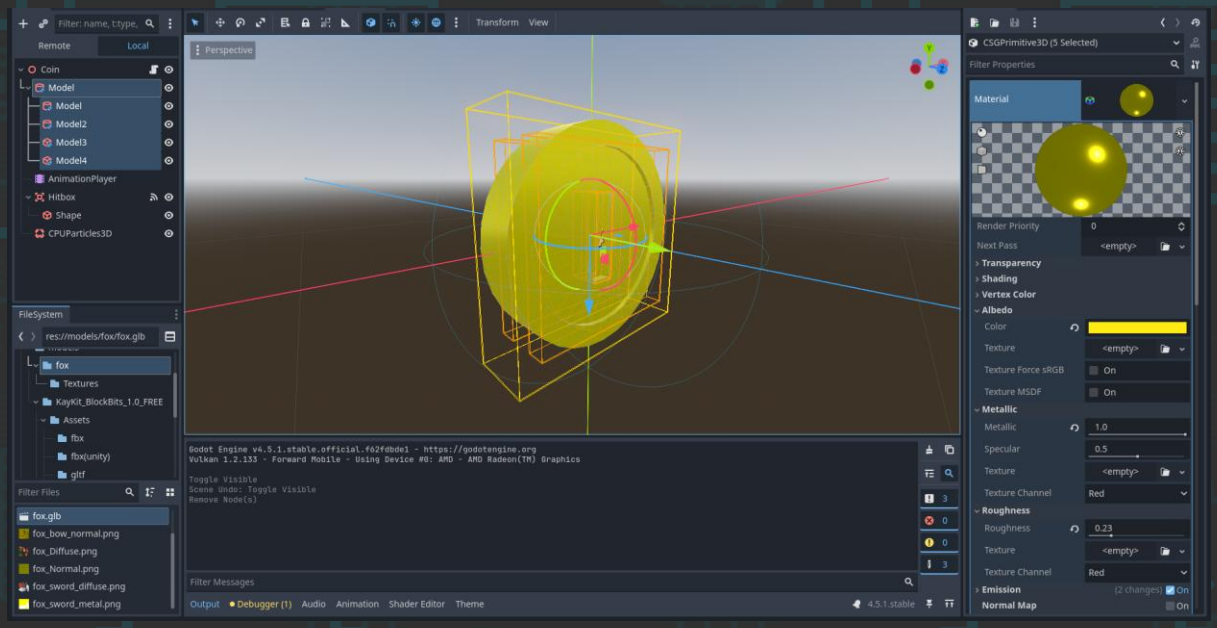
Plano para detectar caida y regresar al respawn



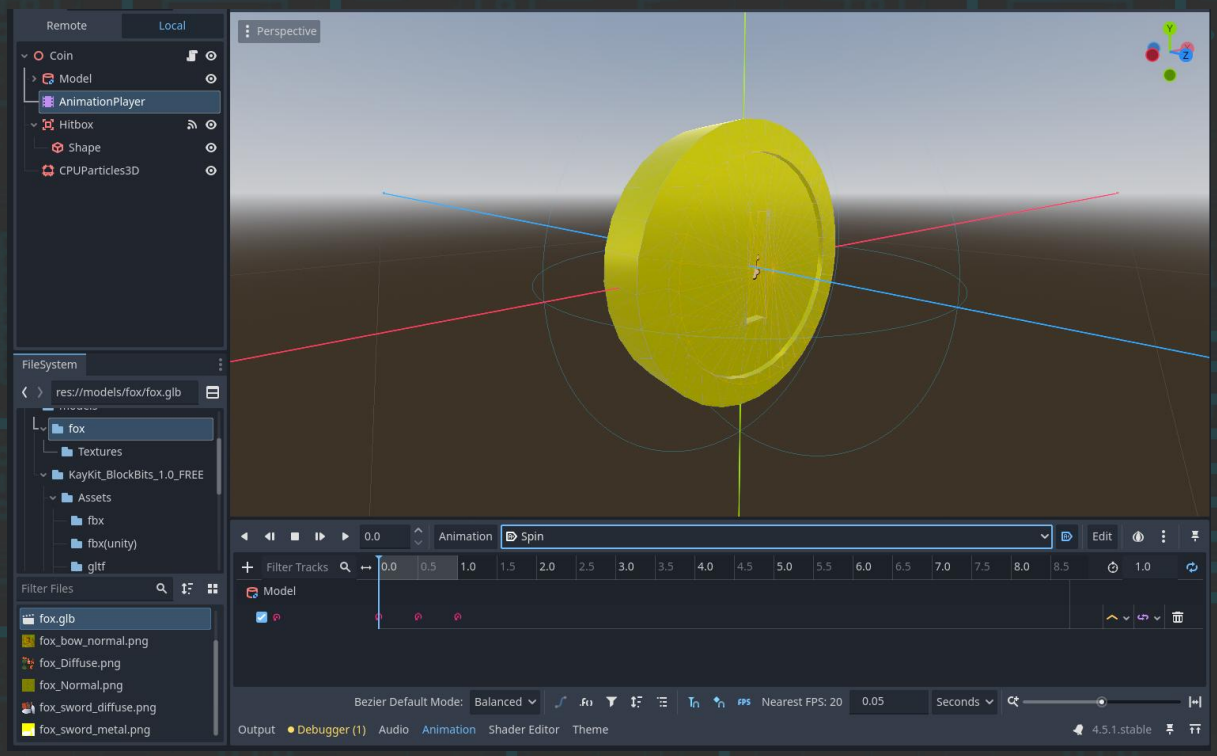
Monedas esparcidas por el nivel



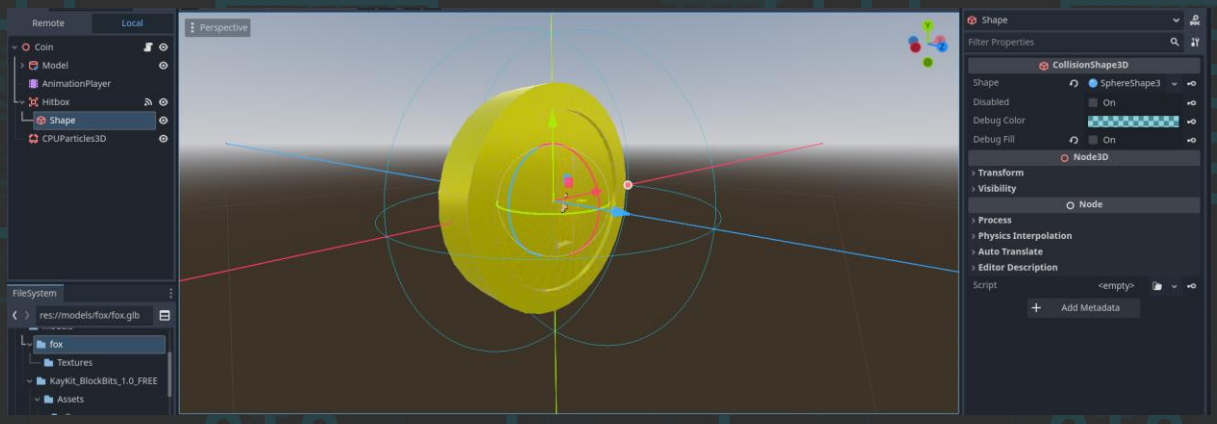
Escena de moneda



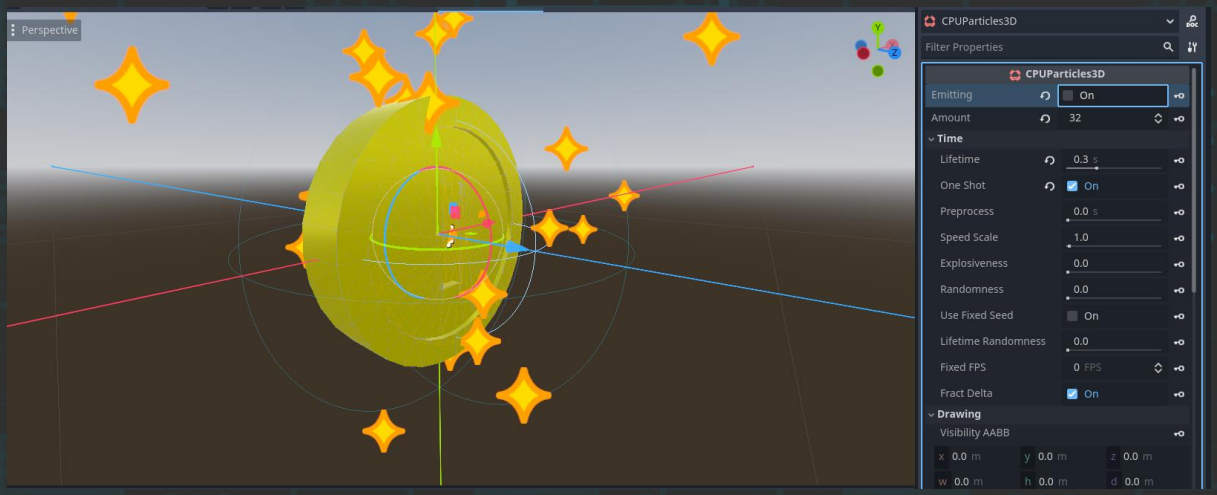
Modelo de moneda creado con nodos CSG



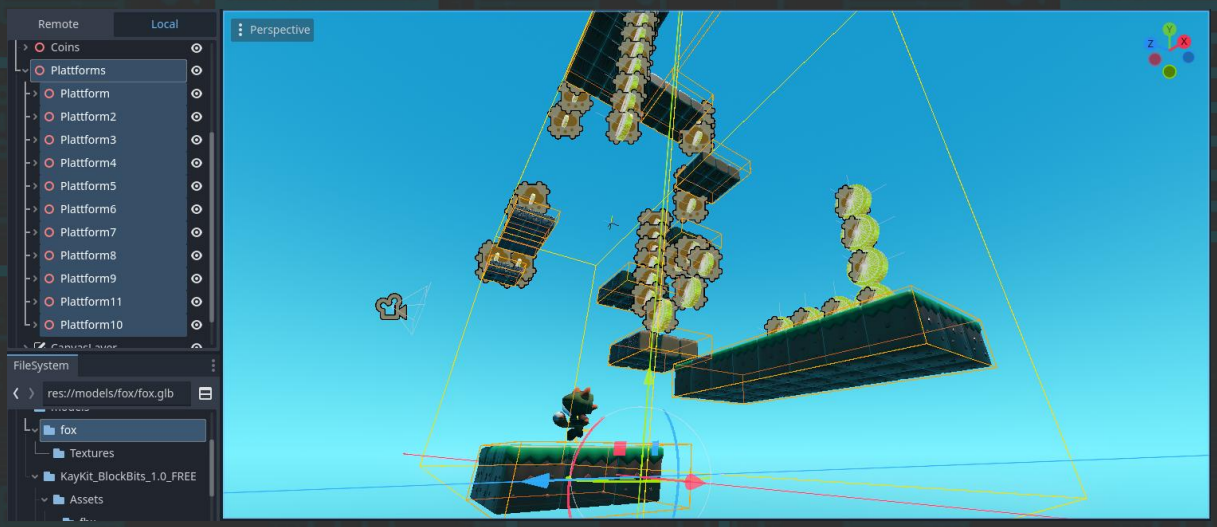
Animación para hacer girar la moneda



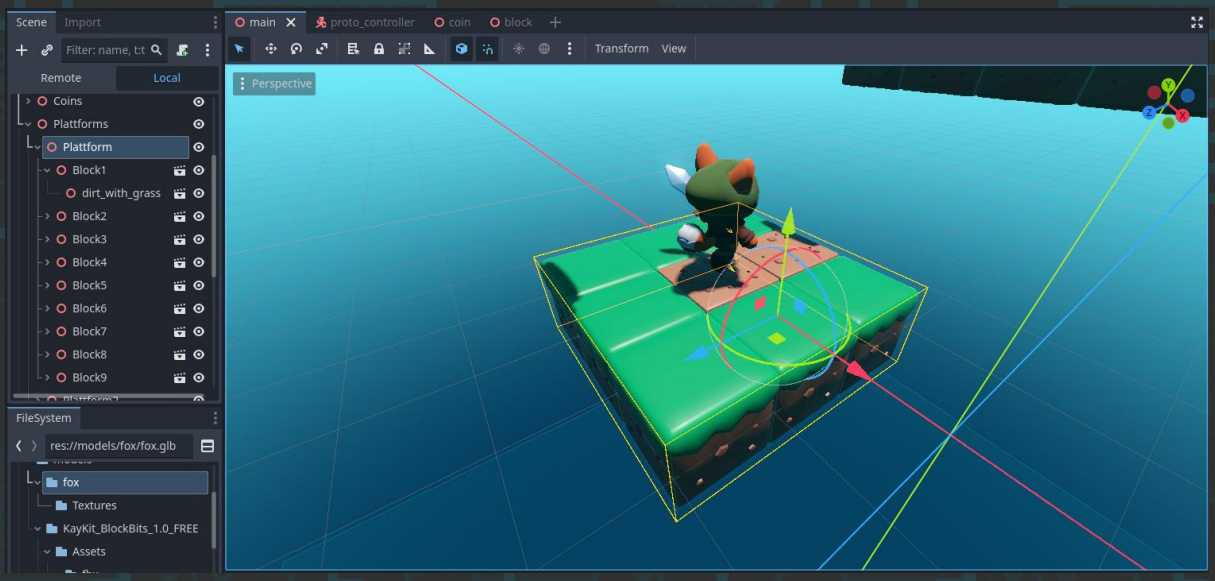
Colisión para ser recolectada



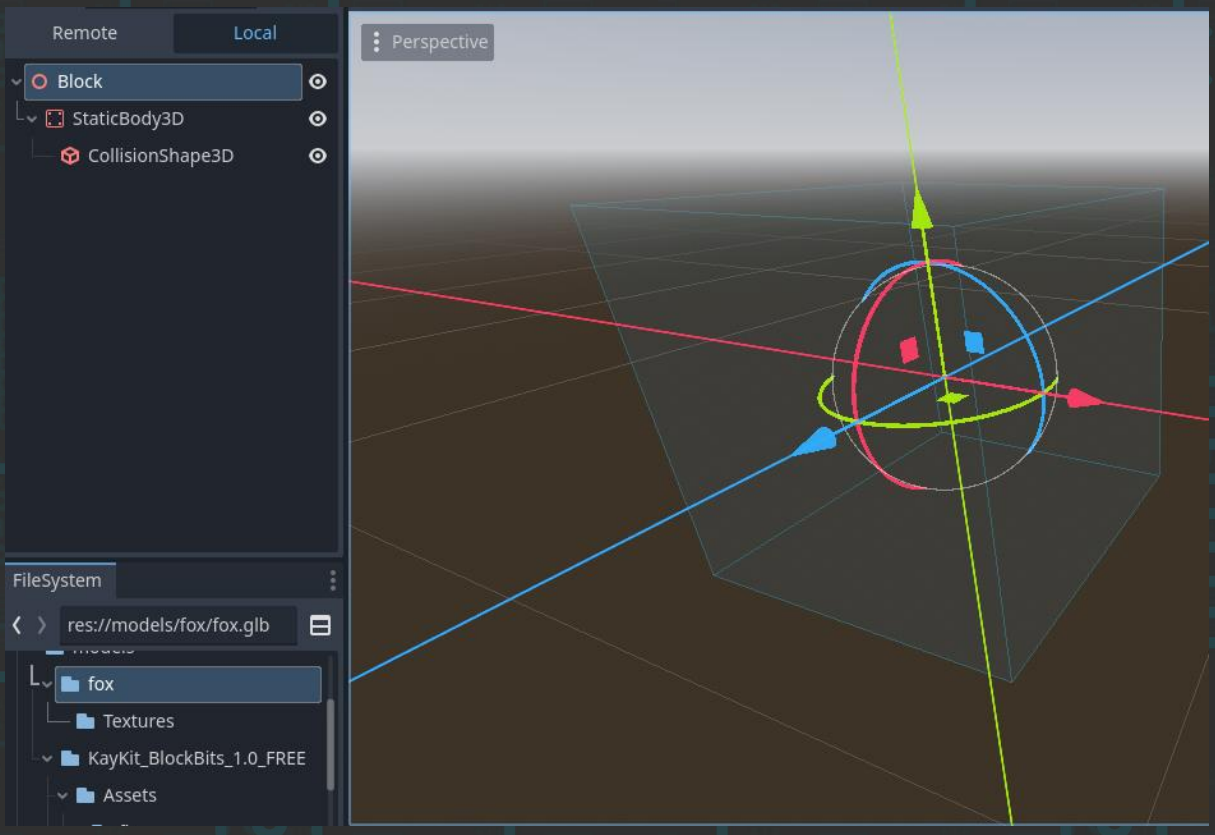
Partículas al ser recolectada



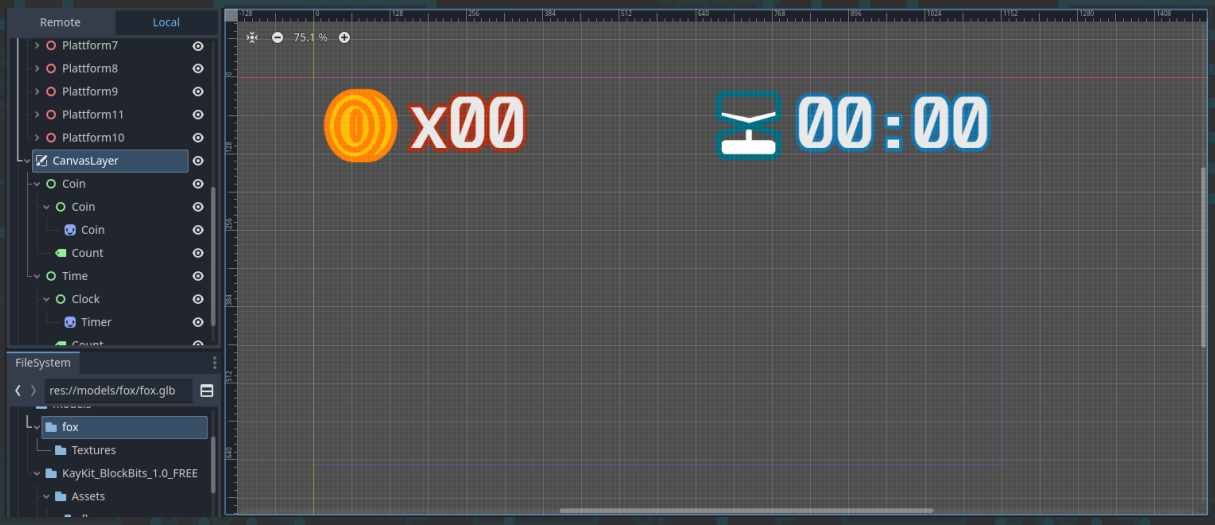
Grupos de plataformas repartidas por el nivel



Grupo de bloques que forman las plataformas



Escena que sirve como base para los bloques con colisión



Interfaz de usuario

Código

Código para manejar al jugador descargado del video de referencia

El manejo de cámara fue editado para cambiar a vista en tercera persona

Se agregaron triggers para las animaciones del modelo 3D

Se sustituyeron las acciones para usar inputs personalizados

```
# ProtoController v1.0 by Brackeys
# CC0 License
# Intended for rapid prototyping of first-person games.
# Happy prototyping!

extends CharacterBody3D

## Can we move around?
@export var can_move : bool = true
## Are we affected by gravity?
@export var has_gravity : bool = true
## Can we press to jump?
@export var can_jump : bool = true
## Can we hold to run?
@export var can_sprint : bool = false
## Can we press to enter freefly mode (noclip)?
@export var can_freefly : bool = false

@export_group("Speeds")
## Look around rotation speed.
@export var look_speed : float = 0.002
## Normal speed.
@export var base_speed : float = 7.0
## Speed of jump.
@export var jump_velocity : float = 4.5
## How fast do we run?
@export var sprint_speed : float = 10.0
## How fast do we freefly?
@export var freefly_speed : float = 25.0

@export_group("Input Actions")
## Name of Input Action to move Left.
@export var input_left : String = "Left"
## Name of Input Action to move Right.
@export var input_right : String = "Right"
## Name of Input Action to move Forward.
@export var input_forward : String = "Front"
## Name of Input Action to move Backward.
@export var input_back : String = "Back"
## Name of Input Action to Jump.
@export var input_jump : String = "Jump"
## Name of Input Action to Sprint.
```

```
@export var input_sprint : String = "Sprint"
## Name of Input Action to toggle freefly mode.
@export var input_freefly : String = "freefly"

var mouse_captured : bool = false
var look_rotation : Vector2
var move_speed : float = 0.0
var freeflying : bool = false

## IMPORTANT REFERENCES
@onready var head: Node3D = $Head
@onready var collider: CollisionShape3D = $Collider
@onready var anim: AnimationPlayer = $fox/AnimationPlayer

func _ready() -> void:
	capture_mouse()
	check_input_mappings()
	look_rotation.y = rotation.y
	look_rotation.x = head.rotation.x

func _unhandled_input(event: InputEvent) -> void:
	# Mouse capturing
	if Input.is_mouse_button_pressed(MOUSE_BUTTON_LEFT):
		capture_mouse()
	if Input.is_key_pressed(KEY_ESCAPE):
		release_mouse()

	# Look around
	if mouse_captured and event is InputEventMouseMotion:
		rotate_look(event.relative)

	# Toggle freefly mode
	if can_freefly and
Input.is_action_just_pressed(input_freefly):
		if not freeflying:
			enable_freefly()
		else:
			disable_freefly()

func _physics_process(delta: float) -> void:
	# If freeflying, handle freefly and nothing else
	if can_freefly and freeflying:
		var input_dir := Input.get_vector(input_left,
input_right, input_forward, input_back)
		var motion := (head.global_basis *
Vector3(input_dir.x, 0, input_dir.y)).normalized()
		motion *= freefly_speed * delta
		move_and_collide(motion)
		return

	# Apply gravity to velocity
	if has_gravity:
```

```
          if not is_on_floor():
               velocity += get_gravity() * 5.0 * delta

     # Apply jumping
     if can_jump:
          if Input.is_action_just_pressed(input_jump) and
is_on_floor():
               velocity.y = jump_velocity
               anim.play("Jump")

     # Modify speed based on sprinting
     if can_sprint and Input.is_action_pressed(input_sprint):
          move_speed = sprint_speed
     else:
          move_speed = base_speed

     #if move_speed > 0.0:

     # Apply desired movement to velocity
     if can_move:
          var input_dir := Input.get_vector(input_left,
input_right, input_forward, input_back)
          var move_dir := (transform.basis *
Vector3(input_dir.x, 0, input_dir.y)).normalized()
          if move_dir:
               velocity.x = move_dir.x * move_speed
               velocity.z = move_dir.z * move_speed
               $fox.rotation.y = (input_dir*Vector2(-1,
1)).angle()-PI/2
               if is_on_floor() and velocity.y <= 0.0:
anim.play("Run")
               if not is_on_floor() and anim.current_animation
!= "Jump": anim.play("Fall")
          else:
               velocity.x = move_toward(velocity.x, 0,
move_speed)
               velocity.z = move_toward(velocity.z, 0,
move_speed)
               if is_on_floor() and velocity.y <= 0.0:
anim.play("Idle")
     else:
          velocity.x = 0
          velocity.y = 0

     #if not is_on_floor() and anim.animation_finished:
          #anim.play("Jump")

     # Use velocity to actually move
     move_and_slide()


## Rotate us to look around.
```

```
## Base of controller rotates around y (left/right). Head rotates
around x (up/down).
## Modifies look_rotation based on rot_input, then resets basis
and rotates by look_rotation.
func rotate_look(rot_input : Vector2):
	look_rotation.x -= rot_input.y * look_speed
	look_rotation.x = clamp(look_rotation.x, deg_to_rad(-85),
deg_to_rad(85))
	look_rotation.y -= rot_input.x * look_speed
	transform.basis = Basis()
	rotate_y(look_rotation.y)
	head.transform.basis = Basis()
	head.rotate_x(look_rotation.x)
	var percent = (rad_to_deg(head.rotation.x)+85)/170.0
	head.position.z = 7.5 * (percent)
	head.position.y = 8.0 * (1-percent) - 2.0


func enable_freefly():
	collider.disabled = true
	freeflying = true
	velocity = Vector3.ZERO

func disable_freefly():
	collider.disabled = false
	freeflying = false


func capture_mouse():
	Input.set_mouse_mode(Input.MOUSE_MODE_CAPTURED)
	mouse_captured = true


func release_mouse():
	Input.set_mouse_mode(Input.MOUSE_MODE_VISIBLE)
	mouse_captured = false


## Checks if some Input Actions haven't been created.
## Disables functionality accordingly.
func check_input_mappings():
	if can_move and not InputMap.has_action(input_left):
		push_error("Movement disabled. No InputAction found
for input_left: " + input_left)
		can_move = false
	if can_move and not InputMap.has_action(input_right):
		push_error("Movement disabled. No InputAction found
for input_right: " + input_right)
		can_move = false
	if can_move and not InputMap.has_action(input_forward):
		push_error("Movement disabled. No InputAction found
for input_forward: " + input_forward)
```

```
            can_move = false
      if can_move and not InputMap.has_action(input_back):
            push_error("Movement disabled. No InputAction found
for input_back: " + input_back)
            can_move = false
      if can_jump and not InputMap.has_action(input_jump):
            push_error("Jumping disabled. No InputAction found for
input_jump: " + input_jump)
            can_jump = false
      if can_sprint and not InputMap.has_action(input_sprint):
            push_error("Sprinting disabled. No InputAction found
for input_sprint: " + input_sprint)
            can_sprint = false
      if can_freefly and not InputMap.has_action(input_freefly):
            push_error("Freefly disabled. No InputAction found for
input_freefly: " + input_freefly)
            can_freefly = false
```

Código global que maneja los contadores

```
extends Node

var main
var coinCount
var timerCount
var coins = 0
var timer = 0

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

func addCoins(qty = 1):
      coins += qty
      coinCount.text = str(coins).pad_zeros(2)
```

Código del nodo principal que ayuda a que el nodo global maneje los contadores y actualice las interfaces

```
extends Node3D

func _ready() -> void:
      Game.main = $"."
      Game.coinCount = $CanvasLayer/Coin/Count
      Game.timerCount = $CanvasLayer/Time/Count

func _process(_delta: float) -> void:
      pass

func addSec():
      Game.timer += 1
```

```
	var time = Game.timer % 3600
	Game.timer = time
	var text =
str(int(time)/60).pad_zeros(2)+":"+str(int(time)%60).pad_zeros(2)
	$CanvasLayer/Time/Count.text = text
```

Función que maneja la caída infinita

```
extends Area3D

func entered(_area: Area3D) -> void:
	$"../Player".position = $"../Respawn".position
```

Código de las monedas que inicializa la animación de obtención

```
extends Node3D

func entered(_area: Area3D) -> void:
	Game.addCoins()
	$AnimationPlayer.play("Collect")
```

Problemas y soluciones

Honestamente el desarrollo fue bastante sencillo y con pocos problemas, simplemente experimentar con lo que hacía cada parámetro me permitió entender como trabajar con cada uno de los elementos que conforman un juego en 3D, incluso muchas de las estructuras de las escenas resultaron ser paralelos casi exactos a las estructuras que he usado anteriormente en juegos 2D.