Institut Puig Castellar
Santa Coloma de Gramenet

Sky Jumper 2

CFGM Sistemes Microinformàtics i Xarxes

Fernando Aldair Cano Panchana
José Bermejo Porcuna

Curs acadèmic

29 de març 2019

B) GNU Free Documentation License (GNU FDL)

C) Copyright

Resum del projecte (màxim 250 paraules):

Tenemos pensado hacer un videojuego 2D sencillo con las cosas básicas que identifican un videojuego y si es posible poner nosotros mismos la textura del juego. Utilizaremos la aplicación Godot Engine, tenemos un poco de experiencia pero la aplicación no es tan complicada así que nos lo facilitará un poco, para poder programar el personaje lo hacemos con scripts, los enemigo se moverán de un lado a otro haciendo X cosa cada uno, el objetivo será esquivar los enemigos, recogiendo objetos con el fin de coger puntos.

Para llevar a cabo este proyecto usaremos varios programas de software libre:

- Godot Engine 3
- Gimp
- Krita

Usaremos Godot Engine porque es un motor nuevo para nosotros y nos servirá de experiencia en programación con nuevos lenguajes, el que usa este es similar a PYTHON 3.0.

Abstract (in English, 250 words or less):

We plan to make a simple 2D video game with the basic things that identify a videogame and if it is possible to put the texture of the game ourselves. We will use the Godot Engine application, we have a little experience but the application is not so complicated so it will facilitate us a bit, to be able to program the character we do it with scripts, the enemy will move from one place to another doing X thing each one , the objective will be to dodge the enemies, collecting objects in order to collect points.

To carry out this project we will use several free software programs:

- Godot Engine 3
- Gimp
- Krita

We will use Godot Engine because it is a new engine for us and it will serve us as an experience in programming with new languages, which is similar to PYTHON 3.0.

Paraules clau (entre 4 i 8):

Godot Engine, GD Script, 2D, Videojuego.

Índex

Lista de figuras

1. Introducción

1.1 Contexto y justificación del Trabajo

Con este proyecto que hemos propuesto, la creación de un videojuego, queremos cubrir la necesidad de entretener y divertir.

Es un tema relevante porque ha los integrantes del grupo siempre nos han gustado mucho los juego, pero ahora queremos hacer los nuestros propios y este proyecto nos puede servir para empezar en este mundillo, estamos muy motivados y con ganas de hacerlo, por eso creemos que el resultado será muy bueno.

Cuando terminemos este proyecto queremos que quede una aplicación, entretenida, divertida y fácil de entender.

1.2 Objetivos del Trabajo

Estos son los objetivos que nos hemos marcados :

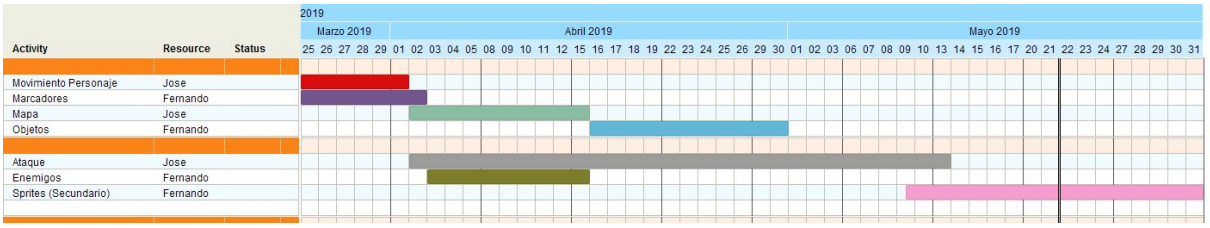
- Queremos conseguir que el personaje se mueva de izquierda a derecha, que suba saltar y atacar.
- Marcadores de vida, punto de objetos (puntos)
- enemigos
- objetos

1.3 Enfoque y método seguido

Queremos hacer nuestro propio producto, esta estrategia es la más apropiada por que si cogemos un juego hecho, no entenderemos nada de su código, aparte que lo que nosotros queremos es aprender programación y saber lo que cuesta hacer un juego desde 0.

1.4 Planificación del proyecto

Como para este proyecto somos 2 distribuiremos la tareas a partes iguales, con más o menos al mismos tiempo para cada tarea:



Nombre de la tarea	Fecha de inicio	Fecha final	Duración (días)	programador
Movimiento	**25/03/2019**	**01/04/19**	**8**	**Jose**
Ataque	**02/04/2019**	**13/05/19**	**30**	**Fernando**
Marcadores	**25/03/2019**	**02/04/19**	**9**	**Fernando/jose**
Enemigos	**11/04/2019**	**21/04/19**	**8**	**Fernando**
Objetos	**02/04/2019**	**11/04/19**	**9**	**Jose**
Mapas	**02/04/2019**	**15/04/19**	**13**	**Jose/Fernando**
Sprites (Secundari)	**15/04/2019**	**28/04/19**	**13**	**Fernando**

1.5 Breve sumario de productos obtenidos

El producte final será un videojuego (un programa ejecutable)

1.6 Breve descripción de los otros capítulos de la memoria

Queremos hacer un juego de plataformas que la pantalla esté fija al personaje y la Movilidad sea de izquierda a derecha, que el personaje pueda saltar para poder pasar los obstáculos y avanzar, conseguir puntos, esquivan a los enemigos (en principio no pueden atacar) con el fin de pasar el nivel y que al final del nivel esté un final boss

2. Desarrollo

2.1 Características principales de Godot Engine

Escenas

Las escenas es una característica de Godot Engine que contendrá el contenido de todos los nodos con sus scripts correspondientes, teniendo acceso a todos los nodos hijos. Esto quiere decir que la **escena** tendrá todo el contenido de nuestro juego.

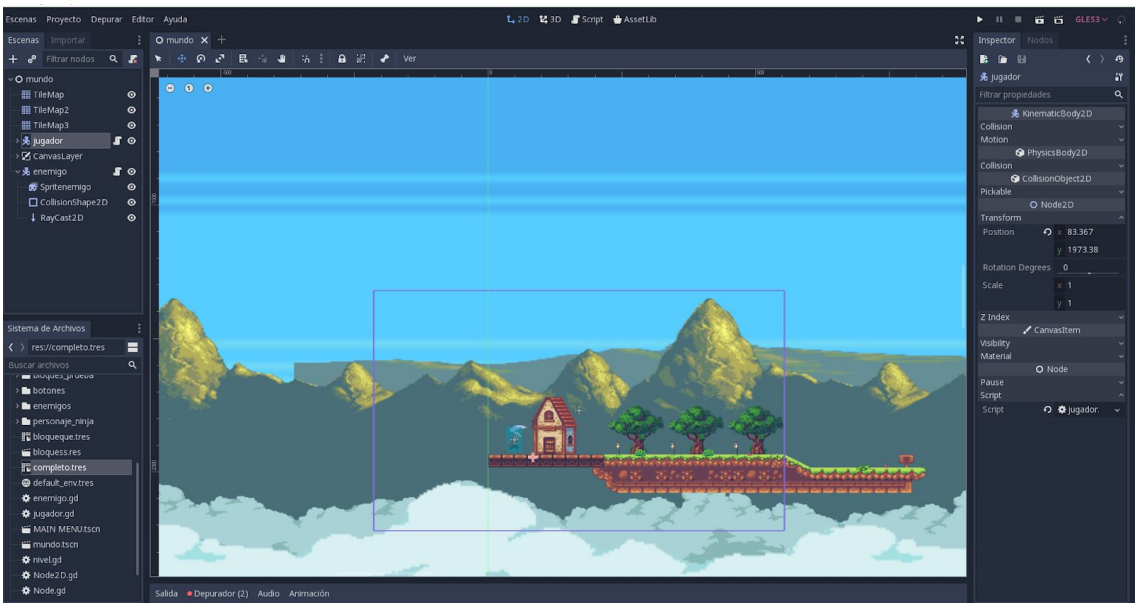
GDSCRIPT

GDScript es el lenguaje de programación que usa Godot Engine, es similar a python (la indentación y muchas palabras clave son similares).

De este lenguaje lo que más hemos usado es (if, elif, else) porque más o menos ya sabíamos usarlo pero también hemos aprendido usar más comandos como por ejemplo (var, enum, func, pass, etc…) en los puntos de más abajo enseñamos cómo los hemos usado.

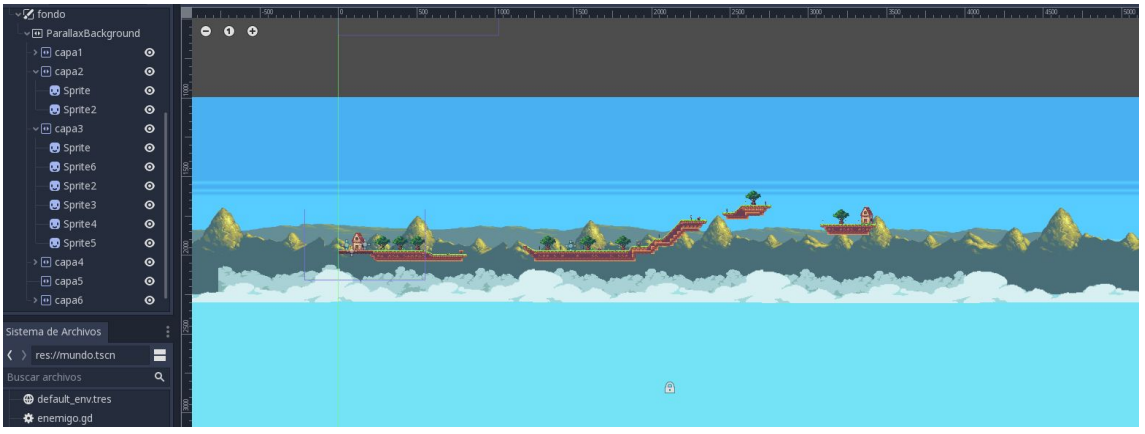
Empezamos el Desarrollo

Para empezar el desarrollo primero tendremos que tener en cuenta las características que usaremos, como las escenas, diferentes nodos y herramientas que nos proporciona Godot, para explicar cada cosa que usamos pondremos capturas.

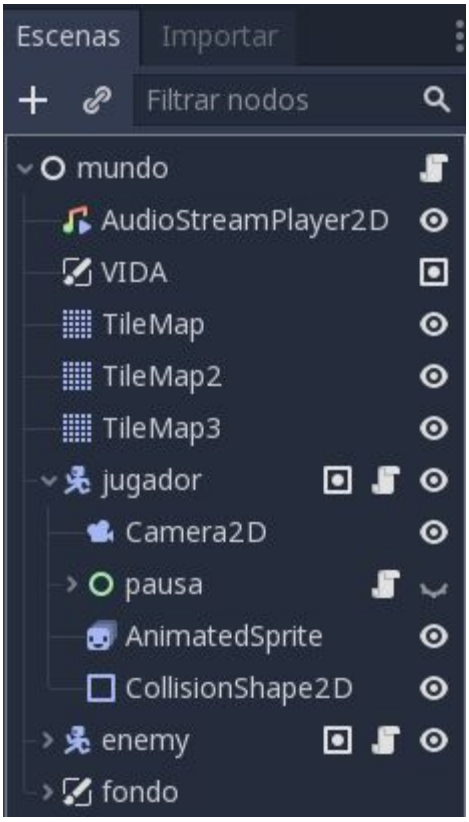


Ejemplo de las Escena 1, esta es la principal del videojuego

2.1.1.Escena: escenario de juego



Esta escena es la contiene todo el mapa y el fondo, esta es la principal del juego.

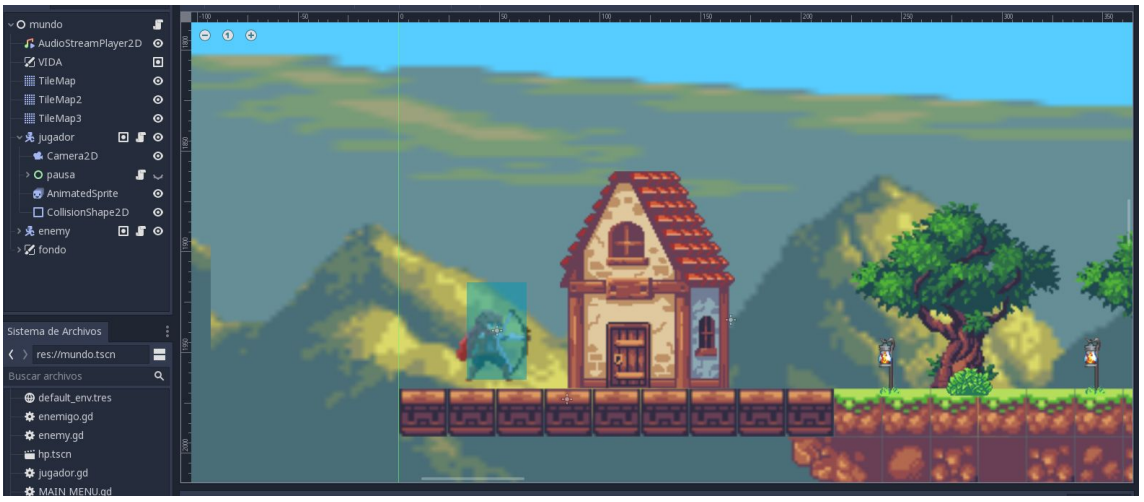


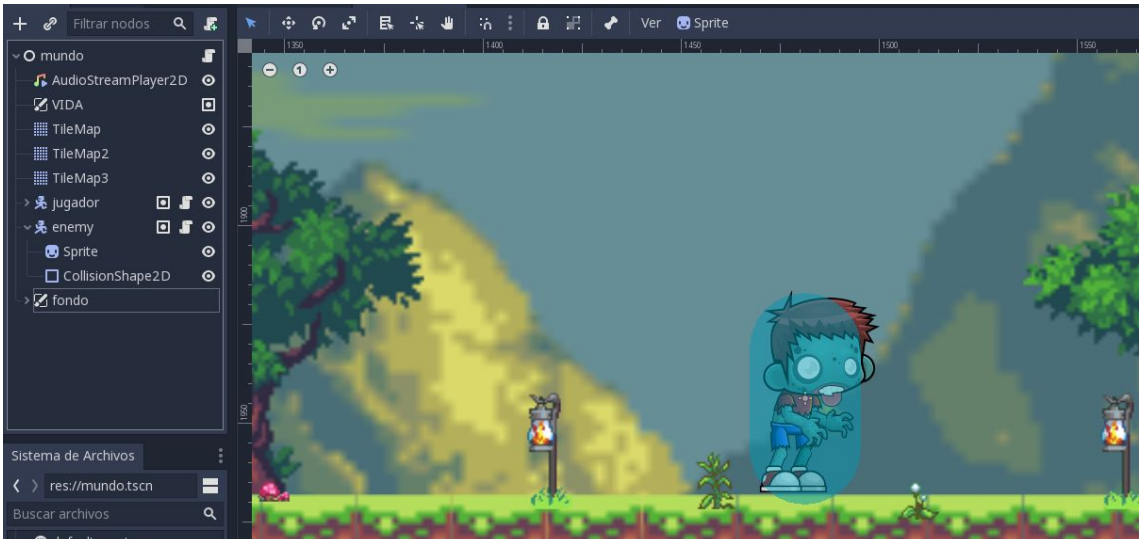
Esto es lo principal que debe tener una escena, cada nodo padre, con su nodo hijo que representan el mapa, jugador, enemigos, etc…

Lo imprescindible es el nodo principal que en este caso lo hemos llamado mundo (**Node2D**) este contiene toda la escena que veremos mientras estamos jugando al videojuego (Personaje, Enemigos, Bloques, Fondo).

También le puedes poner nodos para añadir más características (musica, spawns, temporizadores, etc…), estos no se ven pero son muy necesarios.

2.2.Escena: Jugador





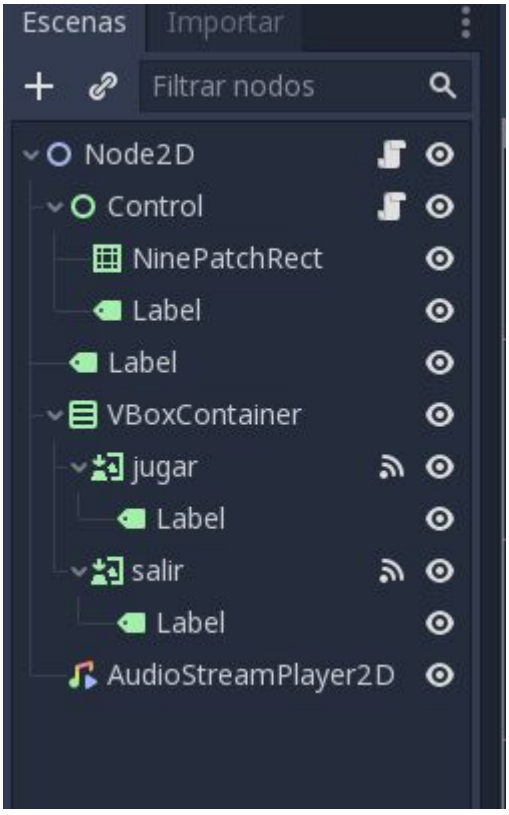
En esta escena se encuentra el nodo del jugador (player) en el que se añaden como nodos hijo un “sprite”, que sería como el aspecto que tendrá nuestro personaje, hemos escogido el que sale en la imagen.

El recuadro azul es un “CollisionShape2D” que lo que representa dentro del juego es como el rango de colisión que tendría nuestro jugador, este nodo se puede configurar a gusto de la acción que se quiera hacer, en el caso del jugador servirá para cuando queramos que haga alguna acción al tocar un enemigo o algún objeto con otro collisionshape.

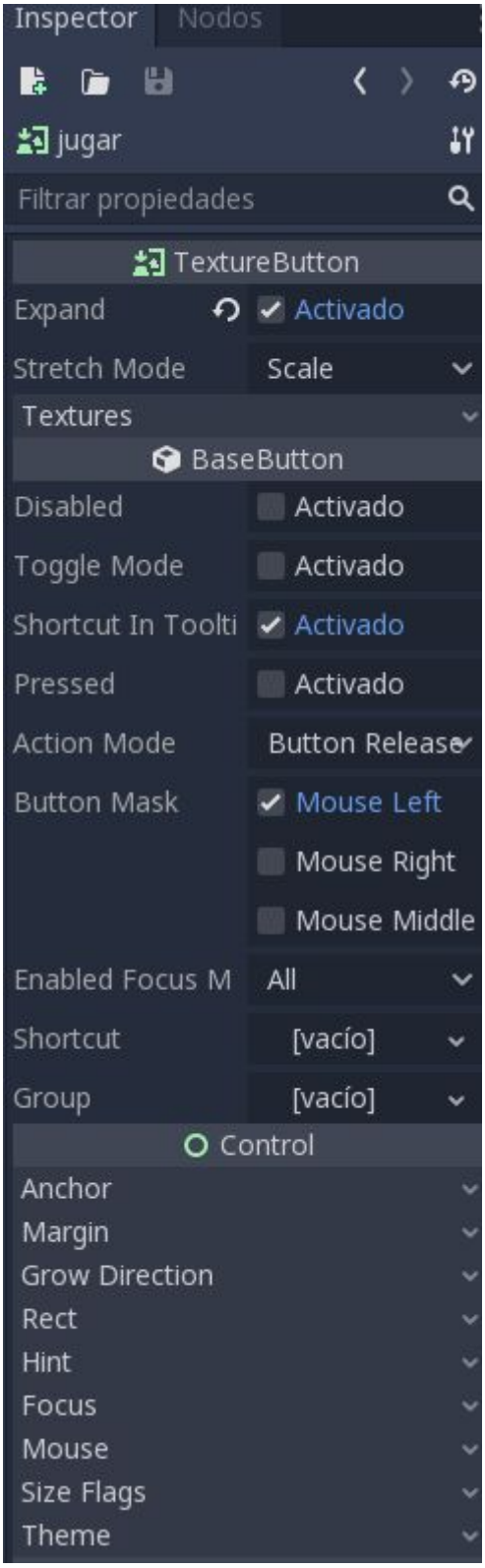
Hemos colocado un nodo que se llama “Camera2D”, su función es que cuando se inicia el juego podamos ver donde está señalando la cámara, está configurada para que la cámara siga al jugador para poder ver todo lo que pasa en cada momento.

Hemos colocada como nodo hijo la función de pausa para que se pueda activar y desactivar una vez iniciado el juego.

Caja herramientas de godot



La escena inicial (MAIN MENU.tscn), el nodo principal que ara todo el trabajo es el (Control), él será el que maneja toda la escena (etiquetas, botones, texturas y sonido), godot nos ofrece una herramienta muy útil para poder conectar los diferentes nodos, por ejemplo en esta escena hemos conectado el botón jugar y el botón salir con el nodo principal control, para que lo gestione cuando se hace click.



Cuando hacemos click en uno de los botones como por ejemplo en el de jugar se nos despliega a la derecha un menú para poder personalizarlo a nuestro gusto, en este caso lo que hemos hecho es poner que la opción Expand para poder hacerlo más pequeño o más grande con el mouse, pero como solo es un botón hubo que ponerle una etiqueta para poder escribir sobre el personalizamos la fuente de la letra, tamaño, color, efecto de sombras.

grupos

Una herramienta muy buena y recomendable son los grupos, para poder juntar todos los enemigos, objetos, etc… , por ejemplo ponemos que todos los que estén en el grupo objetos desaparezcan cuando el jugador pasa por encima.

2.3.Implementacion Scripts

Script: menu

```
extends Node2D

func _ready():
    get_viewport().audio_listener_enable_2d = true
    $AudioStreamPlayer2D.play()
    $AudioStreamPlayer2D.volume_db
```

A cada nodo nos permite añadir scripts y gracias a los scripts es lo que nos permite configurar los nodos a nuestro gusto.
Al agregar un script lo primero que sale en la primera fila es “extends (nombre nodo)”, por en medio aparecen líneas comentadas para explicarnos qué es lo que podemos añadir o lo que hace, una líneas más abajo aparece predeterminadamente una función, como en el ejemplo de arriba.

func _ready():
Está funcion lo que hace es que lee una vez el script una vez iniciado el juego

Script: jugador

```
extends KinematicBody2D
#Constantes para que funcione el juego
const arriba = Vector2(0,-1)
const VELOCIDAD = 210 #Para que camine el personaje
const GRAVEDAD = 420 #
const salto_altura = -210 #Fuerza para saltar
var movimiento = Vector2()
var salto_contador = 0 #Contador inicial de salto
const maximos_saltos_contador = 2
var on_ground = false #Esta tocando el suelo falso
export (PackedScene) var Proyectiles
var esta_atac = false
const flecha = preload("res://flecha.tscn")
var muerto = false
var puntos = 0
export var vida_maxima = 100
export var vida_actual = 100
export var buff_curacion = 1
var caida = false
var moneda_actual
var barra_vida


func _ready():
    set_physics_process(true)
    set_process(true)

    barra_vida = get_tree().get_nodes_in_group("hpj1")[0]

func _process(delta):
    var score = get_node("marcador/RichTextLabel")
    score.text = str(puntos)

func actualizar_barrahp():
    barra_vida.value = vida_actual * barra_vida.max_value /
vida_maxima

...
```

Como hemos explicado antes la primera línea se refiere al nodo, en este caso es KinematicBody2D, a continuación añadimos lo siguiente:

const (objetivo) = x
con la constante asignamos un nombre a una función concreta

var (...)
ponemos por ejemplo que la variable de "movimiento"= al vector2

creamos otra función pero para las físicas del personaje

func _physics_proces(delta):
aquí podemos añadir todos las movimientos que tendrá nuestro personaje

para poder poner la velocidad de nuestro personajes tenemos que aplicar la fórmula de que la velocidad es igual a la aceleración por tiempo (V=a*t), en este caso es:

```
func _physics_process(delta):

    actualizar_barrahp()
    vida_actual += buff_curacion * delta
    vida_actual = clamp(vida_actual, 0, vida_maxima)

    if muerto == false:

      movimiento.y += GRAVEDAD * delta #Fisicas de
movimiento y caida

      if Input.is_action_pressed("ui_right"): #Movimiento
a la derecha
            if esta_atac == false:

                  movimiento.x = VELOCIDAD

                  $AnimatedSprite.flip_h =false #Animacion
sigue recta

                  $AnimatedSprite.play("correr") #Pone la
animacion Correr

            if sign($Position2D.position.x) == -1:
```

```
                $Position2D.position.x *= -1

      elif Input.is_action_pressed("ui_left"):
#Movimiento a la a la izquierda
            if esta_atac == false:

                movimiento.x = -VELOCIDAD

                $AnimatedSprite.flip_h =true #Animacion
cambia hacia atras

                $AnimatedSprite.play("correr")

            if sign($Position2D.position.x) == 1:
                $Position2D.position.x *= -1
      else:
            movimiento.x = 0
            if is_on_ceiling() && esta_atac == false:

                $AnimatedSprite.play("quieto") #Pone la
animacion quieto

      if Input.is_action_just_pressed("ui_up"):
            if esta_atac == false:
                if salto_contador < 2:
                    salto_contador += 1
                    movimiento.y = salto_altura
                    on_ground = false
                    caida = false

      arriba.y += GRAVEDAD

      if Input.is_action_just_pressed("ui_accept") &&
esta_atac == false: # al darle al espacio sale la flecha
            esta_atac = true
            $AnimatedSprite.play("ataque")
            var disparo = flecha.instance()
            if sign($Position2D.position.x) == 1:
                disparo.set_disparo_direccion(1)
            else:
                disparo.set_disparo_direccion(-1)
```

```
            get_parent().add_child(disparo)
            disparo.position = $Position2D.global_position



      if is_on_ceiling():
            if on_ground == false:
                  on_ground = true
                  salto_contador = 0
                  print(on_ground)
      else:
            if esta_atac == false:
                  on_ground = false
                  if movimiento.y < 0:
                        $AnimatedSprite.animation = "salto"
                  else:
                        $AnimatedSprite.animation = "caida"
```

movimiento.y += gravedad * delta

Con el mas (+) podemos hacer que se incremente

Godot nos permite asignar alguna tecla para poder poner un nombre como el ejemplo a continuación:

if Input.is_action_pressed(“nombre”)

en el caso de “nombre” godot nos permite asignar una tecla, hemos asignado las teclas de las flechas del teclado para cuando sean pulsadas el personaje puede hacer diferentes acciones por ejemplo:

← = El personaje se dirige hacia la izquierda de la pantalla
↑ = El personaje da un salto con un máximo de 3 saltos sin tocar el suelo
→ = El personaje se dirige hacia la derecha de la pantalla

if Input.is_action_just_pressed(“ui_accept”)

Es un nombre que está asignada al espacio que en principio es el ataque de nuestro jugador pero con las animaciones se interponen y visualmente no queda bien y no se puede ver la animación de ataque

```
50
51  if is_on_ceiling():
52  #    on_ground = true
53  #    salto_contador = 0
54      if on_ground == false:
55          on_ground = true
56          salto_contador = 0
57          print(on_ground)
58  else:
59      on_ground = false
60      if movimiento.y < 0:
61          $AnimatedSprite.play("doblesalto")
62      else:
63          $AnimatedSprite.play("caida")
64
65
66  movimiento = move_and_slide(movimiento, arriba)
67  pass
68
69  if(Input.is_action_just_pressed("tecla_m")): #reiniciar mundo
70      get_tree().change_scene("res://mundo.tscn")
71  if(Input.is_action_just_pressed("tecla_esc")): #test pausa
72      pass
73
74
75  if(get_slide_collision(get_slide_count()-1) != null):
76      var colisionador = get_slide_collision(get_slide_count()-1).collider
77      if(colisionador.is_in_group("enemigo")):
78          get_tree().change_scene("res://muerto.tscn")
```

Como podemos observar está configurado para poder hacer el doble salto haciendo su animación adecuada y cuando está cayendo tenga otra animación, en este caso de caída.

`if (Input.is_action_pressed(“tecla_m”)):`

Hemos asignado la tecla M para reiniciar el mundo

`if (get_slide_collision(get_slide_count()-1) != null):`

Le estamos preguntando que si a colisionado con algo, y que si colisiona con algo que está dentro del grupo enemigos que aparezca la escena “GAME OVER

Script: enemigo

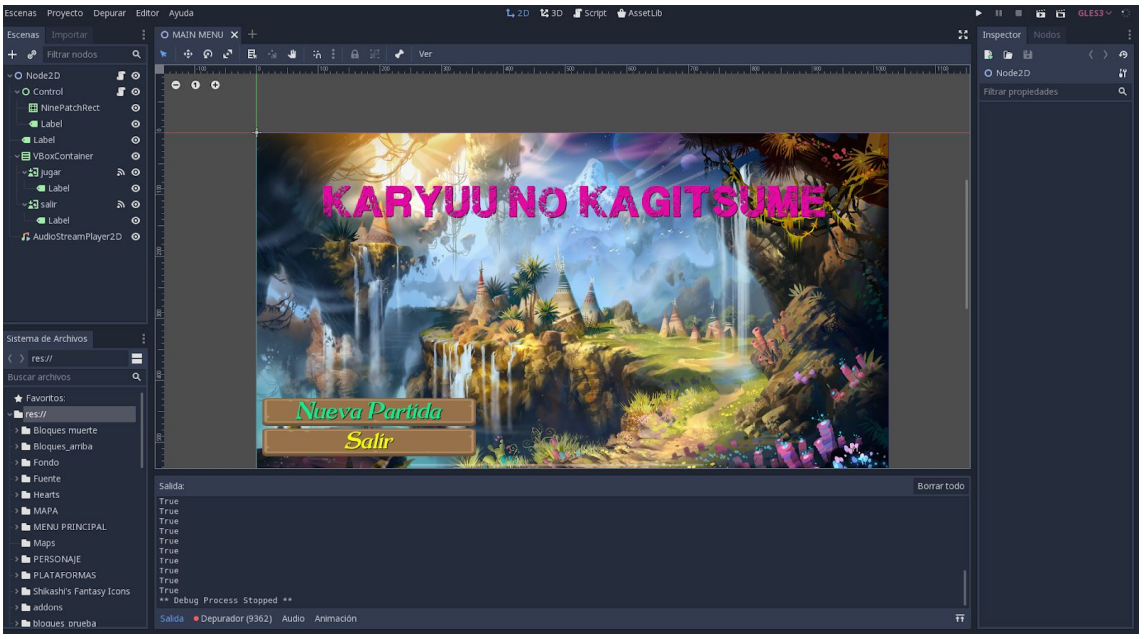
```
extends KinematicBody2D

const VELOCIDAD = 120 #Para que camine el personaje
const GRAVEDAD = 410 #
var movimiento = Vector2()
const techo = Vector2(0,-1)
export(int) var hp = 1

var direccion = 1

var muerto = false

func dead():
    hp = hp - 1

    if hp <= 0:
       muerto = true
       movimiento = Vector2(0, 0)
       $AnimatedSprite.play("muerto")
       $CollisionShape2D.disabled = true
       $Timer.start()


func _physics_process(delta):
    if muerto == false:
       movimiento.x=VELOCIDAD * direccion

       if direccion == 1:
            $AnimatedSprite.flip_h = false
       else:
            $AnimatedSprite.flip_h = true

       $AnimatedSprite.play("correr")

       movimiento.y += GRAVEDAD

       movimiento = move_and_slide(movimiento, techo)

       if is_on_wall():
```

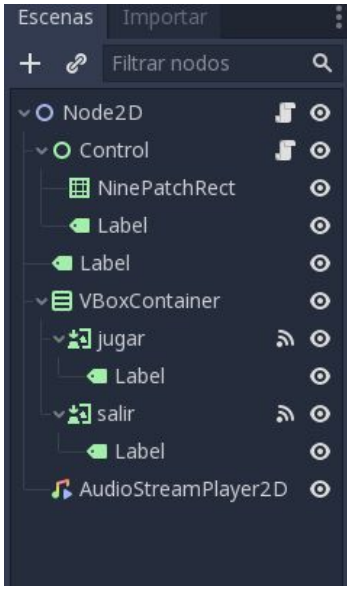
```
            direccion = direccion * -1
            $RayCast2D.position.x *= -1

      if $RayCast2D.is_colliding() == false:
            direccion = direccion * -1
            $RayCast2D.position.x *= -1

      if get_slide_count() > 0:
            for i in range (get_slide_count()):
                  if "player" in
get_slide_collision(i).collider.name:
                        get_slide_collision(i).collider.dead()


func _on_Timer_timeout():
    queue_free()
```

En el script del enemigo hemos logrado que se mueva por sí solo, por ahora la función que tiene es que cuando el jugador colisiona con el enemigo sale una escena diciendo fin del juego. También cuando detecta un borde o una pared gracias al Raycast se dirige hacia la dirección contraria haciendo que se dé la vuelta.

El script consiste en:
Le pregunta si ha colisionado con algo y si colisiona con algo que esté dentro del grupo llamado “player” aparezca la siguiente escena junto a la función muerto.

Creación menu principal del juego





Al iniciar el juego directamente nos llevara al menu y se reproducirá una canción de fondo que se ha añadido dentro del script del nodo padre como si fuese un nodo hijo

```
extends Node2D

func _ready():
    get_viewport().audio_listener_enable_2d = true
    $AudioStreamPlayer2D.play()
    $AudioStreamPlayer2D.volume_db
```

```
extends Control

func _ready():
    pass

func _on_jugar_pressed():
    get_tree().change_scene("res://instrucciones.tscn")

func _on_salir_pressed():
    get_tree().quit()

func _on_Label_gui_input(event):
    pass
```

Como hemos explicado anteriormente la **func_ready():** se ejecuta una vez al iniciar el juego

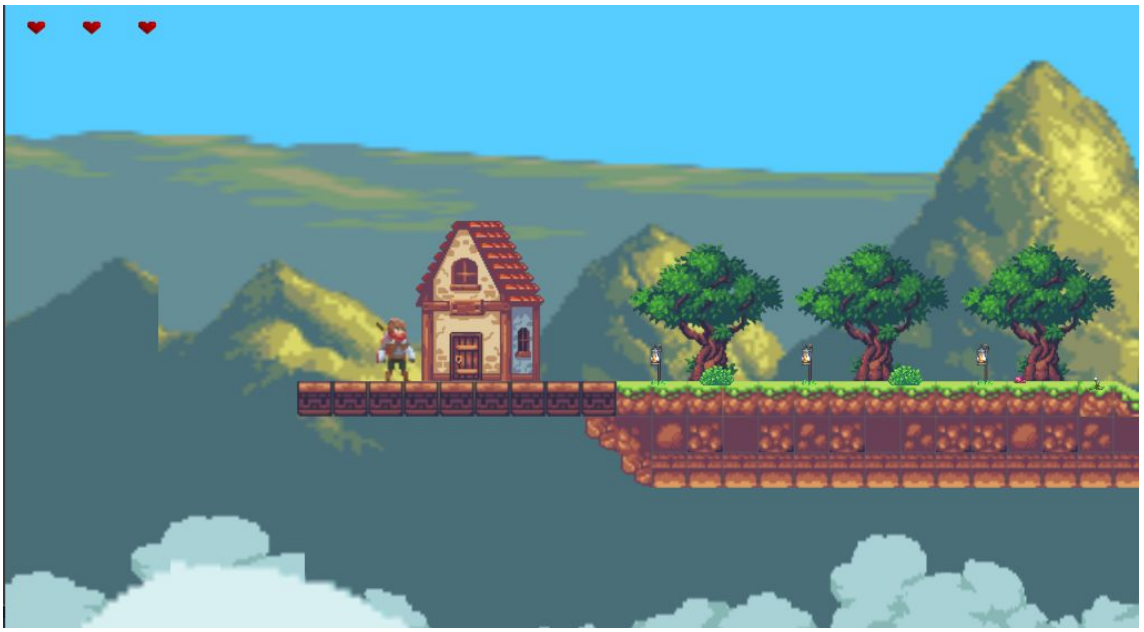
> **`func on_jugar_pressed():`**
> *Está diciendo que cuando se escoja está opcion, nos dirigira a X escena, en este caso nos dirigirá a las escena del juego*

Al iniciar el juego se reproducirá una canción para el menú y nos aparecen dos opciones:

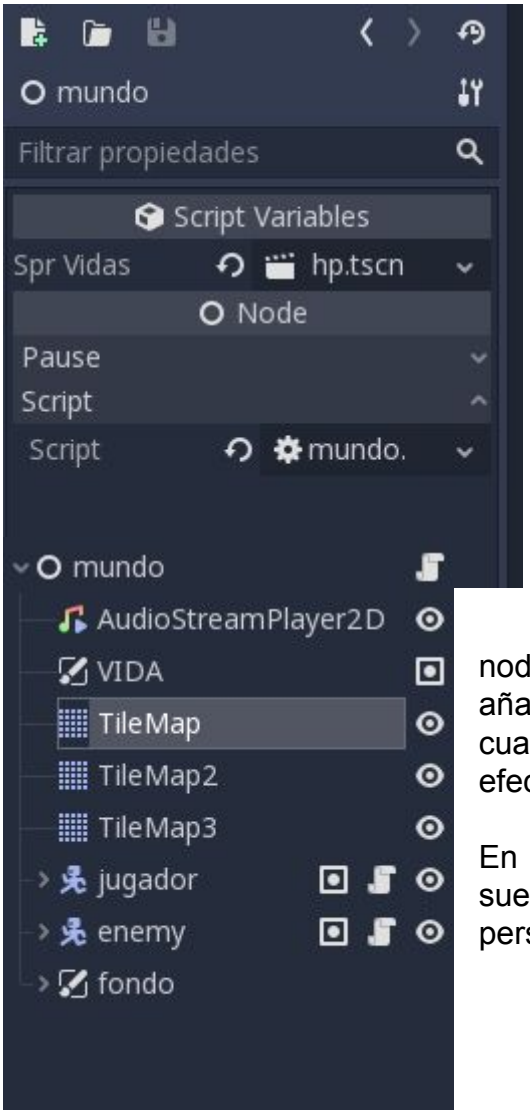
- Nueva partida
- Salir

Darle a la opción de **nueva partida** nos conducirá al juego para poder disfrutarlo, en caso de darle a la opción **salir** se cerrará la venta del juego

Creación de mapa



En esta imagen se puede apreciar todo lo que aparecerá en pantalla durante el juego.

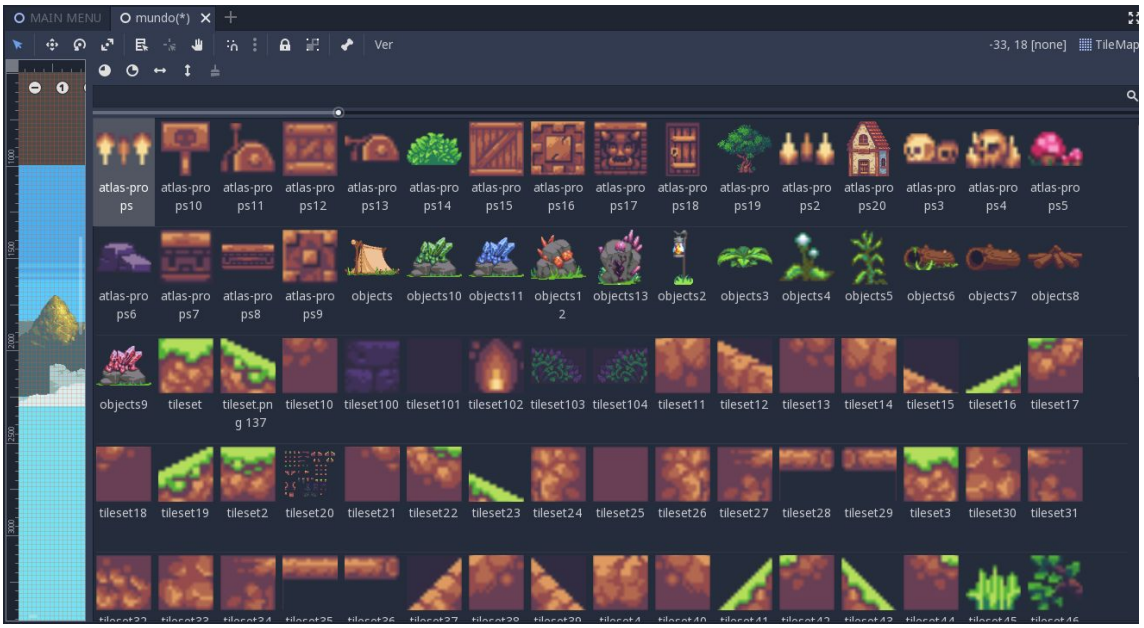


En la imagen de arriba se puede apreciar unos corazones que significan la vida del jugador, esos corazones se han añadido con un script que se encuentra dentro del nodo inicial, que se vincula con una escena dentro del grupo “vida” que hace que la escena que se añade al nodo inicial aparezca solo si hay una escena por encima de él que esté en el grupo “vida”

Durante la creación añadimos un nodo llamado “TileMap” que nos permite añadir el fondo, hemos puesto tres para cuando el personaje se mueva de un efecto de que el fondo se está moviendo

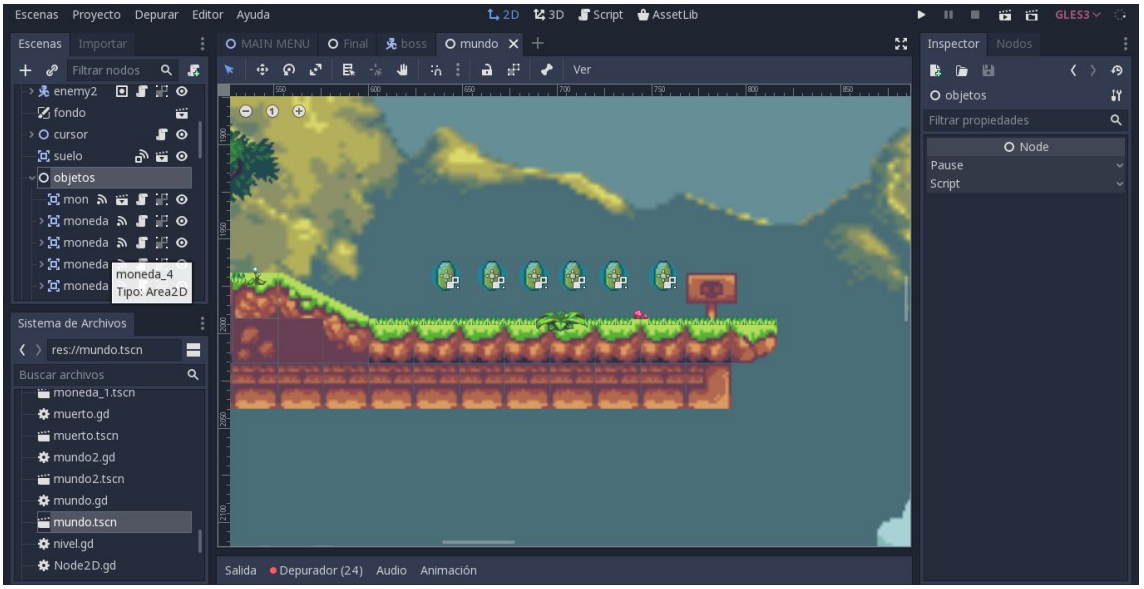
En uno de los TileMap hemos puesto el suelo donde se podrá sostener el personaje para que no caiga al vacío.

En la siguiente imagen se podrá ver qué sprites hemos utilizado:



Estos son los sprites que hemos utilizado para el suelo, todo está encajado para que cada sprite concuerde con el que está a su lado para que visualmente se vea bien, estos sprites han sido descargados de internet.

Monedas



Hemos añadido unos sprites bajados de internet en forma de moneda, le hemos puesto un nodo de area 2D, dentro del nodo se han agregado los nodos: Sprite, Collisionshape2D y un Animationplayer.

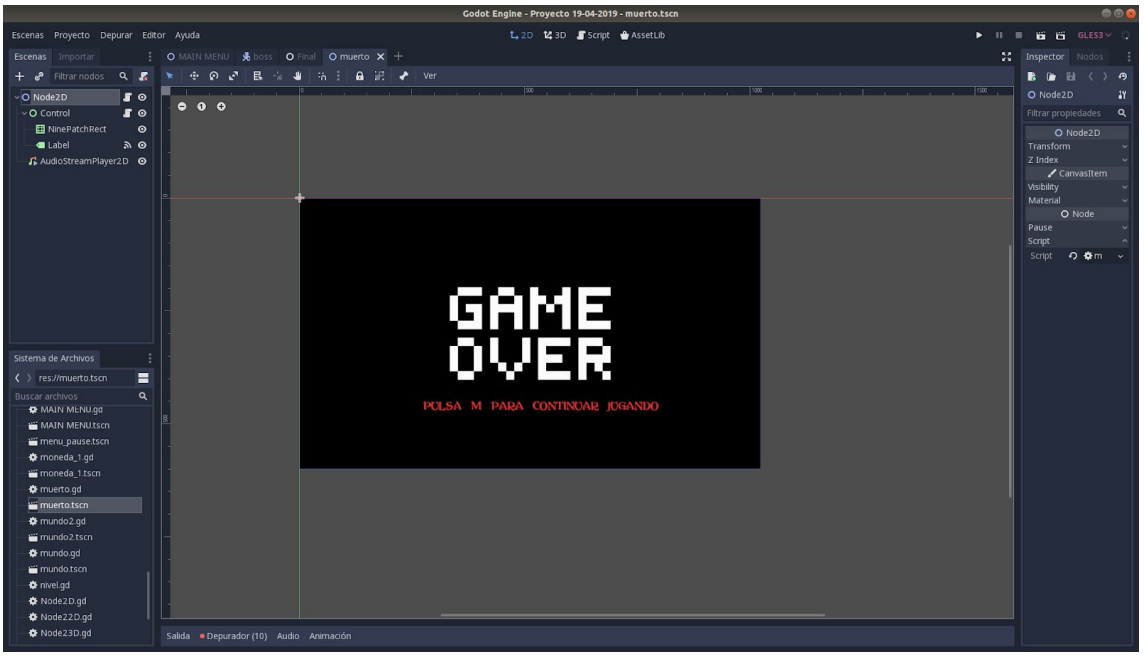
El sprite como hemos explicado anteriormente es para poder poner una imagen para que sea visible dentro del juego, el Collisionshape2D es para que podamos configurarlo como por ejemplo: el jugador toca el área de colisión y pueda hacer X acción. El Animationplayer sirva para poder hacer una animación del sprite, así dando el efecto que la moneda está dando una vuelta sobre sí misma

Sprite moneda

```
extends Area2D

enum estados {quieto}
var estado_actual = estados

func _ready():
    estado_actual = estados.quieto
    get_node("AnimationPlayer").play("quieto")

func _on_moneda_1_body_entered(body):
    queue_free()
```

Para hacer un grupo añadimos **enum** con el nombre **estados** dentro se encontrará una función “quieto”, añadimos una variable en el que decimos que el estado actual es igual a estados y le decimos que el nodo AnimationPlayer ejecute la función **quieto** y cuando lo toque el jugador se borre.

Escena muerto



Hemos añadido una escena que aparece cuando el personaje se queda sin vida, el personaje hace una animación de morirse y cuando acaba sale esta escena con el texto “Pulsa M para continuar jugando”, al pulsar M nos vuelve a la primera escena que es el mundo num.1.

Script

```
extends Node2D

func _ready():
    get_viewport().audio_listener_enable_2d = true
    $AudioStreamPlayer2D.play()
    $AudioStreamPlayer2D.volume_db

func _physics_process(delta):
    if(Input.is_action_just_pressed("tecla_m")): #reiniciar
mundo
      get_tree().change_scene("res://mundo.tscn")
```

En este script lo que hemos puesto es que cuando aparezca salga un audio que dice ”Game over”. Tiene una func para las físicas, le hemos puesto que si se cumple la acción que está entre paréntesis “tecla_m” nos redirige hacia la escena mundo

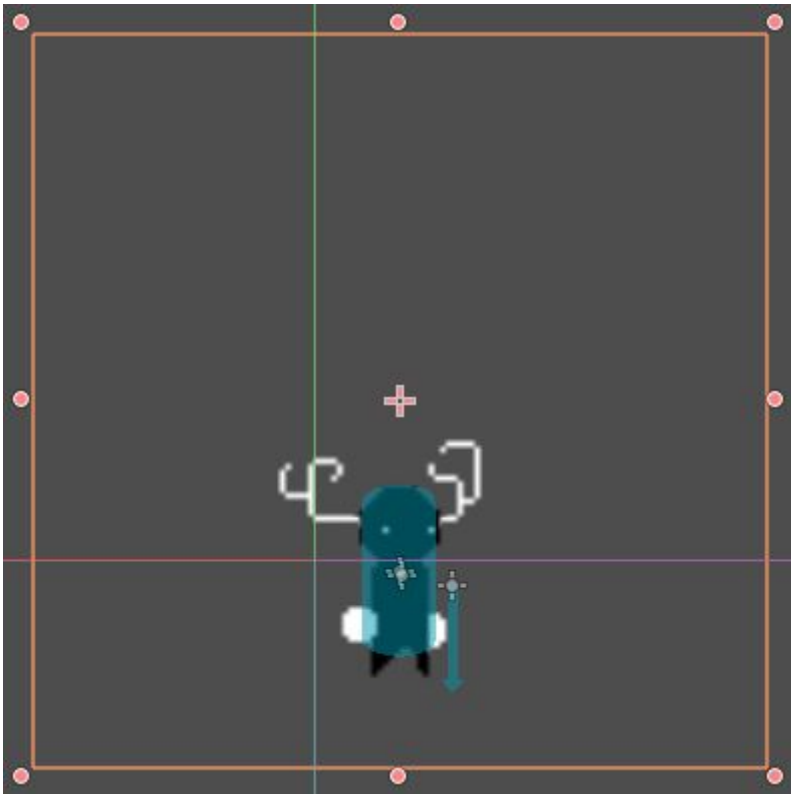
Jugador 1.2

Al jugador le hemos puesto un menu de pause que se activa con el botón “esc” que su función es congelar el juego y para que vuelva a la normalidad tenemos que volver a darle al botón.

```
extends Control

func _input(event):
    if event.is_action_pressed("tecla_esc"):
       var estado_pausa = not get_tree().paused

       get_tree().paused = estado_pausa
       visible = estado_pausa
```

El script tiene una función que si se cumple la siguiente acción “tecla_esc” dice que la variable estado_pausa es igual a que la raíz del directorio está pausado y la raíz paused es igual al estado pausa, que es igual si es visible.

Escena boss



Hemos descargado un sprite gratuito contiene los mismos nodos que el enemigo: un Animatedsprite, un **Collisionshape2D**, un Timer y un **Raycast2D**. Como hemos explicado anteriormente el Animatedsprite está para que se pueda ver dentro del juego, el **collicionshape2D** está para que el jugador pueda detectar un cuerpo y el **Raycast2D** para que pueda detectar algún borde y pueda dar la vuelta para que no caiga fuera del mapa.

Escena instrucciones

Hemos añadido una escena de instrucciones que aparece cuando pasas de la escena menú, en esta escena se informa al jugador los controles del juego.

Script

```
extends Control

func _on_TextureButton_pressed():
    get_tree().change_scene("res://mundo.tscn")
```

En el script agregamos una función de presión por boton que también está en el menú de inicio, su función es que al hacer clic sobre él y hace la acción deseada.

Escena creditos

Para finalizar el proyecto hemos decidido poner como última escena, este final, la escena sale cuando se derrota al enemigo final

Script

```
extends Node

func _on_TextureButton_pressed():
    get_tree().change_scene("res://MAIN MENU.tscn")
```

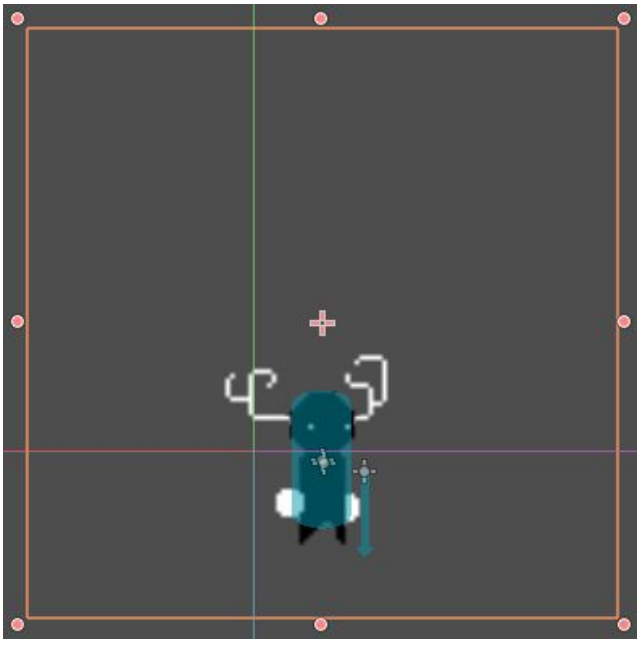
Este script tiene es muy parecido a los anteriores, tiene una función de un botón que al ser presionado por el ratón nos redirige a la escena principal que seria el menu.

3. Bugs

Durante el proyecto hemos sido víctimas de diferentes bugs que molestaban durante la jugabilidad y acciones de nuestro juego. Algunos de esto bugs han sido del sprite del jugador que detectaba como si estuviese en el aire y se muestra la animación de estar quieto y caída, algun otro era por error del **Tilemap**, que se solucionaban fácil cambiando una configuración y otros simplemente que la app no respondía bien y nos veíamos obligados a reiniciar la app.

Uno de los otros bugs era que al cargar la escena por ejemplo de colisión para que el jugador pueda morir no era detectada cuando lo colocamos en otra escena como por ejemplo el mundo 2, para solucionarlo hacemos varias pruebas cambiando distintos valores de las colisiones o el elemento que falle, hasta nos quede con la acción deseada.

El bug del enemigo era que muchas veces no detectaba bien el suelo algunas veces conseguimos solucionarlo pero otras veces recurrimos a un bloque que por alguna razón ese bloque era detectado siempre, muchas veces era que el enemigo tenía el **collision shape2D** muy grande y no se podía cargar bien la escena, otras veces era porque no estaba bien colocado el **Raycast 2D** que simplemente era colocarlo de manera que tocara el suelo para que fuera detectado.



4. Glossario

- **Sprite:** Una imagen que representa al personaje, enemigo u objeto

- **AnimationSprite:** Una serie de imágenes unidas en un mismo archivo una al lado de otra que representa al personaje/enemigos

- **AnimationPlayer:** Un reproductor de animación que se usa para reproducción general de recursos de animación. Contiene un diccionario de animaciones y tiempos de mezcla personalizados entre sus transiciones.

- **Node2D:** Es un objeto de juego 2D, con una posición, rotación, escala y le da control al orden de renderizado del nodo.

- **AudioStreamPlayer:** Sirve para reproducir música de fondo.

- **Escena:** Ejecutar un juego significa ejecutar una escena. Un proyecto puede contener varias escenas, cada nivel del juego es una escena.

- **Boxcollider:** Es un componente básico de un objeto y consiste en una caja invisible alrededor de un objeto y trata sus colisiones. A partir de estas colisiones, se puede tratar en algun script àra realizar diferentes funciones

- **GDScript:** Es el lenguaje nativo de programacion del godot, que se encarga de controlar todas las acciones del juego.

- **Tilemap:** Se utiliza para hacer más fácil la creación del mapa, lo único que se necesitaría serían los sprites junto a sus medidas correspondientes para el uso deseado como por ejemplo el suelo, paredes, etc…

- **KinematicBody 2D**: Se utiliza para hacer personajes, objetos animados.

- **CanvasLayer:** Este es usado para hacer fondo de pantalla animados (Parallax), consiste en hacer capas que se mueven a más velocidad o menos velocidad.

5. Bibliografia

https://godot-doc-en-espanol.readthedocs.io/es/latest/#sec-tutorials

https://godotengine.org/qa/

https://www.reddit.com/r/godot/

https://www.youtube.com/channel/UCgJBHUEJEi7RkjCQx7xR4Wg

https://docs.godotengine.org/es/latest/getting_started/step_by_step/your_first_game.html

https://docs.godotengine.org/es/latest/getting_started/step_by_step/index.html

6. Annexos

Manual del usuario

A continuación vamos a explicar una guia rapida del juego:

Nueva partida:

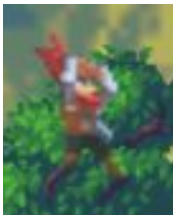
Al darle clic podemos iniciar el juego principal. En el que debemos recoger monedas y matar a los enemigos.Deberemos esquivar o derrotar enemigos, porque si nos quitan toda la barra de vida deberemos volver a comenzar el nivel.

Movimiento jugador:

Para poder movernos por el mapa se usa las flechas izquierda y derecha, flecha arriba para saltar.

 o 

Pulsando la flecha hacia arriba el personaje da un salto, hay hasta 3 saltos.



Al pulsar el espacio el personaje hace la animación de atacar disparando una flecha