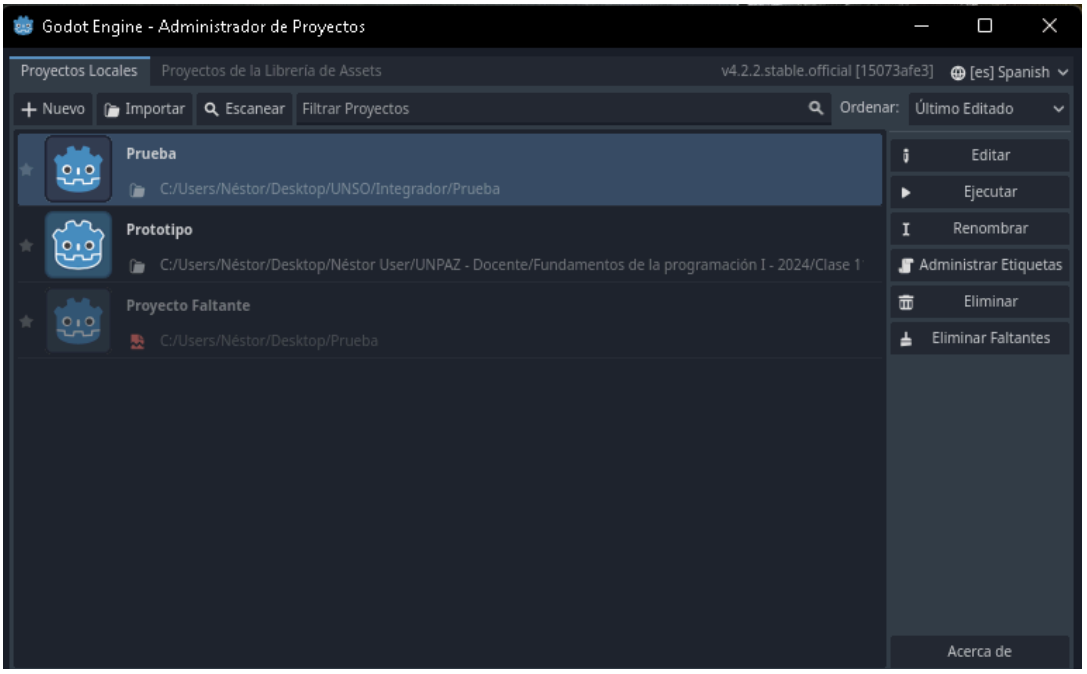
TUTORIAL 1: Captura la bandera

Creación de un proyecto

Utilizaremos GODOT 4.

Hacemos doble click en el ícono de Godot para ejecutar el programa. La primera pantalla que veremos es el gestor de proyectos. Hacemos click en el botón “Nuevo”, nombramos el proyecto, seleccionamos o creamos una carpeta para alojar el proyecto, seleccionamos el renderizador y hacemos click en el botón “Crear y editar”.

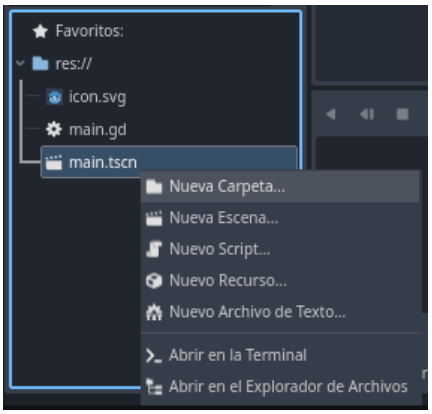


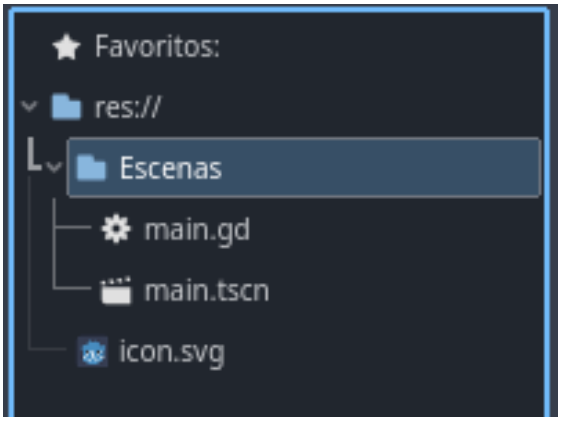
Creación de una escena principal

Ahora estamos ubicados en el área de trabajo de Godot. Por defecto viene predeterminada la vista 3D. Haremos click en el botón 2D que se encuentra en la barra de visualización.

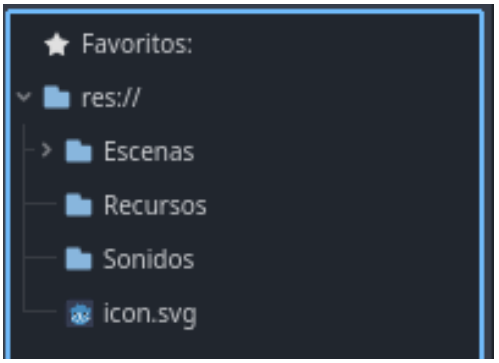
Crearemos una nueva escena con un nodo de control Node2D, lo renombramos Main y adjuntamos un script. Luego guardaremos el proyecto con Ctrl+S.

Una vez que hayamos guardado la escena iremos al sistema de archivos y con click derecho seleccionaremos “Nueva carpeta” de esta forma crearemos una carpeta a la que llamaremos “Escenas” y allí dentro arrastraremos nuestra escena “Main” junto con su script.

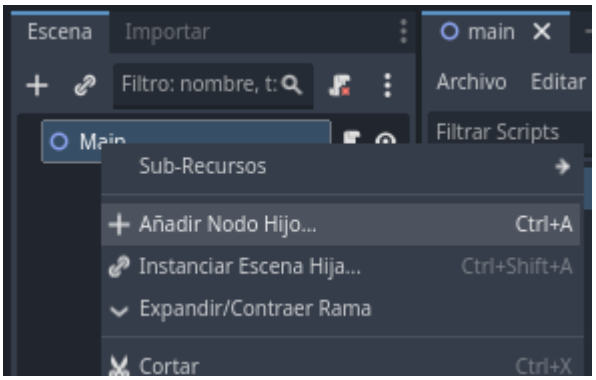


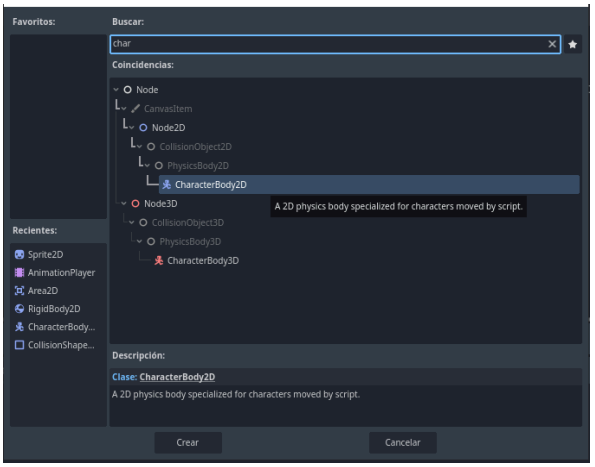


Vamos a adelantarnos algo en la organización y crearemos dos carpetas más para guardar nuestros recursos gráfico y audios, llamaremos a estas carpetas “Recursos” y “Sonidos”.



Luego haremos click derecho en el “Main”, se desplegará una ventana de opciones en donde haremos click en la primera opción “Agregar Nodo Hijo” buscamos y añadimos un “CharacterBody2D”.



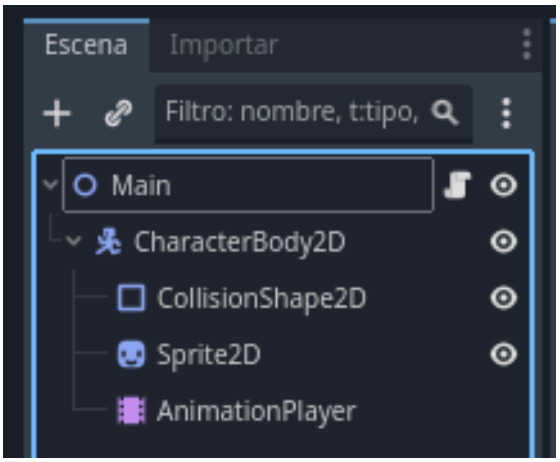


A continuación añadiremos al CharacterBody2D un “CollisionShape2D” y en el inspector seteamos el shape a “Nuevo Capsule Shape2D”

Repetiremos estos pasos para agregar dos nodos más al CharacterBody2D, el nodo “Sprite2D” y el nodo “AnimationPlayer”.

El árbol de nodos debería quedar conformado de la siguiente manera:

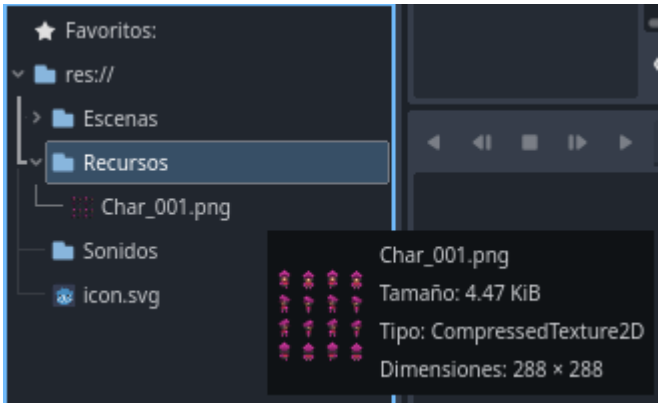
Main
- CharacterBody2D
  - CollisionShape2D
  - Sprite2D
  - AnimationPlayer



Animación

Después de esto seleccionaremos el nodo Sprite2D y ubicamos en el Inspector, el apartado de “Texture” en donde cargaremos el Sprite Sheet de nuestro personaje. El sprite sheet que vamos a utilizar es de 4 sprites de ancho por 4 de alto, cada una de las filas de sprites

corresponde a una animación de 4 frames (cuadros), y lo cargaremos dentro de la carpeta "Recursos" arrastrándolo desde la carpeta de windows al sistema de archivos de Godot y luego a la carpeta "Recursos".



Desplegamos la pestaña "Animation" en el inspector y seteamos el Vframe en 4 y el Hframe en 4.

Seleccionamos el nodo AnimationPlayer en el área de escenas y esto abrirá un sector de trabajo de animaciones en la parte inferior de la pantalla.

Hacemos clic en "Animación" → "Nuevo" y le ponemos un nombre a la animación que vamos a crear, la llamaremos "Abajo".

Ubicamos la imagen de un reloj que indica la duración de la animación y lo seteamos en 0.8, ubicamos el "Ajuste" que indica el paso de la animación y lo seteamos en 0.2. (Esta configuración puede variar a gusto del desarrollador. Cuantos más frames tenga la animación, más fluido será el movimiento, por lo cual, el seteo de tiempos debería modificarse.)

Seleccionamos el nodo Sprite2D y en el inspector ubicamos la propiedad "frame". Seteamos el valor de frame en 0 y luego hacemos click en la imagen de la llave que está al lado. Hacemos clic en "Aceptar" en la ventana de creación de llave. Luego movemos el lector del frame al segundo 0.2, cambiamos el frame a 1 y volvemos a hacer click en la imagen de la llave que está al lado del número del frame. Repetimos esto hasta tener los cuatro frames en la línea de tiempo.

En este punto ya hemos creado la animación con la cual nuestro personaje podrá caminar hacia abajo.

Realizaremos los mismos procedimientos para configurar el resto de las animaciones.

Programando el movimiento

Hacemos click derecho en el nodo CharacterBody2D y luego en "Renombrar", lo llamaremos "Personaje". Estando seleccionado el nodo "Personaje" haremos clic en el botón "Añadir nuevo script". En la ventana que se ha abierto, ubicaremos la propiedad "Plantilla" y le sacaremos la tilde, luego hacemos click en el botón "Crear".

Una vez abierta la hoja de script, debajo del "extends CharacterBody2D" agregaremos:

```
var movimiento = Vector2()
var velocidad = 3

func _physics_process(delta):
	if Input.is_action_pressed("ui_right"):
		movimiento.x = velocidad
		$AnimationPlayer.play("Derecha")

	if Input.is_action_pressed("ui_left"):
		movimiento.x = -velocidad
		$AnimationPlayer.play("Izquierda")
```

```
        if Input.is_action_pressed("ui_up"):
            movimiento.y = -velocidad
            $AnimationPlayer.play("Arriba")

        if Input.is_action_pressed("ui_down"):
            movimiento.y = velocidad
            $AnimationPlayer.play("Abajo")

        if Input.is_action_just_released("ui_right") or
Input.is_action_just_released("ui_left") or Input.is_action_just_released("ui_up") or
Input.is_action_just_released("ui_down"):
            movimiento.x = 0
            movimiento.y = 0
            $AnimationPlayer.stop()

        move_and_collide(movimiento)
        pass
```

De este modo ya hemos diseñado y programado nuestro núcleo de juego.

¿Cómo podemos mejorar nuestro código?

Exacto!!! Subdividiendo y nombrando adecuadamente!!!

```gdscript
extends CharacterBody2D

var movimiento = Vector2()
var velocidad = 3

func _physics_process(delta):
    Mover()
    pass
```

```gdscript
func Mover():
    Mover_Derecha()
    Mover_Izquierda()
    Mover_Arriba()
    Mover_Abajo()
    Parar()
    move_and_collide(movimiento)
    pass
```

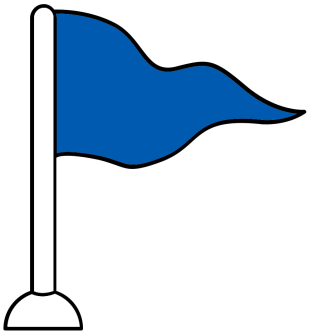
```gdscript
func Mover_Derecha():
    if Input.is_action_pressed("ui_right"):
        movimiento.x = velocidad
        $AnimationPlayer.play("Derecha")
    pass

func Mover_Izquierda():
    if Input.is_action_pressed("ui_left"):
        movimiento.x = -velocidad
        $AnimationPlayer.play("Izquierda")
    pass
```

```gdscript
func Mover_Arriba():
    if Input.is_action_pressed("ui_up"):
        movimiento.y = -velocidad
        $AnimationPlayer.play("Arriba")
    pass

func Mover_Abajo():
    if Input.is_action_pressed("ui_down"):
        movimiento.y = velocidad
        $AnimationPlayer.play("Abajo")
    pass
```

```
43 func Parar():
44     if (Input.is_action_just_released("ui_right") or
45         Input.is_action_just_released("ui_left") or
46         Input.is_action_just_released("ui_up") or
47         Input.is_action_just_released("ui_down")):
48         movimiento.x = 0
49         movimiento.y = 0
50         $AnimationPlayer.stop()
51     pass
52
```

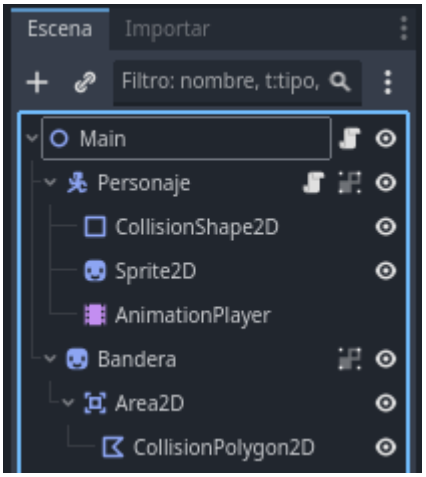
De este modo ya hemos diseñado y programado el movimiento de nuestro personaje, nuestro núcleo de juego. Ahora iremos añadiendo funcionalidades hasta lograr un prototipo de juego tipo “captura la bandera”.

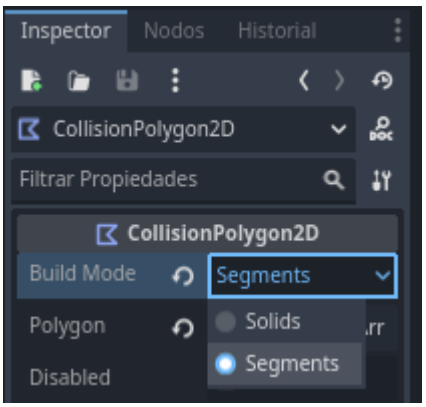
Creando una bandera.



Agregar un Sprite arrastrando la bandera desde la carpeta “Recursos” (previamente cargado en dicha carpeta) del área de sistemas de archivos hasta el área de trabajo. De ser necesario la renombramos Bandera.

Le agregaremos un Area2D, luego un CollisionPolygon2D y en el inspector marcamos el Build Mode en “Segments”, ajustamos el shape al Sprite y modificamos el tamaño.





Creando un enemigo.

Los pasos a seguir son similares a los de la creación de un personaje.

Crear un CharacterBody2D, renombramos “Enemigo”, agregar al Enemigo un CollisionShape2D y un Sprite2D. Agregar una textura al Sprite, utilizaremos para esto el fantasmita que está como recurso en el aula virtual. Luego agregamos un shape al CollisionShape2D. Ajustar el shape al Sprite2D.

Agregar un script al Enemigo, con el siguiente código:

Entre el extend CharacterBody2D y el func _ready():

```
var movimiento = Vector2()
var velocidad = 3
```

En el func _physics_process(delta):

```
move_and_collide(movimiento)
set_vector(get_node("../Bandera").global_position - global_position)
pass
```

y luego la función…

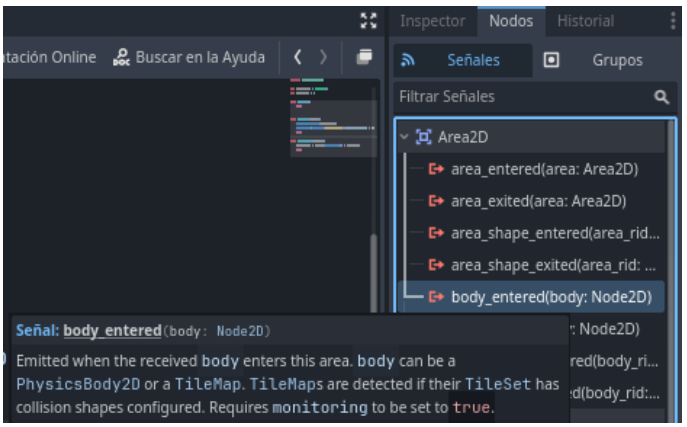
```
func set_vector(vector):
    movimiento = vector.normalized() * velocidad
    pass
```

```gdscript
extends CharacterBody2D

var movimiento = Vector2()
var velocidad = 3

func _physics_process(delta):
    move_and_collide(movimiento)
    set_vector(get_node("../Personaje").global_position - global_position)
    pass

func set_vector(vector):
    movimiento = vector.normalized() * velocidad
    pass
```

Con todo lo hecho hasta ahora tenemos un personaje que podemos mover en cuatro direcciones y un enemigo que se mueve automáticamente. Pero aún no funciona ya que faltan las condiciones de victoria y derrota.

Creando una condición de victoria/derrota

Seleccionaremos el nodo Area2D que pertenece a la bandera y haremos click en la pestaña Nodo que se encuentra al lado de la pestaña del Inspector, ubicamos la señal body_entered(body: Node), haremos doble click en ella y luego click en el Main para conectar la señal con el script.



Agregamos el siguiente script al Main dentro de la función que se ha creado:

```
func _on_Area2D_body_entered(body):
    if body.name == "Personaje":
        $Enemigo.queue_free()
        $Bandera.queue_free()
```

```
        pass

if body.name == "Enemigo":
        $Personaje.queue_free()
        pass
```

```gdscript
14 func _on_area_2d_body_entered(body):
15     if body.name == "Personaje":
16         $Enemigo.queue_free()
17         $Bandera.queue_free()
18         pass
19
20     if body.name == "Enemigo":
21         $Personaje.queue_free()
22         pass
```

Mejorando el código.

Hasta ahora todo bien… Pero… Debemos mejorar la organización y estructura del código según lo visto.

Separando escenas

La bandera debe ser una escena independiente. Para ello vamos a hacer click derecho en el nodo Bandera y seleccionamos la opción “Guardar rama como escena”. Guardaremos esta escena dentro de la carpeta Escenas.

Lo mismo haremos con el personaje y el enemigo.

¿Funciona nuestro prototipo? ¿Por qué dejó de funcionar?

El personaje y el enemigo siguen funcionando, en particular el enemigo gracias al método ***get_node()*** puede en cualquier ámbito en el que esté instanciado buscar y obtener a cualquier nodo llamado “Bandera”.

Pero la bandera, que antes pertenecía al ámbito del Main ahora es una escena independiente, instanciada dentro del Main, pero independiente, y por eso la conexión directa que tenía con el Main mediante la su señal se ha perdido.

Vamos a arreglarlo.

Corrigiendo y mejorando el código de la bandera

Vamos a ir a la escena bandera y adjuntarle un script, debemos desconectar y volver a conectar la señal ***body_entered(body: Node2D)*** con el nodo padre, el sprite “Bandera”. El script debe quedarnos con el siguiente código así:

```
extends Sprite2D

signal gana
signal pierde

func _on_area_2d_body_entered(body):
    if body.name == "Personaje":
        emit_signal("gana")
    else:
        emit_signal("pierde")
        pass
    pass
```

Con este código la bandera detecta cuál es el cuerpo que la está tocando y simplemente envía una señal. Esta señal puede ser recibida por cualquier nodo, así haremos que este sea el Main.

Corrigiendo y mejorando el código del Main

Seleccionamos la escena instanciada “Bandera” en el Main y si observamos en las señales veremos que han aparecido las señales gana() y pierde() que creamos desde la escena “Bandera”. Ahora podemos conectarlas con el Main.

Eliminamos la función anterior y las nuevas deben quedar con el siguiente código así:

func _on_bandera_gana(): *$Enemigo.queue_free()* *$Bandera.queue_free()* *pass*	*func _on_bandera_pierde():* *$Personaje.queue_free()* *pass*

Agregando Splash Screen y un Menú Principal.

¿En qué consisten las Splash Screens? Es un recurso tan sencillo como un fondo simple, generalmente blanco o del color principal de la aplicación, junto con el o los logotipos de nuestra aplicación, videojuego, empresas, etc, y un breve texto como el nombre de la app o las empresas por ejemplo.

En videojuegos que utilizan muchos recursos se usa de pantalla de carga. Es muy importante ya que junto al menú principal son la carta de presentación del videojuego.

El Menú Principal se compone de un conjunto de submenús, en videojuegos son botones, desplegables, links, etc, que nos permite realizar las primeras interacciones con el videojuego. Normalmente está dispuesto en una pantalla con fondo y diseño definido. Desde aquí se accede al resto de sistemas que componen el videojuego.

Agregando un Menú Principal.

1. Creamos una nueva escena con un nodo de control Node2D como padre al que renombraremos “Menu”.

2. Agregaremos al menú un nuevo nodo “ColorRect”, lo renombramos “ModalJugar”, lo ubicaremos en el centro de la pantalla y modificaremos su tamaño estirándolo desde una de sus esquinas.
3. Agregaremos al nodo menú un nodo “Button”, cambiamos su nombre por “Jugar” y en la propiedad Text del inspector escribiremos “Jugar” y lo ubicamos en el centro del ColorRect.
4. Añadiremos una señal “pressed()” del botón y la conectaremos con el menú al que anteriormente le adjuntamos un script. En el script del menú, agregaremos las siguientes líneas de código:

```
1  extends Node2D
2
3  signal jugar
4
5  func _on_Jugar_pressed():
6      emit_signal("jugar")
7      pass
```

5. En la escena del Main instanciaremos el menú y luego en el script del Main conectaremos la señal del menú “jugar”.
6. En el _ready(): añadiremos la siguiente línea de código:
    a. func _ready():
        get_tree().paused = true
        pass

```
1  extends Node2D
2
3  func _ready():
4      get_tree().paused = true
```

7. Dentro de la función _On_Menu_Jugar() añadimos el siguiente código:
    a. func _on_menu_jugar():
        get_tree().paused = false
        $Menu.visible = false

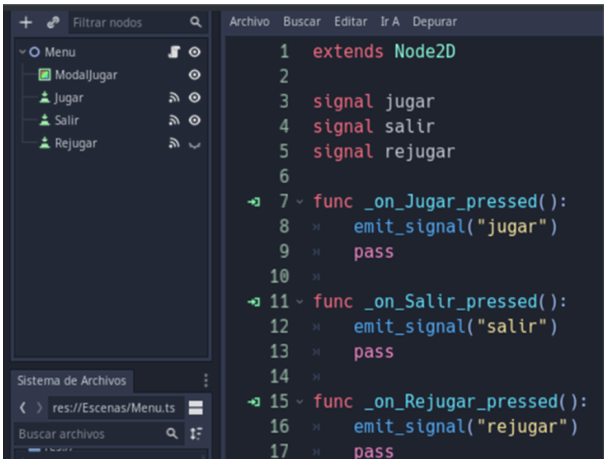
pass

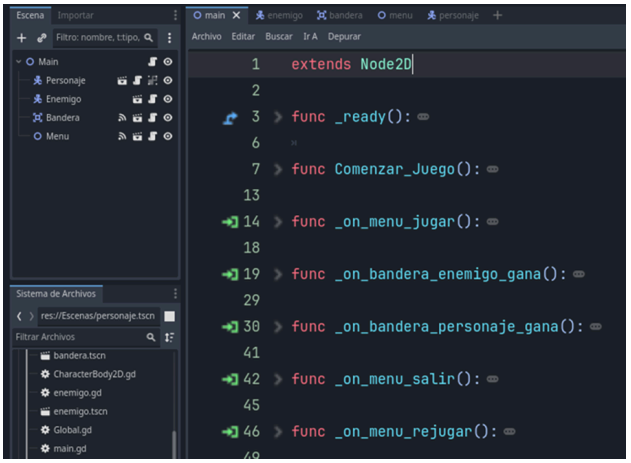
```
14  func _on_menu_jugar():
15      get_tree().paused = false
16      $Menu.visible = false
17      pass
```

8. Ahora seleccionaremos el menú, ubicaremos en el inspector la propiedad “Process”, desplegaremos la pestaña “Mode” y dejaremos tildada la opción “Always”.

Ya para este punto iremos punteando las acciones a realizar entendiendo que ya están familiarizados con el entorno.

9. **CONFIGURAR LAS RESTANTES INTERACCIONES:**
    a. Agregar un botón “Volver a Jugar” y otro “Salir” en el menú.
    b. Añadir las funciones necesarias para que cada botón funcione de la forma esperada.
    c. Terminar de configurar las funcionalidades necesarias para que el juego quede pausado cuando se ejecuta la condición de victoria o de derrota.





Agregando un splash screen.

1. Crear una nueva escena.

2. Proyecto→ Ajustes de Proyecto→ Rendering→ Environment.
3. En Environment setear el Default Clear Color en el color de fondo que deseamos. Luego Cerrar.
4. Creamos una escena SSMain con un nodo de control.
5. Al nodo Main le agregamos un Label, un Sprite y un AnimationPlayer.
6. Descargamos una fuente de la web de tipo .ttf.
7. Seleccionamos el Label y en el Inspector agregamos el texto, luego nos dirigimos a Custom Fonts, hacemos click en vacío y seleccionamos Nuevo DynamicFont.
8. Hacemos click en Nuevo DynamicFont y luego en Font.
9. Agregamos la fuente descargada al Sistema de Archivos y la arrastramos hasta el desplegable vacío.
10. Desplegamos Settings y seteamos el tamaño de la fuente y la ubicación centrada dentro de la pantalla.
11. Seleccionamos el Sprite y cargamos el Texture con el ícono de Godot. Lo centramos en la pantalla.
12. Seleccionamos el AnimationPlayer y creamos una nueva animación llamada Splash_Screen.
13. Seleccionamos el Sprite, en el Inspector seleccionamos Visibility y desde allí hacemos click en Modulate y modificamos el canal alfa a 0.
14. Hacemos click en la llave a un costado de Modulate y le damos crear en la ventana emergente.
15. Seteamos la duración de la animación en 8.
16. Del segundo 0 al 2 seteamos el pasaje del alfa de 0 a 100.
17. Del 2 al 4 se mantiene.
18. Del 4 al 6 vuelve a 0.
19. Hacemos lo mismo para el Label pero seteamos el Modulate con un retraso de 2 segundos.
20. Del 2 al 4 aparece.
21. Del 4 al 6 se mantiene,
22. Del 6 al 8 desaparece.

23. Marcamos Autoreproducir al Cargar.
24. Seleccionamos el nodo SSMain y agregamos un script.
25. Seleccionamos el AnimationPlayer y cambiamos de Inspector a Nodo.
26. Conectamos la señal animation_finished(anim_name: String) al script.

    En el cuerpo de la función agregamos la siguiente línea de código:

    get_tree().change_scene("res://Menu.tscn")

Variables globales

1. Crearemos un nuevo script desde el visor de script Archivo Nuevo script.
2. Renombramos el script a “Global.gd”
3. Agregamos la siguiente variable booleana:
    a. var rejugar : bool = false

```
1  extends Node
2
3  var rejugar : bool = false
4
```

4. Ahora haremos que el script “Global.gd” sea un contenedor de variables gobales.
5. Proyecto Configuración del Proyecto Autoload
6. En ruta agregamos el “Global.gd” y presionamos “Añadir”, nos aseguramos de que quede tildada como variable global y cerramos.

Lo último

1. Desde el menú emitimos las señales “Jugar”, “Salir” y “rejugar”. En el Main administramos las funcionalidades.

```
19 func _on_bandera_enemigo_gana():
20     $Personaje.queue_free()
21     $Menu.visible = true
22     $Menu/Jugar.visible = false
23     $Menu/Jugar.disabled = true
24     $Menu/Rejugar.visible = true
25     $Menu/Rejugar.disabled = false
26     Global.rejugar = true
27     get_tree().paused = true
28     pass
```

```
30 func _on_bandera_personaje_gana():
31     $Enemigo.queue_free()
32     $Bandera.queue_free()
33     $Menu.visible = true
34     $Menu/Jugar.visible = false
35     $Menu/Jugar.disabled = true
36     $Menu/Rejugar.visible = true
37     $Menu/Rejugar.disabled = false
38     Global.rejugar = true
39     get_tree().paused = true
40     pass
```

2. Ahora en el Main conectamos la función _on_menu_rejugar() y añadimos la siguiente línea de código:

    a. func _on_menu_rejugar():
        get_tree().reload_current_scene()
        Pass

```
46 func _on_menu_rejugar():
47     get_tree().reload_current_scene()
48     pass
```

3. Por último modificamos el _ready(): del Main añadiendo la siguiente línea de código:

    a. func _ready():
        if Global.rejugar:
            _on_Menu_jugar()
        else:
            get_tree().paused = true
        pass