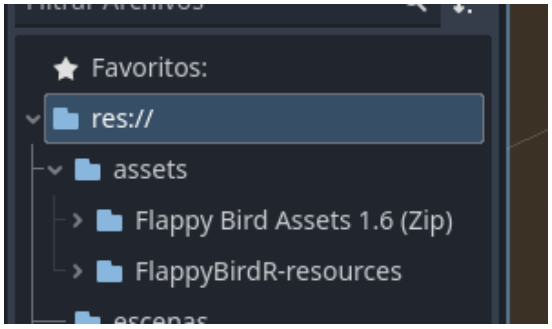
PÁJARO VOLADOR

Hecho por Maikel Yoel Romero Loayza 2ºDAM

Índice

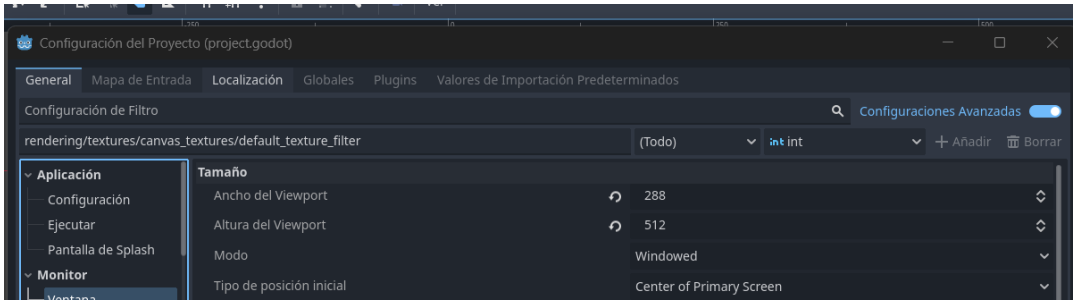
CREACIÓN DE ESCENAS

En primer lugar, encuentro muchos assets en internet pero me voy a quedar con unos que he encontrado en “Itch.io”(están en la bibliografía) y unos que me dan dos tutoriales de youtube que estoy siguiendo o he seguido para hacer el juego. Procedo a descargarlos para usarlos. Luego, los agregamos a Godot moviéndolo a la carpeta correspondiente:

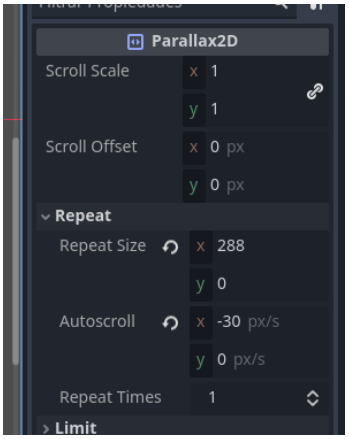


-Una vez tengo en su sitio puedo empezar, vamos a definir el tamaño de la pantalla del juego, lo primero:

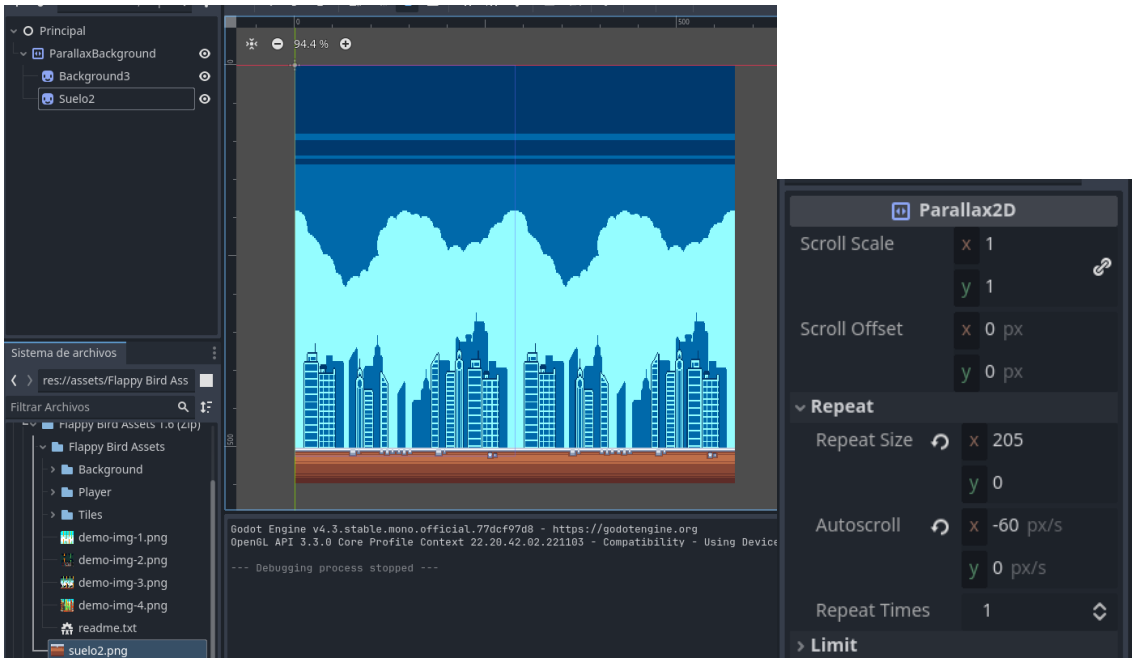
En “Proyecto >> Configuración del Proyecto >> Ventana” ponemos el tamaño que queremos usar para la ventana principal, en nuestro caso 288x512, que será el tamaño de la imagen de background que usaremos:



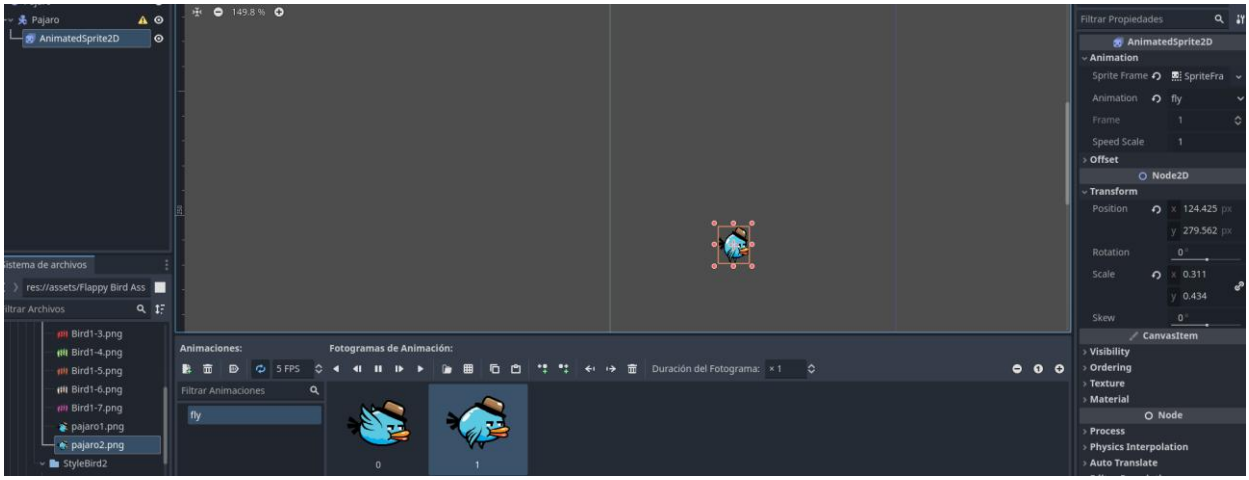
-Ahora creo un parallax en el que pondremos la siguiente configuración para que la imagen se repita de nuevo y para que conforme este ejecutándose el juego que se vaya moviendo el background(Repeat Size para repetir la imagen y Autoscroll para que se desplaze):

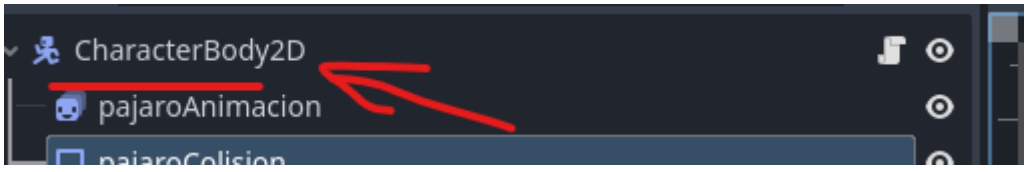


-Repito creando otro parallax, en este caso, para el suelo y le pongo características igual que como con el background pero ponemos la velocidad al desplazarse el doble de rápido para que se vea mejor a la hora de que esté corriendo el juego:



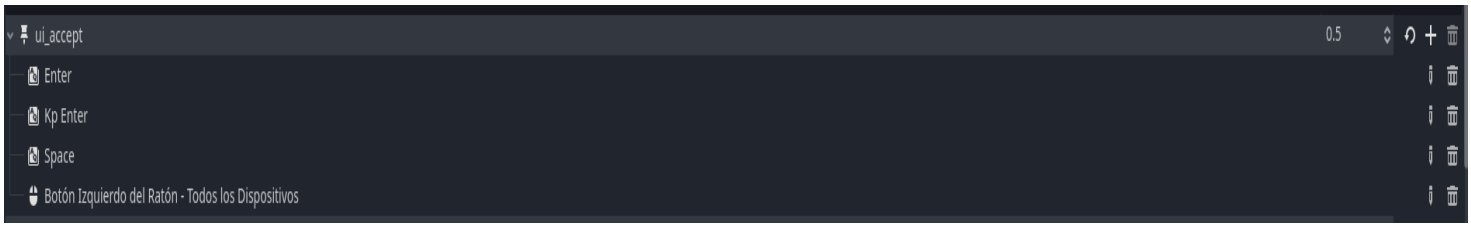
-Ahora creamos el personaje principal creando una nueva escena tipo “CharacterBody2D”, en nuestro caso, un pájaro usando una imagen sacada de internet ya que las de los assets no me servían, arrastramos las 2 imágenes, la normal y la otra cuando tiene las alas abajo. De esta forma con el AnimatedSprite2D el pájaro siempre que esté en el aire estará haciendo el efecto de aleteo que le ponemos:





-Nos tenemos que acordar de pulsar al lado de los FPS para que luego funcione la animación, si no, no funcionará(que es lo que me pasó a mi). Aquí es donde se reproduce el aleteo del pájaro cuando esté en el aire:

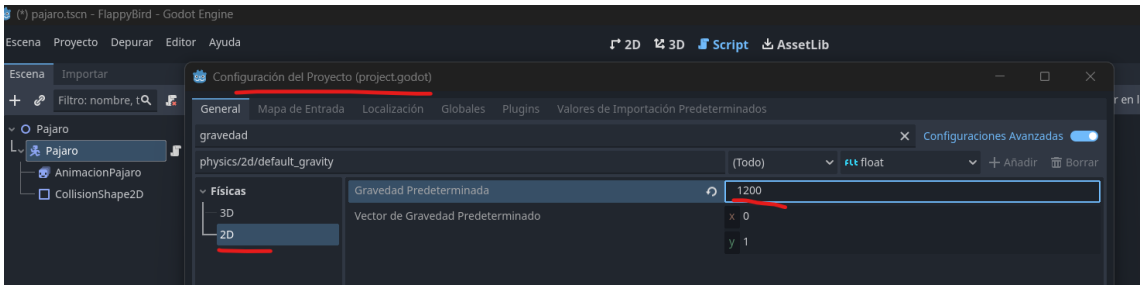
-Nos metemos en “Proyecto >> Configuración del Proyecto >> Mapa de Entrada >> Clickamos en Mostrar Acciones Integradas”, luego buscamos la tecla “space” clickamos y en un nuevo script que creamos ponemos lo siguiente:



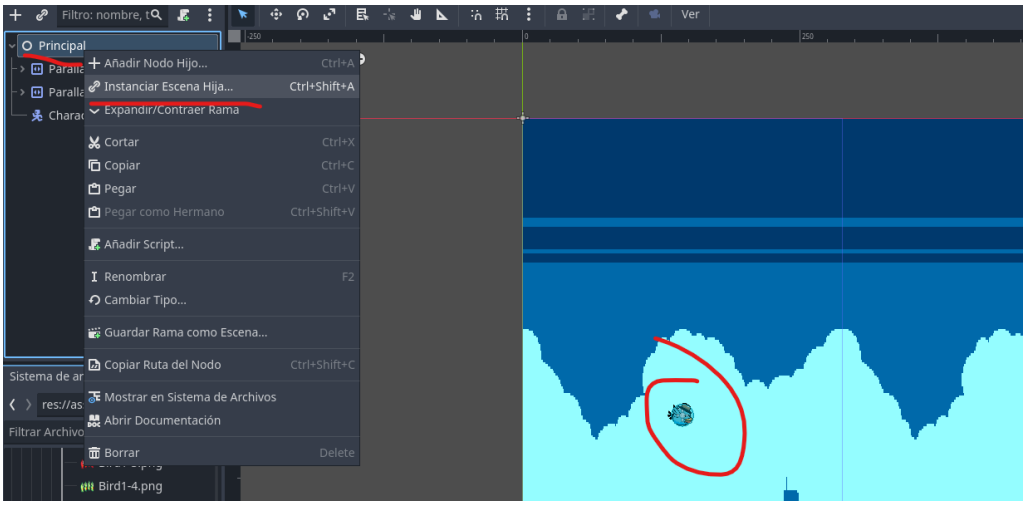

```gdscript
extends CharacterBody2D


const SPEED = 300.0
const JUMP_VELOCITY = -400.0

#CONFIGURACION GRAVEDAD
func _physics_process(delta: float) -> void:
	# Add the gravity.
	if not is_on_floor():
		velocity += get_gravity() * delta
#CONFIGURACION DEL SALTO
	# Handle jump.
	if Input.is_action_just_pressed("ui_accept"):
		velocity.y = JUMP_VELOCITY


	move_and_slide() #sin esto no se mueve el pajaro
```

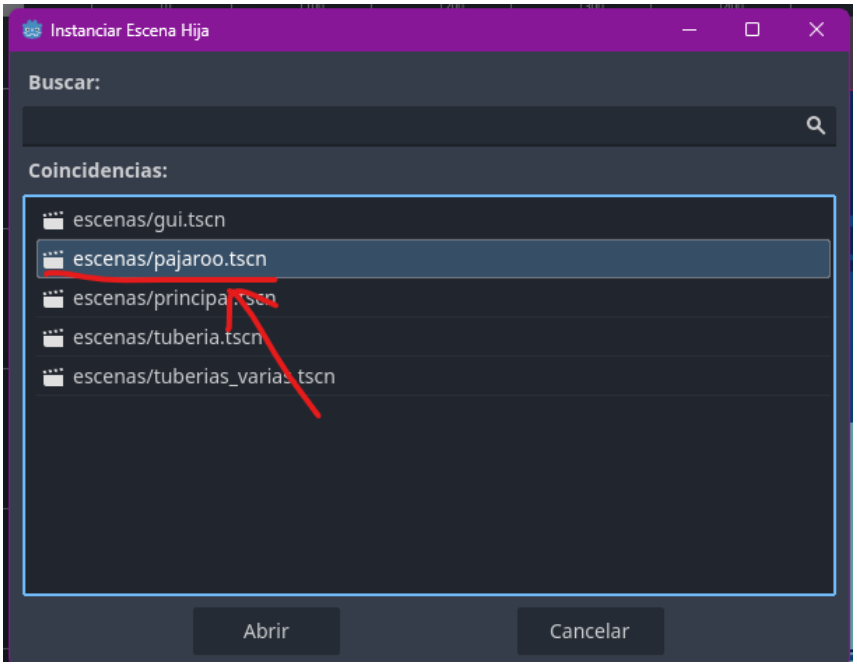
-“ui_accept” es el grupo donde está la tecla espacio que usaremos para saltar, aunque también e puesto manualmente el click izquierdo del ratón para que luego al exportar a Android que me coja los “Tap” con pulsar en la pantalla del móvil.

-Ahora modificaremos la gravedad del personaje para que caiga a una gravedad que pongamos, en mi caso lo he puesto a 1200 porque me parecía que así era algo más complejo al jugar, al final ese es el objetivo. Esto lo encontraremos en “Proyecto >> Configuración del Proyecto >> Físicas”:

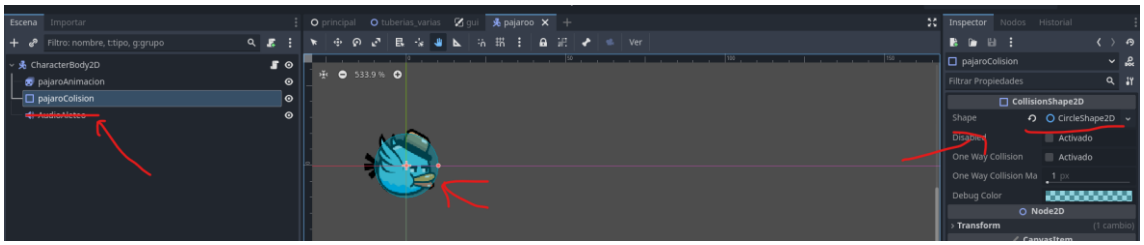


-Para llevar el pájaro a la ventana principal para que se pueda ejecutar todo y que el pájaro y sus ajustes estén ya en la ventana principal, instanciamos la escena hija y seleccionamos la escena correspondiente al pájaro:

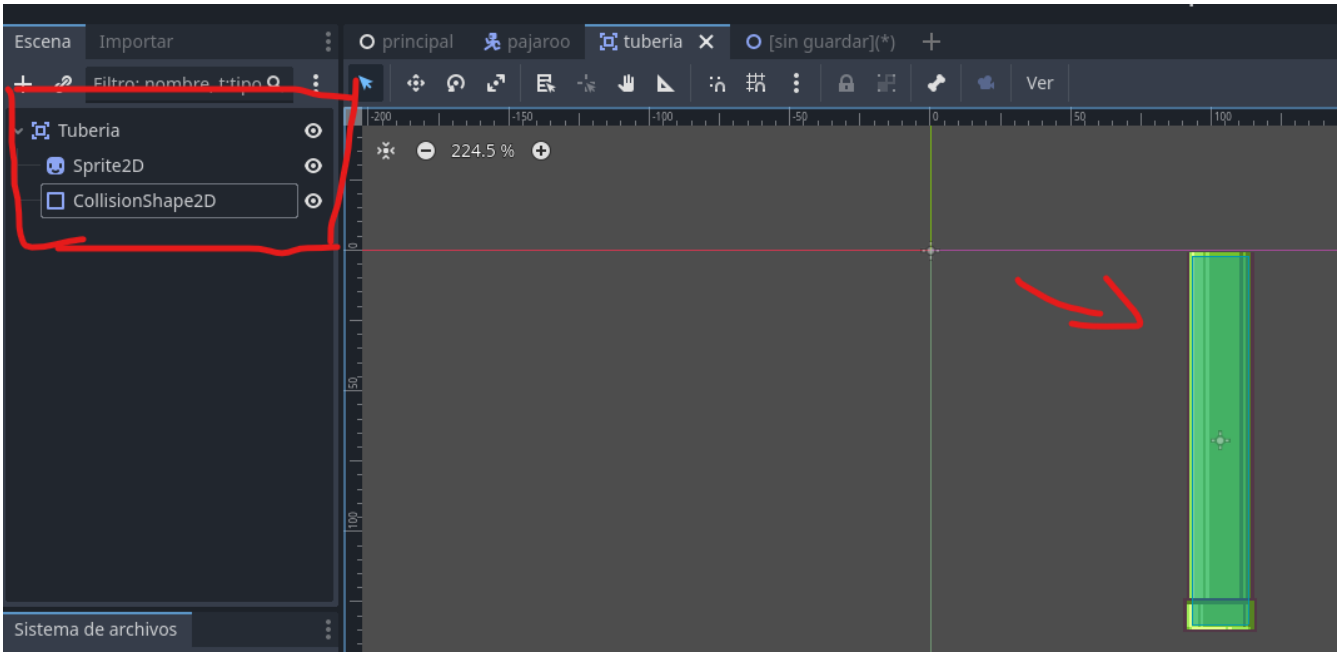




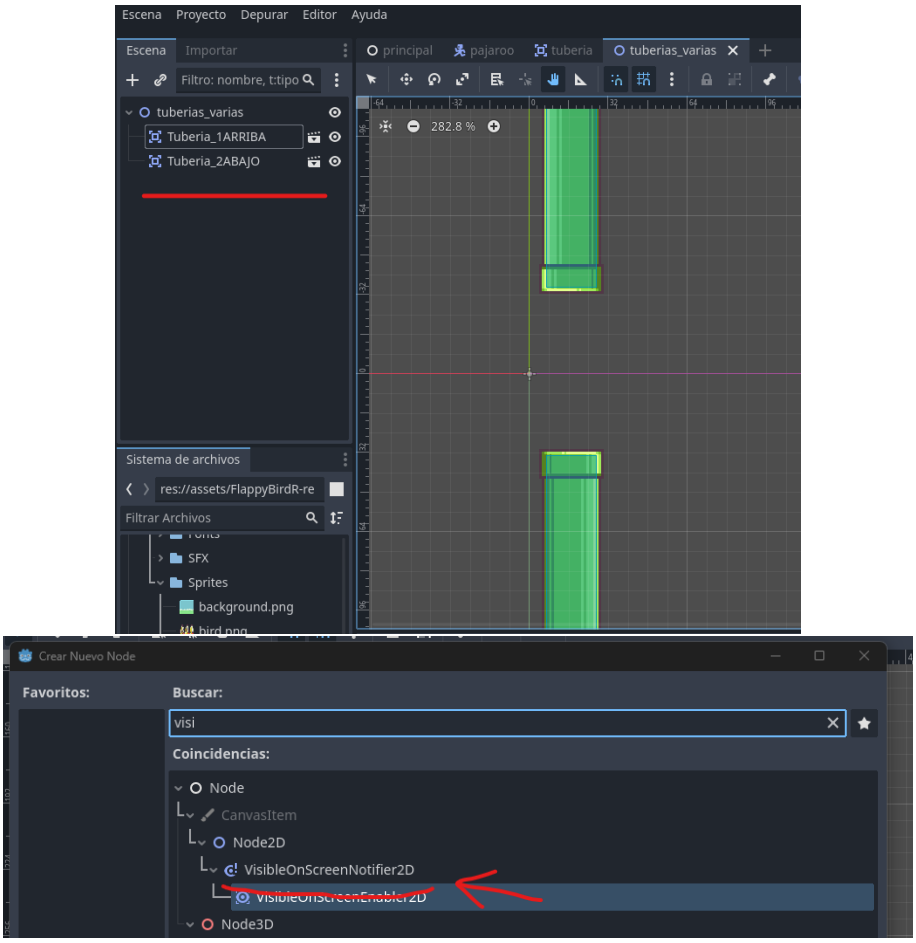
-Al personaje también le tendremos que poner un “CollisionShape2D” para que tenga física y que cuando choque con un obstáculo o con el suelo o por encima de los límites que se pueda morir, si no tiene física no se podrá morir por más que choque, y el juego no tendría sentido. Entonces, creamos un nuevo nodo hijo tipo “CollisionShape2D” y le damos la forma círculo por la forma del pájaro y lo colocamos encima del personaje:



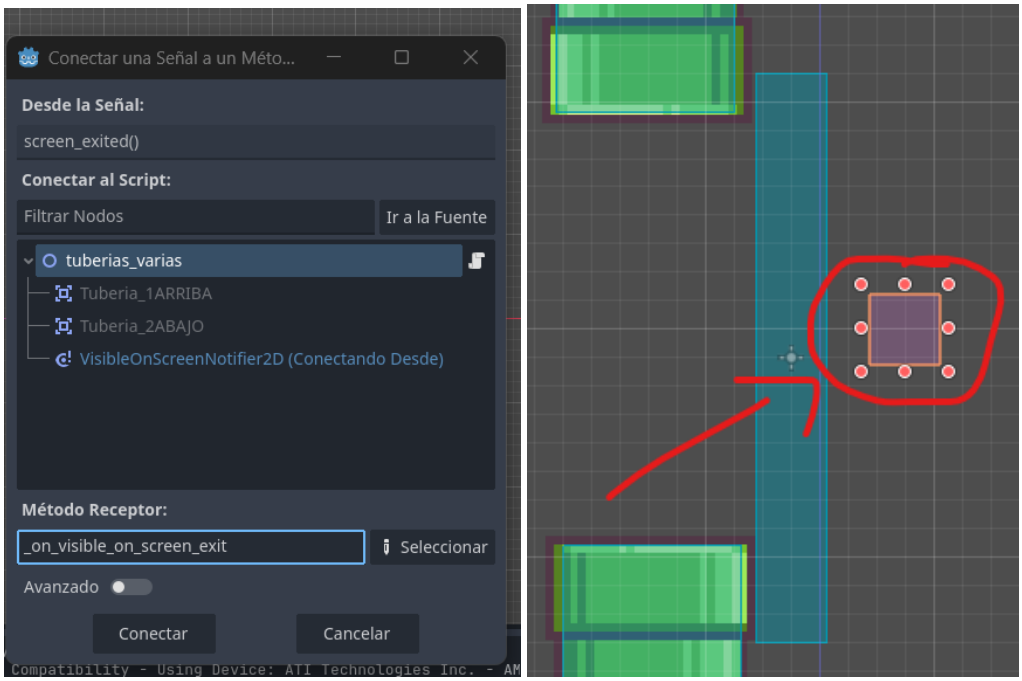
-Una vez tenemos ya el pájaro en la escena podremos poder ejecutarlo para jugar con él. Ahora creo una escena tubería en la que meteremos el asset correspondiente a la tubería que en el “Sprite2D”. Luego, también agregaremos el “CollisionShape2D” a la tubería ya que esto es lo que hará que la tubería tenga física y que se pueda chocar el personaje contra ella:



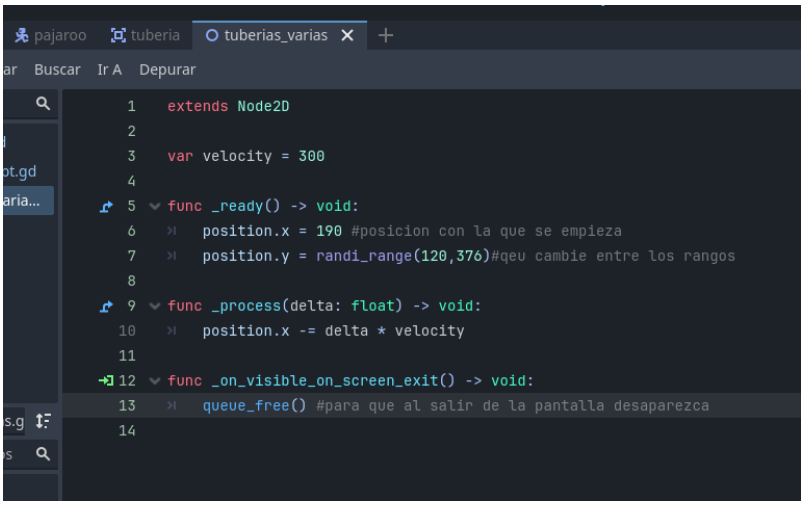
-Ahora, usamos la tubería anterior en una nueva escena donde pondremos una tubería boca arriba y otra tubería boca abajo y luego añadiremos un nodo hijo del tipo “VisibleOnScreenNotifier2D” que usaré para que se creen automáticamente las tuberías:



-El “VisibleOnScreenNotifier2D” tengo que conectarlo a las tuberías para poder hacer el método de la creación de las tuberías automáticamente:

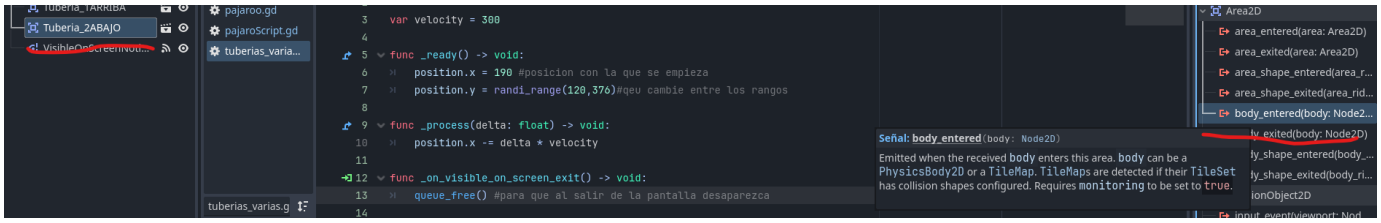


Y en el script que creamos para esta escena metemos lo siguiente, donde se indica la posición en la que salen las tuberías en el eje X, y los rangos en los que tiene que ir variando a la hora de crearse un nuevo conjunto de estas tuberías en el eje Y, entre 120 y 376:

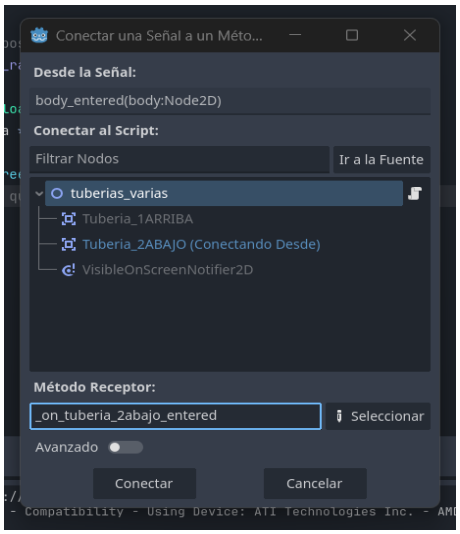


Y en la función que se creó al conectar el “VisibleOnScreenNotifier2D” con la escena ponemos el método “queue_free()” cuya función es hacer que una vez se ha pasado por las tuberías se desaparezcan estas.

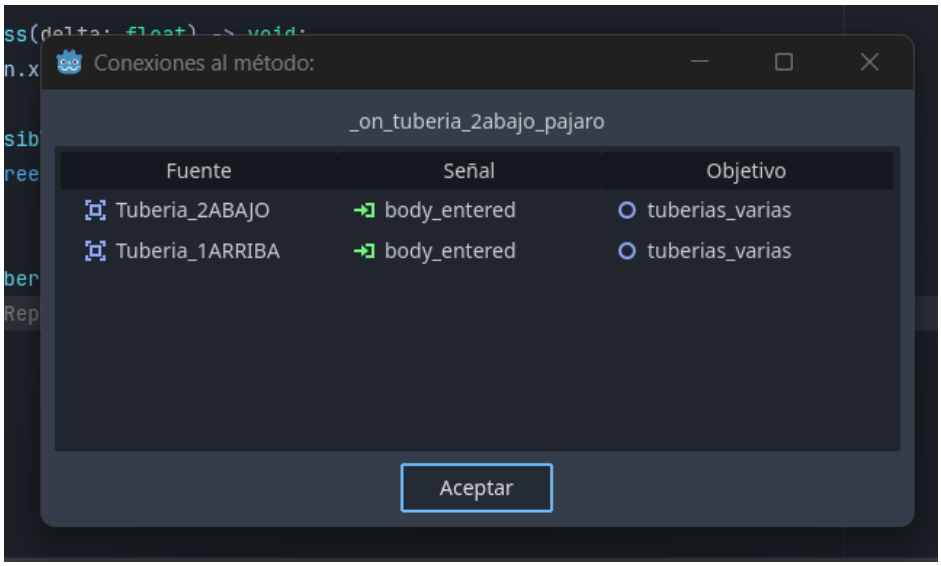
-Luego, para que las tuberías puedan lanzar un mensaje de “GameOver” primero tenemos que hacer que cuando el cuerpo del personaje choque con las tuberías, suelo o límites del cielo que el pájaro muera:



Entonces conectamos los nodos de ambas tuberías con la escena para que se genere el método:



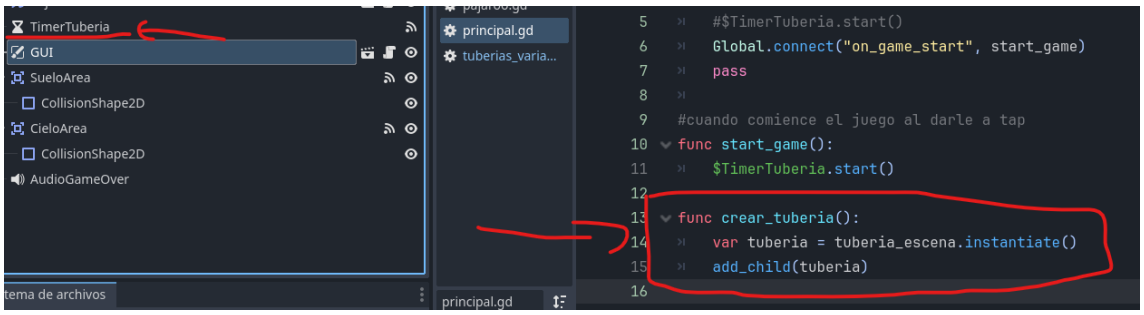
En la siguiente imagen se muestra como están conectadas las tuberías de arriba y la de abajo para que en caso de que el cuerpo del personaje colisione con ellos ocurra algo que pondremos después, esta es la función para que lo del “GameOver” funcione:



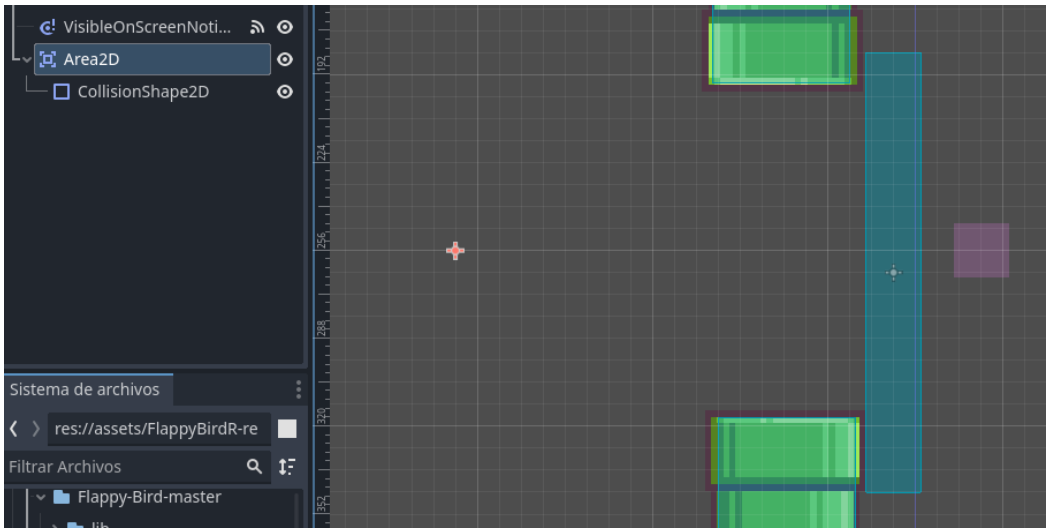
-Y ahora en el script ponemos el siguiente código que es provisional lo del mensaje, solo para comprobar que sí que funciona. Este método sirve para las tuberías tanto la de arriba como la de abajo:

```
14
15
16 func _on_tuberia_2abajo_pajaro(body: Node2D) -> void:
17     if body is Pajaro:          #cuando choque el pájaro muere
18         print("Murió el pájaro")
19
```

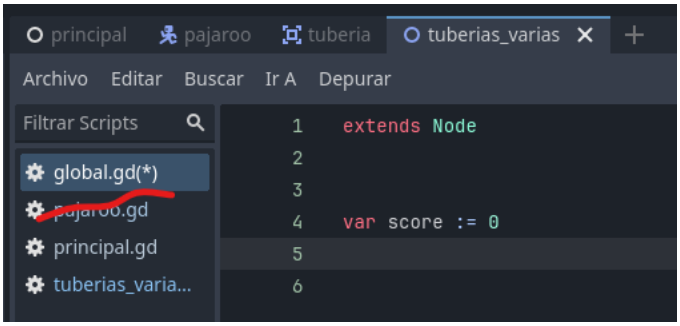
-Ahora, nos vamos a la escena principal y agrego las tuberías como instancia de hijo, pero en este caso por código en el script de la escena, además, agrego un “Timer” que irá creando las tuberías en los distintos rangos puestos anteriormente:



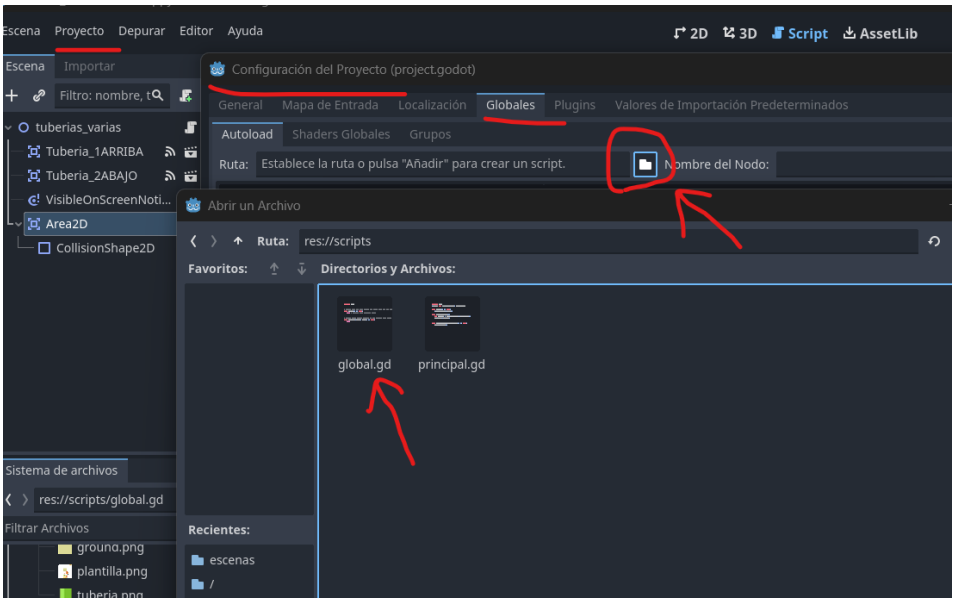
-A continuación, volveré a “tuberías_varias” y crearé un “Área2D” casi entre las tuberías para después aplicarle que cuando el personaje pase entre las tuberías, tocando el “Área2D” creada que se incremente el score que será la puntuación:



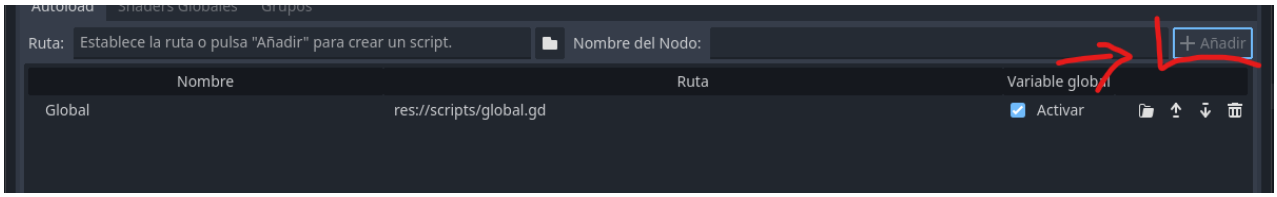
-Ahora creamos un script ‘global’ para la puntuación:



-Luego, nos vamos a “Proyecto >> Configuración de proyecto >> Globales”, pinchamos en la carpetita y seleccionamos el script que hemos creado anteriormente:



-Importante, debemos pulsar en “Añadir” que está a la derecha porque si lo seleccionamos y no añadimos después no estamos consiguiendo hacer nada:



-Una variable “score” en el script global y creamos una función que se llamará “increment_score” que como bien dice su nombre, incrementará la puntuación conforme se vayan consiguiendo pasar entre tuberías:

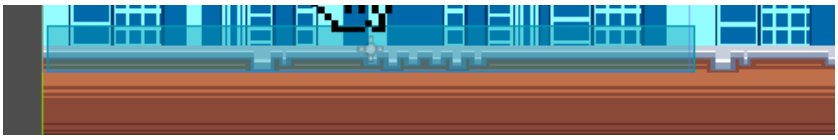
```
5
6  func increment_score():
7      score += 1
8
```

-Después, nos vamos al script “tuberías_varias” y en el llamaremos al método cuando el personaje pase el “Área2D” que habíamos creado anteriormente:

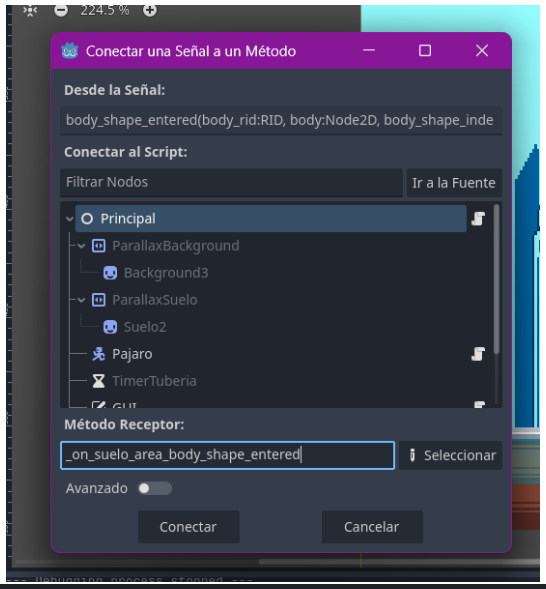
```
21  #método para que al pasar una tuberia se sume un punto
22  func _on_area_2d_body_entered(body: Node2D) -> void:
23      if body is Pajaro:
24          Global.increment_score()
```

-Antes de crear lo que viene siendo las etiquetas del juego (puntuación, menú, game over, etc), se me había olvidado comentar que también he creado una colisión en el suelo y otra en el cielo para que el pájaro no traspase ni el suelo ni se pueda salir de la pantalla por la parte de arriba. Al chocar con estos dos se produce “GameOver”, es decir, muere el jugador.

-Colocamos las dos colisiones una en el suelo y otra por la parte del cielo, ambas son “Área2D” y dentro de cada una de las áreas esta la “CollisionShape2D”:



-A continuación, conectamos el nodo para que cuando choque en el suelo se produzca el “GameOver”. Y repetimos la misma acción con la colisión del cielo:




```gdscript
#choca con el suelo y se para el juego
func _on_suelo_area_body_entered(body: Node2D) -> void:
    if body is Pajaro:
        Global.gameOver()
```

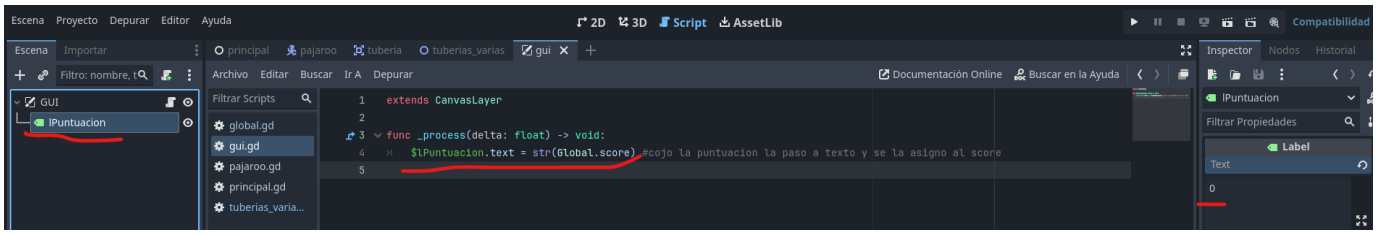
-Él código de cuando choque con el cielo sería similar al de chocar con el suelo:

```gdscript
##funcion para que no se vaya por arriba
func _on_cielo_area_body_entered(body: Node2D) -> void:
    if body is Pajaro:
        Global.gameOver()
        $TimerTuberia.stop()
        $AudioMuerte.play()
```

CREACIÓN DE MENÚ Y GAME OVER

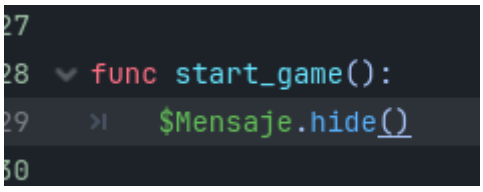
La fuente que escojo para todas las etiquetas que voy a crear es una fuente dada por el autor del video tutorial que seguí para conseguir realizar el juego. Está todo de assets en la bibliografía.

-En primer lugar, voy a crear un label para la puntuación del pájaro al pasar entre las tuberías. En primer lugar, creo un nuevo nodo de tipo “CanvasLayer” y dentro de él agrego el label para la puntuación. Luego lo llamo en el script a la variable “Score” que cree anteriormente y se coge el número y ya lo mostramos con el label:



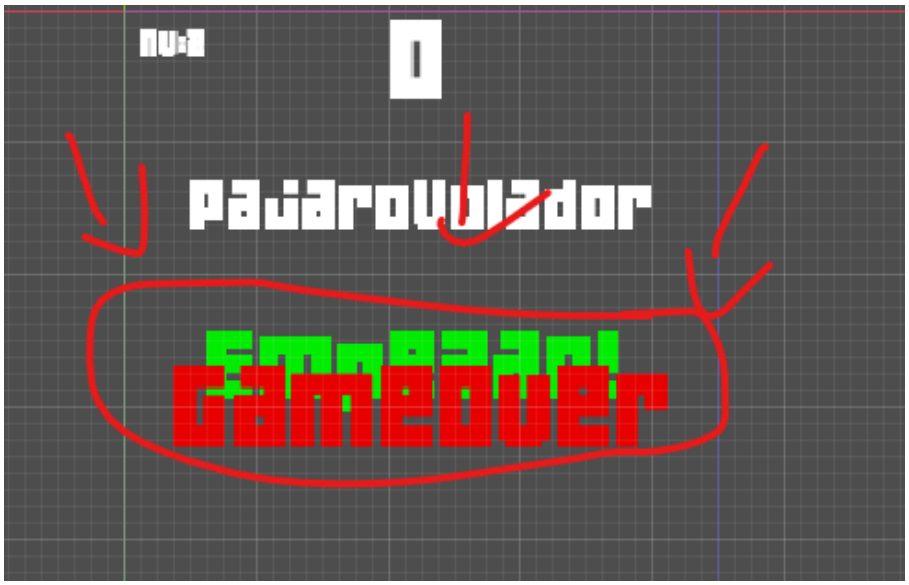
-El menú lo he creado agregando un “Sprite2D” y dentro de este nodo metiendo todos los label que voy a usar para el menú principal así una vez los centre los puedo mover todos a la vez, y a la hora de llamarlo en el script es más fácil y rápido:

-Ahora en el script la función de la siguiente imagen hace que cuando empiece el juego al pulsar en la pantalla que se despeje la pantalla y se muestre en ella solo el pájaro y las tuberías:

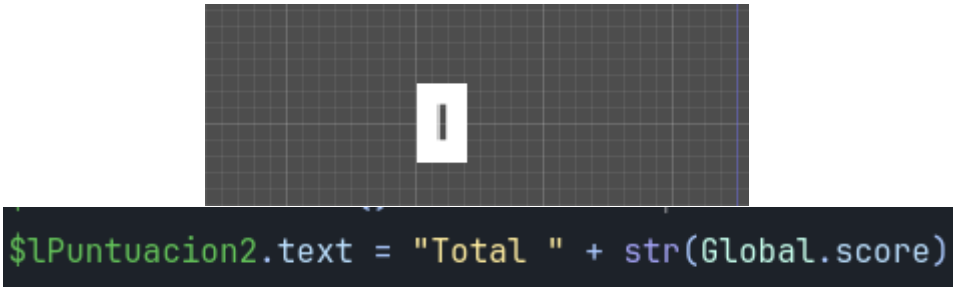


El “.hide()” que usamos en el script es un método que se usa para ocultar algo, en este caso, lo usamos para ocultar las etiquetas creadas anteriormente.

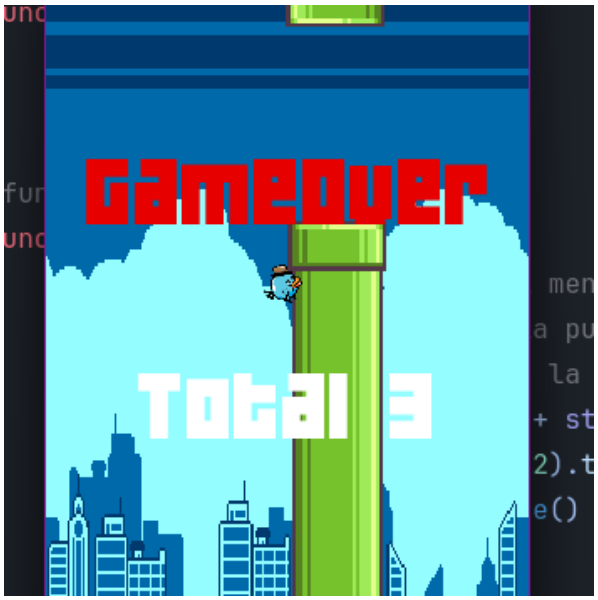
-Con el mensaje de “GameOver” lo hemos creado igual con una etiqueta en este caso de color rojo, lo muestro a continuación:



-Junto con el “GameOver” muestro también una etiqueta del total de la puntuación que hemos hecho, aparece solo el número porque en el script lo pongo como en la segunda imagen:



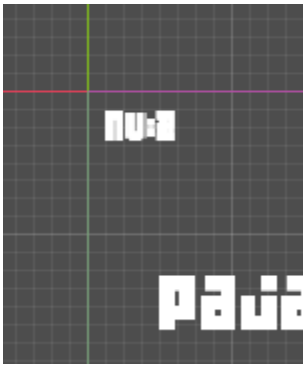
Y cuando terminamos una partida al final se veria de la siguiente manera:



-“GameOver” aparece cuando el pájaro choca con el suelo, el cielo o con las tuberías, y espera 2 segundos para volver al menú principal de nuevo con el “await” que uso y el “reload_current_scene” es para volver a cargar la pantalla de menú y poder comenzar de nuevo:

```
func game_over():
    $lGameOver.show() #muestro el mensaje de "GameOver"
    await get_tree().create_timer(2).timeout #tiempo de mostrar el mensaje
    get_tree().reload_current_scene() ##ejecuta toda la escena de nuevo
```

-Ahora pondré la etiqueta de los niveles, en mi caso, solo puse 3 porque soy incapaz de llegar al nivel 2 y mis compañeros que lo han jugado tampoco. Los pasos para crearlo lo he hecho igual creando 3 label y según la puntuación que se tenga sea un nivel u otro:



-Luego para que se muestre uno u otro en base a la puntuación que lleve el usuario uso el siguiente método(1ºImagen en el script del GUI y la 2º Imagen en el script GLOBAL):

```
func update_level_display():
    $lNv1.hide()
    $lNv2.hide()
    $lNv3.hide()
    match Global.nivel_actual:
        1:
            $lNv1.show()
        2:
            $lNv2.show()
        3:
            $lNv3.show()
```

```
func check_level_up():
    if score == 15 and nivel_actual == 1:
        nivel_actual += 1

    elif score == 30 and nivel_actual == 2:
        nivel_actual += 1
```

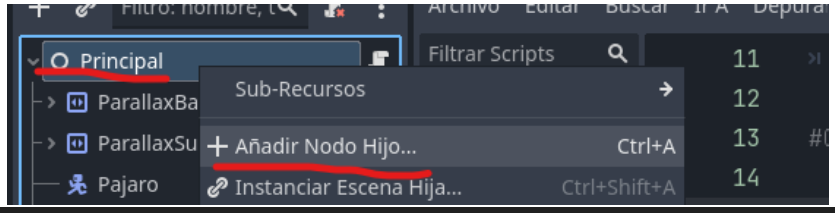
Pongo en principio los 3 niveles ocultos porque hasta que no se empieza a jugar el nivel está oculto para que estéticamente este mucho mejor, una vez se pulsa en pantalla o alguna tecla para empezar aparece el nivel.

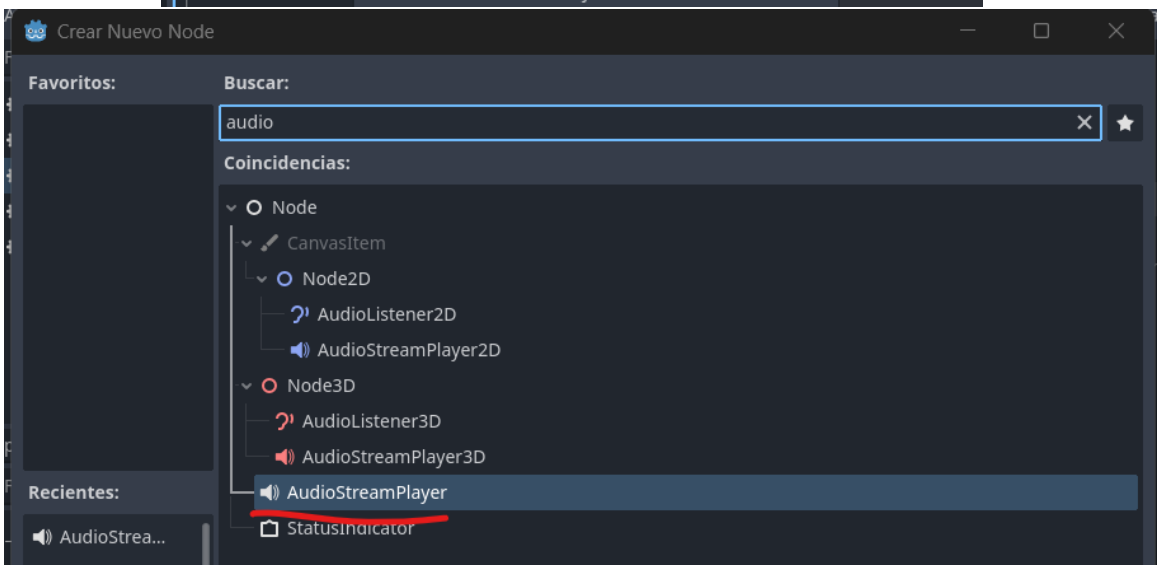
-Lo siguiente, es que quiero darle rotación al pájaro para cuando salte y caiga, es decir, si va hacia arriba que se incline un poco arriba y lo mismo si va para abajo. Lo hago con esta función que si se impulsa hacia arriba rota unos grados y en caso de que no pulses que caiga hacia abajo en otros grados de inclinación diferentes:

```
func rotacion_pajaro():
    if velocity.y >0 and rad_to_deg(rotation)<90:
        rotation += 2 * deg_to_rad(1)
    elif velocity.y > 0 and rad_to_deg(rotation)>-40:
        rotation -= 2 * deg_to_rad(1)
```

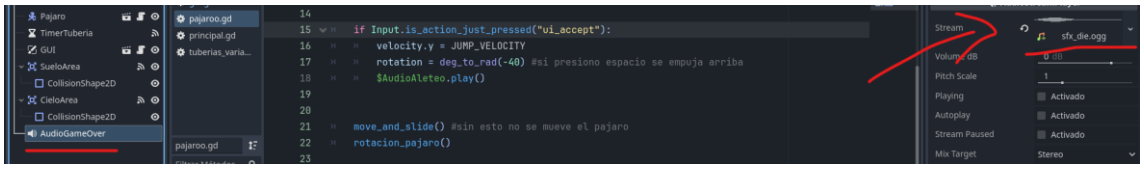
SONIDO

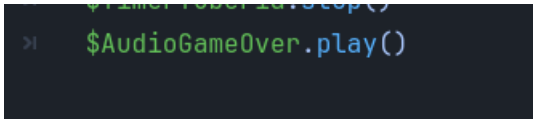
-Para crear el sonido de la muerte nos vamos a “Principal >> Añadir Nodo Hijo >> AudioStreamPlayer”:





-Luego seleccionamos el nodo añadido, y en “assets” o donde tengamos nuestros audios, los cogemos, arrastramos hasta donde se indica en la imagen siguiente. Y para implementarlo en el código lo llamamos como en la segunda imagen y en el script de la escena en el que queramos que se reproduzca:





-Y así repetiríamos el proceso con todos los sonidos que queramos agregar al juego, en mi caso, añadí uno para los puntos, otro para el aleteo del personaje, y por último, el mostrado en las imágenes para el “GameOver”.

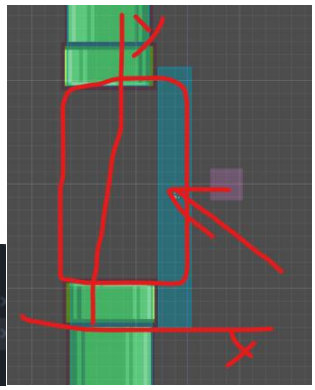
CÓDIGO

En este apartado, voy a explicar un poco las partes de los scripts que sean así un poco más complejas.

-Tuberias_varias:

Esta función es cuando está ejecutándose el juego, indica la posición donde aparecerán las tuberías en "**position.x**" y en "**position.y"** pongo el rango en el que se deben generar el espacio entre ellas, por el que pasa el personaje

```
func _ready() -> void:
    position.x = 190
    position.y = randi_range(120,370)
```



Esta otra es para mover las tuberías generadas en el frame hacia la izquierda para que el personaje pueda ir esquivando el chocar con las tuberías:

```
func _process(delta: float) -> void:
    position.x -= delta * velocity
```

La siguiente función se encarga de que cuando la tubería salga del frame se desaparezca, esto ocurre gracias al método "queue_free()" que es lo que se encarga de que ocurra:

```
func _on_visible_on_screen_exit() -> void:
    queue_free()
```

Ahora voy con la función de cuando el personaje choca con la tubería de abajo o arriba, que se produzca el "GameOver" y que además se produzca el sonido de "GameOver" que le he puesto:

```
func _on_tuberia_2abajo_pajaro(body: Node2D) -> void:
    if body is Pajaro:
        Global.gameOver()
        $AudioGameOver.play()
```

Por último, en este script tenemos la siguiente función en la que llama al script global la función de sumar puntos para que cuando se pase entre las tuberías se sume el punto y que se reproduzca el sonido del punto conseguido:

```
func _on_area_2d_body_entered(body: Node2D) -> void:
    if body is Pajaro:
        Global.increment_score()
        $AudioPunto.play()
```

-***Pájaro:***

La función siguiente comienza con una función global de que si el juego ya ha empezado y el pájaro no está tocando el suelo que se le aplique la gravedad. Luego, en el segundo "if" lo que hace es que si se pulsa la tecla espacio se produce la acción de saltar del personaje de ahí que ponga "**velocity.y**" = JUMP_VELOCITY. Además, le agrego inclinación para cuando salte el pájaro que se incline y el sonido del aleteo al saltar. Por último, los dos métodos de la segunda imagen, el primero es para que el pájaro se mueva, si no se pone ese método el pájaro no hará nada, y el segundo método es para que se aplique correctamente la rotación del pájaro:

```
#CONFIGURACION GRAVEDAD
func _physics_process(delta: float) -> void:
    if Global.is_start:
        if not is_on_floor():
            velocity += get_gravity() * delta

#CONFIGURACION DEL SALTO CON EL ESPACIO
    if Input.is_action_just_pressed("ui_accept"):
        velocity.y = JUMP_VELOCITY
        rotation = deg_to_rad(-40) #si presiono espacio se empuja arriba
        $AudioAleteo.play()
```

```
move_and_slide()
rotacion_pajaro()
```

-Ahora la función siguiente es la que se usa en la función anterior sobre la rotación del personaje. Sí la inclinación del pájaro es menor de 90 grados si añaden 2 grados de inclinación para que se incline hacia abajo y si es mayor de -40 se le quitan 2 grados para que se incline para arriba:

```
func rotacion_pajaro():
    if velocity.y >0 and rad_to_deg(rotation)<90:
        rotation += 2 * deg_to_rad(1)
    elif velocity.y > 0 and rad_to_deg(rotation)>-40:
        rotation -= 2 * deg_to_rad(1)
```

-Gui:

En primer lugar, tenemos la siguiente función con los métodos “hide()” que se usa para esconder algo en pantalla en algún momento determinado(depende de cuando se aplique). Es una función básica en la que oculto los 3 niveles del juego y es simplemente para aplicarlo en la pantalla de menú ya que no me parecía bien que apareciese sin empezar aún el juego:

```
func esconderNivel():
    $lNv1.hide()
    $lNv2.hide()
    $lNv3.hide()
```

Después, tenemos la siguiente función que consiste en que si no pulsamos uno de los botones pertenecientes al grupo de teclas “ui_accept”(espacio, click izq., etc) no se empieza el juego:

```
func _input(event: InputEvent) -> void:
    if Input.is_action_just_pressed("ui_accept"):
        if not Global.is_start : ##==false
            Global.start_game()
```

Ahora, tenemos una función para las etiquetas de nivel, en principio están ocultos y llamamos a la función del script global para que compruebe la puntuación del jugador y según esta, se muestra uno u otro nivel por pantalla:

```
func update_level_display():
    $lNv1.hide()
    $lNv2.hide()
    $lNv3.hide()
    match Global.nivel_actual:
        1:
            $lNv1.show()
        2:
            $lNv2.show()
        3:
            $lNv3.show()
```

La siguiente función se encarga de cambiar constantemente el label hecho para la puntuación que aparece arriba de la pantalla y centrado. Coge una función del script global que va incrementando la puntuación según se va consiguiendo esquivar tuberías y este las pasa a texto y lo muestra:

```
func change_score():
    $lPuntuacion.text = str(Global.score)
    update_level_display()
```

Y ahora tenemos una de cuando empieza el juego, y aquí consiste en ocultar el menú del principio y de llamar al "update_level_display()" para que comience dicha función a trabajar también:

```
func start_game():
    $Mensaje.hide()
    update_level_display()
```

"game_over()", esta función lo realmente importante es el método await que espera 2 segundos después de haber muerto y se vuelve al menú para volver a jugar de nuevo usando el método de debajo "get_tree().reload_current_scene()". Los demás label en el caso de los que tienen método "show()" es para mostrar el label al que están aplicados cuando acaba el juego y los que tienen el método "hide()" es para ocultarlos cuando se produzca un game over:

```
func game_over():
    $lGameOver.show() # Muestra el mensaje de "GameOver"
    $lPuntuacion.hide() # Oculta la puntuación actual
    $lPuntuacion2.show() # Muestra la puntuación total
    $lPuntuacion2.text = "Total " + str(Global.score) # Ac
    await get_tree().create_timer(2).timeout # Tiempo para
    get_tree().reload_current_scene() # Reinicia la escena
```

Por último, la siguiente función lo que hace es escuchar las señales que emite el script global para ver si debe cambiar la puntuación, empezar o acabar el juego. Luego los label con función ya sabemos lo que hace y al final, el "get_tree().paused=false" es para decirle al juego que esté listo para comenzar de nuevo después:

```
func _ready() -> void:
    Global.connect("on_increment_score", change_score)
    Global.connect("on_game_over", game_over)
    Global.connect("on_game_start",start_game)
    $lGameOver.hide()
    $Mensaje.show()
    esconderNivel()
    $lPuntuacion2.hide()
    get_tree().paused = false  #para comenzar de nuevo
```

-Principal

En el script de principal es sobre la escena donde se ejecuta el juego completo, en primer lugar, tenemos la función de cuando se ejecutar que escucha una señal emitida desde el script global de que ha comenzado el juego:

```
func _ready() -> void:
    #$TimerTuberia.start()
    Global.connect("on_game_start", start_game)
    pass
```

En segundo lugar, una función básica que se basa en iniciar el timer de las tuberías para que estas se empiecen a crear en la escena hasta que se produzca algo que las detenga, en el caso de este juego, un “game over”:

```
func start_game():
    $TimerTuberia.start()
```

Esta función en concreto su función es poner las tuberías creadas anteriormente como obstáculo en la escena principal. Lo comenté anteriormente se puede instanciar un nodo hijo de forma manual y por script, esta es la forma por script. Resumidamente, es una instancia de la escena de las tuberías para llamarlas en esta escena principal:

```
func crear_tuberia():
    var tuberia = tuberia_escena.instantiate()
    add_child(tuberia)
```

La siguiente va de la mano con la anterior función, ya que esta crea constantemente las tuberías que han sido instanciadas anteriormente en esta escena:

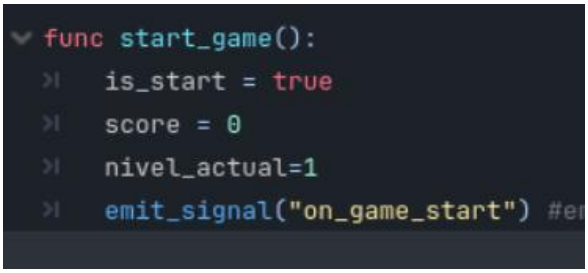
```
func _on_timer_tuberia_timeout() -> void:
    crear_tuberia()
```

Por último en esta escena, tenemos las siguientes funciones que las dos son iguales. En ellas se llama a la función “gameover()” del script global para que esta se ejecute y se para la creación de nuevas tuberías y junto con todo ello se reproduce el sonido del “gameover”. Esto se produce cuando el personaje entra en contacto con el suelo o el cielo encima de los límites:

```
#choca con el suelo y se para el juego
func _on_suelo_area_body_entered(body: Node2D) -> void:
    if body is Pajaro:
        Global.gameOver()
        $TimerTuberia.stop() #para que no se creen mas t
        $AudioGameOver.play()

#funcion para que no se vaya por arriba
func _on_cielo_area_body_entered(body: Node2D) -> void:
    if body is Pajaro:
        Global.gameOver()
        $TimerTuberia.stop()
        $AudioGameOver.play()
```

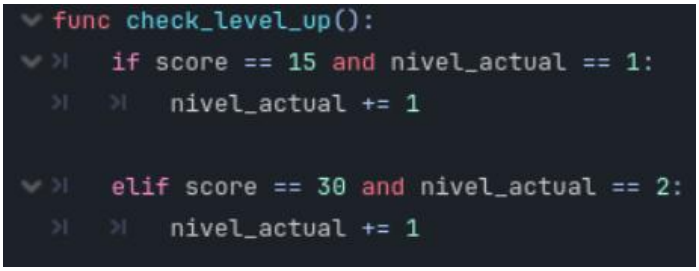
-Global

Vamos con el último script, en la imagen podemos observar la función que hemos llamado en anteriores scripts de que empiece el juego. En esta función tenemos el empezar en true, iniciamos la puntuación en 0, el nivel por el que empieza el nivel 1 y por último se emite la señal de que el juego empieza para que pueda ser escuchado en otros scripts:

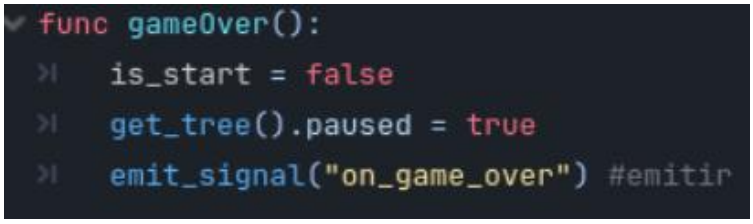


Ahora vamos con la función de incrementar la puntuación conforme se vaya consiguiendo puntuación, va sumando el contador uno a uno, añado la función de checkear el nivel en el que está y se emite la señal del incremento de la puntuación:

A continuación, tenemos la función que se ha llamado en el trozo de código anterior de checkear el nivel en base a la puntuación conseguida, si pasas 15 tuberías se establece el nivel 2 y si pasas las 30 se establece el nivel 3(dudo que alguien llegue a tanto por ello no puse más niveles):

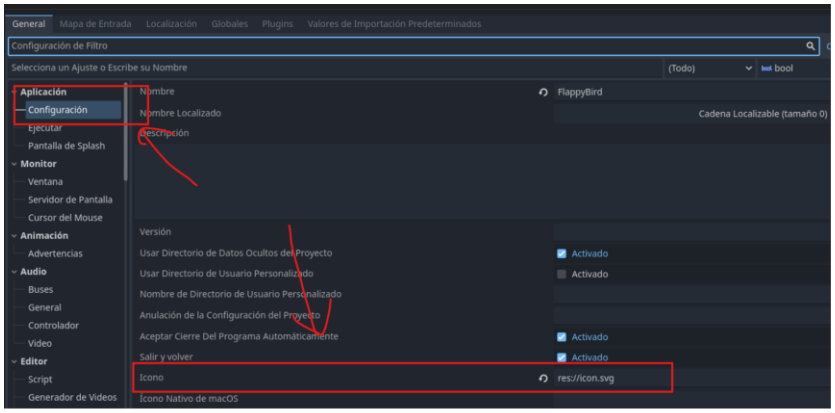


Por último, tenemos la función del game over en la que indicamos en la primera línea que el juego no está ejecución colocándolo como ‘false’, luego él “.paused” es para detener todo por ello se pone en ‘true’ y por último, se emite la señal de que se produjo el “game over”:

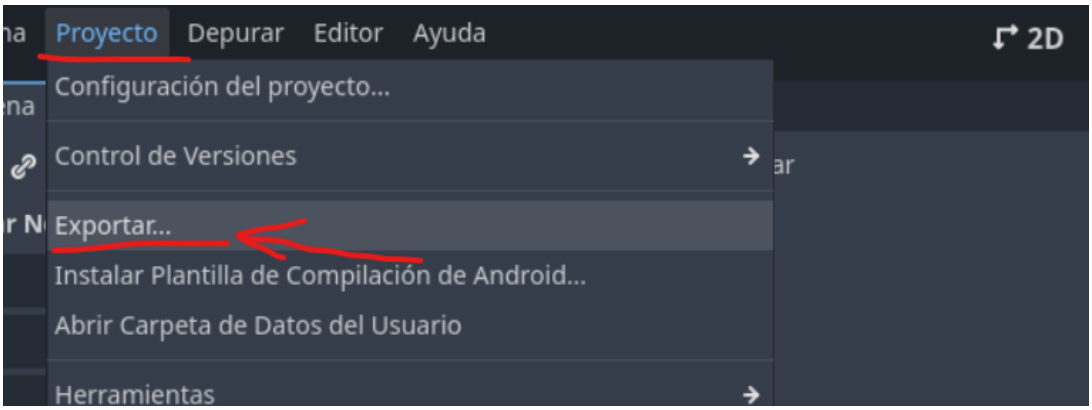


AÑADIR ICONO Y EXPORTAR

-Para añadir el icono al juego seleccionamos la imagen y la ubicamos en alguna carpeta que sea fácil de encontrar, luego nos vamos dentro de godot a “Proyecto >> Configuración de Proyecto >> Aplicación >> Configuración” ahí veremos un apartado denominado “Icono” y seleccionamos el icono que queremos usar:

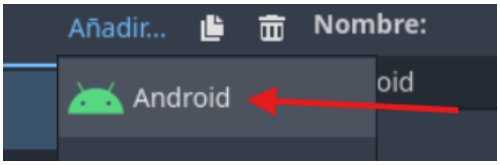


Ya estaría el icono puesto en el proyecto, ahora exportaremos yéndonos a “Proyecto >> Exportar”

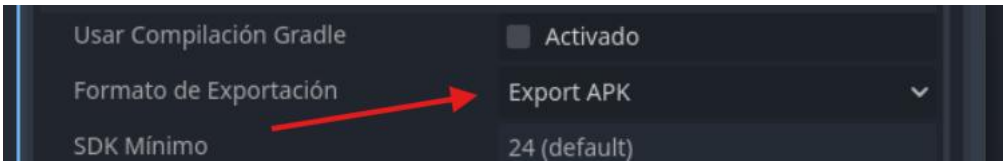


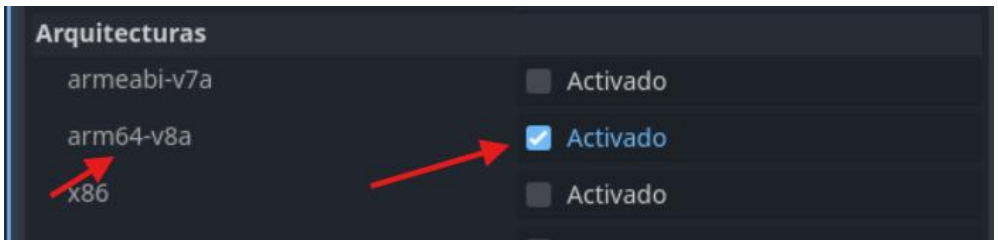
Luego pulsamos en “Añadir”:

Seleccionamos Android, ya que lo exportaremos para Android:

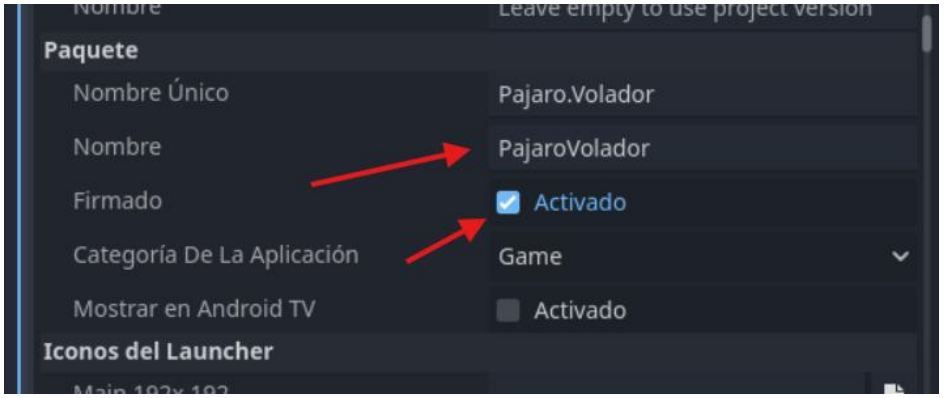


Importante en las opciones tener lo siguiente como muestro en las imágenes para que la exportación se haga correctamente:

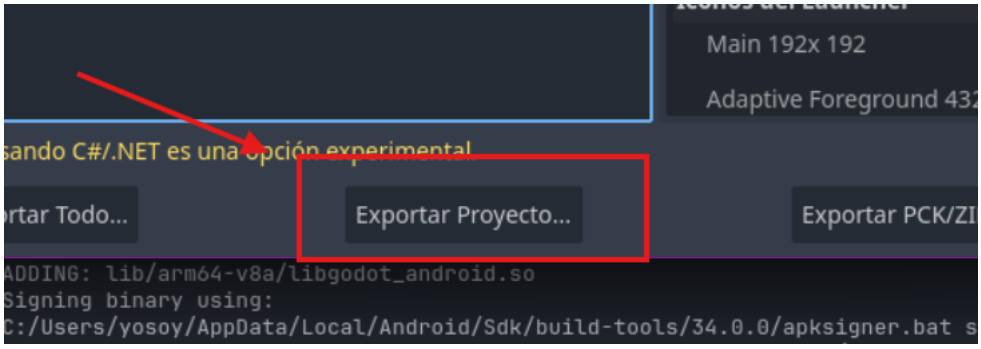




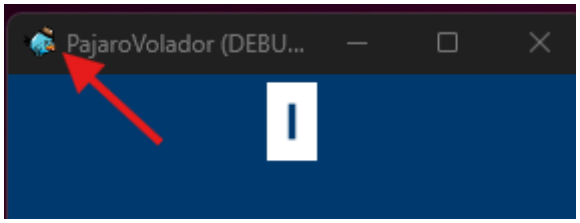
El cambio del nombre es opcional, yo en mi caso si lo he cambiado y activar “Firmado”:



Por último, exportamos el proyecto y se nos guardara donde nosotros queramos:



Y finalmente al abrir el juego podemos observar que el icono ha cambiado al que he puesto, eso sería todo para la exportación de nuestro juego:



CAMBIOS ADICIONALES

He añadido al final, un “high score”(máxima puntuación) que se muestre en el menú guardando para cada usuario su máxima puntuación y motivándolo así a superarse a sí mismo cada vez que juegue:

Esta función se encarga de guardar la puntuación más alta.

```
# Funciones
func save_high_score():
    # Guarda el high score en un archivo para persistir entre sesiones de juego.
    var file = FileAccess.open(score_file_path, FileAccess.WRITE)
    if file != null:
        file.store_var(high_score)
        file.close()

```

El siguiente método es para cargar la puntuación guardada:

```

func load_high_score():
    # Cargar el high score
    if FileAccess.file_exists(score_file_path):
        var file = FileAccess.open(score_file_path, FileAccess.READ)
        if file != null:
            high_score = file.get_var()
            file.close()

```

Y la última es para actualizar la puntuación:

```
func update_high_score(new_score: int):
    # actualizar el scroe maximo
    if new_score > high_score:
        high_score = new_score
        save_high_score()

```

También incremento la velocidad del juego conforme se va ejecutando el juego que vaya aumentando:

Declaro la velocidad que se va a ir sumando(15) luego el tiempo que tarda en incrementarse esta velocidad y el elapsed_time es para ir acumulando el tiempo que se ejecuta:

```
var speed_increase = 15.0
var time_to_increase = 2
var elapsed_time = 0.0
```

Aquí es donde se incrementa realmente se incrementa la velocidad, llamando a “Global” y el método de incrementar la velocidad:

```
    elapsed_time += delta
    if elapsed_time >= time_to_increase and Global.is_start:
        Global.increase_tube_velocity(speed_increase)  # Aumenta la velocidad de las tuberías.
        elapsed_time = 0.0
        print("Velocidad actual de las tuberías: ", Global.tube_velocity)
    position.x -= delta * Global.tube_velocity  # Mueve la tubería hacia la izquierda.
```

Bibliografía

- FlappyBirdR-resources
- *Flappy Bird Assets - Pixel Art by MegaCrash*. (s. f.). itch.io. https://megacrash.itch.io/flappy-bird-assets
- paulkr/Flappy-Bird: 🕹 A remake of the arcade game Flappy Bird, written in Java.
- https://clipart-library.com/free/mouse-cursor-transparent.html
- Eric. (2024, 4 agosto). Flappy Bird, Flying, Challenge, Jump, Score PNG. *PNG Mart*. https://www.pngmart.com/image/178577
- findemor. (2024, 24 abril). *Cómo usar Godot y Aprender desde CERO a hacer juegos* [Vídeo]. YouTube. https://www.youtube.com/watch?v=-_LiMyZGoXw
- CREA TU JUEGO. (2023, 17 septiembre). *Como Crear Flappy Bird en Godot 🐤 | Curso Completo de Flappy Bird para Godot 3* [Vídeo]. YouTube. https://www.youtube.com/watch?v=-BVNBuyvTNY
- BluuGames. (2024, 16 octubre). *Godot 4.3: JUEGO Flappy Bird (Tutorial 2024)* [Vídeo]. YouTube. https://www.youtube.com/watch?v=NDERxitqgSs