Modulo: PROYECTO

	Alumno: Rubén Esquivel Morón
	Curso: 2022/2023
	Ciclo: Grado Superior
	Familia Profesional: Administración De Sistemas Informáticos en Red
Responsable del seguimiento	José Luis Campos Reina
Título del Proyecto	UpLands
Descripción del proyecto	Videojuego de plataformas 2D creado desde cero en el programa Godot Engine

Contenido

1. Introducción al proyecto

Este proyecto, llamado “UpLands” es un videojuego creado por mi a nivel de programación a través de la aplicación “Godot” donde he programado con el lenguaje de programación Python.

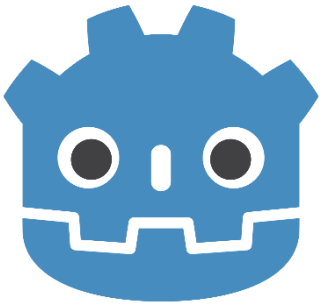
Este juego tiene como peculiaridad, el jugar a través de plataformas para conseguir un único objetivo, subir, de ahí su nombre

2. Origen y contextualización del proyecto

Este proyecto se ha creado para ampliar los conocimientos a nivel personal en cuanto a la programación, diseño, y creación de videojuegos, haciendo de ella, algo visual y accesible para todos los públicos como lo es un videojuego

3. Hablemos de Godot

Godot es un motor de juegos 2D y 3D de uso general diseñado para admitir todo tipo de proyectos. Puede usarlo para crear juegos o aplicaciones que luego puede lanzar en computadoras de escritorio o dispositivos móviles, así como en la web.



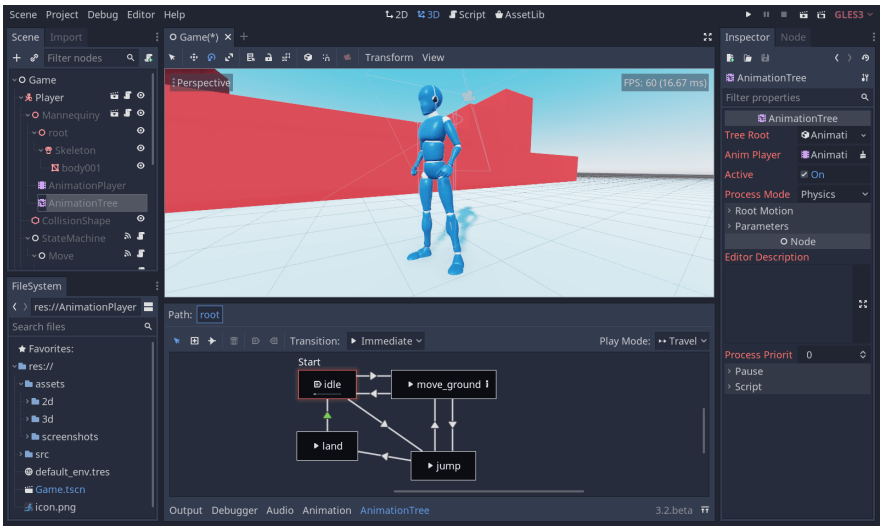
También puede crear juegos de consola con él, aunque necesita fuertes habilidades de programación o un desarrollador para portar el juego por usted.

Godot fue desarrollado inicialmente por un estudio de juegos argentino. Su desarrollo comenzó en 2001, y el motor se reescribió y mejoró enormemente desde su lanzamiento de código abierto en 2014.

Algunos ejemplos de juegos creados con Godot incluyen Helms of Fury.

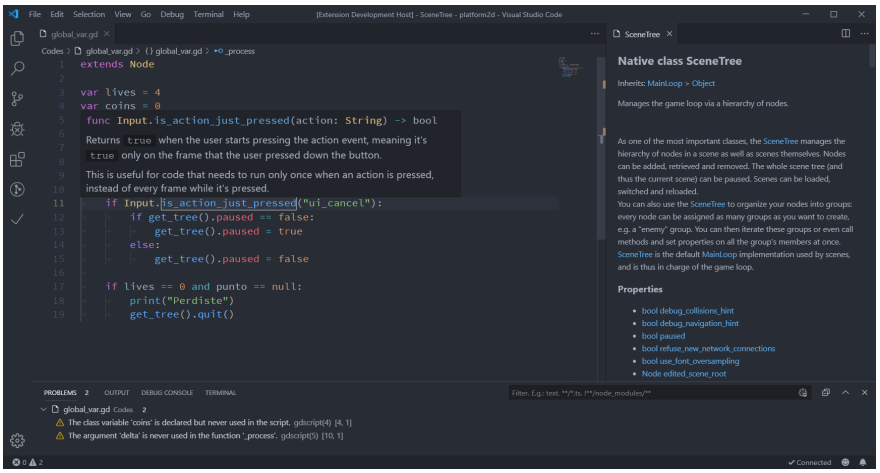
3.1 Como funciona Godot?

Godot integra un editor de juego completo con las herramientas para satisfacer las necesidades más comunes. Incluye un editor de código, de animaciones, de mapas de mosaicos, de sombras, un depurador, un perfilador y más.



El equipo se esfuerza por ofrecer un editor de juegos rico en funciones con una experiencia de usuario consistente. Si bien siempre hay margen de mejora, la interfaz de usuario se sigue perfeccionando.

Por supuesto, si lo prefiere, puede trabajar con programas externos. Apoyamos oficialmente la importación de escenas 3D diseñadas en Blender y mantenemos complementos para codificar en VSCode y Emacs para GDScript y C #. También admitimos Visual Studio para C # en Windows.

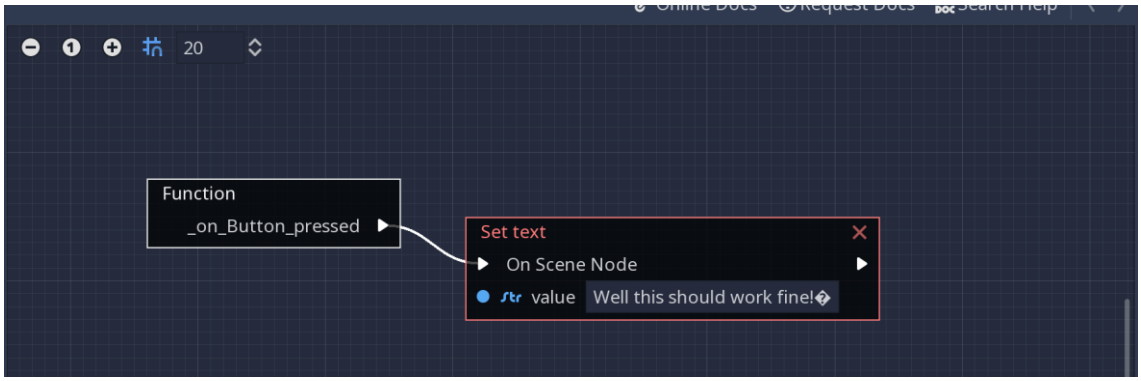


3.2 Lenguajes de programación

Hablemos sobre los lenguajes de programación disponibles.

Puede codificar sus juegos usando “GDScript”, un lenguaje especifico de Godot, el cual, esta integrado con una sintaxis ligera de C# y Python, los cuales, son lenguajes populares en la industria del videojuego

Godot también soporta un lenguaje de programación visual basado en nodos llamado “VisualScript” (hasta la versión 3.5 de godot) el cual se basa en una programación mas visual donde vas conectando nodos para “generar” el codigo



3.3 Godot 3.5 vs Godot 4

Hasta hace relativamente poco, Godot se actualizó a “Godot 4” donde actualizaban muchas zonas en la interfaz y códigos en cuanto a la programación

Mi proyecto ha sido realizado en Godot 4, donde he tenido que adaptar muchos tutoriales y cursos a esta versión, ya que la gran mayoría de información que hay, es de Godot 3.5

4. Aprendizaje del programa

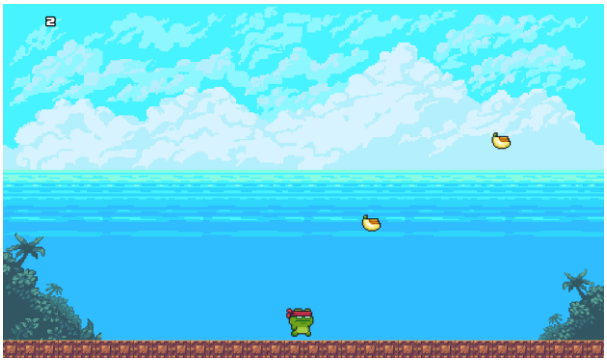
En lo personal, he ido aprendiendo a usar este programa gracias a varios cursos realizados en la plataforma “Udemy” donde se realizan varios ejemplos de juegos básicos donde se aprende tanto al uso del programa, como a su programación y características de Godot.

En estos cursos realizados, podemos encontrar resultados los cuales podéis ver en estas imágenes, donde se realizan varios juegos “base” para tener unas nociones mínimas de todo lo que se puede llegar a hacer con este programa.

En el primero está basado en el clásico juego “Alleyway” donde se va programando todo por capas y teniendo el siguiente resultado

En el segundo, podemos ver un juego en el cual debemos recoger unos plátanos que aparecen aleatoriamente mientras nos vamos moviendo por la plataforma principal.

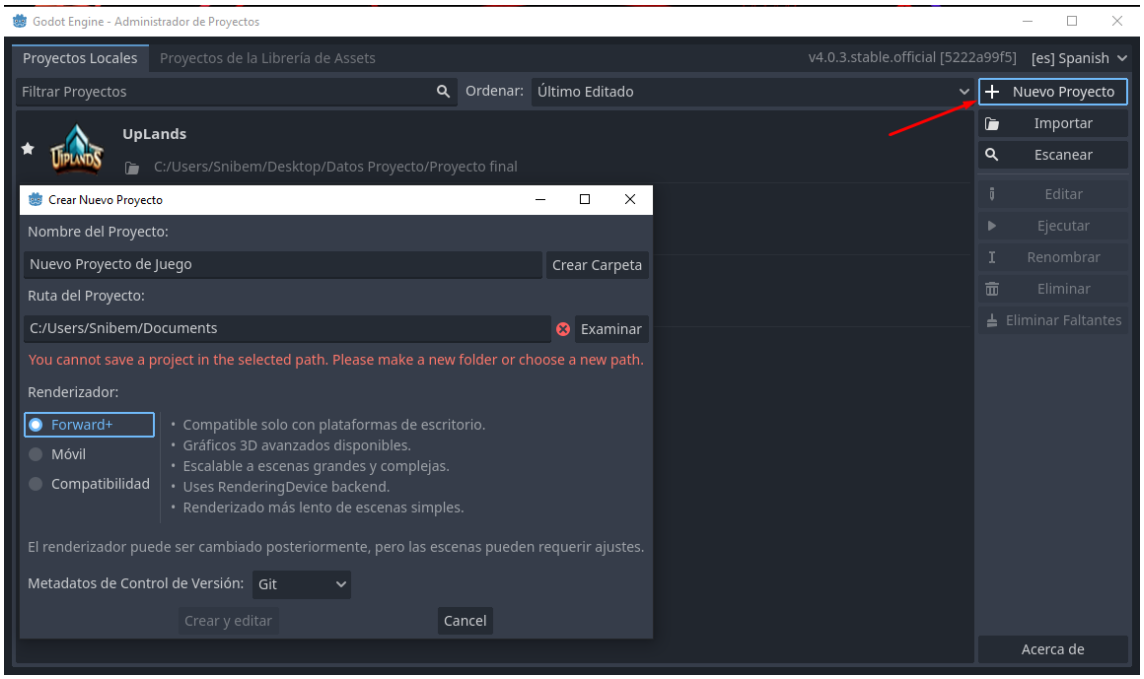
Todo esto, con su programación por detrás, su “diseño de niveles” y la inmensa cantidad de posibilidades que tenemos con Godot



También he aprendido mucho a través de YouTube, donde podremos encontrar muchos tutoriales para crear mecánicas o utilidades en los juegos que se creen

5. Interfaz del programa

Comenzaremos viendo la interfaz de Godot, donde al iniciar el programa, se nos presentará una ventana, donde nos preguntará la ubicación del proyecto, y los tipos de rederizadores, los cuales, se asemejan a los requisitos que necesitemos en nuestro juego



Donde podemos ver las siguientes opciones:

Forward+

- Compatible solo con plataformas de escritorio
- Gráficos 3D avanzados disponibles
- Escalable a escenas grandes y complejas
- Renderizado mas lento de escenas simples

Móvil

- Compatible con plataformas de escritorio y móviles
- Gráficos 3D menos avanzados
- Menos escalable para escenas complejas
- Renderizado rápido de escenas simples

Compatibilidad

- Compatible con plataformas de escritorio, móviles y web
- Gráficos 3D mucho menos avanzados (aun trabajando en ello)
- Destinado para dispositivos de gama baja/antiguos
- Renderizado más rápido de escenas simples

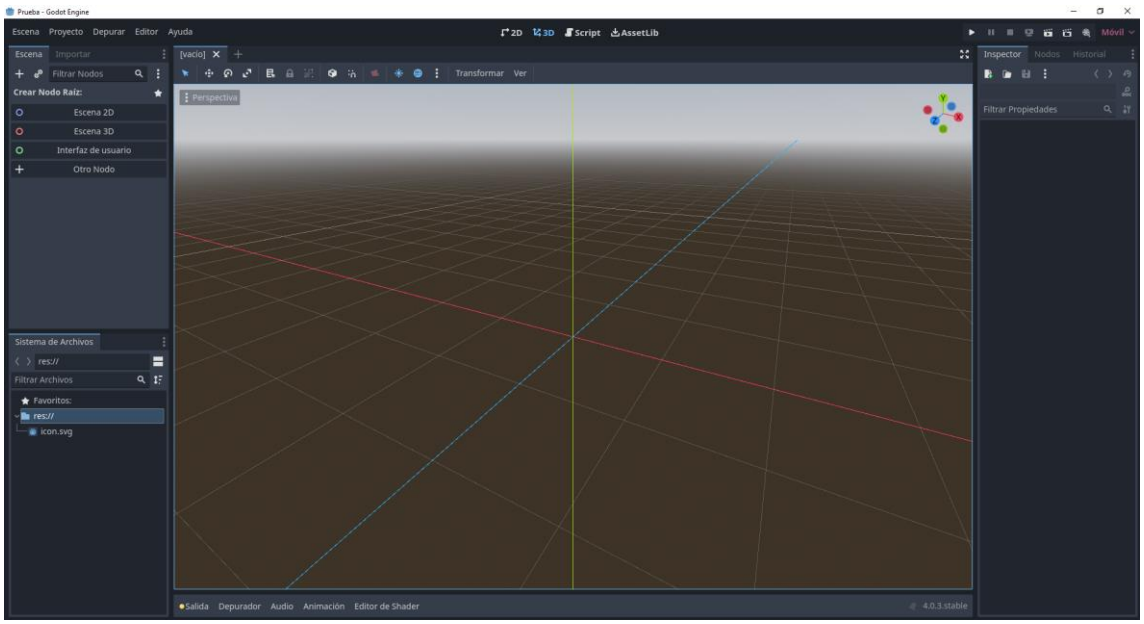
Al seleccionar el que mas nos convenga, podremos visualizar la interfaz principal del programa, donde lo dividiremos en varias zonas

Lo primero que salta a la vista, es la zona central, donde podemos visualizar el editor 2D/3D, el cual podremos utilizar a libre albedrio

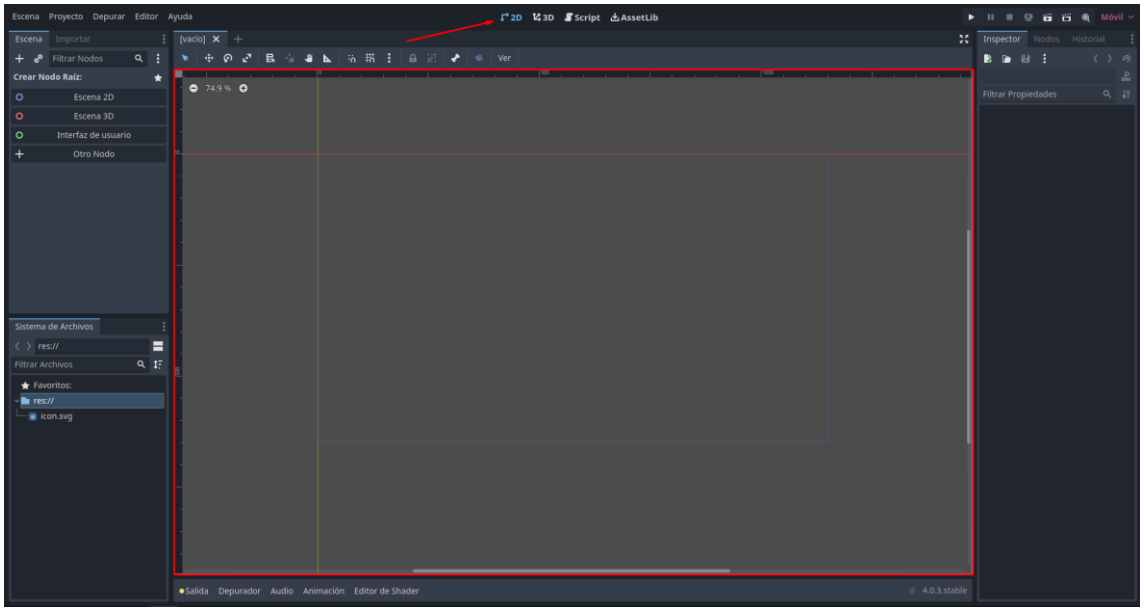
En la zona superior, podremos cambiar entre las diferentes configuraciones de 2D o 3D

Al clicar en cada uno podremos ver como la interfaz del editor, cambia drásticamente

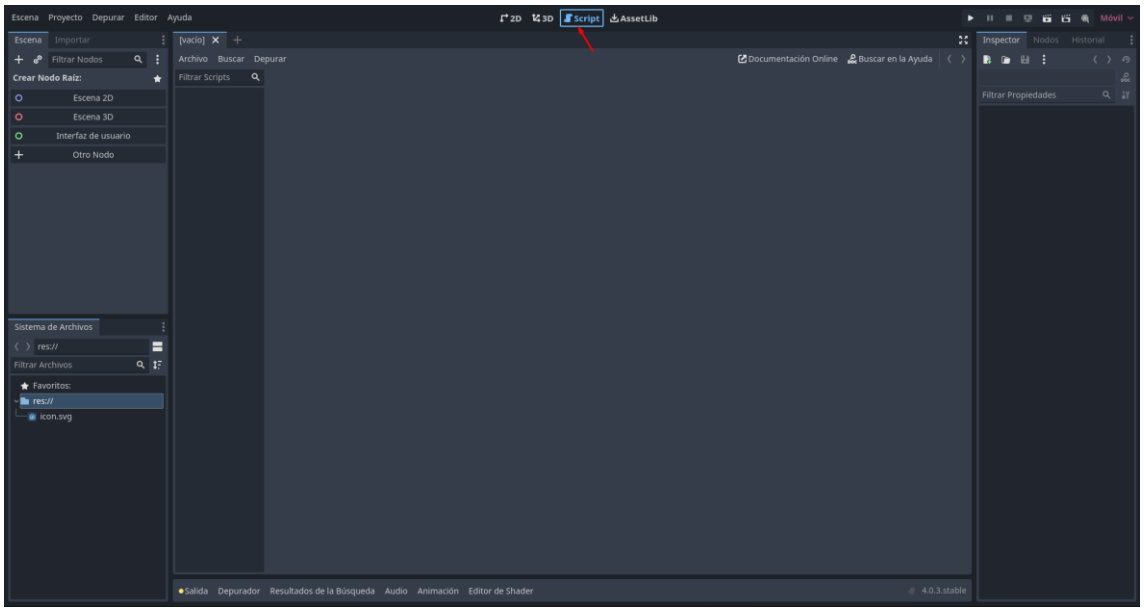
Modo 3D:



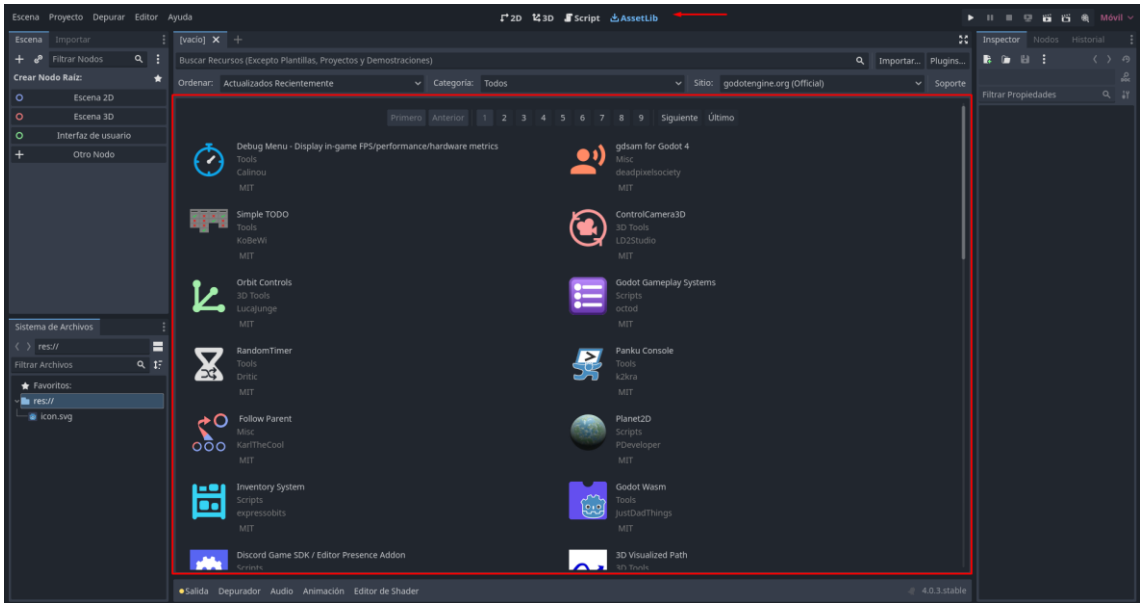
Modo 2D:



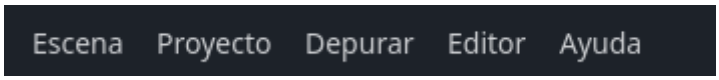
También, podemos clicar donde vemos el botón de “Script” donde podremos agregar scripts a cada zona que creemos en nuestro juego/programa



Por último, al clicar en “AssetLib” podremos visualizar una librería de assets gratuitos con los cuales, podremos utilizar para agregar funcionalidades extra e nuestro programa y juego



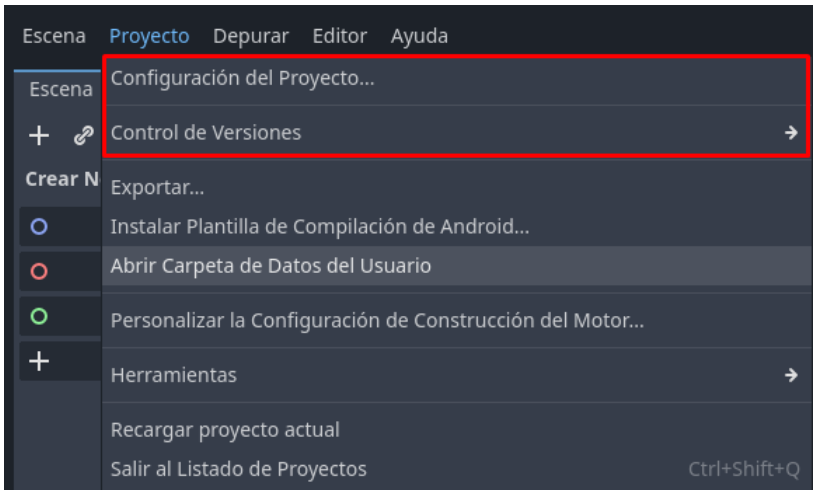
Ahora, vamos a irnos a la zona superior izquierda, donde podremos ver varias opciones de configuración de Godot



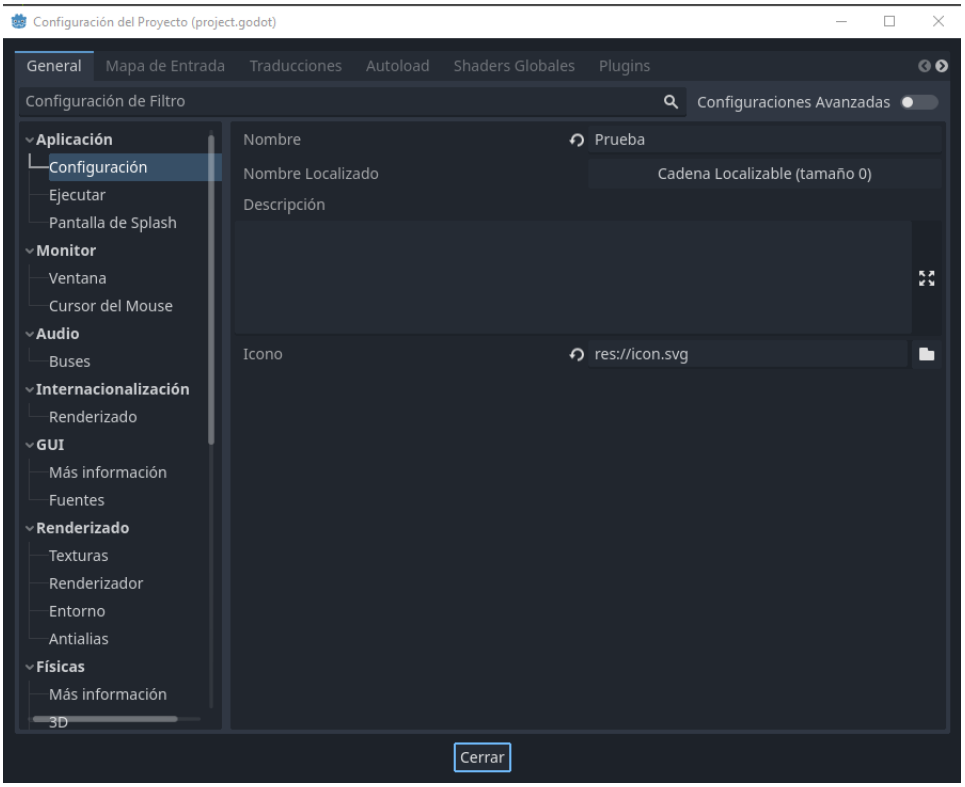
En la pestaña de “Escena”, podemos ver varias opciones, pero ¿que son las escenas?

Una escena en Godot, es una manera de organización y creación de partes de nuestro programa, por ejemplo, tenemos un juego de aviones, pues cada parte de ese juego (escenario, hitbox, en jugador, enemigos, proyectiles…) serian una escena para poder utilizar y modificar individualmente cada uno, para posteriormente unirlos todos en una

En la pestaña “Proyecto” podremos ver varias opciones donde modificar nuestro proyecto o tener un control de versiones con Git entre otras opciones

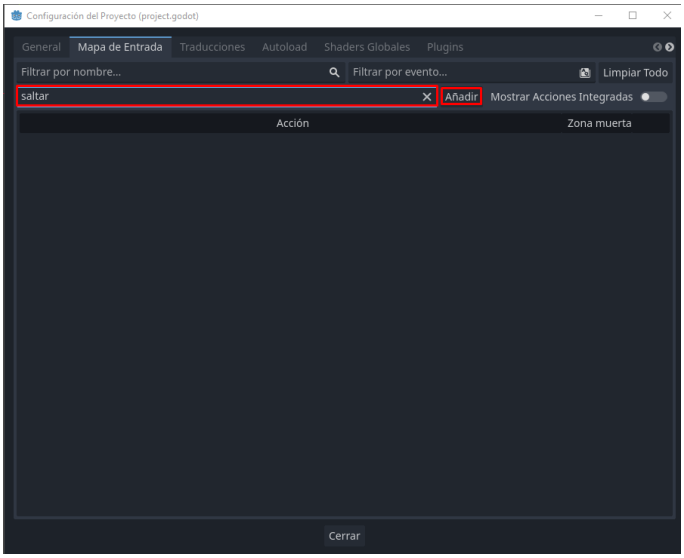


Hablemos sobre el configurador de proyectos, al clicar en dicha opción, podremos ver una ventana donde configurar varias opciones de nuestro juego

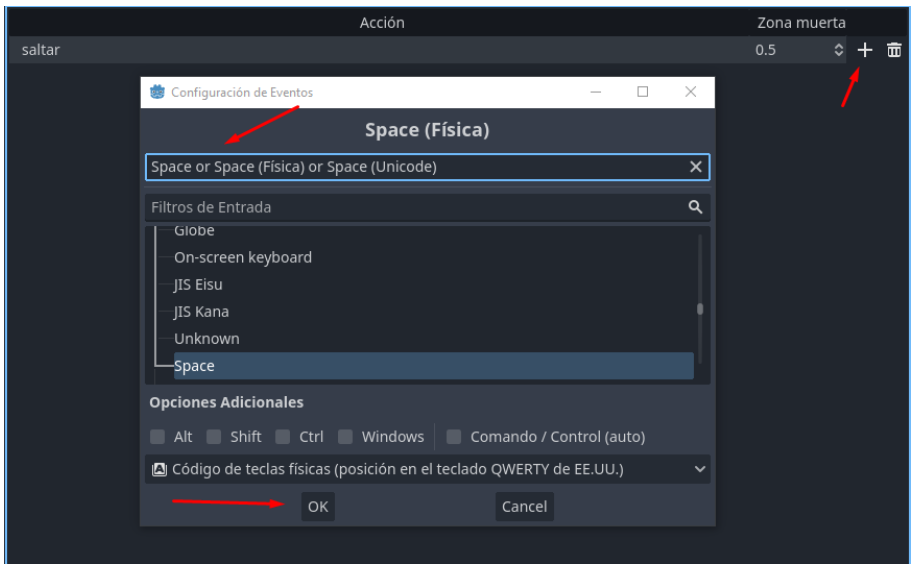


En la pestaña de “General” podemos ver varias opciones para la aplicación, resoluciones, audio, rederizado… los cuales podremos ajustar a nuestro gusto para obtener los mejores resultados con un simple clic.

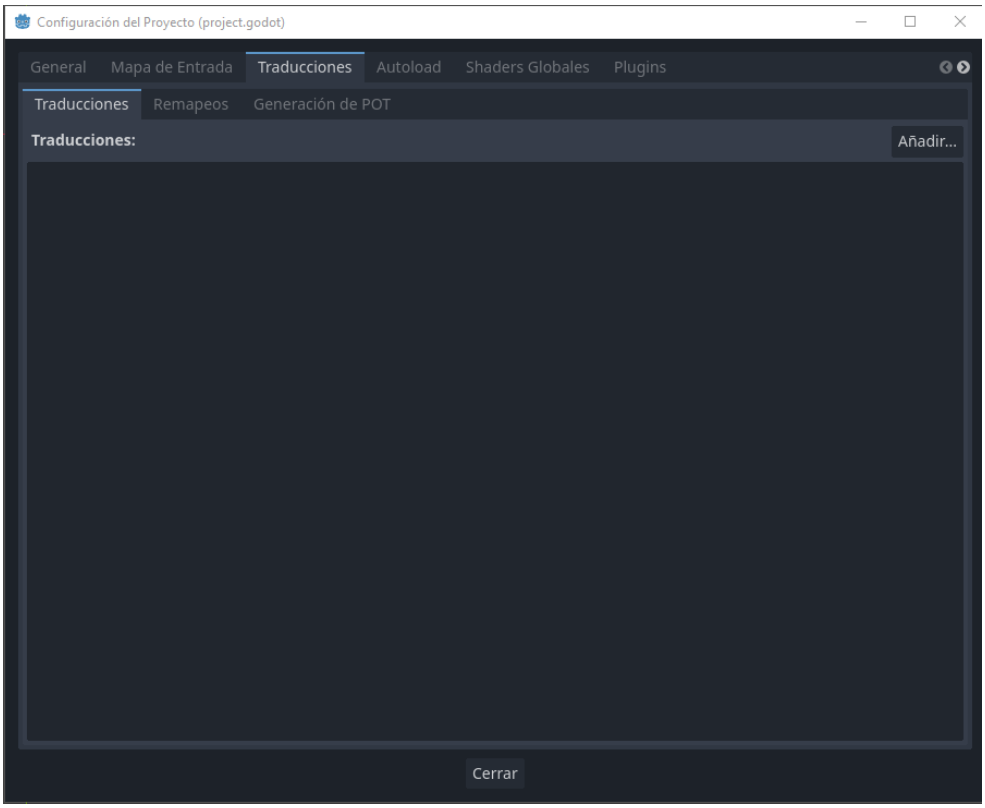
En la pestaña de “Mapa de entrada” definiremos las teclas que vamos a utilizar en nuestro programa, para posteriormente, dar una funcionalidad (o varias) a dicha tecla o botón, para ello, escribiremos el nombre de la acción a ejecutar, y clicaremos en “Añadir”



Al clicar en añadir, deberemos clicar en la tecla que vamos a utilizar para dicha opción tras clicar en el símbolo “+”



En la pestaña “Traducciones” podremos traducir directamente códigos y facilitarnos la escritura de dicho código



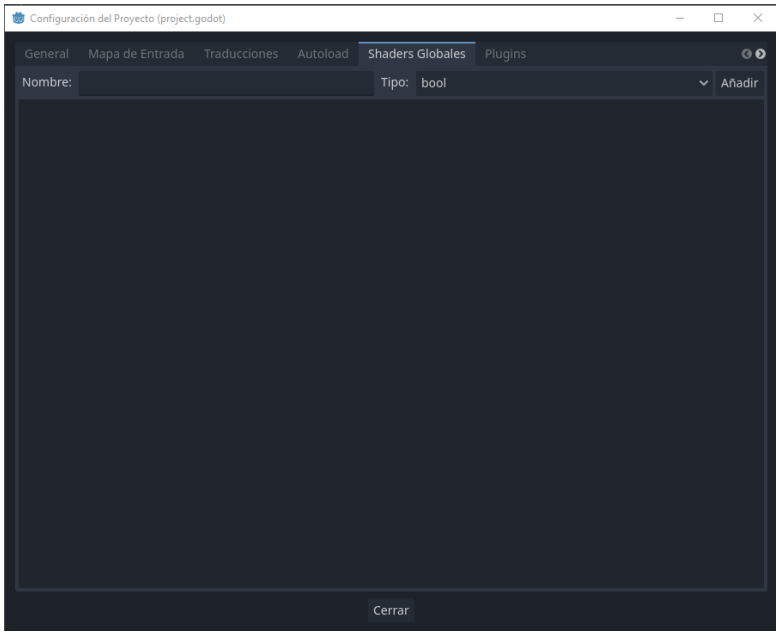
La siguiente pestaña es “Autoload” donde podremos cargar automáticamente una escena o un script

Usando este concepto, puede crear objetos que:

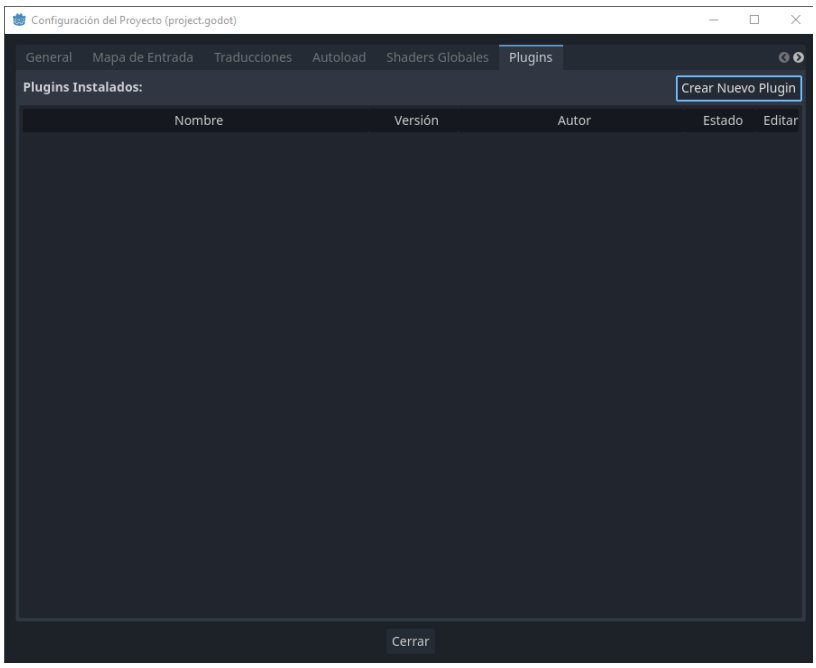
- Siempre están cargados, no importa qué escena esté en ejecución.
- Puedes almacenar variables globales como la información del jugador.
- Puedes manejar el cambio de escenas y las transiciones entre escenas.
- Actúa como un singleton, ya que GDScript no soporta variables globales por diseño.

En la pestaña visual Shaders así como VisualScript es una alternativa para los usuarios que prefieren un enfoque gráfico de la codificación, los VisualShaders son la alternativa visual para crear shaders.

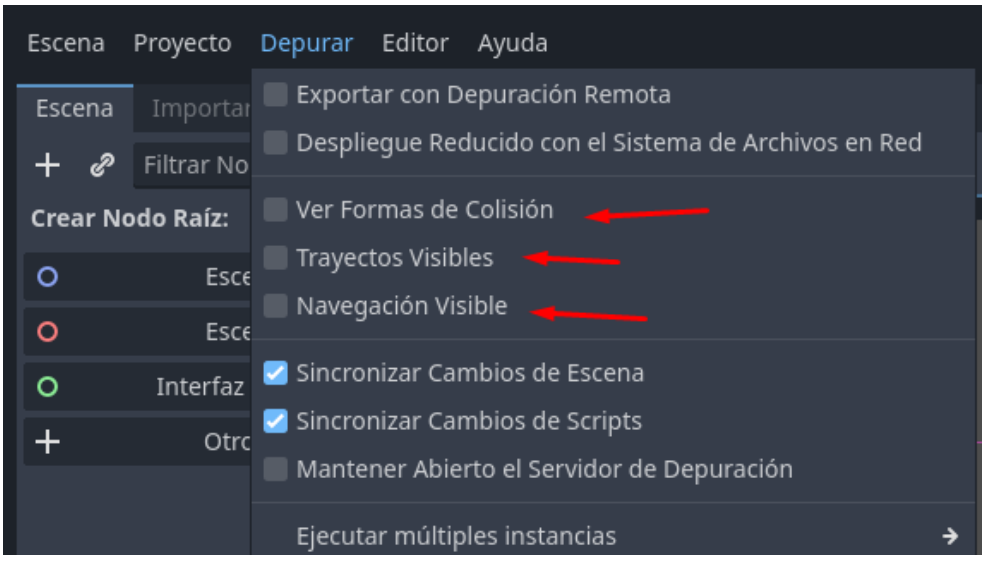
Dado que los shaders están intrínsecamente ligados a lo visual, el enfoque basado en gráficos con vistas previas de texturas, materiales, etc. ofrece mucha más comodidad en comparación con los shaders basados puramente en scripts. Por otra parte, los VisualShaders no exponen todas las características del script de los shaders y el uso de ambos en paralelo podría ser necesario para efectos específicos.



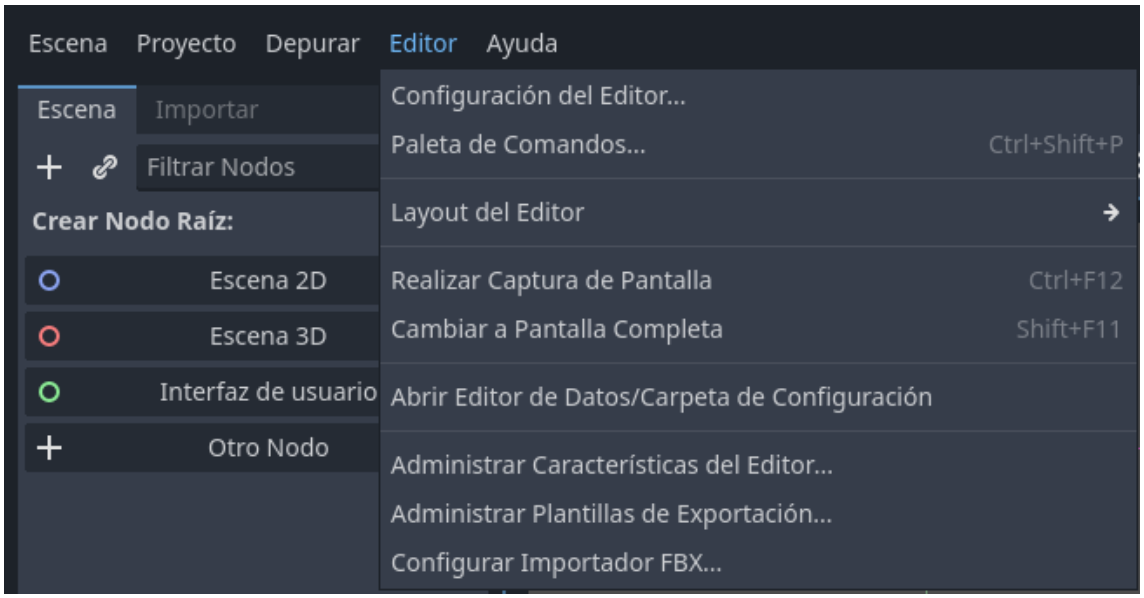
Y por último, podemos ver la pestaña de pluggins, donde podremos desde crear nuestros pluggins como adquirirlos desde la librería anteriormente mostrada



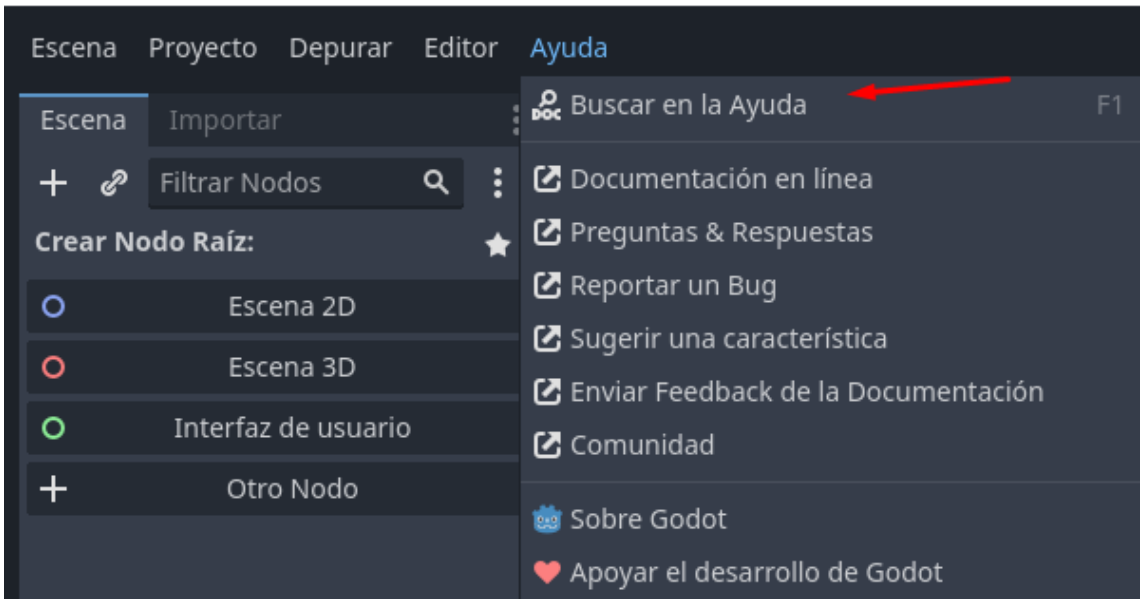
Tras acabar con las pestañas, vamos a la siguiente mostrada en el menú principal, situado en la zona superior izquierda, llamado “depurar” donde podremos ver diferentes ayudas a la hora de ajustar nuestro programa y visualizar diferentes elementos, los cuales para el usuario final serán invisibles, como las zonas de colisión



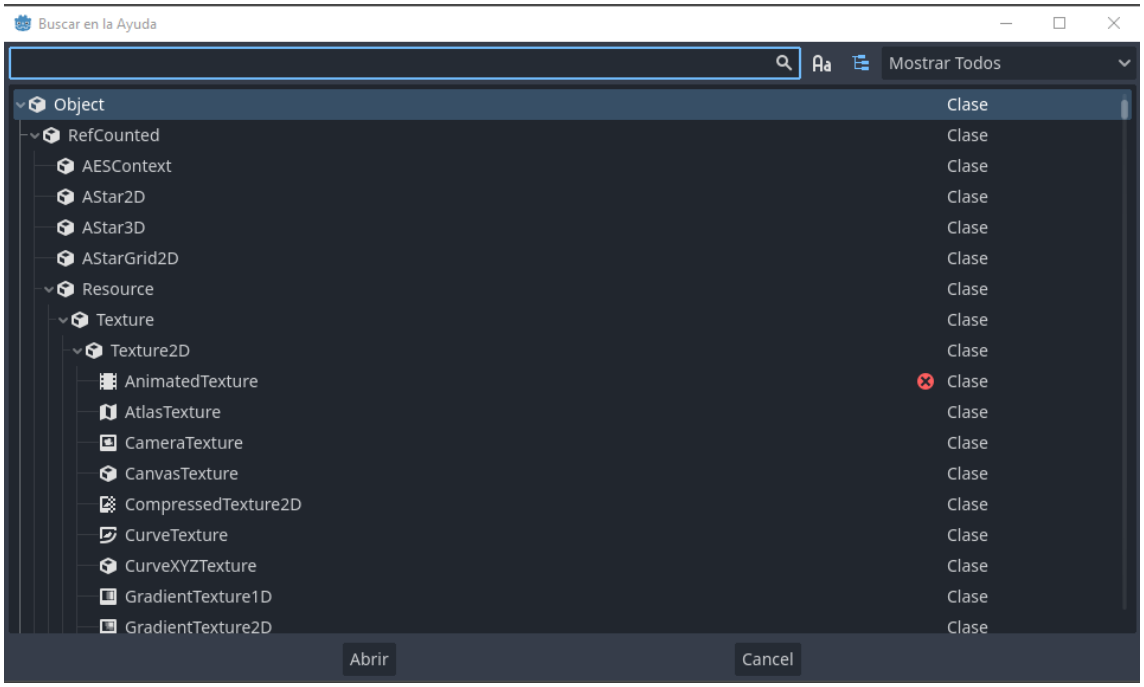
En la pestaña de “Editor” podemos editar zonas del programa y ajustarlo a nuestro gusto para trabajar más cómodamente

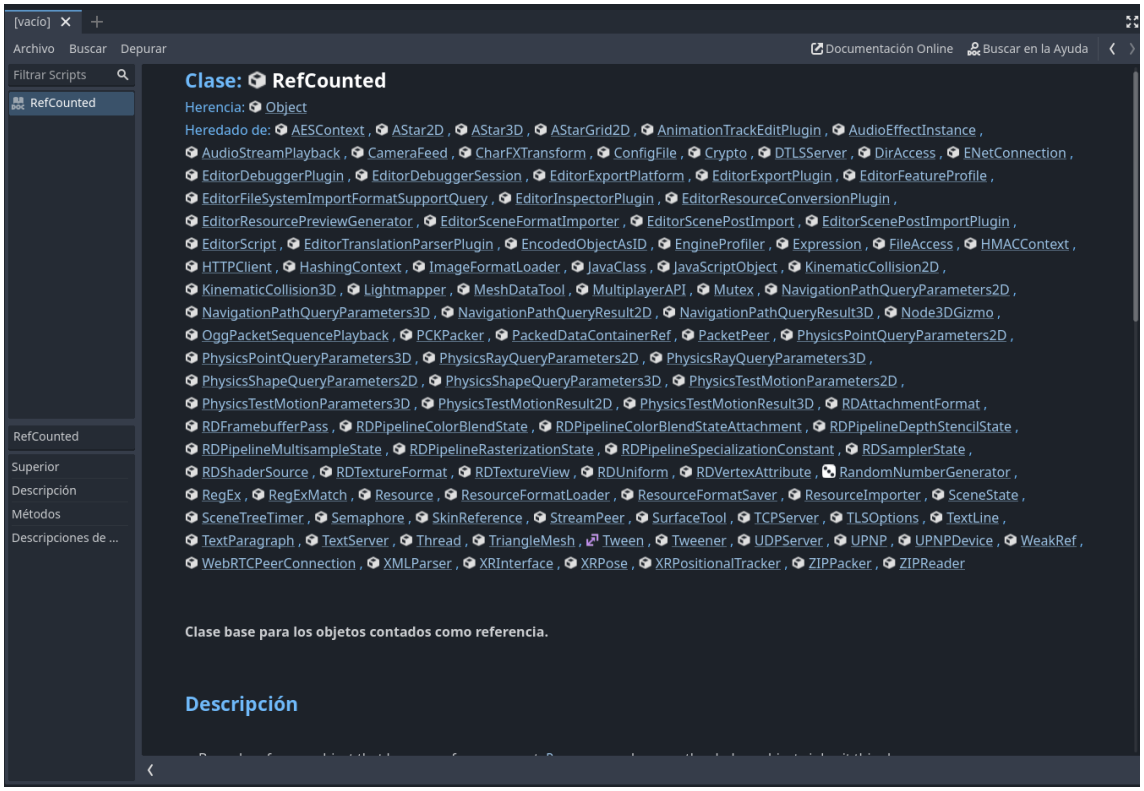


Por ultimo, veremos la pestaña de “Ayuda” donde el propio programa, nos proporciona una documentación de los nodos del programa los cuales son propios de Godot

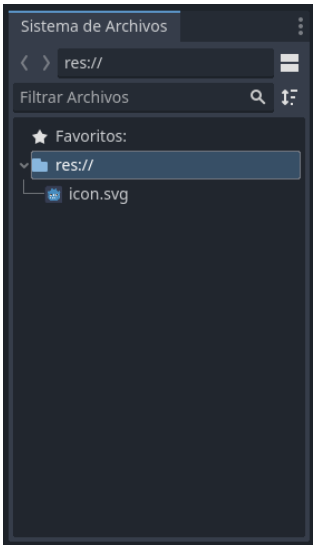


Clicaremos sobre la ayuda y posteriormente, buscaremos el nodo el cual necesitamos, para posteriormente, acceder a una documentación propia





Ya tenemos el menú completado, ahora nos dirigiremos al a zona inferior izquierda, donde podremos ver el sistema de archivos, el cual, es donde podremos visualizar las carpetas que creemos para todo lo que necesitemos (audios, imágenes, assets, escenas, scripts…)



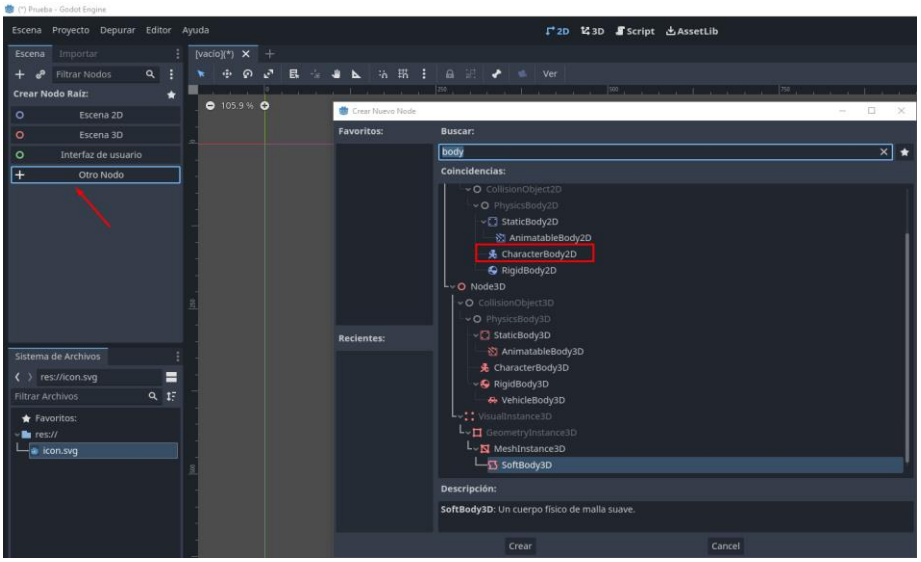
En esta zona, deberemos tener un orden considerado, ya que en función del progreso del juego/programa nos resultará más sencillo la gestión de los recursos agregados

Seguimos con la zona superior al sistema de archivos, en la cual llevaremos el orden de cada escena, con todos sus nodos insertados

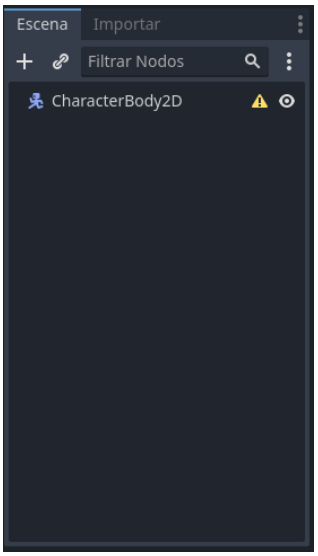
Dependiendo de lo que necesitemos, usaremos un tipo de escena o nodo, en estos casos tenemos las diferentes opciones como “Escena 2D” hecho para juegos en 2D, “Escena 3D” para juegos en 3D, “Interfaz de usuario” en la cual, por ejemplo, podemos agregar el menú de un juego, los botones e interfaz del usuario que utilizará para navegar en el juego antes de ingresar en él, y por ultimo, “Otro nodo” que dependiendo de nuestros requerimientos, puede ser una opción viable ya que no se ajusta en ninguna de las otras opciones

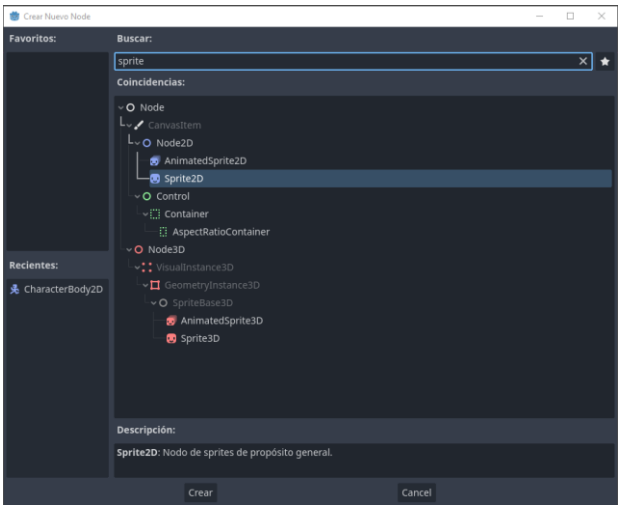
Al clicar en cada una de las opciones, podremos observar diferentes nodos los cuales podremos usar a nuestra libre elección clicando en el icono “+”, por ejemplo, vamos crear un nodo para nuestro personaje principal.

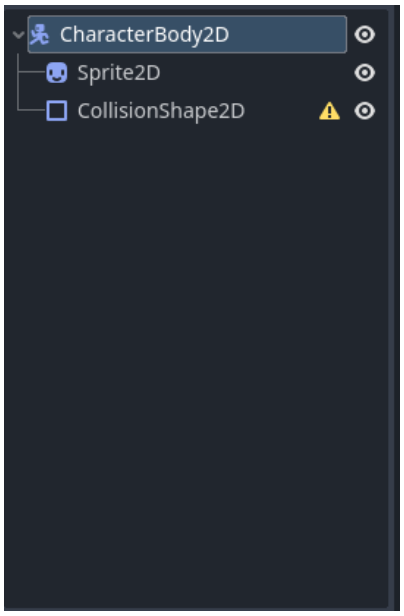
Clicaremos en “Otro nodo” y buscaremos “CharacterBody2D”



Al clicar en crear, se nos agregará el nodo en la zona superior izquierda, pero observamos que nos da un fallo, y es que en este caso, necesitamos que el “personaje” tenga una textura y una colision, para ello, de nuevo clicamos en el “+” y buscamos “Sprite2D” para agregar la texto

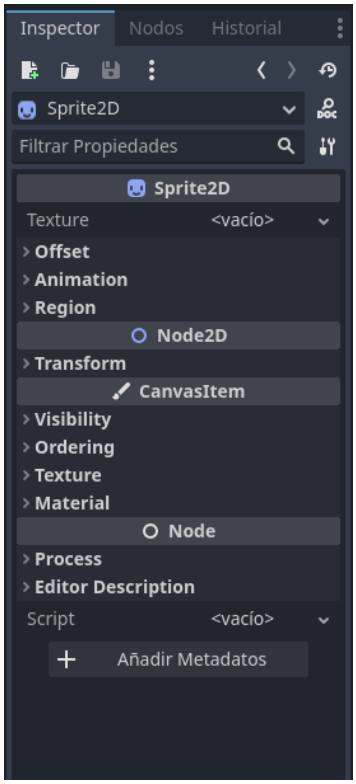






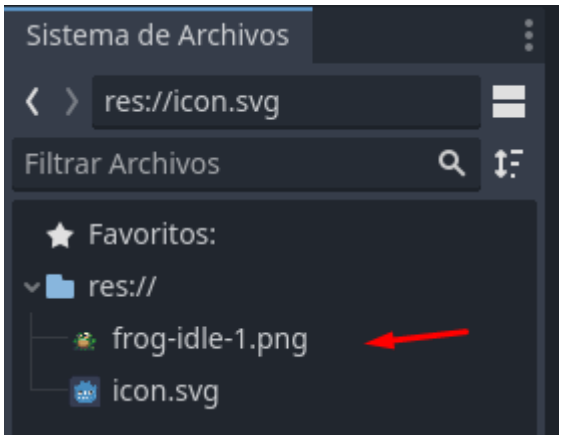
Por ultimo, agregaremos la colision agregando un nodo llamado “CollisionShape2D” y la estructura básica del personaje debería quedarnos de la siguiente forma.

Pero claro, donde esta el personaje, pues lo crearemos ahora, al clicar en el nodo de “**Sprite2D**” podremos ver en la zona de la derecha diferente opciones de dicho nodo.



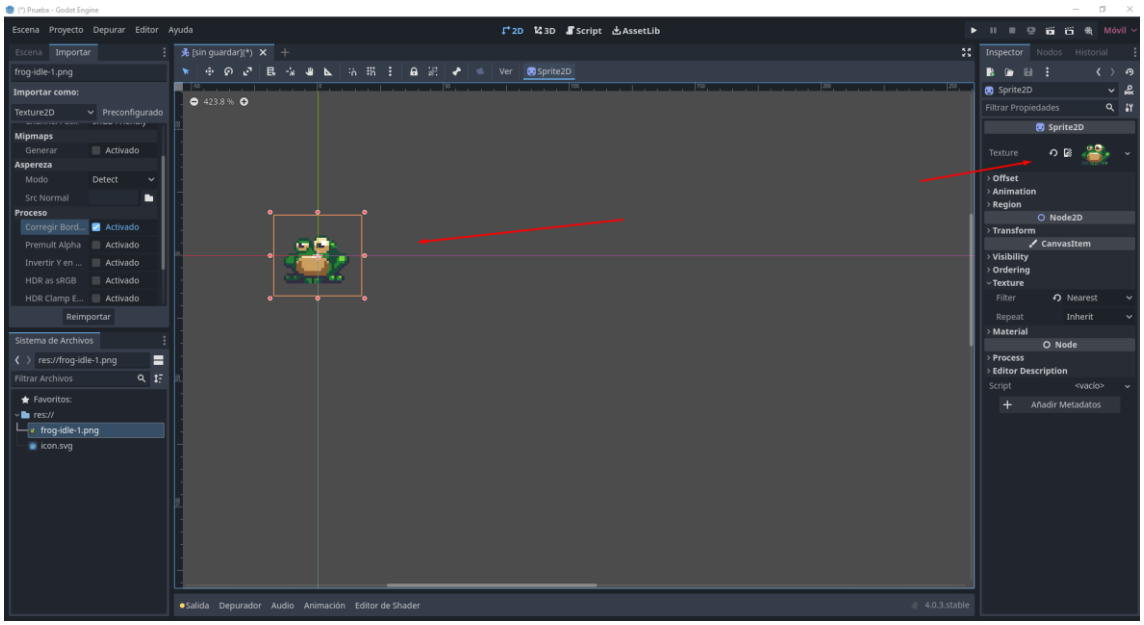
Y en la zona de “Texture” deberemos agregar la imagen de nuestro personaje para que se añada al editor, voy a agregar una personalizada para el ejemplo.

Para ello, simplemente arrastraremos la imagen a la zona del sistema de archivos vista anteriormente en la zona inferior izquierda

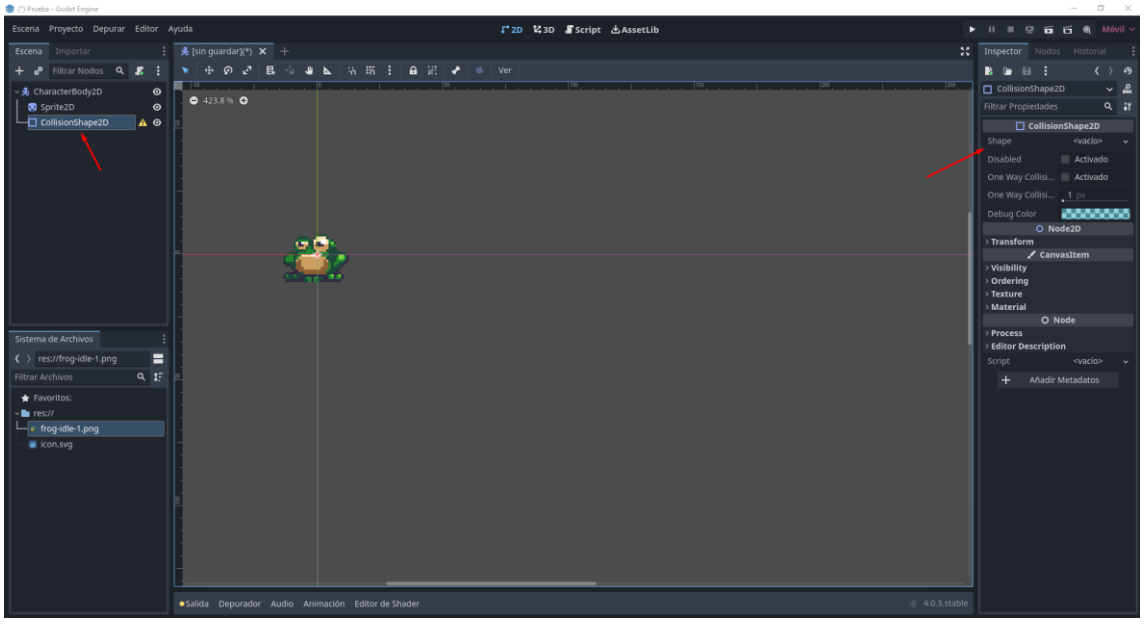


Al tener la imagen en nuestro sistema de archivos, ya podremos utilizarla en cualquier momento, en la cual en este ejemplo, de nuevo arrastraremos del sistema de archivos a la zona de “Texture” en el nodo de sprite2D

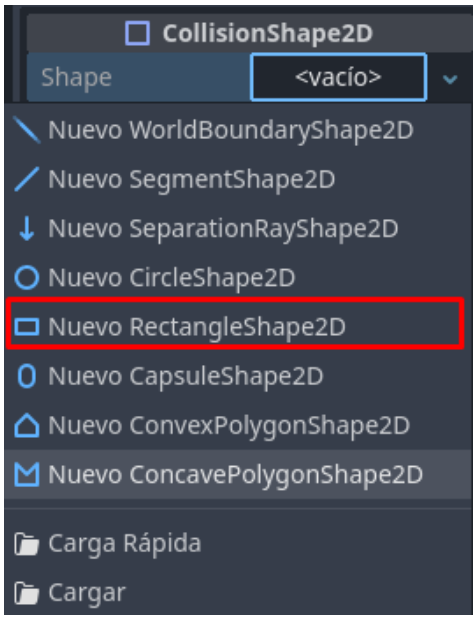
Al arrastrarlo, podremos visualizar en nuestro editor central, que se ha creado correctamente la imagen en el Sprite



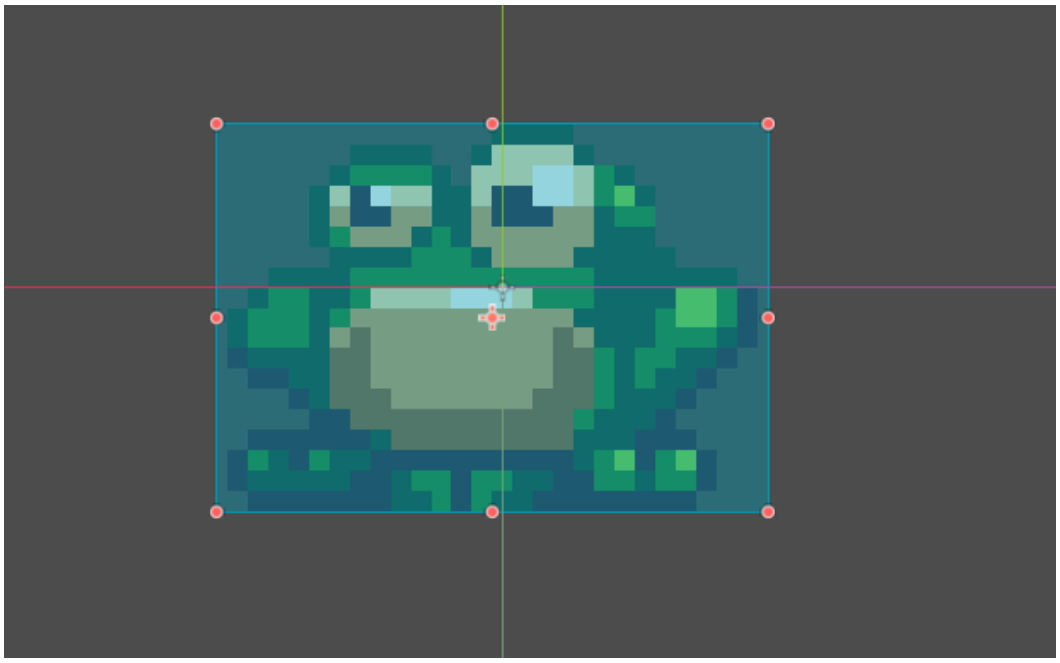
Pero claro, aun falta la colision, la cual deberemos agregarle en las opciones del nodo “collisionshape2D” creado anteriormente, con lo que clicaremos en dicho nodo y podremos ver las siguiente opciones



Ahora, para agregar la colision, deberemos clicar en “Shape” donde nos mostrará un pequeño listado de formas de colision, en mi caso, utilizaré la rectangular



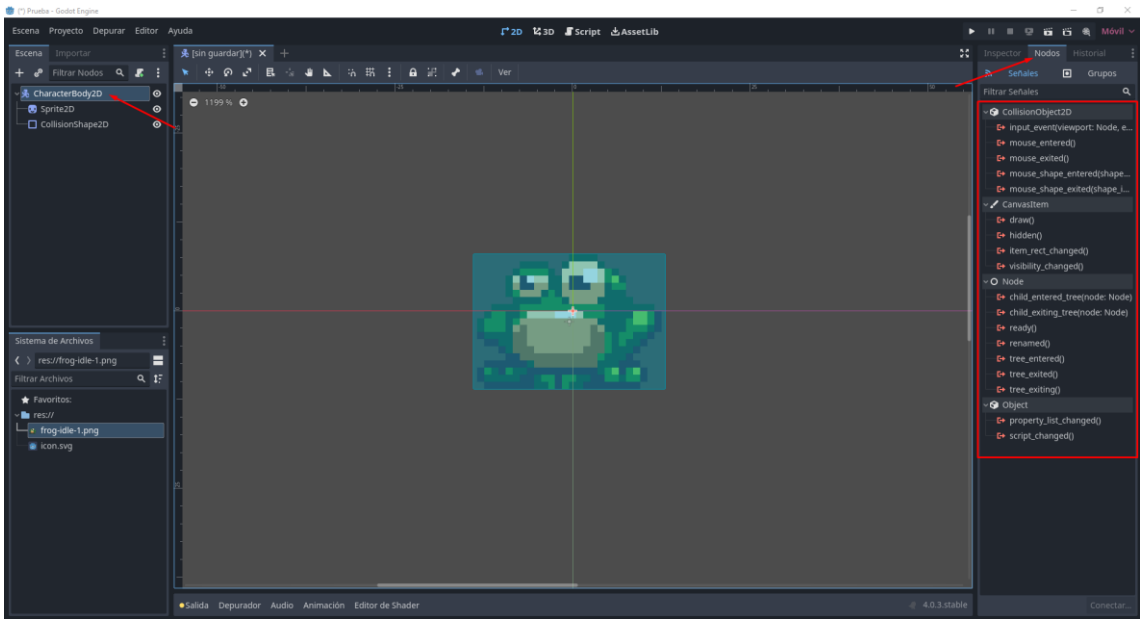
Nos aparecerá el cuadradp/recatangulo de colision, ahora deberemos ajustarlo al personaje para que quede a nuestro gusto



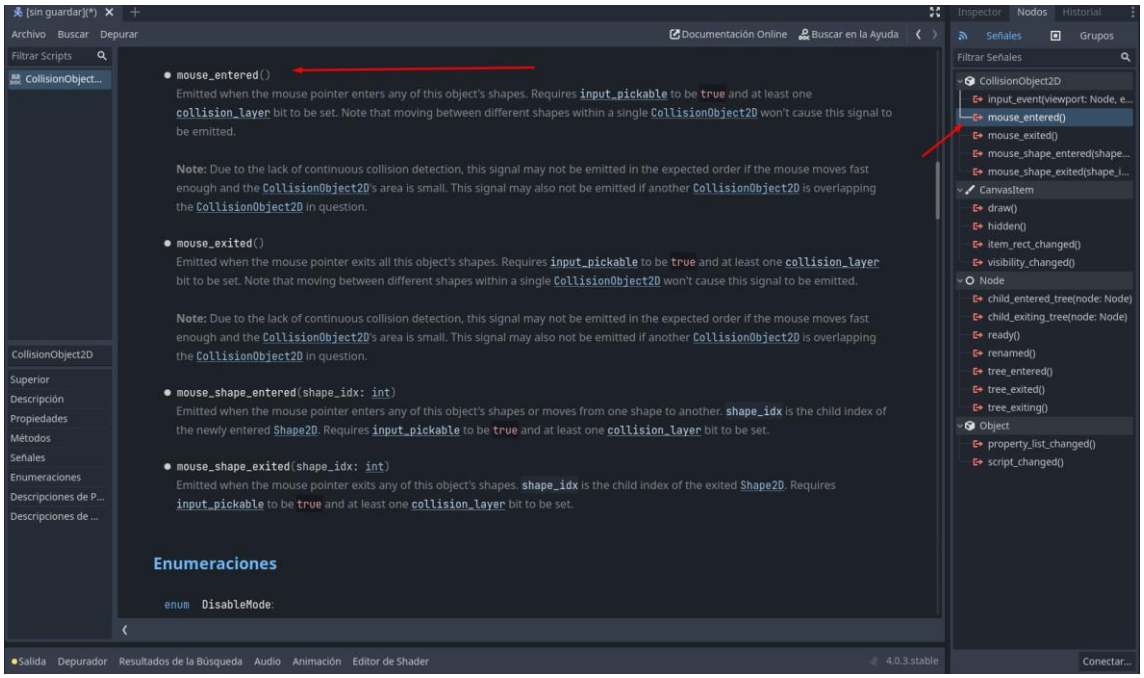
Y ya tendremos nuestro personaje creado, ahora faltaría darle movilidad y ajustar con scripts su movimiento, animaciones, efectos de sonido, y mecánicas principales

Por último, vamos a ver los tipos de nodos que podemos encontrarnos en Godot.

Para ello, deberemos irnos a uno de los nodos utilizados para el personaje (por ejemplo) y clicar a la zona de la derecha en la pestaña de “nodos”



Podemos observar diferentes nodos que podremos utilizar en los scripts que necesitemos, en los cuales, si no sabemos para que sirve cada uno, podemos irnos a la ayuda y buscar su utilidad principal



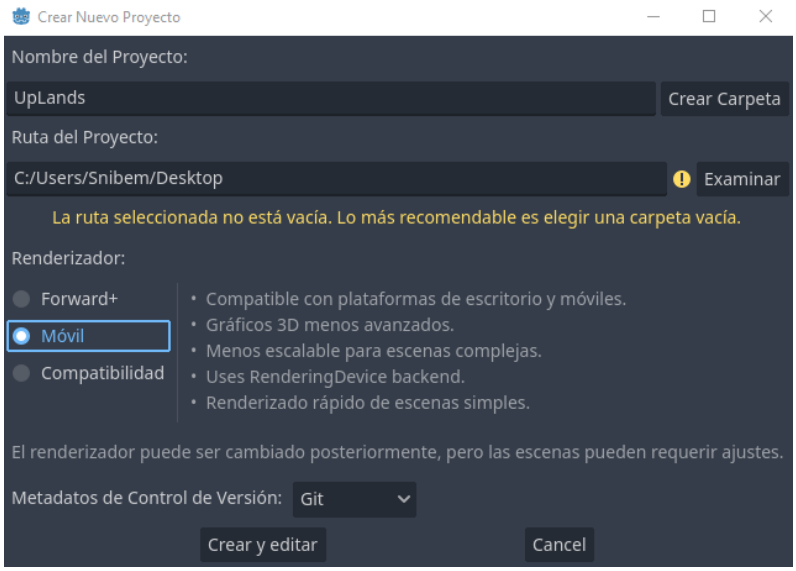
Y ya tendremos un pequeño resumen de la infinidad de cosas que podemos hacer en este programa de libre uso y gratuito

6. Tareas

Tarea 1: Ajustes principales del proyecto

Subtarea 1.1: Renderizado del proyecto

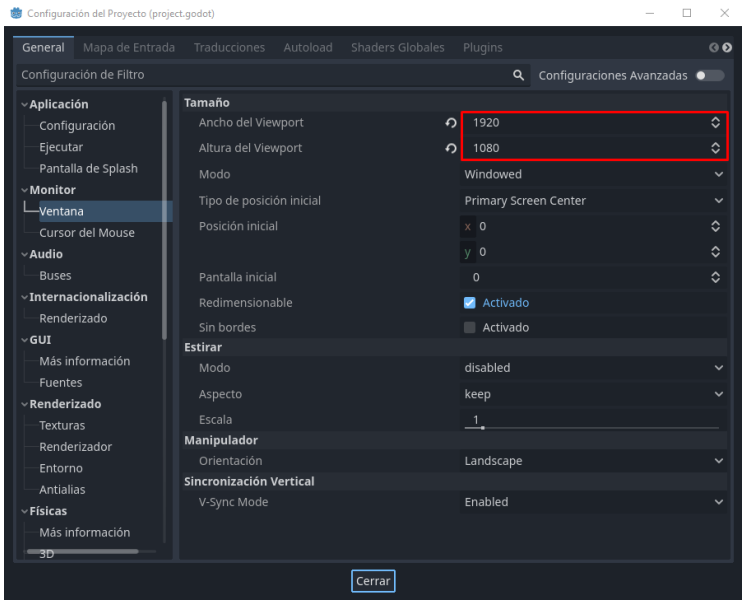
Para comenzar, deberemos escoger el tipo de proyecto que vamos a crear, como es un juego 2D de plataformas pixelado, no requiere de un nivel grafico muy potente, con lo que en el renderizador, escogeremos la opción de “Movil” y crearemos el proyecto



Subtarea 1.2: Resolución del proyecto

También, deberemos ajustar la resolución del juego a partir de los ajustes del proyecto

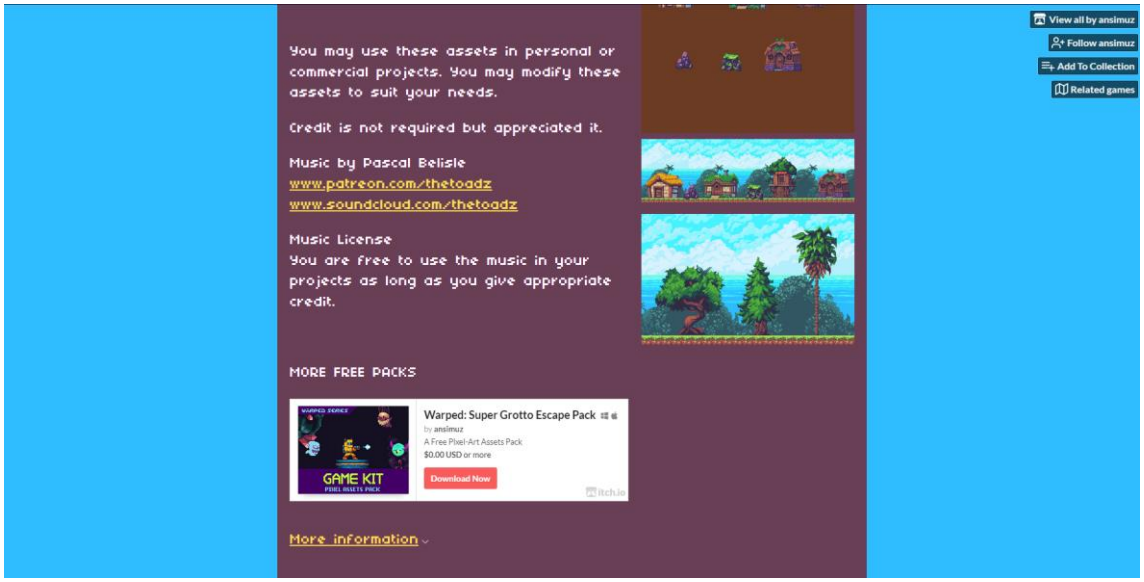
En mi caso, utilizaré una resolución de 1920x1080



Subtarea 1.3: Assets del proyecto

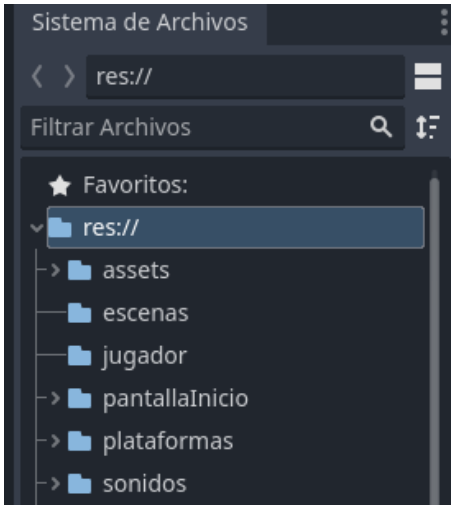
Obtendremos los assets principales de la página de Itch.io donde obtendremos la mayoría de los assets utilizados

https://ansimuz.itch.io/sunny-land-pixel-game-art



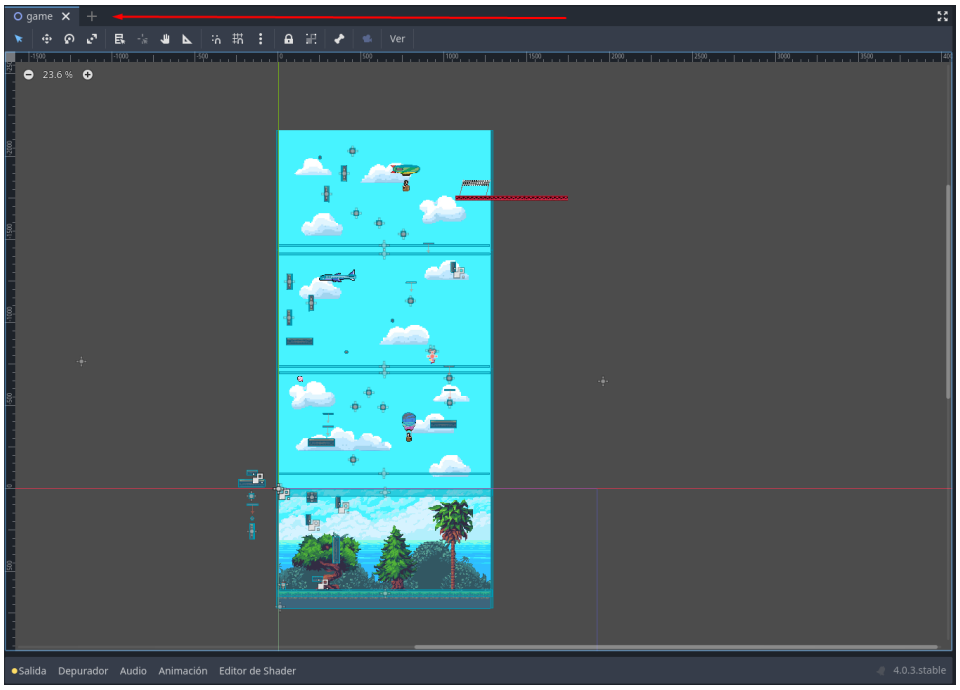
Tarea 2: Creación de árbol de directorios

Al crear el proyecto, deberemos empezar a organizar los assets que utilizaremos, para ello, crearemos el árbol de carpetas, ordenando todo lo que utilizaremos



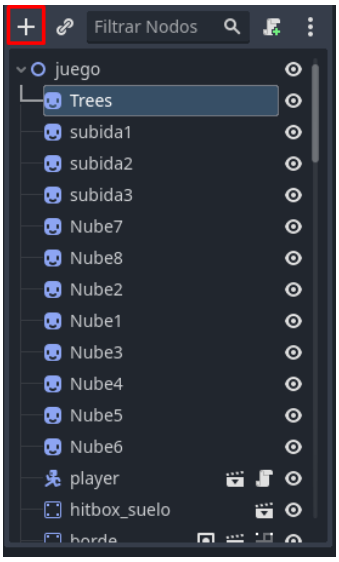
Tarea 3: Escena principal

Comenzaremos con la escena principal del juego, que será el escenario donde jugaremos, a nuestra escena principal, le llamaremos “game”, para ello crearemos una escena y la guardaremos con dicho nombre, esta escena, será donde el jugador estará y desde donde otras escenas se agreguen a esta para que todo esté en orden



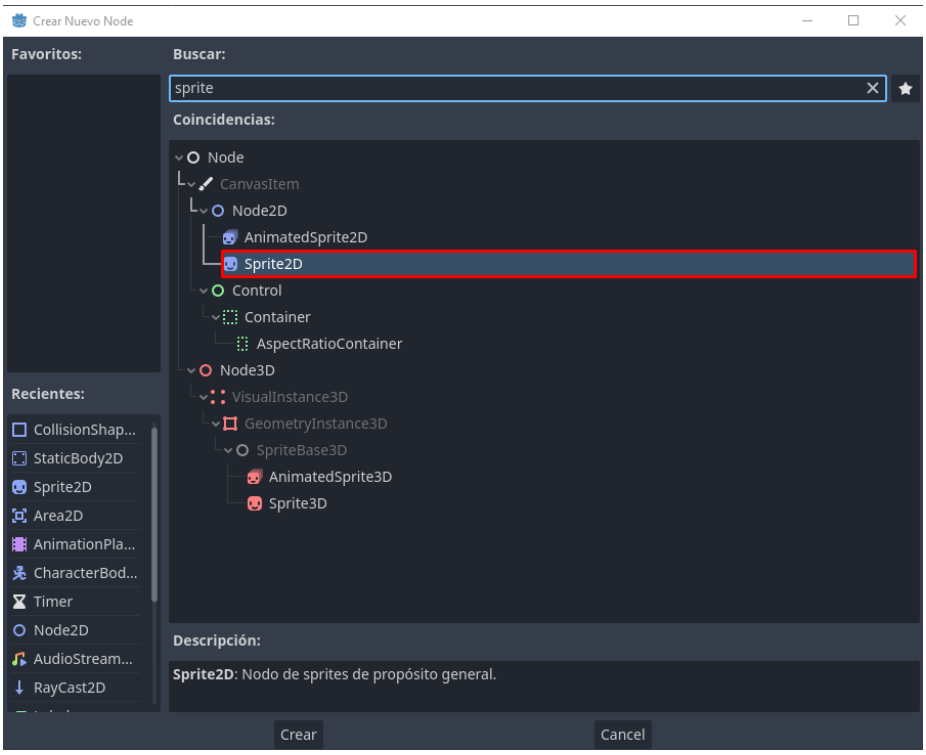
En esta escena, deberemos crear el escenario del juego, para ello, crearemos diferentes nodos

Subtarea 3.1: Creacion de nodos para la escena “Game”

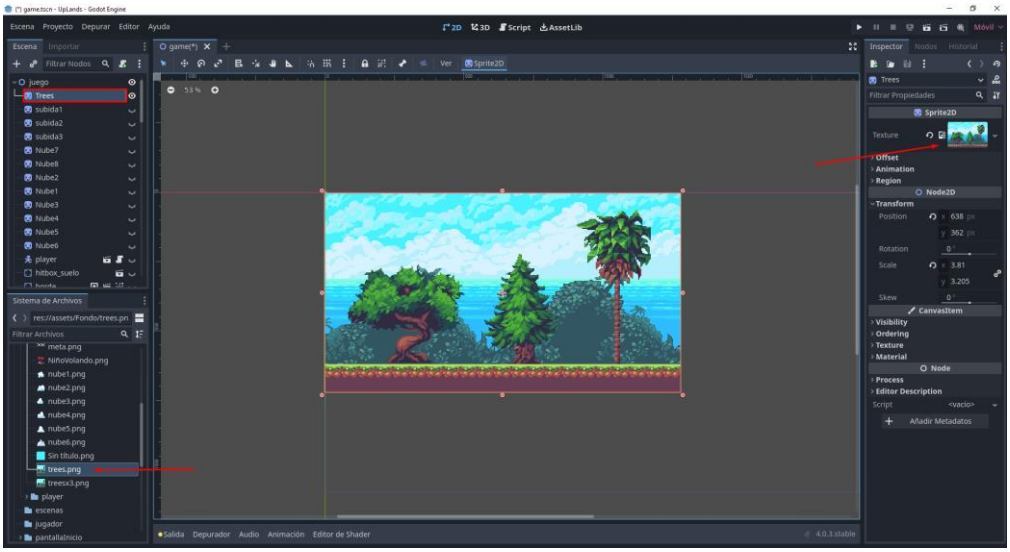


Para la escena “Game” deberemos crear el nodo “Sprite 2D” el cual es un nodo capaz de importar una imagen dentro de nuestra escena

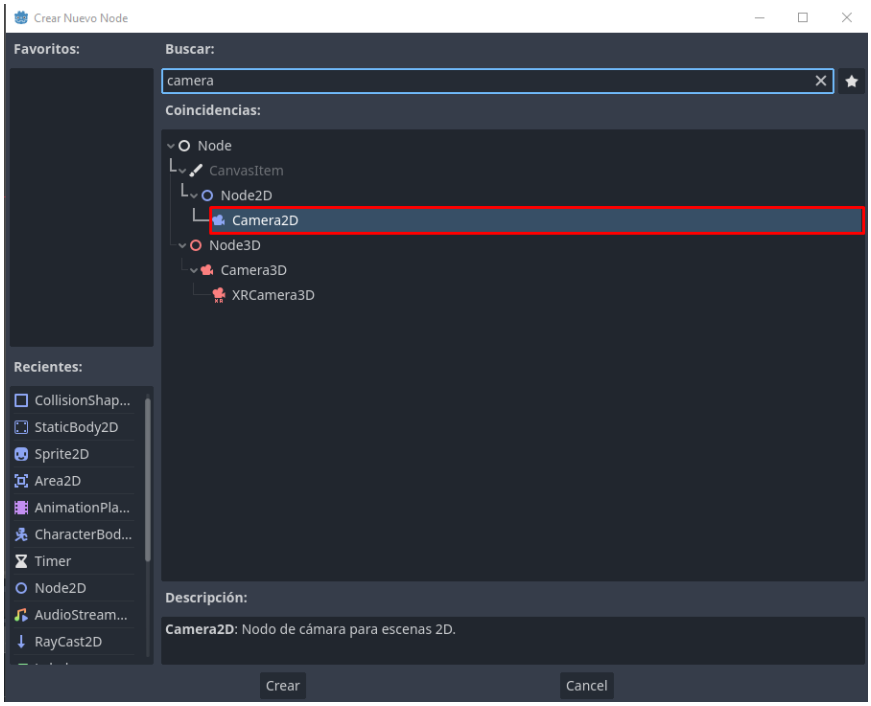
Para crear dicho nodo, deberemos clicar en el símbolo “+” y nos aparecerá una ventana, donde deberemos buscar el nodo que queramos



Crearemos dicho nodo, y le agregaremos la imagen arrastrando desde nuestro sistema de archivos hasta la zona de la derecha del nodo



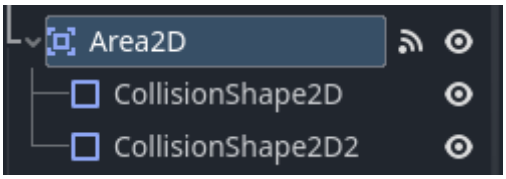
Seguidamente, crearemos un nodo para la cámara del jugador, en la cual queremos que la cámara cambie automáticamente cuando el jugador **suba** por la pantalla, teniéndola de manera fija



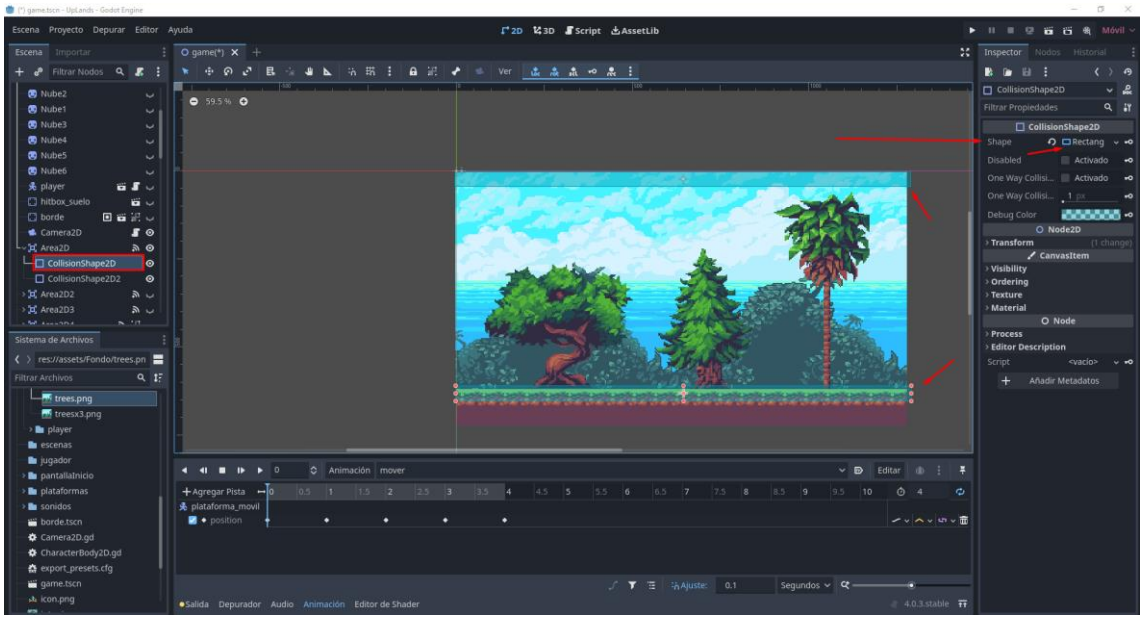
Para ello, deberemos crear un nuevo nodo, pero esta vez, buscaremos uno con el nombre de “Camera 2D”

Al agregarlo, podremos ver como aparece un cuadro rosa, indicando la zona donde la cámara apuntará de manera por defecto, pero esto no nos interesa, ya que queremos que la cámara esté en una posición final, y cambie de posición cada vez que el jugador, suba en el juego y no se le vea en la pantalla, con lo que deberemos generar áreas SIN COLISION para que cuando el jugador interactúe con dicha área, la cámara, cambie nuevamente.

Para originar las áreas sin colisión, deberemos buscar un nodo llamado “Area 2D” en la cual, dentro de ese nodo, deberemos crearle un nodo hijo “CollisionShape2D”, el cual cree esa colisión, pero al estar dentro de un “**Area2D**”, esa colisión no chocará con el jugador, con lo que podrá traspasarla y gracias a ello, enviar una señal a lo que necesitemos

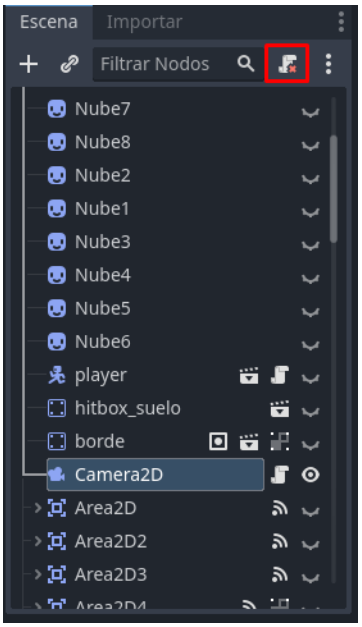


Tras crear dichos nodos, agregaremos esas áreas a nuestro juego, para ello, seleccionaremos los nodos de “collision shape” y clicaremos en “shape” para generar dicha colision y ajustarla a la zona que queramos



Y así, deberemos crear tanta “areas2D” como cambios de cámara tenga nuestro juego

Ahora, deberemos crear un script para el nodo de la cámara, para ello seleccionaremos nuestro nodo y clicaremos sobre el botón marcado en la captura



Al clicar, nos aparecerá donde queremos ubicar nuestro script, y seguidamente, se creará dicho script, el cual, deberemos editar seguidamente

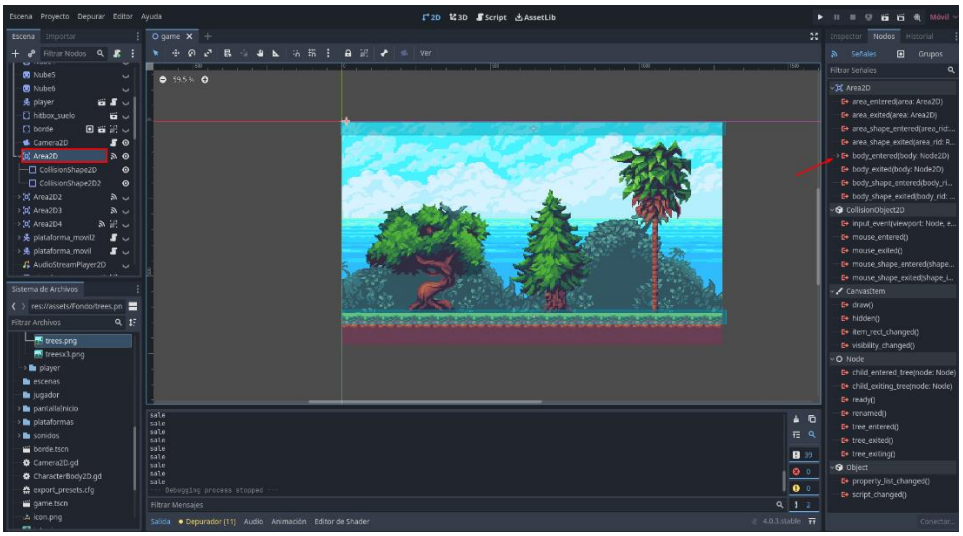
En nuestro script, ubicaremos nuestra cámara en una zona por defecto (aunque físicamente esté ubicado en otro) y segundo el jugador colisione con las áreas creadas anteriormente, dicha cámara cambie de ubicación

Para definir dicha camara por defecto, deberemos utilizar la funcion de godot llamada: “func _ready()” la cual hace que de manera “por defecto” todo lo que este en esa funcion se ejecute desde el principio, con lo que ajustaremos las coordenadas de la camara y esa será la primera camara que veamos por defecto en nuestro juego

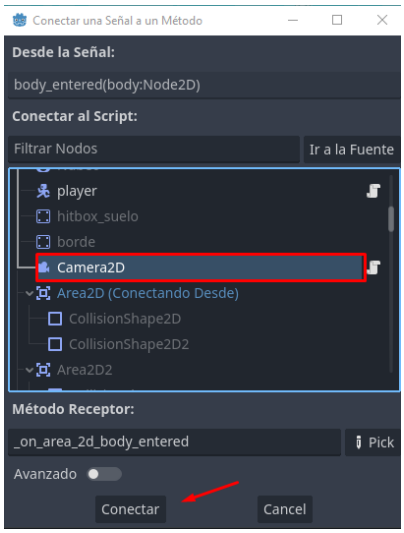
```
func _ready():
    position.x = 638
    position.y = 360
```

Como el juego se basa en subir, la cámara solo modificara solo en su altura

Para cambiar sus coordenadas al subir deberemos conectar nodos del “Area2D” creado anteriormente con nuestro script de la camara para que al colisionar, la camara cambie, para ello, deberemos irnos a nuestra área, y clicar a la derecha en “Nodos” para asi, conectar el nodo “_on_are_2d_body_entered(body)” a nuestro script



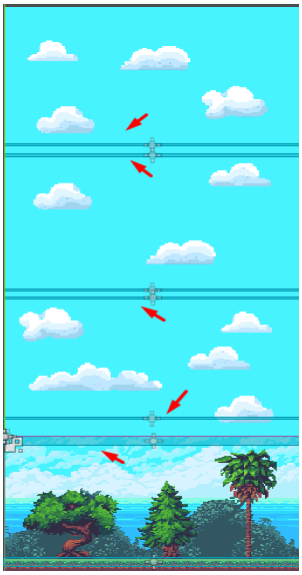
Daremos doble clic y lo vincularemos con nuestro nodo de la cámara



Y ya podremos agregar la posición de la cámara para cada área que creemos haciendo el mismo proceso

```
19 func _on_area_2d_body_entered(body):
20     if body.name == "player":
21         position.x = 638
22         position.y = 360
23
24
25 func _on_area_2d_2_body_entered(body):
26     position.y = -360
27
28
29
30 func _on_area_2d_3_body_entered(body):
31     position.y = -1080
32
33
34
35 func _on_area_2d_4_body_entered(body):
36     position.y = -1800
37
```

Por último, agregaremos más nodos “Sprite2D” para seguir aumentando el juego y crearemos sus áreas para el cambio de cámara cada vez que el jugador suba o baje



Ahora crearemos más escenas y las agregaremos a nuestra escena principal, la cual el jugador tendrá acceso al iniciar el juego

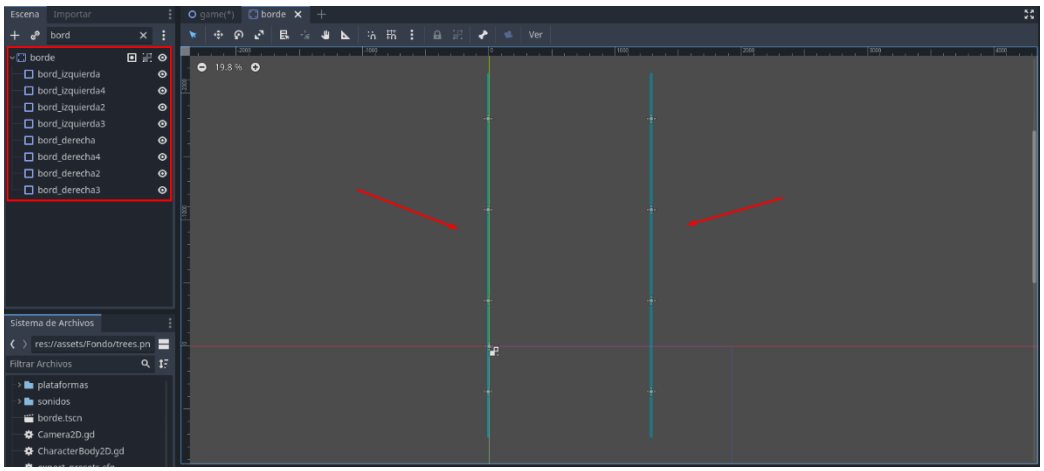
Con esto, ya tendremos nuestra escena principal casi acabada, pero más adelante deberemos agregarle todas las escenas que vayamos creando para completar y tener un orden en el proyecto

Tarea 4: Escena de bordes del mapa

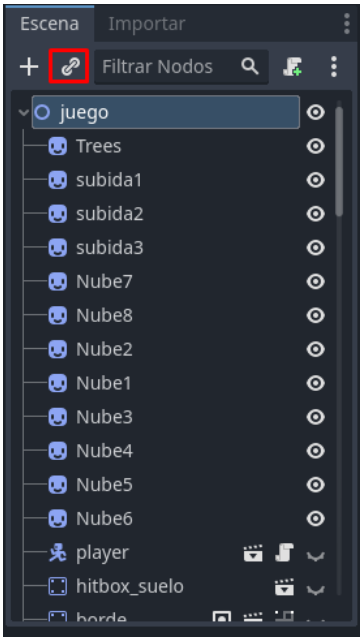
Para la creación de los bordes, crearemos una escena donde deberemos crear dichas colisiones y agregarla a nuestra escena principal del juego

Comenzaremos creando la escena y agregándole el nodo “StaticBody2D” el cual es una colisión que choca con lo que toque a modo de pared.

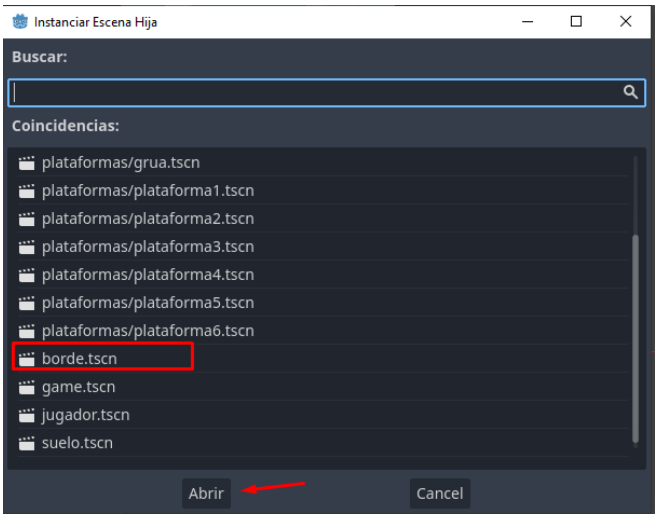
Dentro del nodo, incluiremos todos los nodos que queramos para las paredes de nuestro juego, en mi caso, he creado bordes por cada pantalla, tanto izquierda como derecha



Al tener los bordes creados, solo deberemos instanciar la escena a nuestra escena “Game” o escena principal, para ello, nos iremos a nuestra escena **principal** y clicaremos en el botón marcado en la captura



Al clicar, nos aparecerá una ventana donde podremos instanciar otras escenas a la seleccionada, con lo que clicaremos en la escena de “bordes”



Y ya podremos ver nuestra escena instanciada en la escena principal del juego, con lo que habremos agregado los bordes a nuestro juego

Tarea 5: Escena de jugador

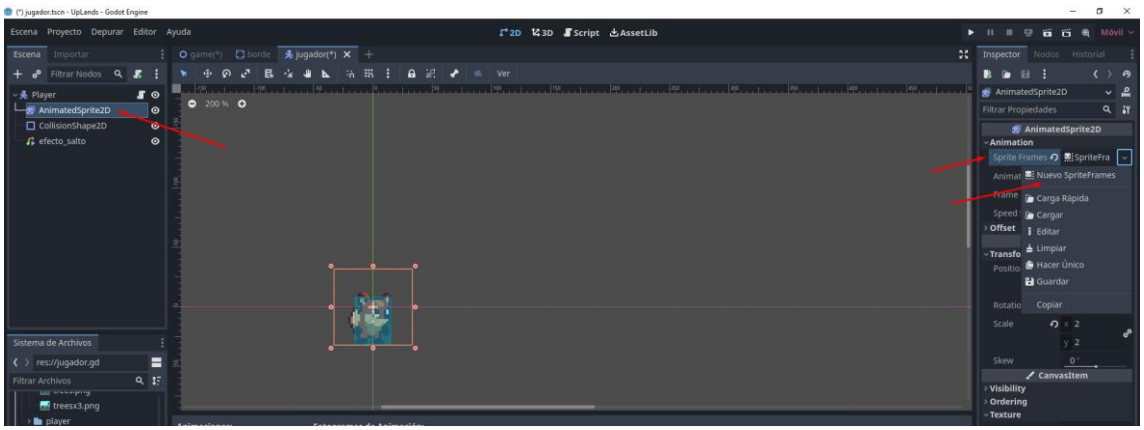
Para nuestro personaje el cual controlaremos, crearemos una escena y le agregaremos el nodo “CharacterBody2D” el cual es un nodo para nuestro personaje

Tras ese nodo, deberemos crear nodos hijos, los cuales serán:

- **AnimatedSprite2D** → Nodo donde pondremos la textura y animaciones de nuestro personaje
- **CollisionShape2D** → Nodo donde agregaremos la colisión de nuestro personaje
- **AudioStreamPlayer2D** → Nodo para agregar sonido al santo de nuestro personaje

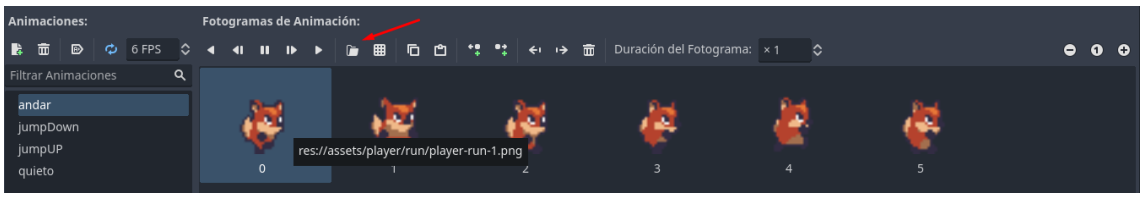
Subtarea 5.1: Textura y animaciones del personaje

Para agregar textura a nuestro personaje, utilizaremos nuestro “AnimatedSprite2D”. Para ello, clicaremos en nuestro nodo y agregaremos u nuevo Sprite Frame en la zona superior derecha

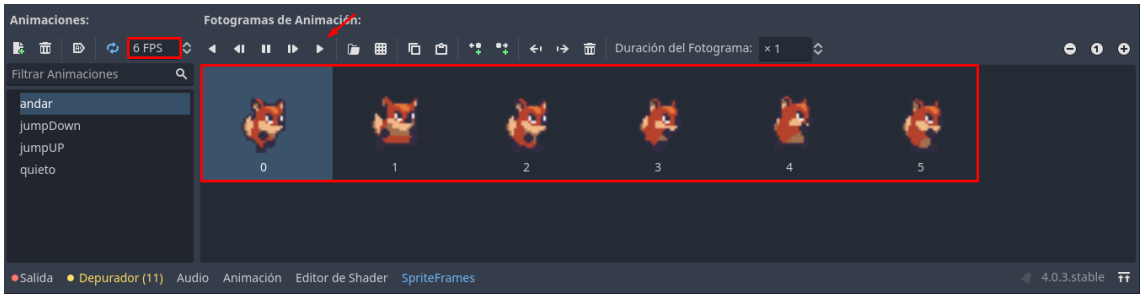


Al clicar, nos aparecerá una ventana en la zona inferior de la pantalla, donde deberemos agregar las animaciones según el estado del personaje (quieto, andando, salto…) para ello, nos dirigiremos a la zona inferior y agregaremos una animación clicando en la zona marcada en la captura

Le cambiaremos el nombre a nuestra preferencia y clicaremos en “Añadir fotograma desde archivo” y agregaremos todas las fotos de la animación que tengamos



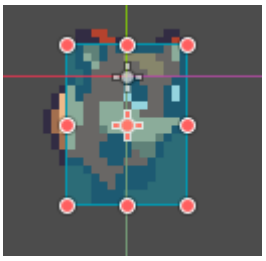
Al agregarlos todos, tendremos la animación creada y podremos verla ejecutarse con el botón marcado en la captura, también, es aconsejable ajustar los FPS a la cantidad de imágenes que tiene cada animación, para que vaya lo mas fluida posible



Y ya podremos hacer todas las animaciones que queramos con el mismo método

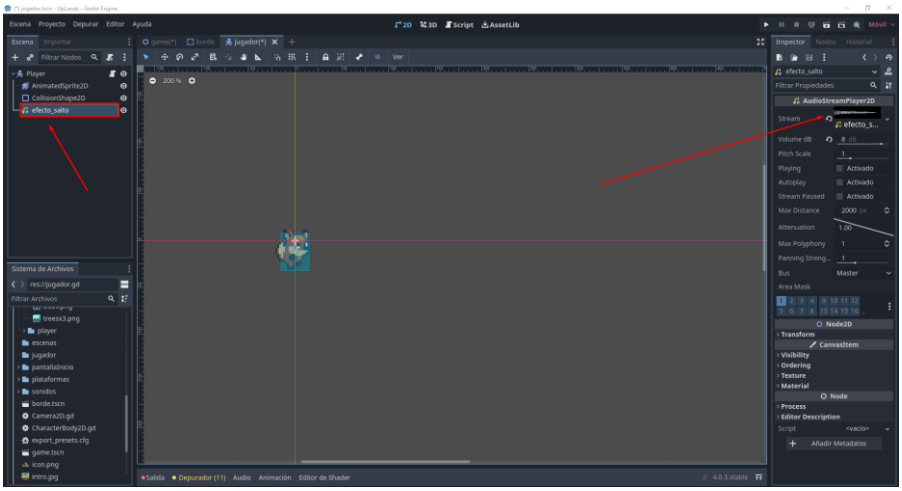
Subtarea 5.2: Colisión del personaje

Para crear dicha colisión, deberemos hacer como hemos visto anteriormente, deberemos crear una colisión que choque con las demás colisiones, con lo que al crear el nodo “CollisionShape2D” simplemente deberemos ajustar la colisión a nuestro personaje



Subtarea 5.3: Efecto de sonido en salto

Este nodo, lo utilizaremos para crear un efecto de sonido cuando nuestro personaje haga la función de saltar, con lo que simplemente agregaremos nuestro nodo “AudioStreamPlayer2D” y le agregaremos el sonido que queramos que tenga el personaje

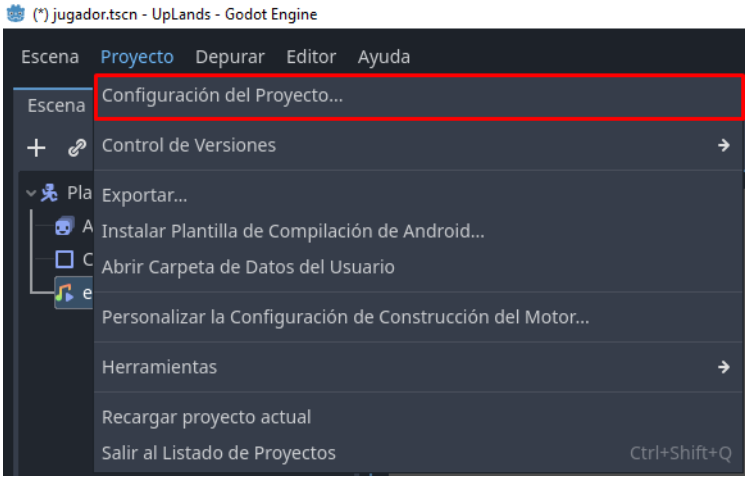


Subtarea 5.4: Controles

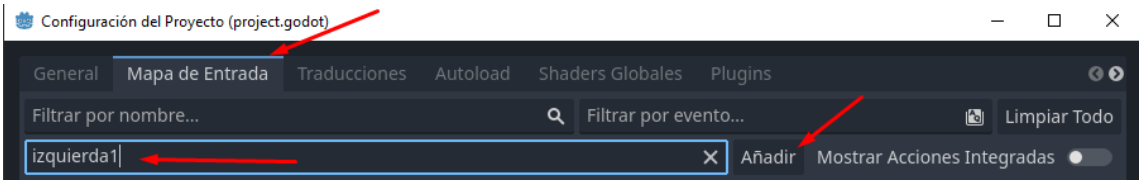
Hemos visto como agregar las animaciones, texturas, colisiones, efectos de salto, pero… ¿Cómo lo ejecutaremos?

Pues para ello, deberemos crear un script en nuestro nodo del jugador, pero antes vamos a asignar las teclas de los controles a nuestro juego

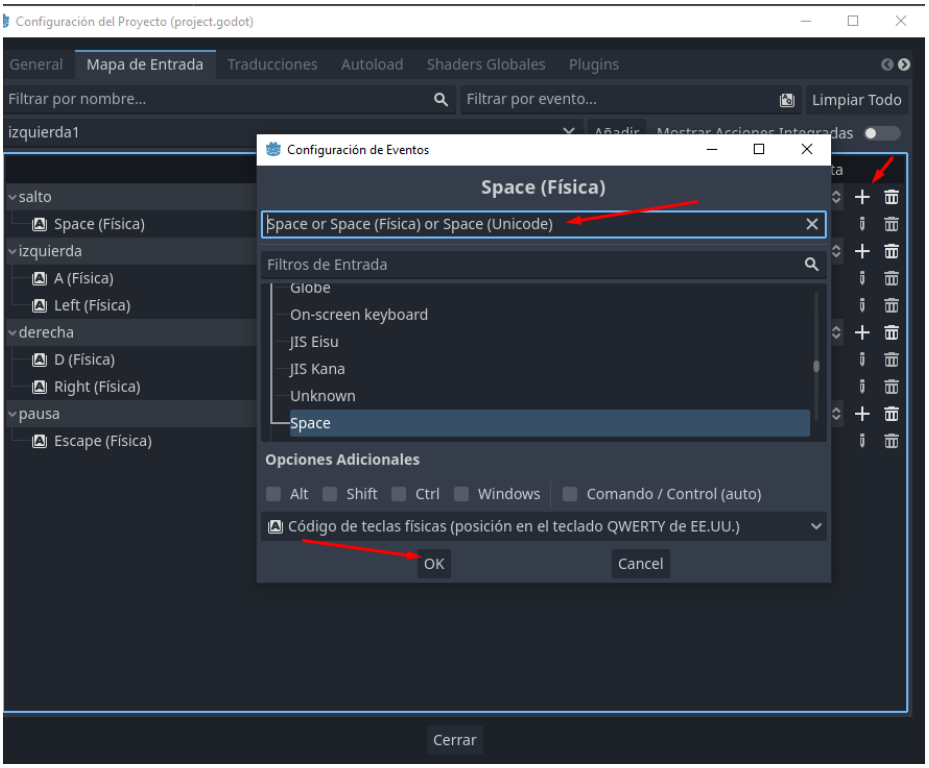
Para ello, deberemos irnos a la asignación de teclas en los ajustes de nuestro proyecto en la zona superior izquierda



Seguidamente, clicaremos en la pestaña “Mapa de entrada” donde agragaremos las acciones que queramos para nuestro juego



Al agregarla, clicaremos en el botón de “+” y clicaremos en la tecla que queramos asignar para dicha acción



Y ya tendremos los controles de nuestro juego asignados, pero ahora deberemos ejecutar esa acción en dicho script

Subtarea 5.5: Script jugador

Al crear nuestro script para el nodo “CharacterBody2D” deberemos tener en cuenta diferentes factores.

Tendremos que crear las funciones para controlar el movimiento del jugador, los efectos visuales, el salto, los efectos de sonido, y la gravedad del juego

Comenzaremos con la animación de estar quieto, la cual utilizaremos la función “Velocity” la cual nos sirve para controlar el movimiento tanto horizontal como vertical del jugador

En este caso, crearemos una función llamada “is_on_floor()” la cual se ejecuta al tener el nodo “characterBody2D” en colision, con lo que crearemos el siguiente código

```
if is_on_floor():
    if velocity.x == 0:
        $AnimatedSprite2D.play("quieto")
    else:
        $AnimatedSprite2D.play("andar")
else:
    if velocity.y <0 :
        $AnimatedSprite2D.play("jumpUP")
    else:
        $AnimatedSprite2D.play("jumpDown")
```

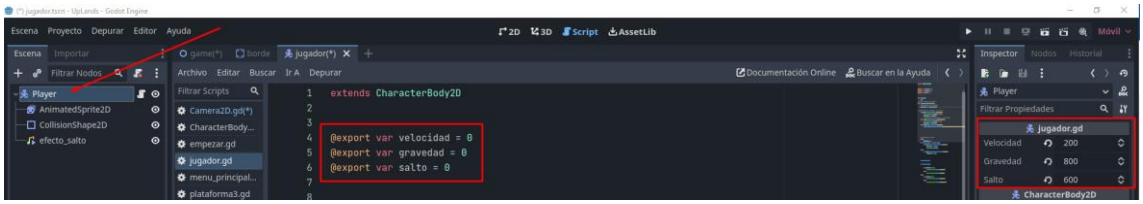
Crearemos un If que dice:

SI el jugador esta en el suelo y SI el movimiento es igual a 0 , ejecutará la animación “quieto” creada anteriormente

Si **NO** que se ejecute la animación de “andar”

Por ultimo, si el jugador NO esta en el suelo (entonces estaría en el aire), y la altura disminuya de 0, ejecute la animación de “JumpUp” y si no ejecute la animación de “JumpDown”

Ahora, exportaremos las variables de “velocidad, gravedad y salto” las cuales serán datos numéricos y con el añadido @export podremos controlar esas variables desde el nodo principal



Ahora ejecutaremos las funciones para el movimiento del personaje, para ello, utilizaremos la función “Input.is_action_pressed(accion):” para controlar el que hacer según el botón seleccionado

En este caso, utilizaremos el siguiente código:

```
if Input.is_action_pressed("derecha"):
    $AnimatedSprite2D.flip_h = false
    velocity.x = velocidad
    # Funcionalidad del movimiento a la izquierda
elif Input.is_action_pressed("izquierda"):
    $AnimatedSprite2D.flip_h = true
    velocity.x = -velocidad
```

Este código se define como:

SI la acción presionada por el jugador es “derecha” NO se invertirá la animación de andar y el personaje se moverá en el eje X hacia la derecha

Con un condicional en el control izquierdo, invertirá la animación de andar y el personaje se moverá en el eje X en negativo para ir a la izquierda

Si nada de esto se cumple, la velocidad en el eje X tendrá un valor de 0 con lo que estará quieto

```
else:
    velocity.x = 0
if is_on_floor() or is_on_wall():
    if Input.is_action_just_pressed("salto"):
        velocity.y = -salto
        $efecto_salto.play()
else:
    if Input.is_action_just_released("salto"):
        velocity.y += 200
```

A esto, agregaremos también la función para el salto con otro IF, en el cual en el siguiente condigo le estamos diciendo que si el jugador esta en el suelo o en un muro, pueda realizar la función del salto, agregando altura al personaje y el efecto de sonido creado anteriormente

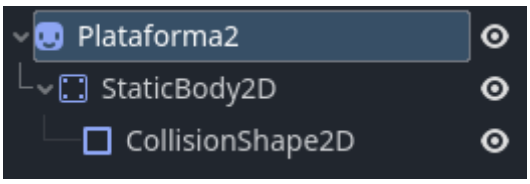
A esto también le diremos que si no esta en el suelo o en un muro y soltamos la tecla de salto, el jugador cancele el salto al instante y caiga

Esto hace que cuanto mas tiempo tengas presionado el botón de salto, más altura tendrás en el salto

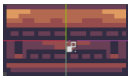
Tarea 6: Plataforma fija

Para Crear las plataformas fijas, deberemos crear una escena nueva con un nodo “Sprite2D” y ajustarle una colision para que el jugador choque con ella y por tanto, pueda subir

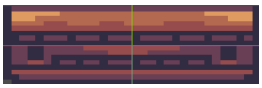
El esquema de nodos quedaría de la siguiente manera:



Utilizaremos los nodos ya vistos para lo siguiente:



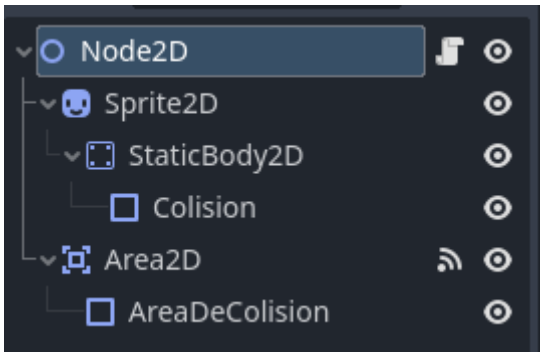
- **Sprite2d** → Asignar la textura de la plataforma
- **StaticBody2D** → Crea una colisión en la cual el jugador pueda chocar con ella
- **CollisionShape2D** → Nodo que crea una colisión ajustada a lo que necesitemos, en este caso, la plataforma



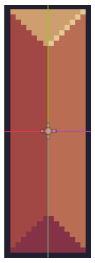
Tarea 7: Plataforma que desaparece

Esta plataforma es un poco mas compleja, ya que deberemos crear un script para controlar cuando desaparecerá y cuando aparecerá dicha plataforma

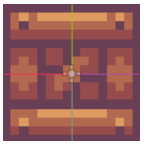
Para comenzar, deberemos crear el siguiente árbol de nodos en la escena creada para la plataforma:



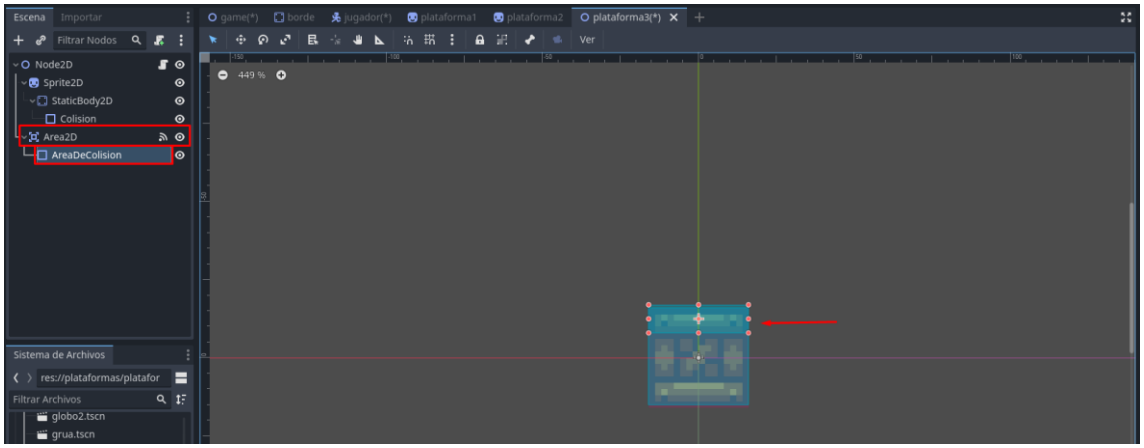
Utilizaremos los nodos ya vistos para lo siguiente:



- **Node2d** → Será el nodo encargado del script que utilizará diferentes funciones
- **Sprite2D** → Nodo encargado de asignar la textura de la plataforma
- **StaticBody2D** → Nodo encargado de ajustar una colision que choque con otras colisiones
- **CollisionShape2D**(Colision) → Nodo encargado de ajustar dicha colision que chocará
- **Area2d** → Nodo encargado de crear una colision que NO chocará con otras colisiones
- **Collisionshape2D**(**AreaDeColision**) → Colision que el jugador deberá tocar



Al crear la plataforma, deberemos ajustar la colision del área la cual el jugador tocará y actuará el script en su causa, en este caso, haciendo desaparecer la plataforma



Subtarea 7.1: Script plataforma

Deberemos crear un script para la plataforma, donde deberemos decir cuando desaparecer y aparecer, para ello, deberemos conectar un nodo del área de colision (la cual NO choca) con nuestro script.

Para conectarlo, deberemos clicar en nuestra area2D y dirigirnos a la pestaña nodos, situada en la zona superior derecha y conectar los nodos “body_entered” y “body_exited” con nuestro script en “node2d” y escribiremos lo siguiente:

```
func _on_area_2d_body_entered(body):
    await(get_tree().create_timer(0.2).timeout)
    $Sprite2D.visible = false
    $Sprite2D/StaticBody2D/Colision.disabled = true
```

Aquí estamos detectando cuando el jugador (en este caso) esta tocando el area2d.

Al tocar el área, estamos haciendo que nuestro código tenga un delay de 0.2 y seguidamente desaparezca, con lo que el jugador tiene que tener un tiempo de reacción perfecto para volver a saltar, ya que esta haciendo que el nodo “Sprite2D” desaparezca y la colision de desactive

```
func _on_area_2d_body_exited(body):
    await(get_tree().create_timer(1).timeout)
    $Sprite2D.visible = true
    $Sprite2D/StaticBody2D/Colision.disabled = false
```

Y con esta función, estamos haciendo que cuando no haya ninguna colisión tocando el área, la plataforma aparezca con 1 segundo de delay, haciendo la plataforma visible y activando de nuevo la colisión

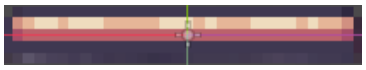
Tarea 8: Plataforma traspasable

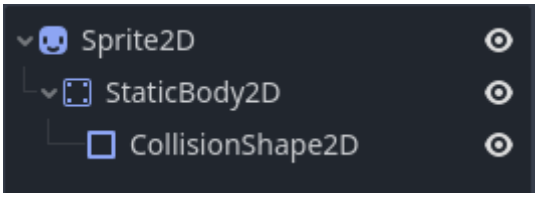
Para crear una plataforma traspasable (desde abajo en este caso) deberemos crear una plataforma normal, como hemos hecho anteriormente con los siguientes nodos:

Sprite2D → Nodo para agregar la textura a la plataforma

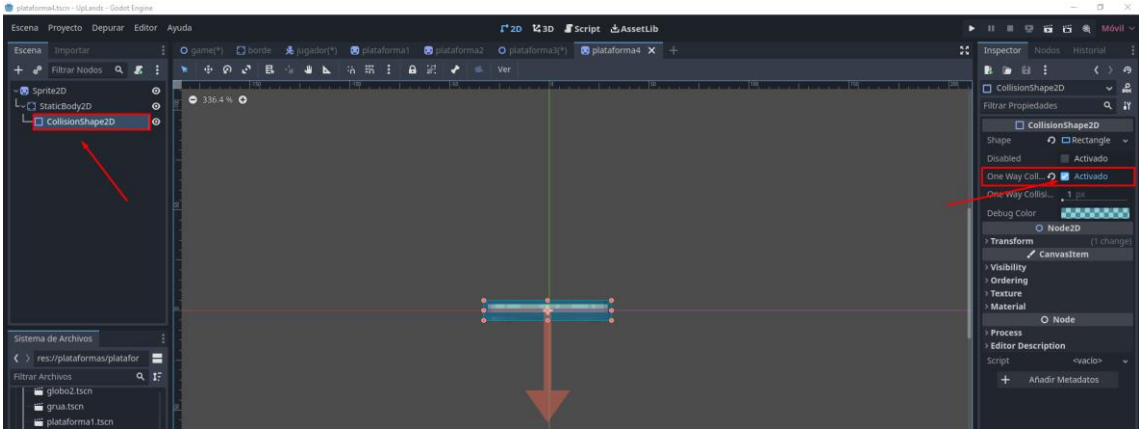
StaticBody2D → Nodo para agregar una colisión que choque con otras colisiones

CollisionShape2D → Nodo para ajustar dicha colisión





Al crearla, simplemente deberemos activar la opción “One Way Collision” en el nodo de “CollisionShape2D” la cual hace que la colisión solo actúe desde la dirección desde donde esté la flecha que aparecerá al activar la opcion



Tarea 9: Plataforma globo

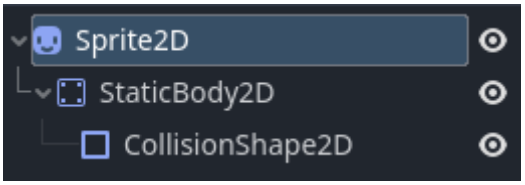
De nuevo, crearemos una plataforma base, la cual en este caso es un globo, pero tendrá la particularidad de que la colision será redonda, con lo que el jugador resbalará a no ser que caiga en el centro de dicho circulo

Para ello, deberemos hacer el mismo árbol de nodos que una plataforma normal:

Sprite2D → Nodo para agregar la textura a la plataforma

StaticBody2D → Nodo para agregar una colisión que choque con otras colisiones

CollisionShape2D → Nodo para ajustar dicha colisión



En este caso, ajustaremos el tipo de colision redondeada en el tipo de colision

Tarea 10: Plataforma móvil

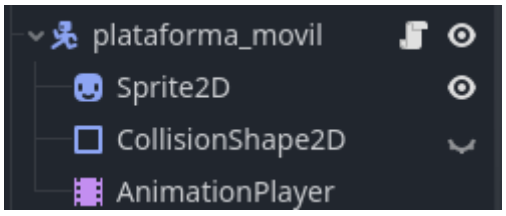
Para crear una plataforma en movimiento, deberemos utilizar el siguiente árbol de nodos, ya que es algo diferente a una plataforma normal:

CharacterBody2D → Nodo que utlizaremos para la creación de la plataforma

Sprite2D → Nodo para agregar la textura a la plataforma

CollisionShape2D → Nodo para ajustar dicha colisión

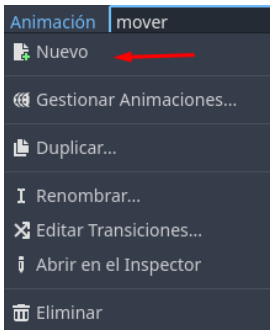
AnimationPlayer → Nodo que agregará la animación del movimiento



Importante* Esta plataforma deberemos hacerla en la escena del juego, ya que cada plataforma se deberá generar de manera individual, ya que cada una tendrá su movimiento especifico

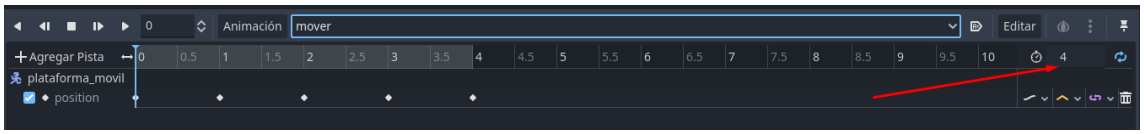
Crearemos dicho árbol de nodos, con su textura y colisión y nos dispondremos a crear la animación, la cual será el movimiento de dicha plataforma

Para crear ese movimiento, deberemos irnos al nodo de “AnimationPlayer” y nos aparecerá un menú en la parte inferior de la pantalla, en la cual, deberemos clicar en el botón de “Agregar pista” y seleccionando la opción de “Pista de propiedades” y por último, deberemos seleccionar el nodo de “CharacterBody2D” haciendo referencia al nodo padre de la plataforma.



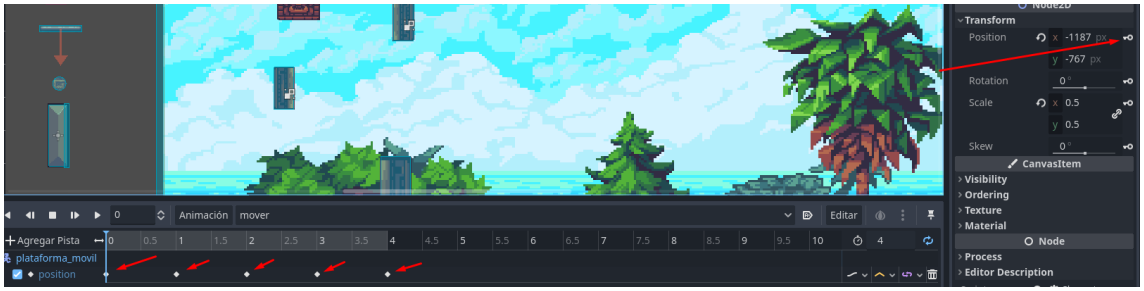
Ahora, clicaremos en el botón de “Animacion” y clicaremos en “Nuevo” para crear esa animacion

Seguidamente, ajustaremos la cantidad de segundos que tendrá la animación de la plataforma



Ahora, ajustaremos la posición de la plataforma, por cada segundo que pase en dicha animación

Por cada segundo que pase, deberemos tener una coordenada de la plataforma, y clicaremos en el botón marcado en la captura para marcar dicha coordenada

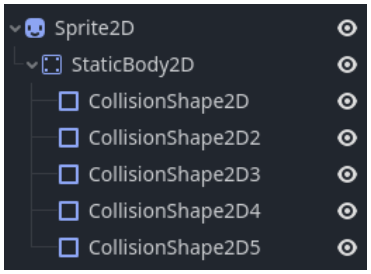


Y teniendo toda la animación creada, ya podremos utilizar la animación del movimiento con la plataforma creada

Tarea 11: Otras plataformas

Hay otras plataformas normales dentro del proyecto, las cuales están agregadas de manera individual y sin ninguna especialidad

- **Plataforma avión:**



Sprite2D → nodo de textura

StaticBody2D → Nodo de colisión que choca

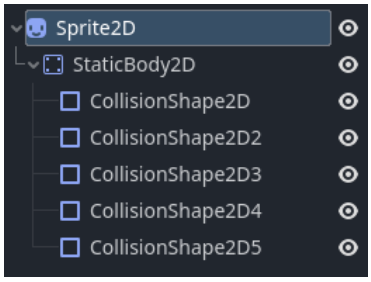
CollisionBody2D → Ajuste de colisión

- **Plataforma Globo2:**

Sprite2D → nodo de textura

StaticBody2D → Nodo de colisión que choca

CollisionBody2D → Ajuste de colisión

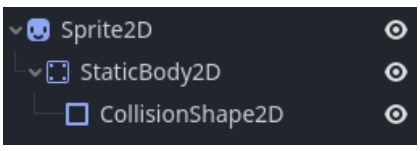


- **Plataforma 5:**

Sprite2D → nodo de textura

StaticBody2D → Nodo de colisión que choca

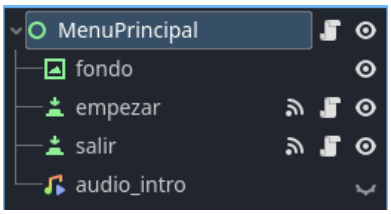
CollisionBody2D → Ajuste de colisión



Tarea 12: Menú principal

Para crear el menú principal deberemos crear una escena la cual contenga un nodo de interfaz de usuario, en la cual agregaremos una imagen de fondo y dos botones para poder jugar o salir del juego junto con la canción de fondo del menú

El árbol de nodos debería quedar de la siguiente manera:



- **Interfaz de usuario:** Nodo que controla todo el menu
- **TextureRect**: Agrega una imagen para el fondo del menú principal
- **Buttom**: Agrega un nodo de botón para ejecutar diferentes funciones
- **AudioStreamPlayer2D**: Nodo para agregar música de fondo

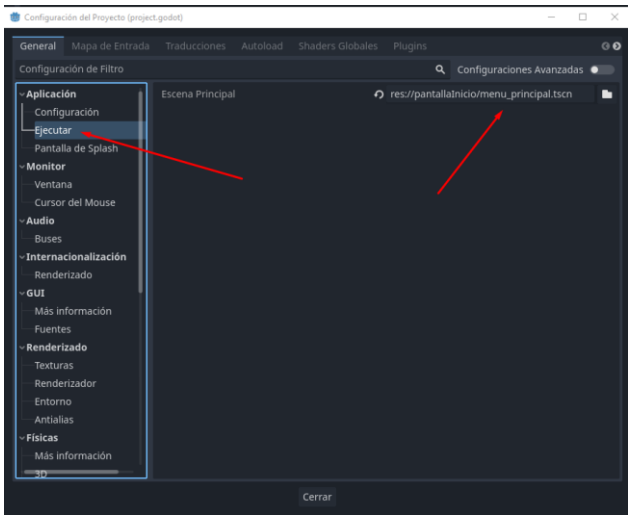
Subtarea 12.1: Música del menú

Para ejecutar la música de fondo, deberemos crear un script en el nodo principal donde escribiremos el siguiente código para que la música se ejecute cuando la escena del menú este en ejecución

```
func _ready():
    $audio_intro.play()
```

Subtarea 12.2: Botón Iniciar partida

Antes de configurar el botón de iniciar el juego, deberemos ajustar el proyecto, para que la ventana principal del juego sea la del menu, para ello, deberemos irnos a la configuración del proyecto, y ajustar la escena principal a la del menu principal



Tras eso, podremos configurar el botón de “Iniciar partida” haciendo un script en dicho botón para que la partida comience y se cambie de escena

```
func _on_pressed():
    get_tree().change_scene_to_file("res://game.tscn")
```

Con este código, estamos haciendo que el al clicar en el botón, se cambie la escena

Subtarea 12.3: botón Salir del juego

Este botón cerrará el juego al ser clicado, con lo que de nuevo, deberemos crearle un script para que el juego se cierre al clicar el botón

```
func _on_pressed():
    get_tree().quit()
```

Tarea 13: Otros

También he añadido un ultimo control al juego que permite volver al menú principal estando jugando para que el jugador pueda salir del juego

Para ello, nos dirigimos al script del jugador y agregamos la siguiente línea

```
if Input.is_action_pressed("derecha"):
    $AnimatedSprite2D.flip_h = false
    velocity.x = velocidad
    # Funcionalidad del movimiento a la izquierda
elif Input.is_action_pressed("izquierda"):
    $AnimatedSprite2D.flip_h = true
    velocity.x = -velocidad
elif Input.is_action_pressed("pausa"):
    get_tree().change_scene_to_file("res://pantallaInicio/menu_principal.tscn")
```

Cuando pulsemos el botón de “pausa” nos cambiará la escena al menu principal del juego

7. Bibliografía

Cursos Obtenidos:

- https://www.udemy.com/course/godot-3-primer-videojuego/learn/lecture/19635248?start=1380#overview

- https://www.udemy.com/course/curso-basico-de-godot-2d/learn/lecture/35234462?start=480#overview

- https://www.udemy.com/course/curso-basico-de-godot-2d-2aparte/learn/lecture/35794818?start=30#overview

- https://www.udemy.com/course/curso-basico-de-godot-2d-3aparte/learn/lecture/36003928?start=630#overview

- https://www.udemy.com/course/curso-basico-de-godot-2d-4aparte/learn/lecture/36308760?start=15#overview

- https://www.udemy.com/course/curso-basico-de-godot-2d-5aparte/learn/lecture/36995396?start=15#overview

- https://www.youtube.com/watch?v=wN2QUQ8BJCw&list=PL-L2FQ9FXTPV4vYq-NmARUHZVZ7ICKL0j&index=1&t=157s

- https://www.youtube.com/watch?v=a5cPaRsM6sU&list=PL-L2FQ9FXTPV4vYq-NmARUHZVZ7ICKL0j&index=2&t=572s

- https://www.youtube.com/watch?v=nC9LUawEeMY&list=PL-L2FQ9FXTPV4vYq-NmARUHZVZ7ICKL0j&index=3

8. Cronograma

Cronograma