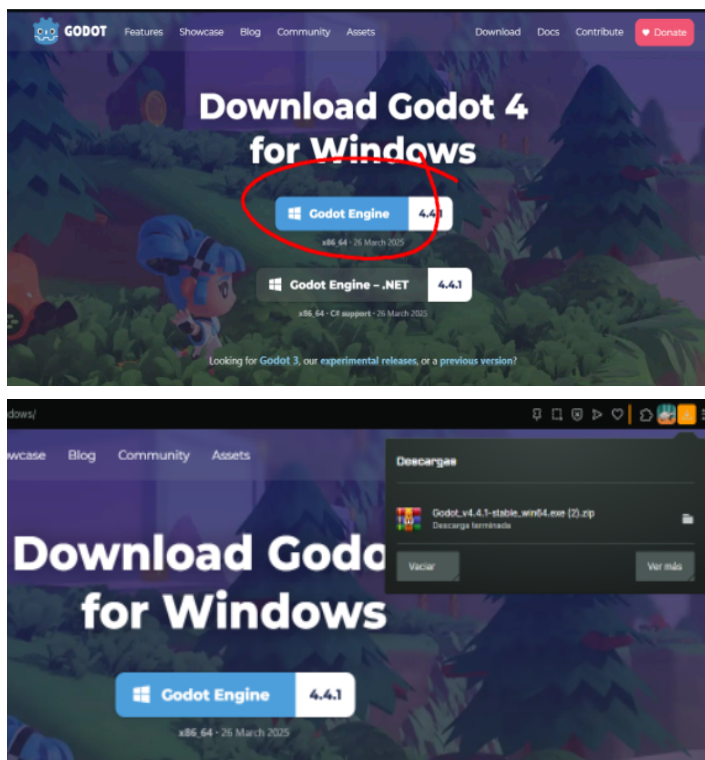


SpaceShip 2D – Tutorial patrocinado por Fabricio Mejías y creado con Godot 4.4.1.

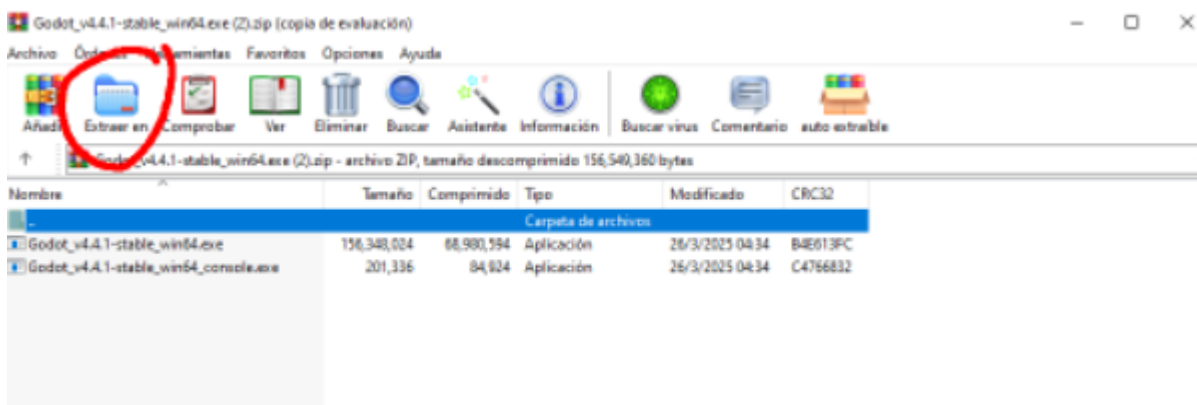
Escrito paso por paso a lo criollo de cómo lo fui entendiendo

Descargar Godot 4.4.1 -> <https://godotengine.org/download/windows/>

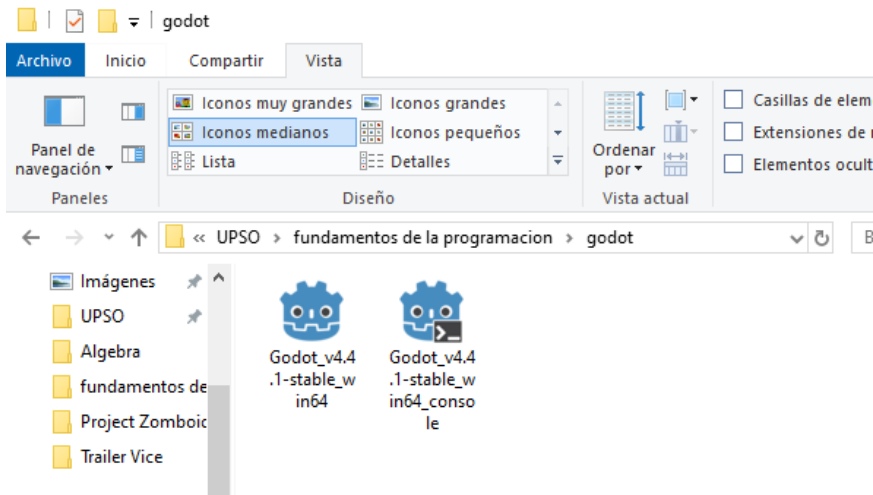
Este link te lleva a la pag de Godot para descargar su ultima verison (4.4.1), al hacer click en ese botón te debería empezar a descargar un archivo .zip (archivo comprimido) que tenes que abrir con el programa [winrar](#) o cualquier otro programa que te permita descomprimir el archivo. **Normalmente el archivo lo podes abrir desde el navegador, si no lo encontras puede llegar a estar en la carpeta descarga de tu computadora (puede variar según el tipo de Windows, nose)**



Una vez abierto lo tienen que descomprimir en algún lado, **una carpeta nueva en el escritorio, documentos, descargas, donde quieran.**

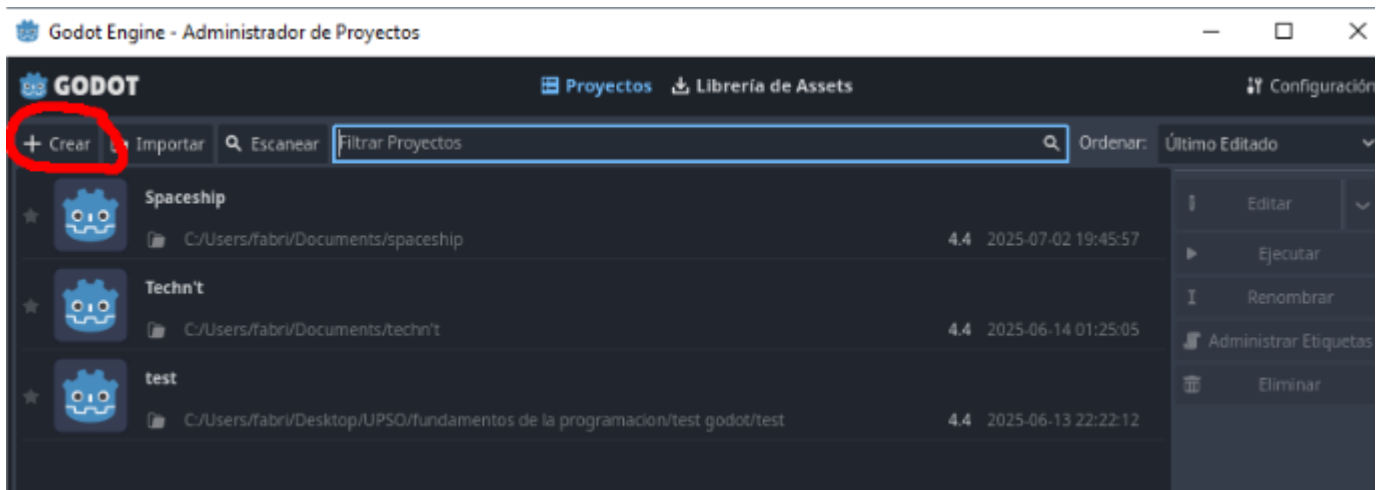


Les debería quedar asi

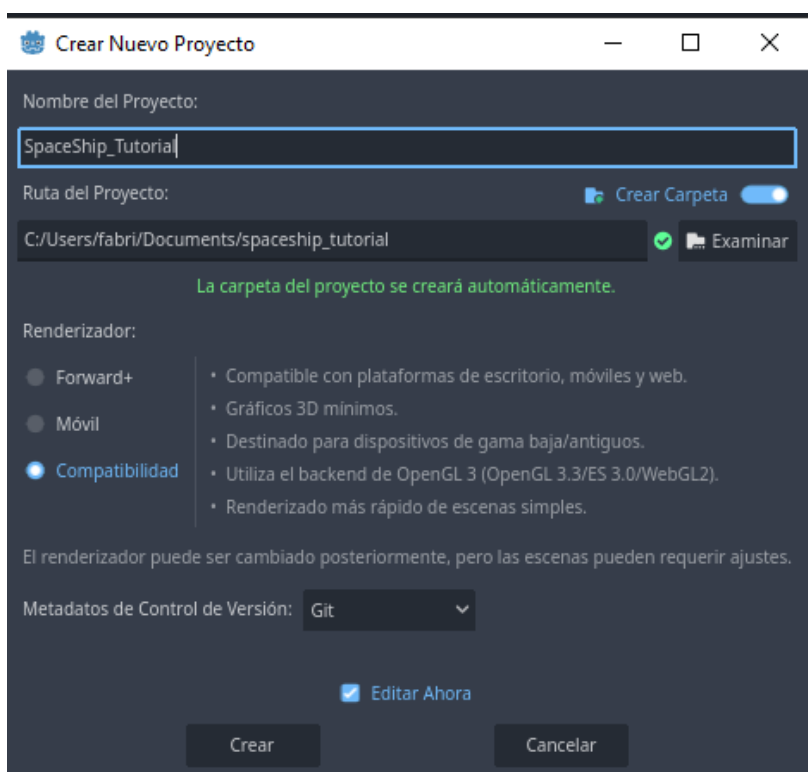


El que van a usar es **godot_v4.4.1-stable_win64** o parecido, el otro nose que es, debe ser para abrir la consola ni idea no probe.

Cuando lo abren les debería aparecer algo parecido a esto



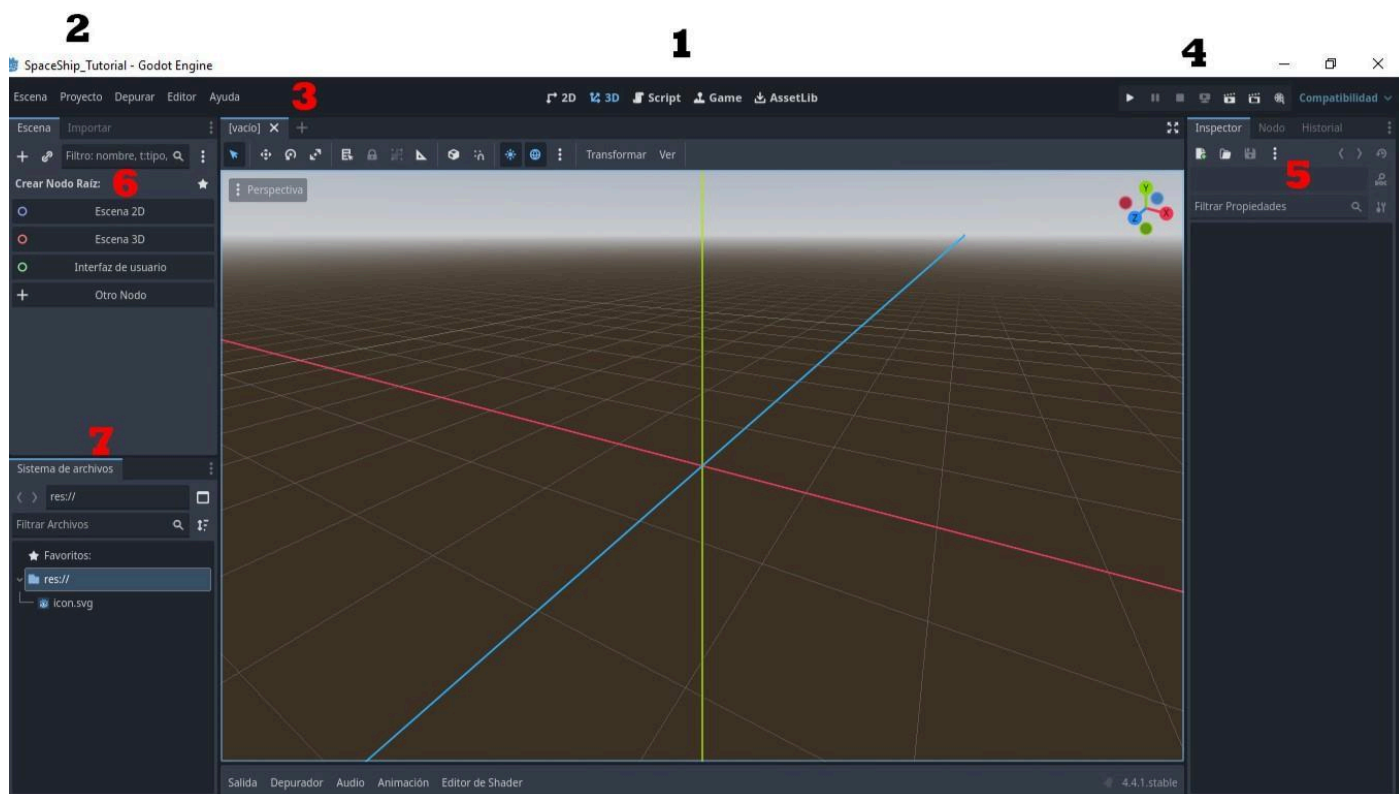
A mi me salen los 3 proyectos que fui armando en estos días, ustedes van a ir a **Crear**



<- Nombre del proyecto

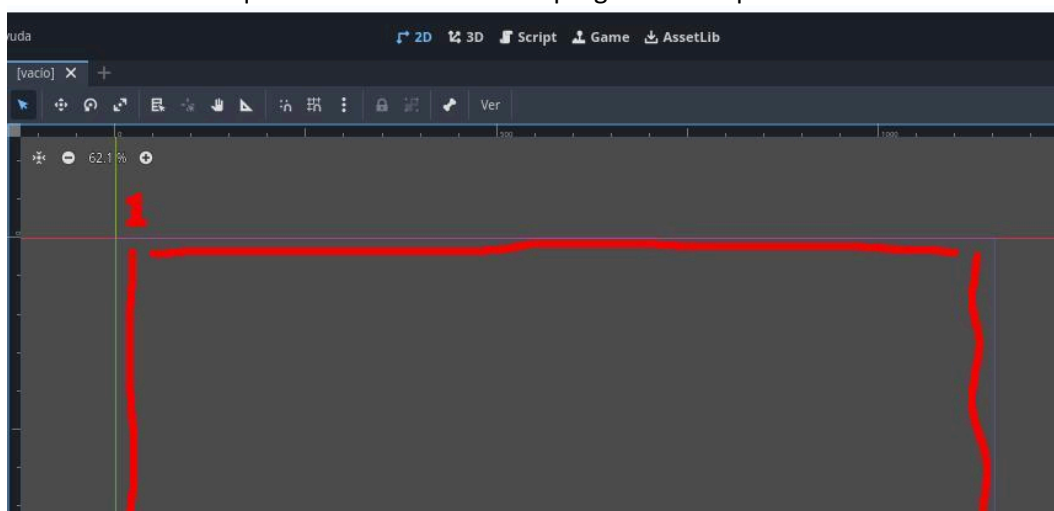
<- Ruta del proyecto, donde se van a guardar las cosas del proyecto, se puede cambiar haciendo click en "examinar" y elegir la ubicación.

El resto no toque nada, al final tocas **Crear** Ahí abajo para arrancar con el proyecto



Bienvenido a la interfaz del programa, ahora te explico mas o menos que significada cada una de las cosas que ves ahí.

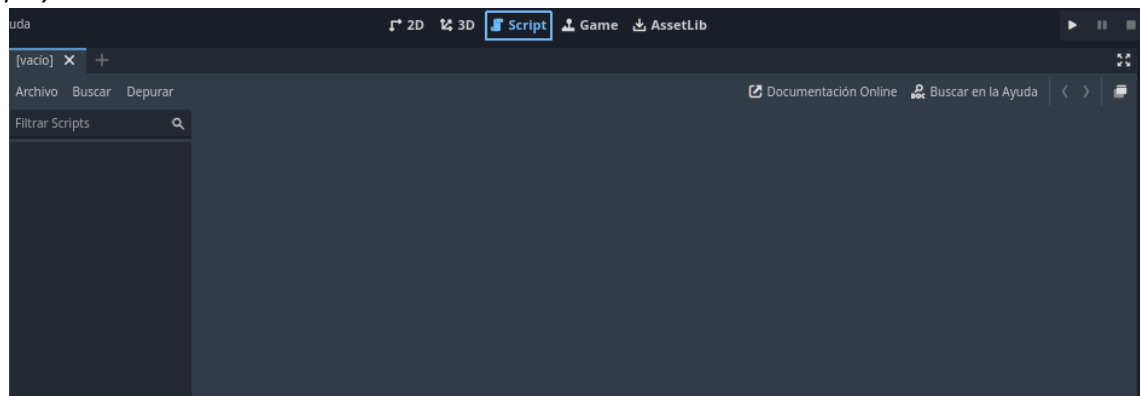
- 1.) Estas son las vistas que vas a tener dentro del programa. Las que nos van a interesar son las **2D** y **Script**



Vista 2D, nos va a poner en un plano 2D, con las líneas roja y verde que muestran los ejes X e Y respectivamente y esa cuadrado rojo que se forma con color medio violeta es el **Viewport** y el numero 1 es la coordenada (0,0) de la escena.

¿Que es el Viewport? Es lo que se va a ver cuando le pongas play a la escena. Eso tiene un ancho y un largo que se puede configurar mas adelante en otro lugar.

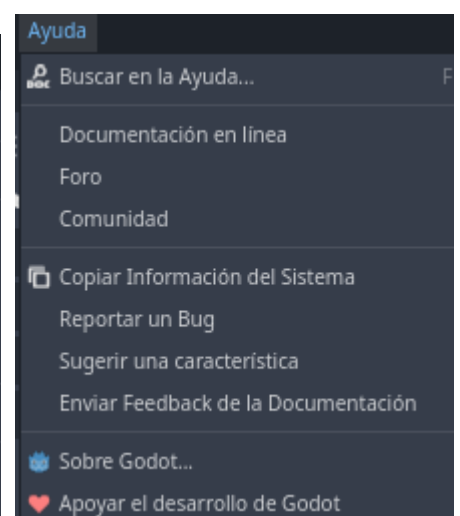
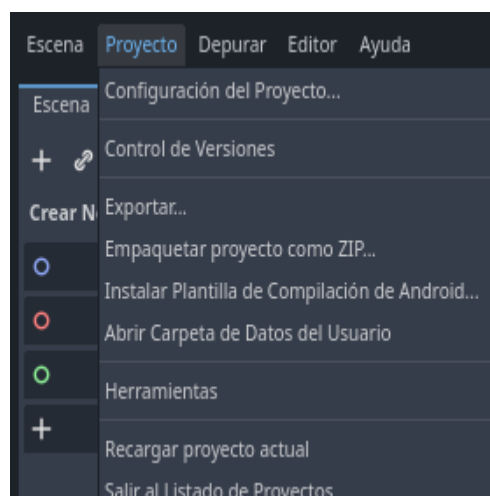
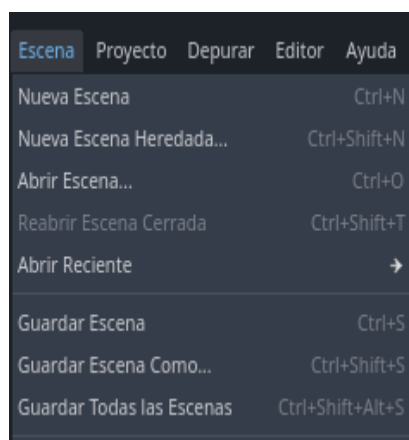
Que es una **escena**? Puede ser todo, una nave, un enemigo, una barrera invisible, el laser que disparan las naves, el terreno que se va a ver de fondo, etc. Y también va a ser lo que se “reproduce” cuando le demos play.



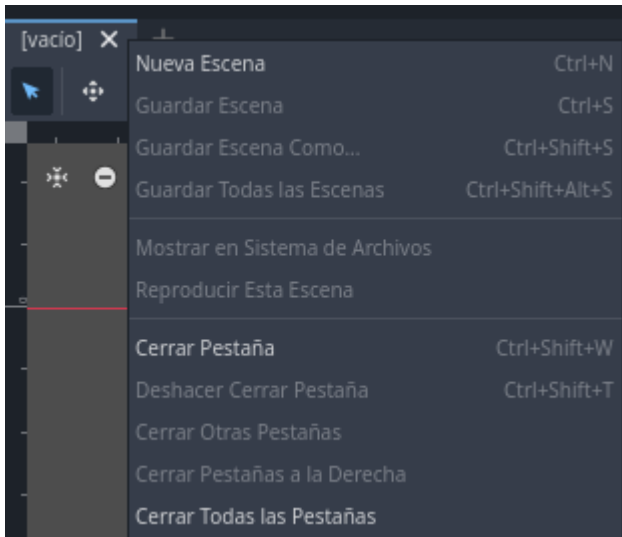
Y esta es la **vista de script**, no hay nada todavía porque no hay scripts pero después les muestro como queda cuando lo van llenando.

Pueden **cambiar de vista** tocando ahí arriba los botones **2D** y **script**, a mi me costó un toque darme cuenta de eso.

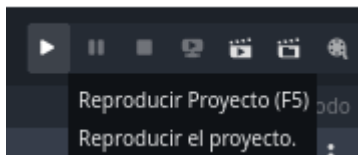
2) Acá está para configurar las diferentes cosas del **proyecto o la escena**, no vi que es depurar, ni editor, en **ayuda** pueden encontrar link a la documentación para aprender un poco más sobre las diferentes cosas del motor o de los scripts.



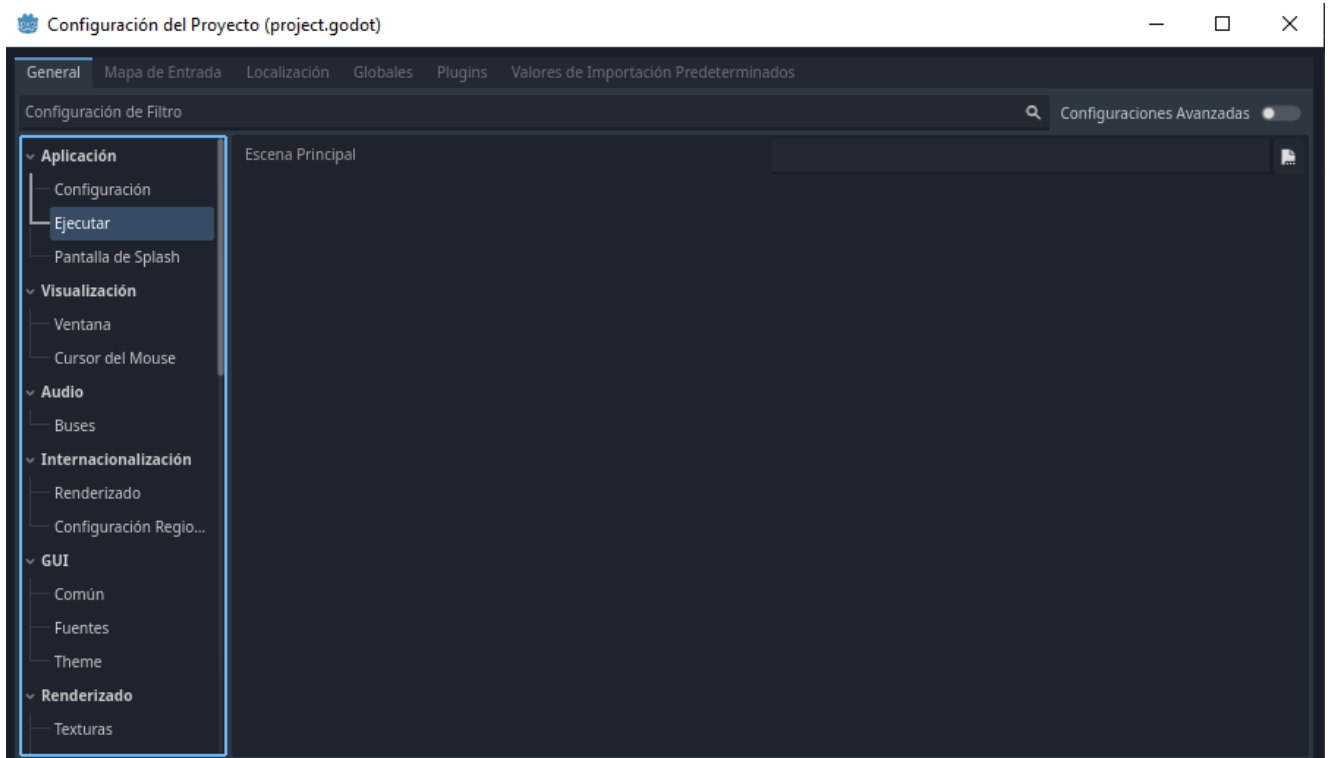
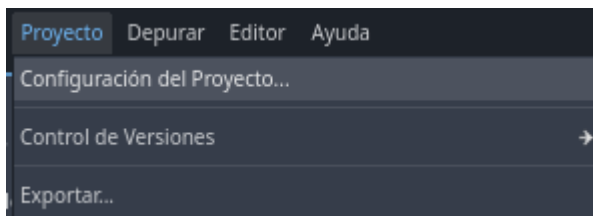
3) Aca pueden ir las **escenas** que vayamos creando para tener un acceso más rápido, **se puede agregar una escena rápidamente tocando el “+”** y **haciendo clic derecho sobre las escenas podemos tener diferentes opciones**, como por ejemplo reproducir solo esa escena (para probar algo dentro de esa escena)



4) Ahí arriba a la derecha están las opciones para reproducir la **escena principal** y otras escenas.

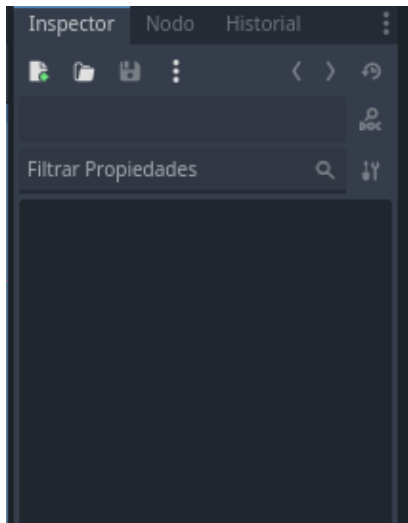


¿Que es la **escena principal**? Es la escena que se va a reproducir **primero que todas**. Se puede configurar en la configuración de proyecto (2)



Menu de configuración del proyecto, acá pueden cambiar varias cosas como la **escena principal** en “Aplicación -> ejecutar” o el tamaño del **Viewport** en “visualización -> ventana”

5) Esto es el inspector, es una vista muy útil, acá te muestra todas las **PROPIEDADES** que van a ir teniendo los **nodos** que seleccionemos en 6), también está la **pestaña “nodo”** donde puedes utilizar “**señales**” para ayudarte a escribir el script. (no confundirse con lo que es un nodo en si, acá muestra otras propiedades de los nodos)



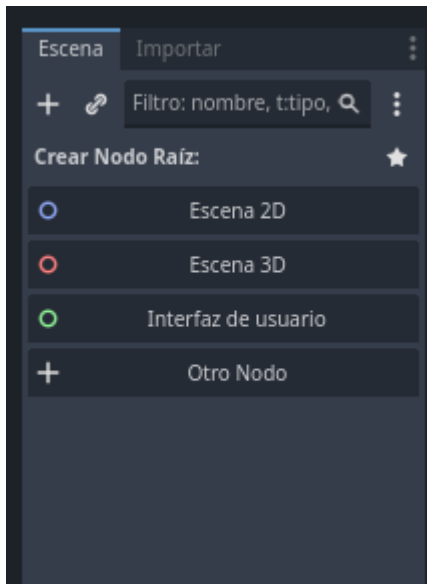
Ahora esta vacío porque no tengo ningún nodo seleccionado.

¿Que son las propiedades? Van a ser elementos que modifican al **nodo** de alguna forma, por ejemplo, el más común es “transform” que es básicamente donde está el nodo, que escala va a tener o rotación dentro de la escena.

¿Que son los nodos? Los nodos son los **componentes básicos dentro de cada escena**, que quiere decir esto, quiere decir que **cada escena va a tener nodos con sus propias características y propiedades**. Imagínate una escena “jugador” ¿de que está compuesta esa escena jugador? Puede tener un **nodo de textura** que le da una imagen, otro **nodo colisión** que le da una colisión para que pueda interactuar con el entorno, puede tener un **nodo vacío** que contenga un **script**, etc.

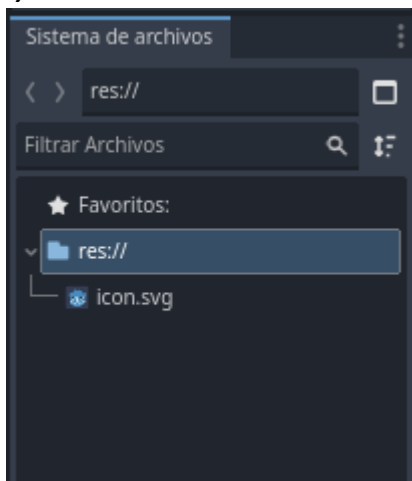
¿Qué son las señales? Son ayudas a la hora de escribir un script, por ejemplo un botón va a tener una señal de “cuando me presionan” que te va a pedir que lo unas a un script, esto va a hacer que en ese script se cree una nueva línea de código que se va a ejecutar cuando ese botón sea presionado. Mas adelante pongo ejemplos de su uso.

6) Acá van a ir todos los nodos de la escena



Quando creas una nueva escena te pide crear un nodo raíz, es como el nodo principal de la escena. Esas 3 opciones son para facilitar la creación de la escena que quieras hacer. Por ejemplo, un menú principal sería una interfaz de usuario, donde te ayuda a buscar botones y texto para agregar. Si quieres crear una escena jugador vas a crear una escena 2D donde te facilitan los nodos que puedes llegar a usar, como sprite para darle imagen por ejemplo.

7) Por ultimo aca te va a mostrar los **recursos** del proyecto, ya sean escenas, imágenes, sonidos, etc.



Estos **recursos** los puedes agregar arrastrando las cosas en esta zona o ir a la ruta del proyecto (que la definís cuando creas el proyecto la primera vez) y ahí lo acomodas.

Bueno ahora que mas o menos explique cada zona de la interfaz empezamos a armar el videojuego.

Para crear el videojuego tengo que tener en claro de que esta compuesto el videojuego que en este caso es un SpaceShip.

Voy a necesitar:

- Una nave que el jugador pueda controlar
- Enemigos u obstáculos
- Un nivel donde se encuentre todo eso.
- Condición de victoria y derrota

Eso sería así por arriba como para tener algo básico donde después puedo ampliar

Primero vamos por el jugador, pero pueden empezar por donde quieran.


Necesito saber cómo va a controlar el jugador su nave, de que va a ser capas esa nave, que forma o textura va a tener. Todo eso como mínimo.

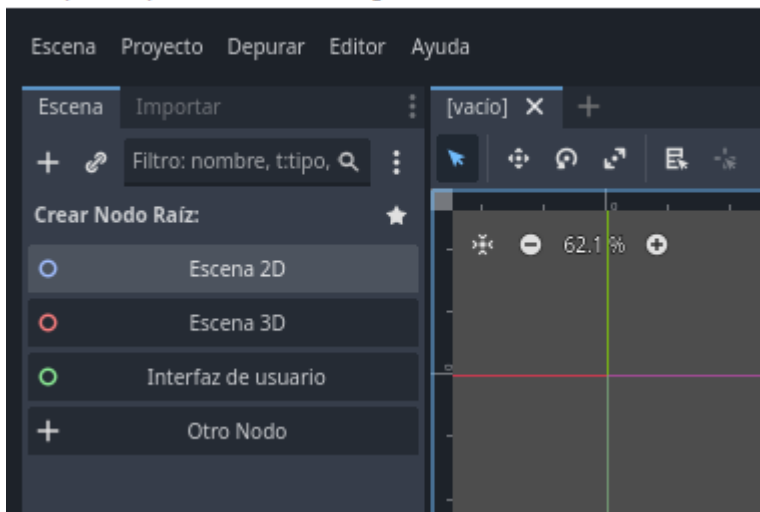
Yo primero me arme un boceto a lo art attack para tener una idea de como lo quería.



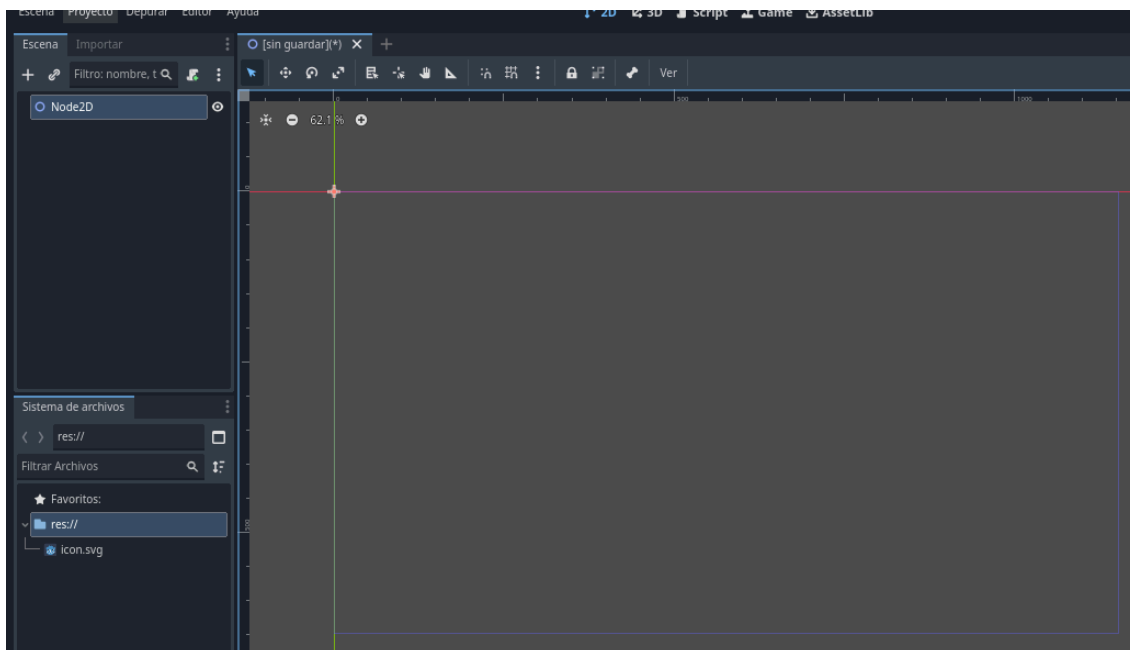
Entonces mi jugador apunta hacia arriba, se puede mover en 4 direcciones, puede disparar, va a tener una textura de nave (algo que apunte hacia arriba)

Con eso básico en mente me voy a godot.

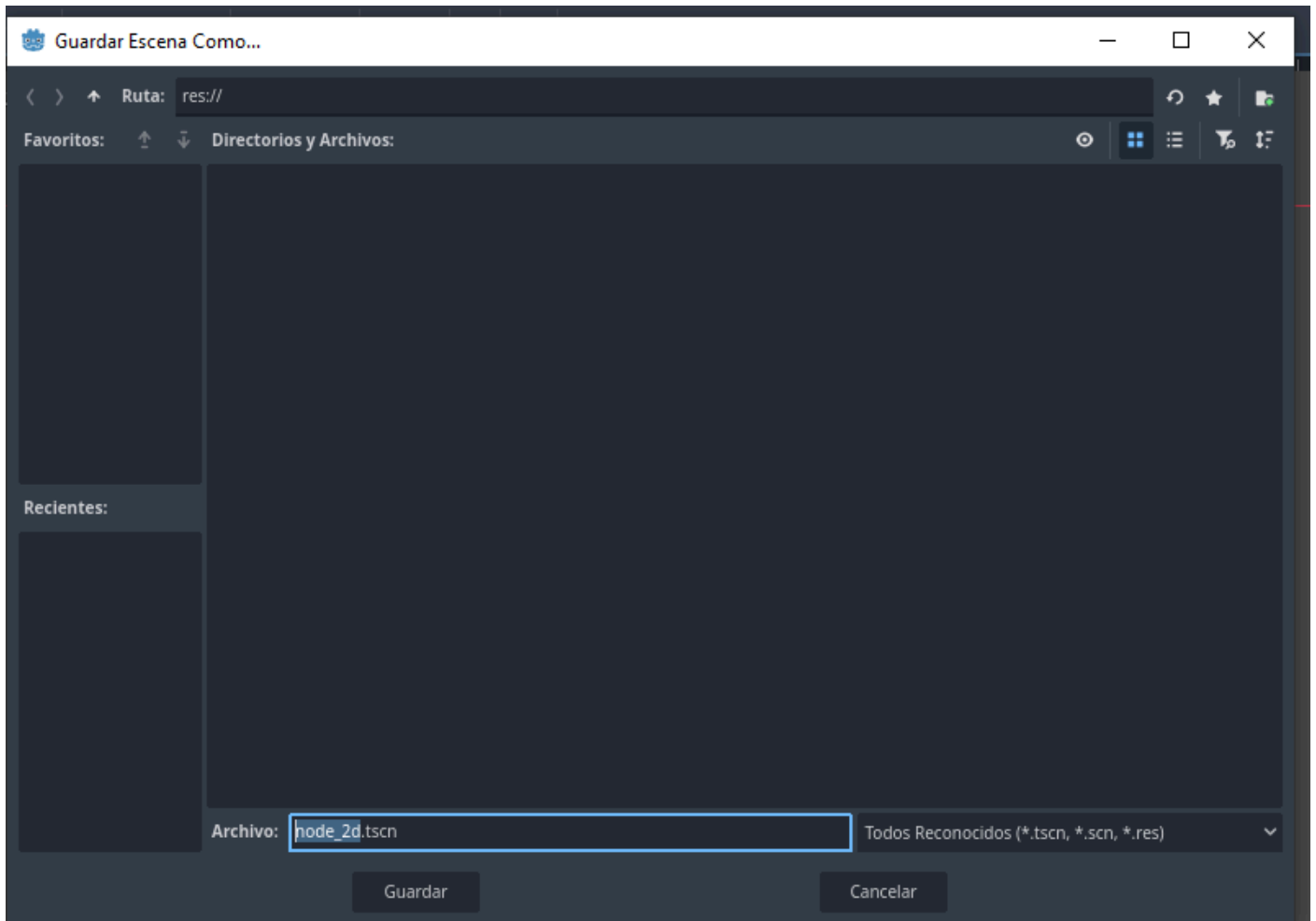
 SpaceShip_Tutorial - Godot Engine



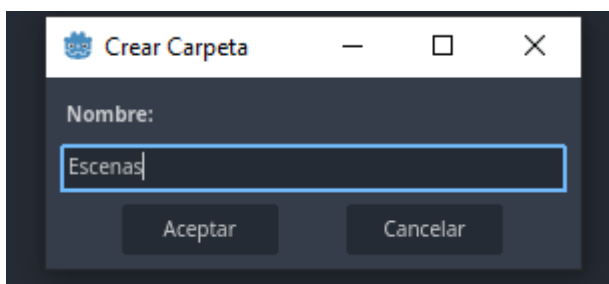
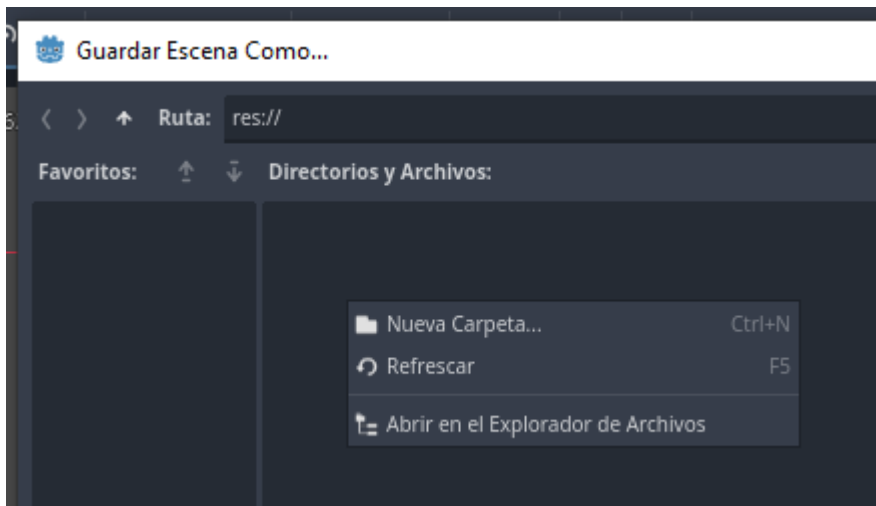
Creo una escena 2D, porque mi nave de jugador es 2D




Me crea la escena y me agrega un nodo 2D. y ahora yo hago algo para mantener un orden dentro del programa. Con Control+S podés guardar la escena actual o puedes hacer clic derecho en "sin guardar" y te da la opción de guardar. La primera vez que creas una escena te va a pedir que la nombres cuando la guardas

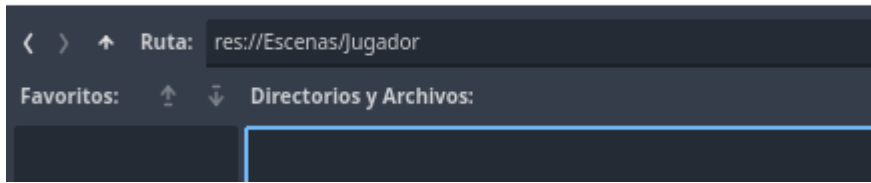


Haciendo clic derecho en medio del cuadro me da la opción de crear una nueva carpeta

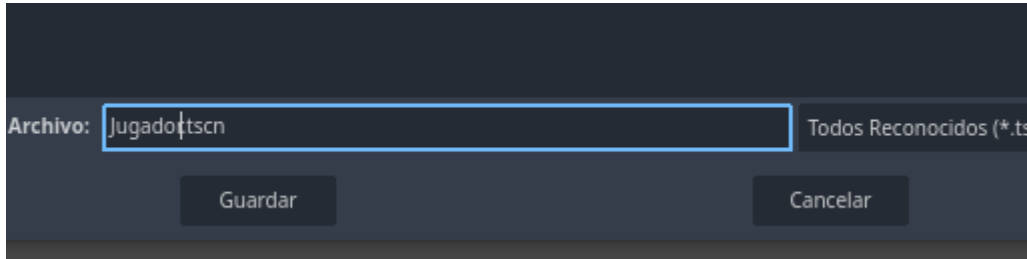


Yo creo una carpeta llamada escenas y nuevamente creo otra carpeta con el nombre de la escena que voy a guardar, en este caso se va a llamar Jugador, Una vez cree esas 2 carpetas(que se puede ver que son 2 en la ruta)

 Guardar Escena Como...



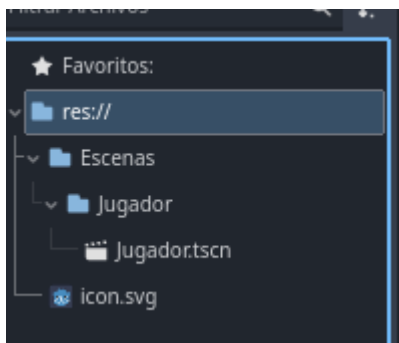
Ahí recién guardo la escena con el nombre Jugador de nuevo.



¿Que hice entonces? Cree una carpeta llamada Escenas, luego dentro de esa carpeta cree otra carpeta llamada Jugador y dentro de esa ultima carpeta guarde la escena.

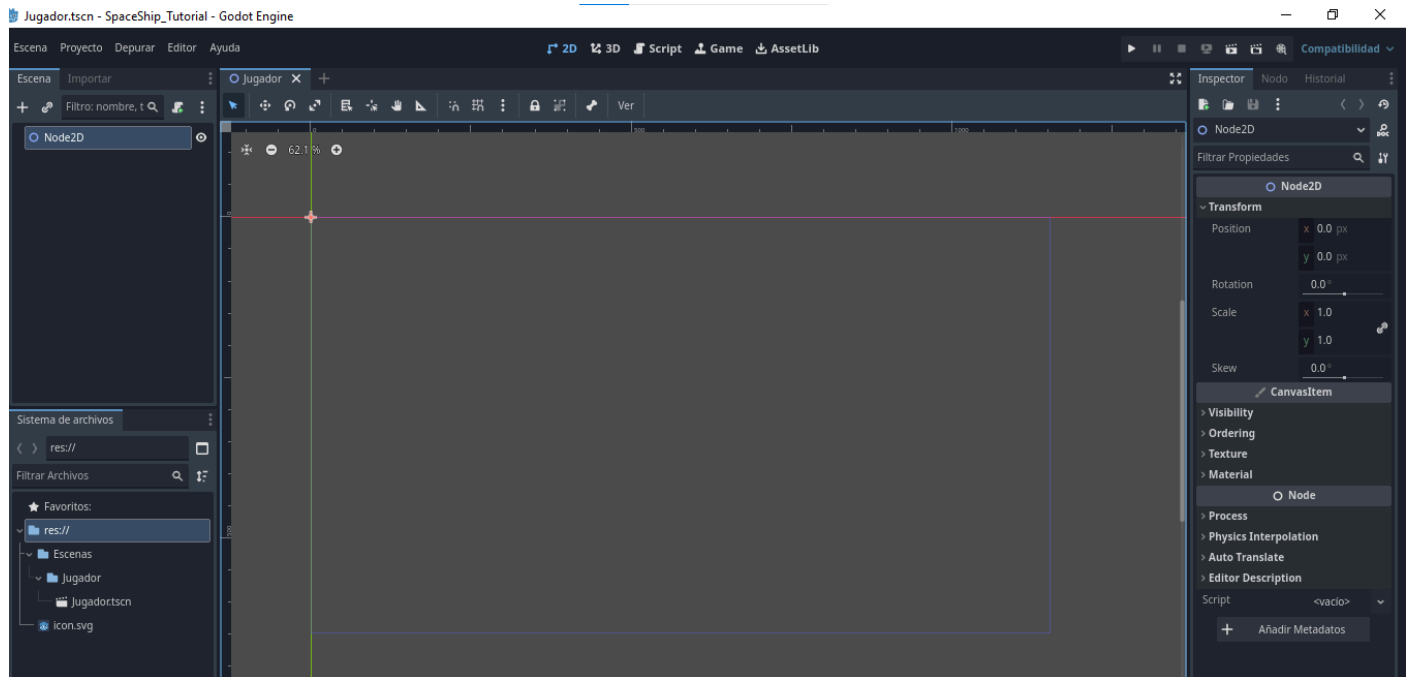
¿Por qué? Porque soy medio esquizofrénico, pero además porque ayuda mas adelante cuando tenemos 200 escenas con diferentes nombres y diferentes texturas y diferentes scripts. Entonces empezando a guardar cada escena dentro de su propia carpeta es mucho mas ordenado a la hora de ponerle imágenes o scripts.

Nota: la terminación .tscn del archivo quiere decir que ese archivo es una escena de godot. Esto va a servir mas adelante cuando queramos cambiar de escena pero esta bueno tenerlo en cuenta ahora.



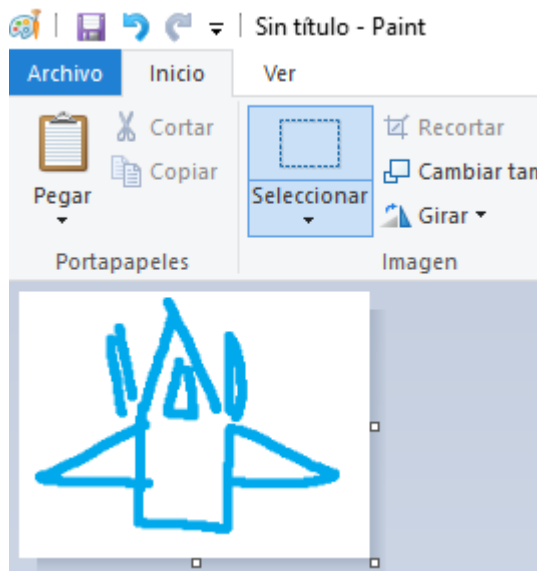
Les debería quedar asi abajo a la izquierda, donde van los recursos.

Ahora tenemos una escena llamada Jugador que tiene un nodo principal o raíz llamado Node2D



Si se fijan en el inspector ahora salen varias cosas, esas son las propiedades del nodo "Node2D" pueden hacer clic en transform por ejemplo y les muestra las propiedades en este caso de transform. Como la posición, rotación, etc. Otro dato, si quieren pueden cambiar esos valores en el inspector y se ven reflejados en la escena, en este caso no hay nada todavía pero ahora con una textura lo van a ver mas facil

Habíamos dicho que la escena jugador tenía varias cosas, voy a empezar por la mas visual, la textura. Como es un juego 2D solo necesito una imagen, voy a usar mi super sketch de art attack para que vean que se puede utilizar cualquier cosa. Pero mas adelante vamos a ir a itch.io para aprender a buscar "assets" mas bonitos



Me lo arme en Paint, me guarde la imagen y ahora es la parte importante. Tengo que llevarla a la carpeta de recursos del proyecto, ahí tienen varias formas de llevarla.

Yo había guardado mi Paint en el escritorio y lo arrastre hasta la carpeta de Jugador dentro de godot. Pueden abrir la carpeta del proyecto de donde sean que la hayan creado y dentro de esa carpeta les debería salir la carpeta Escena y dentro de esa la carpeta jugador.

En mi caso los proyectos se guardan dentro de la carpeta Documentos. Entonces puedo ir a la carpeta documentos, buscar spaceship_tutorial, o el nombre que le hayan puesto. Y ahí dentro deberían ver esto

PROJECT

- .godot
- Escenas
- .gitattributes
- .gitignore
- icon
- icon.svg.import
- project.godot

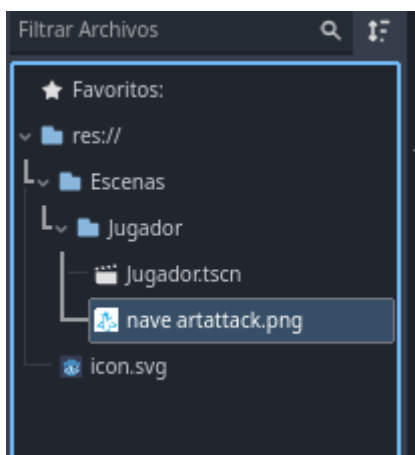
De ahí entran a escena.

Nombre	Fecha de modificación	Tipo	Tamaño
Jugador	3/7/2025 02:34	Carpeta de archivos	

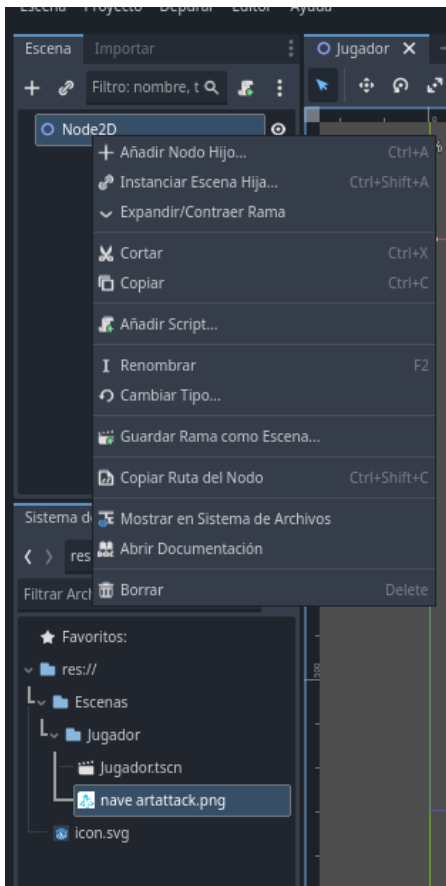
Luego entran a la carpeta Jugador.

Nombre	Fecha de modificación	Tipo	Tamaño
Jugador.tscn	3/7/2025 02:34	Archivo TSCN	
nave artattack	3/7/2025 02:32	Archivo PNG	
nave artattack.png.import	3/7/2025 02:32	Archivo IMPORT	

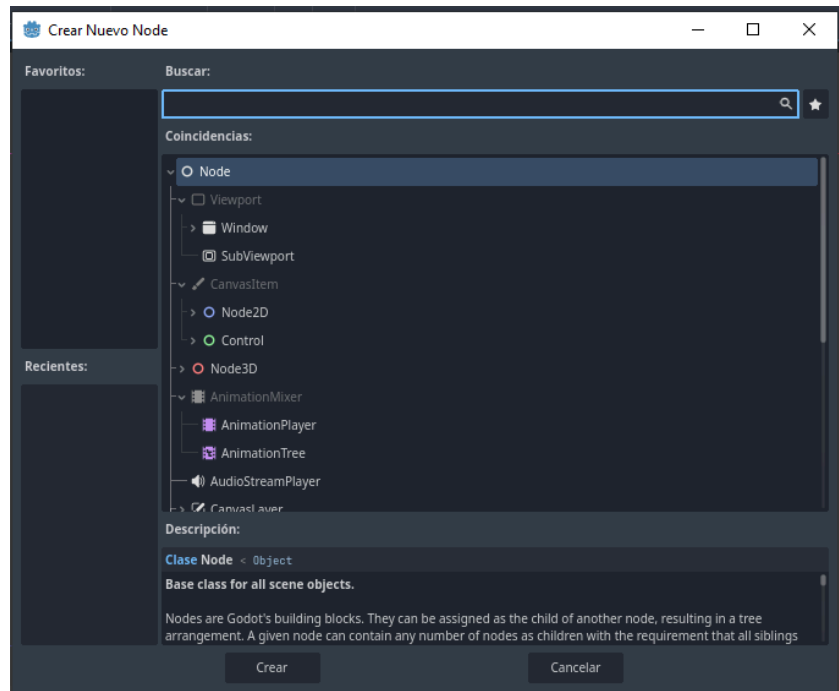
Y bueno en mi caso ya había subido la imagen, pero sino subieron nada solo deberían ver “jugador.tscn”, si pegan la imagen dentro de esa carpeta, cuando vayan a godot les va a cargar algo (la imagen nueva) y deberían poder verla dentro del programa.



Ahora que tengo mi **Recurso** en forma de **textura(imagen)** necesito crear un nodo que lo contenga.



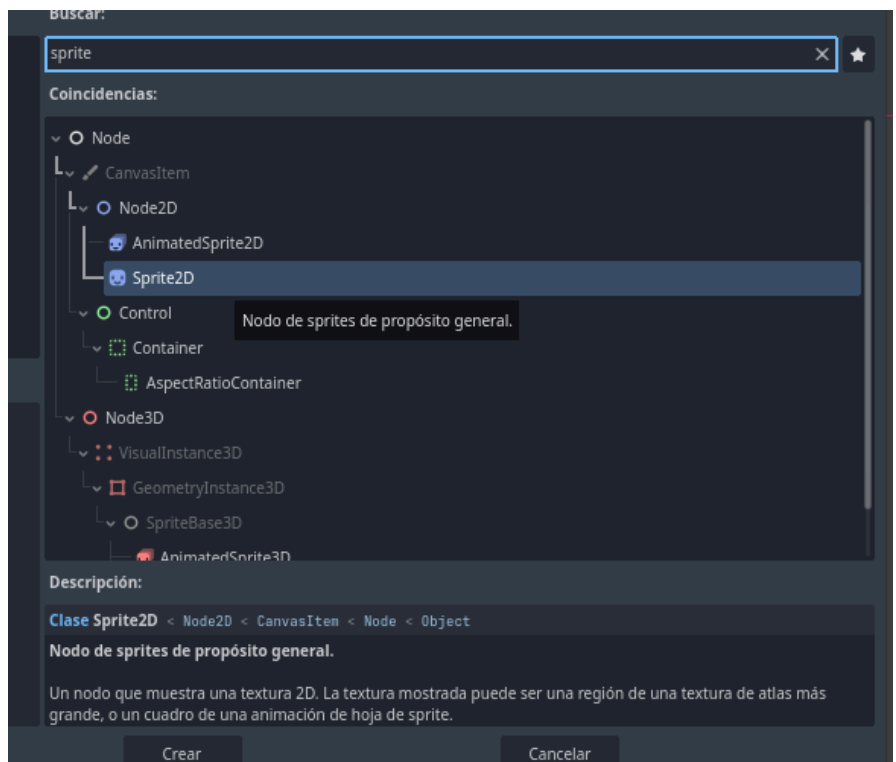
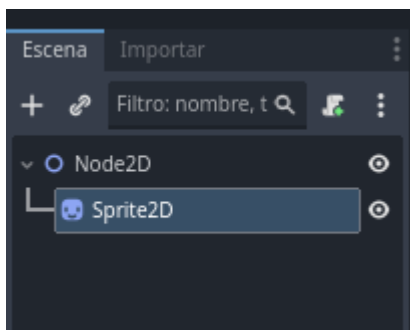
Aca le pueden hacer clic derecho sobre Node2D y le añaden un nodo hijo, esto les va a abrir una nueva ventana.

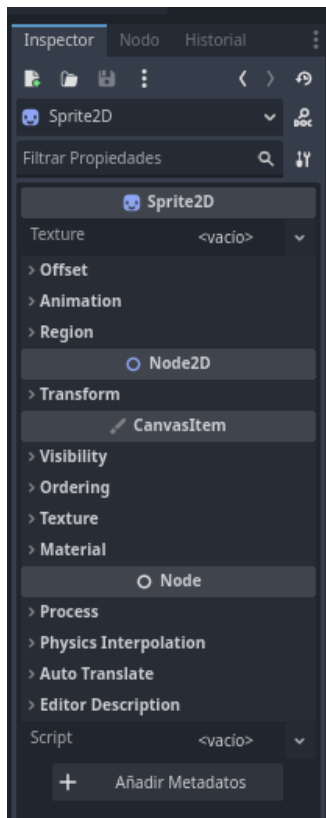


Ahí tienen que buscar un nodo que este preparado para recibir una textura, en el caso de godot se llama Sprite2D. Una vez lo encuentren le pueden hacer doble click y lo van a agregar como nodo hijo de Node2D.

¿Qué significa nodo hijo? Significa que ese **nodo hijo** está **ligado al padre** y todo **cambio que se le realice al nodo padre** le va a **afectar al nodo hijo**.

Así en criollo quiere decir que por ejemplo si muevo de posición a Node2D, el nodo hijo Sprite2D lo va a seguir





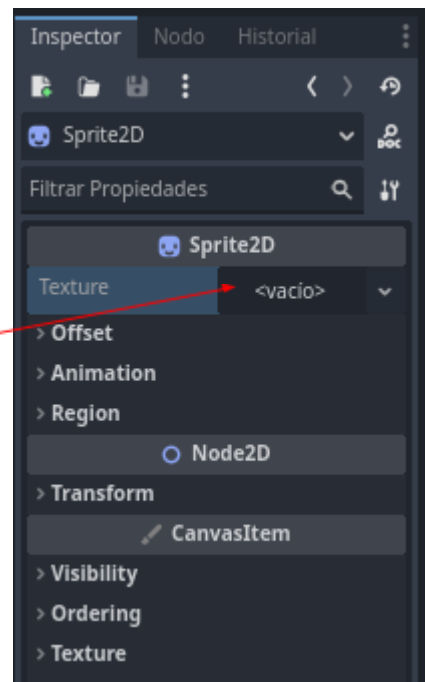
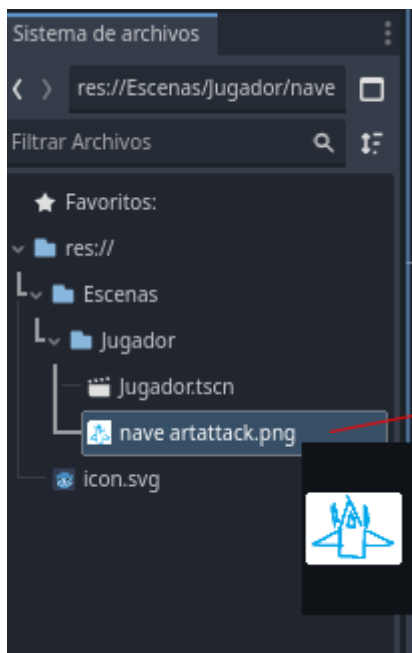
Ahora que tengo un nuevo nodo Sprite2D lo selecciono y voy a ver que en el inspector (a la derecha) van a haber más propiedades.

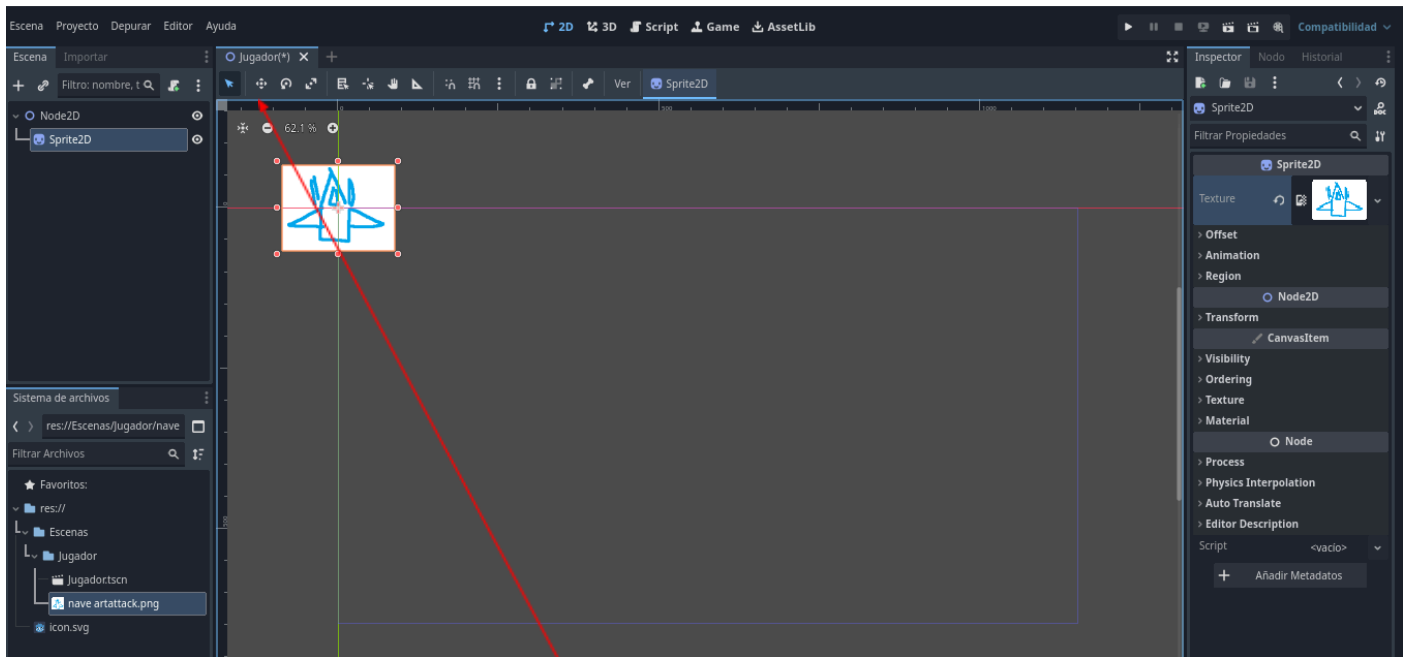
La que nos interesa es esta



Esa propiedad es la que va a hacer que ese nodo tenga textura (imagen)

Si le hacen clic a la flechita al lado de “vacío” les va a agarrar un ataque así que es mejor arrastrar la imagen desde donde están los recursos (abajo a la izquierda) hasta ese lugar.





Les debería quedar algo así (quizás más lindo)

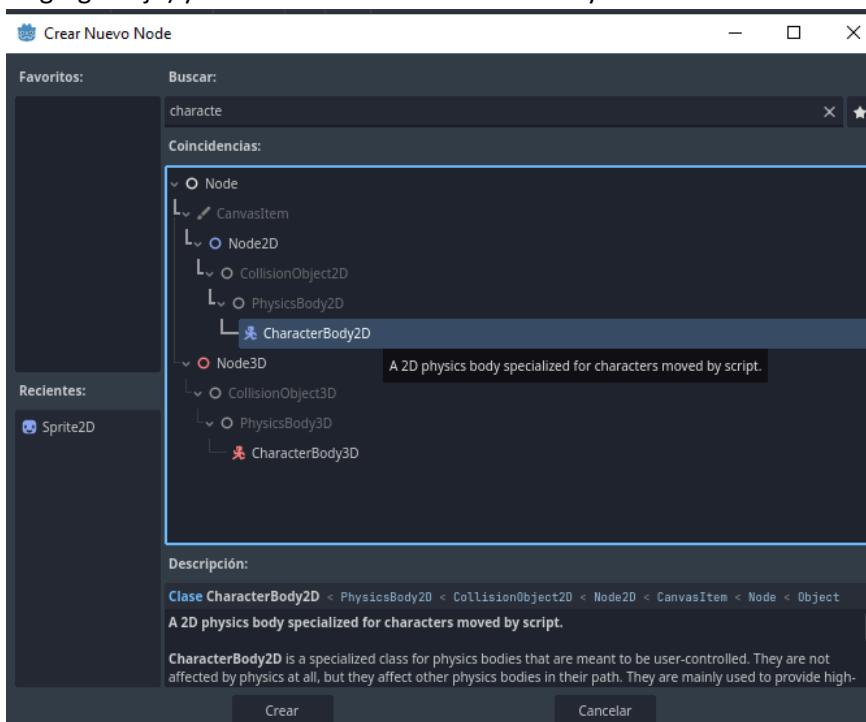
Ahora, ¿qué hicimos entonces? Le dimos un nodo de textura a nuestro nodo principal y le metimos una imagen para visualizarlo, si quieren ahora pueden mover el nodo principal o padre y van a ver que se mueve el nodo hijo de 'Sprite2D'.

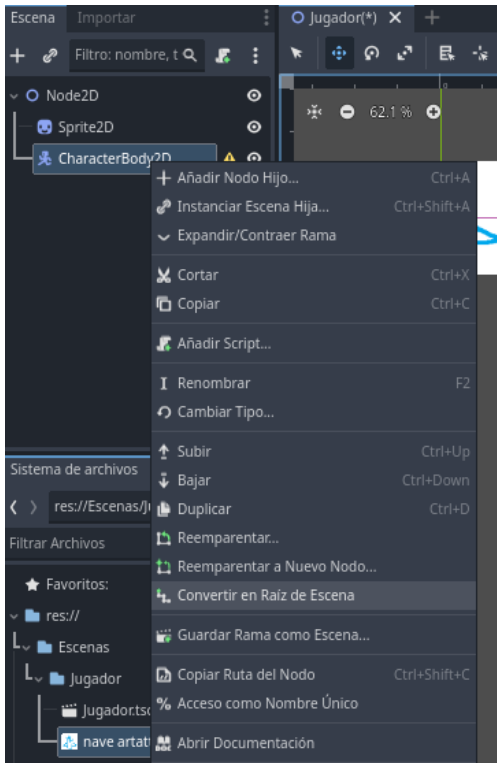
Pueden utilizar la herramienta de mover donde está marcado. Siempre que tengan dudas de qué hacer cada cosa pueden ponerle el mouse encima un ratito y a veces les dice el nombre o lo que hace. Asegúrense de tener seleccionado el nodo padre (Node2D) cuando vayan a mover así mueven todo junto. También pueden probar mover el nodo hijo a otro lado y luego mover el nodo padre para que vean cómo están conectados.

Ahora que tenemos una imagen vamos a hacer un script para moverla. Hay diferentes formas de mover un nodo dentro de Godot.

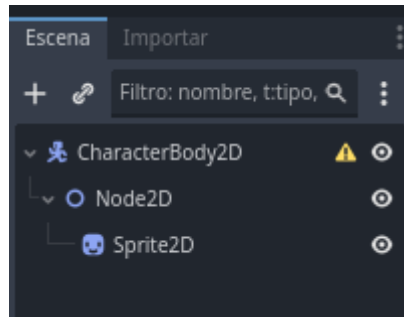
Yo utilicé el tutorial propio de Godot https://docs.godotengine.org/es/4.x/tutorials/2d/2d_movement.html y también se los recomiendo, pero primero para que lo hagan hay que cambiar unas cosas para que les quede bien seguir ese tutorial.

Primero es cambiar el nodo raíz o nodo padre, para esto se crea un nuevo hijo (clic derecho sobre el nodo padre -> agregar hijo) y buscamos el nodo 'CharacterBody2D'.



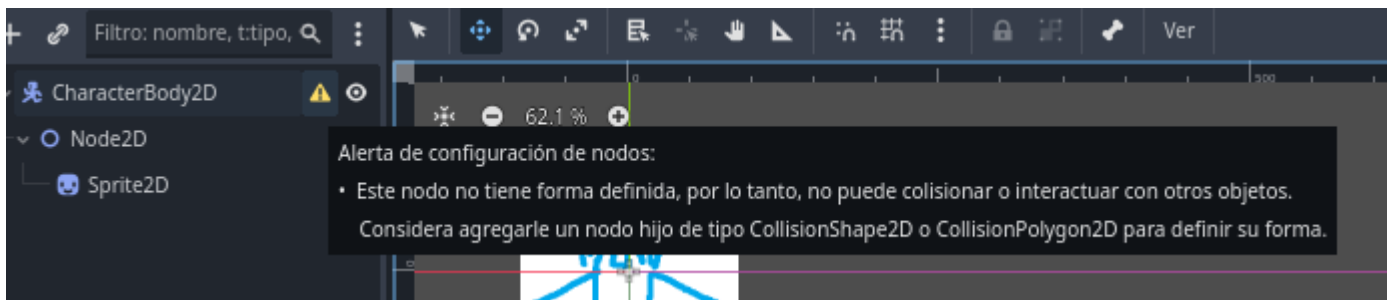


Una vez lo tengamos como hijo, le hacemos clic dercho y seleccionamos Convertir en “Raíz de escena”. Esto va a hacer que cambie el nodo padre a ese. Entonces quedaría algo así.

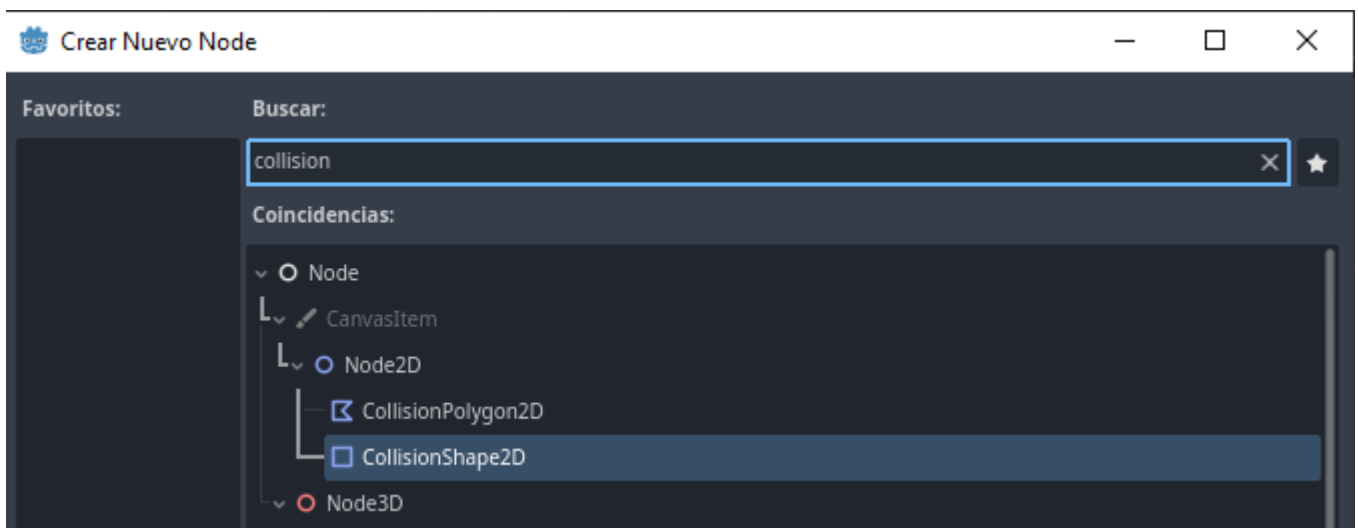


Ahora characterBody2D es el nodo padre o raíz.
Node2D es nodo hijo de CharacterBody2D
Sprite2D es nodo hijo de Node2D

La alerta amarilla pueden ponerle el mouse arriba y les va a decir la advertencia que está tirando.

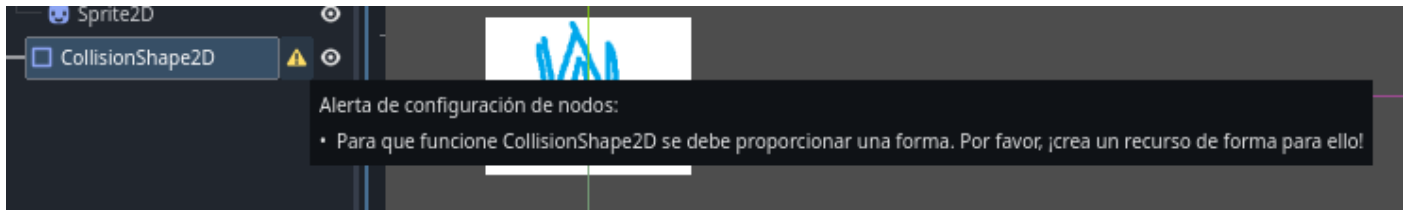


Esto quiere decir que le falta un nodo tipo “CollisionShape2D” así que también se lo agregan como nodo hijo de characterbody2d.

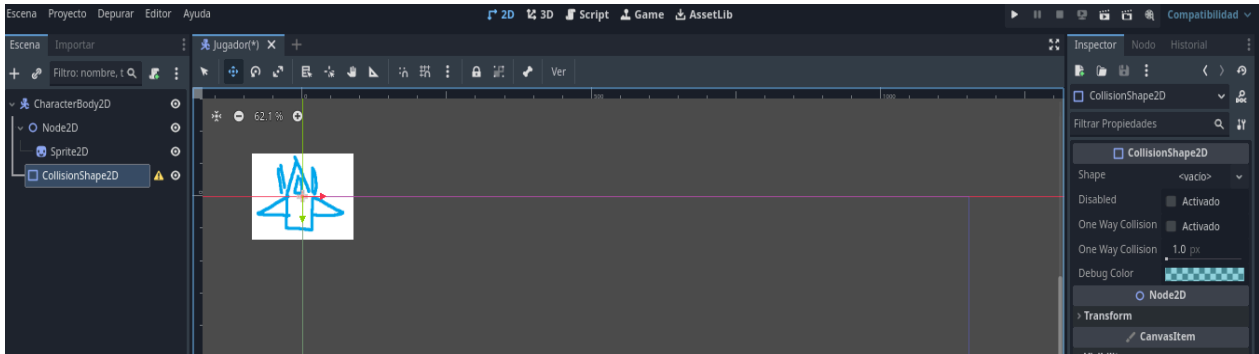


Pueden usar cualquiera, yo usé ese que está seleccionado.

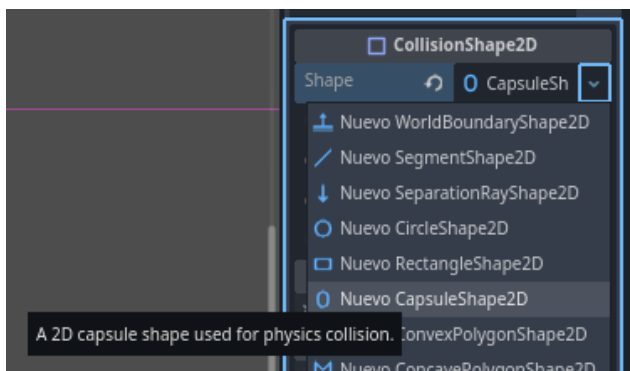
Ahora que tienen un nodo collision hay que definirle una forma



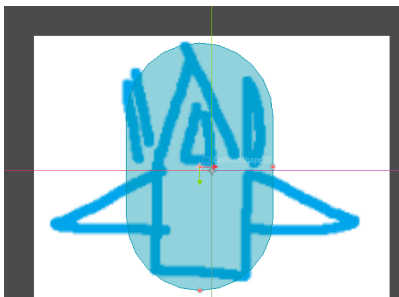
Para eso van al inspector, acuérdense de tener seleccionado el nodo colisión para ver las propiedades del mismo.



Donde dice shape es lo que hay que cambiar, ahora si pueden tocar la flechita y seleccionar la forma que consideren va a tener su nave, en mi caso es tipo una capsula

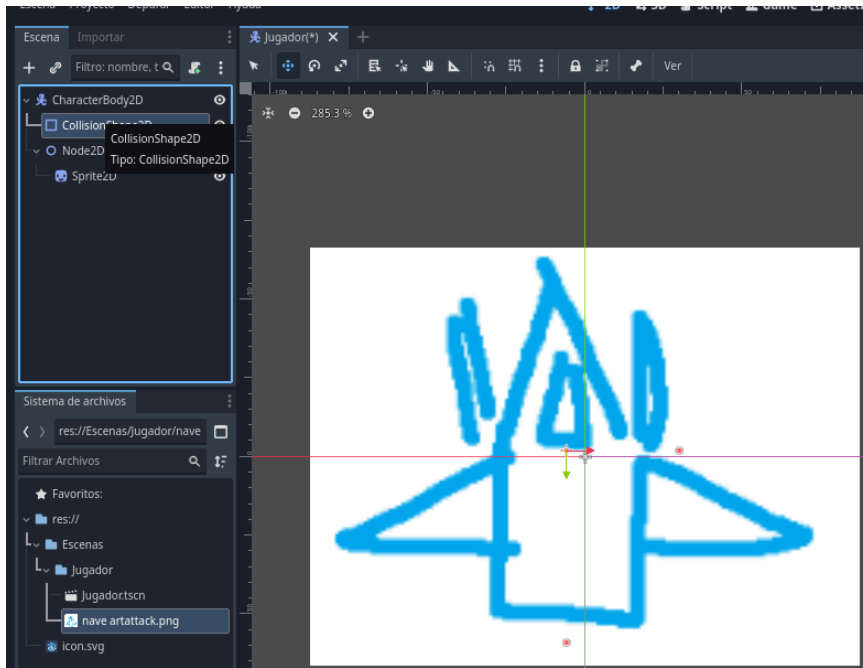


Una vez definida ya no deberían tener mas alertas y deberían ver una sombra celeste sobre su imagen(esta sombra solo es visible cuando estamos editando y no se ve cuando ejecutamos la escena). Ahora es cuando modificamos la capsula para que abarque la figura de nuestra textura, en mi caso la nave.

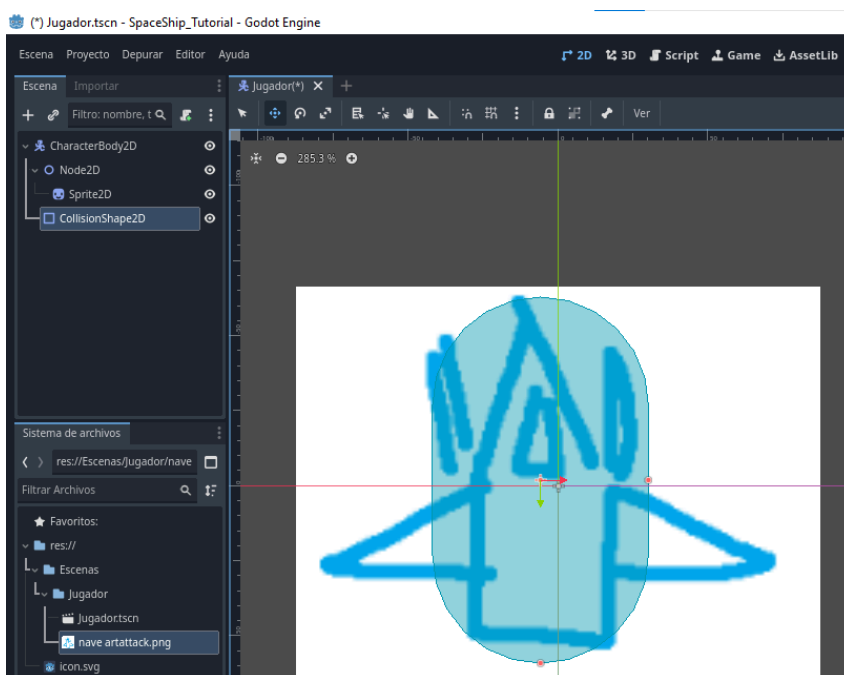


Pueden mantener presionado los botones rojos de la capsula para modificar el tamaño.

Si por alguna razón no ven la sombra celeste, puede ser porque el nodo colision este “debajo” de la imagen



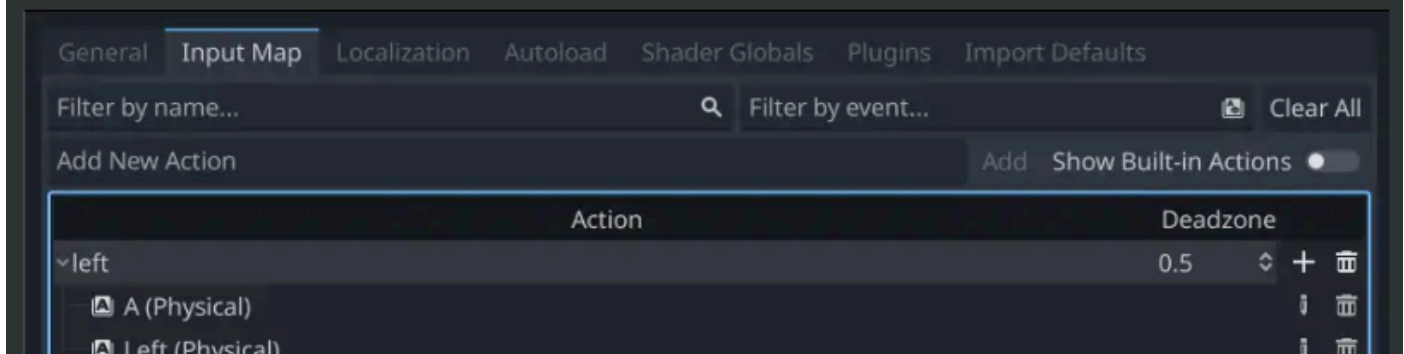
Esto se arregla moviendo el nodo collision mas abajo que el nodo donde esta la textura, arrastrando el nodo collision sobre el nodo padre characterbody



¿Ahora que hicimos? Cambiamos el nodo padre por un nodo characterbody2d. eso es porque los nodos están preparados para ser manejados por scripts y cada nodo tiene su forma, el tutorial pide que utilicemos ese nodo padre porque debe ser el que está preparado para ser manejado por un jugador. Después agregamos un nodo hijo de collisionshape2d, que hace eso? Le da un área que nos permite interactuar con el entorno (chocar una pared, recibir daño de un asteroide, etc.)

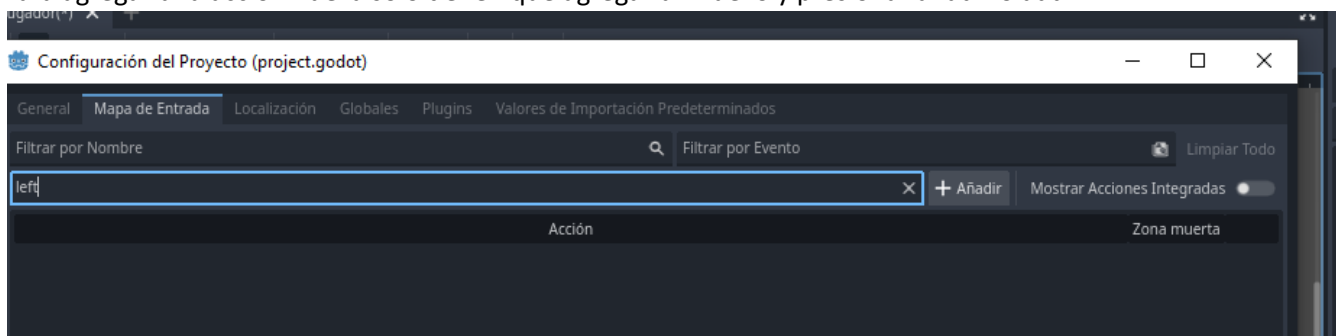
Ahora que tienen todo lo necesario para continuar el tutorial yo voy a comentar ciertas cosas nomas.

Abre **Proyecto -> Ajustes del Proyecto** y selecciona la pestaña **Mapa de entradas**. Agrega las siguientes acciones (ver **InputEvent** para más detalles):

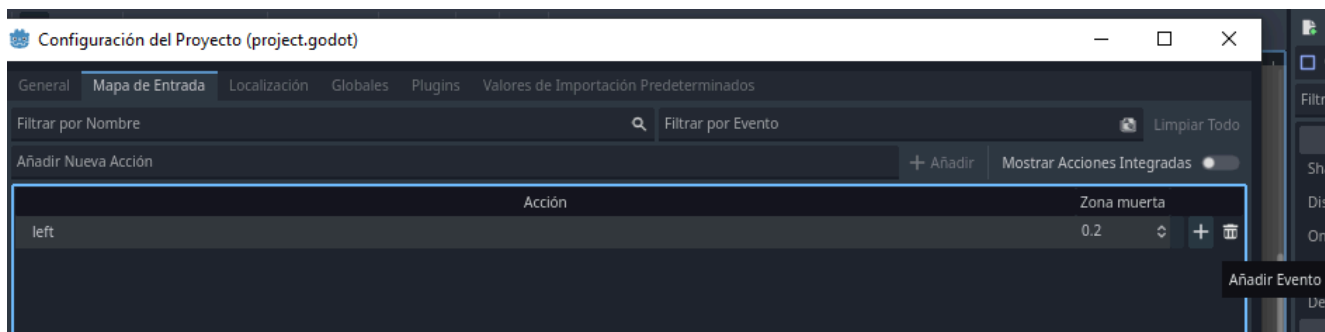


Aca pueden agregar “acciones” que van a estar ligadas a teclas, lo que les pide es agregar las acciones que les muestra. Esto es para después en el script hacer referencia a estas acciones para que el programa entienda que por ejemplo “acción left” es tocar la tecla a o la flecha izquierda.

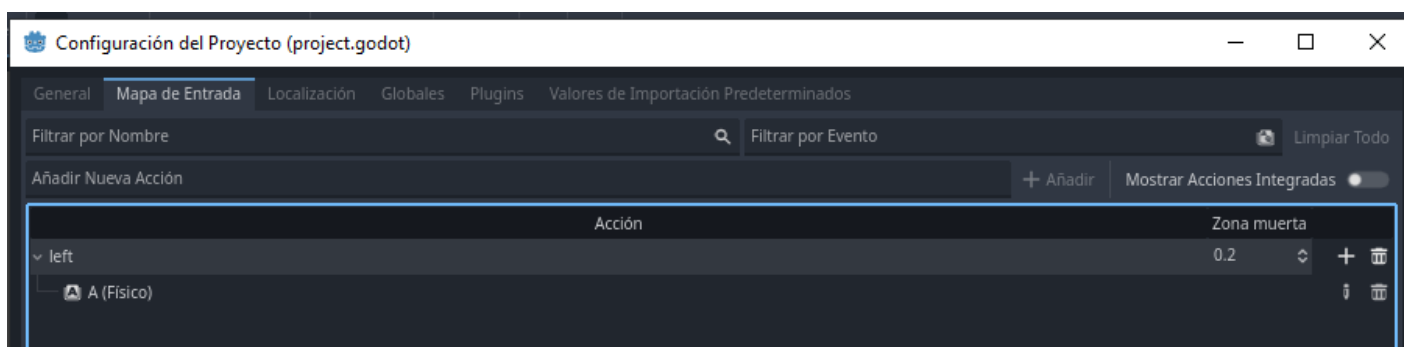
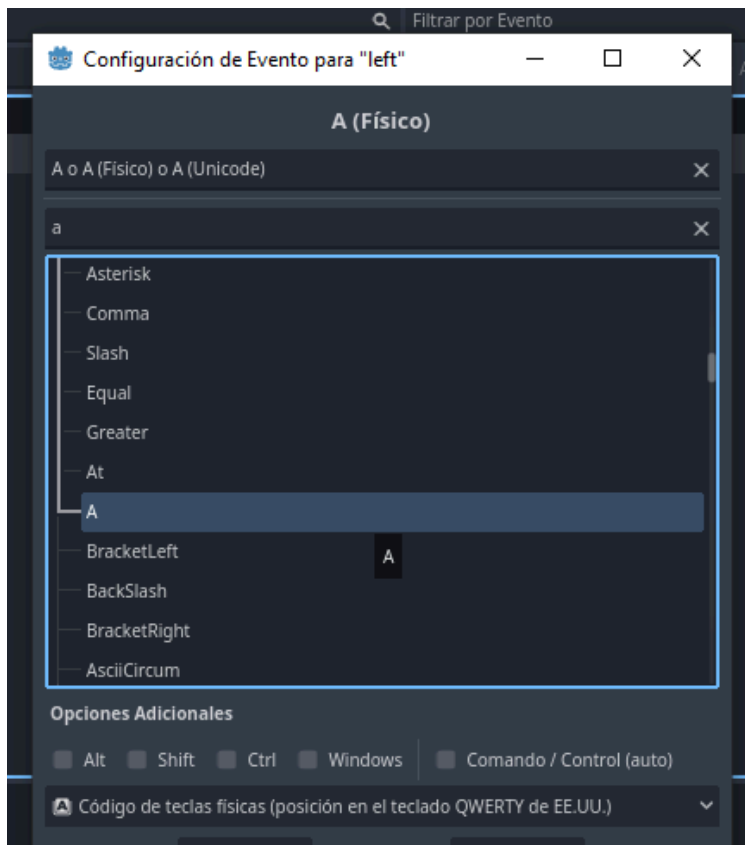
Para agregar una acción nueva solo tienen que agregar un nuevo y presionar añadir o add.



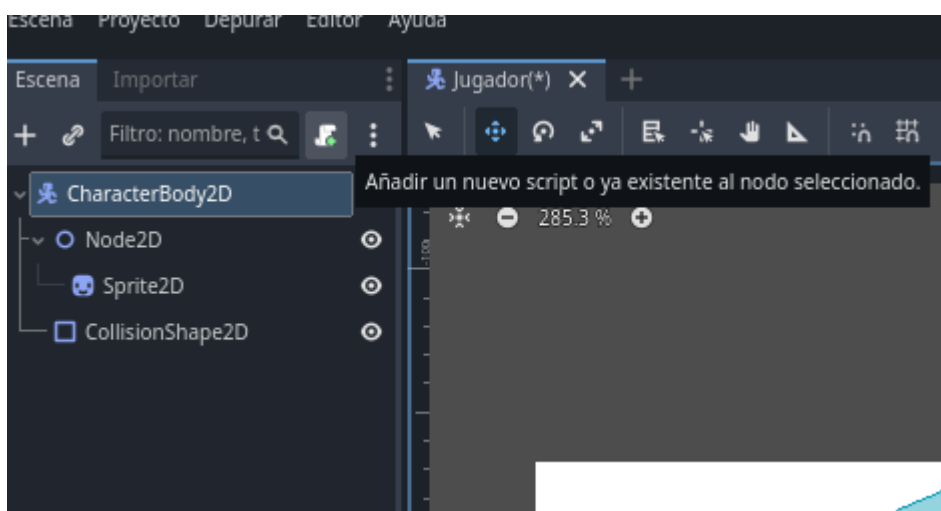
Una vez crean la acción le tienen que asignar la tecla



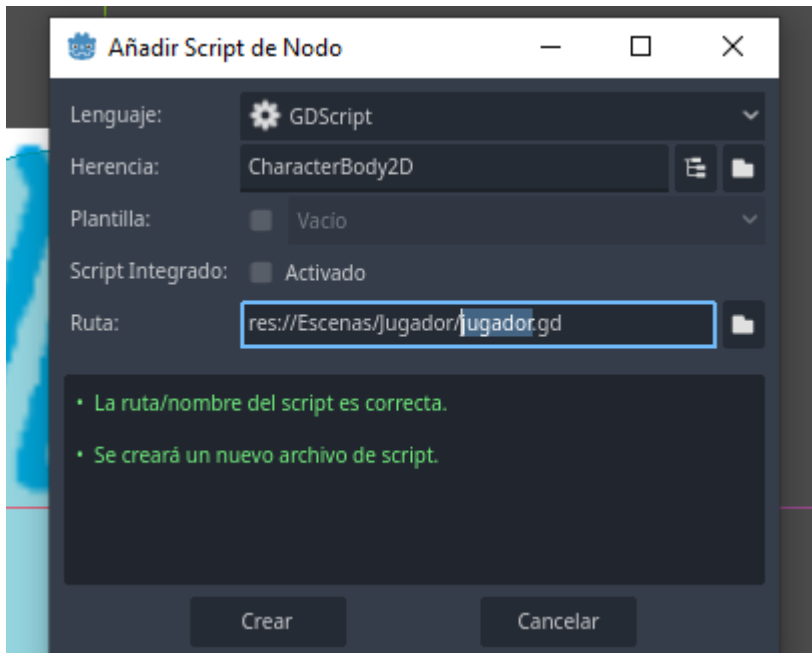
Y buscan con el filtro



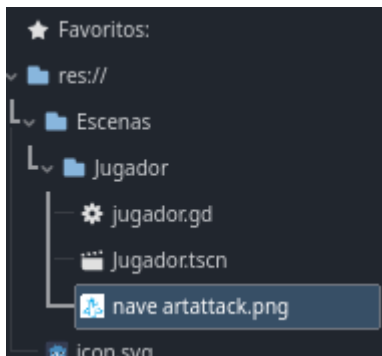
Una vez completan todos los pide que agreguen un script al nodo padre characterbody2d. Para eso solo tienen que seleccionar este botón **Mientras este CharacterBody2D seleccionado.**



Esto les va a querer crear un script ligado a ese nodo



Ahí lo crean y como esta dentro de su propia carpeta queda todo ordenado



Jugador.gd es el nombre del script, lo identifican por la terminación .gd.

Añade un script al character body y agrega el siguiente código:

GDScript


C#

```
extends CharacterBody2D

@export var speed = 400

func get_input():
    var input_direction = Input.get_vector("left", "right", "up", "down")
    velocity = input_direction * speed

func _physics_process(delta):
    get_input()
    move_and_slide()
```

En la función `get_input()`, usamos `Input`  `get_vector()` para verificar los cuatro eventos clave y la suma devuelve un vector de dirección.

Esto es el código que debería quedarle en el script para poder mover su nave con los botones

Voy a intentar explicar que hace cada parte pero no estoy seguro de que entiendan.

Extends	no se, pero cada script tiene que empezar con el nombre de su nodo.
@export	es para que se vea la variable en el inspector
var speed = 400	esta definiendo una variable llamada "speed" con un valor de 400

```
func get_input():  
    var input_direction = Input.get_vector("left", "right", "up", "down")  
    velocity = input_direction * speed
```

Esta definiendo una nueva función llamada get_input. Lo que hace esa función es crear una variable llamada input_direction que tiene un valor de acuerdo a la tecla que se presione,

Input.get_vector("left", "right", "up", "down") es otra función que pertenece a godot y busca las teclas que pertenecen a esa acción y las "devuelve" en un vector en base a la tecla presionada (es lo que configuraron arriba con las teclas). Creo.

Una vez que tiene la tecla que presiono, la multiplica por la velocidad, que lo define arriba y eso le da una "velocidad" al objeto (en este caso a CharacterBody2D)

```
func _physics_process(delta):  
    get_input()  
    move_and_slide()
```

Esta función es la que se usa para que un script se ejecute en cada "frame" de la escena, básicamente está andando constantemente buscando que hacer mientras el script está funcionando. O dicho de otra forma mientras la escena este siendo reproducida.

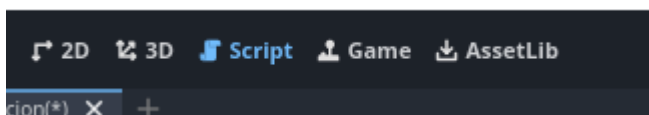
En este caso esta ejecutando la función que definió al principio, que detecta si alguna de las teclas está siendo presionada, y de ser así hace un cálculo y modifica la velocidad del objeto, al ser constante podemos manejar el objeto con las teclas que definimos.

Move_and_slide no estoy seguro que hace, pero debe ser parte del movimiento.

Nota a tener en cuenta. Las variables pueden tener cualquier nombre. Todo lo que escribamos en el código tiene que ser preciso, para el programa "input" e "Input" son 2 cosas diferentes. Esto es algo que todavía no vimos muy a fondo y yo tampoco se explicarlo mucho, pero es para que entiendan maso menos que está pasando en el script de movimiento.

Una vez tengan el script lo guardan (Control+S) y ya pueden ejecutar la escena, deberían poder mover su objeto con las teclas.

Entonces que hicimos? Agregamos unas acciones dentro del proyecto y les vinculamos unas letras, agregamos un script al nodo padre o raíz CharacterBody2D y codearon o copiaron y pegaron el código para hacer que el objeto se pueda mover con las teclas.



Recuerden que pueden usar estos botones para cambiar la vista de script a 2D

Ahora tenemos una **escena jugador** que posee varios nodos y un script que permite mover el objeto. Joya.

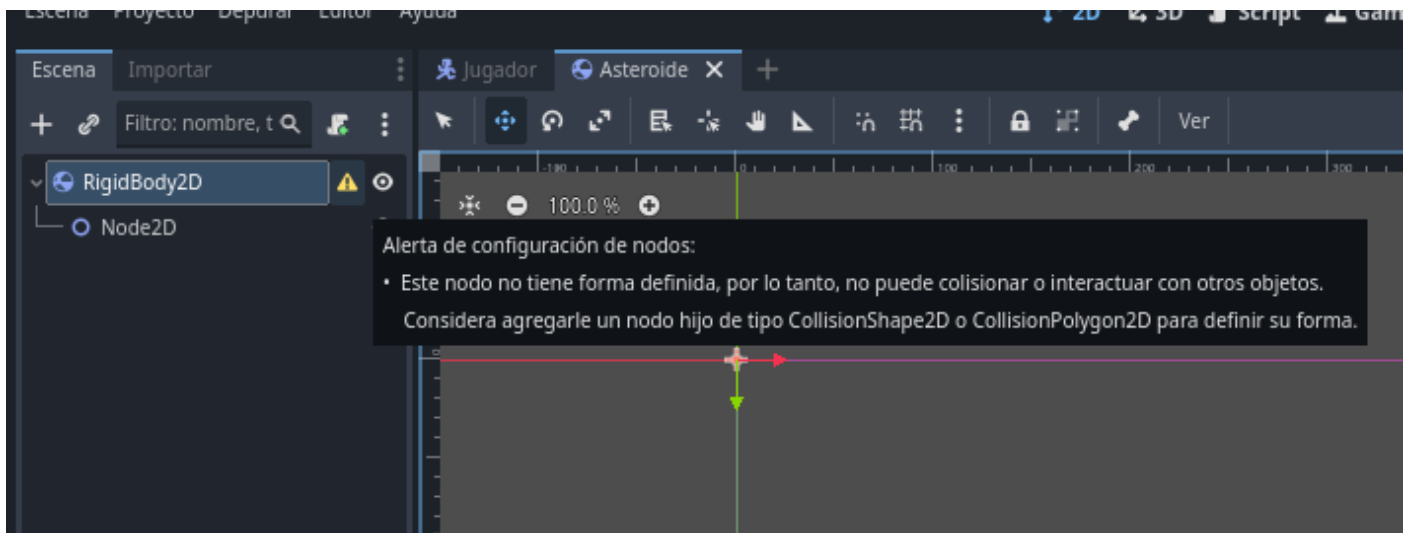
Lo próximo que podemos hacer es un obstáculo o enemigo así que creamos **otra escena** y nos preguntamos ¿qué hace el obstáculo o el enemigo? ¿Qué forma tienen? ¿Qué **comportamiento** va a tener? Pero primero creamos la escena (Control+n o clic en el “+” al lado de la escena del jugador) y la guardamos en su respectiva carpeta.

Yo así a lo simple se me ocurrió que un “enemigo” sea una piedra o asteroide que caiga desde arriba de la pantalla. Entonces necesito:

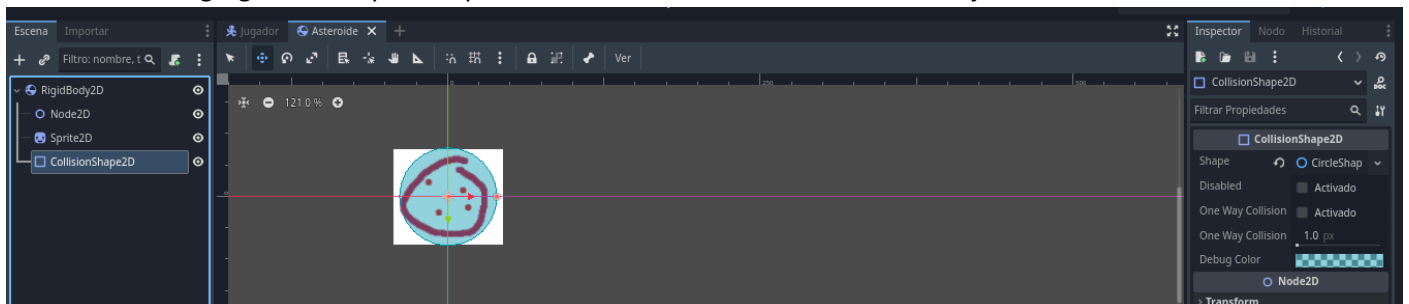
Una textura de asteroide, una forma de hacer aparecer enemigos allá arriba y que bajen o se muevan para abajo.

El tutorial de la upso encara el movimiento utilizando un nodo de RigidBody2d. Este nodo tiene la particularidad que se ve afectado por la “física” del motor, quiere decir que le afecta la gravedad. Entonces solo necesita que aparezcan arriba y la gravedad los atrae hacia “abajo”.

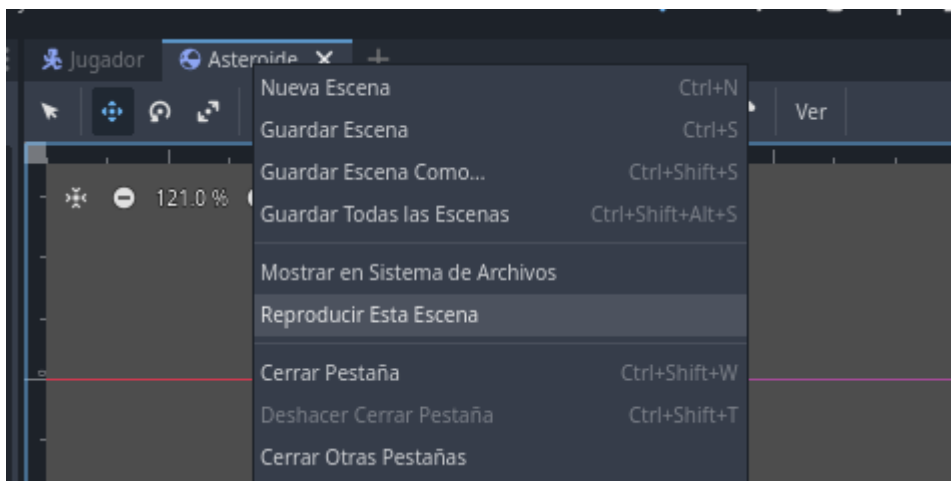
Empezamos agregando un nodo RigidBody2D y colocándolo como nodo raíz.



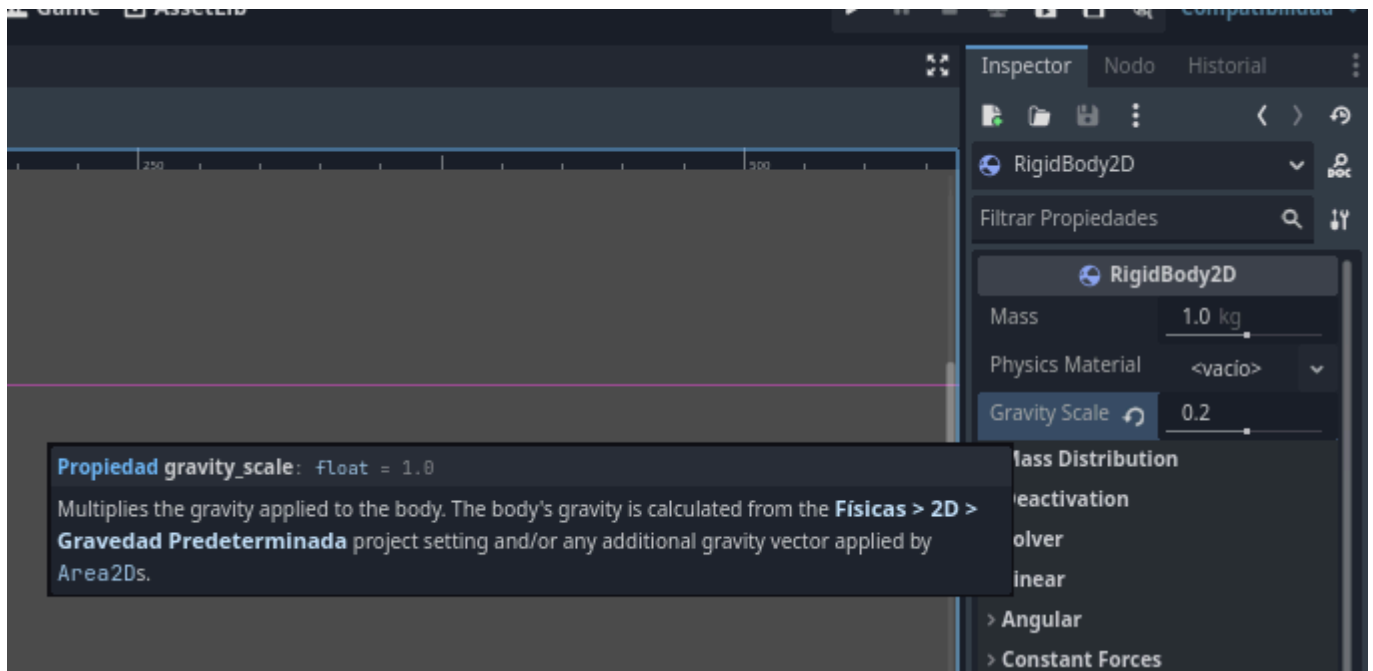
Ahora también agregamos un sprite2d para darle una textura. Y un collider así no jode la advertencia.



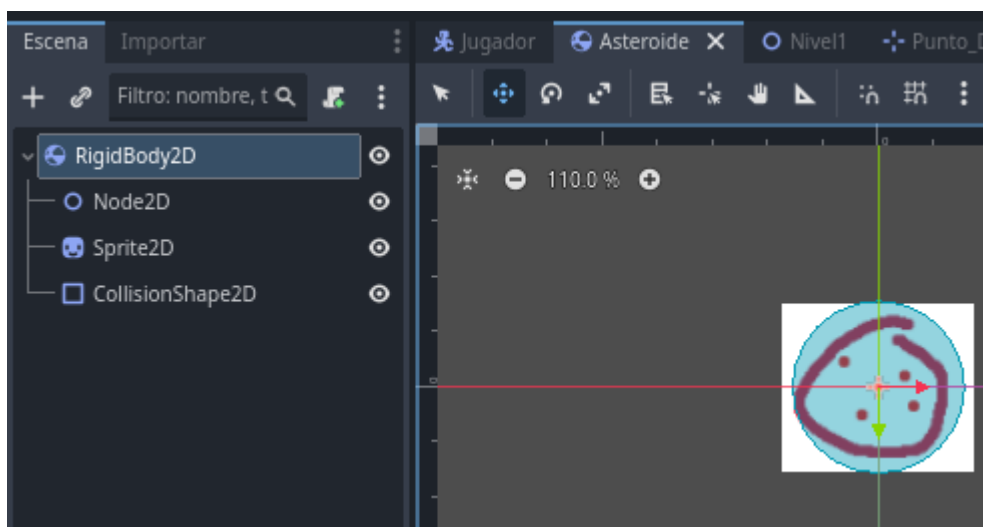
Ahora si ejecutamos la escena del asteroide (clic derecho sobre la escena asteroide -> reproducir esta escena)



Podemos ver como que la roca cae porque es afectada por la gravedad. Pero quizás sea demasiado rápida así que volvemos a la escena del asteroide, seleccionado el RigidBody2d y en el inspector buscamos la propiedad gravity scale y le bajamos el valor. Guardamos la escena y probamos de nuevo.

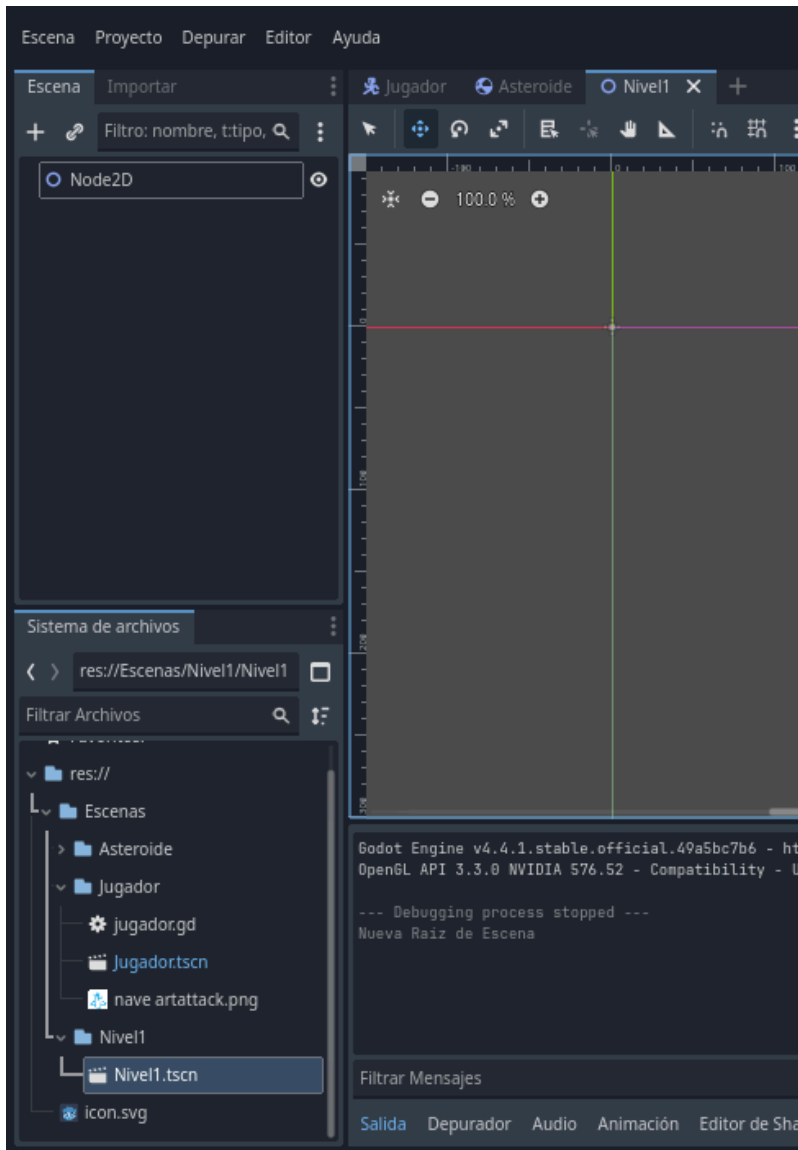


¿Que hace gravity scale? Es que tan fuerte le afecta la gravedad a ese nodo. Al bajarle el valor, como por ejemplo 0.2, vemos que ahora la gravedad le afecta menos, haciendo que baje más lento. Esto es una de las formas que tenemos de mover objetos que no requieran unos scripts muy complejos, podemos aprovechar las propiedades de los nodos para hacer estas cosas.

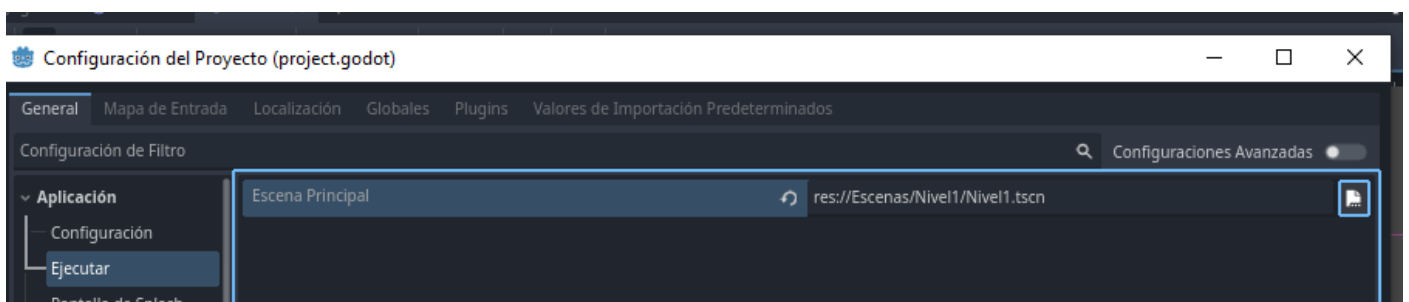


Ahora tenemos una escena asteroide que contiene un nodo padre o raíz RigidBody2D que le afecta la gravedad, un nodo textura (la imagen de la piedra) y un nodo colisión. Pero necesitamos ahora una forma de que aparezca la roca arriba y que aparezcan varias.

Para eso primero creamos una escena 2D (control +n como atajo rapido) donde va a ocurrir el gameplay o donde se va a jugar el videojuego. Esta escena principal va a contener las otras escenas (jugador, enemigos, fondo, punto de aparición de enemigos, etc.). Esta escena va a tener su propia carpeta llamada Nivel1, y la escena misma tambien se va a llamar Nivel1.

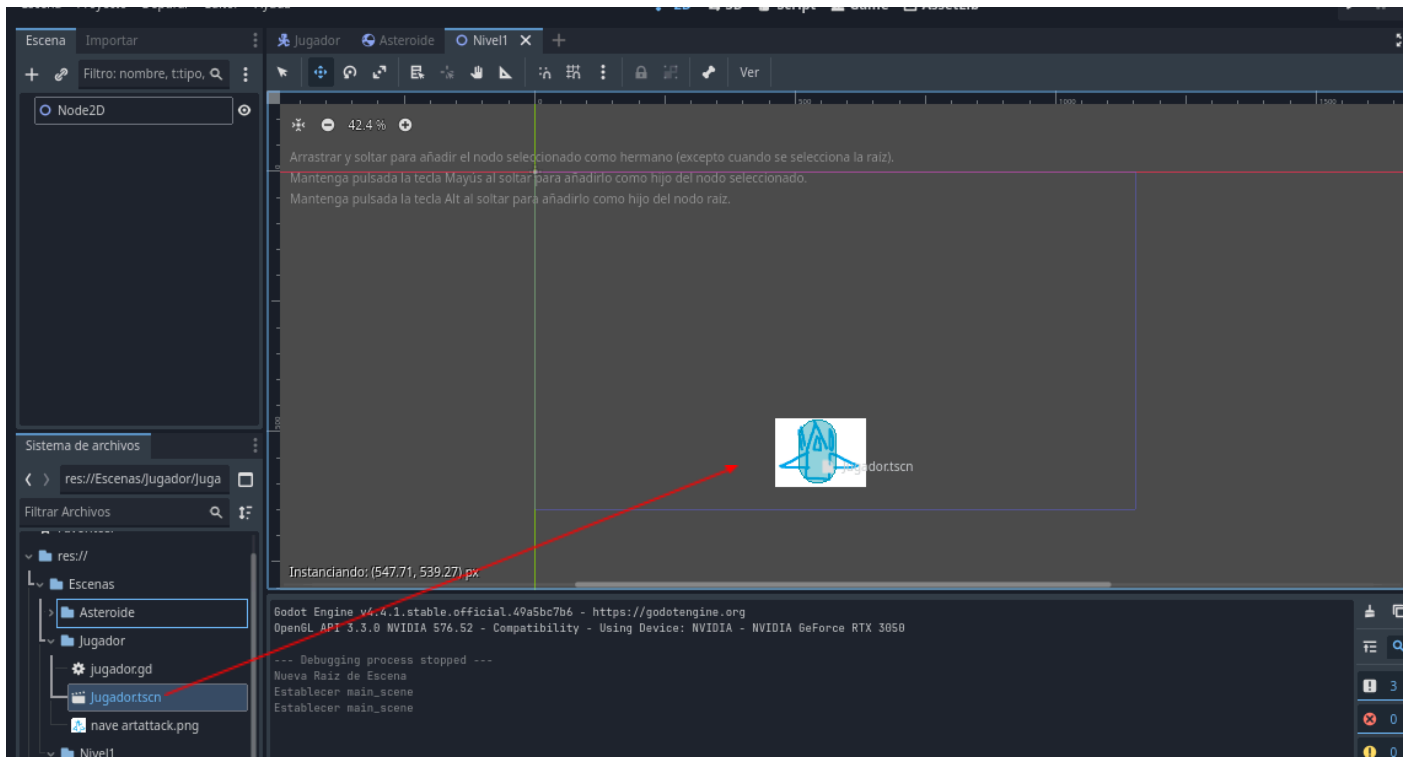


Una vez creada, la tenemos que definir como escena principal. Para eso vamos a la configuración del proyecto, pestaña general, ejecutar y cambiamos la escena principal por la de nivel1.

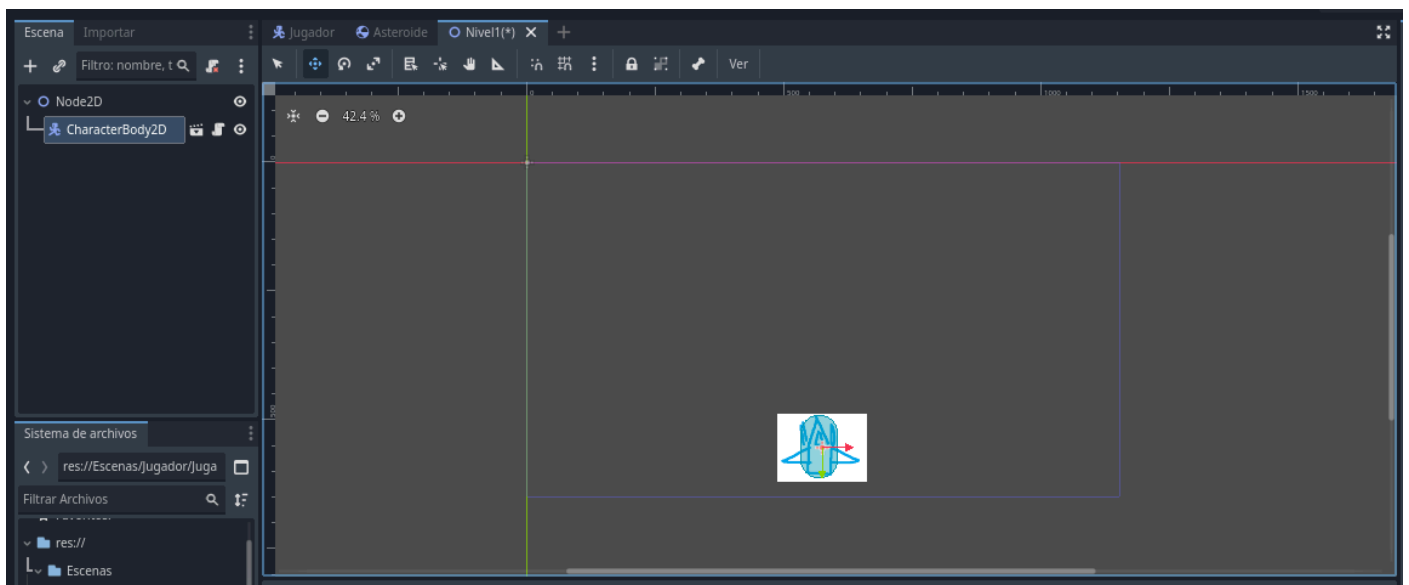


Ahora cada vez que ejecutemos la escena principal (F5 como atajo rápido) va a ejecutar nivel1.

Pero la escena actualmente no tiene nada así que primero vamos a agregarle la escena jugador.



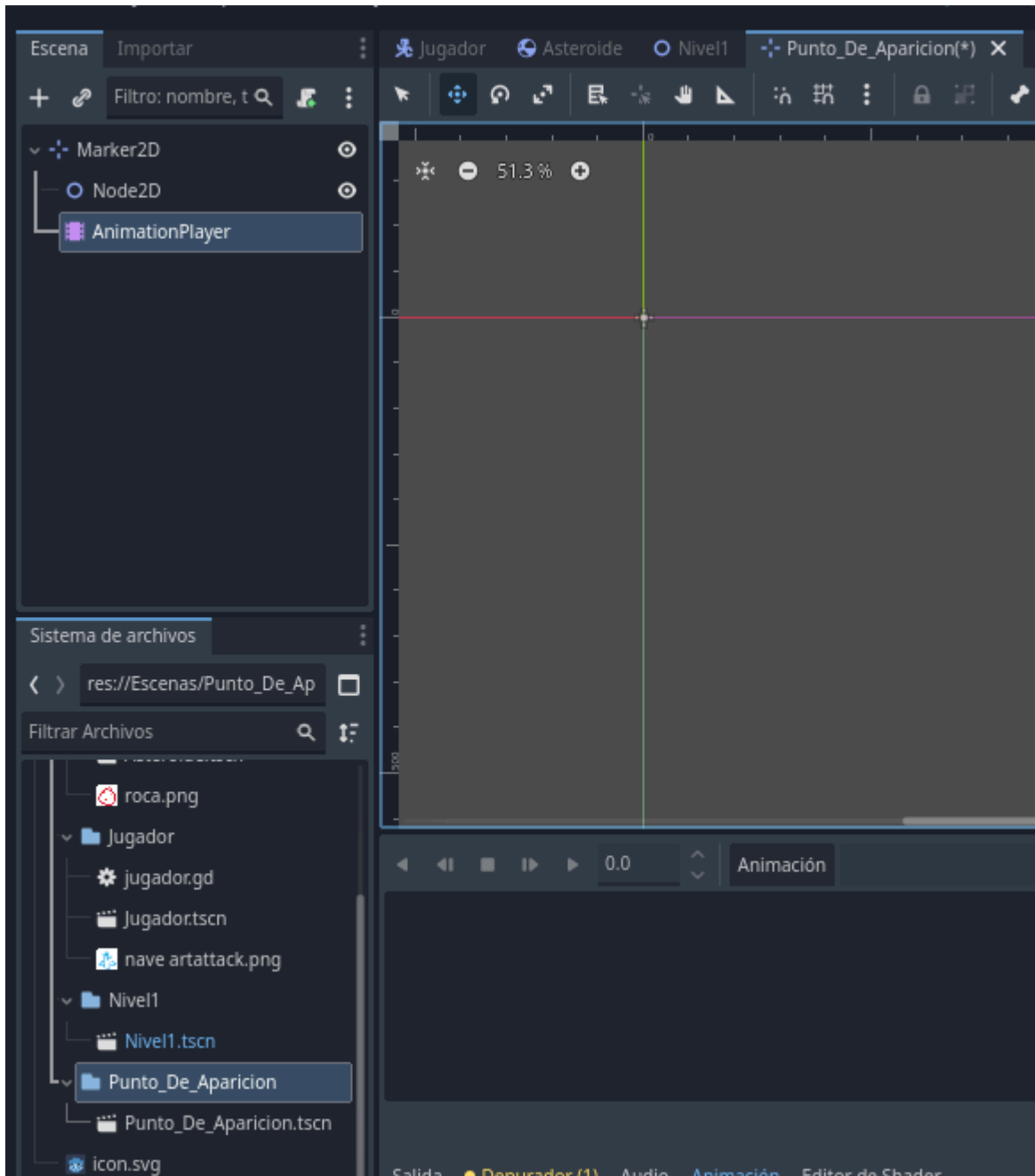
Se puede arrastrar desde los recursos a la pantalla de la escena de nivel1



Ahora la escena Nivel1 tiene la escena Jugador como si fuese un nodo. La diferencia es que ahora si cambiamos la escena jugador también se verá afectada en la escena Nivel1. Por eso también es conveniente tener diferentes escenas para diferentes cosas.

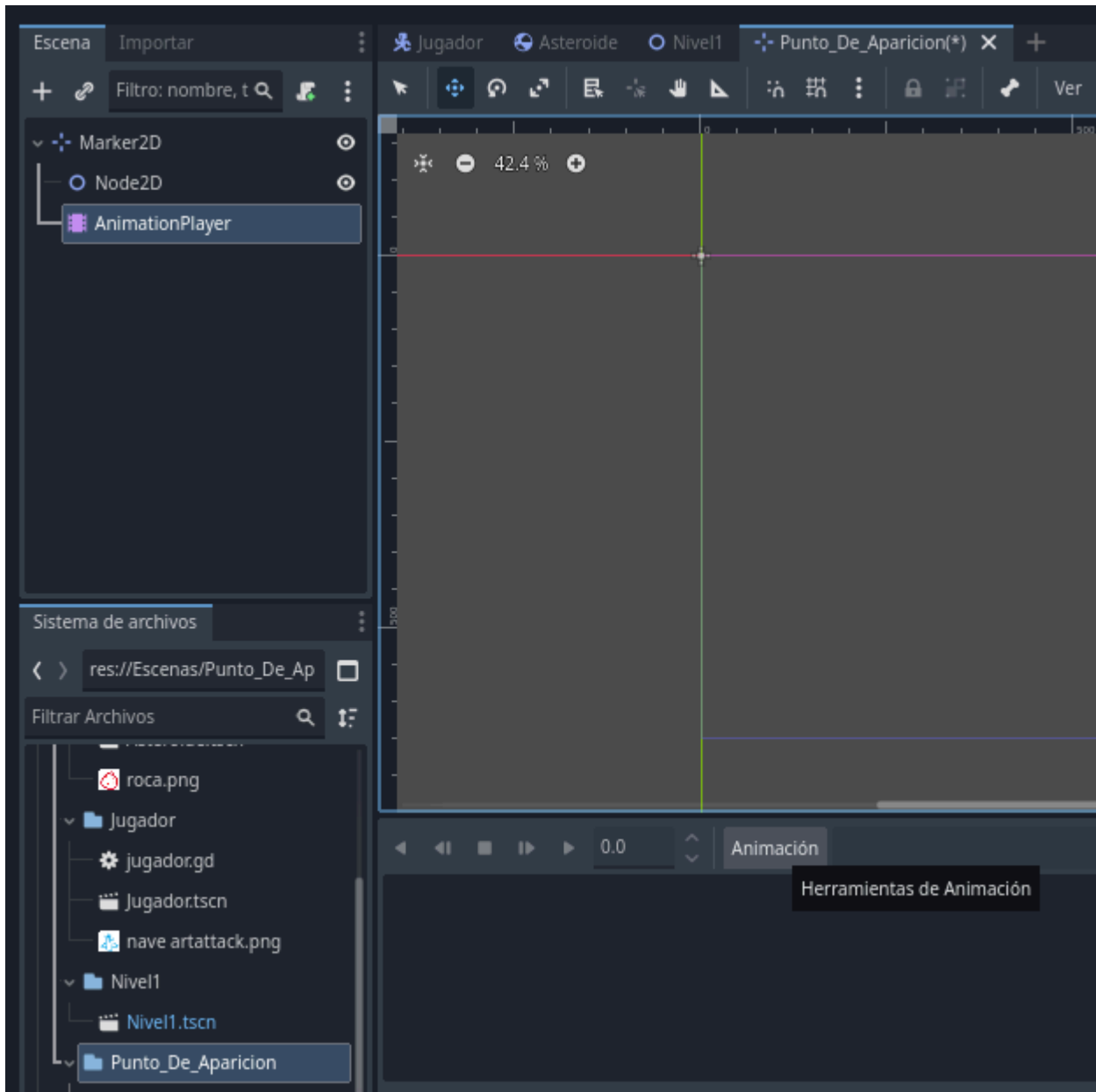
Tocando F5 ahora podemos jugar en la escena Nivel1 y manejar nuestro jugador en al escena Nivel1. Como próximo objetivo seria agregar que los asteroides aparezcan desde arriba y caigan, asi el jugador debería esquivarlas.

Para eso vamos a crear una nueva escena 2D (Control + N), llamada Spawn_Point o Punto_De_aparicion, como quieran y lo guardan dentro de su propia carpeta.



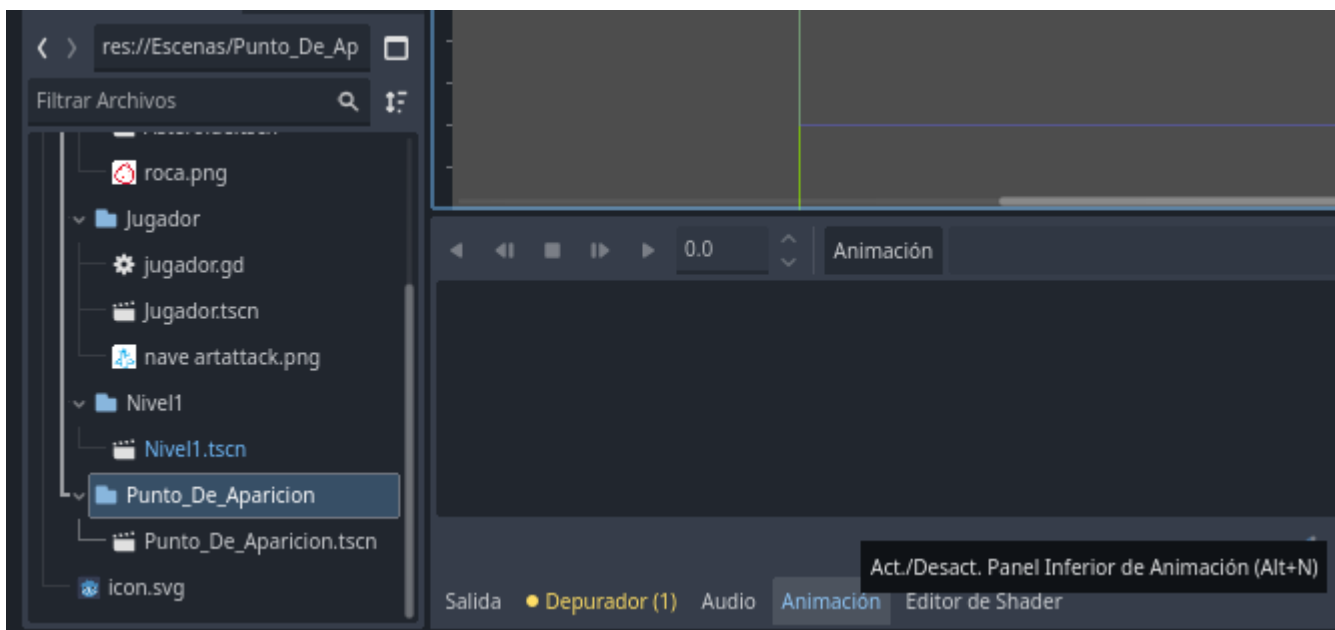
Dentro de esta nueva escena le agregamos un nodo Marker2D y lo convertimos en Nodo padre o raíz. A ese nuevo padre le agregamos otro nodo llamado AnimationPlayer. El marker2D no estoy seguro que hace, pero el tutorial lo hacia asi. El nodo AnimationPlayer da la posibilidad de agregar animaciones al nodo, en este caso queremos agregar una animación que mueva al marker2D de la punta izquierda a la punta derecha.

Para eso nos vamos al nodo AnimationPlayer y donde dice animación, podemos agregar una nueva animación

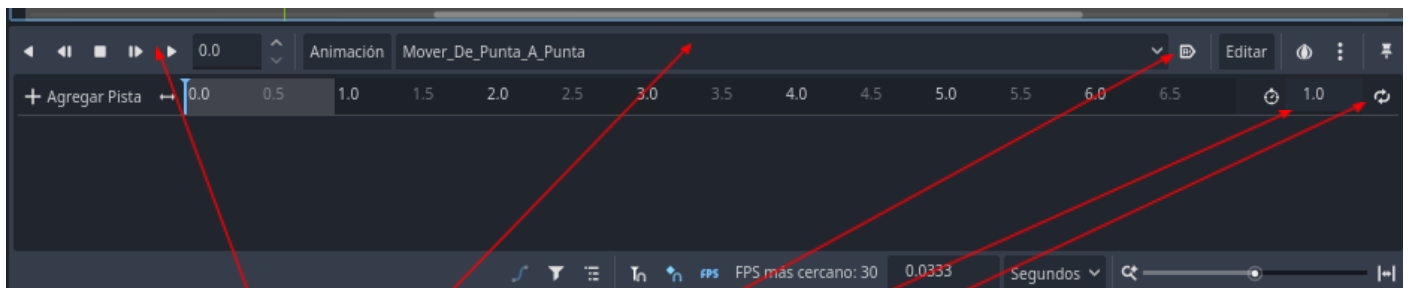
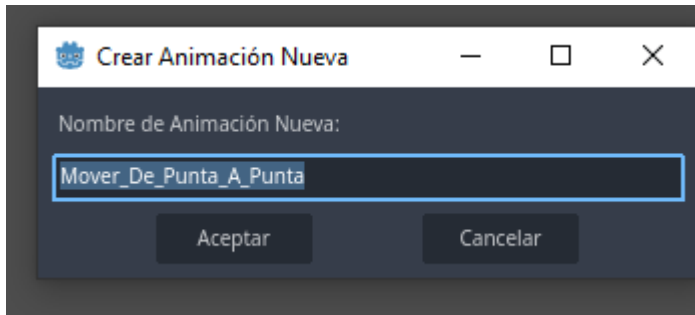
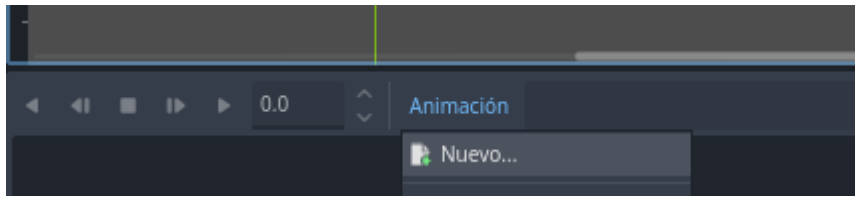


Yo lo veo así, me acuerdo que al profesor se le veía diferente así que no sé si ustedes también lo ven así.

Si por alguna razón no ven eso quizás tengan que activar el panel haciendo clic acá



Ahora cuando crean una animación les va a preguntar el nombre, pónganle el nombre de lo que va a hacer aplicando masomenos lo que vimos viendo (que se una verbo y eso)



Ahora les paso a explicar donde encontré algunas propiedades que vamos a usar.

Panel de reproducción de la animación: Aca pueden reproducir la animación dentro de la escena para ver como se va modificando.

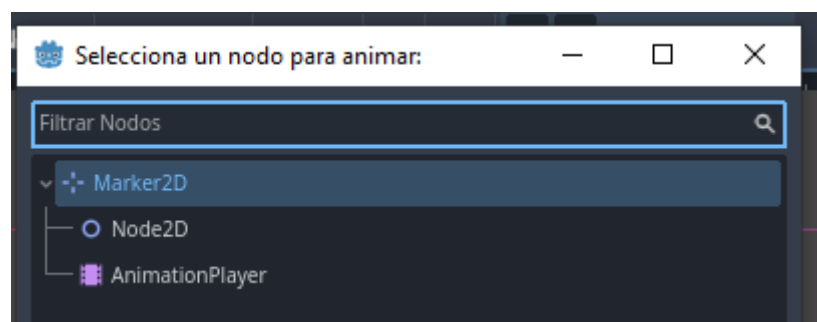
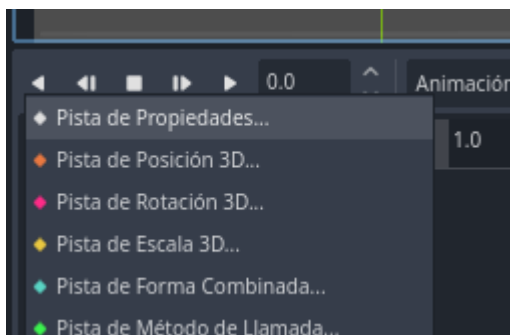
Nombre de la animación que tenemos seleccionada actualmente: aca van a encontrar una lista de todas las animaciones que vayan creando

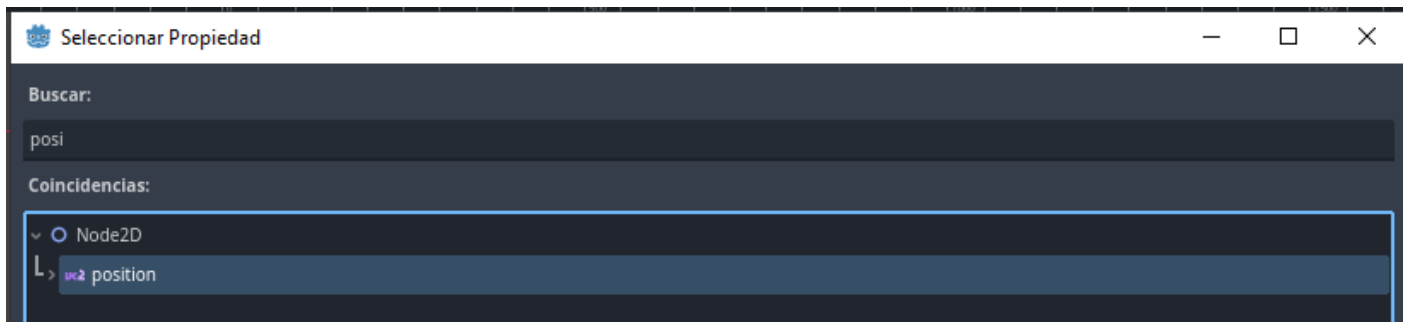
Reproducir automáticamente al cargar: Si tienen este botón activado la animación arranca apenas cargue la escena.

Duración de la animación: Cuanto va a durar la animacion

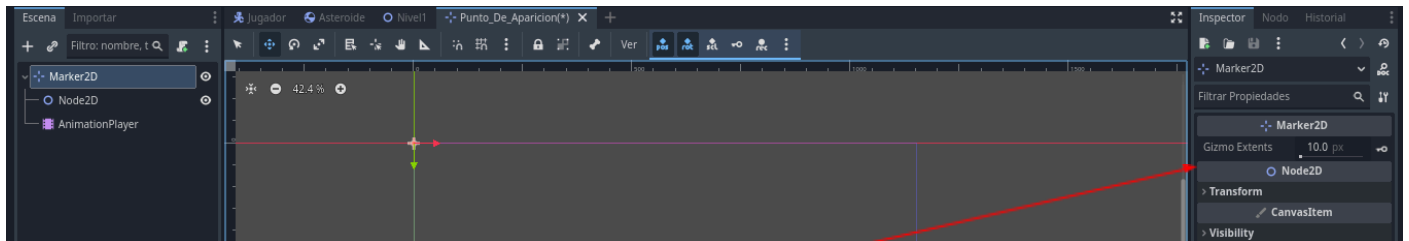
Loop de animación: si tienen este botón activado la animación se va a repetir una vez termine

Agregar pista: Aca es donde van a elegir que propiedad va a ser modificada en la animación, en nuestro caso queremos modificar la propiedad position del nodo Marker2D





Aca sale Node2D porque Marker2D en si mismo tiene un node2D dentro, que se encarga de la posición del nodo. Que no se confunda con el Node2D que esta en la escena aparte.



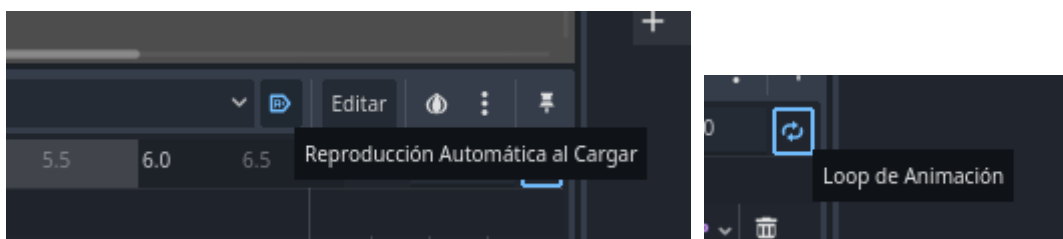
Como se ve aca, Marker2D ya de por si tiene un nodo llamado Node2D que tiene la propiedad transform. Este es el que necesitamos modificar.

Ahora que tenemos una nueva pista hay que agregale "claves" o "keys" que son puntos dentro de la duración de la animación donde se van a modificar las propiedades, en nuestro caso la position.



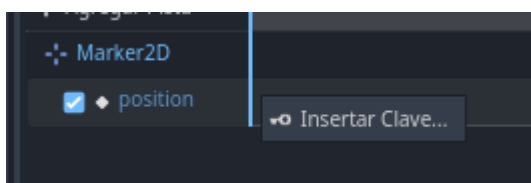
Le aumentan la duración de la animación al tiempo que quieran, esta duración es el tiempo que va a tomar en ir hasta la punta y volver. Yo le mande 6 segundos.

Despues active tanto la reproducción automática con el loop de animación.

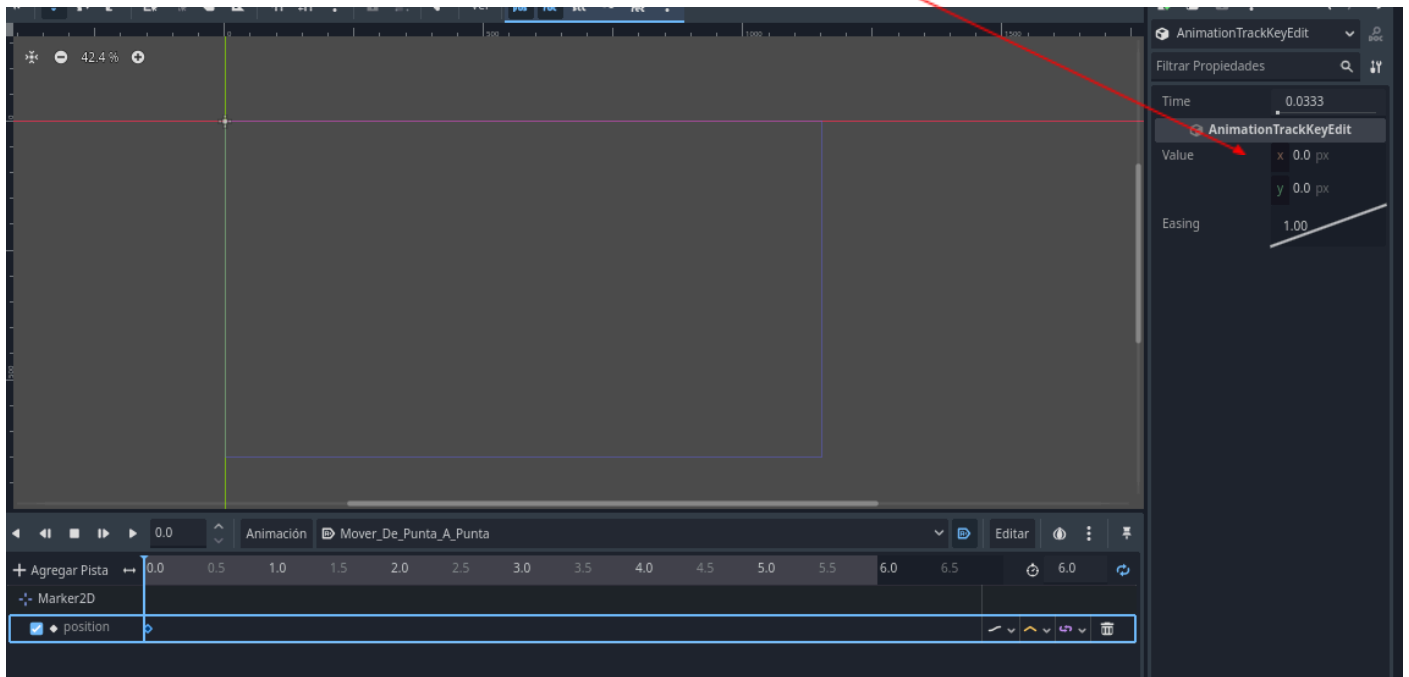


Con estas 2 opciones activadas entonces cuando la escena "Spawn_point" o "punto_de_aparacion" (o como hayan nombrado la escena) se cargue va a reproducir la animación y cuando termine la va a volver a reproducir.

Ahora agregamos las keys dentro de la animación, yo le hago click derecho ahí adentro y me permite colocar la clave o key

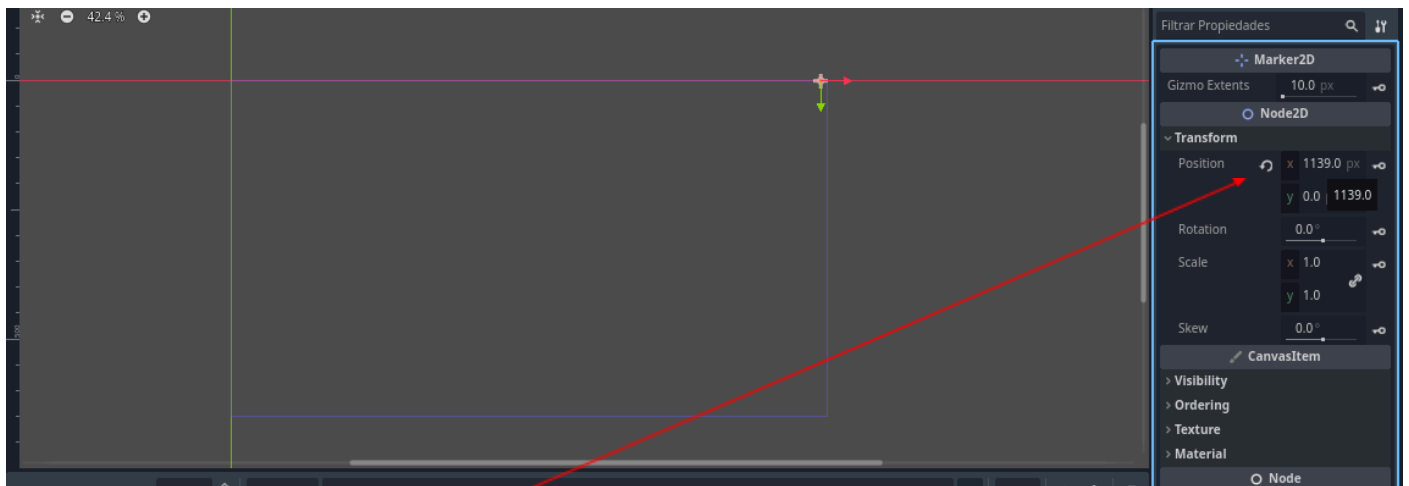


Ahora si cliquean ese punto que se les creo van a poder ver las propiedades que nos permite cambiar en la animación

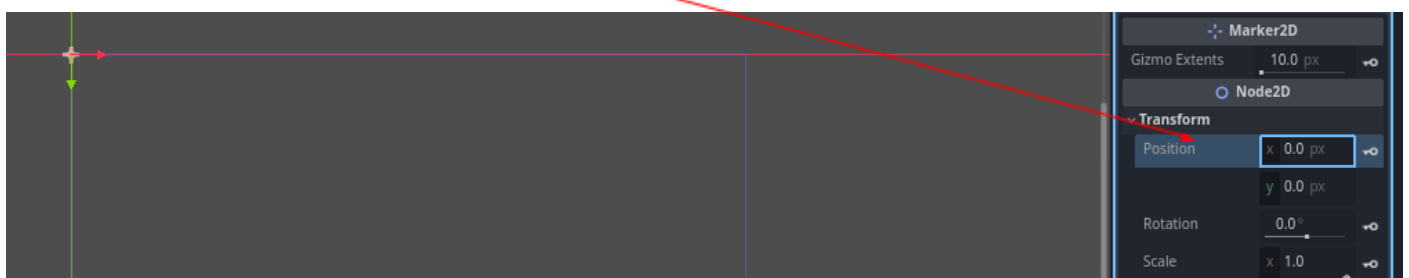


Esto quiere decir que cuando se reproduzca la animación (que es apenas se carga la escena) en el segundo 0.033 la posición de Marker2D va a ser X= 0 , Y=0, el punto de origen, **cambiamos el time a 0** para evitar advertencias mas adelante, y ahora necesito crear otra key mas adelante en la animación que lo lleve a la punta del viewport.

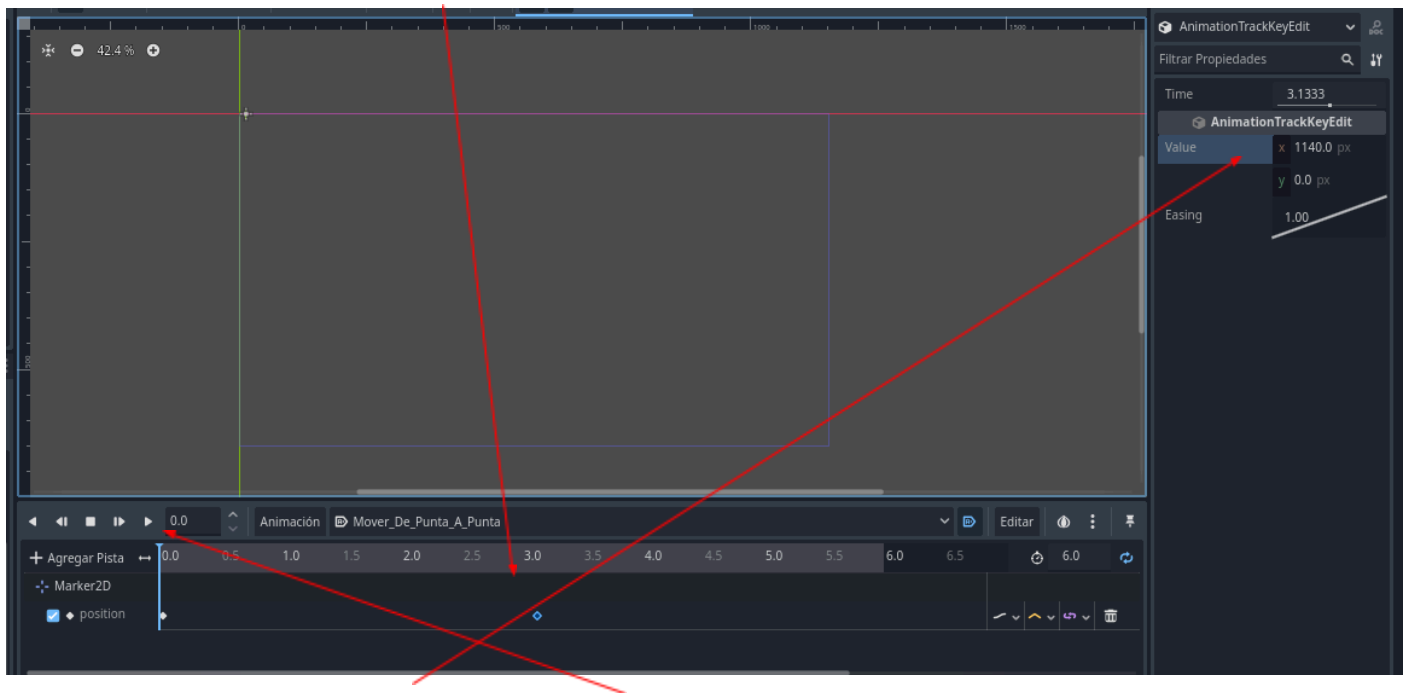
Yo para saber hasta donde llevarla agarre el Marker2D y lo arraste hasta la punta del ViewPort manualmente



Entonces fijándome la propiedad transform del nodo puedo ver que tengo que llevar el nodo a la posición X = 1140 maso menos. Vuelvo a dejar el Nodo en la posición 0,0 por las dudas.



Y ahora en la animación agrego una key a mitad maso menos de la animación, en mi caso a los 3 segundos.

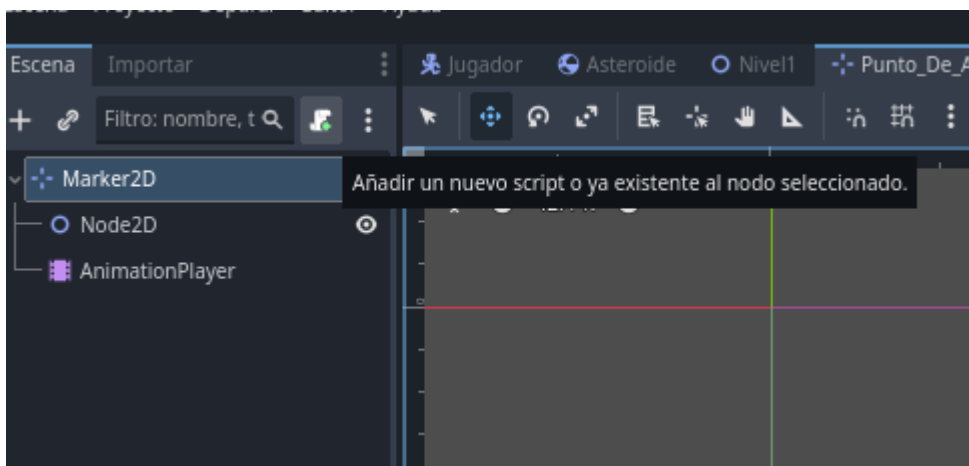


Y seleccionando el nuevo punto le cambio la propiedad X a 1140. Ahora si le dan play a la animación deberían ver como el nodo se mueve solo por ahí arriba. También debería funcionar si ponen play a la escena, pero al no tener textura no van a ver como se mueve.

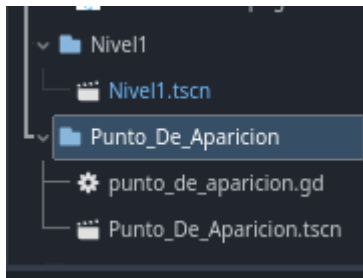
El valor de la propiedad X va a variar según el ancho de su viewport, así que tienen que ver eso primero antes de asignarle el valor, puede que el mío les quede muy ancho o se quede corto.

Ahora tenemos una escena que se mueve por la parte superior de un lado a otro, de qué nos sirve eso? Sirve como punto de aparición para nuestro asteroide, porque sería aburrido que solo aparezca en un lugar. Entonces al tener una escena que se mueva podemos crear una escena asteroide cada cierto tiempo y la escena asteroide como la tenemos armadas simplemente cuando aparezca se va a dejar caer. Esto lo vamos a hacer con un script.

Así que le agregamos un nuevo script a nuestro nodo padre o raíz de la escena `Spawn_point` o `punto_de_aparicion`.

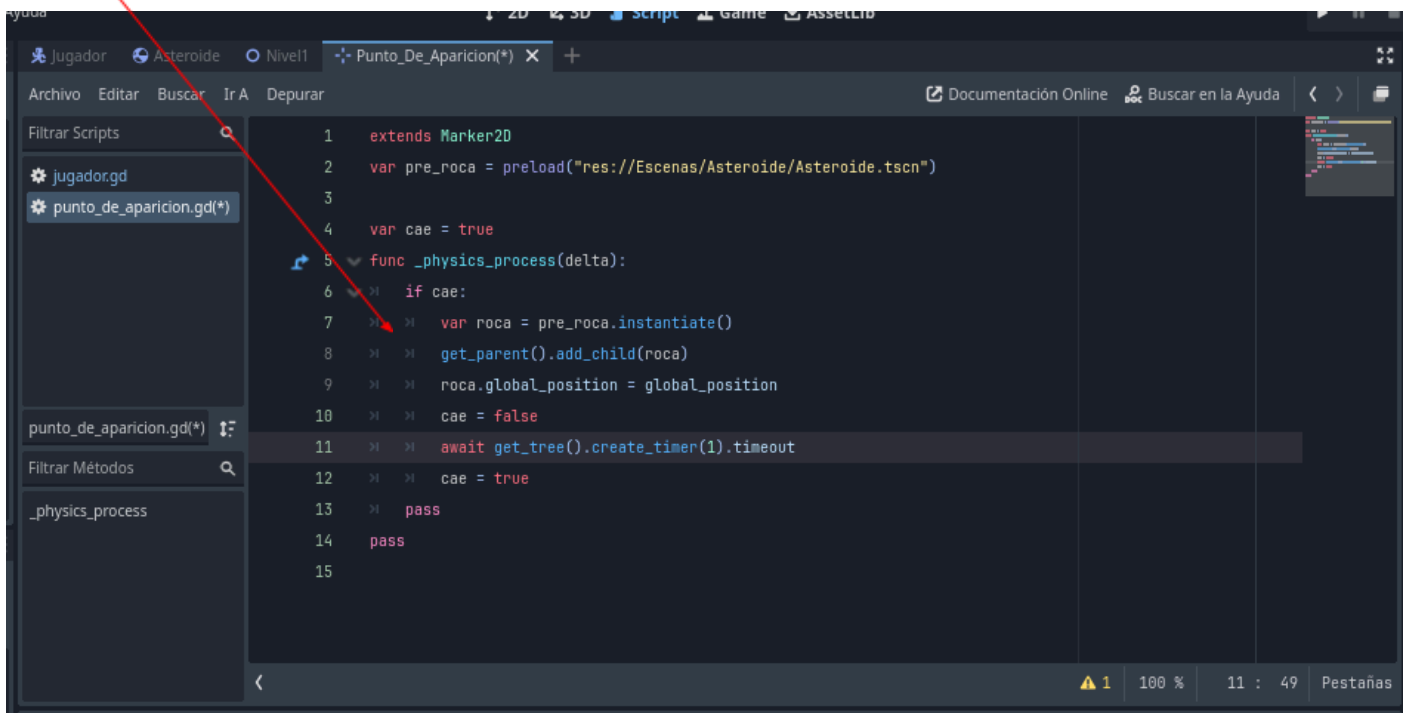


Con el mismo nombre y debería crearse en la carpeta de la escena



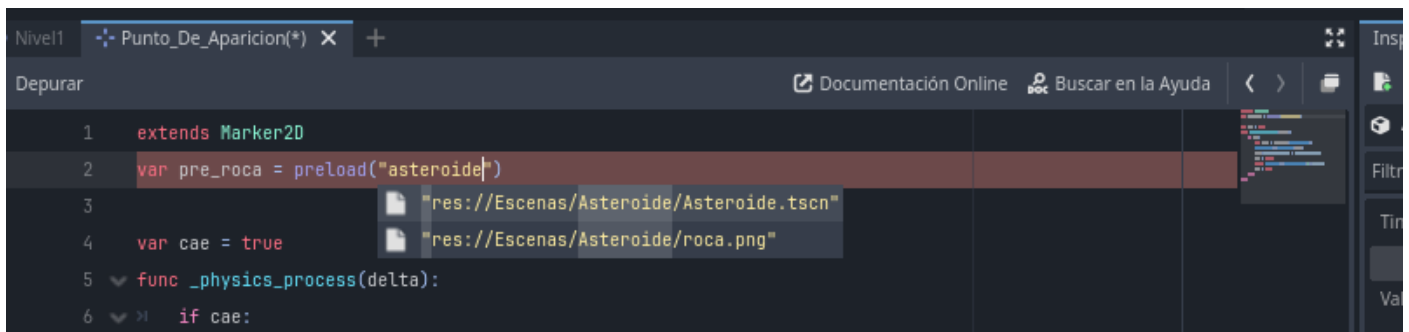
Si copian y pegan lo del tutorial de la puso se va a romper todo.

El código final les debería quedar asi, recuerden utilizar el TAB para crear la sangría que se ve reflejada con esos iconos



Ahora que hace cada cosa de eso?

Se crea una variable pre_roca con un valor raro, esa línea de código "preload" es parte de godot y básicamente lo que hace es guardar una escena en la variable, para eso necesita la ruta de donde esta la escena. Por eso es importante tambien tener cada escena en carpetas diferentes y saber reconocer los nombres



Cuando empiezas a escribir el nombre te debería ayudar a encontrar la escena que buscas. Haciéndole doble clic lo auto completa. Fijense la terminación .tscn que significa que es una escena. Todo ese chocio debería ir entre paréntesis y dentro de comillas, si no les sale en amarillo u otro color diferente algo están haciendo mal.

Luego se crea una variable cae con un valor true. Esto es simplemente una variable booleana que puede ser verdadera o falsa, aca arranca definiéndola como verdadera o true.

Luego en la función `_physics_process(delta)` que es la que se ejecuta una banda de veces por segundo van a hacer varias cosas.

Primero hay un condicional que va a chequear si `cae = true`. (que aunque no este escrito el programa lo interpreta asi) Si esa condición se cumple va a ejecutar el bloque siguiente

```
var roca = pre_roca.instantiate()
get_parent().add_child(roca)
roca.global_position = global_position
```

Que va a hacer eso? Medio compliado de explicar pero básicamente crea una variable y le pone de valor la escena que ya precargamos (el asteroide) luego la pone en la posición del nodo padre(el marker2d que esta dando vueltas)

```

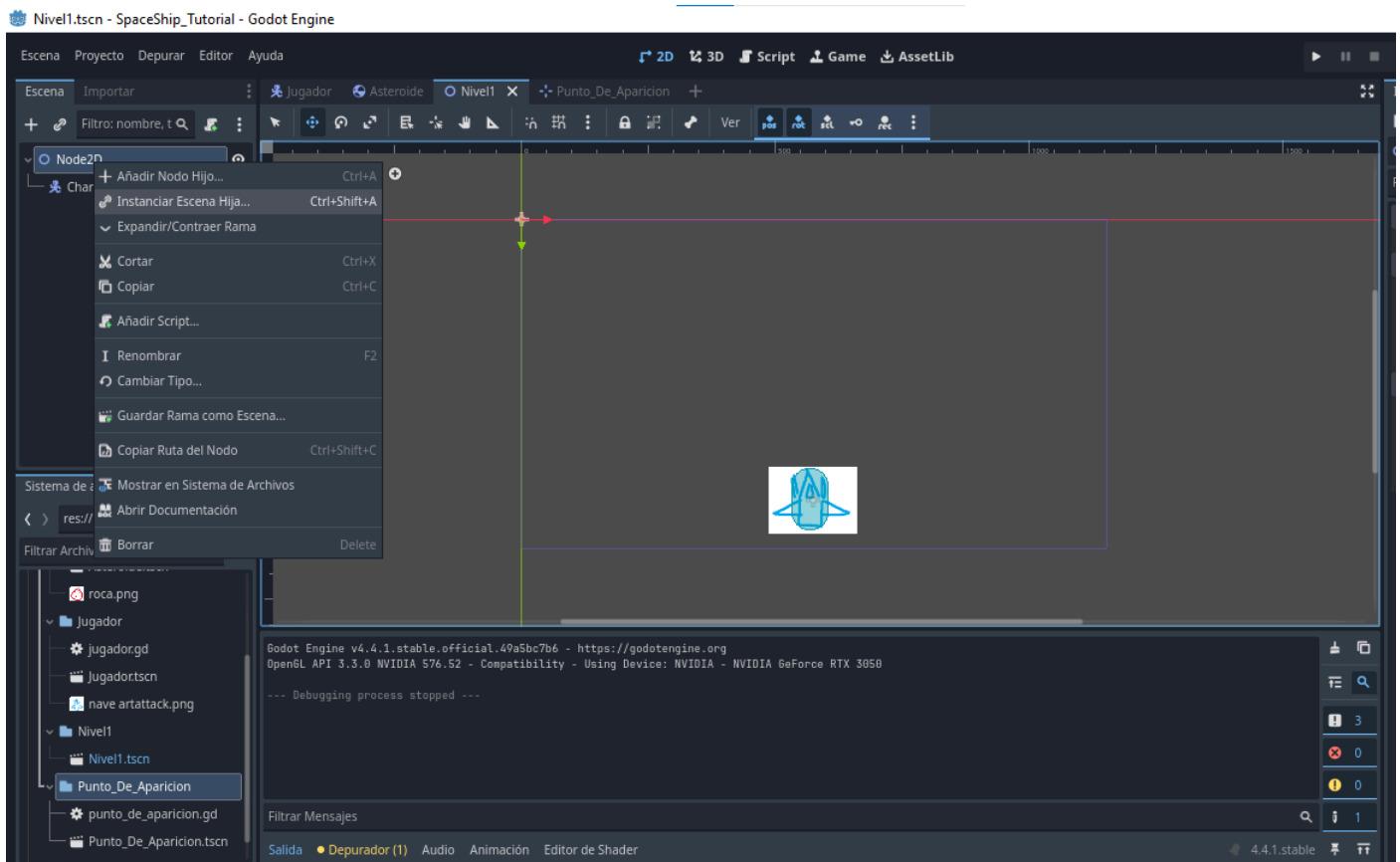
In [ ]: cae = False
In [ ]: await get_tree().create_timer(1).timeout
In [ ]: cae = True
In [ ]: pass
In [ ]: pass

```

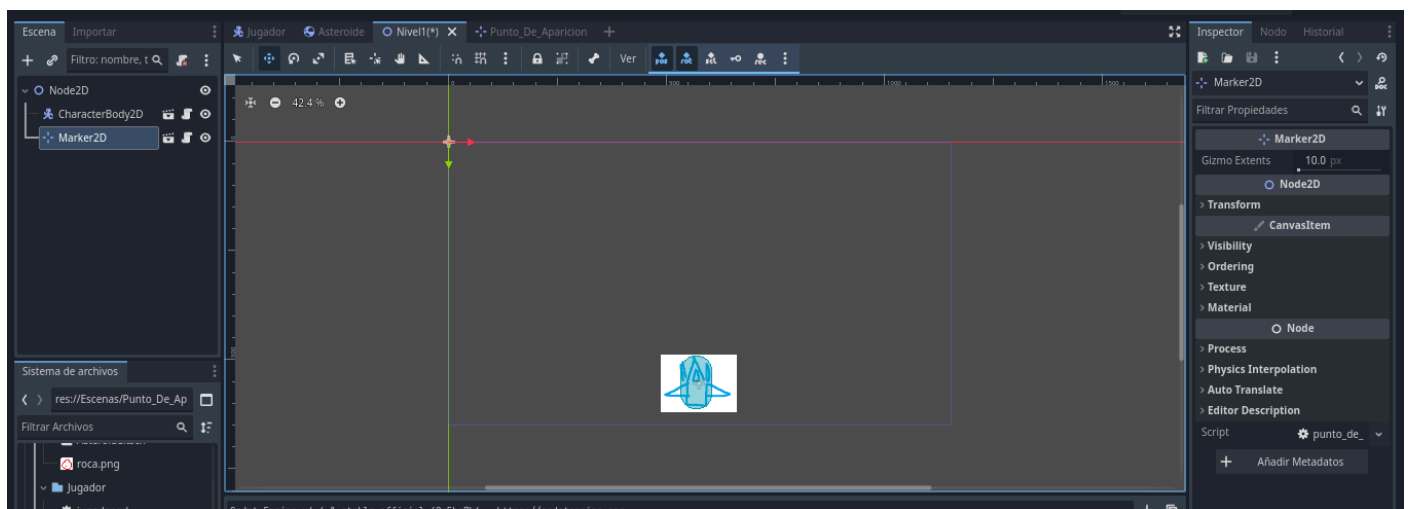
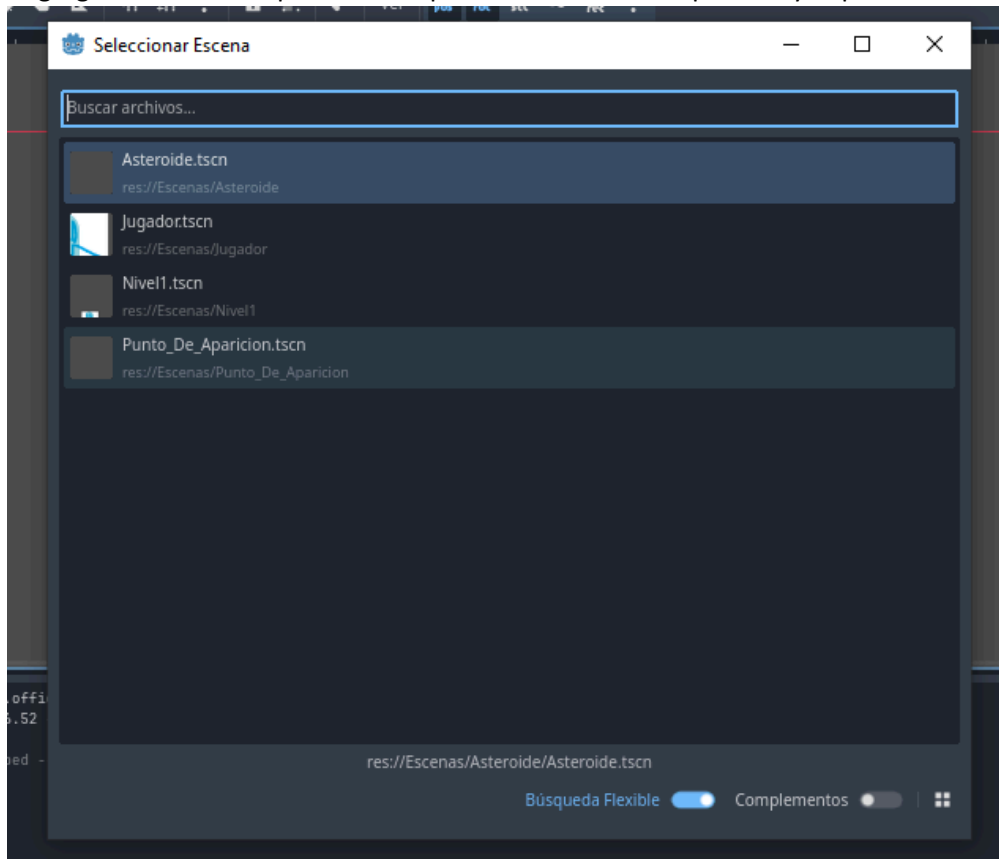
Luego la ultima parte, una vez termino de crear el asteroide y colocarlo, pone la variable cae como falso (para que no se creen un millón de piedras por segundo porque recuerden que para crear una piedra se tiene que cumplir la condición de que “cae” sea verdadera o true) luego crea un temporizador de 1 segundo que al finalizar va a volver a poner la variable cae como verdadera.

En resumen lo que hace el código es crear una escena asteroide cada 1 segundo en la posición del nodo donde se encuentra el script. Esto si lo guardan y ejecutan la escena(F6 para ejecutar la escena actual, que es diferente a la escena principal) de “spawn_point” o “punto_de_reaparicion”. Deberían ver como se crean ahora varias escenas de asteroides que una vez creadas se ven afectadas por la gravedad y caen.

Ahora podemos poner esta nueva escena que crea asteroides en diferentes puntos dentro de la escena principal Nivel1 de la misma forma que pusimos la escena jugador, podemos arrastrar desde recurso o hacemos click derecho sobre el nodo padre o raíz de la escena Nivel1



Y agregamos la escena punto de reaparición o el nombre que le hayan puesto.



Ahora si, si le dan F5 para ejecutar la escena principal, deberían poder mover la nave del jugador y ver los asteroides caer y al tener colision van a ver como los podes chocar.

Con esto termino por ahora por falta de tiempo, espero les sirva y despues veo si armo otra parte, tengan en cuenta que hay muchas otras cosas que se pueden agregar y pulir y agregar por ejemplo la condición de derrota y victoria, el fondo para el nivel1, los sonidos y música. La habilidad de disparar un laser que destruya los asteroides. Un borde en el viewport para evitar que el jugador salga de los limites. Y una forma de eliminar los asteroides que pasan de largo para que no consuman memoria. Todo eso esta dentro del tutorial de la upso que fui siguiendo, el único error que encontré fue cuando creaba el laser. Pero el resto lo pude seguir.