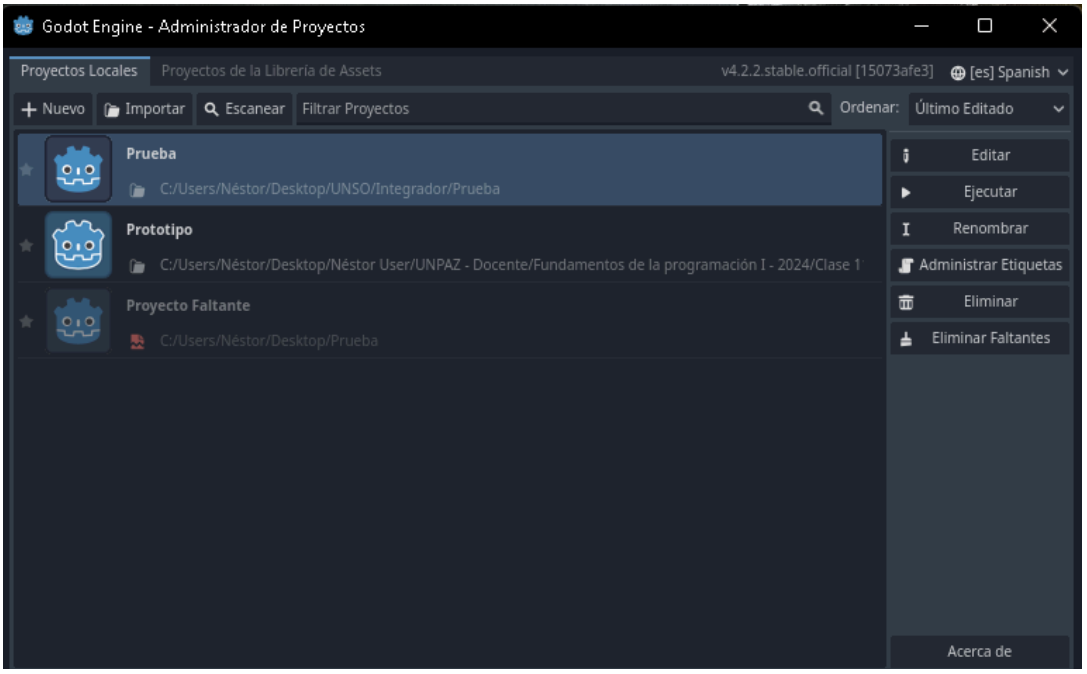
TUTORIAL 2: Spaceship

Primero programaremos el núcleo de un videojuego estilo Spaceship que tiene por mecánica principal el movimiento en dos direcciones y disparos de una nave. Luego crearemos un enemigo y una condición de victoria y derrota. Así habremos hecho un prototipo de videojuego más.

Creación de un proyecto

Utilizaremos GODOT 4.

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

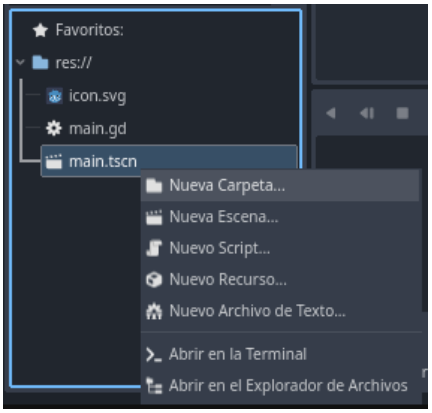


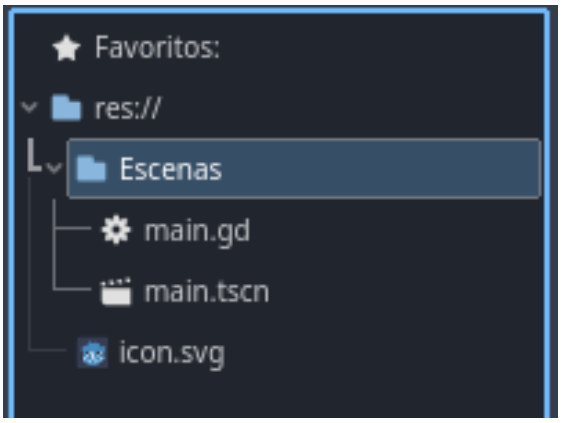
Creación de una escena principal

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

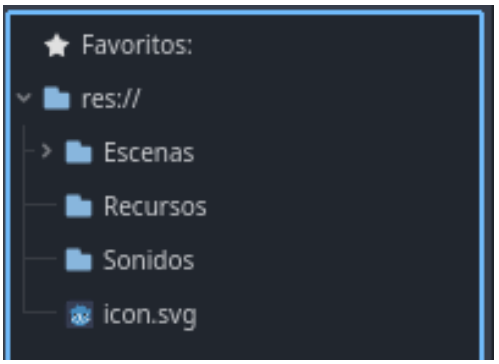
Crearemos una nueva escena con un nodo de control Node2D, lo renombramos Main y adjuntamos un script. Luego guardaremos el proyecto con Ctrl+S.

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





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

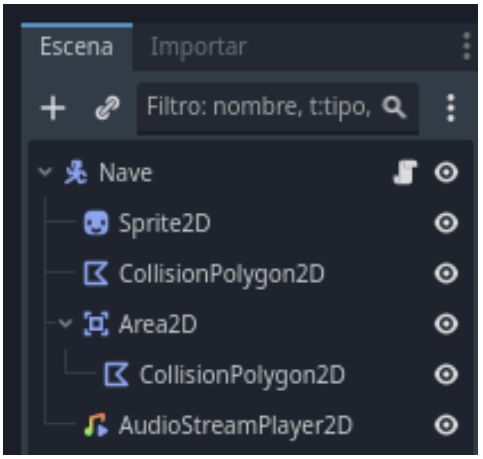


Luego sumaremos todos los recursos necesarios al sistema de archivos:

BubbleBobble.ttf – Fondo.png - Nave.png - Rayo.png - Roca.png - Explosion.png - Laser_sound.ogg

2- Creación de Nave.

Para la creación de la nave utilizaremos el siguiente árbol de nodos:



Al nodo Nave le añadiremos además un script con el siguiente código:

```
extends CharacterBody2D

const SPEED = 300.0

func _physics_process(delta):
	var direction = Input.get_axis("ui_left", "ui_right")
	if direction:
		velocity.x = direction * SPEED
	else:
		velocity.x = 0

	move_and_slide()
	pass
```

No nos olvidemos de guardar la nave como una escena que llamaremos igual que el nodo principal, ***nave***. Y siguiendo las buenas prácticas creamos el método Move().

```
extends CharacterBody2D
```

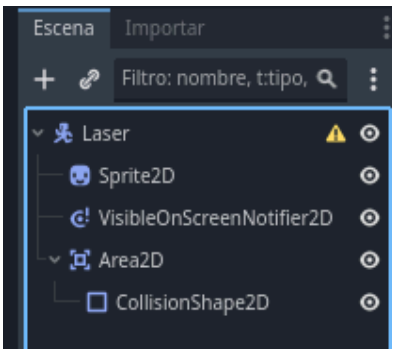
```
const SPEED = 300.0

func _physics_process(delta):
	Move()
	pass

func Move():
	var direction = Input.get_axis("ui_left", "ui_right")
	if direction:
		velocity.x = direction * SPEED
	else:
		velocity.x = 0

	move_and_slide()
	pass
```

Dentro del Main debemos crear dos nodos StaticBody2D, añadirles un CollisionShape2D y colocarlos en ambos laterales de la pantalla de modo que cuando nuestra nave se acerque, colisione y no salga de la pantalla. Renombramos a los StaticBody2D Pared1 y Pared2.

3- Creación de Láser.

Ahora vamos a hacer que nuestra nave dispare un láser, para esto vamos a crear otra escena cuyo nodo principal será un CharacterBody2D llamado ***Laser*** al igual que la escena.

El árbol de nodos debe quedar de la siguiente manera:



Agregaremos un script al nodo Laser con el siguiente código:

```
extends CharacterBody2D

func _physics_process(delta):
	position.y -= 10
	pass
```

Ahora seleccionemos el VisibleOnScreenNotifier2D y luego desplegamos la pestaña Nodo que está junto a la del Inspector. Esto nos permitirá ver el panel de señales y grupos. En el panel de señales seleccionamos la función screen_exited(), damos click al botón conectar que está debajo de todo y en la ventana que se nos aparece vamos a seleccionar nuestro nodo Laser y nuevamente hacemos click en el botón conectar.

Esto habrá añadido una función a nuestro código que modificaremos y debe quedar de la siguiente manera:

```
extends CharacterBody2D

func _physics_process(delta):
	position.y -= 10
	pass
func _on_visible_on_screen_notifier_2d_screen_exited():
	queue_free()
	pass
```

La función *_on_visible_on_screen_notifier_2d_screen_exited()* detecta si nuestra escena, el láser, ha salido fuera de la pantalla; si esto es así, mediante el método queue_free() le decimos que libere la escena. De esta forma no tendremos elementos vivos dentro de la escena pero que no podamos ver por estar fuera de la pantalla, no cumplen ninguna función y que ocupan procesamiento en la RAM.

Ya tenemos la nave y el láser, ahora debemos hacer que la nave dispare a este láser. Para esto vamos nuevamente a la escena nave.

Agregaremos a nuestro código dos líneas que son, una variable de control y una línea que precarga la escena ***laser*** dentro de la escena ***nave*** y la guarda en una variable para reconocerla y podamos trabajar con ella desde donde estamos.

```
var disparo = true
var pre_laser = preload("res://Escenas/laser.tscn")
```

Agregaremos un método llamado Shot() dentro del _physics_process(delta), que al detectar que se presiona la flecha arriba, mediante una alternativa condicional, instancie al laser dentro de la nave.

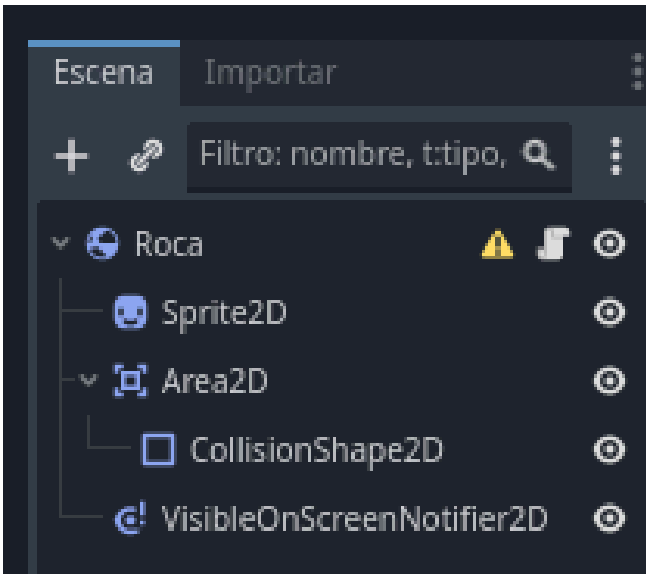
```
func _physics_process(delta):
	Move()
	Shot()
	pass

func Shot():
	if Input.is_action_just_pressed("ui_up") and shot:
		var laser = pre_laser.instantiate()
		get_parent().add_child(laser)
		laser.position.x = position.x
		laser.position.y = position.y - 50
		shot = false
		await get_tree().create_timer(0.5).timeout
		shot = true
	pass
```

Este código instancia al láser como hijo de la nave y lo posiciona en la posición de la nave con una leve diferencia en el eje Y para que parezca que el laser sale de la punta de la nave. Con la variable de control y un cronómetro controlamos la cadencia de los disparos.

Con esto terminamos nuestro núcleo, ahora proseguiremos con el prototipo final.

4- Creación de un enemigo.

Nuestro enemigo será una roca y para esto utilizaremos lo aprendido en las clases anteriores. La crearemos como una escena aparte, el árbol de nodos sería el siguiente:



Añadiremos también un script a la roca para usar más adelante.

Ahora haremos que la roca aparezca cada 1 segundo en una posición comprendida sobre el margen superior de la pantalla, es decir fuera de la pantalla, y del ancho de la pantalla. Para esto vamos a añadir en el Main un nodo llamado Marker2D. A este nodo vamos a llamarlo Spawn_1 y le agregamos un AnimationPlayer como hijo.

Posicionamos al Spawn_1 en las coordenadas X = 0, Y = -32. Luego creamos una animación que llamaremos Move y que dure 6 segundos. En el segundo 0 las coordenadas son las iniciales, X = 0, Y = -32. En el segundo 3 las coordenadas son X =720 (o el ancho de la pantalla que esté configurado), Y = -32. Debemos activar los botones ***Reproducción automática al Cargar*** y ***Loop de Animación***.

Añadiremos también un script al Spawn_1 con el que estaremos precargando la roca dentro de Spawn_1, instanciándola en la posición del nodo y controlando la cadencia.

```
extends Marker2D

var pre_roca = preload("res://Escenas/roca.tscn")
```

```
var cae = true

func _physics_process(delta):
	if cae:
		var roca = pre_roca.instantiate()
		get_parent().add_child(roca)
		roca.global_position = global_position
		cae = false
		await get_tree().create_timer(1).timeout
		cae = true
		pass
	pass
```

Nos falta ahora crear las condiciones del videojuego, que el láser desaparezca cuando toca una roca, que la roca desaparece cuando toca el láser y que la nave desaparezca cuando toca una roca.

5- Programando condiciones.

Lo que debemos lograr es que cada escena reconozca cuándo y con quién está colisionando. En este caso vamos a hacerlo mediante detección de áreas y no de colisiones. Para esto vamos a utilizar las Area2D con sus correspondientes CollisionShape2D de la nave, la roca y el láser.

Haciendo caso omiso a la advertencia de los nodos que acusan haber quedado sin un CollisionShape2D, lo primero que haremos con estas tres escenas es añadir o etiquetar sus áreas dentro de un grupo. Para esto seleccionamos el Area2D de cada escena y vamos a la pestaña Nodo que está junto a la del Inspector. Hacemos click en Grupos y en este panel creamos un nuevo grupo con el nombre de cada escena.

Ahora vamos a Nave, seleccionamos el Area2D desplegamos la pestaña Nodo y en el panel de Señales conectamos la señal area_entered(area: Area2D) con el script de la nave.

En el script de la nave modificamos la función que se creó para que nos quede de la siguiente manera:

```
func _on_area_2d_area_entered(area):
	if area.is_in_group("Roca"):
		queue_free()
	pass
```

Haremos lo mismo para la roca:

```
func _on_area_2d_area_entered(area):
	if area.is_in_group("Laser") or area.is_in_group("Nave"):
		queue_free()
	pass
```

Y para el láser:

```
func _on_area_2d_area_entered(area):
	if area.is_in_group("Roca"):
		queue_free()
	pass
```

6- Sonidos y animaciones.

Agregamos un nodo AudioStreamPlayer2D al laser.

Agregamos el recurso de audio al stream y tildamos la casilla Autoplay.

Hacemos lo mismo para agregar música al juego. El recurso lo creamos dentro del Main.

Para la animación nos vamos a la roca y añadimos un nodo AnimationPlayer al cual le añadiremos dos animaciones “Idle” y “Explotion”.

A la Idle simplemente le crearemos una key de textura en el Sprite2D y marcaremos ***Reproducción Automática al Cargar***.

Para Explotion seteamos la duración y el ajuste en 1 y 0.1; cargaremos al Sprite2D el recurso en texture y creamos una key. Luego setamos el Hframe en 10 y creamos otra key.

Configuramos la animación creando las llaves necesarias cada 0.1 seg. comenzando por el frame 9 hasta el 0.

Por último seteamos la propiedad “Disable” del CollisionShape2D” y creamos una key.

7- Fondo.

Crearemos una escena nueva con un nodo ParalaxBackground como nodo padre y lo renombramos Fondo. El árbol de la escena debe quedar así:

Agregamos la textura al Sprite2D, ajustamos y modificamos la propiedad repeat marcando la opción Enabled.

Desplegamos la propiedad Material y seteamos el tipo en Nuevo ShaderMaterial, hacemos click en el material cargado y en la propiedad shader seleccionamos NewShader. Hacemos click en el NewShader para que se habilite el script GLSL al que agregaremos la siguiente secuencia de código:

```
 shader_type canvas_item;

uniform float scroll_speed;

void fragment(){
        vec2 shifteduv = UV;
        shifteduv.y -= TIME * scroll_speed;
        vec4 color = texture(TEXTURE, shifteduv);
        COLOR = color;
}
```

Una vez hecho esto, en la propiedad scroll_speed seteamos la velocidad a la que queremos que el fondo se mueva.

Por último, modificamos la propiedad Mirroring en su eje Y del Paralaxlayer en 648 para este caso.