El desarrollo del prototipo del juego Space Invaders se ha desarrollado de la siguiente manera:

Antes de empezar me gustaría aclarar varias cosas en el código:

```
$AnimationPlayer.play("Destruido")
```

Todo lo que empiece por $ esta cogiendo un nodo dentro del mismo nodo donde se halle el script.

```
signal enemy_bonus_eliminado
```

Signal prepara una señal desde código

```
emit_signal("enemy_bonus_eliminado")
```

Emit_signal emite la señal preparada

```
enemybonus.connect("enemy_bonus_eliminado", Callable(get_parent(),"sumar_puntos_enemy_bonus"))#1° parametro ¿donde
```

connect recibe la señal y a continuación hace la llamada a un método que puede estar ubicado en el mismo script de connect o en otro.

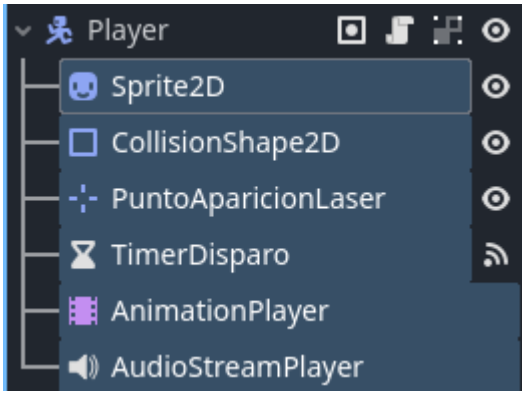
$AudioStreamPlayer.play() sirve para reproducir audio cargado en ese tipo de nodo.

```
@onready var timer_disparo = $TimerDisparar
```

@onready sirve para meter en una variable un nodo presente en el nodo de ese mismo script. Hecha las aclaraciones se empezará con la explicación.

En primer lugar, tenemos un nodo principal que es de tipo Node (está no tiene ubicación por lo que es de las mejores opciones para ser padre de todos los nodos que se irán instanciando en el juego)

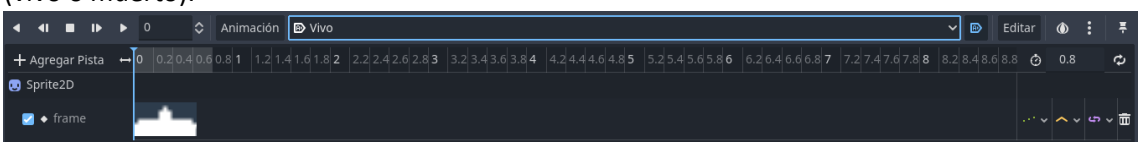
Por otro lado, tenemos al jugador, que es un nodo de tipo CharacterBody2D. Este es un nodo independiente en el juego que tiene como hijos estos otros nodos:

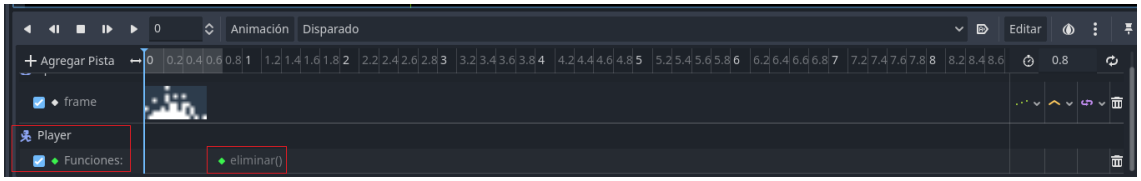


- **Sprite2D** tiene asociado un spritesheet (dos imágenes en un mismo png), 1 de vivo y otra de muerte.

En Hframes y Vframes vamos seleccionando la imagen que queremos añadir al AnimationPlayer y el botón de la llave es para insertarlo en la animación (línea del tiempo).

- **CollisionShape2D** es el collider que detectará la colisión del player cuando lo toquen
- **Maker2D** llamado PuntoAparicionLaser, de donde sale la bala laser del player cuando dispara.
- **Timer** llamado TimerDisparo que calcula el tiempo de disparo entre bala y bala.
- **AnimationPlayer** donde preparamos las 2 imágenes del spritesheet que se utilizarán (vivo o muerto).

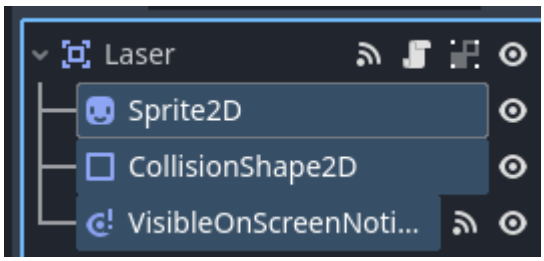




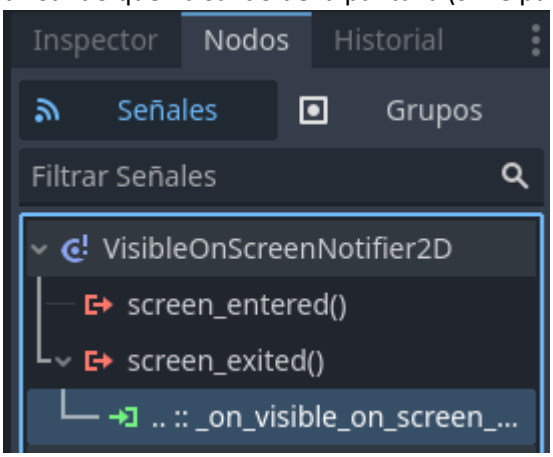
Es importante destacar que desde este nodo se pueden llamar a funciones creadas en el script del nodo padre (Player)

- **AudioStreamPlayer** nodo para añadir el efecto sonoro del disparo cuando se realice. Importante matizar que solo se reproducen audios en formato wav u ogg.

Asimismo, también observamos al nodo laser (la bala del player cuando dispara), este nodo tiene como hijos los siguientes:



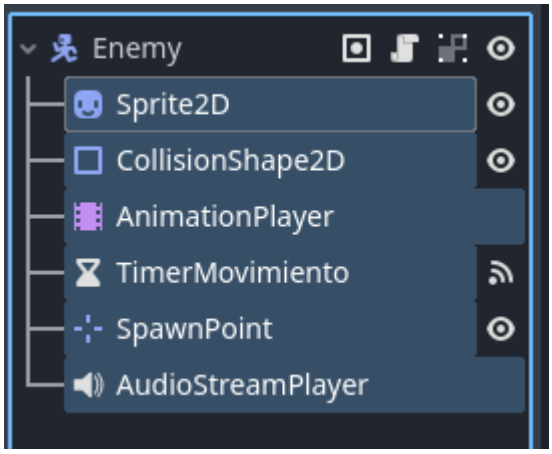
- Sprite2D: Única imagen que representa el láser
- CollisionShape2D para detectar la colisión del láser hacia otros objetos.
- VisibleOnSreenNotifier2D que es un nodo donde envía una señal al mismo láser avisando que ha salido de la pantalla (sirve para eliminarse él mismo)



Es importante señalar que el player esta metido en un grupo llamado “Nave”, esto sirve para que a la hora de tocar el player con un misil del enemigo, en el script del misil poder hacer un filtro mediante grupo y hacer algo solamete con el player (lo mismo sucede con los bloques, se creo un grupo llamado “GrupoBloques” para hacer la rotura del mismo solo a ellos)

```
func _on_body_entered(body: Node2D) -> void:
    if body.is_in_group("Nave") or body.is_in_group("GrupoBloques"):
        body.destruir()
        queue_free() #borrar en memoria
```

Al mismo tiempo tenemos además el nodo Enemy, que es muy parecido al player, presentando los siguientes nodos:

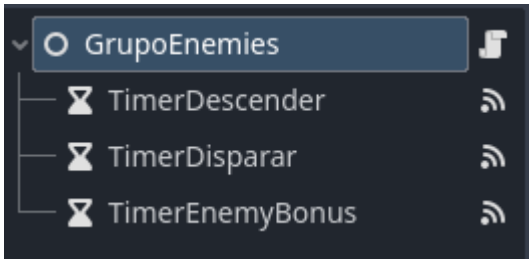


Aquí cabe destacar el nodo hijo TimerMovimiento, que marca el tiempo de espera entre movimiento horizontal del grupo de enemigos.

En este sentido, Enemy esta metido en otro grupo llamado “Aliens”, para hacer ciertas cosas en ellos cuando el láser del player los toca. El nodo láser tiene una señal llama body_entered que nos sirve para cuando el grupo Aliens o GrupoBloques toque con el nodo se realicen cosas.

```gdscript
func _on_body_entered(body: Node2D) -> void:
    if body.is_in_group("Aliens"):
        body.explotar()
        get_parent().remove_child(self) #Voy al padre para que quite un hijo (a este mismo objeto láser). Es el padre e
        queue_free() #Elimina este mismo objeto de la memoria

    elif body.is_in_group("GrupoBloques"):
        body.destruir()
        if !is_queued_for_deletion(): #!a.is_queued_for_deletion() ---> Si no esta en la cola para borrarse
            get_parent().remove_child(self) #Voy al padre para que quite un hijo (a este mismo objeto láser). Es el pad
            queue_free() #Elimina este mismo objeto de la memoria
```

GrupoEnemies es un nodo de tipo Node muy especial, porque aquí dentro se instanciarán todos los enemigos (para tenerlos en un nodo organizados) y donde se pueden hacer ciertas funciones en grupo como calcular el tiempo que tardan en descender, tiempo de disparo o tiempo de spawn del enemigo de bonus.



Cada contador tiene una señal llamada timeout() asociada al script del nodo GrupoEnemies

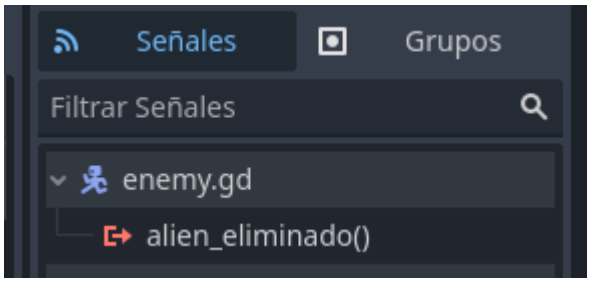
Hacer mención a la función _ready del nodo GrupoEnemies, que se ejecuta nada más empezar el juego, donde hace posible instanciar todos los enemigos que se quieran a través de un bucle for, se le calcula la posición de cada uno, se añade a un array de enemigos para luego eliminarlo de la lista cuando lo toque el láser del player, etc.

```gdscript
func _ready() -> void:
	$TimerEnemyBonus.wait_time = randf_range(10.0,20.0)
	$TimerEnemyBonus.start()
	for j in range(4):
		lista_enemies.append([])
		for i in range(8):
			var enemy = Enemy.instantiate()
			enemy.global_position = Vector2(40 + 20 * i,40 + 20 * j)
			self.add_child(enemy)
			lista_enemies[j].append(enemy)
			enemy.connect("alien_eliminado", Callable(self,"eliminar_alien"))
			enemy.connect("alien_eliminado", Callable(get_parent(),"sumar_puntos_alien"))#1° parametro ¿donde esta la s
```

En este punto aparece por primera vez la sintaxis enemy.connect… ¿Qué es esto?, no es más que la creación de nuestra propia señal. Para ello hemos abierto el script del nodo enemy y hemos puesto lo siguiente:

```gdscript
14	signal alien_eliminado
```

Alien_eliminado será el nombre de la señal que queremos crear por código.




```gdscript
func eliminar():
	emit_signal("alien_eliminado", self)
	get_parent().remove_child(self) #Voy al padre para que quite un hijo (a este mismo objeto enemigo). Es el padre el
	queue_free() #Elimina este mismo objeto de la memoria
```

Con emit_signal(“alien_eliminado”, self) estamos emitiendo esa misma señal para que la reciba alguien pasando como parámetro ese mismo objeto enemy (self). Esta misma señal la recibirá el nodo de grupo_enemies cuando se ejecute la función eliminar().

Y en GrupoEnemies teníamos esto:

```
func _ready() -> void:
    $TimerEnemyBonus.wait_time = randf_range(10.0,20.0)
    $TimerEnemyBonus.start()
    for j in range(4):
        lista_enemies.append([])
        for i in range(8):
            var enemy = Enemy.instantiate()
            enemy.global_position = Vector2(40 + 20 * i,40 + 20 * j)
            self.add_child(enemy)
            lista_enemies[j].append(enemy)
            enemy.connect("alien_eliminado", Callable(self,"eliminar_alien"))
            enemy.connect("alien_eliminado", Callable(get_parent(),"sumar_puntos_alien"))#1° parametro ¿donde esta la s

func eliminar_alien(a):
    for fila in lista_enemies:
        for i in range(len(fila)-1): #i empieza en 0, len = longitud de la fila (en este caso es 8), -1 porque la lista
            if(a == fila[i]):
                fila.erase(i)
                #print("eliminado el alien de la lista")
```

Desglosemos el significado del primer método enemy.connect.

enemy: es el enemigo actual.

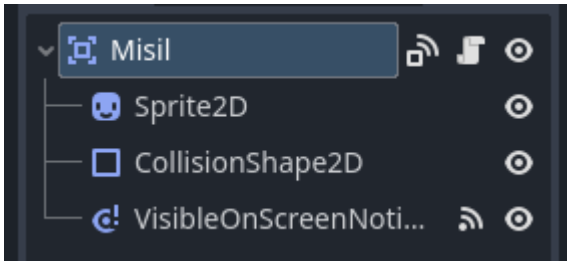
“alien_eliminado”: es el nombre de la señal que acabamos de comentar.

self: es el destinatario de quien va a recibir la señal, que en este caso es en este mismo script

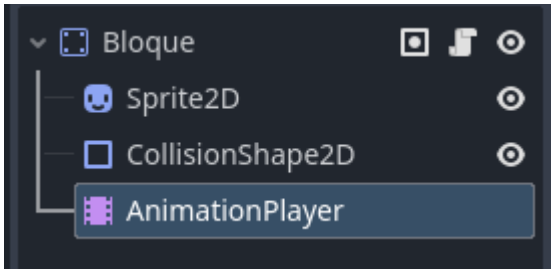
“eliminar_alien”: es el nombre de un método que esta en este mismo script que recibe como parámetro (a), es decir, el enemigo.

Si nos fijamos el otro connect tiene un parámetro llamado get_parent(), esto significa que el destinatario de la llamada de dicha señal será el padre de este mismo objeto GrupoEnemies, en este caso sería main.

El nodo Misil que instancia el enemigo al disparar es lo mismo que el nodo láser que instancia el player al disparar.

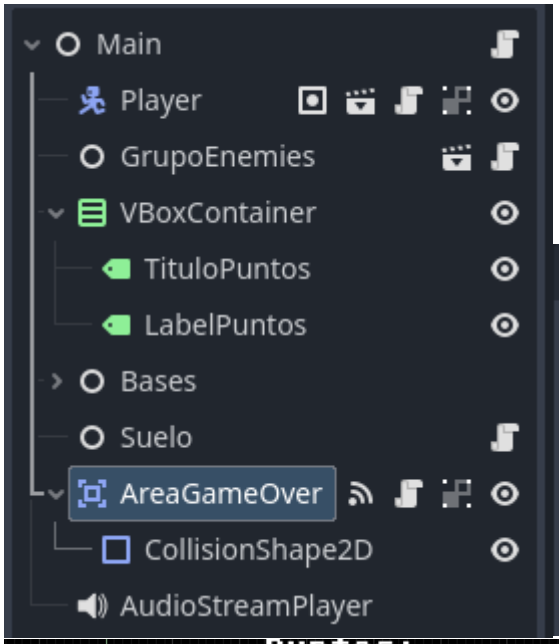


El nodo bloque tiene presenta los siguientes nodos hijos:



Aquí se resalta que hay 2 sprites, uno de normal y otro de dañado, cuando el bloque es dañado por los enemigos el Sprite se cambia de normal a dañado, para tener un mejor control sobre eso se creó un grupo para los bloques llamado “GrupoBloques”.

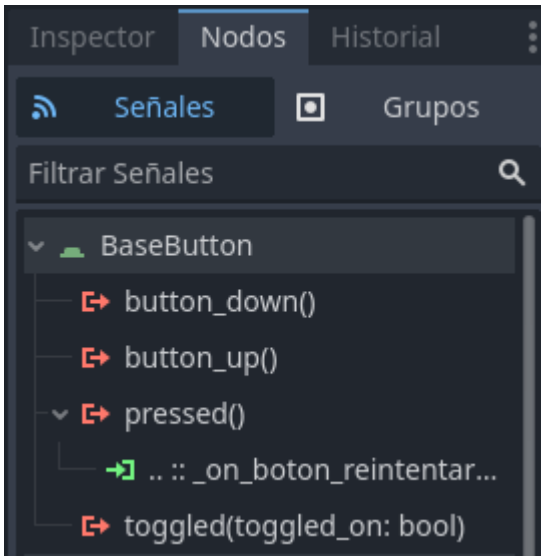
Del mismo modo, se ha realizado una escena a parte llamada GameOver que se ejecuta cuando el player pierde la partida (cuando es dado por el proyectil del enemigo o cuando el mismo proyectil impacta contra la zona baja donde está el player, esto se hizo gracias a un nodo de tipo area2D llamando a una señal de tipo área_entered)




```gdscript
func _on_area_entered(area: Area2D) -> void:
    if area.is_in_group("misiles"):
        get_tree().change_scene_to_file("res://Escenas/game_over.tscn")
```

El grupo “misiles” son los proyectiles de los enemigos.

Ahora solo queda crear un nodo 2D con elementos de canvas y con un botón de reintentar conectado a una señal de cuando se presiona para cargar de nuevo la escena de juego.

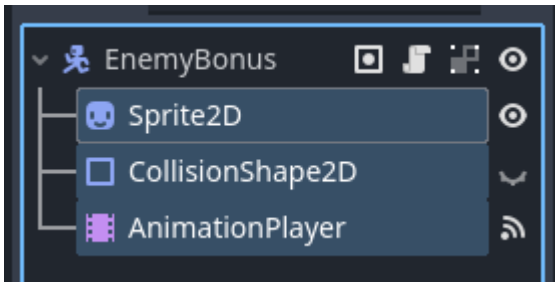



```gdscript
func _on_boton_reintentar_pressed() -> void:
    get_tree().change_scene_to_file("res://Escenas/main.tscn")
```

Para finalizar esta implementado el enemigo del bonus, que da mucha más puntuación que un enemigo normal. Este solo tiene un código para moverse de izquierda a derecha y además emite una señal creada por código como la que vimos anteriormente para ser llamada cuando sea eliminado por el player.

```gdscript
extends CharacterBody2D

signal enemy_bonus_eliminado
var velocidad = 40

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

func _on_animation_player_animation_finished(anim_name: StringName) -> void:
	if anim_name == "Destruido":
		emit_signal("enemy_bonus_eliminado")
		queue_free()

func explotar():
	$AnimationPlayer.play("Destruido")

```

Además, este nodo tiene los siguientes nodos hijos:



Es igual que en enemigo y player, tiene 2 sprite, uno de normal y otro de destruido que se activará cuando el player lo alcance mediante una señal desde el AnimationPlayer