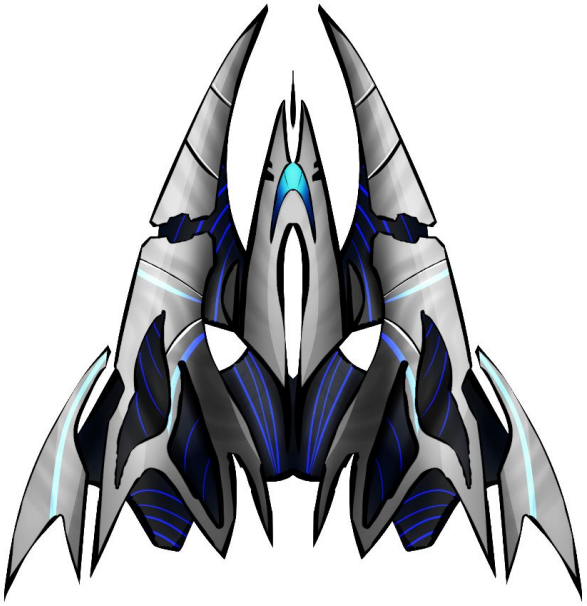
Institut Puig Castellar

Santa Coloma de Gramenet

DODGE



Eric Álvarez Moleón
Xiao Chao Ying
Curso 2n SMX

B) GNU Free Documentation License (GNU FDL)

C) Copyright

Resumen del Projecte

Nuestra idea principal es crear un juego 2D con la aplicación Godot, nuestro personaje principal será una nave y tendremos que esquivar asteroides que irán apareciendo, cuanto más tiempo aguantes mas puntos obtendrás y más posibilidades tendrás de chocarte.
Crearemos varias escenas con sus respectivos scripts para que el juego funcione correctamente.

ÍNDICE

1. Introducción

1.1. Contexto y justificación del Trabajo

Nuestra idea principal es crear un juego 2D simple pero entretenido donde poder retar a tus compañeros para ver quién consigue la mayor puntuación.

El proceso de crear el videojuego ha sido difícil porque no sabíamos bien de qué iba a tratar y no teníamos mucha idea de como utilizar Godot, al final decidimos crear un juego arcade donde el objetivo es intentar esquivar asteroides con una nave.

1.2. Objetivos del Trabajo

Estos son los objetivos que nos hemos marcados :

- Nuestra nave pueda moverse por el espacio.
- Crear 2 tipos de asteroides de diferente tamaño que entren por cualquier lugar y a diferentes velocidades.
- Poner puntuación.
- Cuando pierdas vuelva a la pantalla de inicio.

1.3. Enfoque y método seguido

Queremos aprender a crear un videojuego desde cero, con la ayuda de internet hemos conseguido entender cómo funciona Godot y todo lo necesario para que el juego funcione (scripts, escenas, nodos…)

1.4. Planificación del proyecto

Para este proyecto hemos trabajado a partes iguales, un dia uno se ponía a los mandos del juego mientras el otro ayudaba buscando información, imágenes y ampliando el documento.

1.5. Breve sumario de productos obtenidos

El producto final será un juego para jugar en móvil.

1.6. Breve descripción de los otros capítulos de la memoria

Queremos crear un juego arcade donde una nave pueda moverse libremente por el espacio y esquive los meteoritos que aparecerán aleatoriamente por la pantalla, cada segundo que aguantes conseguirás 1 punto al marcador.

2. Desarrollo

2.1. Godot Engine

Para la creación del juego utilizaremos Godot, dentro de Godot utilizaremos las siguientes herramientas:

Escenas

Una escena se compone de un grupo de nodos ordenados jerárquicamente, dentro de las escenas añadiremos los nodos necesarios para hacer funcionar nuestro juego.

GDSCRIPT

Es el lenguaje que utiliza godot para la programación.

Nodos

Los nodos son elementos fundamentales para crear un juego, un nodo puede realizar una variedad de funciones especializadas

- Tiene propiedades editables.
- Se puede extender (para tener más funciones).
- Se puede añadir a otro nodo como un hijo.

3. Creación del juego

Para crear el juego lo hemos dividido en cuatro escenas:

-Player

-Meteorito

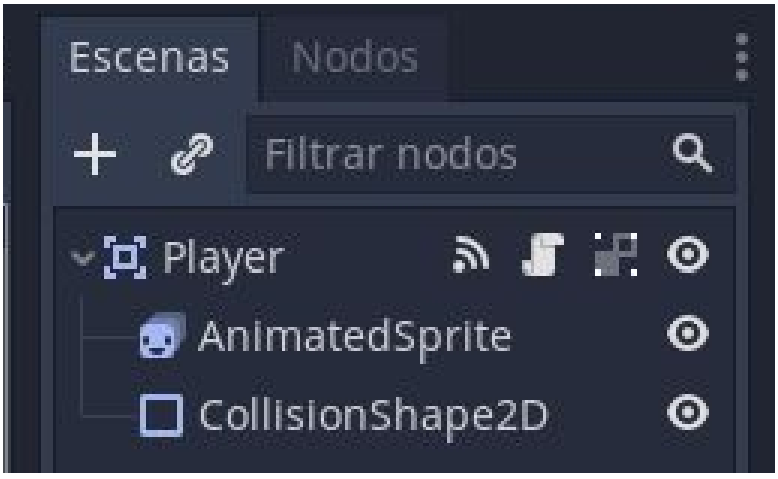
-Interfaz

-Mundo

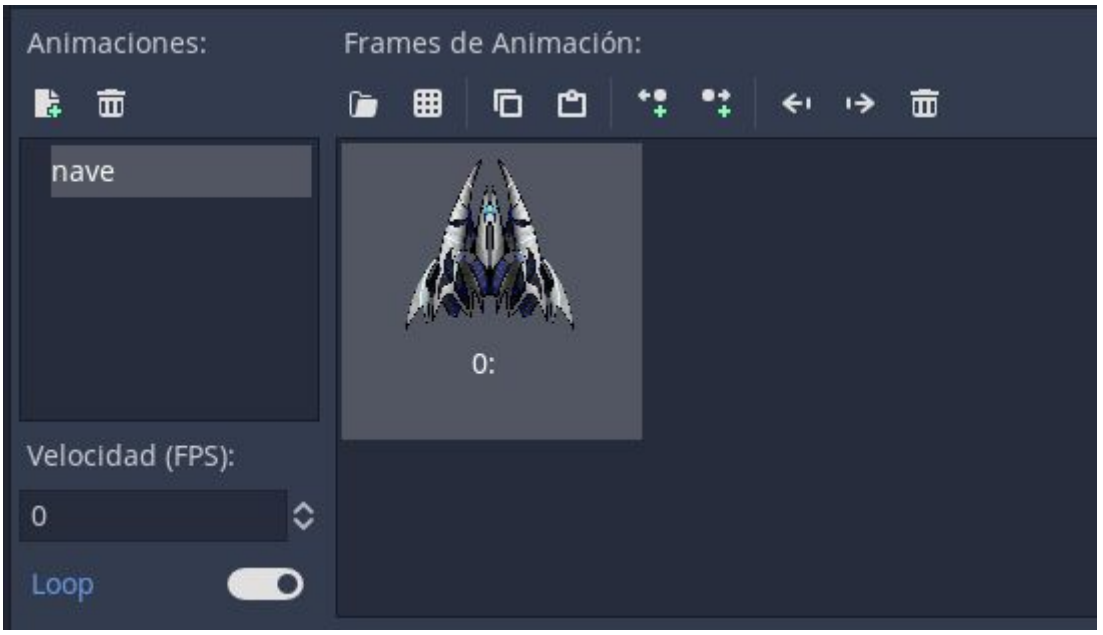
Cada parte tiene sus correspondientes scripts y configuraciones para que funcionen correctamente cuando iniciemos el juego.

3.1 Escena Player

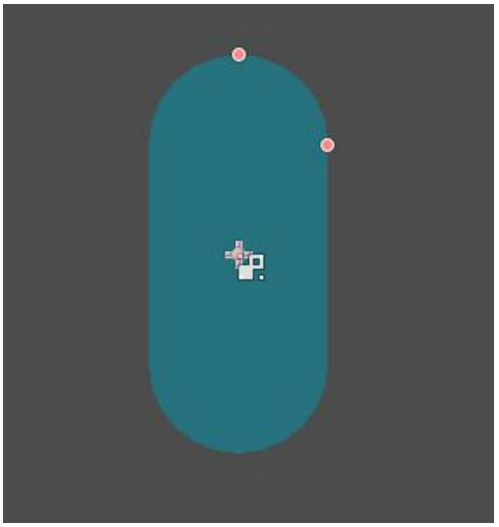
Nuestro juego consta de un personaje principal que será una nave espacial, para crear la nave necesitaremos una escena con los siguientes nodos:



AnimatedSprite: Es donde añadimos el personaje principal, le podemos poner un nombre y ajustar la velocidad de los frames por segundo entre otras configuraciones, en nuestro caso la nave solo tendrá 1 frame.



CollisionShape2D: Es una capa que se utiliza para determinar el cuerpo de un objeto/personaje.



El resultado de la escena anterior será nuestra nave con el área de colisión que le hemos añadido.

Después de haber creado la nave con la zona de colisión tendremos que añadir un script para hacer que la nave pueda moverse y muchas cosas más.

3.1.1 Script de escena Player:

Ahora tenemos que crear todos los scripts necesarios para que la nave se pueda mover en una zona determinada para que no sobrepase los límites establecidos y como el juego trata de esquivar tendremos que añadir que si la nave se choca se reinicie el juego entre otras cosas.

```
1   extends Area2D
2
3   export (int) var Velocidad
4   var Movimiento = Vector2()
5   var limite
6   signal golpe
7
8   func _ready():
9       hide()
10      limite = get_viewport_rect().size
11
12  func _process(delta):
13      Movimiento = Vector2() #Reiniciar el valor
14
15      if Input.is_action_pressed("ui_right"):  #Derecha
16          Movimiento.x += 1
17      if Input.is_action_pressed("ui_left"):   #Izquierda
18          Movimiento.x -= 1
19      if Input.is_action_pressed("ui_down"):
20          Movimiento.y += 1
21      if Input.is_action_pressed("ui_up"):
22          Movimiento.y -= 1
23
24      if Movimiento.length() > 0:  #verificar si se esta moviendo
25          Movimiento = Movimiento.normalized() * Velocidad #Normalizar la velocidad
26
27      position += Movimiento * delta #Actualizar posiciones
28      position.x = clamp(position.x, 0, limite.x)
29      position.y = clamp(position.y, 0, limite.y)
30
31      if Movimiento.x != 0:
32          $AnimatedSprite.animation = "nave"
33
34
35  func _on_Player_body_entered(body): #colision con un objeto
36      hide()
37      emit_signal("golpe")
38      $CollisionShape2D.disabled = true #desactivar colision
39
40  func inicio(pos):
41      position = pos
42      show()   #Mostrar personaje
43      $CollisionShape2D.disabled = false;
```

Este es script completo que hemos creado y ahora lo explicaremos detalladamente.

```
1   extends Area2D
2
3  export (int) var Velocidad
4  var Movimiento = Vector2()
5  var limite
6  signal golpe
7
```

Extends Área2D se refiere el tipo de nodo inicial de la escena.

Export (int) var Velocidad añade una nueva variable dentro del inspector.

Var Movimiento = Vector2() sirve para poder posicionar el personaje en la pantalla para moverlo y **Vector2** es una variable que contiene 2 valores **X** y **Y**, que significa vertical y horizontal.

Hemos añadido la **variable límite** para que tenga un límite y no se salga de la pantalla y **signal golpe** que es una variable para que cuando la nave colisione con un meteorito emita una señal para que ocurra algo después.

```
8  func _ready():
9      hide()
10     limite = get_viewport_rect().size
11
```

Func _ready sirve para cuando se crea un objeto por primera vez en la escena, dices lo que quieres que pase acompañado de **Hide** para que se esconda el personaje cuando inicie el juego.

Limite = get_viewport_rect().size se utiliza para definir un límite que será el tamaño de la pantalla

```
12 func _process(delta):
13     Movimiento = Vector2() #Reiniciar el valor
14
15     if Input.is_action_pressed("ui_right"):  #Derecha
16         Movimiento.x += 1
17     if Input.is_action_pressed("ui_left"):   #Izquierda
18         Movimiento.x -= 1
19     if Input.is_action_pressed("ui_down"):
20         Movimiento.y += 1
21     if Input.is_action_pressed("ui_up"):
22         Movimiento.y -= 1
```

Func _process es lo que va pasar cada momento en el juego por segundo y lo que pones dentro se va repitiendo una y otra vez.

Movimiento = Vector2 sirve para que cuando dejes de pulsar una tecla se pare de mover el personaje y reinicie el valor.

Para que la nave se mueva necesitamos asignar unas teclas y poner el lugar hacia donde se van a desplazar y funciona de la siguiente manera:

```
if Input.is_action_pressed("ui_right"):  #Derecha
    Movimiento.x += 1
if Input.is_action_pressed("ui_left"):   #Izquierda
    Movimiento.x -= 1
if Input.is_action_pressed("ui_down"):
    Movimiento.y += 1
if Input.is_action_pressed("ui_up"):
    Movimiento.y -= 1
```

if Input.is_action_pressed("ui_right"):
Movimiento.x += 1

Si pulsas la flecha de la derecha la nave se moverá hacia la derecha 1 punto, si mantienes el botón pulsado se repetirá el paso hasta que sueltes, los otros tres comandos restantes hacen lo mismo pero en las direcciones que tiene asignadas.

```
if Movimiento.length() > 0:  #verificar si se esta moviendo
    Movimiento = Movimiento.normalized() * Velocidad #Normalizar la velocidad

position += Movimiento * delta #Actualizar posiciones
position.x = clamp(position.x, 0, limite.x)
position.y = clamp(position.y, 0, limite.y)

if Movimiento.x != 0:
    $AnimatedSprite.animation = "nave"
```

if movimiento.length() > 0 si la variable movimiento es mayor que 0 significa que se está moviendo y el comando que incluye es para normalizar a la velocidad y además, necesitamos añadir **position += Movimiento * delta**, para que el juego sea consistente y funcione igual en todos los ordenadores y que la velocidad sea la misma.

El comando **position.x = clamp(position.x, 0, limite.x)** funcionara con el comando **Limite = get_viewport_rect().size** que hemos mostrado anteriormente, **clamp** en esta línea significa clavar en el punto inicio de horizontal de la pantalla que es 0 hasta el final, el otro es lo mismo pero vertical para que no sobrepase los límites.

if Movimiento.x != 0 → si el movimiento x no es igual a 0.

$AnimatedSprite.animation = "nave" → El signo dolar es para buscar el nombre del nodo que le ponemos y busca la animación “nave”.

El comando anterior significa que si el personaje se está moviendo se cambiará a la animación nave, pero en nuestro caso no cambiará nada.

```
35 func _on_Player_body_entered(body): #colision con un objeto
36     hide()
37     emit_signal("golpe")
38     $CollisionShape2D.disabled = true #desactivar colision
39
40 func inicio(pos):
41     position = pos
42     show()   #Mostrar personaje
43     $CollisionShape2D.disabled = false;
```

Func _on_player_body_entered(body): Es el comando que se activará cuando se choquen las colisiones.

Hide significa que si se chocan que se esconda el personaje.

Emit_signal (“golpe”) emite la señal cuando reciba un golpe.

$CollisionShape2D.disabled = true significa que se desactive la colisión cuando reciba un golpe para que solamente reciba uno.

Func inicio (pos): Esto funciona cuando el juego se inicie y colocara el personaje en el punto de inicio, cuando pierdas la partida volverá a colocar el personaje en el centro en vez de donde ha muerto.

Position = pos es para pasar el valor al nombre pos. (o el que pongas en la función).

Show es para que volvamos a mostrar el personaje.

$CollisionShape2D.disabled = false activa la colisión que hemos desactivado anteriormente cuando se emitía la señal de golpe

3.2 Escena Meteorito

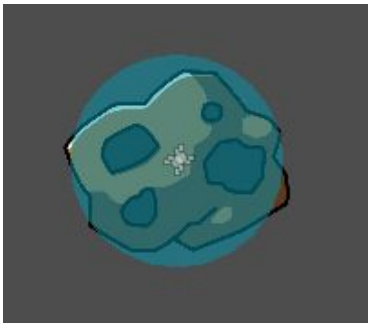
Nuestro juego necesita un enemigo que serán dos meteoritos uno grande y otro más pequeño, para crear los meteoritos necesitaremos una escena con los siguientes nodos:

AnimatedSprite: Lo utilizamos para añadir los dos tipos de meteoritos.

CollisionShape2D: Añadimos dos capas de colisión a los meteoritos.

VisibilityNotifier2D: Sirve para que cuando el meteorito salga de los límites del mapa se elimine.

- El resultado de la escena anterior será nuestros meteoritos con el área de colisión que le hemos añadido.



El tamaño del CollisionShape del meteorito grande hay que ajustarlo a mano en el script porque por defecto guarda las medidas del primer meteorito.

3.2.1 Script de escena Meteorito:

```
1  extends RigidBody2D
2
3  export (int) var velocidad_min
4  export (int) var velocidad_max
5  var tipo_roca = ["grande", "pequeño"]
```

-**Extends RigidBody2D** se refiere el tipo de nodo inicial de la escena.

-**Export (int) var Velocidad_min** es para añadir un nueva variable de velocidad mínima del meteorito dentro del inspector.

-**Export (int) var Velocidad_max** es lo mismo que el **min** pero velocidad máxima.

-**Var tipo_roca = [“grande”, “pequeño”]** se refiere a la variable del tipo de meteorito que tenemos, en nuestro caso solo tenemos dos, uno grande y otro pequeño.

```
8  func _ready():
9      $AnimatedSprite.animation = tipo_roca[randi () % tipo_roca.size()]
10
11     if $AnimatedSprite.animation == "grande":
12         $CollisionShape2D.scale.x = 1
13         $CollisionShape2D.scale.y = 1.8
14
```

$AnimatedSprite.animation = tipo_roca[randi () % tipo_roca.size()] esto significa que las animaciones que tenga animatiedSprite cambia a la variable tipo de roca y cogerá una aleatoriamente entre el tamaño.

if $AnimatedSprite.animation == "grande": este comando es el que he comentado antes que hace falta para cambiar el tamaño del meteorito grande que quiere decir lo siguiente:
Si el meteorito que sale en el juego es el grande tendrá la siguiente configuración.

- **$CollisionShape2D.scale.x = 1** cambiamos la escala y el tamaño de colisión horizontal es 1

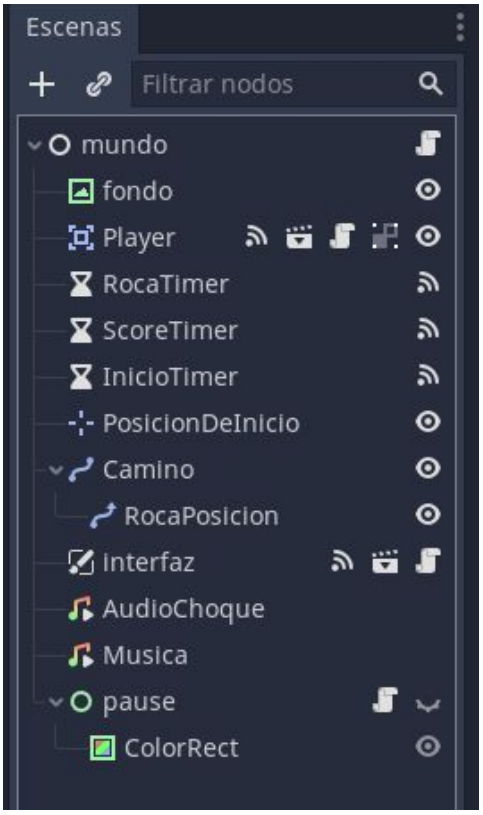
- **$CollisionShape2D.scale.y = 1.8** y el tamaño de colisión vertical es 1.8

Para hacer el siguiente paso tenemos que ir al apartado de nodos y añadir el nodo **Visibility**, seleccionar la opción de **screen_exited** y dentro del script se creará una línea.
La función que se ha creado tenemos que añadirle **queue_free()** que significa que cuando salga el nodo (Meteorito) de la pantalla se elimine.

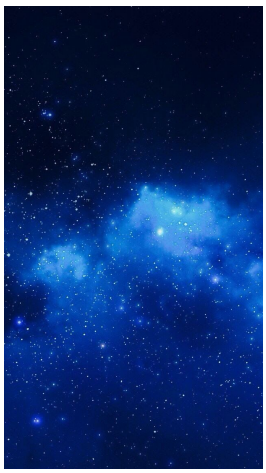

```
16 func _on_VisibilityNotifier2D_screen_exited():
17     queue_free() #eliminar roca
18
```

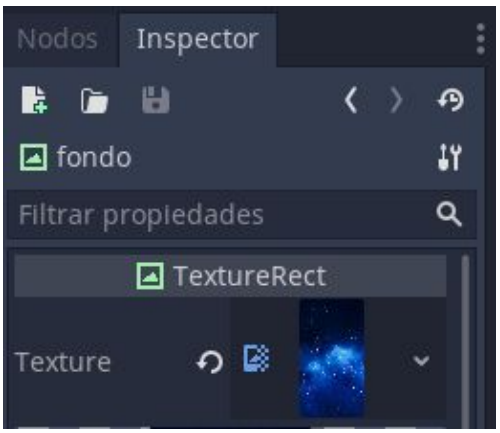
3.3 Escena Mundo

La siguiente escena la hemos llamado Mundo porque es la escena principal porque los siguientes nodos son imprescindibles para el funcionamiento del juego.

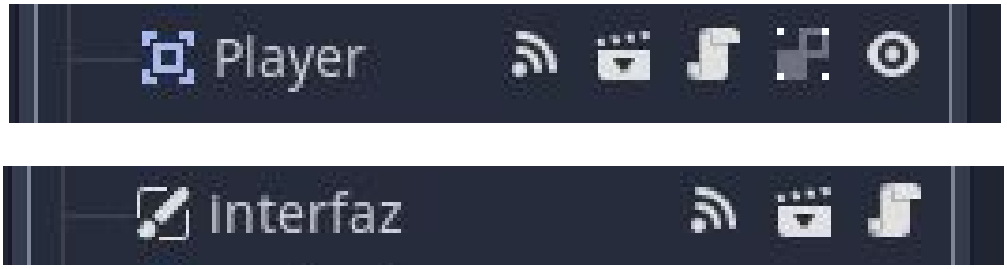


-El fondo que hemos elegido para nuestro juego es el siguiente, la manera de insertarlo es muy fácil, solo hay que ponerlo en el inspector





Como hemos dicho antes la escena mundo será la principal donde se iniciara el juego y para que aparezca el la nave y la interfaz tendremos que conectarlas a esta escena (Mundo)



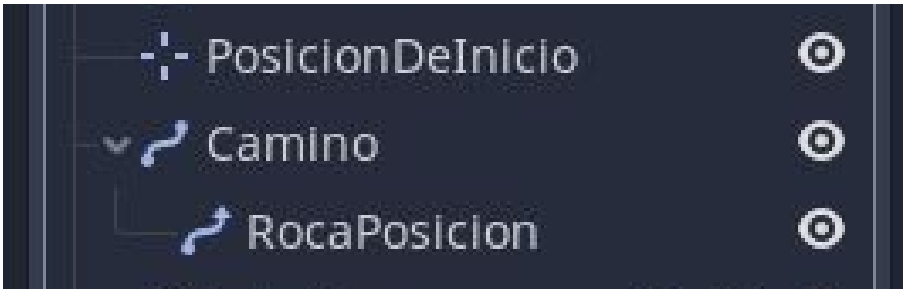
Los siguientes nodos son los que se encargan de controlar, son como temporizadores donde algo ocurre cada cierto tiempo.

Rocatimer → cada cuando se va crear una roca

Scoretimer → el controlador del tiempo, puedes configurar como tienes 1 punto cuando pasa 1 segundo, 2 puntos 2 segundos ….

Iniciotimer → cuando inicia el juego se inician los dos nodos anteriores.

El nodo posición de inicio se encarga de colocar la nave en la posición deseada, el nodo camino funciona seleccionado el tamaño de la pantalla del juego para que entren los meteoritos y el nodo restante indica como entraran.

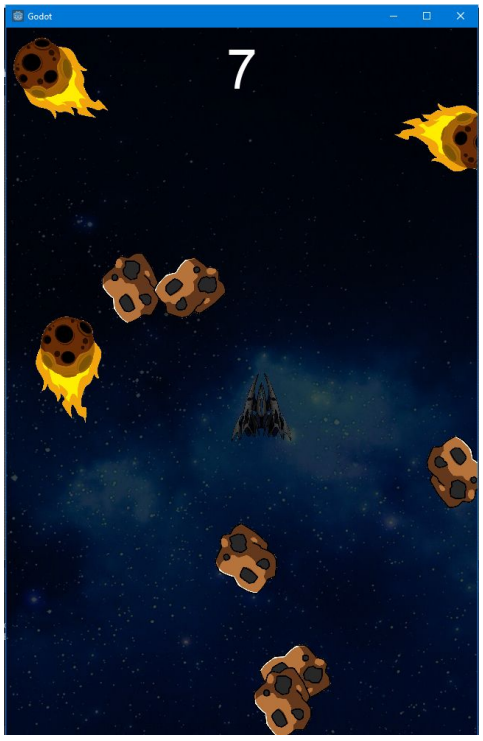


También hemos añadido música que se iniciara cuando pulsemos el boton start y cuando la nave se choque con un meteorito sonará una explosión

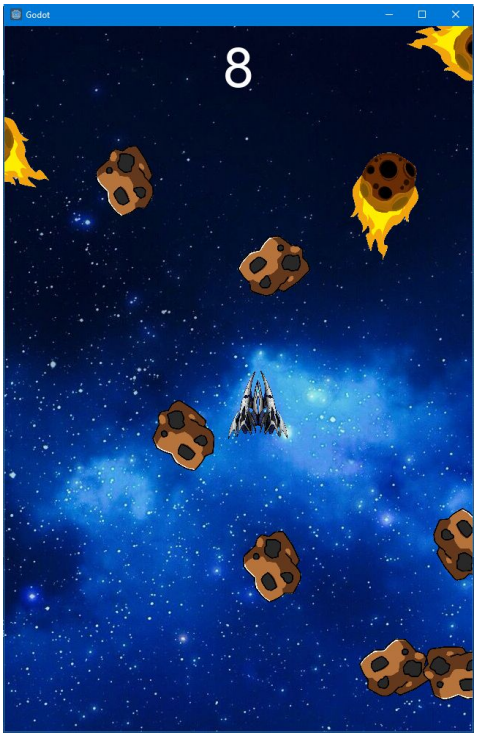
y por último hemos creado el estado de pausa para que cuando el jugador necesite hacer una pausa y no quiera cerrar el juego puede pulsar la tecla seleccionada que en este caso es el espacio y se pausara el juego, hemos añadido el nodo colorRect a pause para que cuando se pulse el botón de pausa la pantalla se ponga en un tono más oscuro, para salir del modo pausa solo hay que volver a pulsar el botón espacio.

Diferencias cuando se activa el modo pause:

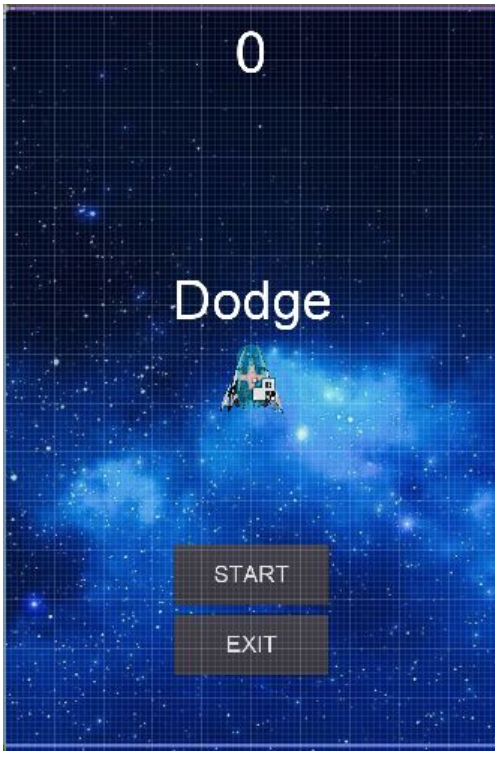
Modo pause:



Modo normal:

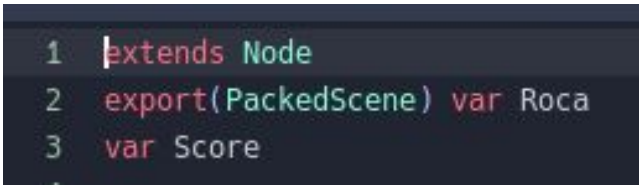


La escena Mundo completada quedaría de la siguiente manera:



3.3.1 Script de escena Mundo:

Para que todo lo creado anteriormente funcione necesitaremos añadir las correspondientes ordenes en el script.



Extends Node se refiere el tipo de nodo inicial de la escena.

Export (PackedScene) var Roca lo utilizaremos para crear una variable de roca dentro del inspector, y el **var Score** sirve para añadir una variable de puntos.

La función **func_ready** significa lo siguiente:

- Cada vez que inicie el juego, el meteorito saldrá por cualquier lugar.

La función **func nuevo_juego** significa lo siguiente:

- Tener el valor de puntos como número
- El player que es la nave se mostrará en la posición de inicio
- Se iniciara el tiempo de juego
- Mostrará el mensaje esquiva!!!
- Se actualiza el score
- La música del juego empiece a sonar

```
17
18 func game_over():
19     $ScoreTimer.stop()
20     $RocaTimer.stop()
21     $interfaz.game_over()
22     $AudioChoque.play()
23     $Musica.stop()
24
```

La función **func game_over** significa lo siguiente:

- Deja de contar los puntos
- Dejaran de salir meteoritos
- Se mostrará el mensaje game over en la pantalla
- Se escuchará el sonido que hemos añadido cuando se choque la nave
- La música del juego se parará.

```
25 func _on_InicioTimer_timeout():
26     $RocaTimer.start()
27     $ScoreTimer.start()
28
```

Cuando se inicie la función **InicioTimer_timeout** empezaran a salir los meteoritos y el score empezará a contar.

```
30 func _on_ScoreTimer_timeout():
31     Score += 1
32     $interfaz.update_score(Score)
33
```

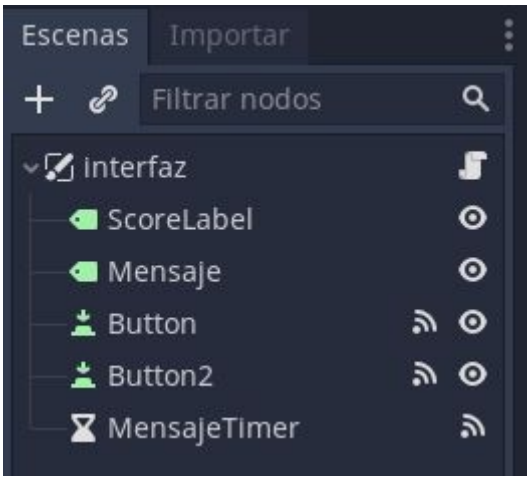
La función de **ScoreTimer_timeout** actualizará el score sumando 1 punto cada segundo mientras el juego esté funcionando.

```
36 func _on_RocaTimer_timeout():
37     #seleccionar un camino aleatorio
38     $Camino/RocaPosicion.set_offset(randi())
39
40     var R = Roca.instance()
41     add_child(R)
42
43     #Seleccionar una direccion
44     var d = $Camino/RocaPosicion.rotation + PI/2
45
46     R.position =$Camino/RocaPosicion.position
47
48     d += rand_range(-PI /4, PI /4)
49     R.rotation = d
50     R.set_linear_velocity(Vector2(rand_range(R.velocidad_min, R.velocidad_max), 0).rotated(d))
51
```

La siguiente función sirve para que los meteoritos entren de manera aleatoria al juego y a diferentes velocidades

3.4 Escena Interfaz

Nuestro juego necesita un marcador, unos botones para jugar o salir, para crear las interfaces necesitaremos una escena con los siguientes nodos:



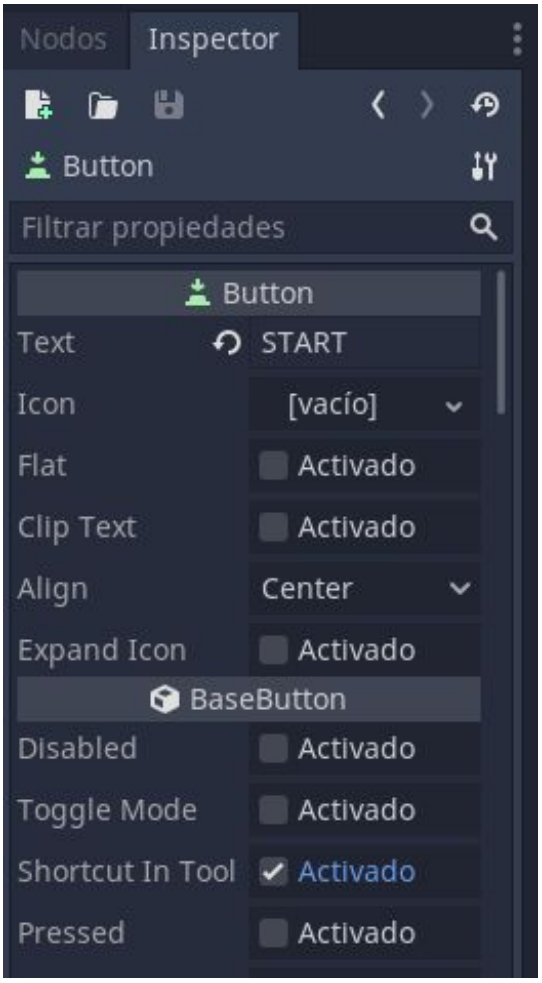
Score label mostrará la puntuación que vayamos consiguiendo, cuando iniciamos el juego el marcador empieza en 0 y va subiendo conforme más tiempo aguantemos.

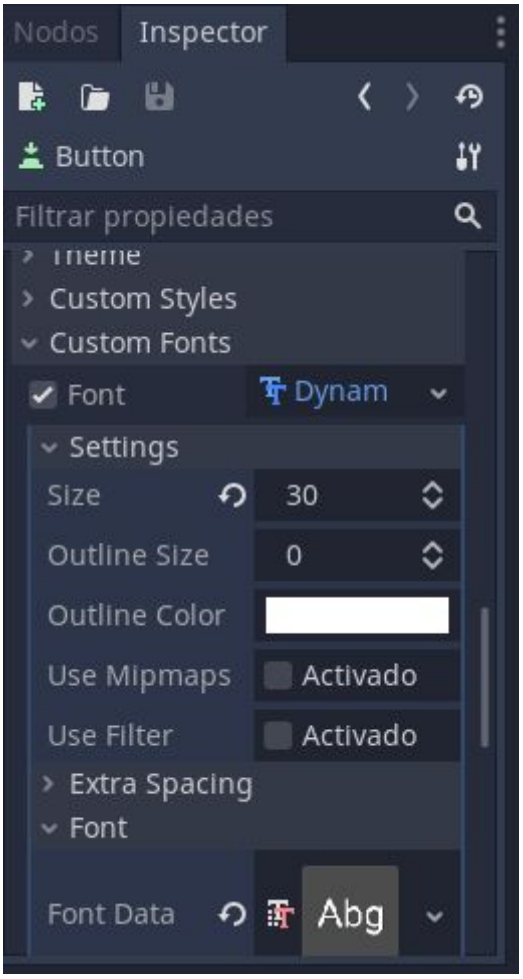
Sistema de puntuación: 1 segundo = 1 punto

El nodo mensaje es el nombre de nuestro juego que saldrá en grande cuando lo iniciemos

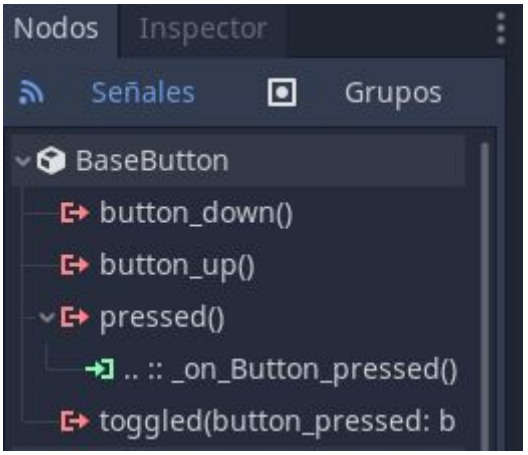
Para crear los botones solo hay que añadir el nodo button y en el inspector ajustarlo y poner la medida que veamos que quede bien para nuestro juego

Ejemplo botón start:





Cuando ya lo tengamos a nuestro gusto tenemos que crear una señal.



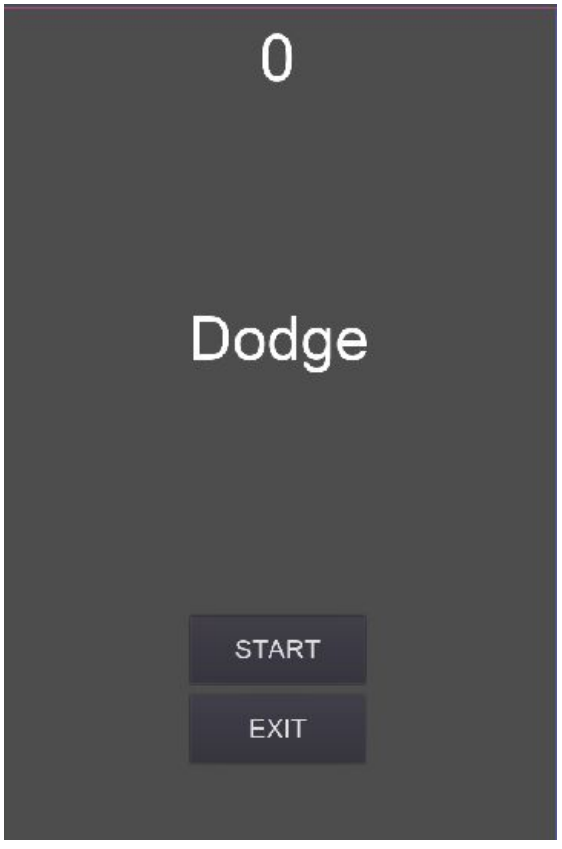
La señal que hemos creado es que cuando se pulse el botón start emita la señal “iniciar juego” y la función entera significa que cuando se pulse el botón, se escondan(start, exit) y se inicie el juego.

```
28 func _on_Button_pressed():
29     $Button.hide()
30     emit_signal("iniciar_juego")
31     $Button2.hide()
```

El botón Exit es como el start, se crea una señal que emite que el juego se cierre.

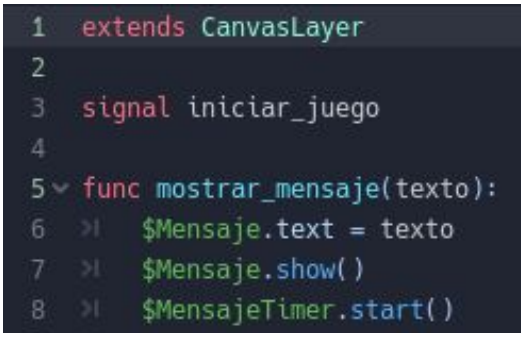
```
33 func _on_Button2_button_up():
34     get_tree().quit()
```

- El resultado de la escena interfaz será el siguiente:



3.4.1 Script de escena Interfaz:

Para que la interfaz funcione bien y se muestre todo en orden y cuando se deba mostrar tendremos que indicarlo en el scrip de la interfaz



Extends CanvasLayer Es un tipo de nodo inicial de la escena.

Signal iniciar_juego Es una señal que cuando se emite el juego funciona y se muestran los objetos.

La función **Func mostrar_mensaje(texto)** Es un texto que es como una pequeña variable.

$Mensaje.text = texto Muestra el texto que hemos puesto en la escena mundo.

$Mensaje.show() muestra el texto que tenga el mensaje.

$MensajeTimer.start() Controla el tiempo cuando muestre el mensaje.

En **func game_over** estan todas las líneas necesarias para que se muestre el mensaje cuando mueras y se vuelvan a mostrar los botones y el nombre para volver a jugar otra partida.

Mostrar_mensaje("GAME OVER") → mostrará la palabra game over.

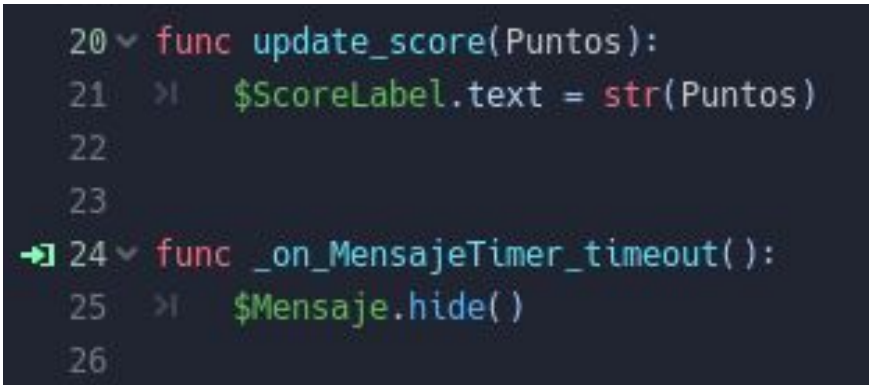
Yield($MensajeTimer, "timeout") → Esto sirve para que los mensajes no se muestren a la vez, en este caso se va mostrar el mensaje GAME OVER durante el tiempo y luego saldrán los botones 1 y 2 y el mensaje (nombre del juego).

$Button.show() → Que muestre el botón de JUGAR

$Button2.show() → Que muestre el botón de EXIT

$Mensaje.text = "Dodge" → Escribes el mensaje que quieres dentro de la comilla.

$Mensaje.show() → Que muestre el mensaje.



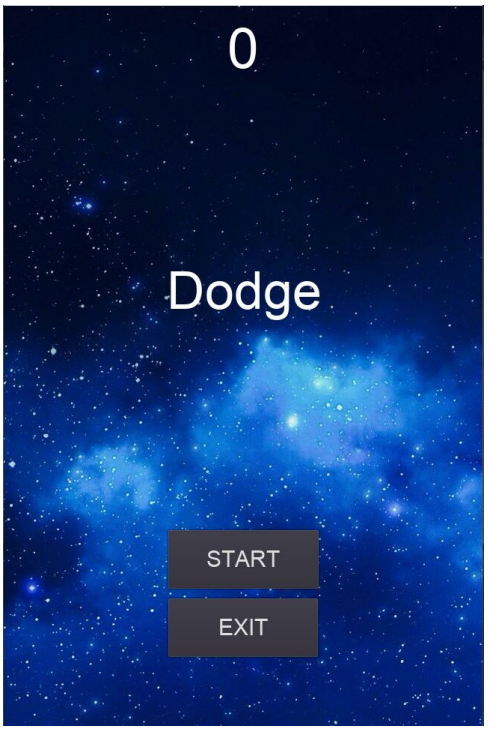
Func update_score (Puntos) es para actualizar los puntos

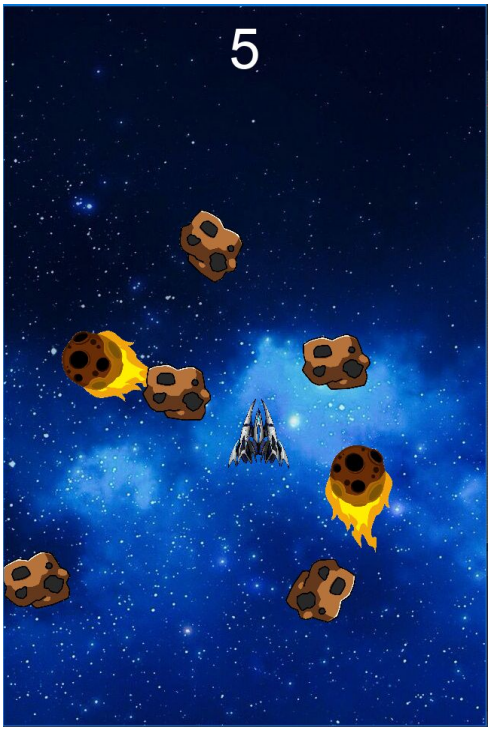
$ScoreLabel.text = str (Puntos) Aquí es donde te dice qué quieres actualizar, en este caso es el **ScoreLabel**, **str** es para convertir en un tipo de texto, añadimos **str** porque nuestro puntos esta en forma de numero.

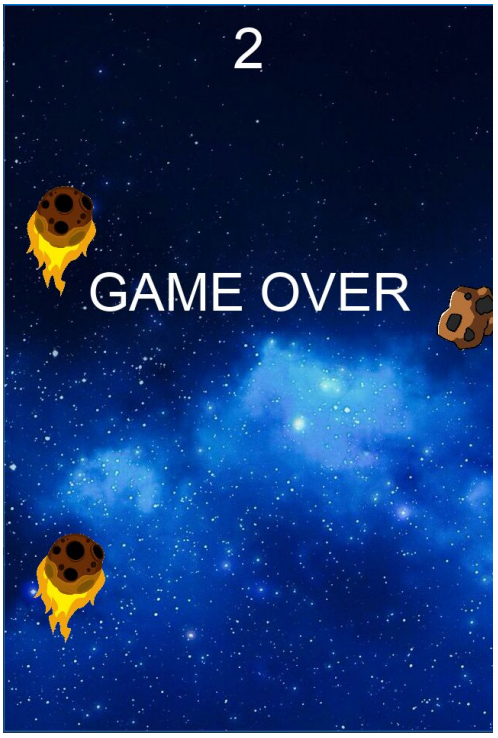
Func _on_MensajeTimer_timeout sirve para cuando se acabe el tiempo que se oculte el mensaje.

Func _on_Button_pressed() Como he explicado al principio cuando presionas el boton de jugar manda la señal iniciar juego y se ocultan los botones. y la función **func _on_Button2_button_up()** cuando se presione el botón exit, se apagará el juego.

4. Resultado final







En las imágenes anteriores podemos observar que una vez abierto el juego nos muestra el score en 0, el nombre del juego y los botones start y exit, cuando le damos a start el juego empieza y en la pantalla nos mostrará el mensaje que hemos añadido antes que es Esquiva!!! y empezará a sonar la música y aparecerán los meteoritos y empezara el juego.

El final del juego lo decidirá la habilidad o suerte que tenga el jugador porque el juego está creado de manera que la aparición de los meteoritos es aleatoria.

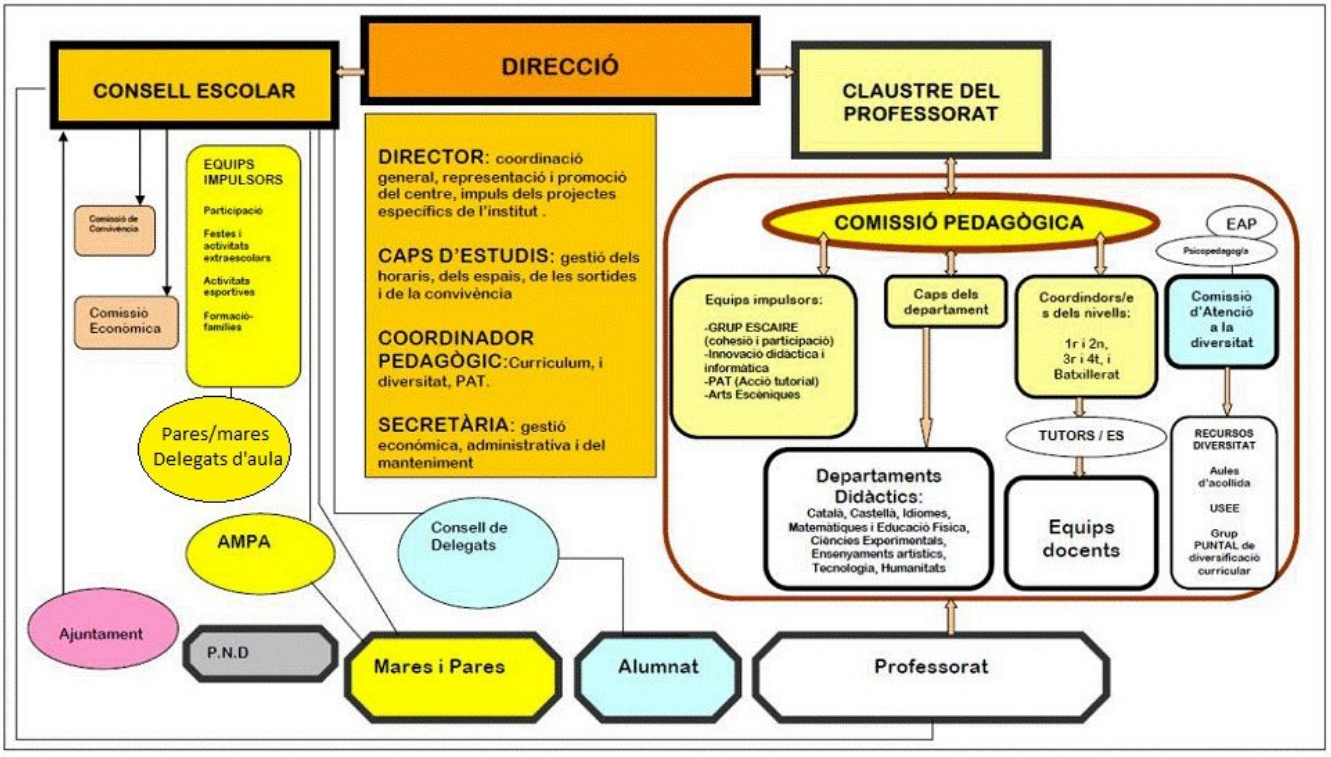
Cuando el jugador se choque con un meteorito sonará una explosión y a continuación se mostrará el mensaje de game over y los meteoritos dejaran de aparecer y volverá a la pantalla de inicio y el jugador decidirá si juega otra o se sale del juego.

5. Blocs Pràctiques

BLOC 1

Yo hago las prácticas en un instituto en vez de una empresa. El instituto que hago de prácticas se llama Ramón Berenguer IV que Tiene más de 60 años de experiencia.

Esto es la estructura que tiene el instituto Ramon Berenguer IV.



Como podemos ver esto esta formado en tres sectores importantes, son estos tres: Consell escolar, Claustre del professorat o comissió pedagògica y Direcció. A parte de estos sectores como sale en el esquema podemos observar que hay un cuadro naranja en el medio del imagen que forma parte de sector Direcció.

Cada sector hay varios subsectores, excepto el cuadro naranja que es el Director, Caps d'estudis, Coordinadora pedagógica y secretaria. Mi trabajo dentro del instituto está ubicado en secretaria, encargado de mantenimiento de informática y mantenimiento de los equipos.

El responsable de que yo cumpla mi función dentro del instituto sería el profesor de informática. Las funciones generales del encargado del profesor serían las siguientes:

- Mandar trabajo a nosotros y nos enseña cómo solucionar errores del sistema del PC, fallos de los dispositivos.

- Cuando acabemos los trabajos, teníamos un documentos para trabajar, porque hay problemas de equipos o de sistema de los otros profesores o alumno.

El objetivo que debo conseguir en el instituto es saber cómo solucionan los errores especiales o errores típicos y cómo montar un torre o PC.

BLOC 2

En este instituto me gustaría subir paso a paso. Ahora estoy trabajando en la secretaría de informática como un ayudante, pero con el tiempo y con los conocimientos aprendidos, me gustaría subir a un trabajador normal de informática, y si es posible el encargado de secretaría de informática.

Para conseguir esto, he pensado una serie de etapas que tienes que pasar. Estas etapas están divididas en dos y conseguir hacer cada una de estos de la mejor manera será muy importante para el futuro. Las etapas son las siguientes:

- Etapa camino

- Etapa final

En cada etapa hay puntos importantes que lo tienes que conseguir y así puedes subir el objetivo de llegar a ser Gerente General o más.

Etapa de la camino: La primera etapa, la de la camino, será el inicio del camino. En esta etapa tendrás la oportunidad de tener muchas experiencias y relaciones con muchas personas y empresas.

En esta parte tienes que pensar tus futuros: ¿Qué quiero hacer? ¿Dónde te gustaría trabajar? Ponerse objetivos, subir tu formación y es importante saber que, esta parte no tienes límites, podrás buscar cualquier trabajo de informatica.

La etapa del camino puedes saber tu carrera de lo que quieres hacer y estudiar lo posible de las cosas para subir al límite.

Y no deberías buscar trabajos de mucho dinero, sino tienes que aprender más y tener mucha experiencia.

Etapa de final: La segundo etapa y la última que es la de final, sería el medio del camino hacia final. En esta etapa ya tienes muchos conocimientos y experiencias conseguidas.

Una vez tenido conocimientos y experiencias suficientes, pues tu objetivo de esta parte es buscar a un empresa, del que te gusta y trabajar a tope para subir al puesto que querías llegar usando tus conocimientos aprendidos, experiencias tenidas y personas conocidos en la etapa anterior.

Etapa camino   Etapa final

|->->->->->->->->->->->->->->->->->|->->->->->->->->->->->->->->->->->|

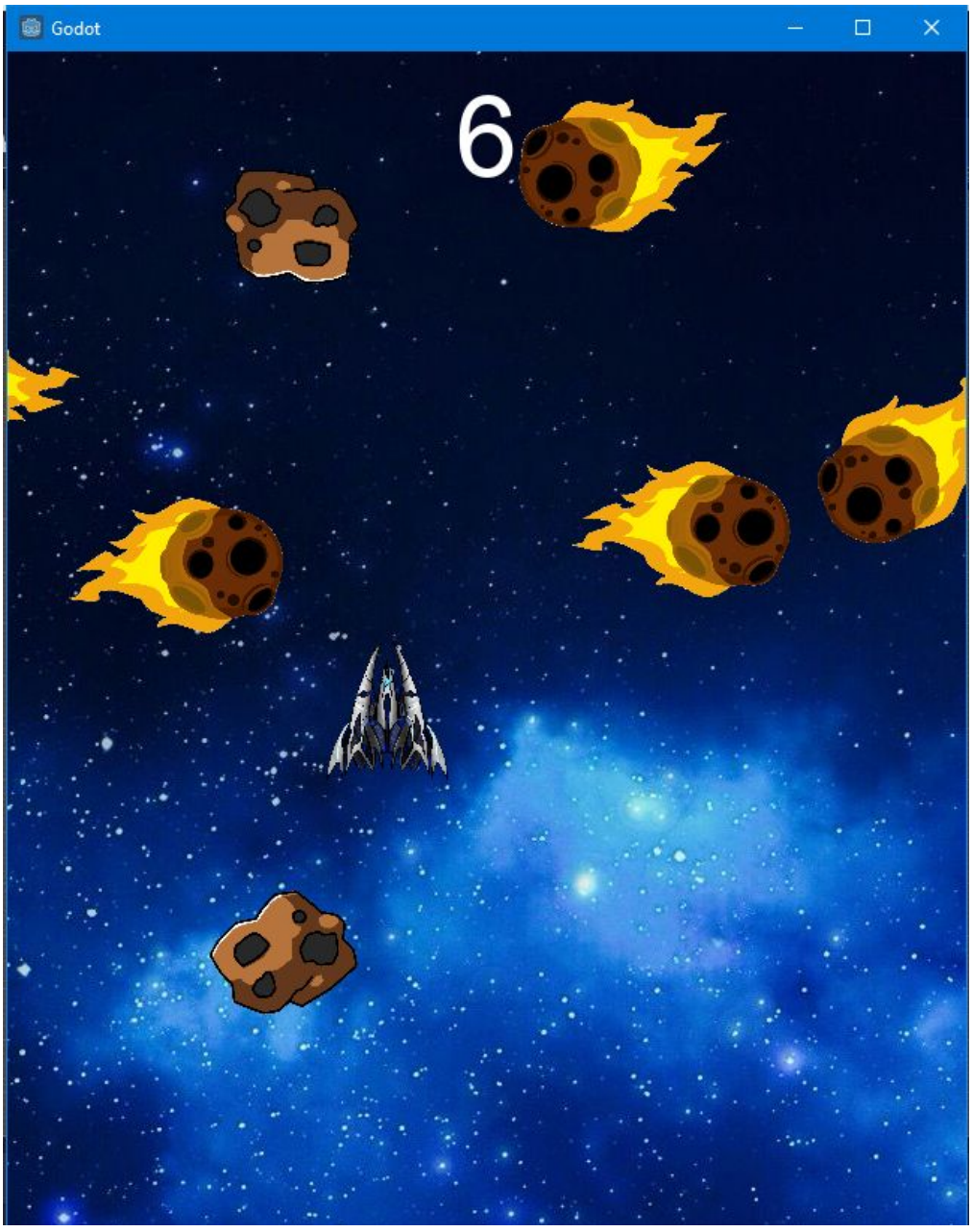
BLOC 3

El proyecto de síntesi está ubicado en el instituto Ramón Berenguer IV, es donde estoy haciendo practicas.

Para conseguir que el juego funcione tienes que crear las siguientes características dentro del Godot, que es donde lo hemos creado el juego:

- Escena del jugador y el script
- Escena del Enemigo y el script
- Escena del Mundo y el script
- Escena del Interfaz y el script

Captura de que el juego está funcionando.



Tener este juego en el instituto nos va servir mucho. Porque este juego ayuda a los estudiante que tenga más ganas de aprender y es un juego fácil y entretenido. Este juego puede ser corto o largo dependiendo del jugador del tiempo que aguante.