

Arkanoid 2D: Acercamiento

Autor: Ignacio Humire

Herramientas:

Para realizar la tarea se ocupó godot como principal y única herramienta para trabajar en el modelamiento y computación gráfica necesaria para realizar el proyecto planteado.

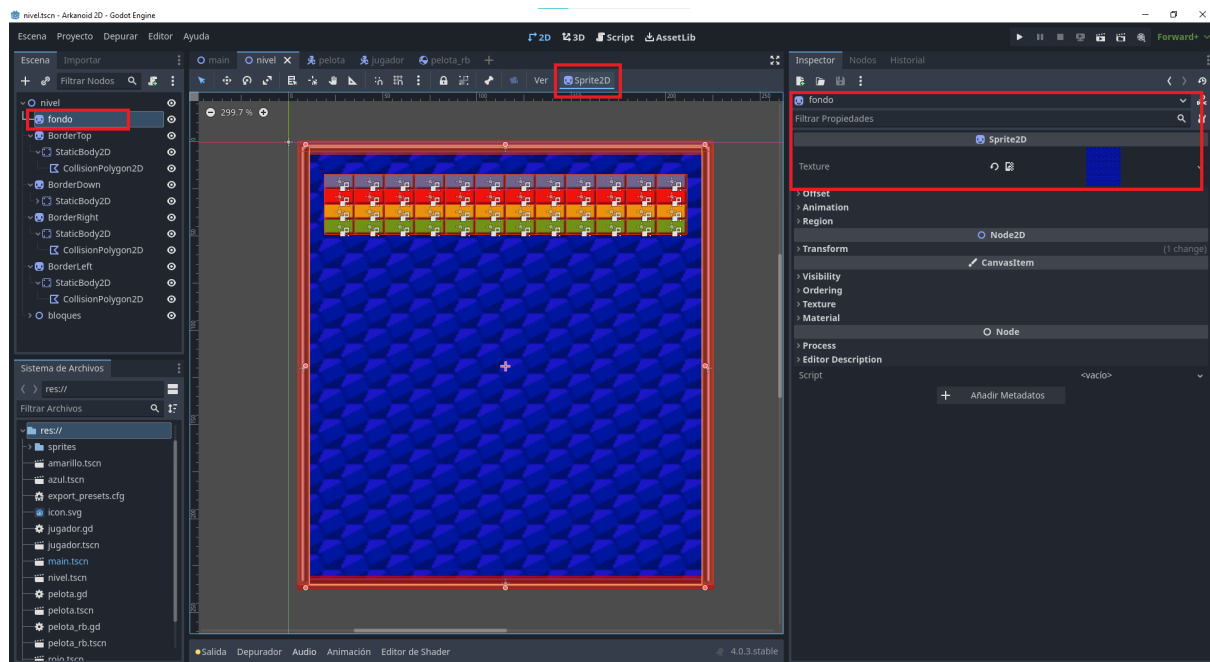
Objetivos:

A continuación se presentarán todos los objetivos propuestos, incluyendo los de un comienzo y los que fueron surgiendo a medida que fluía el trabajo:

Creación de figuras simples + texturas:

Para el proceso de creación de figuras existe la posibilidad de utilizar imágenes descargadas de Internet. Esta funcionalidad está disponible en la misma herramienta y se puede emplear mediante el uso del nodo sprite 2D.

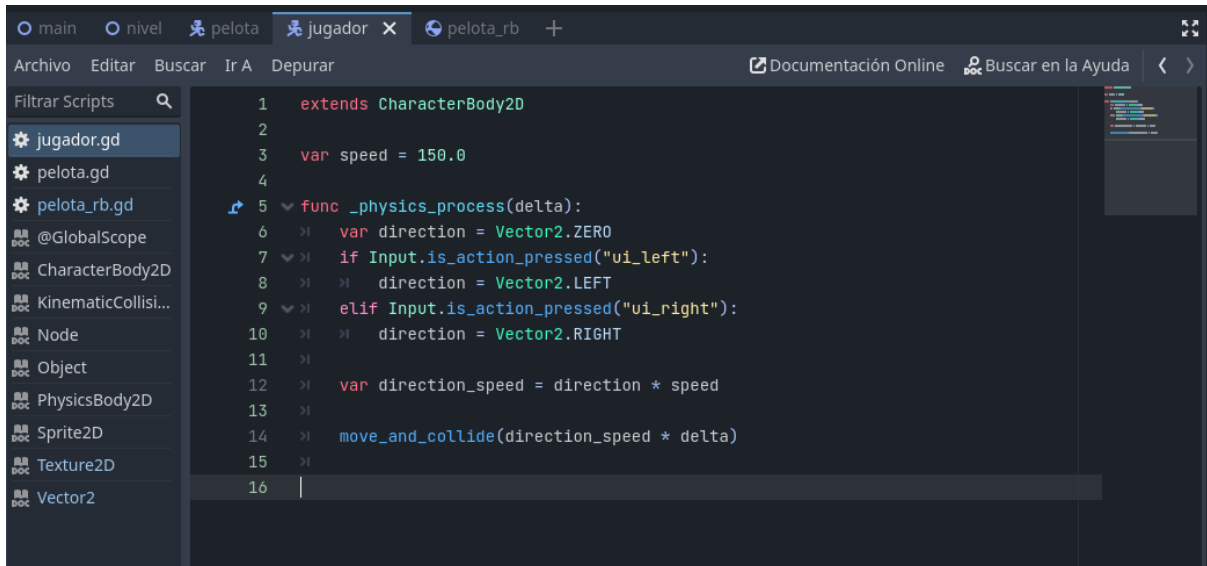
A continuación un pantallazo con el nivel:



Controladores/inputs:

Tenemos un nodo llamado “player” que funciona a través de inputs, el cual funciona gracias a un script .gd que le da tal cualidad.

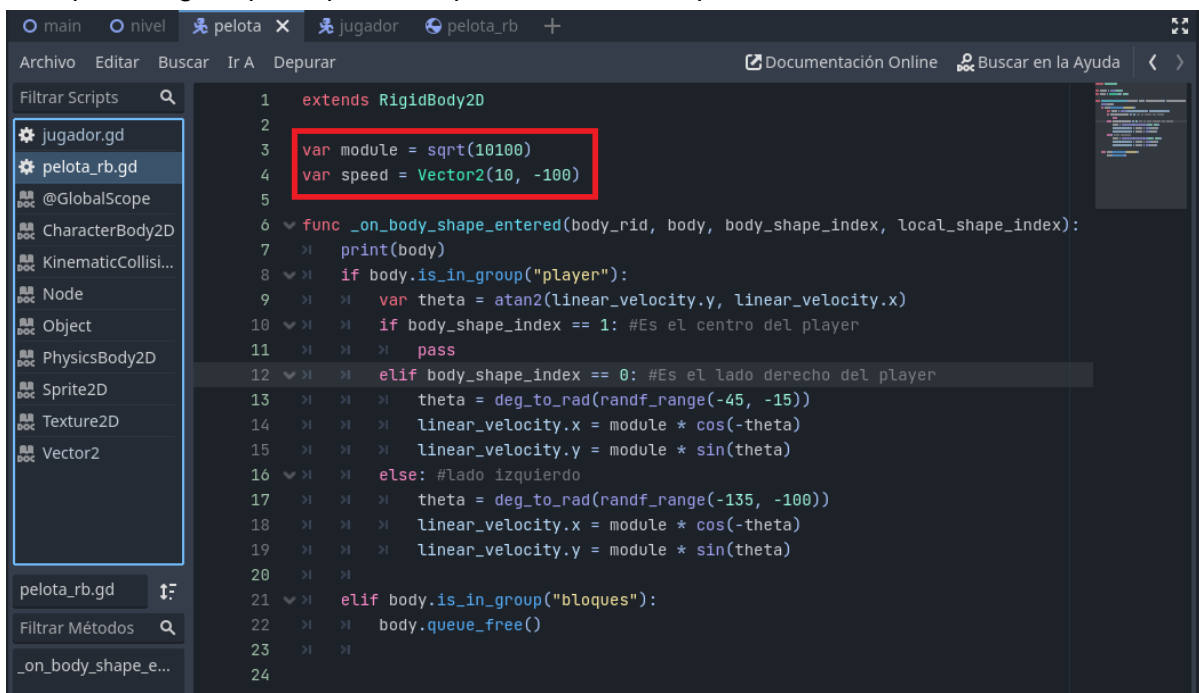
A continuación un pantallazo del script .gd del nodo:



```
1 extends CharacterBody2D
2
3 var speed = 150.0
4
5 func _physics_process(delta):
6     var direction = Vector2.ZERO
7     if Input.is_action_pressed("ui_left"):
8         direction = Vector2.LEFT
9     elif Input.is_action_pressed("ui_right"):
10        direction = Vector2.RIGHT
11
12    var direction_speed = direction * speed
13
14    move_and_collide(direction_speed * delta)
15
16
```

Movimiento y colisión de las figuras:

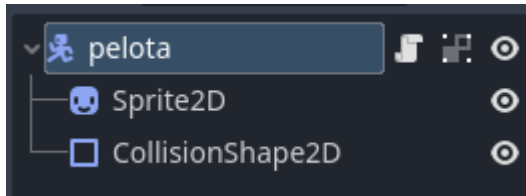
Para poder lograr que la pelota se pueda mover esta posee una velocidad:



```
1 extends RigidBody2D
2
3 var module = sqrt(10100)
4 var speed = Vector2(10, -100)
5
6 func _on_body_shape_entered(body_rid, body, body_shape_index, local_shape_index):
7     print(body)
8     if body.is_in_group("player"):
9         var theta = atan2(linear_velocity.y, linear_velocity.x)
10        if body_shape_index == 1: #Es el centro del player
11            pass
12        elif body_shape_index == 0: #Es el lado derecho del player
13            theta = deg_to_rad(randf_range(-45, -15))
14            linear_velocity.x = module * cos(-theta)
15            linear_velocity.y = module * sin(theta)
16        else: #lado izquierdo
17            theta = deg_to_rad(randf_range(-135, -100))
18            linear_velocity.x = module * cos(-theta)
19            linear_velocity.y = module * sin(theta)
20
21    elif body.is_in_group("bloques"):
22        body.queue_free()
23
24
```

Posteriormente, para permitir la interacción y colisión entre los nodos, se asigna a los sprites 2D de las figuras un nodo hermano que representa una figura geométrica de colisión. Esta figura de colisión es utilizada para indicar al programa cómo debe comportarse cuando dos objetos sólidos interactúan entre sí.

A continuación un ejemplo con la pelota:

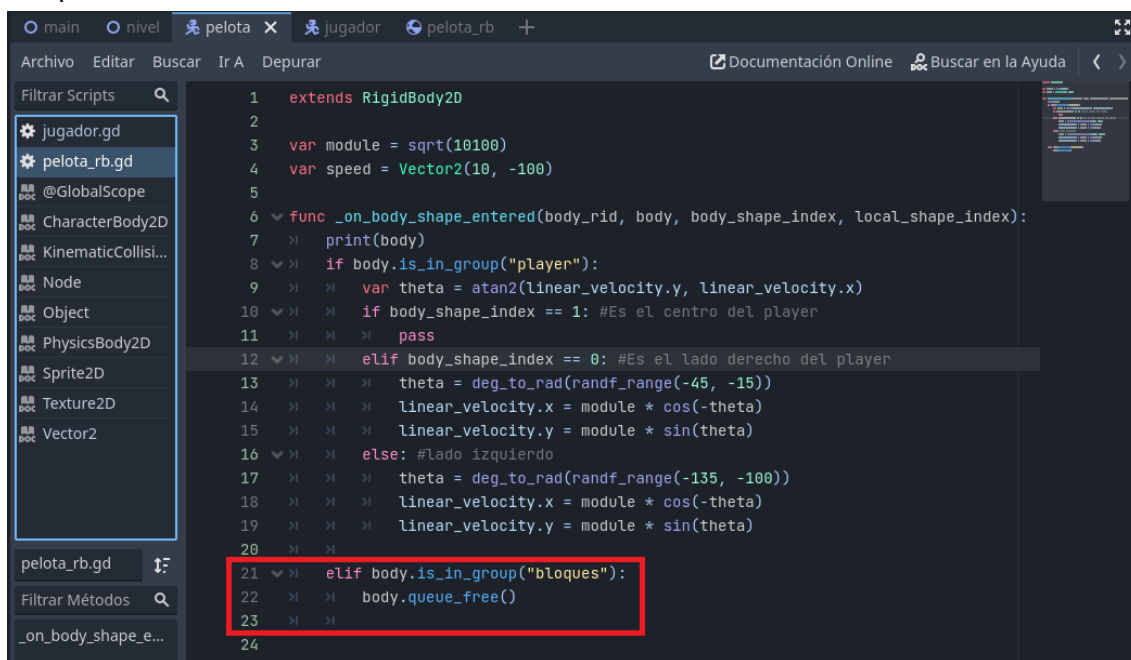


Sonido de fondo:

Este objetivo no fue abordado durante el desarrollo del proyecto, ya que a medida que avanzaba en el trabajo me di cuenta de que no estaba alineado con los objetivos principales de la tarea ni del curso.

Jugabilidad:

Aunque la jugabilidad no era un objetivo central del proyecto, consideré que sería interesante implementar la funcionalidad de romper/desaparecer los bloques al colisionar con la pelota.



Para lograr la funcionalidad de romper los bloques al interactuar con ellos, se implementó la creación de un grupo que contiene todos los nodos de los bloques. Al detectar una colisión entre la pelota y algún bloque perteneciente a ese grupo, se activa el código correspondiente para romper/desaparecer el bloque en cuestión.

Obs: En el programa implementado no se ha incluido la funcionalidad de perder cuando la pelota colisiona con el límite inferior del nivel. En otras palabras, la pelota puede rebotar indefinidamente sin que exista una condición de derrota.