Universidad Autónoma de Baja California

Docente: Eloísa del Carmen García Canseco

Física para programadores de videojuegos

MonkeBarrel

Reyes Alvarez Ernesto
Gómez Carrillo Nataly

27/05/21

Objetivo.

El propósito de este proyecto es crear un videojuego utilizando las leyes de la física. Esto implica que no se deben utilizar el motor de físicas del *engine* de videojuegos en el que se esté desarrollando el videojuego.

Introducción.

El tiro parabólico es uno de los movimientos que más se utilizan dentro de los videojuegos, es por esto que decidimos utilizar este movimiento en nuestro videojuego. El juego que creamos es una recreación de un nivel bonus del videojuego *Donkey Kong Country*, juego creado por el estudio britanico *RARE* y publicado por *Nintendo* el año de 1994, como dato extra, este videojuego solo es creado por *RARE*, ya que la licencia le pertenece a *Nintendo*. Este videojuego fue creado en el engine Godot.

Figura. 1. Ejemplo del nivel bonus.

Descripción del videojuego.

El juego en sí es bastante sencillo, consta de un barril el cual está girando sobre su propio eje justo en la parte inferior central del escenario, el escenario es un pasillo vertical amplio, dentro de él hay 3 hileras de plátanos los cuales debes recoger para ganar, la manera de obtener todos los plátanos en disparando a *Donkey Kong* con el barril, el disparo se ejecuta si presionas el botón izquierdo de tu mouse y una

vez nuestro personaje haya sido disparado se puede mover de izquierda a derecha presionando los botones “A” y “B” respectivamente, para hacer un segundo disparo *Donkey Kong* debe regresar al barril. Si obtienes todos los plátanos, tendrás la victoria, pero si caes al vacío perderás y tendrás que volver a jugar.

Figura 2. Escenario completo.

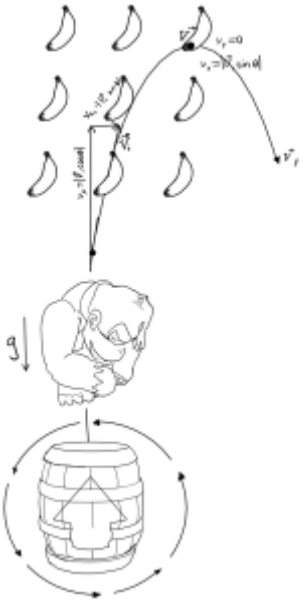

Figura 3. Esquema de cómo funciona el juego.

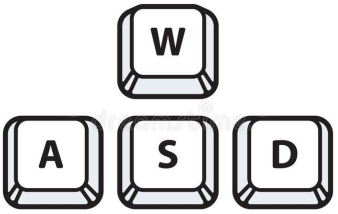

Figura 4. Moverse de izquierda a derecha

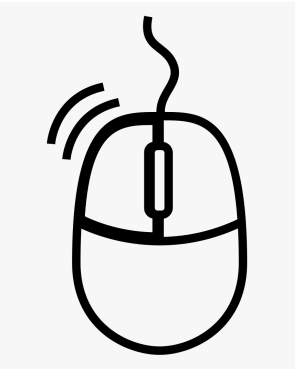

Figura 5. Botón de disparo

De todas las variables que se utilizan para hacerlo funcionar las únicas que se modifican constantemente es la dirección en la cual *Donkey Kong* será lanzado, y la variable que cuenta los plátanos que aún se

encuentran activos, esta variable tiene la función de accionar la pantalla de victoria cuando haz recolectados todos los plátanos. Ninguna de estas variables se muestran en pantalla, lo único en pantalla que se muestra son los plátanos que aún se encuentran activos, *Donkey Kong*, el barril, las paredes y el fondo.

El juego cuenta con un menú inicial el cual cuenta con dos botones, “Jugar” y “Salir”, “Jugar” te dirige al juego, mientras que “Salir” cierra el juego. Igualmente, existen dos pantallas más, la de victoria y la de derrota, como se mencionó con anterioridad, la escena de victoria aparece cuando colectas todos los plátanos, y la escena de derrota aparece cuando caes al vacío; ambas pantallas tienen un botón que te redirige al menú inicial.

Figura 6. Menú inicial

Figura 7. Pantalla de victoria

Figura 8. Ejemplo de cómo pierdes en el juego

Figura 9. Pantalla de derrota

Desarrollo del videojuego.

Empecemos con el barril, este objeto se inicializa creando un *KinematicBody2D*, este hará la función de nodo padre, posteriormente se le agregarán los nodos hijos, estos serán un *Sprite*, el cual es la imagen del barril, un *CollisionShape2D*, el cual se encarga de detectar colisiones y finalmente un *Position2D*, este es el más importante ya que de aquí saldrá nuestro proyectil.

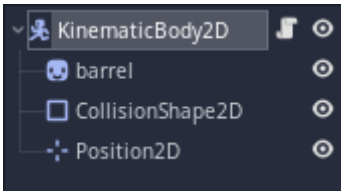

Figura 10. Estructura del barril.

Nuestro siguiente objeto es nuestro proyectil, *Donkey Kong*, su estructura es muy parecida a la del barril, pero en esta ocasión vamos a agregar a dos nodos hijos más, una *Camera2D*, la cual se utiliza para que la cámara siga al proyectil en todo momento, y un *Area2D* con un el cual tiene un *Collision2D* como nodo hijo, esto tiene la función de desaparecer al *Donkey Kong* cuando toca al barril, para dar la impresión de que regresó al barril.

Figura 11. Estructura del proyectil

Los plátanos son otro objeto importante dentro de nuestro juego, ya que si los colectas todos ganas. Su estructura es la siguiente, su nodo padre es un *Area2D*, esto es debido a que no queremos que cuando nuestro proyectil colisione con los plátanos, estos últimos detengan al *Donkey Kong*, lo cual sucedería si fuera un *KinematicBody2D* con colisiones. Como nodos hijos tenemos un *CollisionShape2D*, el cual sirve para detectar cuando nuestro proyectil entre en contacto con los plátanos, y finalmente, un *AnimatedSprite* que se encarga de animar al plátano. Para animar los plátanos en el *Inspector* se agrega un *Nuevo SpriteFrames* de das click y te abrirá una pantalla en la parte inferior, ahí debes agregar cada imagen del plátano y posteriormente se animará mediante código.

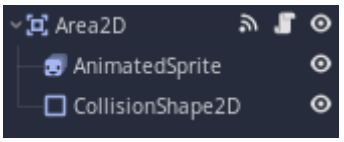

Figura 12. Estructura del plátano

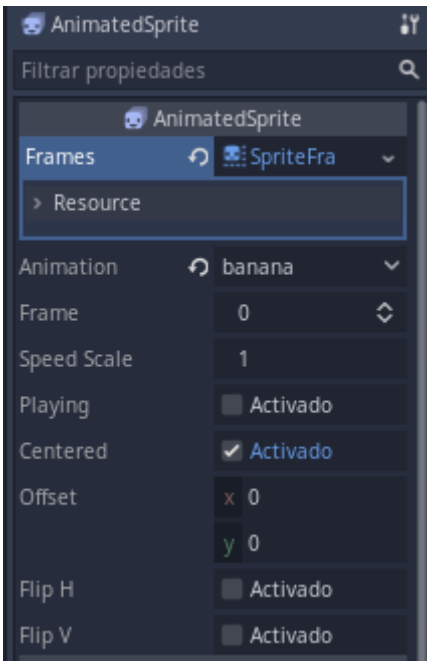

Figura 13. Configuración de la animación.

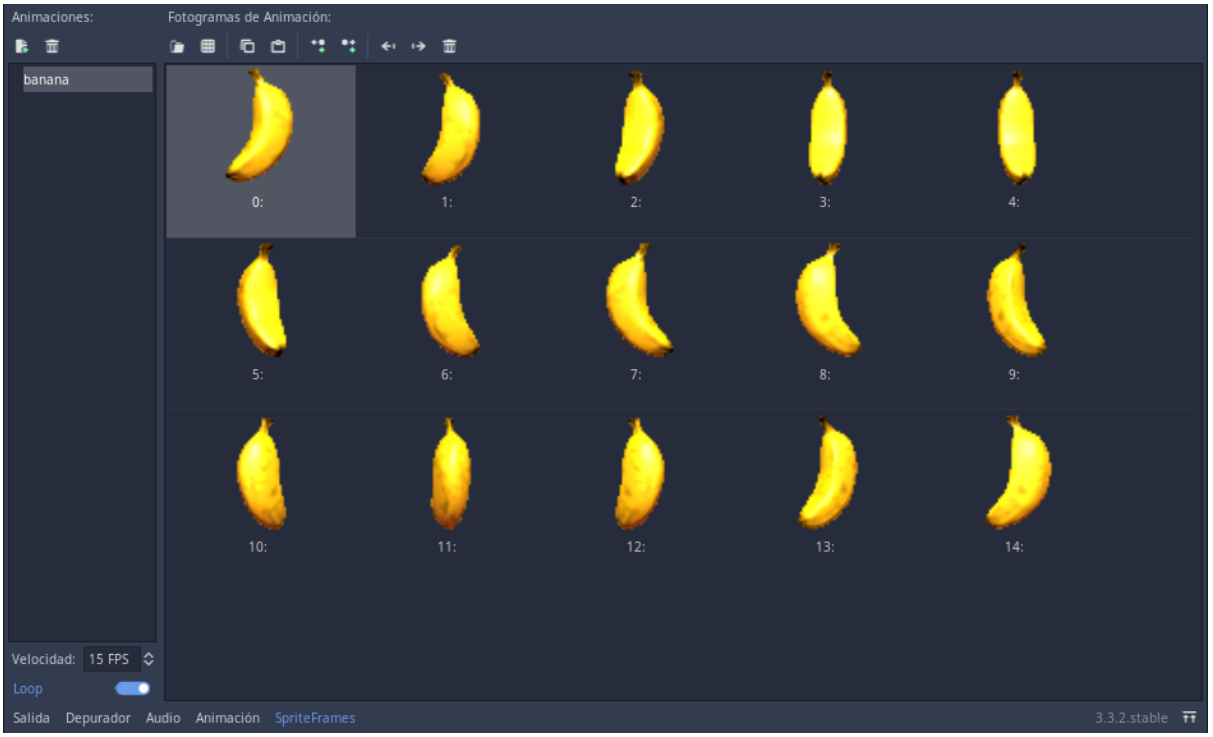

Figura 14. Frames de animación.

Nuestro escenario, también nombrado como *world*, cuenta con un *Node* como nodo padre, y sus nodos hijos son, la escena del barril, un *Node*, el cual contiene a todos los plátanos que se encuentran en pantalla, cada uno de los plátanos es la escena del plátano. Enseguida, tenemos dos *CanvasLayers*, cada uno cuenta con un

ParallaxBackground como nodo hijo, que a su vez tiene un nodo hijo *ParallaxLayer* la cual contiene uno de los fondos, esto dos últimos nodos tienen la función de que el fondo se mueva junto con el personaje para que el juego sea más inmersivo. Posteriormente tenemos tres *StaticBody2D* con colisiones para representar nuestras paredes y un techo para que nuestro proyectil no salga de nuestro escenario, luego, tenemos un *Area2D* con una colisión en la parte inferior del escenario para detectar cuando el *Donkey Kong* caiga al vacío, esto significa que haz perdido. Y finalmente, tenemos un *AudioStreamPlayer* para colocar la música ambiental del videojuego. El archivo de audio se llama *donkey-kong-country-bonus-room-blitz-restored.ogg* y se coloca en el *Inspector*.

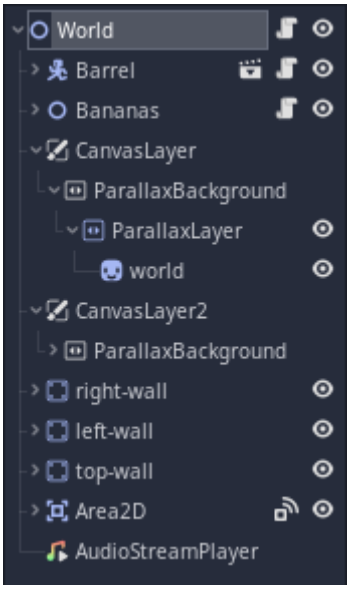

Figura 15. Estructura del world

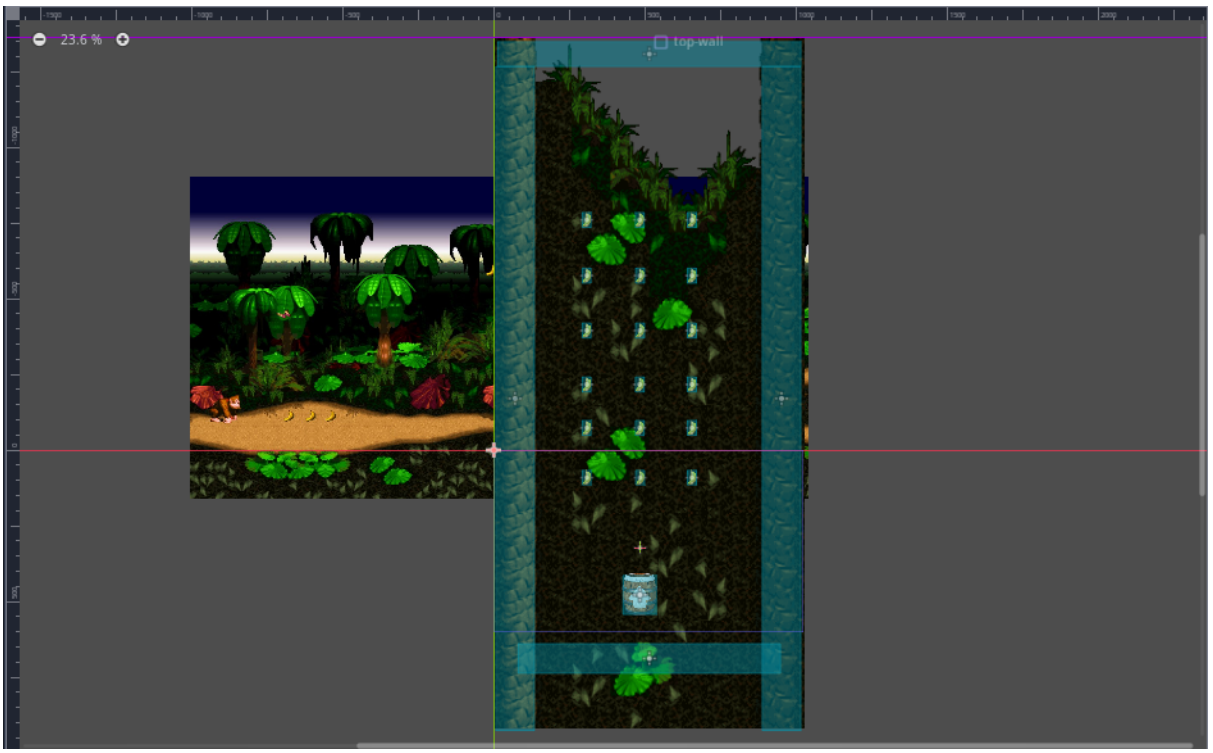

Figura 16. Posición de los nodos en el juego

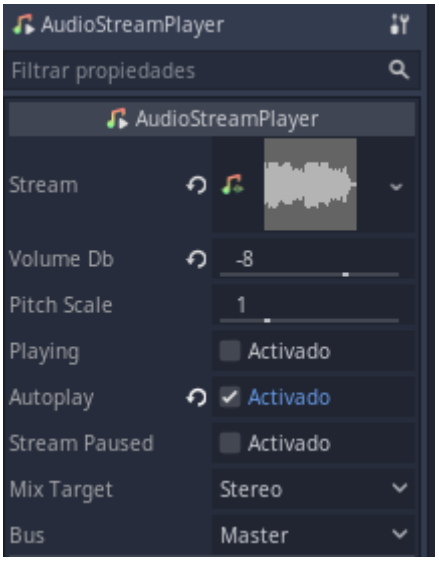

Figura 17. Configuración de la música en el world

En nuestro menú inicial el nodo padre es un *Node*, como hijo tenemos un *VBoxContainer* el cual tiene dos *TextureButton* los cuales contienen el sprite del botón que se utiliza también en las pantallas de victoria y derrota. Posteriormente hay un *ParallaxBackground* para colocar el fondo, finalmente, tenemos un nodo *RichTextLabel* en donde se escribe el texto que se muestra en pantalla. Para poder

escribir con la fuente de texto que se muestra en el menú debe descargar dicha fuente de internet dicha fuente se llama *Fipps-Regular*, una vez descargada te vas al *Inspector* que se encuentra del lado derecho de la pantalla, recuerda que debes tener seleccionado el nodo *RichTextLabel*, seleccionas la pestaña de *Custom Fonts*, activas *Normal Font*, agregas un *Nuevo Dynamic Font*, en los *Settings* cambias el *Size* a 40 y en la pestaña de *Font* agregas la fuente que ya mencioné con anterioridad.

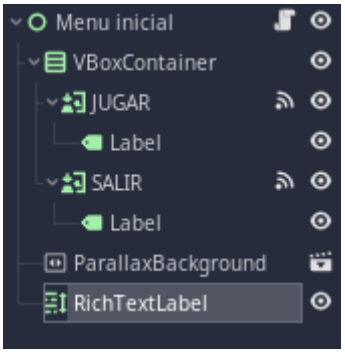

Figura 18. Estructura del menú inicial

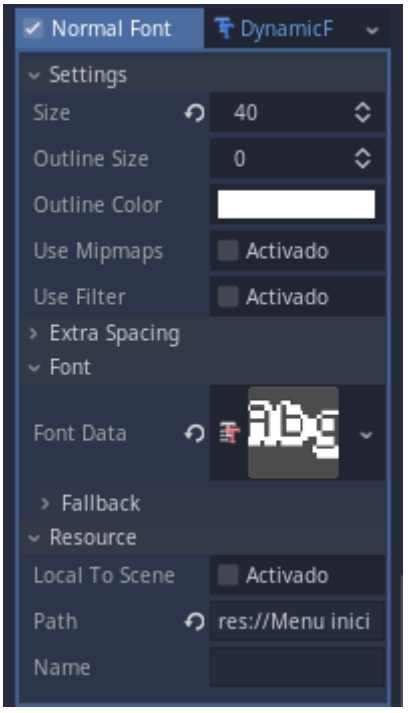

Figura 19. Configuración del texto.

Las pantallas de victoria y derrota son prácticamente iguales y contienen los mismos nodos que el menú inicial.

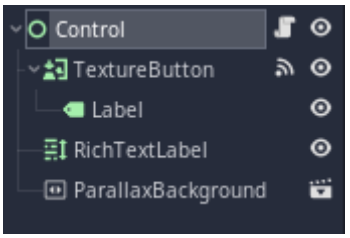

Figura 20. Estructura de la pantalla de victoria

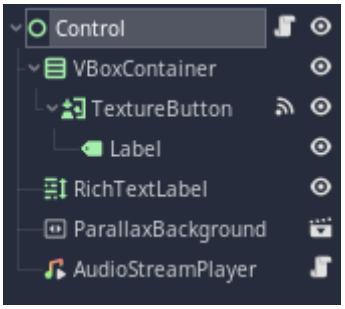

Figura 21. Estructura de la pantalla de derrota

Como se ve, la pantalla de derrota tiene un *AudioStreamPlayer* por lo que hay un audio corriendo cuando pierdes. A diferencia de la música de ambiente del world que se configuró en el *Inspector* en esta ocasión se hizo mediante código. Realmente no hay diferencia alguna entre uno y otro, es simplemente otra opción de cómo hacerlo.

```
1  extends AudioStreamPlayer
2
3  #Audio de la escena Game_Over
4  func _ready() -> void:
5      #Variable que contiene el archivo de audio
6      var audio: AudioStreamOGGVorbis = preload("res://Sound/Lost Life.ogg")
7      self.set_stream(audio) #Se aplica el audio
8      self.set_volume_db(3.0) #Volumnes del audio
9      self.play() #Con esto se escucha la musica
10
```

Figura 22. Código de audio en la escena de derrota.

Para que los botones de nuestras pantallas de victoria y derrota funcionen se utilizó el siguiente código.

```
extends Control

#Cuando presiones el botón te llevara al menú inicial
func _on_TextureButton_pressed() -> void:
	get_tree().change_scene("res://Menu inicial.tscn")

```

Figura 23. Código utilizado para la pantalla de victoria y derrota.

El menú inicial tiene un código muy parecido al anterior, debido a que igual solo hay interacción con los botones.

```
extends Control

#Al momento de precionar este botón te llevará al juego
func _on_JUGAR_pressed() -> void:
	$VBoxContainer/JUGAR.get_tree().change_scene("res://world.tscn")

#Al momento de precionar este botón cerrará el juego
func _on_SALIR_pressed() -> void:
	get_tree().quit()
```

Figura 24. Código del menú inicial.

Como se mencionó anteriormente la animación de los plátanos se hace mediante código, pero además se le agrega un emisor de señales el cual nos servirá para contar la cantidad de plátanos que han sido tomados.

```gdscript
extends Area2D

#Se obtiene el nodo donde se realiza la animación
onready var _animated_sprite: AnimatedSprite = get_node("AnimatedSprite")
signal banana_colected

func _ready() -> void:
	_animated_sprite.play("banana") #Corre la animación de la banana
	connect("body_entered", self, "_on_body_entered") #Conecta la banana con la funcion

#En esta funcion se realiza la desaparición del platano a la hora de que
#el Donkey Kong colisiona con ellas
func _on_body_entered(body: Node) -> void:
	if body.is_in_group("monke"):
		#Esta es una señal que se emite al recolectar los platanos
		#su funcion radica en saber cuantos de estos han sido recolectados
		emit_signal("banana_colected")
		queue_free() #desaparece el nodo del platano
		get_node("CollisionShape2D").disabled = true #Desactiva las colisiones
```

Figura 25. Código del plátano.

El código que cuenta los plátanos se encuentra en el *Node* que contiene a todos los plátanos colocados en nuestro world.

```gdscript
extends Node2D

var banana_count: = 18

#Aquí se cuenta la cantidad de platanos, una vez recolectados se cambia la escena
#a la escena de victoria
func _on_banana_banana_colected() -> void:
	banana_count -= 1
	print(banana_count)
	if banana_count == 0:
		get_tree().change_scene("res://WIIN.tscn") #escena de victoria

```

Figura 26. Código que cuenta plátanos.

El código que hace girar al *Donkey Kong* se encuentra directamente en su *Sprite*.

Figura 27. Código que hace girar al Donkey Kong.

Para saber en cuando haz perdido se creó un código el cual pertenece al *Node* padre de nuestro world, el cual detecta cuando el *Donkey Kong* atraviesa el *Area2D* que se encuentra en la parte inferior de la pantalla.

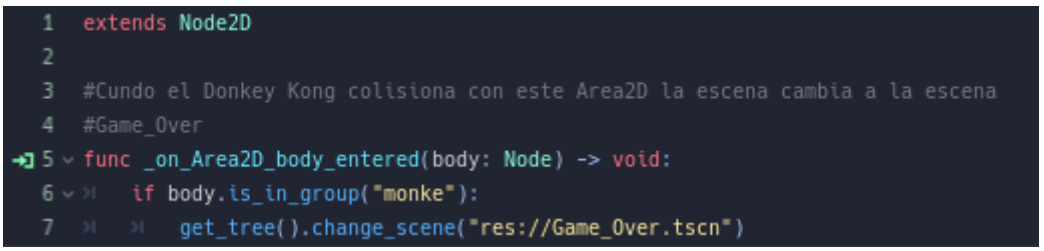

Figura 28. Código de derrota.

Los dos códigos que se encargan de llevar las físicas a cabo son los del barril y el *Donkey Kong*. En el barril se carga la escena del *Donkey Kong*, esto con el fin de poder hacer que aparezca al momento de presionar el botón derecho del mouse, así como también poder enviarle algunos datos que se calculan en este código, como, la dirección a la que debe dirigirse nuestro proyectil, y la gravedad.

```gdscript
extends KinematicBody2D

#Se carga la escena de monke
var monke_scene = preload("res://monke.tscn")
#Se obtiene el nodo de position
onready var monke_spawn: Position2D = get_node("Position2D")

var _vel: = 0.0 #velocidad de rotacion
var _rotacion #rotacion en grados del barril
var shooting = false #Bool que sirve para saber si se ha disparado
var gravity: = 10.0 #gravedad
var speed: = 200.0 #velocidad inicial
var direction: = Vector2() #dirección del disparo

#Se activan las fisicas y los inputs
func _ready() -> void:
    set_physics_process(true)
    set_process_input(true)

#Input del disparo
func _input(event: InputEvent) -> void:
    if event.is_action_pressed("shoot"):
        shooting = true
    elif event.is_action_released("shoot"):
        shooting = false
```

Figura 29. Primera parte del código del barril.

```gdscript
func _process(delta: float) -> void:
    if shooting:
        fire_once() #funcion que se encarga del disparo
    _vel = 5.0
    rotate(_vel * delta) #hace rotar el barril
    _rotacion = fmod(rotation_degrees, 360) #recibe valores de rotacion, no son los del barril
    movimiento(_rotacion) #funcion que calcula la trayectoria del Donkey Kong

#Funcion encargada de que el Donkey Kong salga disparado
func shoot() -> void:
    var monke = monke_scene.instance() #Se pone en una variable las instancias del Donkey Kong
    monke.set_global_position(monke_spawn.get_global_position()) #Obtiene su posición global
    monke.shoot(direction, gravity) #Envia valores a una funcion de la escena Monke
    get_parent().add_child(monke) #Hace aparecer al Donkey Kong

#funcion que se encarga del disparo
func fire_once() -> void:
    shoot()
    shooting = false

#funcion que calcula la trayectoria del Donkey Kong
func movimiento(grados) -> void:
    direction.x = cos(grados) * speed
    direction.y += sin(grados) * speed
```

Figura 30. Segunda parte del código del barril.

El código del *Donkey Kong*, se encarga de hacer que nuestro proyectil se mueve, también nos hace posible moverlo de izquierda a derecha utilizando las teclas “A” y “D” respectivamente.

```
extends KinematicBody2D

#Velocidad inicial de movimiento horizontal provocado por los inputs de movimiento
var speed: = Vector2(100.0, 1.0)
var velocity: = Vector2.ZERO #Velocidad de movimiento horizontal
var _gravity: = 0.0 #variable que recibirá la gravedad
var _movement: = Vector2() #variable que recibirá la trayectoria del Donkey Kong

#Cuando el Donkey Kong toque el barril desaparecerá dando el efecto que
#entró nuevamente al barril
func _on_body_entered(body: Node) -> void:
    if body.get_name() == "Barrel":
        queue_free()
    get_node("CollisionShape2D").disabled = true

#Funcion que recibe los datos dadon en la escena del barril
func shoot(direction, gravity) -> void:
    _gravity = gravity
    _movement = direction
    set_physics_process(true) #Activa las fisicas

#Funcion que se encarga de calcular las fisicas
func _physics_process(delta: float) -> void:
    var dir = movimiento() #Calcula el movimiento horizontal provocado por los inputs de movimiento
    _movement.y += _gravity * delta #calcula el movimiento en y
    _movement.x = speed.x * dir.x #calcula el movimiento en x
    _movement = move_and_slide(_movement) #Hace que el Donkey Kong se mueva

#Calcula el movimiento horizontal provocado por los inputs de movimiento
func movimiento() -> Vector2:
    return Vector2(Input.get_action_strength("move_right") - Input.get_action_strength("move_left"),
    1.0)
```

Figura 31. Código completo del *Donkey Kong*

Para configurar los botones que el usuario utilizará para jugar, debes entrar a *Proyecto*, luego a *Mapa de entrada* y ahí podrás configurarlos.

Figura 32. Controles.

Conclusiones.

Ernesto: A la hora de crear esta recreación fue bastante nostálgico, *Donkey Kong Country* fue el primer videojuego que jugué, fue el que me introdujo al mundo de los videojuegos y por ello le tengo bastante aprecio. En un principio pensé que crearlo sería sencillo, pero fue más complicado de lo imaginado, principalmente en la parte de obtención de dato, obtener los datos necesarios para hacer que el *Donkey Kong* saliera volando en el ángulo correcto, hacia donde el barril está apuntando fue imposible, hacer que saliera desde la salida del barril era sencillo, pero que siguiera esa trayectoria no lo fue, igualmente al momento de caer lo hacía demasiado lento, y cuando salía disparado lo hacía a velocidades diferentes de manera aleatoria o simplemente no se elevaba y esos son errores que no logró aún comprender.

Necesito seguir estudiando como funciona Godot, este fue el primer videojuego que creó en este engine y no será el último, de eso estoy seguro. Una vez domine mejor Godot, regresaré a este proyecto y corregir esos errores que no comprendo aún.

Nataly: Este proyecto definitivamente fue un reto para mi, en este curso fue la primera vez que que escuche de un *engine* de videojuegos y también mi primera vez haciendo un juego, hubiera sido una gran idea llevar programación orientado a objetos antes de llevar esta clase, por el hecho de que nunca había hecho nada parecido, me tocó iniciar de ceros sin saber que son los nodos, el nodo padre, las escenas y que se tiene que hacer cada una por separado, Ernesto me ayudó mucho a entender que hace cada cosa, elegir Godot como nuestro *engine* por que hay muchos videos tutoriales y documentación que te ayudan con la mayoría de problemas que tengas, al momento de programar las ecuaciones físicas del juego me sentí un poco más familiarizada ya que el lenguaje de Godot es similar a python y con python tengo más experiencia ya que ahí realizó la mayoría de las prácticas computacionales, definitivamente me gustaría aprender más del mundo de los videojuegos y de su realización.

Referencias.

Fundamentals of physics extended, 10th edition, David Halliday, Robert Resnick, Jearl Walker

https://www.youtube.com/watch?v=BlPqzLFbgoE

https://www.youtube.com/watch?v=RbvteTMELL8

https://www.youtube.com/watch?v=DNAiZUuaW_U&t=450s

https://www.youtube.com/watch?v=mjWwWIEyib8&t=1795s

https://www.youtube.com/watch?v=4hRmSmy2Ays

https://www.youtube.com/watch?v=ucz96DSfSZs

https://www.youtube.com/watch?v=6ziIyx60N6I&t=1447s

https://www.youtube.com/watch?v=Mc13Z2gboEk&t=5740s

https://www.youtube.com/watch?v=9YMAogsxETc

https://www.youtube.com/watch?v=GvD2idlFEqY&t=251s