Guía Definitiva: Crear PONG en Godot 4

Práctica intensiva de 2 horas para estudiantes de ingeniería y desarrollo de videojuegos.

Motor: Godot Engine 4.x | Lenguaje: GDScript

1. Estructura de Nodos del Proyecto (`Main`)

Para que el juego funcione correctamente con las físicas y señales que veremos más adelante, el árbol de nodos en tu panel de escena debe verse exactamente de la siguiente manera:

```
O Main (Node2D + script main.gd)
├── ⊡ ParedArriba (StaticBody2D)
│   └── □ CollisionShape2D
├── ⊡ ParedAbajo (StaticBody2D)
│   └── □ CollisionShape2D
├── ⊡ LineaCentral (StaticBody2D)
│   └── □ CollisionShape2D
├── -- Player (CharacterBody2D + script player.gd)
│   ├── ☻ Sprite2D
│   └── □ CollisionShape2D
├── -- Enemy (CharacterBody2D + script enemy.gd)
│   ├── ☻ Sprite2D
│   └── □ CollisionShape2D
├── ⊡ Ball (Area2D + script ball.gd + Señal body_entered)
│   ├── ☻ Sprite2D
│   └── □ CollisionShape2D
├── ⊡ GolDerecha (Area2D + Señal area_entered)
│   └── □ CollisionShape2D
└── ⊡ GolIzquierda (Area2D + Señal area_entered)
    └── □ CollisionShape2D
```

> **Configuración de Pantalla Recomendada**
>
> Ve a **Proyecto > Configuración del proyecto > Pantalla** y define un ancho de 1152 y alto de 648 píxeles. Coloca las paredes superior e inferior en los límites exactos (Y=0 e Y=648).

2. Código del Nodo Raíz (`main.gd`)

Este script se encarga de gestionar el marcador global de puntos y reiniciar la posición de la pelota cuando esta cruza las zonas de gol laterales.

```
extends Node2D

var score_left: int = 0
var score_right: int = 0

# Referencia al nodo de la pelota
@onready var ball: Area2D = $Ball

# Activado cuando la pelota cruza el borde izquierdo (Punto para la derecha)
func _on_gol_izquierda_area_entered(area: Area2D) -> void:
	if area == ball:
		score_right += 1
		print("Punto para la derecha! Marcador: ", score_left, " - ", score_right)
		ball.reset_ball()

# Activado cuando la pelota cruza el borde derecho (Punto para la izquierda)
func _on_gol_derecha_area_entered(area: Area2D) -> void:
	if area == ball:
		score_left += 1
		print("Punto para la izquierda! Marcador: ", score_left, " - ", score_right)
		ball.reset_ball()
```

3. Código del Jugador (`player.gd`)

Asignado al nodo `Player` (CharacterBody2D). Permite el movimiento vertical fluido limitado a los márgenes de la ventana.

```
extends CharacterBody2D

@export var speed: float = 400.0

func _physics_process(delta: float) -> void:
	var direction := 0.0
	if Input.is_action_pressed("ui_up") or Input.is_action_pressed("w"):
		direction -= 1.0
	if Input.is_action_pressed("ui_down") or Input.is_action_pressed("s"):
		direction += 1.0

	velocity.y = direction * speed
	move_and_slide()

	# Limitar la paleta dentro de los márgenes superior e inferior
	global_position.y = clamp(global_position.y, 50, 598)
```

4. Código del Enemigo / IA (`enemy.gd`)

Asignado al nodo `Enemy` (CharacterBody2D). Incluye un margen de tolerancia para evitar el parpadeo o vibración por sobrecorrección.

```
extends CharacterBody2D

@export var speed: float = 300.0
@onready var ball: Area2D = get_node("../Ball")

func _physics_process(delta: float) -> void:
	if ball:
		var direction := 0.0
		var tolerance := 10.0 # Margen para evitar vibración

		if ball.global_position.y < global_position.y - tolerance:
			direction = -1.0
		elif ball.global_position.y > global_position.y + tolerance:
			direction = 1.0

		velocity.y = direction * speed
		move_and_slide()

		global_position.y = clamp(global_position.y, 50, 598)
```

5. Código de la Pelota (`ball.gd`)

Asignado al nodo `Ball` (Area2D). Controla el movimiento vectorial continuo, el incremento de velocidad al golpear paletas y los rebotes limpios contra las paredes.

```
extends Area2D

var speed: float = 400.0
var direction: Vector2 = Vector2.ZERO

func _ready() -> void:
	reset_ball()

func _process(delta: float) -> void:
	global_position += direction * speed * delta

func reset_ball() -> void:
	global_position = Vector2(576, 324) # Centro de la pantalla
	speed = 400.0
	var random_x = randf_range(-1.0, 1.0)
	while abs(random_x) < 0.3:
		random_x = randf_range(-1.0, 1.0)
	var random_y = randf_range(-0.6, 0.6)
	direction = Vector2(sign(random_x), random_y).normalized()

# Maneja las colisiones con paletas y paredes estáticas
func _on_body_entered(body: Node2D) -> void:
	if body.name == "Player" or body.name == "Enemy":
		direction.x = -direction.x
		speed += 25.0
		var diff = global_position.y - body.global_position.y
		direction.y = diff / 40.0
		direction = direction.normalized()

	elif body.name == "ParedArriba" or body.name == "ParedAbajo":
		direction.y = -direction.y
```

6. Conexión de Señales Clave

Para que el juego funcione sin errores de tipos, verifica las siguientes conexiones en el panel **Nodos** de Godot:

- **Ball (`body_entered`):** Conectar al propio script de la pelota para manejar rebotes con paletas y paredes.
- **GolDerecha (`area_entered`):** Conectar al nodo `Main` apuntando a la función `_on_gol_derecha_area_entered`.
- **GolIzquierda (`area_entered`):** Conectar al nodo `Main` apuntando a la función `_on_gol_izquierda_area_entered`.