

# Arkanoid: Deconstrucción y Renacimiento en Godot

## Un Informe Técnico para Desarrolladores y Preservacionistas Digitales

La recreación de *Arkanoid* (Taito, 1986) en un motor moderno trasciende la programación; es un acto de preservación digital y un estudio profundo sobre la evolución de las mecánicas de juego. Este documento es una hoja de ruta exhaustiva para una reconstrucción fiel, desde la integridad narrativa hasta la arquitectura de software.



ARKANOID



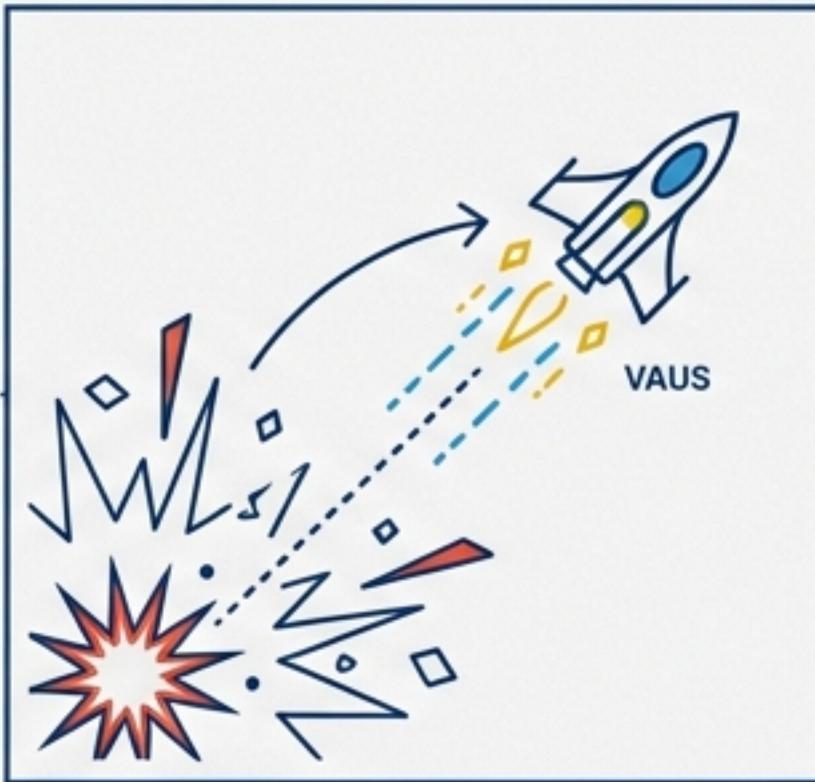
# El Archivo Narrativo: Informe de Misión

A menudo subestimado, *Arkanoid* posee una narrativa de ciencia ficción que contextualiza su jugabilidad abstracta.



## El Incidente de la Nave Nodriza

La historia comienza con la destrucción de la nave colonia 'Mothership Arkanoid' por una fuerza desconocida.



## La Nave de Escape Vaus

El jugador no controla una 'pala', sino la nave de escape modelo **Vaus**, eyectada momentos antes de la destrucción de la Arkanoid.



## El Laberinto Espacial

La Vaus es atrapada en un 'espacio deformado', una distorsión dimensional creada por el antagonista principal.

# El Antagonista: DOH (Dominate Over Hour)



## Descripción de la Entidad

El enemigo final es una entidad consciente biomecánica con control sobre el espacio-tiempo. Su nombre, “Dominador sobre la Hora”, explica mecánicas como el “Warp”.

## Iconografía

Se manifiesta como una cabeza de piedra gigante similar a un Moai, una amalgama de tecnología antigua y alienígena.

### El Bucle Temporal (Final Arcade)

Al derrotar a DOH, el tiempo fluye hacia atrás, la Arkanoid se reconstruye, y el texto final advierte: “el viaje solo ha comenzado”, atrapando a la Vaus en un ciclo eterno.



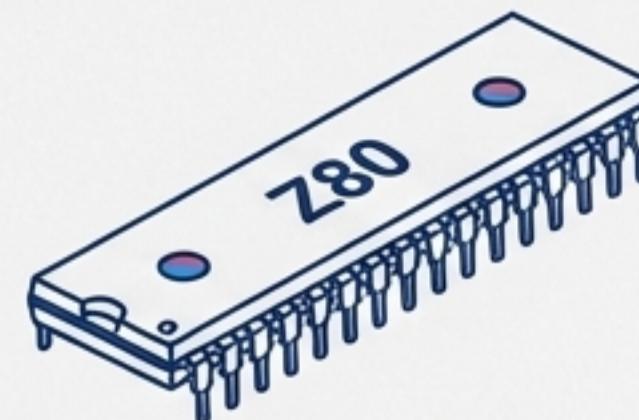
### El 13º Amanecer Estelar (Final Verdadero)

En la secuela se revela un final definitivo, donde la humanidad encuentra un nuevo hogar y DOH es desterrado “al olvido”.



# El Reto de la Traducción: De la Lógica Z80 a los Nodos de Godot

## Lógica Original (1986)



**Hardware:** Procesador Z80, lógica discreta basada en ciclos de reloj.

**Colisión:** Comprobación de superposición de sprites en momentos específicos del barrido de pantalla.

Controlador: 'Spinner' analógico que permite aceleración proporcional.

## Lógica Moderna (Godot 4.x)



**Motor:** Orientado a objetos, basado en nodos (PhysicsServer2D).

**El Riesgo:** Un uso ingenuo de `RigidBody2D` resulta en un comportamiento “demasiado realista”, traicionando el *game feel* original.

**La Solución:** Usar `CharacterBody2D` y calcular manualmente los vectores de rebote para replicar el control preciso de 1986. Implementar una curva de aceleración para simular la inercia del 'spinner'.

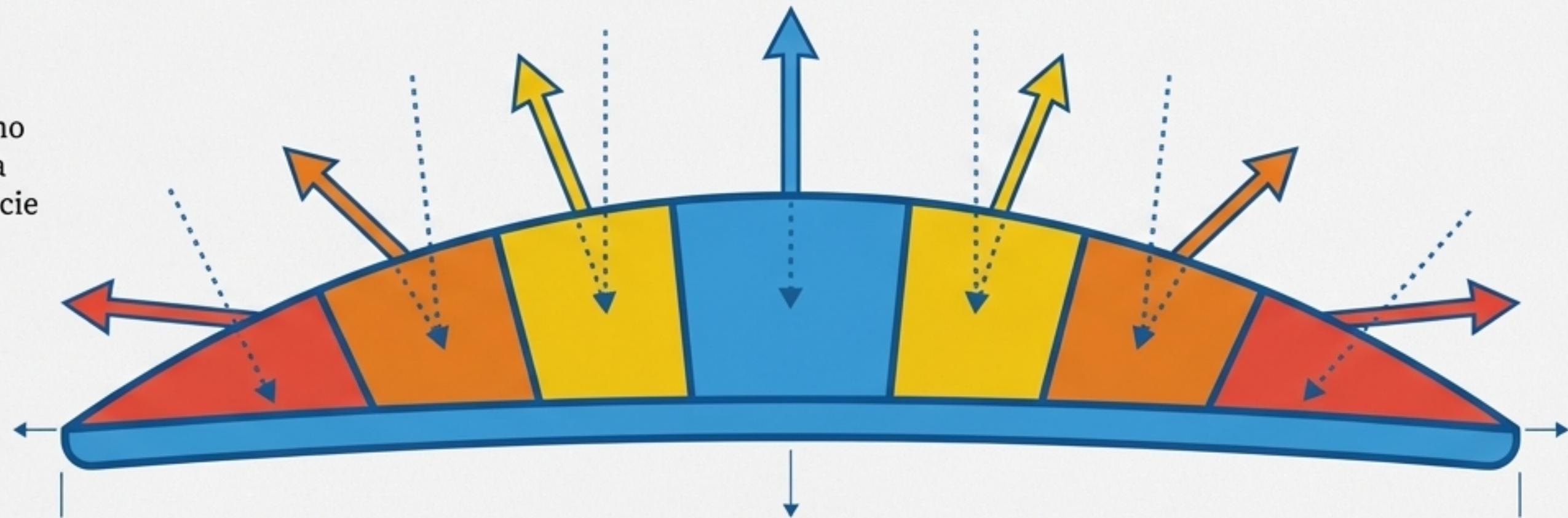
# La Anatomía del Rebote: El "Efecto Inglés"

## Principio Fundamental

El ángulo de rebote de la pelota no depende del ángulo de incidencia, sino de **dónde** golpea en la superficie de la pala. La pala actúa como una superficie convexa, no plana.

## Zonas de la Pala

- **Centro:** Rebote vertical o con ángulo conservado.
- **Bordes:** Rebote agudo con mayor velocidad horizontal.



## Lógica de Implementación

Esta mecánica permite a los jugadores 'apuntar' la pelota. La recreación fiel exige implementar una fórmula que calcule el nuevo ángulo basado en la distancia del punto de impacto al centro de la pala.

```
# Lógica conceptual para el rebote
var diferencia_x = centro_pelota.x - centro_pala.x
var ancho_pala = colision_pala.shape.extents.x
var factor_rebote = diferencia_x / (ancho_pala / 2) # Rango: -1.0 a 1.0
var angulo_nuevo = factor_rebote * ANGULO_MAXIMO
vector_velocidad = Vector2.UP.rotated(deg_to_rad(angulo_nuevo)) * velocidad_actual
```

# La Jerarquía de los Ladrillos y su Economía de Puntuación

## Ladrillos de Color (Estándar)

Color	Puntos
	50
	60
	70
	80
	90
	100
	110
	120

## Ladrillo Dorado (Gold)



Durabilidad: Infinita (Indestructible)

Función: Obstáculo de terreno.

## Ladrillo Plateado (Silver)

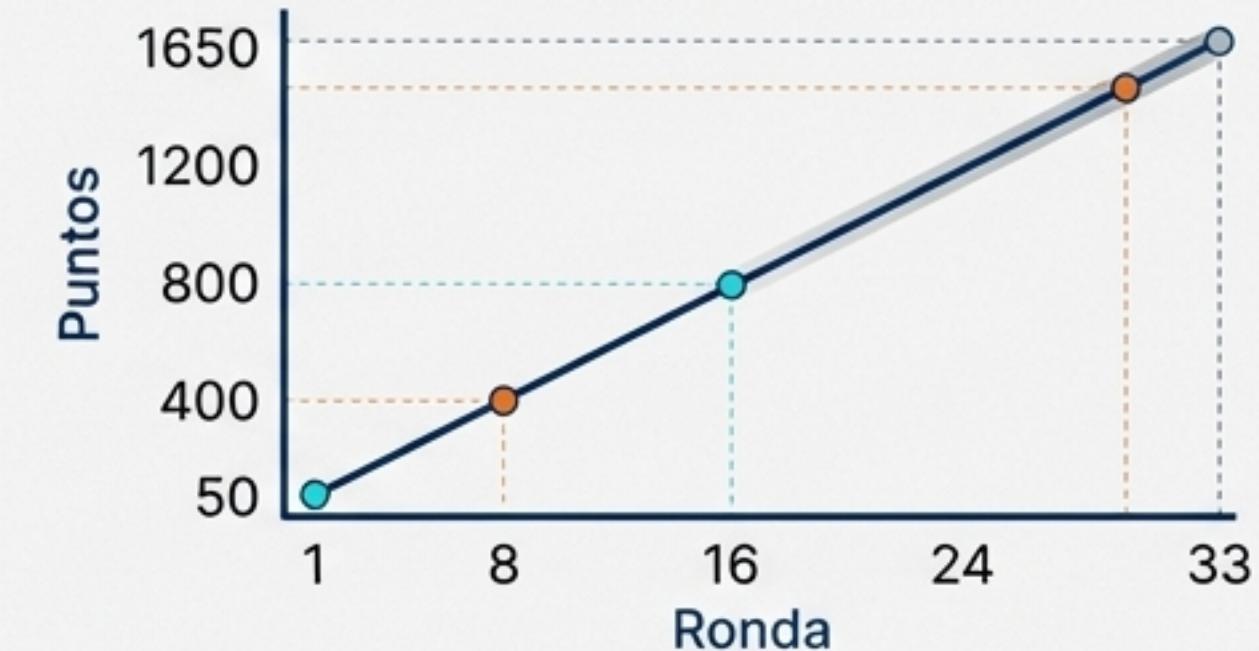


Durabilidad: Requiere múltiples impactos; la resistencia aumenta con el número de ronda (2 golpes en rondas 1-8, 3 en 9-16, etc.).

Puntuación Dinámica:

‘50 puntos \* Número de Ronda’

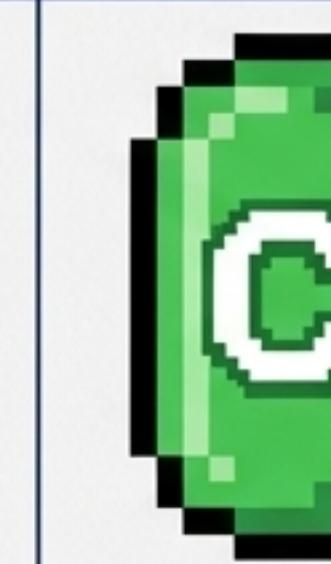
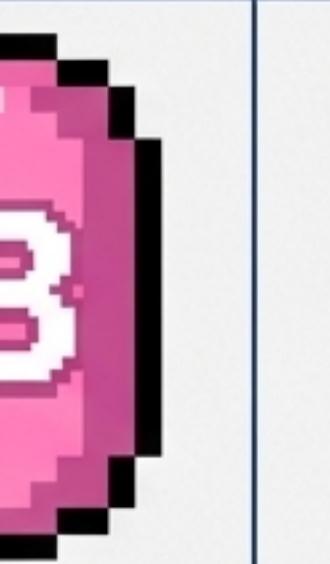
### Valor del Ladrillo Plateado



**\*Insight de Diseño:** Esta fórmula convierte un simple obstáculo en el objetivo de mayor valor y riesgo en el juego tardío. En la ronda 30, un ladrillo plateado vale 1,500 puntos, incentivando la maestría.\*

# El Arsenal Estratégico: Catálogo de Potenciadores

**Regla Clave:** Solo un potenciador puede estar activo a la vez (excepto la vida extra).

						
S (Naranja) - Slow	L (Rojo) - Laser	C (Verde) - Catch	E (Azul) - Expand	D (Cian) - Disruption	B (Rosa) - Break	P (Gris) - Player
Reduce la velocidad de la pelota a su valor base.  <i>Implementación:</i> Instanciar escenas 'LaserProjectile.tscn' desde 'Marker2D' en la Vaus.	Transforma la Vaus.  <i>Implementación:</i> Instanciar escenas 'LaserProjectile.tscn' desde 'Marker2D' en la Vaus.	La pelota se adhiere a la pala.  <i>Implementación:</i> Emparentar temporalmente el nodo de la pelota a la Vaus.	Alarga la Vaus.  <i>Implementación:</i> Escalar el 'Sprite2D' Y el 'CollisionShape2D' en el eje X.	Divide la pelota en tres.  <i>Implementación:</i> Usar un grupo ('Balls') para gestionar la condición de derrota.	Abre una puerta de 'Warp' para saltar el nivel.	Otorga una vida extra.

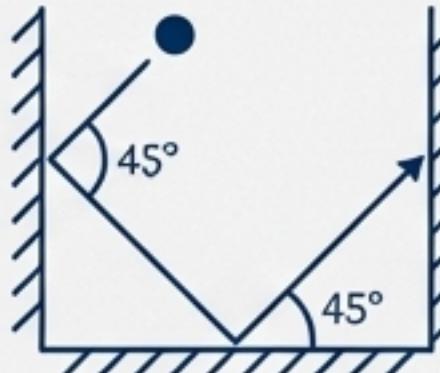
# Amenazas Cinéticas: Catálogo de Enemigos

A diferencia de *Breakout*, *Arkanoid* presenta enemigos móviles que descienden y actúan como peligros cinéticos, no como atacantes directos.



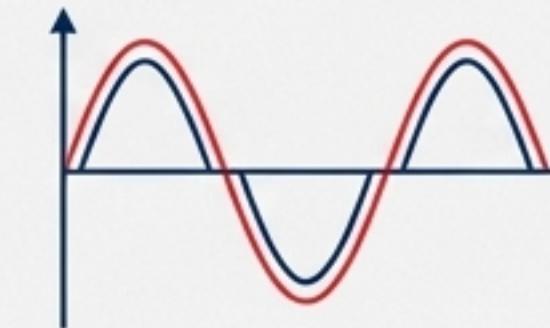
**Konerd**

Forma geométrica, movimiento diagonal simple rebotando en paredes.



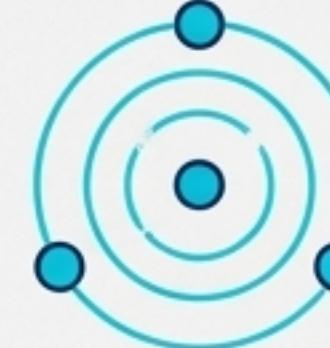
**Pyradok**

Pirámide segmentada, movimiento en zig-zag cerrado. (\*Godot: Animar `PathFollow2D` sobre una curva sinusoidal.\*)



**Tri-Sphere**

Tres esferas orbitales, pulsan cambiando su tamaño de colisión.



**Opopo**

Similar a una medusa, movimiento errático o aleatorio.



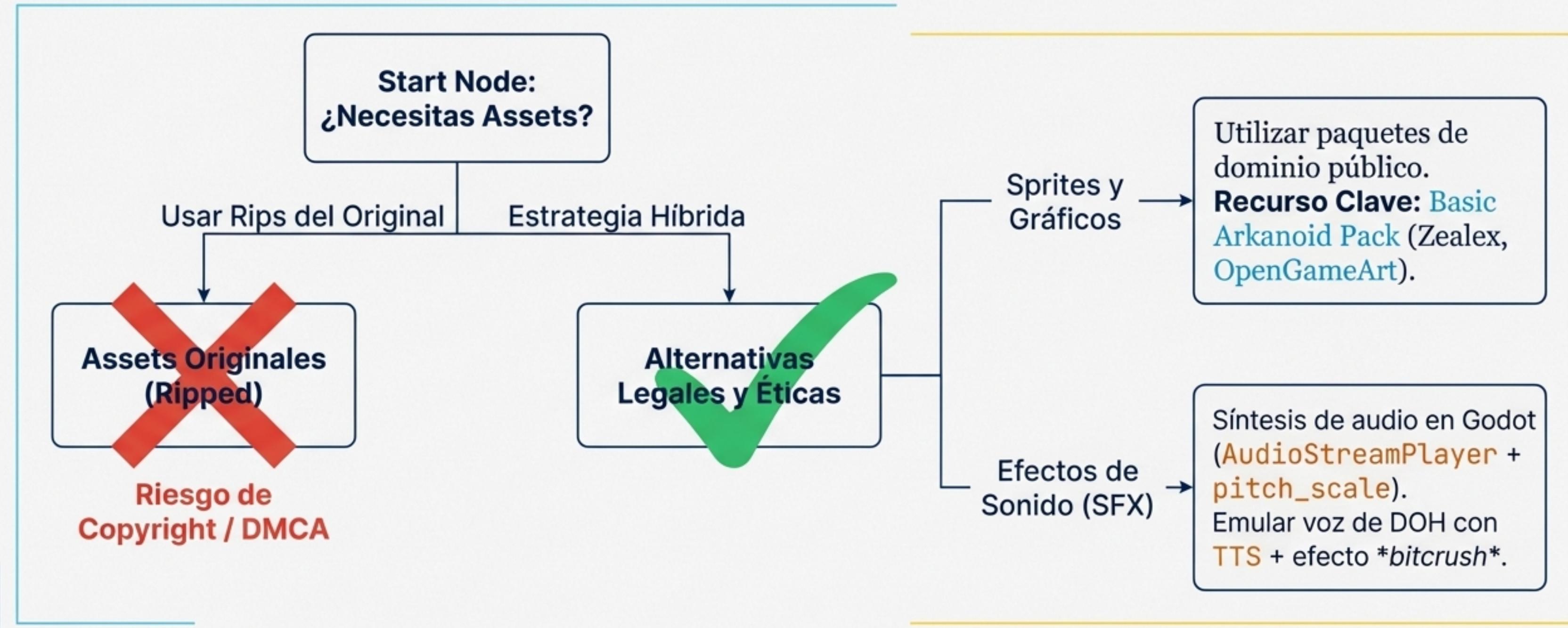
## Reglas de Interacción

- Pelota vs Enemigo:** Enemigo destruido, la pelota rebota con un ángulo aleatorizado.
- Vaus vs Enemigo:** Vaus destruida, se pierde una vida.
- Láser vs Enemigo:** Enemigo destruido, otorga 100 puntos.



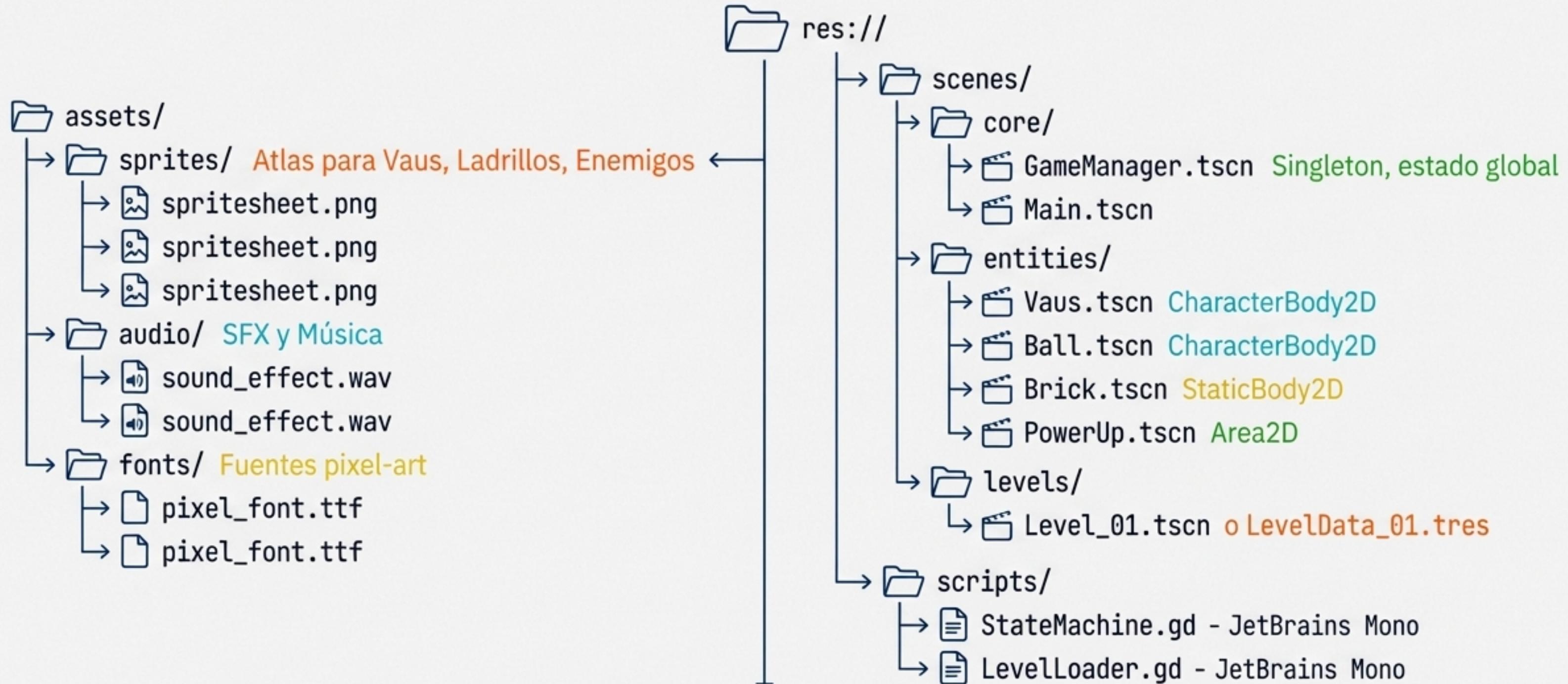
# Estrategia de Activos: Adquisición Legal y Ética

La Realidad Legal: No existe un repositorio público legal con los assets originales de Taito debido al copyright. Los repositorios que los alojan son eliminados vía DMCA.



# El Plano del Proyecto: Arquitectura de Archivos en Godot

Una arquitectura modular es esencial para gestionar un proyecto de esta magnitud. Se recomienda la siguiente estructura para mantener el orden y la escalabilidad.

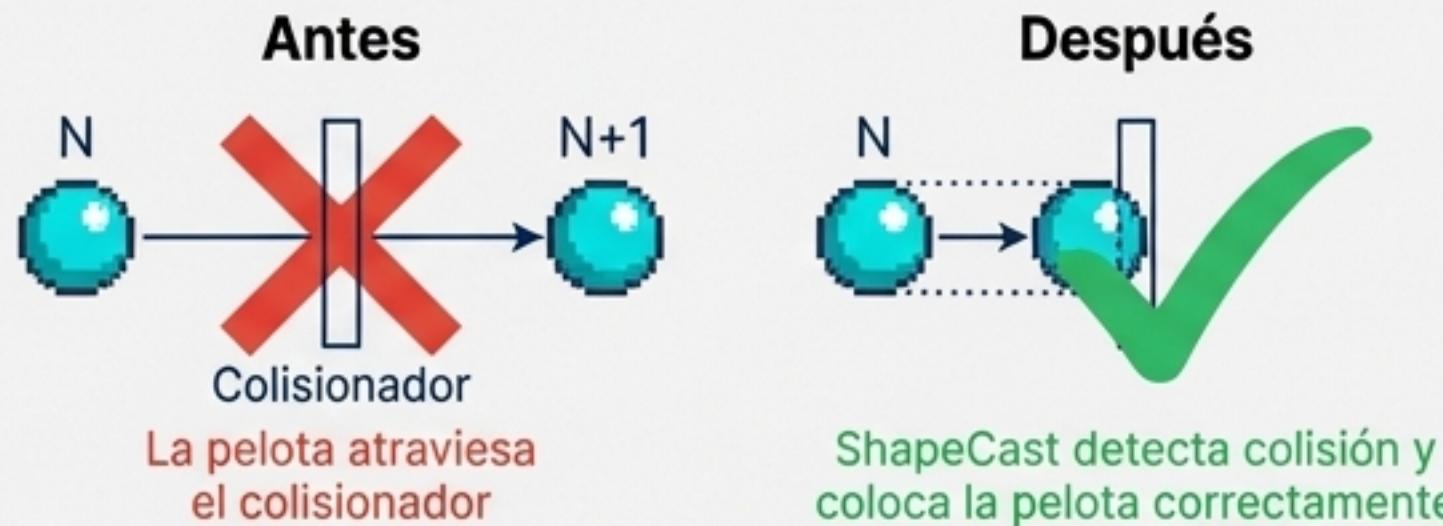


# Soluciones de Ingeniería: Tunneling y Generación de Niveles

Un enfoque técnico para resolver problemas comunes en el desarrollo de Arkanoid.

## “Tunneling” (Atravesar Objetos)

**Causa:** A altas velocidades, el desplazamiento de la pelota por frame es mayor que el grosor del colisionador.



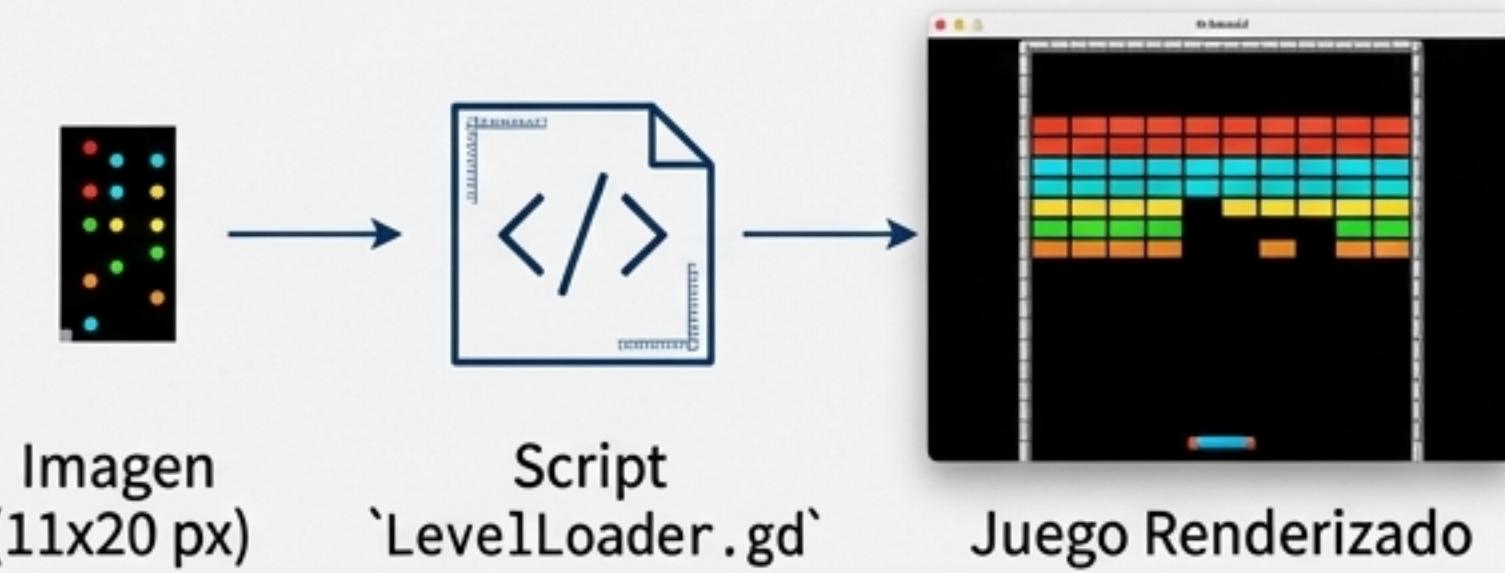
**Solución en Godot 4:** Usar `ShapeCast2D`. Antes de mover la pelota, lanzar un ShapeCast en la dirección del movimiento. Si detecta colisión, mover la pelota exactamente al punto de impacto, no más allá.

**Alternativa:** Aumentar `physics\_ticks\_per\_second` en la configuración del proyecto (ej. de 60 a 120), a costa de un mayor uso de CPU.

## Problema 2: Generación de Niveles: Generación de Niveles por Datos

**Método Ineficiente:** Colocar ladrillos manualmente en el editor.

**Solución Escalable:** Usar imágenes (ej. 11x20 píxeles) donde cada píxel representa un ladrillo. Un script `LevelLoader` lee el color de cada píxel e instancia la escena `Brick.tscn` correspondiente en la posición correcta.



# El Enfrentamiento Final: Plano de la Batalla contra DOH

En la Ronda 33, se deshabilita la generación de ladrillos y se instancia la escena del jefe, `BossDoh.tscn`.



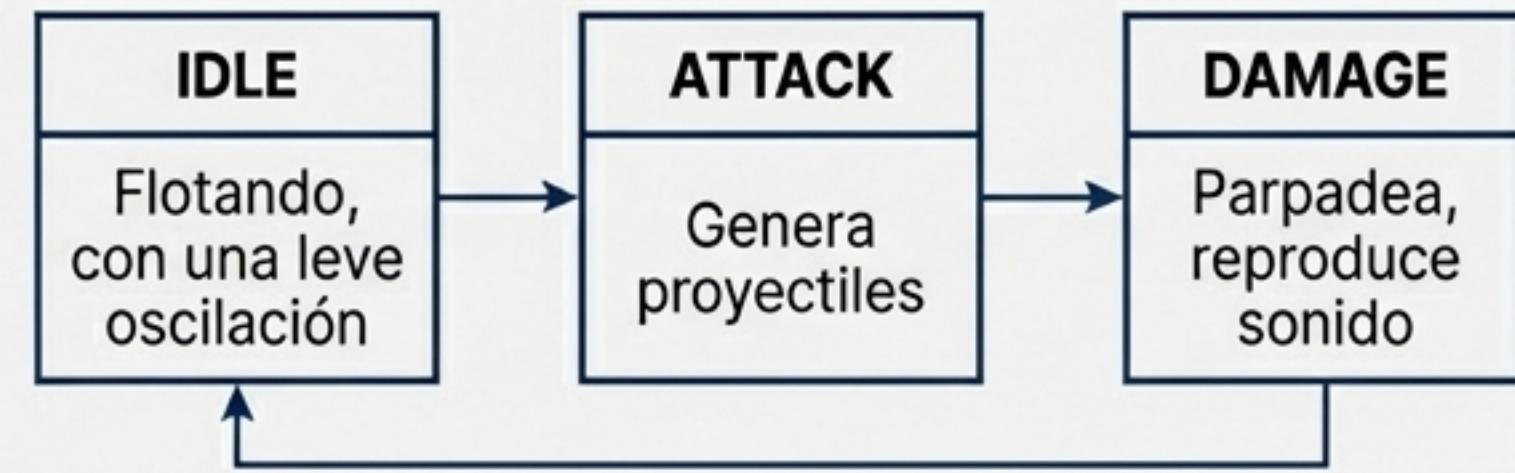
Vulnerable Hitbox

DOH no tiene una única área de colisión. El cuerpo principal es invulnerable; solo la boca o los ojos reciben daño.

Se deben usar múltiples nodos `CollisionShape2D`.

Invulnerable

## Componente 2: Máquina de Estados Finitos (FSM)



## Componente 3: Proyectiles del Jefe

DOH dispara sus propios proyectiles, que deben ser esquivados por la Vaus mientras se intenta golpear a DOH con la pelota.



# Más que un Clon: Un Artefacto de Software

```
func _physics_process(delta):  
    move_and_slide()
```

```
        signal brick_broken(points)
```

“ La recreación de *Arkanoid* es una lección magistral de diseño de juegos. Exige rigor matemático para la física, creatividad arquitectónica para los potenciadores y sensibilidad artística para capturar su atmósfera. ”

```
var ball_speed = 400.0  
var ball_speed =  
var ball_speed =  
var seitst_roldst_dFE = 0  
return broken
```



Al adherirse a las especificaciones detalladas en este informe, el resultado no será una simple copia, sino un artefacto de software que respeta y preserva la memoria de uno de los grandes hitos de la historia del arcade.

```
func ball_broken(ball):  
    var ball_speed = 400.0  
    signal brick_broken(points)
```

```
func _brick_process():  
    func _physics_process(delta):  
        move_and_slide()
```

```
func _ball_speed = 400 as ball_speed:  
    var ball_speed = 400. as 400.
```

```
func __move_spree(delta):  
    move_and_slide()
```

```
func _brick_broken(points):  
    signal brick_broken(points)
```

# El Ciclo Eterno

El final original del arcade atrapaba al jugador en un bucle temporal. Al recrear Arkanoid, aseguramos que su ciclo continúe, no como una prisión para la Vaus, sino como una experiencia atemporal para nuevas generaciones de jugadores y desarrolladores.



**“el viaje solo ha comenzado...”**