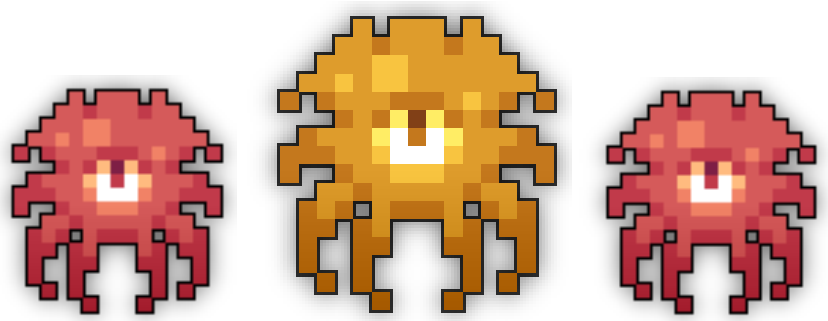
# Angryliens



**Curso: Física para programadores de videojuegos 2021-1**

**Universidad Autónoma de Baja California**

**Facultad de ciencias**

**28 de mayo de 2021**

**Alumnos:**

**Eliaf Yahir Garcia Loya**

**Armando Rosas Pérez**

**Docente:**

**Eloísa del Carmen García Canseco**

## Objetivo:

Antes de la creación de motores gráficos para videojuegos, no había separación entre áreas como la gráfica y la física. No fue hasta los años 80´s , 90´s que se empezaron a implementar estos motores en la creación de videojuegos.

En esta práctica se busca programar un videojuego 2D en Godot Engine e implementar el área física necesaria, y no usar las físicas integradas al motor. De esa manera, tener un mejor entendimiento del funcionamiento de los motores gráficos.

## Introducción:

En los juegos 2D de los últimos años, han existido varias franquicias exitosas, una de ellas ha sido AngryBirds siendo el juego más vendido de la historia en soportes móviles, juego que consiste en rasgos generales en la eliminación de enemigos disparando pájaros a través de un cañón. En cambio un juego como Space Invaders, hecho a finales de la década de los 70 no sólo fue un rotundo éxito, sino que cambió la forma en que vemos los videojuegos, este último creado prácticamente sin motor gráfico, donde se busca eliminar a los alienígenas disparando desde una nave mientras estos se acercan.

Así que inspirados por estos míticos juegos, decidimos crear una mezcla, dando inicio a Angryliens, que funciona con mecánicas similares a los anteriores.

La oportunidad de implementar el tiro parabólico como el área física primordial, y generar enemigos que reaccionen a las colisiones entre ellos y los disparos, fue la combinación perfecta para cumplir con el objetivo de este proyecto. Además, en los últimos años ha existido un creciente interés por la temática alienígena, por lo tanto decidimos generar ideas acordes a las tendencias actuales.

# Descripción del videojuego:

## Parte 1

El videojuego comienza con la aparición de un menú de inicio que despliega cuatro opciones:

Jugar: comienza el juego.

Tutorial: muestra en pantalla los controles del juego

Créditos: despliega en pantalla a los desarrolladores del juego.

Salir: cierra el juego.


Imagen ilustrativa 1. Menú de inicio.

## Parte 2.

Al presionar la opción de jugar en el  menú de inicio, nos lleva a la escena del juego donde este mismo se ejecuta, comenzamos con una puntuación de 0 y cinco vidas. Conforme avanza el tiempo, aparecen 4 distintos enemigos de manera aleatoria en la parte derecha de la pantalla, que se mueven de distintas formas hasta llegar a la izquierda, donde restan una vida al jugador. Los enemigos son eliminados al disparar el arma del jugador y acertar en la “hitbox” de estos, cada enemigo eliminado suma un punto.

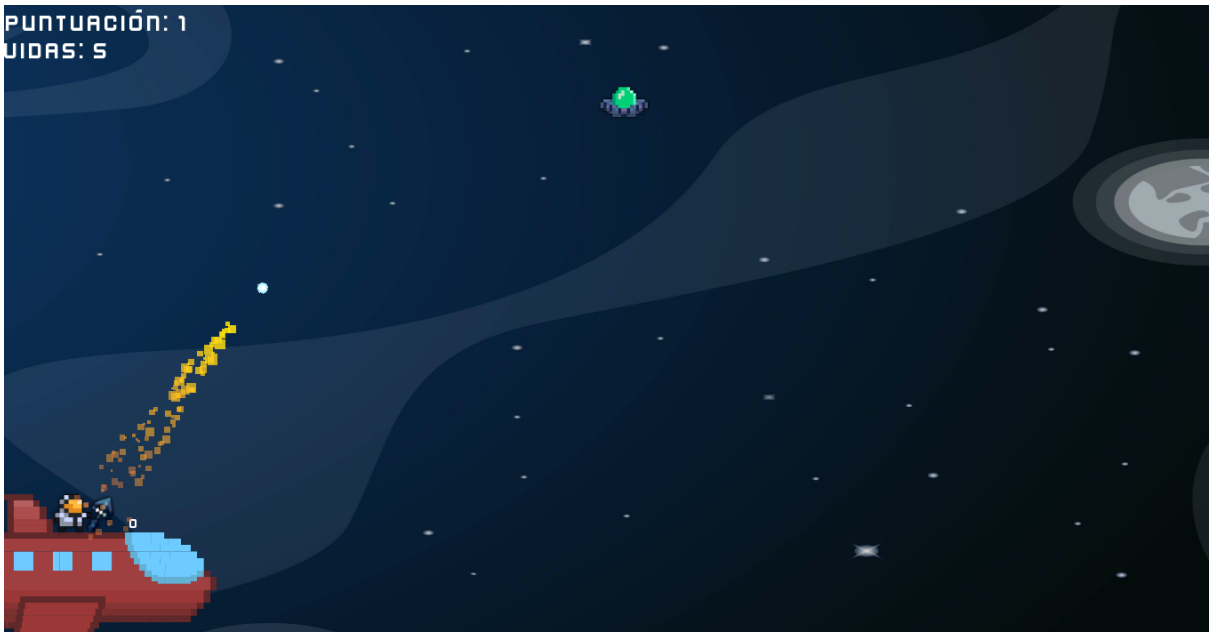



Imagen ilustrativa 2. Inicio del juego.

Cada 4 y 8 enemigos aparecerá un enemigo especial, y al décimo enemigo aparecerá un jefe enemigo.

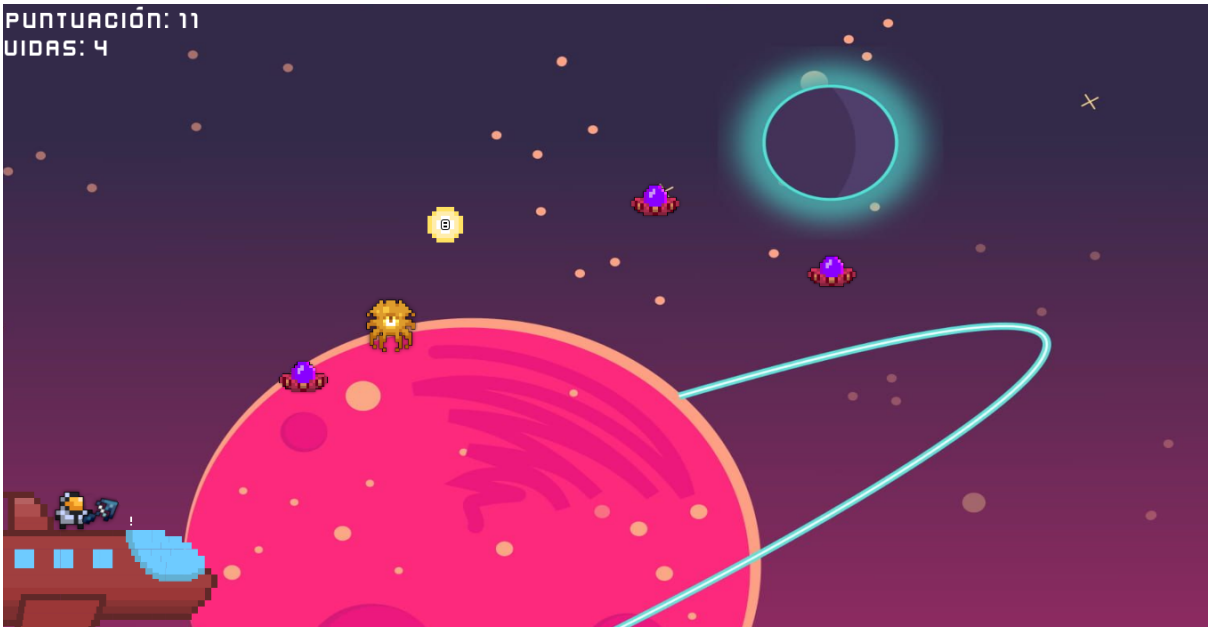



Imagen ilustrativa 4. Enemigo final.

## Parte 3.

Cuando el contador de vidas llega a cero, el juego termina y nos envía a la escena del menú final, donde tenemos dos opciones:

Jugar: volvemos a la escena del juego y podemos comenzar una nueva partida.

Menú: nos envía al menú de inicio.


Imagen ilustrativa 5. Menú final.

## Desarrollo del proyecto:

Para este proyecto se crearon 12 escenas, y 16 scripts, un script para las variables globales, uno de control, dos para el escudo del jefe enemigo, y 12 para cada escena.

Las escenas fueron las siguientes:

**MainMenu.tscn:** esta escena cuenta con 16 nodos, siendo el principal “Node2D” que tiene de hijo al nodo de control. Tal y como se muestra en la figura ilustrativa 5. Cuenta con dos scripts, uno para el movimiento de los “sprites” en el menú principal y otro de control para el funcionamiento de las opciones del menú. Cuenta con 4 “TextureButton”, uno para el “NinePatchRect”, otro para “VboxContainer”, 5 “labels”, 5 “sprites” y un “AudioStreamPlayer”.

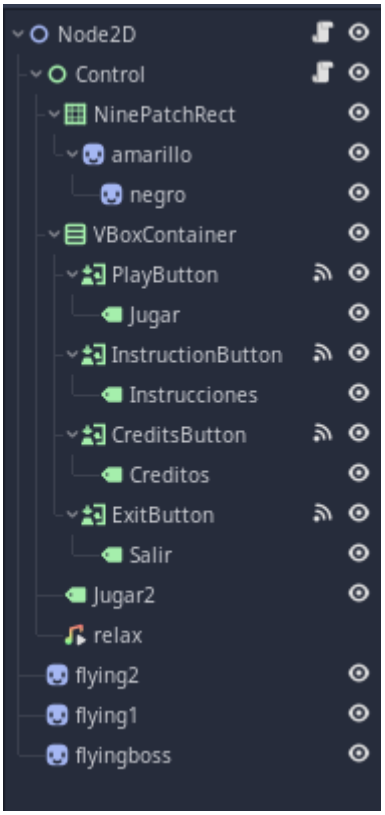



Figura Ilustrativa 5. Nodos de la escena MainMenu.tscn

**Tutorial.tscn:** hay 11 nodos en esta escena, “Node2D”, “Control”, ”NinePatchRect”, “VBoxContainer”, un “TextureButton”, dos “label”, tres “sprites” y un “AudioStreamPlayer”. El Node2D tiene un adjunto un script que permite el uso del botón “regresar” , y  el nodo de control tiene adjunto el script de control.

**Credits.tscn:** cuenta con 8 nodos, Node2D, el de Control, “NinePatchRect”, “VBoxContainer”, un “TextureButton”, 2 “label” y un “AudioStreamPlayer”. Hay adjunto dos scripts, uno para el Node2D, que permite el uso del botón “regresar” y otro para el nodo de control.

**Bullet.tscn:** 5 nodos se encuentran en esta escena, Node2D, un “sprite”, “Area 2D”, “CollisionPolygon2D” y “Particles2D”. Cuenta con un script que describe el comportamiento de tiro parabólico en la bala disparada por el arma del jugador.

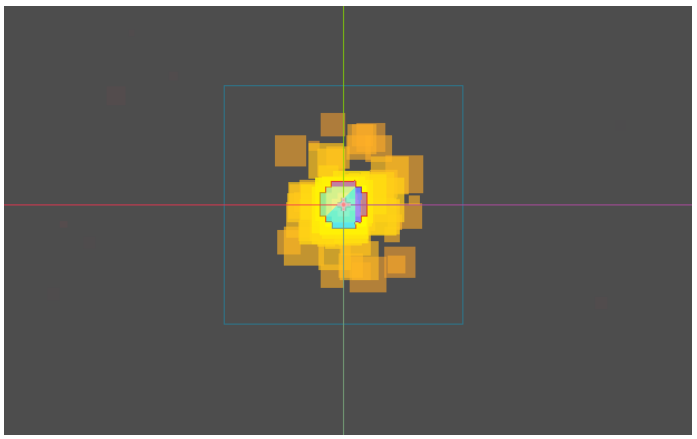


Figura ilustrativa 6. Escena Bullet.tscn

**AdvEnemy.tscn:** Contiene los nodos que describen al enemigo de comportamiento cosinusoidal. Entre ellos se encuentran el Sprite, así como el área de colisión (Area 2D) y su polígono respectivo de colisión (CollisionPolygon2D)

Imagen Ilustrativa 7. $CO_2$-Zînüzoidãl: Movimiento cosinusoidal.

**BasicEnemy.tscn** : Contiene los nodos que describen al enemigo de comportamiento básico (movimiento en una línea recta, con velocidad constante). Entre ellos se encuentran el Sprite, así como el área de colisión (Area 2D) y su polígono respectivo de colisión (CollisionPolygon2D).

Imagen ilustrativa 8. Vêisyc: Movimiento de derecha a izquierda.

**MiniEnemy.tscn :** Contiene los nodos que describen al enemigo de comportamiento básico (movimiento en una línea recta, con velocidad constante). Entre ellos se encuentran el Sprite, así como el área de colisión (Area 2D) y su polígono respectivo de colisión (CollisionPolygon2D). Cabe destacar que Area2D de este enemigo es más chica que el enemigo básico.

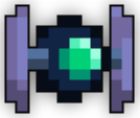


Imagen ilustrativa 9. Mini: Movimiento de derecha a izquierda.

**GreenEnemy.tscn :** Contiene los nodos que describen el comportamiento del enemigo “Big-g-reen”, (movimiento diagonal, con velocidad constante). Entre ellos se encuentran el Sprite, así como el área de colisión (Area 2D) y su polígono respectivo de colisión (CollisionPolygon2D). Este enemigo además tiene un nodo AudioStreamPlayer para reproducir el sonido cuando se teletransporta.

Imagen ilustrativa 10. Bi-g-rêên: Movimiento diagonal y de teletransportación.

**ShieldEnemy.tscn:** Esta escena pertenece a un enemigo que no está actualmente implementado en el juego, su comportamiento es aquel de **B.O.S.S**, sin embargo, su movimiento sería circular de derecha a izquierda. Este enemigo contiene los nodos de cualquier otro enemigo, a diferencia de de un Sprite adicional con su respectiva Area2D y CollisionShape2D, el cual representaría el escudo que estuviera girando alrededor de este enemigo.

**BossEnemy.tscn:** Contiene los nodos para el enemigo jefe. Entre ellos se encuentran el Sprite, así como el área de colisión (Area 2D) y su polígono respectivo de colisión (CollisionPolygon2D). Además, este enemigo contiene elementos adicionales como una label para denotar el tiempo de un Timer de 9 segudos, el cual se encarga de tomar el tiempo necesario para atacar al jugador. Este enemigo también contiene un sprite que representa su vida faltante. Por último, este jefe contiene un Sprite adicional, el cual sirve como escudo.

Imagen ilustrativa 11. B.O.S.S.: Gira alrededor de un punto en la pantalla y posee un escudo.

**GameOver.tscn:** Esta escena contiene la información referente a un Game Over: los botones para la decisión del usuario y el texto que indica que ha perdido la partida.

## Conclusiones:

Llevar a cabo un videojuego con las físicas como objetivo principal es una experiencia bastante positiva, debido a que se puede aplicar el conocimiento visto en clase, así como también divulgar de cierta manera el comportamiento físico de lo que vivimos día con día. Además, es necesario mencionar que el desarrollo de videojuegos con físicas independientes al motor puede ser una manera calmada de entrar en el ámbito de desarrollo de videojuegos, esto es debido a que los motores de física tienden a ser muy robustos y complejos, volviéndose complicados de entender y operar. Por lo que usar directamente las ecuaciones físicas puede resultar más sencillo viniendo de un esquema de trabajo tradicional respecto a la programación.

## Referencias:


https://docs.godotengine.org

Moebs, W. (2016, August 3). *4.3 Projectile Motion – University Physics Volume 1*. Pressbooks. https://opentextbc.ca/universityphysicsv1openstax/chapter/4-3-projectile-motion

KidsCanCode. (n.d.). *Using KinematicBody2D :: Godot Recipes*. Retrieved May 28, 2021, from https://kidscancode.org/godot_recipes/physics/godot3_kinematic2d/

TexasGateway. (2020, February 25). *5.3 Projectile Motion | Texas Gateway*. https://www.texasgateway.org/resource/53-projectile-motion

Javatpoint. (2019, March 12). *Godot Tutorial - Javatpoint*. Www.Javatpoint.Com. https://www.javatpoint.com/godot