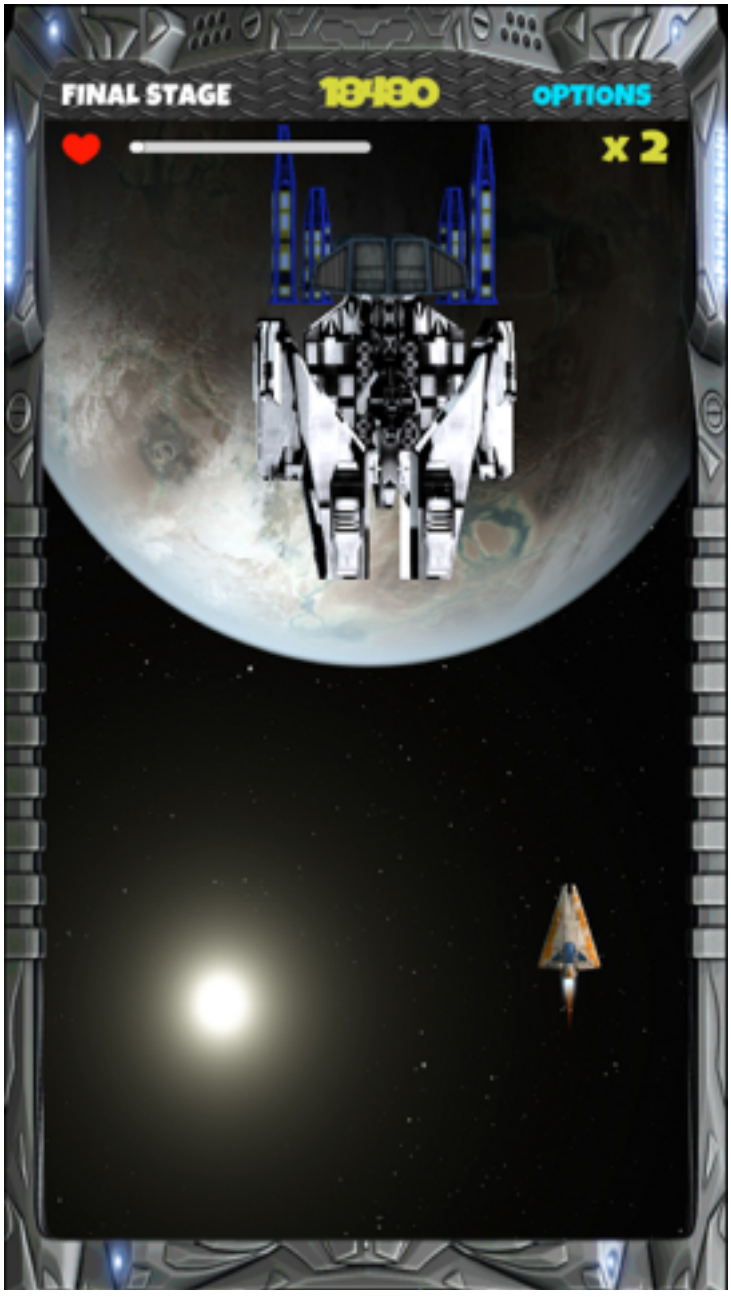
UNIVERSITAT POLITÈCNICA DE VALÈNCIA



Proyecto de Unity para dispositivos Android:

Galaxy Wars

Autor:

Pagès Rodríguez, Eduard

Director:

Linares Jordi

TESINA PARA LA OBTENCIÓN DEL TÍTULO DE:

Máster en Desarrollo de Aplicaciones sobre Dispositivos Móviles

Contenido

Introducción

Descripción del problema

Desarrollar videojuegos para Android mediante el IDE AndroidStudio y el empleo de Java es una tarea demasiado laboriosa como para ser rentable.

Usando una plataforma para el desarrollo de videojuegos como Unity, no solo se pueden obtener los mismo resultados, sino obtener una calidad y comportamientos físicos mejores en un menor tiempo.

Objetivos

Aprender a utilizar las herramientas para la creación de videojuegos en dispositivos móviles utilizando el motor *Unity3D*.

Motivación

La industria de los videojuegos es un sector económico que no para de crecer. Se espera que este sector siga creciendo en los próximos años.*(1)*

Siendo un sector en crecimiento, me resulta interesante formarme profesionalmente para poder entrar en este sector.

Tecnologías utilizadas

En el proyecto se ha empleado el sistema de Audio Mixer incorporado en *Unity5*. Este sistema permite controlar los flujos de las fuentes de audio.

Audio Mixer permite cambiar los niveles del volumen, agrupar fuentes de audio y procesar sonidos con efectos.

Se ha empleado para la persistencia la serialización y deserialización de estructuras de datos en estructuras de texto y viceversa en formato binario a través de la clase *BinaryFormatter* de la plataforma *.NET,* creando así un fichero de almacenamiento de las puntuaciones obtenidas.

Para poder mostrar las puntuaciones almacenadas se hacen a partir de elementos de Interfaz de Usuario como son las listas y el scroll.

También se ha empleado un canvas con un campo de texto al estilo *Dialog* del sistema operativo Android para la introducción del campo nombre para así poder identificar y guardar las puntuaciones obtenidas por el usuario.

En algunos casos se ha empleado imágenes *Normal Maps* para simular superficies de objetos con relieves sin que por eso se emplee superficies tridimensionales, con el fin de mostrar superficies más reales y con menor coste computacional.

Estas imágenes son la representación de otra imagen pero a través de las componentes RGB que simulan los 3 ejes X, Y y Z. Los *Normal Maps* representan la superficie de otra imagen y juntas se emplean para crear Materials.

Aunque los modelos importados para el proyecto usan este tipo de imágenes, en el proyecto se han creado de propias para la representación de las superficies *Background* de algunos niveles.

Se ha empleado el servicio gratuito de 'Normal Maps Online' para generar estas imágenes. (2)

Arquitectura de la aplicación

Elementos empleados en la aplicación:

MODELS

- Shaders (materials).

- Objetos en formato '*.fbx*' .

- Descripción: Modelos tridimensionales importados del '*Asset store*' tanto para las naves como para los modelos de Asteroides. Estos modelos definen como Unity va a importar estos modelos. Se emplean para generar GameObjects a partir de estos modelos. Se ha empleado el '*Scale Factor*' para crear los diferentes tamaños de asteroides y naves BigBoss.

MATERIALS

- Textures (png).

- Sprites (2D y UI).

- Normal Maps.

- Descripción: Definen como debe ser la piel de los GameObjects y elementos de UI. Muchos materiales han sido importados junto a los modelos importados del '*Asset Store*'. Otros materiales se han creado para generar naves de otros colores y pintar el '*Background*' de los niveles usando '*Normal Maps*'

AUDIO MIXER

- 2 groups: Music y Sound Effects.

- 2 snapshots: Paused y Unpaused.

- Descripción: Tabla de mezclas que permite la configuración de grupos de audio, efectos de sonido y manipulación durante el *runtime*.

Se ha empleado para manipular la relación entre los efectos de audio i la música usando *Lowpass filter* y *DuckVolume*. También se emplea para manipular el audio según si el juego se encuentra en pausa o no y para permitir al usuario modificar la relación de audio entre música y efectos de sonido.

AUDIO SOURCE

- Música.

- Efectos para las explosiones.

- Efectos para los disparos.

- Importados desde *Asset Store* dentro del paquete Space_shotter.

PERSISTENCIA DE LAS PUNTUACIONES

- Método: BinaryFormatter.

- Script: ScoreManager (static).

- Usado en: GameManager, ScoreScreen.

- Descripción: Se guardan los datos en un fichero *.dat* dentro del espacio reservado de la aplicación dentro de el Sistema Operativo Android (/data/data/nombre_del_paquete/files).

Este sistema se emplea para almacenar las puntuaciones obtenidas.

PREFERENCIAS DE USUARIO

- Método: PlayerPrefs.

- Script: PauseMenuManager.

- Usado en: Menu y MenuGame (prefabs)

- Descripción:Se emplean las preferencias de usuario para definir el comportamiento de Audio Mixer y activar/desactivar el sensor de acelerómetro.

INPUTS

- GameOjects: InputNameCanvas (InputField/keyboard), MenuCanvas y Player.

- Sensores: Acelerómetro y TouchScreen.

- Scripts: InputNameCanvas, PlayerController.

GAMEOBJECT

- Componentes presentes según el tipo de GameObject:
  - Transform o RectTransform (Posición de cada elemento).
  - Rigidbody (Para cálculos de la física).
  - SphereCollider, BoxCollider, CapsuleCollider (Solidez del objeto).
  - Scripts (Para el comportamiento).
  - AudioSource (recursos de audio).
  - AudioListener (Oídos para un entorno 3d)
  - Partycle System (para explosiones, efectos de estrellas y propulsión)
  - Mesh Filter, Mesh Renderer y Shaders (para la representación visual del gameobject).

SCRIPTS

- Lenguaje empleado: C#.
- Cualquier comportamiento que debe tener un gameObject.
- Clases static para el uso compartido de datos entre objetos y escenas

SCENES

- Cada una de las pantallas que componen la aplicación

Escenas y GameObjects que las componen:

MainMenu

- GameObjects:
  - MainCamera.
  - Directional Light.
  - CanvasBotones
  - MenuMain (Dialog activo al pulsar 'Settings').
  - InputNameCanvas (Dialog activo al pulsar 'Start').
  - EventSystem (Autogenerado para elementos de UI).
  - BackgroundMusic (Contiene el AudioSource de la música).
- Descripción: Escena de entrada a la aplicación. Aqui se puede empezar la partida, introducir el usuario, configurar las preferencias de usuario, mirar las puntuaciones y salir de la aplicación

Main y Level_02 al Level_09

- GameObjects:
  - MainCamera.
  - Lighting : 3 tipos de luces (Main Light, Fill light, Rim Light).
  - GameController (Manejador del nivel).
  - MenuGame (Dialog activo al pulsar 'Options').
  - Player (Jugador).
  - StartField (Sistema de partículas en algunos niveles).
  - Quad (Background del nivel, puede ser móvil o estático).
  - Boundary (Límites de la escena).
  - Canvas (Elementos de UI).
  - EventSystem (Autogenerado para elementos de UI).
  - BackgroundMusic (Contiene el AudioSource de la música).
- Descripción: Cada uno de los niveles que componen el juego. Cada dos niveles termina con un Jefe Final de nivel (BigBoss)

Level_10 y ScoreScene

- MainCamera.

- Directional Light.

- Canvas (Elementos de UI).

- EventSystem (Autogenerado para elementos de UI).

- BackgroundMusic (Contiene el AudioSource de la música).

- Descripcion: Level_10 es la pantalla que aparece al terminar el juego y muestra la puntuación obtenida. ScoreScene muestra una lista con las 15 mejores puntuaciones.

Listado de GameObjects empleados:

A continuación se detallan los GameObjects más importantes empleados durante la ejecución de la aplicación. Se hace distinción entre GameObject, como un elemento individual y diferente en cada escena, y Prefab, una copia exacta de un GameObject que se repite durante la ejecución del juego.

CONTROLADOR DE CADA NIVEL

- GameObject: GameController.
- Scripts: GameManager.
- Tag: GameController.
- Layer: Default.
- Scene: Main y Level_02 a Level_09.
- Componentes: Transform.
- Descripción: Este GameObject es el encargado de la mecánica de juego, pasar de nivel, mostrar o ocultar algún elemento de UI o ir instanciando oleadas de enemigos.

ELEMENTOS DE UI DURANTE LA PARTIDA (Prefab)

- Prefab: 'Canvas'.
- Scripts: SimpleTouchPad, SimpleTouchAreaButton.
- Scene: MainMenu, Main y niveles del 2 al 9.
- Tag: Untagged.
- Layer: UI.
- Childs: Barra Superior, HealthCanvas, Marco, GameOver Text, Fire Zone, Movement Zone, Restart Button, Quit Button, Ready Text, NextLevel Text.
- Descripción: En este canvas se compone de todos los elementos presentes durante la ejecución de los niveles: Marco de la pantalla, barra de vida, las zonas de touch de la pantalla, así como elementos que aparecerán durante el ciclo de vida como es un fundido a negro, textos que informan al usuario el inicio y fin de nivel, game over y los botones para mostrar las opciones, re-empezar el juego o salir a la pantalla principal de la aplicación.

BACKGROUND DE CADA NIVEL

- GameObject: 'Quad'.
- Scripts: Done_BGScroller.
- Scene: MainMenu, Main y niveles del 2 al 9.
- Tag: Untagged.
- Layer: Default.
- Descripción: El Quad es el background de cada nivel, y este se puede mover en el eje de la z para dar efecto de movimiento en el espacio.

AREA DE JUEGO (Prefab)

- Prefab: Boundary.
- Scripts: DestroyByBoundary.
- Tag: Boundary.
- Layer: Default.
- Scene: Main y Level_02 a Level_09.
- Componentes: Transform, BoxCollider.

Descripción: Este elemento es invisible y define el área de juego.Cualquier elemento que salda de los límites de esta área será destruido fuera de la pantalla.

BOTONES DEL MENÚ PRINCIPAL

- GameObject: CanvasBotones.
- Scripts: MainMenuBotones, PauseMenuManager.OptionsMenuCanvas().
- Scene: MainMenu.
- Tag: Untagged.
- Layer: UI.
- Opciones:
    - Start.
    - Score.
    - Settings.
    - Salir de la aplicación.

DIALOG PARA INTRODUCIR NOMBRE (Prefab)

- Prefab: InputNameCanvas.
- Scripts: InputNameCanvas, OutInputNameCanvas.
- Scrips Aux.: ScoreManager.
- Tag: Untagged.
- Layer: UI.
- Scene: MainMenu.

MÚSICA (Prefab)

- Prefab: 'BackgroundMusic'.
- Tag: Untagged.
- Layer: Default.
- Scene: Todas.
- Componente: AudioSource.

MENÚ DE OPCIONES (Prefab)

- Prefab: Menu y MenuGame.
- Scripts: PauseMenuManager, AudioMasterMixerLvls.
- Scene: MainMenu, Main y niveles del 2 al 9.
- Tag: Untagged.
- Layer: UI.
- Opciones:
    - Activar/desactivar audio.
    - Activar/desactivar acelerómetro (Excepto en MainMenu).
    - Nivel de volumen de la música.
    - Nivel de volumen de los efectos de sonido.
    - Salir de la partida (prefab MenuGame).
- Descripción: Las opciones mostradas serán almacenadas como preferencias de usuario a través de la clase static de UnityEngine.PlayerPrefs.

PLAYER (Prefab)

- Prefab: 'Player'.
- Scripts: PlayerController, PlayerWeaponController, PlayerHealthManager, Mover.
- Tag: Player.
- Layer: Default.
- Scene: Main y Level_02 a Level_09.
- Componentes: Transform, Rigidbody, MeshFilter 'Vehicle_player_ship', MeshRenderer ,materials('vehicle_player_ship_metal_mat, vehicle_player_ship_glass_mat).

LASER JUGADOR (Prefab)

- Prefab: 'Bolt'.
- Parent: runtime 'Player'
- Scripts: Mover.
- Tag: Bolt.
- Layer: Default.

LASER ENEMIGOS (Prefab**)**

- Prefab: 'Enemy_Bolt', 'Enemy_BoltX2', 'Enemy_BoltX3'
- Parent: runtime 'Enemies'.
- Scripts: Mover, DestroyByContact, EnemyHealtManager.
- Tag: Enemy_bolt.
- Layer: Default.
- Descripción: Distintos laser con diferentes velocidades y daño causado.

SHOTSPAWN (Prefab**)**

- GameObject: 'Shotspawn'
- Parent: Casi en todos los vehículos.
- Tag: Untagged.
- Layer: Default.
- Descripción: punto de origen para los disparos. Su ubicación es local respecto al objecto que lo contiene.

NAVES ENEMIGAS (Prefab**)**

- Prefab: múltiples naves enemigas.
- Scripts: Mover, DestroyByContact,EnemyHealtManager, WeaponController, EvasiveManeuver, ManiobraEvasiva, ManiobraEvasiva2, ManiobraEvasivaBigBoss.
- Tag: Enemy, (BigBoss para las nave Jefe).
- Layer: Default.
- Scene: Main y Level_02 a Level_09 (runtime).
- Componentes: Transform, Rigidbody, MeshFilter 'Vehicle_player_ship', MeshRenderer ,materials(multiples según vehículo), sistema de partículas de propulsión y explosión.

Descripción:Existen 9 tipos diferentes de naves enemigas y 5 tipos de BigBoss (4 de ellos son iguales que las naves pequeñas excepto alguna customización en sistema de partículas). Entre estas naves hay diferentes tipos de movimientos y comportamientos, así como agrupaciones de estas naves para formar 3 tipos básicos de escuadrones.

En total hay 34 prefabs de naves pequeñas y 5 de naves BigBoss.

Muchas de estas naves cambiarán su movimiento complejo a uno más simple si entran en contacto con otras naves enemigas, de esta forma se evitan colisiones no deseadas de la física.

ASTEROIDES (Prefab)

- Prefab: Asteroid, Asteroid2, Asteroid3, Asteroide1_big, Asteroide2_big, Asteroide3_big.
- Scripts: Mover, DestroyByContact,EnemyHealtManager, RandomRotator
- Tag: Enemy.
- Layer: Default.
- Scene: Main y Level_02 a Level_09 (runtime).
- Componentes: Transform, Rigidbody, MeshFilter 'Prop_asteroid_0x', MeshRenderer ,materials(prop_asteroid_0x_mat), Collider.

Descripción: Hay 3 asteroides pequeños y estos mismo con el triple de tamaño y health.

Propulsión (Prefab)

- Prefab: 'engines_player', 'enemy_engines'.
- Tag: Untagged.
- Layer: Default.
- Parent: 'Player' y algunos vehículos.
- Descripción: Sistema de partículas para la propulsión de algunas naves.

SISTEMA DE PARTÍCULAS DE ESTRELLAS(Prefab)

- Prefab: 'StarField'
- Tag: Untagged.
- Layer: Default.
- Scene: En algunos niveles.
- Descripción: Contiene 2 sistemas de partículas, una para las estrellas

Listado de Scripts empleados:

AudioMasterMixerLvls

- Variable:
  - public AudioMixer masterMixer;
- Funciones:
  - public void SetSfxLvl(float sfxLvl)
  - public void SetMusicLvl(float musicLvl)
  - public void ActivateSounds(bool on)
- Prefab: MenuGame y Menu
- Scene: Todas las escenas.
- Descripción: Este script se encarga de manejar los niveles de audio a través de AudioMixer.

CongratulationsScript

- Variable:
  - public Text infoText;
- Funciones:
  - void Start()
  - void irAScoreScreen()
- GameObject: Canvas.
- Scene: Level_10
- Descripción: Este script mostrará un text informando que se ha terminado el juego con la puntuación obtenida.

DestroyByBoundary

- Variable:
- Funciones:
  - void OnTriggerExit (Collider other)
- Prefab: Boundary.
- Scene: Main y del Level_02 al Level_09.
- Descripción: Destruirá cualquier objeto que supere los límites del collider asignado a Boundary. Estos límites se encuentran fuera de la vista del juego.

DestroyByContact

- Variables:
  - public GameObject explosion;
  - public int scoreValue;
  - private GameManager gameController;
  - GameObject player;
  - PlayerHealthManager playerHealth;
  - EnemyHealtManager enemyHealth;
- Funciones:
  - void Start()
  - void OnTriggerEnter()
- Prefabs: Naves Enemigas, Laser Enemigo y asteroides.
- Scene: runtime de los niveles Main y Level_02 al Level_09
- Descripción: Este script detectará mediante '*tags*' que objetos entran en contacto. Aquí se determina si se quita vida al 'Player' y a los enemigos. Si uno de los objetos es un laser ('*Bolt*' o '*Enemy_Bolt*'), este será destruido después de la colisión.

DestroyByTime

- Variables:
  - public float lifeTime;
- Funciones:
  - void Start()
- Prefabs: explosion_enemy, explosion_player, explosion_asteroid.
- Scene: runtime de los niveles Main y Level_02 al Level_09
- Descripción: Destruirá el sistema de partículas 2 segundos más tarde de ser instanciado el objeto.

DrawItem

- Variables:
- Funciones:
  - void OnDrawGizmos()
- Prefabs: Maniobra1,Maniobra2, Maniobra3 loop, Maniobra4 loop y en algunos items de algunas naves.
- Scene: runtime de los niveles Main y Level_02 al Level_09
- Descripción: Se dibuja una esfera amarilla segun las coordenadas del gameObject que contiene. Sirve para marcar el path de movimiento de algunas naves.

EnemyHealtManager

- Variables:
  - public int startingHealth;
  - private int currentHealth;
  - private GameManager gameManager;
- Funciones:
  - void Awake()
  - public void TakeDamage (int amount)
  - void Death ()
- Prefabs: Todas las naves enemigas y asteroides.
- Scene: runtime de los niveles Main y Level_02 al Level_09
- Descripción: Maneja la vida restante de todo lo que puede ser destruido por el Player. Si el objeto es destruido sumará puntos en el GameManager.

EvasiveManeuver

- Variables:
  - public Boundary boundary;
  - public float tilt;
  - public float dodge;
  - public float smoothing;
  - public Vector2 startWait;
  - public Vector2 maneuverTime;
  - public Vector2 maneuverWait;
  - public float speed;
  - private float currentSpeed;
  - private float targetManeuver;
  - private Rigidbody rb;
- Funciones:
  - void Awake()
  - void Start ()
  - IEnumerator Evade ()
  - void OnTriggerEnter (Collider other)
  - void FixedUpdate ()
- Prefabs: Algunas naves enemigas.
- Scene: runtime de los niveles Main y Level_02 al Level_09
- Descripción: Fija la posición inicial de la nave y el movimiento en el eje X de forma aleatoria. Si la nave en contacto con otro objeto enemigo invierte el desplazamiento en el eje x.

ManiobraEvasiva

- Variables:
  - public Done_Boundary boundary;
  - public float tilt;
  - public float dodge;
  - public float smoothing;
  - public Vector2 startWait;
  - public Vector2 maneuverTime;
  - public Vector2 maneuverWait;
  - public float speed;
  - private float currentSpeed;
  - private float targetManeuver;
  - private Rigidbody rb;
  - public float tiempo;
- Funciones:
  - void Awake()
  - void Start ()
  - IEnumerator Evade ()
  - void OnTriggerEnter (Collider other)
  - void FixedUpdate ()
- Prefabs: Algunas naves enemigas.
- Scene: runtime de los niveles Main y Level_02 al Level_09
- Descripción: Similar al script de EvasiveManeuver. Si la nave en contacto con otro objeto enemigo desactiva este script y activa EvasiveManeuver.

Mover

- Variables:
  - private Rigidbody rb;
  - public float speed;
- Funciones:
  - void Start ()
  - void OnTriggerEnter (Collider other)
- Prefabs: Algunas naves enemigas y los asteroides.
- Scene: runtime de los niveles Main y Level_02 al Level_09
- Descripción: Las naves enemigas y asteroides tienen el movimiento más simple, desplazarse a través del eje z (de arriba abajo).

ManiobraEvasiva2

- Variables:
  - public Boundary boundary;
  - public float tilt;
  - private Rigidbody rb;
  - public float tiempo;
  - public string iTweenLoop = “none";
  - public Transform[] paths = new Transform[4];
  - private Transform[] pathsItem = new Transform[3];
- Funciones:
  - void Awake()
  - void Start ()
  - void OnDrawGizmos()
  - void ObtenPosicionInicial()
  - void OnTriggerEnter (Collider other)
  - void FixedUpdate ()
- Prefabs: Algunas naves enemigas.
- Scene: runtime de los niveles Main y Level_02 al Level_09
- Descripción: Las naves enemigas se mueven a través de un path de posiciones mediante el script de iTween. Este path es dibujado también. Si la nave entra en contacto con otra nave enemiga, se desactiva este script y se activa EvasiveManeuver.

ManiobraEvasivaBigBoss

- Variables:
  - public Boundary boundary;
  - public float tilt;
  - private Rigidbody rb;
  - public float tiempo;
  - public string iTweenLoop = “pingPong";
  - public Transform[] paths = new Transform[8];
- Funciones:
  - void Awake()
  - IEnumerator Start ()
  - void OnDrawGizmos()
  - void FixedUpdate ()
- Prefabs: BigBoss1, BigBoss2, BigBoss3, BigBoss4, BigBoss5.
- Scene: runtime de los niveles Level_02, Level_04, Level_06, Level_08 y Level_09.
- Descripción: Las naves enemigas se mueven a través de un path de posiciones mediante el script de iTween. Este path es dibujado también.

GameManager

- Variables:
  - public GameObject[] naves, meteors, escuadrones, navesLoop;
  - public GameObject bigBoss, player;
  - public Vector3 spawnValues;
  - public int numAmenazasXOleada, numAteroidesXOleada;
  - public Text scoreText,gameOverText,readyText,lvlTxt, nextLvlText;
  - public Image fadeBlack;
  - public GameObject restartButton, quitButton;
  - public float startWait, spawnWaitMax, spawnWaitMin, waveWait;
  - public string lvl, nextLvl;
  - public Text currentLives;
  - Quaternion spawnRotation;
- Funciones:
  - void Awake()
  - void Start ()
  - IEnumerator SpawnWaves ()
  - void AddScore()
  - void UpdateScore()
  - void UpdateParametros()
  - void GameOver()
  - void RestartGame()
  - void Quit()
- GameObject: GameController.
- Scene: runtime de los niveles Main y Level_02 al Level_09
- Descripción: Este script maneja la dinámica de cada nivel. Desde aquí se accede a la clase ScoreManager y ManejadorStatic para actualizar el estado de las puntuaciones, guardarlas si es necesario y llevar un control de las vidas del jugador. Este script a más controla las oledas de enemigos (y asteroides) y los va instanciando cada cierto tiempo dentro de un rango de valores para evitar la monotonía de la aparición de amenaza… Cuando empieza el nivel y termina se hace un fundido a negro y se hace visible unos Text informativos que surgen con animaciones de escala iTween. Si en el editor marcamos que hay 'BigBoss', en la última oleada aparece el jefe final de nivel. Tanto si no hay o es vencido, se pasa al siguiente nivel y se almacena la puntuación. Si el jugador es destruido se le van restando vidas hasta que llega a cero. En este momento aparecen el Text de 'Game Over!' y los botones de 'Restart' y 'Quit'. Al pulsar 'Restart' se empieza el juego de nuevo. En el editor se puede elegir que tipo de amenazas afrontará el jugador.

InputNameCanvas

- Variables:
  - Canvas canvas;
  - public InputField nombreInput;
- Funciones:
  - void Start ()
  - void AceptarNombre()
  - void ShowCanvas ()
- Prefab: InputNameCanvas.
- Scene: MainMenu.
- Descripción: Muestra un dialogo con una caja de texto para poder introducir el nombre y almacenar las puntuaciones y empezar la partida.

MainMenuBotones

- Variables:
  - InputNameCanvas inputfield
  - public GameObject inputCanvas;
- Funciones:
  - void Start ()
  - public StartButton()
  - public QuitButton ()
  - public ScoresButton ()
- GameObject: CanvasBotones.
- Scene: MainMenu.
- Descripción: Maneja los botones de la escena para re-dirigir al usuario a otras escenas.

ManejadorStatic

- Variables:
  - **public const int** DefaultLives.
  - private static int livesPlayer = DefaultLives;
  - public static int LivesPlayer
- Descripción: Es una clase static que almacena las vidas del jugador.

ManejadorStatic

- Variables:
  - public InputNameCanvas inputfield;
- Funciones:
  - public void OnPointerDown(PointerEventData data)
- Escenas: MainMenu.
- GameObject: OutputCanvas
- Descripción: Este gameObject se encuentra dentro de InputNameCanvas y sirve para recoger las pulsaciones del usuario fuera del marco de un dialogo para cancelarlo y desaparezca.

PauseMenuManager

- Variables:
  - Slider musicSlider;
  - Slider effectsSlider;
  - public AudioMixerSnapshot paused;
  - public AudioMixerSnapshot unpaused;
  - Toggle audioToggle;
  - Toggle accelerometerToggle;
  - float DefaultVolumeLevel;
  - float DefaultEffectsLevel;
- Funciones:
  - void Start ()
  - public OptionsMenuCanvas()
  - public Pause()
  - public Lowpass()
  - void CargarPrefs()
  - void Guardar()
- Prefab: Canvas y MenuCanvas.
- Scene: MainMenu, Main y los niveles desde el Level_02 hasta el Level_09.
- Descripción: Maneja la aparición del dialogo de menu, se guardan las preferencias de usuario, los volumenes de audio y la transición entre estados de paused/unpaused.

ManejadorStatic

- Variables:
  - public float tumble;
  - private Rigidbody rigidBody;
- Funciones:
  - void Start ()
- Descripción: Rotación aleatoria de los asteroides.

PlayController

- Variables:
  - public SimpleTouchPad touchPad;
  - public float speed;
  - public float tilt;
  - public Boundary boundary;
  - public bool UseAccelerometer;
  - private Vector3 accelerationraw;
  - private Vector3 acceleration;
  - private Vector3 movement;
  - private Vector2 direction;
  - private Rigidbody rigidBody;
  - private Quaternion calibrationQuaternion;
- Funciones:
  - void Start ()
  - void FixedUpdate()
- Prefab: Player.
- Scene: Main y los niveles desde el Level_02 hasta el Level_09.
- Descripción: Maneja el movimiento de la nave Player, este puede ser mediante accelerómetro o a través de pulsaciones en la pantalla.

PlayWeaponController

- Variables:
  - public SimpleTouchAreaButton fireButton;
  - public float fireRate;
  - public GameObject shot;
  - public Transform shotSpawn;
  - private float nextFire;
  - private AudioSource audioDisparo;
- Funciones:
  - void Start ()
  - void Update()
- Prefab: Player.
- Scene: Main y los niveles desde el Level_02 hasta el Level_09.
- Descripción: Maneja la creación de intancias de Bolt y la activación de efectos sonoros a través de pulsaciones en la pantalla.

PlayHealthManager

- Variables:
  - public int startingHealth;
  - public int currentHealth;
  - public Slider healthSlider;
  - public GameManager gameController;
  - public GameObject playerExplosion;
  - private Renderer render;
  - private MeshCollider col;
  - private GameObject engines_player;
  - private Vector3 posicionInicial;
  - private PlayerWeaponController weaponPlayer;
  - bool isDead;
  - public bool damaged;
- Funciones:
  - void Awake ()
  - void Blink ()
  - public void TakeDamage()
  - void Death ()
  - void Reiniciar ()
- Prefab: Player.
- Scene: Main y los niveles desde el Level_02 hasta el Level_09.
- Descripción: Maneja la vida restante del Player y como mostrarla la vida a traves del Slider. Cuando la nave es golpeada muestra un efecto de parpadeo. Si la Nave es destruida se instancia una explosión y si aún le quedan vidas al jugador, reinicia la nave en su posición inicial.

PlayWeaponController

- Variables:
  - public float fireRate;
  - public GameObject shot;
  - public Transform shotSpawn;
  - public float delay;
- Funciones:
  - void Start ()
  - void Fire()
- Prefab: Naves enemigas.
- Scene: Runtime en Main y los niveles desde el Level_02 hasta el Level_09.
- Descripción: Maneja la creación de intancias de Enemy_Bolt y la activación de efectos sonoros por parte las naves enemigas.

static ScoreManager

- Variables:
  - public static int limitScores;
  - private static int puntuacionMaxima;
  - public static int PuntuacionMaxima;
  -public const string nombreDefault;
  - public static string Nombre;
  - private static string rutaArchivo;
- Funciones:
  - public static void Guardar()
  - public static List<Puntuaciones> Cargar()
  - static List<Puntuaciones> CrearDefaultPuntuaciones()
- Clase Interna:
  - public class Puntuaciones : IComparable<Puntuaciones>
- Descripción: Maneja las puntuaciones entre pantallas y guarda y carga las puntuaciones obtenidas. Se crea un archivo en el almacenamiento interno del dispositivo, dentro del espacio reservado dentro de la aplicación.

ScoreScreen

- Variables:
  - public GameObject itemPrefab;
  - public int itemCount;
  - public int columnCount;
- Funciones:
  - void Awake()
  - public void irMenu()
- Escena: ScoreScreen.
- GameObject: Scrollable.
- Descripción: Instancia las puntuaciones almacenadas y manejadas por ScoreManager.

SimpleTouchAreaButton

- Variables:
  - private bool touched;
  - private int pointerID;
  - private bool canFire;
- Funciones:
  - void Awake ()
  - public void OnPinterDown()
  - public void OnPinterUp()
  - public bool CanFire()
- Prefab: Player.
- Scene: Runtime en Main y los niveles desde el Level_02 hasta el Level_09.
- Descripción: Maneja las pulsaciones de pantalla para que Player dispare.

SimpleTouchPad

- Variables:
  - public float smoothing;
  - private Vector2 origin;
  - private Vector2 direction;
  - private Vector2 smoothDirection;
  - private bool touched;
  - private int pointerID;
- Funciones:
  - void Awake ()
  - public void OnPinterDown()
  - public void OnPinterUp()
  - public void OnDrag()
  - public Vector2 GetDirection()
- Prefab: Player.
- Scene: Runtime en Main y los niveles desde el Level_02 hasta el Level_09.
- Descripción: Maneja las pulsaciones de pantalla para mover al Player.

Vistas y Diagrama de funcionamiento:

GameObject: GameController

Script : GameManager

Función:

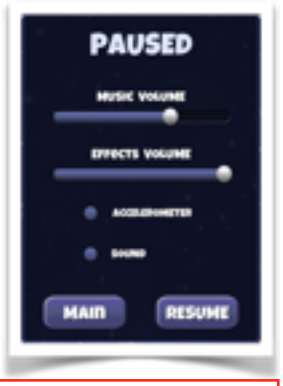
- Instancia amenazas
- Controla el UI
- Cambia de escena al completar nivel
- Decide cuando guardar puntuaciones

Prefab: Canvas/Marco/OptionMenu:

Script: PauseMenuManager

Función:

- Mostrar y ocultar dialogo.
- Transición de audio unpaused/paused
- Administrar MasterMixer
- Activar/desactivar acelerómetro
- Guardar preferencias de usuario



Prefab: Boundary

Script:

- DestroyByBoundary: Destruir cualquier gameobject que sobrepase los límites.

Prefab(Runtime): Cualquier Enemigo o Asteroide

Scripts:

- DestroyByContact: Recibir impactos.
- Mover(Diferentes scripts de mov.): Movimiento en ejes Z y X
- EnemyHealthManager: Maneja la vida
- WeaponController: Dispara según parámetros de 'fireRate' y 'delay'

Prefab: Player

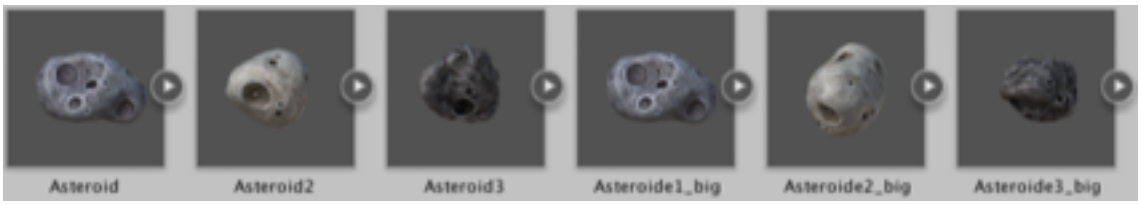
Scripts :

- PlayerController: Mover la nave. (Enlazado con Canvas/MovementZone)
- PlayerWeaponController: Disparar (Enlazado con Canvas/FireZone)
- PlayerHealthManager: Maneja la vida (Enlazado con GameController y Canvas/SliderHealth)
- Mover: Al pasar de nivel se deshabilitan los scripts anteriores y se activa este para desplazar la nave fuera de la pantalla.

Principales Elementos Visuales:

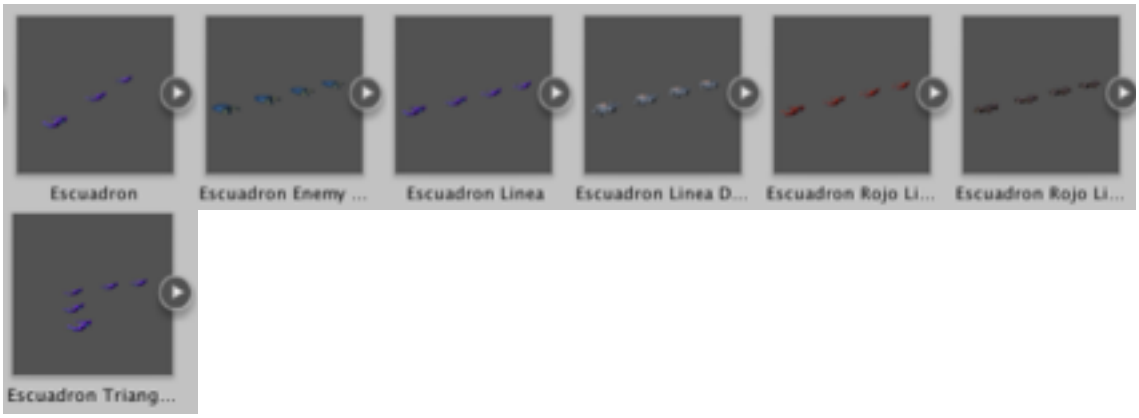
Player:

Asteroides:



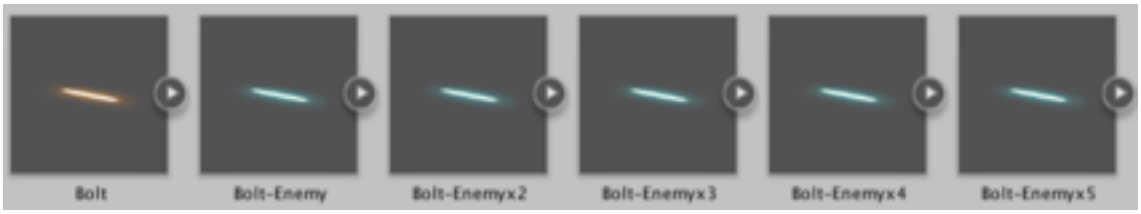
Naves Enemigas:

Escuadrones:



BigBoss:

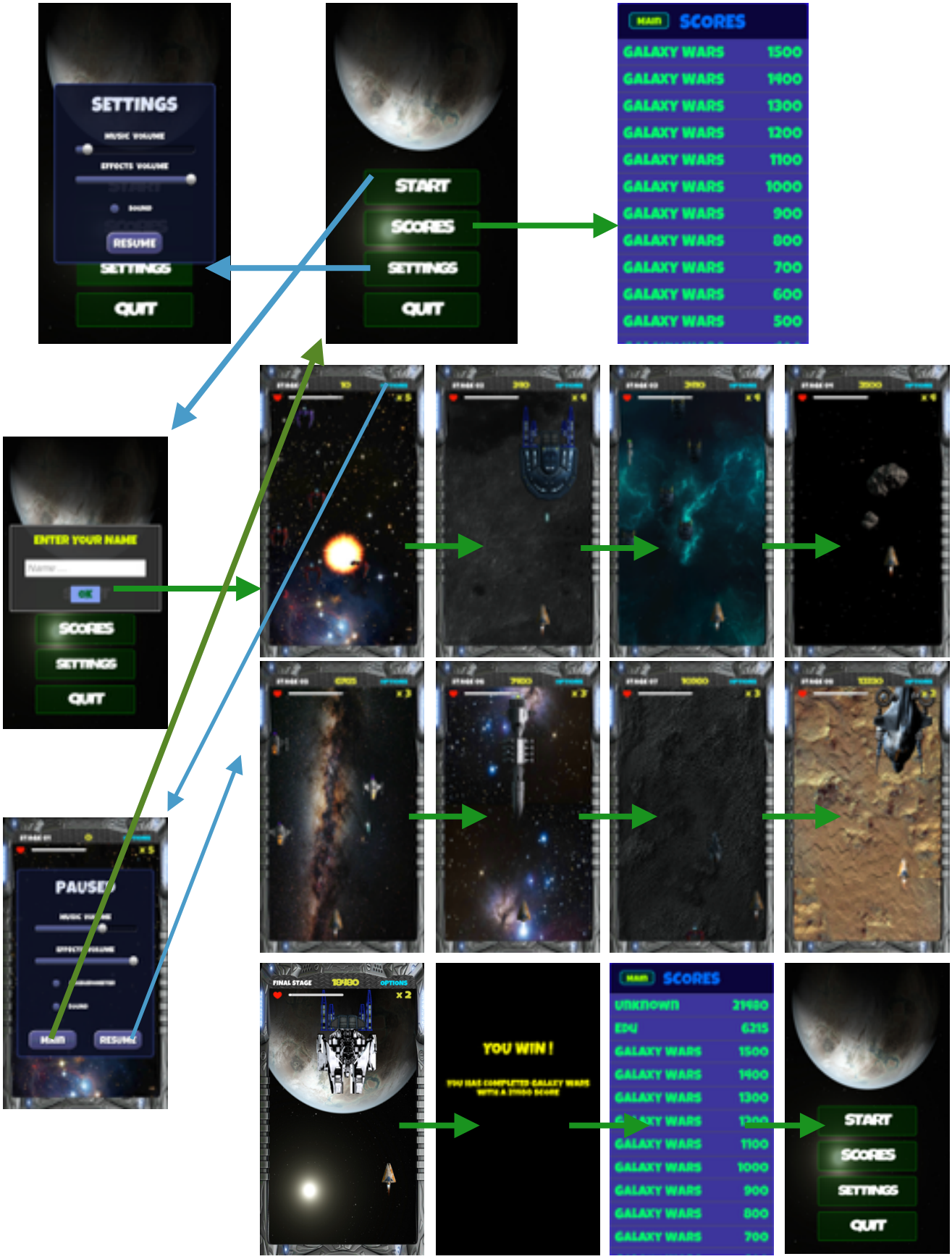
Proyectiles:



Canvas:

Navegación entre Escenas:

Navegación entre escenas Aparece diálogo



Justificación de las principales decisiones de diseño:

MasterMixer:

He planteado el uso de MasterMixer para poder regular los clips de música y efectos de sonido para que cuando coincida la reproducción de muchos clip en un mismo momento destaquen los efectos de sonido por encima de la música. Esto se consigue creando dos canales de audio, uno para la música y el otro para los efectos, donde los efectos son desviados al canal de música a través de un efecto de sonido llamado 'duckvolime', marcando un umbral de frecuencia y si en este canal se supera el umbral, el canal de música es atenuado para permitir que se escuchen los efectos por encima de la música. Cuando las frecuencias bajan de este umbral, la música vuelve a recuperar su volumen.

A más a más, como el uso de MasterMixer permite hacer distinciones entre grupos, sirve para que el usuario pueda graduar la relación entre música y efectos de sonido.

MasterMixer permite también a través de 'snapshots' diferentes configuraciones para los mismo canales. Esto se ha empleado para crear los 'snapshot' de 'paused' y 'unpaused', que tiene el efecto que cuando se muestra el dialogo de configuración el juego entra en pausa y el MasterMixer baja el umbral de volumen. Una vez se reanuda el juego vuelve a la configuración hecha por el usuario.

BinaryFormatter:

Para guardar las puntuaciones se podrían haber empleado diferentes métodos. He decidido emplear este método ya que permite *serializar* una estructura de datos definida como la clase 'Puntuaciones' y crear un archivo de texto dentro del espacio de memoria reservado para la aplicación dentro del sistema operativo Android.

Me ha parecido más correcto este método al de emplear los PlayerPrefs para guardar las puntuaciones, ya que su uso mas correcto serían para las preferencias de usuario, y más sencillo de emplear que una base de datos como SQLite, tan solo para almacenar una simple lista de puntuaciones donde tampoco tendría ningún sentido hacer una 'query' de estos datos.

NormalMaps:

El empleo de los 'NormalMaps' permite a un objeto mostrar relieve sin que por ello implique un aumento de polígonos para representarlo, ahorrando un coste computacional elevado y dando un efecto más real de los materiales que lo usan.

De esta manera se ha usado para la materialización de los fondos en los niveles 'Level_02', 'Level_07' y 'Level_08', que representan superficies de planetas o lunas.

Para crear estos 'NormalMaps' se ha empleado para tal fin la web 'NormalMapOnline', que permite hacer un mapa normalizado de cualquier imagen de forma rápida sin tener que usar un modelador 3d.

Animaciones iTween:

Estas animaciones permiten mover objetos a partir de la componente 'Transform' y no mediante fuerzas. En algunos casos, las naves enemigas se mueven a partir de animaciones 'iTween' a traves de un 'path' de puntos en el espacio, permitiendo animaciones complejas y loops en algunos casos, como por ejemplo las animaciones de las naves 'BigBoss'. En otros casos se emplea 'Rigidbody' que esta ligado a la física de fuerzas para mover estas naves.

Cuando una nave es animada a partir de 'iTween', la componente 'Rigidbody' sigue presente, ya que esta componente permite un movimiento simulado de la nave cuando vira hacia los lados, pero está marcada como 'isKinematic', para permitir que este objeto sea desplazado a través de un 'path' con de la componente 'Transform'.

Sin embargo, cuando un objeto con 'Rigidbody' es marcado como 'isKinematic' y tiene un 'Collider' marcado como 'isTrigger', este objeto 'pierde' la fisíca de objeto sólido, y otros objetos pueden atravesarlo. Para evitar este efecto indeseado estas naves lanzarán un 'trigger' para informar que han entrado en contacto con otras naves. Cuando esto ocurra, estas naves desactivarán su animacion 'iTween', 'isKinematic' se desmarcará del 'RigidBody', y se activará el script de 'EvasiveManeuver', que proporciona un movimiento errático y la nave se volverá 'sólida'. Si la nave entra en contracto con otra, el motor físico actuará como resultado.

Boundary:

El 'Boundary' marca los límites bidimensionales del juego, aunque en el fondo es un objeto 3d, los gameObject que se desplazan no lo harán a través del eje y.

Cualquier 'gameObject' que sobrepase estos limites será destruido, ya que estos límites se encuentran fuera de pantalla y no tendría sentido conservar-los. Esto se consigue con el script 'DestroyByBoundary', con la función 'OnTriggerExit()' y marcando el 'Collider' del objeto como 'isTrigger'.

En algunos casos, como las naves enemigas tienen diferentes ratios de disparo, en pocos casos puede llegar esta nave en los límites del 'Boundary' y disparar. Como el origen del disparo está por delante de la nave, este punto podría superar los límites del 'Boundary' y el disparo instanciado ya se encontraría fuera de los límites y nunca se llamaría la función de 'OnTriggerExit', por que ya estaría fuera. Como resultado tendriamos un objeto 'Enemy_Bolt' que no seria destruido, y bloquearia la 'Corutine()' llamada 'SpawnWaves' del script 'GameManger' en un bucle infito, ya que esta corrutina hace comprobaciones de si aún quedan objetos etiquetados con el 'tag' 'Enemy' , siempre encontraría un 'Enemy' pero este nunca podrá ser destruido y la corrutina no avanzaría nunca.

Para solventar este problema se ha creado otro objeto 'Boundary' por detrás del primero haciendo la función de muro. Si un objeto consigue instanciar fuera del primer 'Boundary', viajará hasta encontrar el segundo y al atravesarlo será entonces destruido.

Modelo de datos

BinaryFormatter:

El modelo de datos empleado es una lista de objetos de la clase 'Puntuaciones' serializada a través de 'BinaryFormatter' y almacenada como archivo en el 'path' del sistema '/data/data/paquete_de_la_aplicacion/files/puntuaciones.dat'.

La clase 'Puntuaciones' implementa la interfaz 'IComparable<Puntuaciones>' que permite comparar elementos de una lista de puntuaciones a partir del campo 'puntuacion' y ordenarlos de mayor a menor puntuacion. Esta clase tiene el siguiene esquema:

```
[Serializable]
public class Puntuaciones : IComparable<Puntuaciones>{

        public int puntuacion;
        public string nombre;

        public Puntuaciones(string nombre, int puntuacion){
                this.puntuacion = puntuacion;
                this.nombre = nombre;
        }

        public int CompareTo(Puntuaciones otraPuntuacion){
                if (this.puntuacion > otraPuntuacion.puntuacion) {
                        return -1;
                }else if(this.puntuacion < otraPuntuacion.puntuacion){
                        return 1;
                }else
                        return 0;
        }
}
```

Conclusiones

El videojuego consta de 9 niveles que deben ser superados consecutivamente sin tener la posibilidad de guardar la partida y volver a retomarla después. Componiéndose de 9 niveles y con 5 vidas disponibles desde el inicio de la partida, no hace falta que esto sea así. Si en un futuro se plantea la posibilidad de ampliar el número de niveles si sería necesario, bien implementar una lógica para guardar el nivel actual, o si el número de niveles no es demasiado alto, implementar un sistema de recuperación de vida y aumentar el número de vidas a través de paquetes de 'salud' que deben ser recogidos y que aparecieran de forma aleatoria al destruir enemigos.

Como consideración personal, encuentro que se debe implementar diferentes armas, tanto para el 'Player' como para las naves enemigas, para así aumentar la diversidad del juego y hacerlo más variado de cara al usuario.

No obstante estas consideraciones, creo que he cumplido el objetivo marcado al emplear diferentes herramientas de Unity y poder crear un videojuego completo con aspecto tridimensional, aunque el juego sea básicamente en dos dimensiones, sus componentes si son tridimensionales. He podido emplear herramientas que no se han podido tocar durante el curso como es el empleo de la persistencia de datos usando 'BinaryFormatter', el uso de MasterMixer para manejar el audio (nueva carácterística de unity 5.x) y crear imágenes 'Normal Map' para generar relieves sin coste computacional.

Anexos

Listado de fuentes entregadas

El proyecto entregado consiste en un Zip cuyo contenido se lista a continuación:

- Archivo apk para arquitecturas Arm7.
- Archivo apk para arquitecturas x86.
- Memoria del proyecto.
- Imágenes del juego.
- enlace a presentación en youtube
- Assets del proyecto:
    - Prefabs.
    - Materials
    - Scenes
    - Audio
    - Scripts

- Textures (debe incorporarse dentro de la carpeta Assets dentro de un proyecto Unity)

Manual de usuario

PANTALLA TÁCTIL:

El uso de la pantalla táctil está definida como input por defecto. La mitad inferior izquierda se usa para el movimiento de la nave como si fuera la cruceta de un PAD. La mitad inferior derecha se pulsa cuando se quiere disparar.

ACELERÓMETRO:

Para activar el acelerómetro hay que pulsar el botón de opciones que aparece en la esquina superior derecha de cada nivel del juego. Hay que tener en cuenta que cuando se activa el acelerómetro se calibra en ese momento, así que la posición del dispositivo ha de ser la que se empleará mientras se juega. Para disparar se sigue usando la pantalla táctil.

Fuentes

(1). http://www.eae.es/news/2015/01/26/el-mercado-del-videojuego-en-espana-movio-763-millones-de-en-2014-con-un-crecimiento-del-31-respecto-al-2013

(2) http://cpetry.github.io/NormalMap-Online/