



**IES AUGUSTO GONZALEZ DE
LINARES**
DEPARTAMENTO DE INFORMATICA

**INVESTIGACIÓN SOBRE
DESARROLLO MOVIL**

**PROGRAMACIÓN MULTIMEDIA Y
DISPOSITIVOS MÓVILES**

**GRADO SUPERIOR DE DESARROLLO DE
APLICACIONES MULTIPLATAFORMA**

2023/2024

Díez de Paulino, Albano

Índice

1. RECURSO DE APRENDIZAJE 1 (RA1)	2
1.1. (CE-C) Se han identificado las tecnologías de desarrollo de aplicaciones para dispositivos móviles.	2
1.2. (CE-D) Se han instalado, configurado y utilizado entornos de trabajo para el desarrollo de aplicaciones para dispositivos móviles.	3
1.3.(CE-E) Se han identificado configuraciones que clasifican los dispositivos móviles en base a sus características.	4
1.4. (CE-G) Se ha analizado la estructura de aplicaciones existentes para dispositivos móviles identificando las clases utilizadas.	4
1.5. (CE-H) Se han realizado modificaciones sobre aplicaciones existentes.	5
1.6. (CE-I) Se han utilizado emuladores para comprobar el funcionamiento de las aplicaciones.	6
2. RECURSO DE APRENDIZAJE 2 (RA2)	8
2.1. (CE-A) Se ha generado la estructura de clases necesaria para la aplicación.	8
2.2.(CE-B) Se han analizado y utilizado las clases que modelan ventanas, menús, alertas y controles para el desarrollo de aplicaciones gráficas sencillas.	9
2.3. (CE-C) Se han utilizado las clases necesarias para la conexión y comunicación con dispositivos inalámbricos.	11
2.4.(CE-D) Se han utilizado las clases necesarias para el intercambio de mensajes de texto y multimedia.	11
2.5. (CE-E) Se han utilizado las clases necesarias para establecer conexiones y comunicaciones HTTP y HTTPS.	12
2.6. (CE-F) Se han utilizado las clases necesarias para establecer conexiones con almacenes de datos garantizando la persistencia.	13
2.7.(CE-G) Se han realizado pruebas de interacción usuario-aplicación para optimizar las aplicaciones desarrolladas a partir de emuladores.	15
2.8 (CE-H) Se han empaquetado y desplegado las aplicaciones desarrolladas en dispositivos móviles reales.	15
2.9. (CE-I) Se han documentado los procesos necesarios para el desarrollo de las aplicaciones.	17
Tabla de Ilustraciones.	19
Bibliografía	20

1. RECURSO DE APRENDIZAJE 1 (RA1)

1.1.(CE-C) Se han identificado las tecnologías de desarrollo de aplicaciones para dispositivos móviles.

Desde la salida del iPhone en junio del 2009, la sociedad ha aumentado el uso de dispositivos móviles, desde casi no tener un móvil por persona en un hogar hasta tener mas de uno por persona, además estos dispositivos han evolucionado y los fabricantes han ido añadiendo mas funcionales que tienen que ser desarrolladas. Para facilitar el desarrollo a los programadores se han creado kit de desarrollos (SDK) según el S.O que usan dichos dispositivos.

Actualmente el mercado de S.O en dispositivos móviles está dominado en primer lugar por Android (*Propietario Google*) y en segundo lugar IOS (*Propietario Apple*).

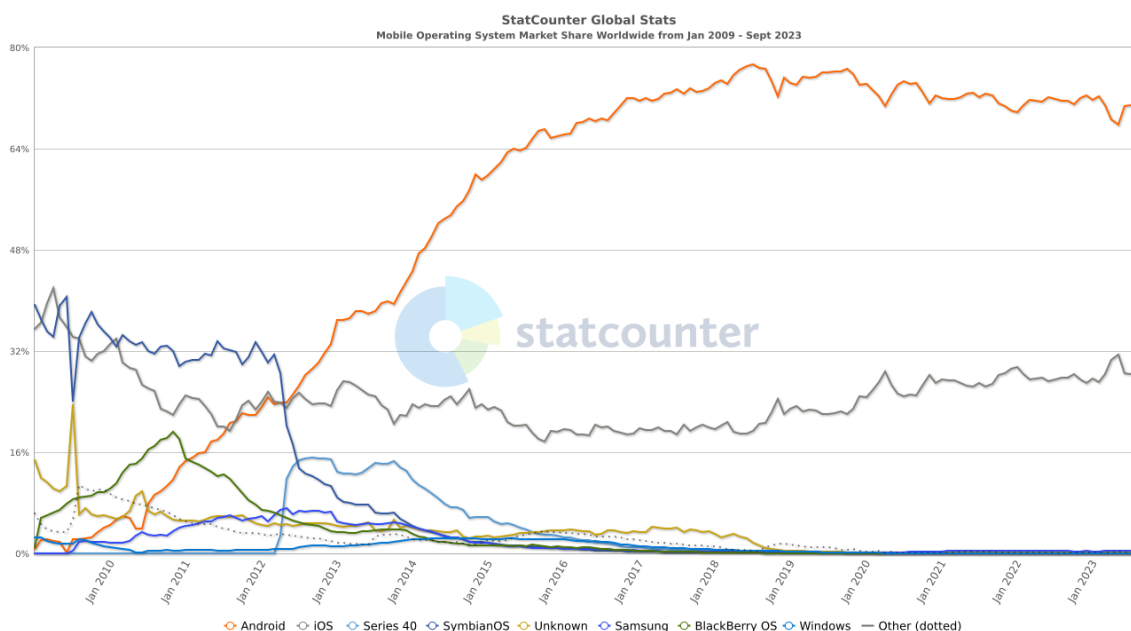



Tabla 1 – Estadísticas de uso S.O en dispositivos móviles

El desarrollo para estos dos sistemas está supeditado a los lenguajes de Kotlin (*Oficial de Google para Android desde 2019*), Java (*Lenguaje original para Android y en uso actualmente*) y Swift (*Lenguaje oficial de todos los S.O de Apple*), todos ellos disponen de SDK, el más famoso es el Java Development Kit (*JDK*), por su antigüedad y robustez.

Sobre estos SDK, se han creado frameworks que facilitan aun más el desarrollo de aplicaciones, como por ejemplo React Native o Flutter. Además, si se pretende desarrollar juegos existen los SDK especializados llamados GDK.

	Ciclo: Desarrollo de Aplicaciones Multiplataforma Modulo: Programación Multimedia y Dispositivos Móviles
	Título: Investigación sobre Desarrollo Móvil

1.2.(CE-D) Se han instalado, configurado y utilizado entornos de trabajo para el desarrollo de aplicaciones para dispositivos móviles.

Para manejar los diferentes JDK del apartado 1.1 se han desarrollado entornos de trabajo (*IDEs*) que permiten usar diferentes funcionalidades, por ejemplo, el compilado y ejecución automática del código, autocompletado de código o corrección de errores en tiempo de escritura, entre otras.

Los principales IDEs del mercado son Android Studio para Android y Xcode para iOS.

Además, se tiene la posibilidad de usar un motor de videojuegos comercial como Unity, Unreal Engine o Godot para desarrollar en dispositivos móviles añadiendo un módulo extra. *(Nota: Para instalar Unity, primero se instala el Unity Hub y luego la versión que se desea).*

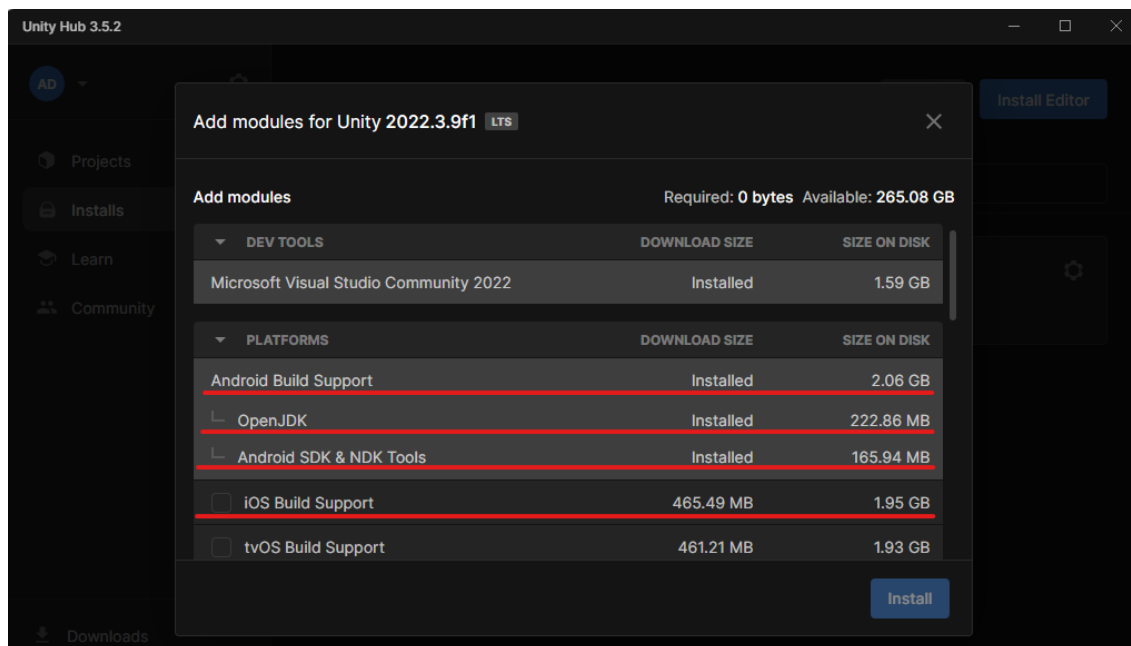


Ilustración 1 – Módulos Unity 2022.3.9f1

1.3.(CE-E) Se han identificado configuraciones que clasifican los dispositivos móviles en base a sus características.

Los dispositivos móviles se clasifican según su propósito y funcionalidad. Aquí te dejo una descripción de cada tipo:

- **Propósito general:** Incluye dispositivos como computadoras, tablets y teléfonos inteligentes avanzados. Se caracterizan por su capacidad para realizar una amplia gama de actividades y sintetizar funciones. Son el medio de trabajo para muchas personas, permitiendo realizar actividades que generan alta productividad.
- **Propósito de entretenimiento:** Este grupo incluye dispositivos diseñados para proporcionar diversión a las personas, por ejemplo, las consolas portátiles (*Nintendo Switch, Steam Deck o Asus ROG Ally*)

Además, los dispositivos móviles también pueden clasificarse según el usuario objetivo, como móviles para niños y móviles para adultos mayores. También se pueden distinguir por funcionalidades básicas y avanzadas, por ejemplo, los smartphones, tablets o wearable.

1.4.(CE-G) Se ha analizado la estructura de aplicaciones existentes para dispositivos móviles identificando las clases utilizadas.

Al empezar en el desarrollo para dispositivos móviles por el motor gráfico Unity, el primer paso es realizar el primer videojuego de la historia (*Pong*), ya que este es el más fácil de programar.

En Unity se puede realizar con dos clases básicas (*Scripts*) en el lenguaje C#, la primera clase se encarga de calcular el movimiento de la pelota, y la segunda del movimiento de las palas.

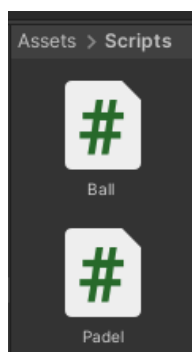


Ilustración 2 – Scripts Básicos Pong en Unity

Dichos *Scripts* son agregados en los *GameObjects* correspondientes, ya que si no son agregados nunca serán ejecutados.

1.5.(CE-H) Se han realizado modificaciones sobre aplicaciones existentes.

Al videojuego del Pong clásico se le pueden añadir varias mejoras en la legibilidad del código y funciones de videojuegos más nuevos, para ello realice lo siguiente.

Crear un *Main Menu* con las acciones de jugar una partida o salir de la aplicación, para ello se crea un nuevo *script* que controle.

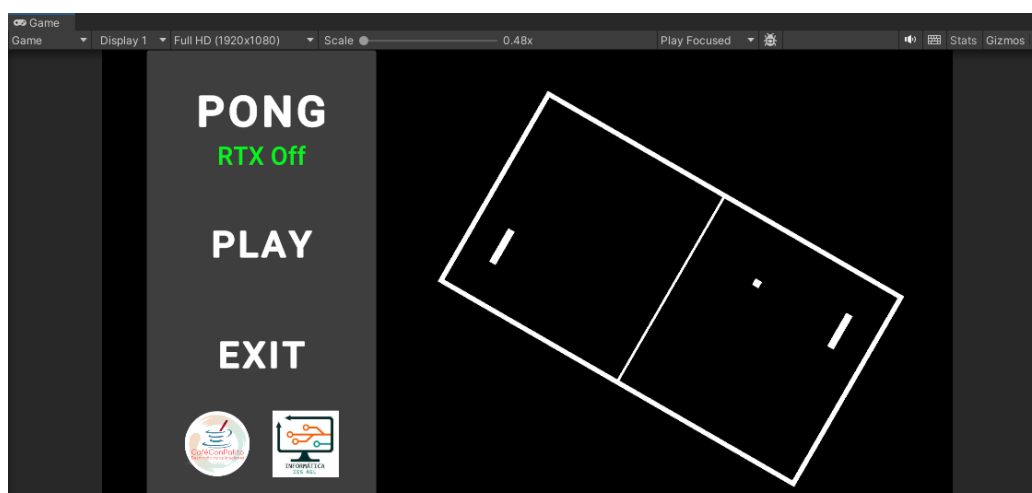


Ilustración 3 – Main Menu Pong

```

MainMenu.cs
Assembly-CSharp
1 using System.Collections;
2 using System.Collections.Generic;
3 using UnityEngine;
4 using UnityEngine.SceneManagement;
5
6 public class MainMenu : MonoBehaviour
7 {
8
9     void Update()
10     {
11         if (Input.GetKeyDown(KeyCode.Escape))
12         {
13             exit();
14         }
15     }
16
17     //Metodo para ser ejecutado cuando se de al boton play
18     //0 referencias
19     public void play()
20     {
21         SceneManager.LoadScene("Pong");
22     }
23
24     //Metodo para ser ejecutado cuando se de al boton exit
25     //1 referencia
26     public void exit()
27     {
28         Application.Quit();
29     }
30
31 }
    
```

Código 1 – Script Main Menu

Además, el control del juego le centralice en un script llamado GameManager.cs con el patrón de diseño singleton para que solo haya una instancia de esta clase durante el tiempo de ejecución y así eliminar posibles errores de simultaneidad.

```
// Instancia estática para ser accedida desde cualquier lugar
public static GameManager instance;

// Mensaje de Unity | 0 referencias
void Awake()
{
    // Comprueba si la instancia ya existe
    if (instance == null)
    {
        // Si no, establece la instancia a esta
        instance = this;
    }
    // Si la instancia ya existe y no es esta:
    else if (instance != this)
    {
        // Entonces destruye este objeto. Esto refuerza nuestro patrón Singleton, lo que significa que solo puede haber una instancia de un GameManager.
        Destroy(gameObject);
    }

    // Establece este para no ser destruido cuando se recargue la escena
    DontDestroyOnLoad(gameObject);
}
```

Código 2 – Patrón Singleton en Script GameManager.cs

Al centralizar el control del juego en una sola clase y escribir la línea `DontDestroyOnLoad()`; nos permite transferir información entre escenas del mismo proyecto sin tener mucha complicación o solo usar un Game Manager para todas las escenas.

1.6.(CE-I) Se han utilizado emuladores para comprobar el funcionamiento de las aplicaciones.

Para comprobar el correcto funcionamiento del juego en diferentes plataformas portátiles se han buscado emuladores para no tener que adquirir el hardware necesario o el kit de desarrollo de los fabricantes.

El más usado para emular el sistema Android es el BlueStacks y para Nintendo Switch es el Yuzu.

Además, hay emuladores para consolas portátiles retro como mGBA para emular la GameBoy Advance en dispositivos con Android o RetroArch que recopila la mayoría de los emuladores de consolas de Nintendo para PC.

Pero para ser más eficiente me he instalado el paquete *Device Manager* para Unity que me permite probar como seria mi juego en muchos dispositivos móviles sin necesitar hacer un “Build”. (Nota: Desde Unity 2021 ya viene integrado por defecto en las instalaciones y solo amplía el número de dispositivos que se pueden emular).

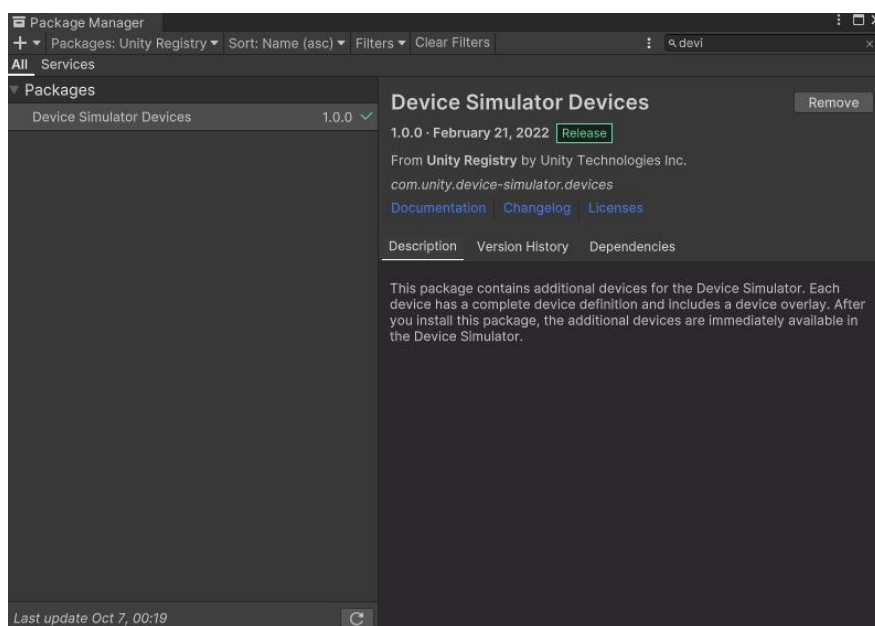


Ilustración 4 – Package Manager Unity 2022

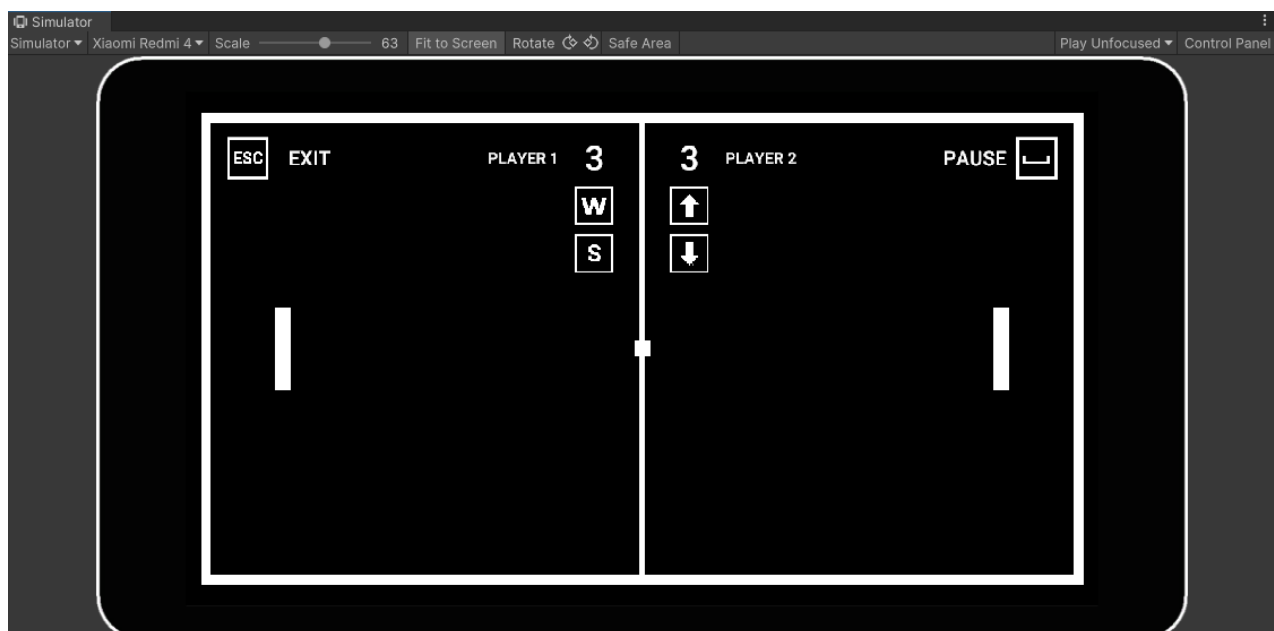


Ilustración 5 – Unity Simulando un Móvil

2. RECURSO DE APRENDIZAJE 2 (RA2)

2.1.(CE-A) Se ha generado la estructura de clases necesaria para la aplicación.

Para tener un *clean code*, lo mejor es usar el paradigma de programación POO (*Programación Orientada a Objetos*), siguiendo este paradigma Unity crea una estructura de *Game Objects* llamada *Hierarchy*.

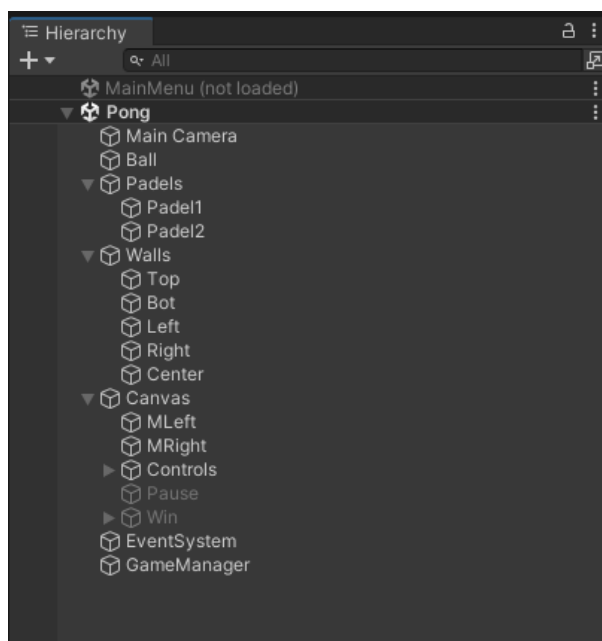


Ilustración 6 – Pestaña Hierarchy Unity

Además, cada *Game Object* tiene diferentes módulos que definen al objeto o realizan funciones, por ejemplo, el módulo *Transform* define la posición, rotación y escala de *Game Object* en la escena, o el módulo *Script* determina el código que será ejecutado por el *Object*.

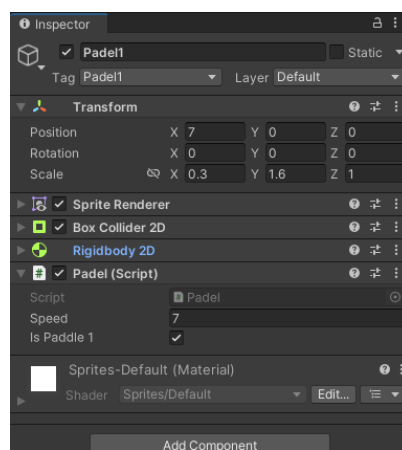


Ilustración 7 – Pestaña Inspector Unity

Para el proyecto de Pong se ha generado la siguiente estructura de Scripts y escenas.

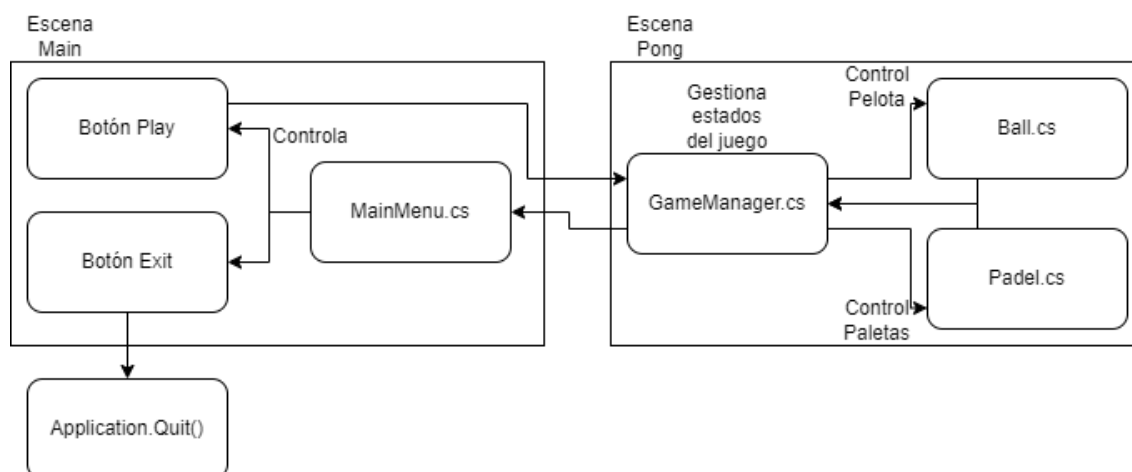


Diagrama 1 – Estructura de clases y escenas en el proyecto Pong

2.2.(CE-B) Se han analizado y utilizado las clases que modelan ventanas, menús, alertas y controles para el desarrollo de aplicaciones gráficas sencillas.

Para realizar interfaces en Unity, lo primero es crear un *gameObject* de tipo *canvas* que define la dimensión de la interfaz o si es escalable o no, entre otras cosas.

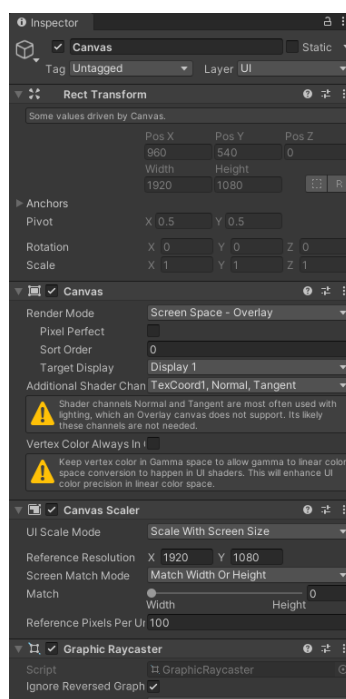


Ilustración 8 – Inspector de un Canvas

El segundo paso es crear un *gameObject* de tipo *EventSystem* para que Unity pueda interactuar con los elementos del canvas, por ejemplo, un botón o un slider.

Si se introduce un elemento UI en la escena con el menú contextual, Unity introduce automáticamente los dos *gameObject* anteriores.

Para trabajar con las ventanas se pueden usar los elementos *legacy* de Unity, pero lo más común en la actualidad es usar la librería Text Mesh Pro, que brinda más funcionalidades que los elementos antiguos no tienen, como el autosize.

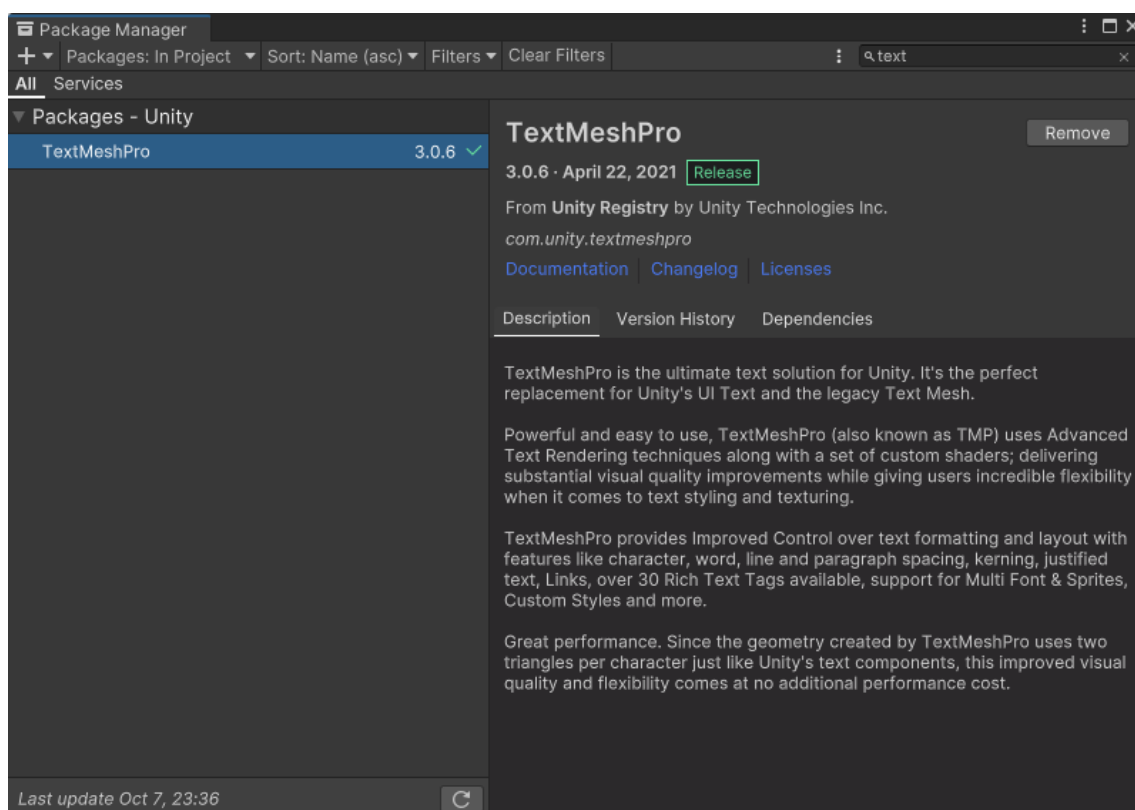


Ilustración 9 – Librería Text Mesh Pro

2.3.(CE-C) Se han utilizado las clases necesarias para la conexión y comunicación con dispositivos inalámbricos.

En Android estudio se puede usar la clase WifiP2pManager para establecer una conexión P2P, para ello hay que establecer unos permisos especiales.

Si lo que se desea es usar el Bluetooth de los dispositivos inalámbricos, Android Studio provee tres clases para su manejo con sus métodos y propiedades correspondientes, bluetoothManager, bluetoothAdapter, BluetoothDevice, BluetoothSocket.

Además, en Unity se provee a los desarrolladores una API de alto nivel para la conexión entre dispositivos inalámbricos vía redes de internet, denominada por sus siglas HLAPI. La documentación oficial de Unity la describe de la siguiente forma:

El API de Alto Nivel (HLAPI) es un sistema para construir capacidades multijugador para juegos de Unity. Está construido encima de la capa del nivel menor del transport de la comunicación en tiempo real, y maneja las tareas comunes que son requeridas para juegos multijugador. Mientras que la capa de transporte soporte cualquier tipo de topología de red, el HLAPI es un sistema servidor autoritaria; aunque solo permite que uno de los participantes sean un cliente y el servidor al mismo tiempo, por lo que no hay un proceso dedicado al servidor requerido. Trabajando en conjunto con los servicios de internet, esto permite a que los juegos multijugador sean jugados a través de la internet con el menor trabajo posible de los desarrolladores.

2.4.(CE-D) Se han utilizado las clases necesarias para el intercambio de mensajes de texto y multimedia.

Dentro de Android Studio hay dos paquetes que permiten al desarrollador realizar comunicaciones SMS y MMS, dichos paquetes son android.telephony.VisualVoicemailSms para SMS y android.provider.Telephony.Mms para MMS.

Dentro de Unity este nos permite crear una estructura cliente servidor con la clase NetworkManger y a su vez la clase NetworkMessages permite mandar mensajes en red.

Si en Unity se está haciendo una aplicación para Universal Windows Platform, dentro de la Player Settings>Publishing Settings>Capabilities>Chat, se pueden habilitar los SMS y MMS

2.5.(CE-E) Se han utilizado las clases necesarias para establecer conexiones y comunicaciones HTTP y HTTPS.

En ciertos proyectos es necesario realizar peticiones HTTP a través de los métodos propios (*GET*, *PUT*, *POST* y *DELETE*) para ello Unity crea la clase *UnityWebRequest*. Para ello te permite crear tres tipos de Objetos.

- Un objeto **UploadHandler** maneja la transmisión de datos al servidor
- Un objeto **DownloadHandler** maneja la recepción, buffering y post-procesamiento de datos recibidos del servidor
- Un objeto **UnityWebRequest** maneja los otros dos objetos, y también maneja el flujo de control HTTP. Este objeto es dónde los encabezados personalizados y URLs se definen, y dónde la información de error y redireccionamiento se almacena.

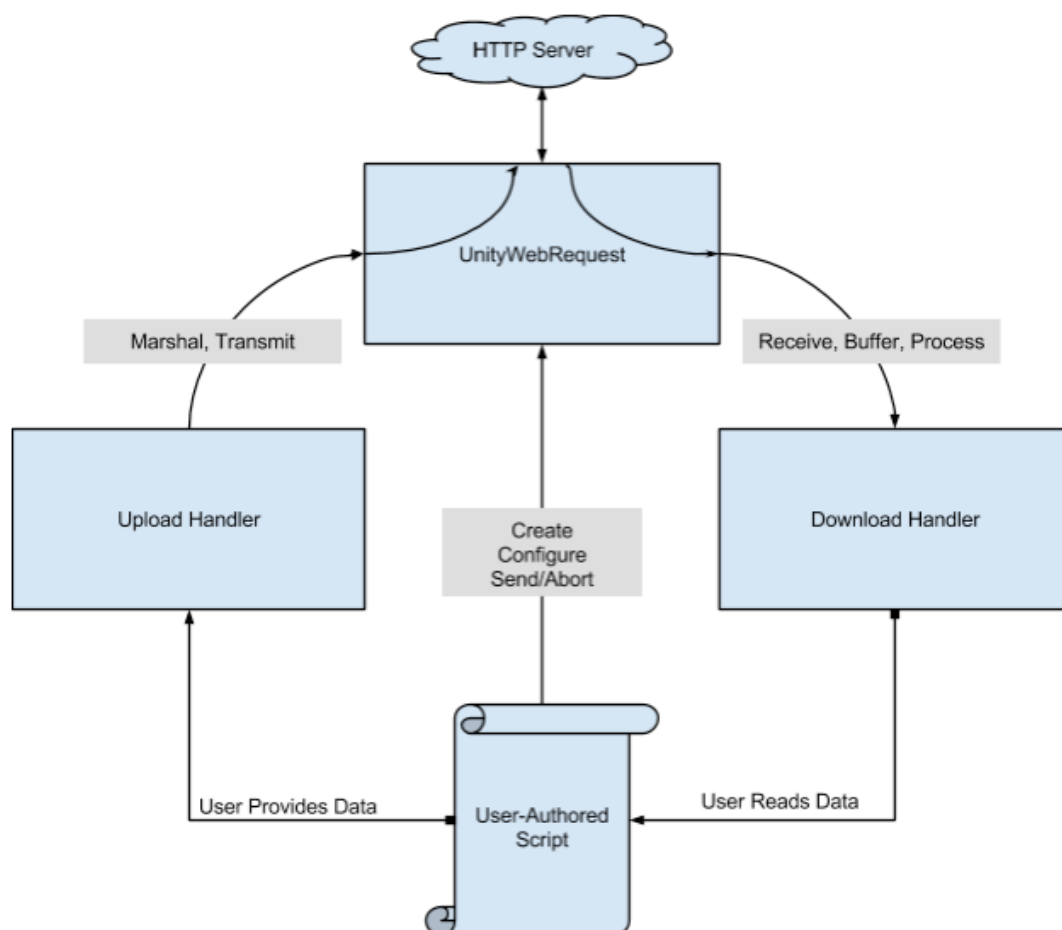


Diagrama 2 – Diagrama de Clases *UnityWebRequest*

Esta clase nos permite, con pocas líneas de código, interactuar con un servidor HTTP, a continuación, muestro un código para una petición GET para recuperar un archivo de binarios.

```
using UnityEngine;
using System.Collections;
using UnityEngine.Networking;

public class MyBehaviour : MonoBehaviour {
    void Start() {
        StartCoroutine(GetText());
    }

    IEnumerator GetText() {
        UnityWebRequest www = UnityWebRequest.Get("http://www.my-server.com");
        yield return www.SendWebRequest();

        if(www.isNetworkError || www.isHttpError) {
            Debug.Log(www.error);
        }
        else {
            // Show results as text
            Debug.Log(www.downloadHandler.text);

            // Or retrieve results as binary data
            byte[] results = www.downloadHandler.data;
        }
    }
}
```

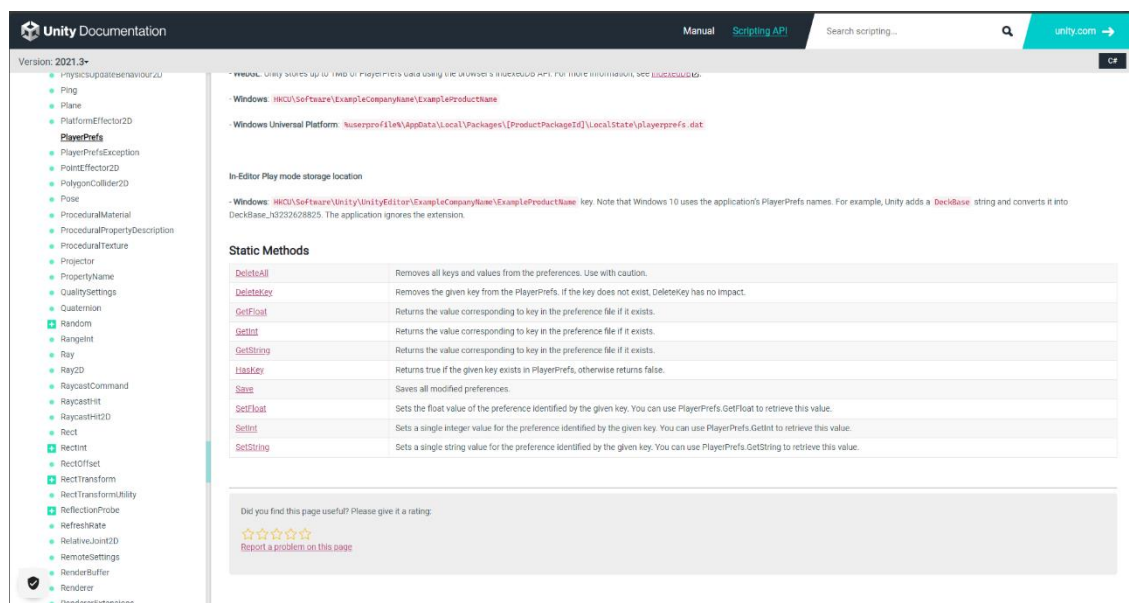
Código 3 – Script para realizar una petición GET

2.6.(CE-F) Se han utilizado las clases necesarias para establecer conexiones con almacenes de datos garantizando la persistencia.

Los proyectos un poco avanzados tienen la necesidad de almacenar datos de forma persistente, para ello en Unity existe tres formas, los *PlayerPrefs*, archivos de binarios y conexión a Bases de Datos.

	PlayerPrefs	Archivos Binarios	BBDD
Ventajas	Archivos de texto muy simples Fácil de Usar en Unity	No se pueden manipular de forma simple. Guardados en local	El SGDB garantiza la integridad y persistencia de los datos
Desventajas	Muy vulnerables a cambios, modificaciones o accesos no deseados	Si son muy grandes pueden llegar a la ejecución del proyecto al leerlos o escribirlos	Datos almacenados en un servidor, el cual aumenta los costes del proyecto

Los *PlayerPrefs* son fáciles de programar ya que Unity desde la versión 2021 implementa una clase (*PlayerPrefs*) para crear estos tipos de ficheros con unos pocos métodos simples.



Web 1 – Métodos clase PlayerPrefs

Si lo que se desea es crear ficheros de binarios, .net y c# permite la serialización de estos, con las clases del paquete System.Runtime.Serialization, además .net permite la serialización a XML y JSON para transmitir información vía internet.

Si se decide por usar una base de datos relacional, se pueden usar las clases de paquete Microsoft.Data.SqlClient, lo más común es usar el conector del sistema gestor de la propia base de datos usando los tres datos básicos para la conexión (ruta al servidor, base de datos a usar y usuario que realiza la conexión). Además, se puede usar una base de datos NoSQL si las necesidades del proyecto así lo requieren.

2.7.(CE-G) Se han realizado pruebas de interacción usuario-aplicación para optimizar las aplicaciones desarrolladas a partir de emuladores.

A lo largo del desarrollo del juego de Pong he realizado diferentes pruebas con mis compañeros para comprobar el correcto funcionamiento en iPhone 13 Pro Max y cuando mis compañeros no podían dedicarme tiempo usaba el emulador de Unity, ya que este me brinda un emulador para el mismo móvil.

El mayor problema que he tenido es que el escalado automático de la interfaz grafica no se ajustaba de forma correcta.

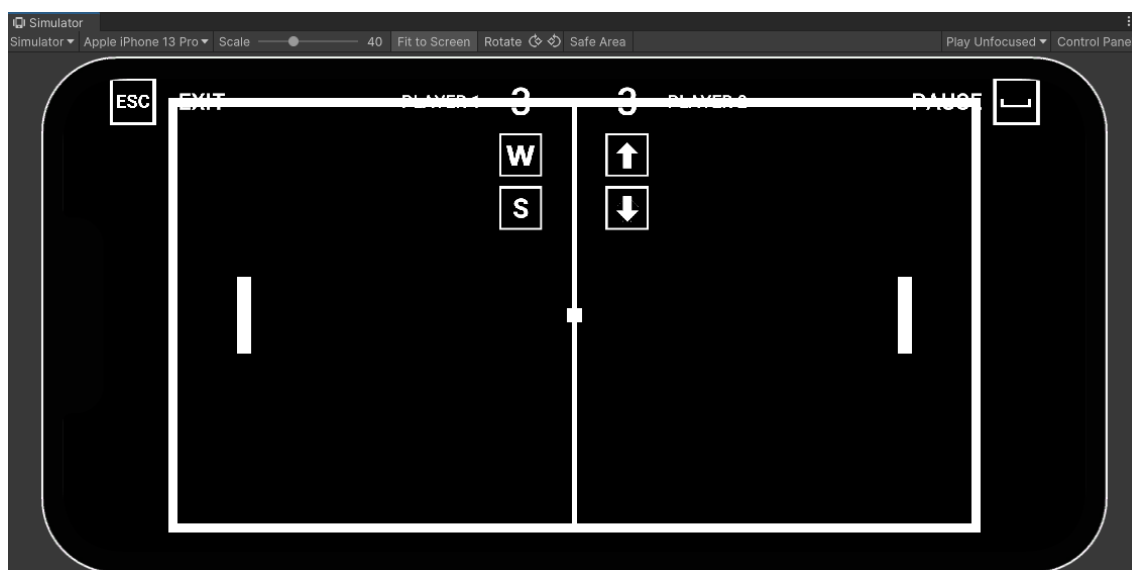


Ilustración 10 – Pestaña Simulator Unity con iPhone 13 Pro Max

2.8.(CE-H) Se han empaquetado y desplegado las aplicaciones desarrolladas en dispositivos móviles reales.

Unity permite la exportación a dispositivos móviles, tanto Android como iPhone, al solo disponer de móviles Android y no tenía la posibilidad de pedir a mis compañeros un iPhone, las pruebas finales las he realizado en un POCO X3 PRO.

Como se dijo en el Apartado 1.2 de este mismo documento, hay que tener los módulos necesarios para poder hacer un “Build”.

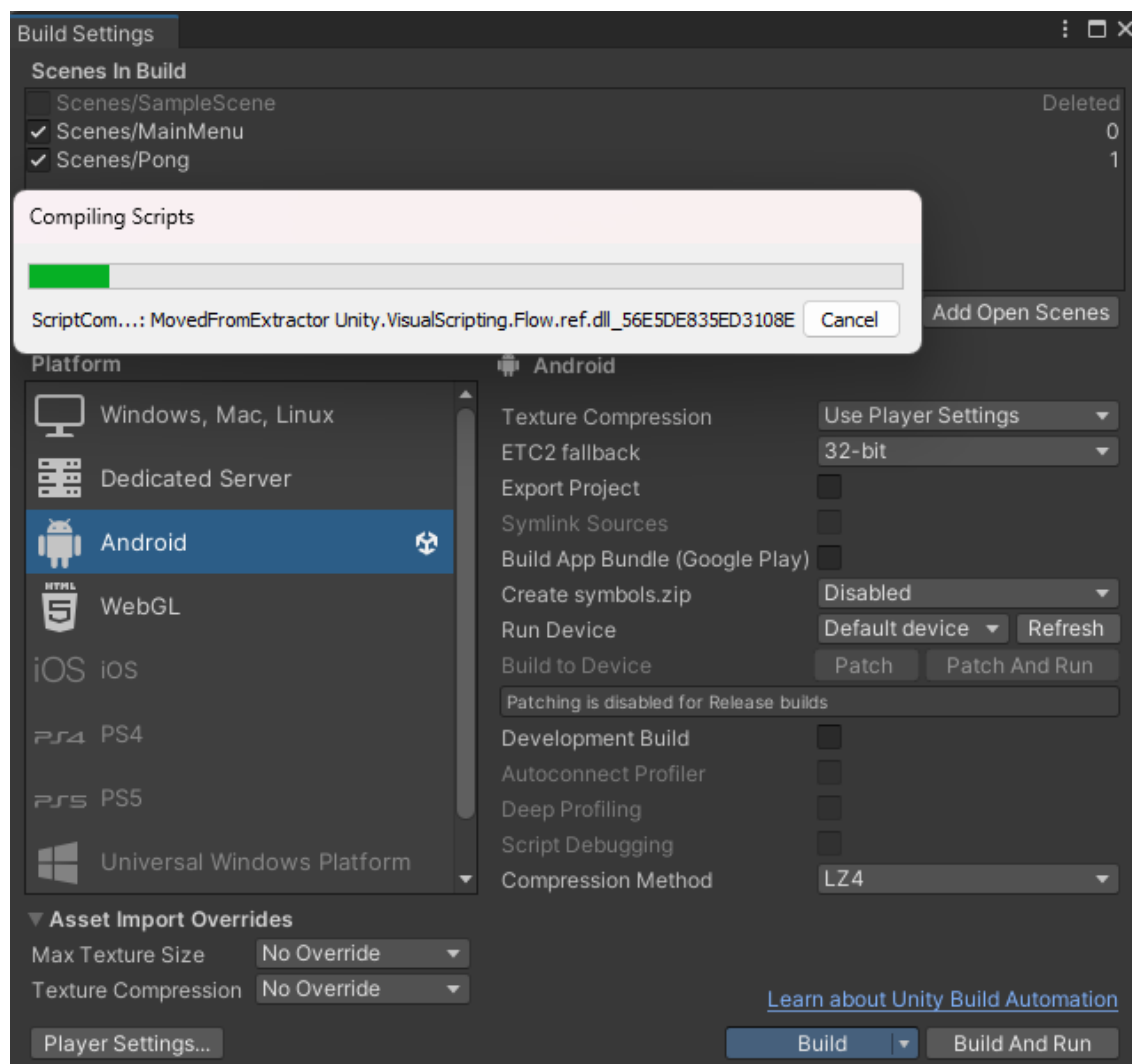
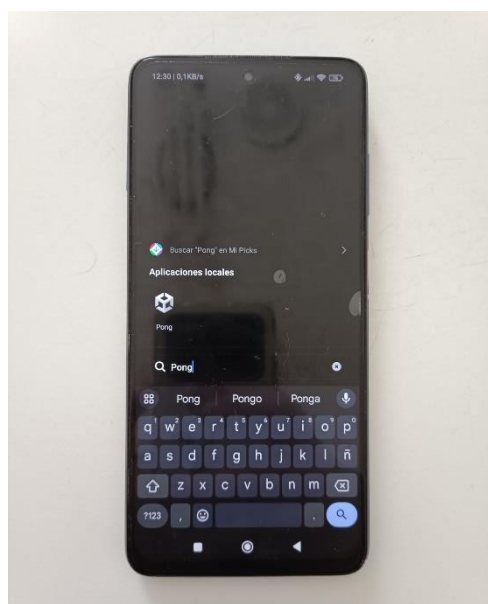


Ilustración 11 – Generar APK desde Unity



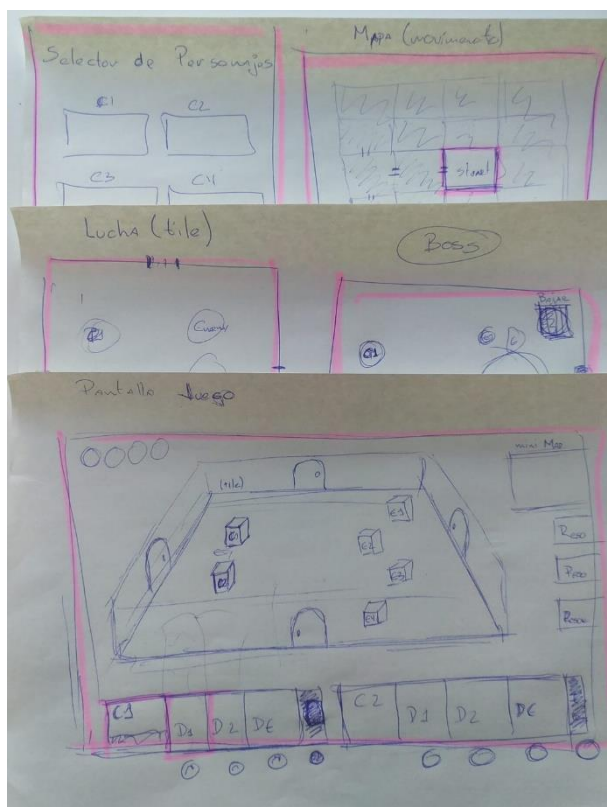
Fotografía 1 – APK Instalada



Fotografía 2 – APK Funcionando

2.9.(CE-I) Se han documentado los procesos necesarios para el desarrollo de las aplicaciones.

Aunque para el Pong no se realizó una documentación ya que era un proyecto para aprender a controlar Unity, ya hemos empezado un nuevo proyecto, en el cual se esta siguiendo un proceso de Documentación, el primer paso fue plasmar la idea que teníamos en un folio en sucio.



Fotografía 3 – Folios plasmando idea del Juego

Tras tener la idea plasmada empezamos un GDD (*Game Document Design*) para tener una estructura bien definida para empezar el desarrollo, este documento no es fijo y se ira cambiando y ampliando a lo largo del desarrollo para que todo el desarrollo del proyecto quede plasmado en un documento.

Combate simple (Enemigos Min 1 - Max 3)

Inicio Escena

- Al entrar a la sala los personajes se colocan en su lugar.
- Aparecen los enemigos.

Desarrollo de turnos

- Se lanzan automáticamente los dados del Jugador.
- Se lanzan los dados del enemigo.
- **Turno del Jugador**
 - El jugador puede o no relanzar uno de sus dados utilizando un botón para relanzar y otro para quedarse con la tirada inicial.
 - El dado especial mostrara actualizara los valores de los dados a los que este afecte aumentando su valor y pintándolo de VERDE
 - Se aplican
 - Uso de dados en combate, se realizará arrastrando el dado al objetivo, el objetivo dando un aura de color este. Esto bloqueara el relanzar dados si aún no se había realizado.
 - Dado Ataque: Arrastrando el dado al enemigo que se quiere golpear, aura Roja
 - Dado Defensa: Se arrastra el dado de defensa al personaje que se le quiere aplicar la defensa. (esto puede ser a un aliado) Aura Azul
 - Es necesario presionar un botón para terminar el turno del Jugador
- **Turno del Enemigo**
 - Los enemigos utilizaran el valor del dado obtenido atacando, defendiendo o utilizando habilidad especial.
 - La defensa que utilizan los enemigos se guardará hasta el siguiente turno para poder defenderse del jugador, al inicio del siguiente turno del enemigo luego desaparecerá
 - Al terminar todos los dados del enemigo termina automáticamente su turno.
- Si quedan Jugadores o Enemigos vivos se reinicia el desarrollo de los turnos.

3

Ilustración 12 – Instantánea del GDD



	Ciclo: Desarrollo de Aplicaciones Multiplataforma Modulo: Programación Multimedia y Dispositivos Móviles
	Título: Investigación sobre Desarrollo Móvil

Tabla de Ilustraciones

WEB 1 – MÉTODOS CLASE PLAYERPREFS	14
FOTOGRAFÍA 1 – APK INSTALADA	16
FOTOGRAFÍA 2 – APK FUNCIONANDO	16
FOTOGRAFÍA 3 – FOLIOS PLASMANDO IDEA DEL JUEGO	17
TABLA 1 – ESTADÍSTICAS DE USO S.O EN DISPOSITIVOS MÓVILES	2
ILUSTRACIÓN 1 – MÓDULOS UNITY 2022.3.9f1	3
ILUSTRACIÓN 2 – SCRIPTS BÁSICOS PONG EN UNITY	4
ILUSTRACIÓN 3 – MAIN MENU PONG	5
ILUSTRACIÓN 4 – PACKAGE MANAGER UNITY 2022	7
ILUSTRACIÓN 5 – UNITY SIMULANDO UN MÓVIL	7
ILUSTRACIÓN 6 – PESTAÑA HIERARCHY UNITY	8
ILUSTRACIÓN 7 – PESTAÑA INSPECTOR UNITY	8
ILUSTRACIÓN 8 – INSPECTOR DE UN CANVAS	9
ILUSTRACIÓN 9 – LIBRERÍA TEXT MESH PRO	10
ILUSTRACIÓN 10 – PESTAÑA SIMULATOR UNITY CON IPHONE 13 PRO MAX	15
ILUSTRACIÓN 11 – GENERAR APK DESDE UNITY	16
ILUSTRACIÓN 12 – INSTANTÁNEA DEL GDD	18

	Ciclo: Desarrollo de Aplicaciones Multiplataforma Modulo: Programación Multimedia y Dispositivos Móviles
	Título: Investigación sobre Desarrollo Móvil

Bibliografía

Anon., s.f. *dev.mysql.com/doc/connector-net*. [En línea]

Available at: <https://dev.mysql.com/doc/connector-net/en/connector-net-connections-string.html#connector-net-connections-open>

Anon., s.f. *developer.android.com - BlueTooth*. [En línea]

Available at: <https://developer.android.com/guide/topics/connectivity/bluetooth?hl=es-419#java>

Anon., s.f. *developer.android.com - P2P*. [En línea]

Available at: <https://developer.android.com/training/connect-devices-wirelessly?hl=es-419>

Anon., s.f. *developer.android.com - SMS*. [En línea]

Available at: <https://developer.android.com/reference/android/telephony/VisualVoicemailSms?hl=en>

Anon., s.f. *docs.unity3d.com - NetworkManager*. [En línea]

Available at: <https://docs.unity3d.com/es/530/Manual/UNetManager.html>

Anon., s.f. *docs.unity3d.com - NetworkMessages*. [En línea]

Available at: <https://docs.unity3d.com/es/2021.1/Manual/UNetMessages.html>

Anon., s.f. *docs.unity3d.com - Player Settings*. [En línea]

Available at: <https://docs.unity3d.com/2022.3/Documentation/Manual/class-PlayerSettingsWSA.html#Configuration>

Anon., s.f. *docs.unity3d.com - PlayerPrefs*. [En línea]

Available at: <https://docs.unity3d.com/2021.3/Documentation/ScriptReference/PlayerPrefs.html>

Anon., s.f. *docs.unity3d.com - UnityWebRequest*. [En línea]

Available at: <https://docs.unity3d.com/es/2019.4/Manual/UnityWebRequest.html>

Anon., s.f. *https://docs.unity3d.com - HLAPI*. [En línea]

Available at: <https://docs.unity3d.com/es/530/Manual/UNetUsingHLAPI.html>

Anon., s.f. *learn.microsoft.com - Serialización*. [En línea]

Available at: <https://learn.microsoft.com/es-es/dotnet/standard/serialization/>

Anon., s.f. *yuzu-emu.org*. [En línea]

Available at: <https://yuzu-emu.org/>

apps.apple.com, 2023. *apps.apple.com*. [En línea]

Available at: <https://apps.apple.com/es/app/xcode/id497799835?mt=12>

bluestacks.com, s.f. *bluestacks.com*. [En línea]

Available at: <https://www.bluestacks.com/es/index.html>

clasificacionde.org, 2022. *www.clasificacionde.org*. [En línea]

Available at: <https://www.clasificacionde.org/clasificacion-de-dispositivos-moviles/>

developer.android.com, 2023. *developer.android.com*. [En línea]

Available at: <https://developer.android.com/studio>

godotengine.org, 2023. *godotengine.org*. [En línea]

Available at: <https://godotengine.org/>

pccomponentes.com, 2022. *pccomponentes.com*. [En línea]

Available at: <https://www.pccomponentes.com/tipos-de-moviles-y-sus-caracteristicas>

spdigitalagency.com, 2023. *spdigitalagency.com*. [En línea]

Available at: <https://spdigitalagency.com/es/blog/popular-frameworks-in-2023-for-mobile-application-development>

statcounter.com, 2023. *statcounter.com*. [En línea]

Available at: <https://gs.statcounter.com/os-market-share/mobile/worldwide/#monthly-200901-202309>

unity.com, s.f. *unity.com*. [En línea]

Available at: <https://unity.com/es>

unrealengine.com, s.f. *unrealengine.com*. [En línea]

Available at: <https://www.unrealengine.com/>

Wikipedia.org, 2023. *Wikipedia.org - Iphone1*. [En línea]

Available at: [https://es.wikipedia.org/wiki/IPhone_\(1.%C2%AA_generaci%C3%B3n\)](https://es.wikipedia.org/wiki/IPhone_(1.%C2%AA_generaci%C3%B3n))

Wikipedia.org, 2023. *Wikipedia.org - Kotlin*. [En línea]

Available at:

[https://es.wikipedia.org/wiki/Kotlin_\(lenguaje_de_programaci%C3%B3n\)#::~:~:text=El%207%20de%20mayo%20de,alternativa%20al%20compilador%20Java%20est%C3%A1ndar.](https://es.wikipedia.org/wiki/Kotlin_(lenguaje_de_programaci%C3%B3n)#::~:~:text=El%207%20de%20mayo%20de,alternativa%20al%20compilador%20Java%20est%C3%A1ndar.)