DEPARTAMENTO DE INFORMÁTICA
DESARROLLO DE APLICACIONES MULTIPLATAFORMA

Autor : MIGUEL RODRIGUEZ GUERRA
Tutor: ROBERTO GONZALEZ SIERRA

Análisis, diseño y desarrollo de un minijuego utilizando Unity

San Andrés del Rabanedo, Mayo de 2019.

Índice de contenido

1 . Estudio del problema y análisis del sistema.

1.1 Introducción.

“El Whack-A-Mole (Guacamole en España) es un juego arcade conocido por todos en la época de 1977. Una máquina Whack-A-Mole típica consiste en un cajón grande a nivel de la cintura con cinco orificios en su parte superior y un mazo negro grande y suave. Cada orificio contiene un solo topo de plástico y la maquinaria necesaria para moverle hacia arriba y hacia abajo. Una vez que el juego comienza, los topos comenzarán a aparecer de sus agujeros al azar. El objetivo del juego es forzar a los topos de vuelta a sus agujeros golpeándolos directamente en la cabeza con el mazo, lo que aumenta la puntuación del jugador. Cuanto más rápido se haga, mayor será la puntuación final.

El gabinete tiene una lectura de tres dígitos de la puntuación del jugador actual y, en modelos posteriores, la mejor puntuación de la lectura del día. El mazo generalmente se sujeta al juego con una cuerda para evitar que alguien se vaya con él.

Las versiones actuales de Whack-A-Mole incluyen tres pantallas para Bonus Score(al acertar topos seguidos sin fallar), High Score (la puntuación mas alta del día) y la puntuación actual del juego. Las versiones para el hogar, distribuidas por Bob's Space Racers, incluyen solamente una pantalla para mostrar la puntuación actual.

Si el jugador no golpea un topo dentro de un cierto tiempo o con suficiente fuerza, eventualmente se hundirá en su agujero sin puntuación. Aunque el juego comienza lo suficientemente lento como para que la mayoría de las personas golpeen a todos los topos que aumentan, aumenta gradualmente su velocidad, ya que cada topo pasa menos tiempo sobre el agujero y con más topos fuera de sus agujeros al mismo tiempo. Después de un límite de tiempo designado, el juego termina, independientemente de la habilidad del jugador. La puntuación final se basa en el número de topos que golpeó el jugador.”
-”https://searchsecurity.techtarget.com/definition/whack-a-mole”-
-”"もぐら叩きを作った男"[A man who made Whac-A-Mole] (in Japanese)”-

Este proyecto se basa a grandes rasgos en hacer una versión propia de lo que sería un juego de Whack-A-Mole antiguo con un menú principal, una pantalla en la que jugar y golpear a los topos, conseguir hacer los I/O de la información del usuario, registrar una Base de Datos para la aplicación utilizando FireBase y aprender las bases de la arquitectura de Unity como motor gráfico.

1.2 Finalidad.

Este proyecto tiene como finalidad el iniciarme en el mundo de la creación de videojuegos que utilizan Unity como motor gráfico y también el ayudarme a aprender sobre el mismo de una manera general al plantearme problemas que debo investigar por mi cuenta. Ademas ayudará a reforzar los conocimientos que ya he conseguido a lo largo del curso y que serán utilizados para facilitar la comprensión de este nuevo motor y lenguaje así como para sentar las bases y conseguir unos conocimientos generales que me ayuden a mejorar mis habilidades como usuario de Unity y como programador en general.

Además, sirve como proyecto de prueba en el que añadir o quitar funciones dependiendo de lo que esté aprendiendo en el momento, al ser un juego simple pero interactivo, se pueden añadir un montón de singularidades(Rachas de golpes, objetos que caen con los que interactuar y que dan algun bonus al jugador, conocimientos sobre diseño gráfico, creación de sprites...) sin que el juego pierda su esencia arcade y así practicar y probar nuevas opciones en cuanto a como será el juego al final.

1.3 Requisitos.

Ofrecer a los usuarios un rato de diversión, al final lo que importa en un juego es que jugarlo aporte cierta dificultad al usuario, así es como se convierte en entretenido y competitivo. El Whack-A-Mole es un gran ejemplo de como un juego muy simple puede ofrecer horas y horas de diversión además de reforzar los reflejos, mejorar la coordinación ojo-mano y desarrollar diversas áreas del cerebro que tienen que ver con la orientación y el movimiento espacial.

2 Recursos

2.1 Recursos Hardware.

Para la realización de este proyecto se han utilizado :

- Un portátil Asus X550V Intel Core i7-6700HQ 2,6Ghz ,8GB RAM , NvidiaGeforce950M, 1TB.
- Un ratón Victsing 5500dpi.

2.2 Recursos Software.

- Unity : es un motor de videojuego multiplataforma creado por Unity Technologies. Unity está disponible como plataforma de desarrollo para Microsoft Windows,OS X y Linux.

  Utilizado para la creación del juego así como de los scripts y el código interno utilizados para el mismo, además posee una herramienta llamada Inspector que nos permite manipular mover e interactuar con todos los objetos presentes en la escena, lo cuál facilita un montón la parte visual y la distribución de los componentes internos de nuestro juego.

- Firebase : Utilizado para la creación de la base de datos, la ejecución de reglas de escritura y lectura así como la conexión con la aplicación y el guardado de datos en la misma.

- OpenOffice Writer: Utilizado para la creación de este archivo.

- VisualStudio : Utilizado para la edición de scripts, se conecta automáticamente a Unity para facilitar el compilado.

3 Planificación

3.1 Planificación temporal

El proyecto se basa en 4 partes principales:

- Creación de la escena principal y código de los topos para conseguir que el juego sea jugable, actualizado de las puntuaciones cuando el usuario acierta,etiquetas para mostrar tiempo y puntuación actual, botón de reinicio, tabla de puntuaciones final con datos de usuario... todo lo necesario para que el juego pueda ser jugado y las puntuaciones recogidas.

- Creación de las distintas pantallas por las que navegar en el juego, menú principal que conecte con el juego, diseño de las pantallas para que se asemejen a un diseño clásico de Whack-A-Mole.

- Creación de la Base de datos en la que guardar las puntuaciones y recuperarlas para la pantalla de puntuaciones.

- Redactar la memoria de proyecto.

Las primeras dos son las que mas tiempo consumen por lo que tendrán un mayor plazo de ejecución, unos 3 meses, dejando los dos meses restantes para la base de datos, aprender sobre Firebase y redactar el final de la memoria dado que el resto será escrito a medida que se vaya programando el juego y surjan problemas que solucionar y curiosidades que mencionar.

3.2 Planificación económica

Este proyecto no ha sido pensado para ser distribuido a nivel comercial debido a que es una práctica de aprendizaje por lo que los ingresos serán nulos y todos los gastos necesarios serán depositados personalmente por su creador.

4 Desarrollo y pruebas

Descripción por apartados de las distintas fases del proyecto, incluyendo la documentación técnica y explicación del desarrollo del proyecto (esquemas, capturas de pantalla, ficheros de configuración, características técnicas, códigos fuente, logs de instalaciones, diseño de entidades y relaciones, etc.). Realización y descripción de las pruebas realizadas para verificar y/o mejorar el correcto funcionamiento del sistema

4.1 Inicio del programa

La primera parte del proyecto es la creación de la base del juego y el posicionamiento de los distintos Objetos tanto de UI como físicos que nos encontraremos al jugar, necesitaremos unos botones que actúen como topos así como dos Text que nos muestren el tiempo restante y la puntuación actual. Los PlaceHolders mantendrán la posición de los topos sea cual sea la resolución de la pantalla y cada Botón poseerá un texto interno que nos muestre información de su estado.

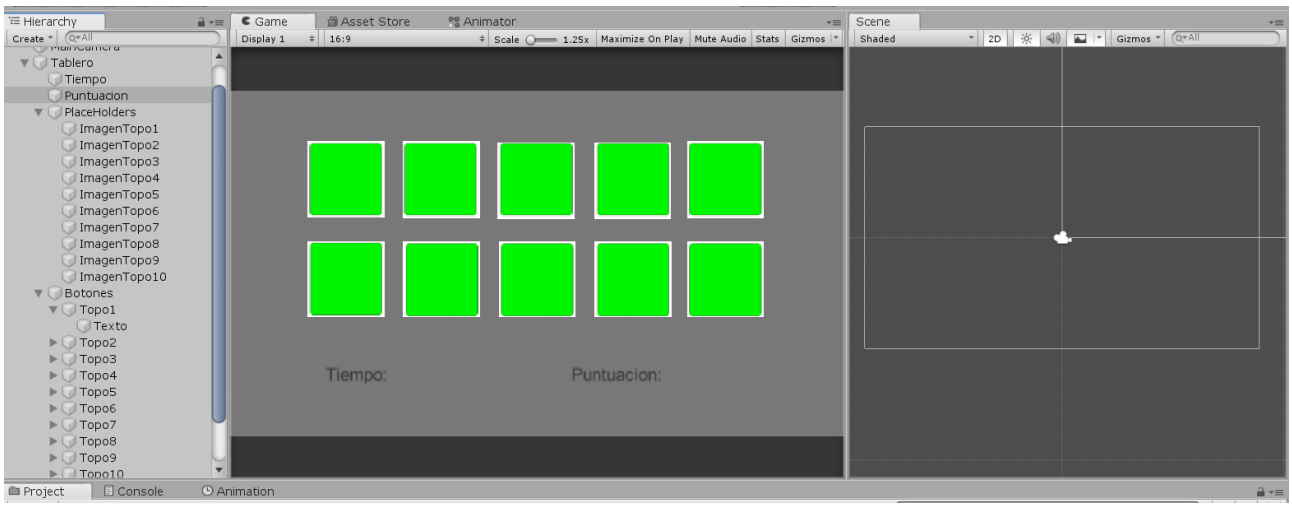


Ilustración 1: GameScene preview

El primer problema que me encontré fue entender como funcionan los PlaceHolders, al principio cuando posicionaba los botones, si cambiaba de resolución de pantalla producía errores, se solapaban, fue el primer error grave que tuve que solucionar debido a que en el curso de Udemy no enseñan nada de PlaceHolders al enfocarse en juegos de otro estilo. La solución la encontré en un vídeo que explicaba perfectamente como "sujetar" distintos componentes que queremos fijos utilizando los nombrados PlaceHolders -"youtube.com/watch?v=Oq0YMlWwd1I".

Posteriormente a tener todos nuestros componentes fijos y comprobar que todos tienen su botón y su texto creamos el primer script "ButtonManager" que se encargará de manejar el comportamiento de los Botones.

4.2 ButtonManager

Nuestro primer Script se encargará del comportamiento de los Botones, constará de dos variables scoreManager y timeManager que se encargarán de la puntuación y de el tiempo de juego respectivamente.

El código general de la clase sería :

```
public class ButtonManager : MonoBehaviour {

    public ScoreManager scoreManager;
    public TimeManager timeManager;

    //Si añadimos un Tooltip y pasamos el ratón por encima en el editor, nos mostrara este mensaje.
    [Tooltip("Conectar todos los botones a esto")]
    public GameObject[] botones;//Creamos un Array de botones.


    //Comenzamos con 0,5 segundos, luego será aleatorio.
    float randomTime = .5f;

    //Poniendolo publico podemos controlarlo mas facilmente desde ejecucion cambiando los valores para ver que es mas natural.
    public float hideTime = 1.0f;

    // Use this for initialization
    void Start () {

        //Al comienzo de la partida ponemos todos los botones a inactivo.
        for (int i = 0; i < botones.Length; i ++)
        {
            botones[i].SetActive(false);
        }

        //Tambien comenzamos una coroutine, un evento tardío que comienza a iterar los topos aleatoriamente.
        StartCoroutine(MostrarTopo());
    }
```

Ilustración 2: Código de ButtonManager

Además añadiremos aquí 3 métodos que se encargarán de la iteración de los topos así como de controlar cuando uno ha sido golpeado.

En el primer método conseguimos que los topos se muestren continuamente con un corto delay, para después comenzar una SubRutina que los vuelva a esconder. Utilizando el método mostrado aquí :

```
//Esto se repite hasta el fin del juego.
IEnumerator MostrarTopo(){

    yield return new WaitForSeconds(randomTime);

    //Permite que la coroutine funcione siempre y cuando el juego este activo.
    if (timeManager.gameOver == false)
    {
        //Escoge un boton aleatorio.
        int botonAleatorio = Random.Range(0, botones.Length);

        //Comprobamos que el boton al que estamos accediendo este inactivo.
        if (botones[botonAleatorio].activeInHierarchy == false)
        {
            botones[botonAleatorio].SetActive(true);//Lo activamos

        }

        //Creamos un nuevo tiempo aleatorio
        randomTime = Random.Range(0, 1f);

        //Llamamos a la coroutine en ella misma para que se repita constantemente.
        StartCoroutine(MostrarTopo());

        //Desactiva el botón al cabo de un rato para que ya no sea pulsable.
        StartCoroutine(EsconderTopo(botones[botonAleatorio]));
    }

}
```

Ilustración 3: Método MostrarTopo

Ahora solo nos queda terminar el código del script añadiendo los métodos que se encargan de ocultar los topos tras un corto tiempo así como de controlar que el topo activo haya sido golpeado.

```
IEnumerator EsconderTopo(GameObject miBoton)
{
    yield return new WaitForSeconds(hideTime);

    //Comprobamos que siga activo (Verde) para no esconder un boton que ha sido pulsado.
    if (miBoton.GetComponentInChildren<Image>().color == Color.green)
    {
        //Lo ponemos inactivo para poder llamarlo de nuevo
        miBoton.gameObject.SetActive(false);

        //Reducimos la puntuacion del jugador al no darle.
        scoreManager.Puntuar(5, false);

    }


}
```

Ilustración 4: Método EsconderTopo

```
IEnumerator TopoGolpeado(GameObject miBoton)
{
    yield return new WaitForSeconds(.5f);
    //cambia el color de nuevo a verde y resetea el texto
    miBoton.GetComponentInChildren<Text>().text = "";
    miBoton.GetComponentInChildren<Image>().color = Color.green;

    //Lo ponemos inactivo para que pueda ser llamado de nuevo.
    miBoton.gameObject.SetActive(false);


}
```

Ilustración 5: Método TopoGolpeado

Utilizando un código de color (Verde, activo) (Rojo, inactivo) conseguimos que el usuario sepa cuando debe golpear y también cuando ha golpeado, también facilita la referencia al objeto, al saber que su color será verde si y solo si esta activo, lo cuál facilita la codificación.

4.3 GameManager

Para la conexión continua de los Topos con el tiempo y la puntuación y para facilitar la referenciación a la información que se muestra en pantalla es muy común la utilización de un Objeto GameManager al cuál se le aplican varios scripts a la vez como vemos en la Imagen :

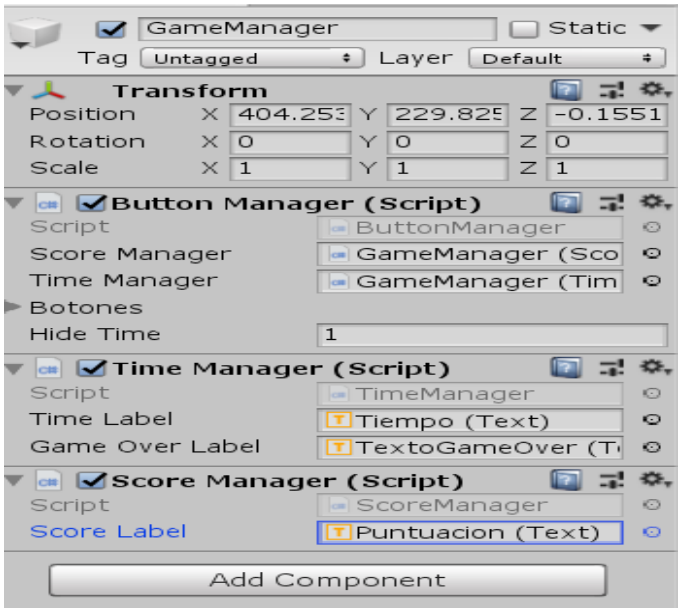


Ilustración 6: Ejemplo de varios Scripts en un mismo Objeto

Lo que conseguimos con esto es la facilidad para mantener toda la información de nuestro juego, ejecutándose en un mismo Objeto, lo cual hace que todo sea mas accesible.

4.4 ScoreManager & TimeManager

Nuestros siguientes scripts son el encargado de la Puntuación y el encargado del Tiempo. Como se puede apreciar en la imagen :

```
public void Puntuar(int cantidad, bool acertado){

    if(acertado == true)
    {
        //Si acertado es true significa que ha acertado y debemos sumar

        //puntuacion actual
        score = PlayerPrefs.GetInt(ScoreTypes.PlayerScore.ToString());
        //le sumamos la amount
        score += cantidad;
        //La guardamos
        PlayerPrefs.SetInt(ScoreTypes.PlayerScore.ToString(), score);

        //Aumentamos los golpes acertados tambien.
        int golpes = PlayerPrefs.GetInt(ScoreTypes.Golpes.ToString());
        golpes += 1;
        PlayerPrefs.SetInt(ScoreTypes.Golpes.ToString(), golpes);


    }
    else
    {
        //Aumentamos los golpes fallidos para mostrarlos al final de partida.
        int fallos = PlayerPrefs.GetInt(ScoreTypes.Fallos.ToString());
        fallos += 1;
        PlayerPrefs.SetInt(ScoreTypes.Fallos.ToString(), fallos);
    }

    //Actualizamos la Puntuacion.
    scoreLabel.text = "Puntuacion:" + score;

}
```

Ilustración 7: Método Puntuar

El script ScoreManager tendrá un método Puntuar que será el que reciba por parámetros si el jugador acertó o fallo (con un booleano) para sumar o restar a la puntuación total así como tambien actualizar la información al ir acertando o fallando.

El script TimeManager se encargará de contar el tiempo que ha transcurrido desde que comenzamos el juego, contiene una variable pública que nos deja modificar el tiempo de juego según lo que nos parezca mejor.

En la imagen:

```
// Update is called once per frame
void Update () {


    //Comienza la cuenta atras.
    maxTime -= Time.deltaTime;// con -= contamos hacia atrás.

    //Necesitamos convertirlo a segundos para que se muestre correctamente
    int seconds = Mathf.RoundToInt(maxTime % 60f);

    //Si no hemos llegado a 0
    if (seconds >= 0)
    {
        //Actualizar la etiqueta Tiempo : con los segundos restantes
        timeLabel.text = "Tiempo: " + seconds.ToString("00");

    }
    else
    {
        //Solo se llama una vez en el Update.
        if (gameOver == false)
        {

            gameOver = true;
            //Activamos la Label de gameOver
            gameOverLabel.gameObject.SetActive(true);

            //Tras un corto periodo, cambia la escena.
            StartCoroutine(ChangeScene());
        }

    }
}
```

Ilustración 8: Método Update

Observamos el método Update() que consta de un tiempo máximo y una función simple que cuenta hacia atrás hasta llegar a 0. Cuando llega a 0 se comienza la CoRutina ChangeScene que nos lleva a la pantalla de puntuaciones con nuestra información sobre la partida.

```
IEnumerator ChangeScene()
{
    //Ajustamos el delay a 2 segundos, posteriormente se mostrará la puntuación
    yield return new WaitForSeconds(2);
    //Cambiamos la escena utilizando el SceneManager y uno de nuestros enum que pasaremos a String primero.
    SceneManager.LoadSceneAsync(GameScenes.ScoreScene.ToString());

}
```

Ilustración 9: Cómo cambiar de escena

4.5 MainMenu

Para crear un MainMenu se necesita una nueva escena, las nuevas escenas deben ser referenciadas en la “Build” del proyecto para que pasen a formar parte de este y funcionen sin problemas. Creamos nuestra nueva escena y dentro de ella añadimos varios botones que nos permitiran navegar por las distintas pantallas del juego, comenzar partida o salir del mismo. También agregamos dos imagenes, una es la Imagen título del proyecto y la otra el background :

Ilustración 10: ImagenTitulo

Ilustración 11: Background del juego

Ahora que tenemos nuestro Menú Principal con su parte gráfica casi bonita vamos a realizar el Script con las funciones para los botones.

4.6 LevelManager

Necesitamos un Script con 3 Métodos, uno para cada botón del menú, y cada uno nos debe

llevar a una Escena diferente por lo que en este Script usaremos el `UnityEngine.SceneManagement;`

Una librería de Unity que nos permite controlar el paso de una escena a otra de forma sencilla y rapida. Gracias a `SceneManager.LoadSceneAsync();` podremos cambiar de escena con solamente el nombre de ésta.

```
public class LevelManager : MonoBehaviour
{
    // Start is called before the first frame update
    void Start()
    {

    }

    // Update is called once per frame
    void Update()
    {

    }

    public void StartGame()
    {
        SceneManager.LoadSceneAsync(GameScenes.GameScene.ToString());

    }

    public void ExitGame()
    {
        Application.Quit();
    }

    public void ShowScores()
    {
        SceneManager.LoadSceneAsync(GameScenes.ScoreTable.ToString());
    }
}
```

Ilustración 12: LevelManager

Como se puede apreciar en la imagen solo necesitamos una línea de código en cada método para pasar entre pantallas, pero aún no tenemos las pantallas restantes así que de momento dejaremos los métodos sin conectar a los Botones y solo conectaremos el de Exit.

4.7 ScoreScene

La escena en la que se mostrarán las puntuaciones cuando el juego haya terminado , necesitaremos varias Label en las que mostrar la puntuación así como los fallos y aciertos que tuvo el jugador. También necesitaremos unos cuantos botones para hacer la navegación, podemos volver a jugar (RESTART) podemos volver al menú sin guardar nuestra puntuación y podemos guardar nuestra puntuación, lo cual nos llevará a la pantalla de SaveScore.

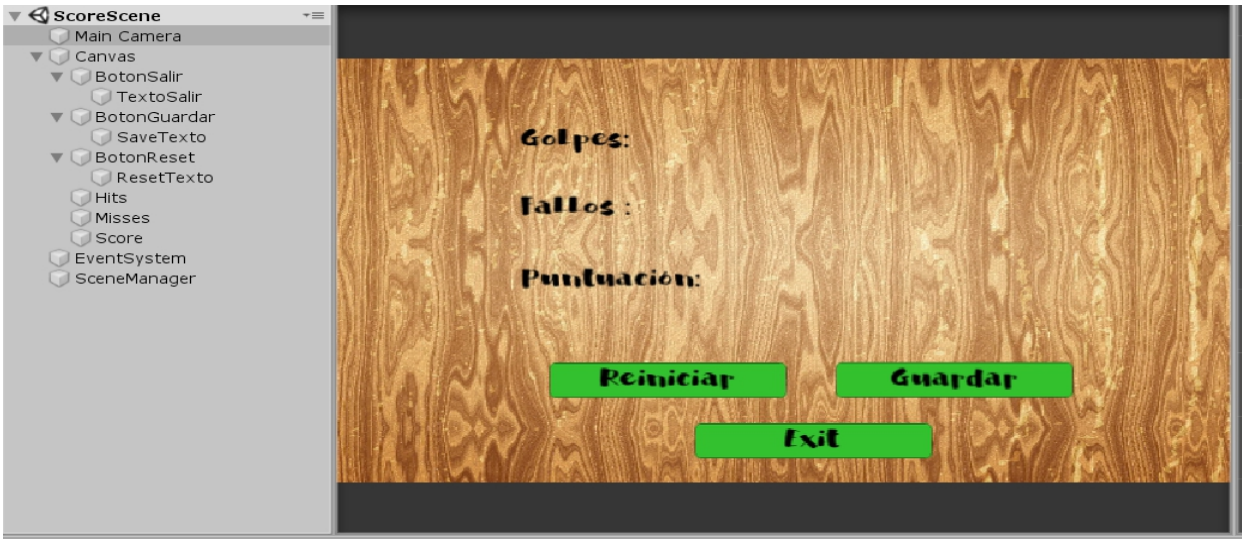


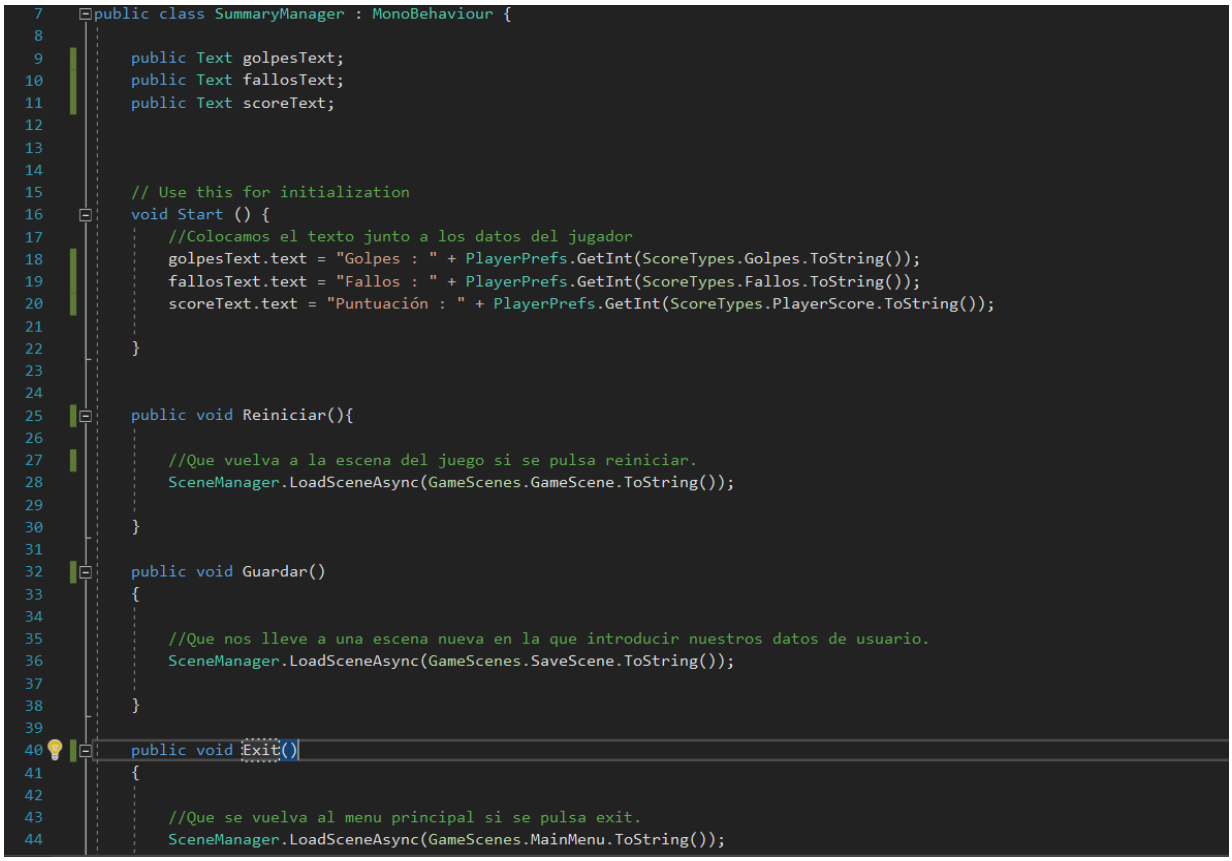
Ilustración 13: Escena que muestra la puntuación

Después de haber configurado el BackGround con nuestra imagen background y de editar un poco el color de la escena así como la Fuente utilizada para las letras del juego (dafont.com/DK Garden) quedaría algo así, faltando por supuesto el Script que se encargará del comportamiento de la escena y de llevarnos a distintas escenas dependiendo lo que queramos.

4.8 SummaryManager

Este Script debe referenciar a las Label que mostrarán la información ,obtener ésta y añadírsela. Para obtener la información del usuario utilizamos una herramienta propia de Unity llamada PlayerPrefs (Preferencias de usuario) que nos permite guardar datos durante la ejecución del juego y dejarlos accesibles desde cualquier lugar.

Además debe tener el código de nuestros 3 botones :



```
public class SummaryManager : MonoBehaviour {

    public Text golpesText;
    public Text fallosText;
    public Text scoreText;



    // Use this for initialization
    void Start () {
        //Colocamos el texto junto a los datos del jugador
        golpesText.text = "Golpes : " + PlayerPrefs.GetInt(ScoreTypes.Golpes.ToString());
        fallosText.text = "Fallos : " + PlayerPrefs.GetInt(ScoreTypes.Fallos.ToString());
        scoreText.text = "Puntuación : " + PlayerPrefs.GetInt(ScoreTypes.PlayerScore.ToString());

    }


    public void Reiniciar(){

        //Que vuelva a la escena del juego si se pulsa reiniciar.
        SceneManager.LoadSceneAsync(GameScenes.GameScene.ToString());

    }

    public void Guardar()
    {

        //Que nos lleve a una escena nueva en la que introducir nuestros datos de usuario.
        SceneManager.LoadSceneAsync(GameScenes.SaveScene.ToString());

    }

    public void Exit()
    {

        //Que se vuelva al menu principal si se pulsa exit.
        SceneManager.LoadSceneAsync(GameScenes.MainMenu.ToString());
```

Ilustración 14: SummaryManager

4.9 SaveScene

La escena en la que permitiremos mediante el uso de un InputField, la introducción del playerName que el usuario elija para posteriormente guardarlo en un Objeto de la clase User , clase básica (posee solamente un nombre de usuario y una puntuación, que es lo que guarda en la base de datos) y Serializable (permite ser guardada en la Base de Datos al convertirse en una clase de tráfico de datos, un IO) mostrada aquí :

```
[Serializable]
public class User
{
    public string userName;
    public int playerScore;

    public User()
    {
        userName = SaveScore.playerName;
        playerScore = SaveScore.playerScore;
    }

    public User(User user)
    {
        this.userName = user.userName;
        this.playerScore = user.playerScore;
    }

    override
    public string ToString()
    {
        return userName + " " + playerScore;
    }
}
```

Ilustración 15: Clase User para la BBDD

Además debe poseer el botón que confirme el guardado y devuelva al jugador al menú principal, donde podrá volver a jugar, comprobar su puntuación, o salir. Para realizar el guardado en la base de datos creamos una nueva tabla en Firebase.

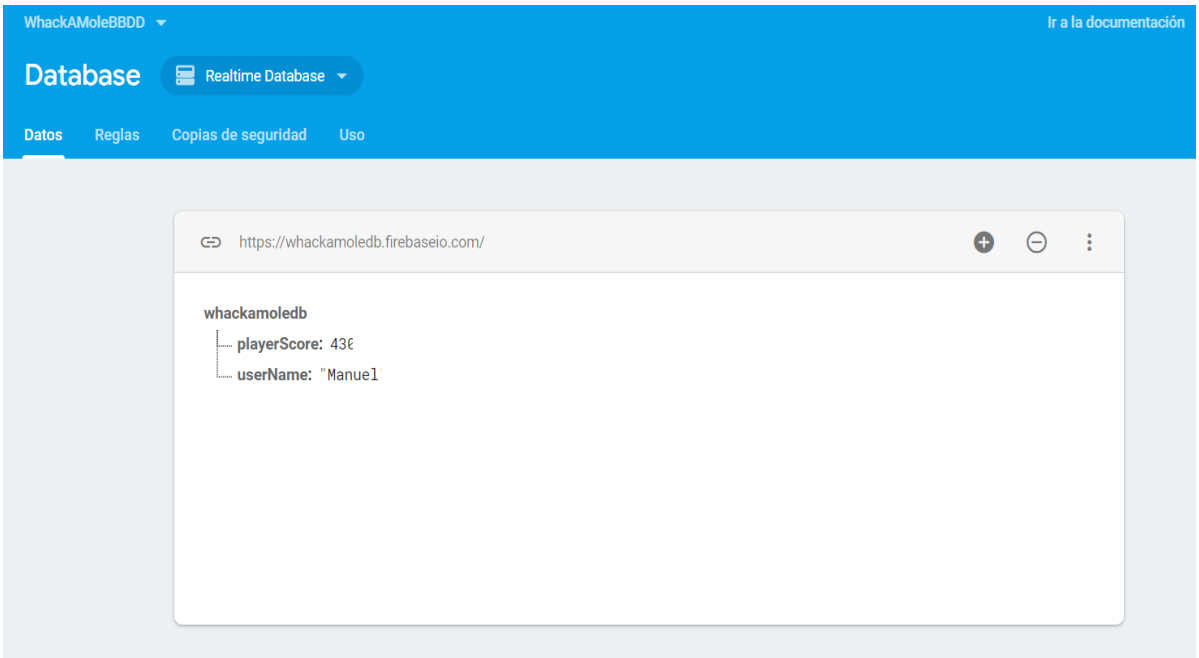


Ilustración 16: Ejemplo de tabla en Firebase

Es necesario poner la base de datos en modo RealTimeDataBase, para poder interactuar con ella en cualquier momento y probar nuestro código.

4.10 SaveScore

Para nuestra escena de guardar la puntuación conseguida necesitaremos un scripts con el método que asignarle al botón de guardar, que recogerá el nombre escrito en el InputField y la puntuación guardada anteriormente , creará un User con esos datos y lo mandará con una sentencia PUT() a la base de datos, en la que se guardará como un objeto Ipromise.<User>.

Para conectar con la base de datos necesitaremos un UnityAsset, que conseguiremos gratuitamente en la AssetStore propia de Unity, una biblioteca gigante llena de proyectos y pruebas hechos por otros programadores, con mucho contenido que podemos importar a nuestro proyecto. El Asset que buscamos se llama RestClient y se utiliza para realizar comunicaciones con Bases de Datos en Firebase.

```
public class SaveScore : MonoBehaviour
{

    public Text scoreText;
    public InputField nameText;

    public static int playerScore;
    public static string playerName;


    // Start is called before the first frame update
    void Start()
    {
        playerScore = PlayerPrefs.GetInt(ScoreTypes.PlayerScore.ToString());
        scoreText.text = "Tu puntuación : " + playerScore;
    }

    //Se lo añadimos al boton de guardar para que recoja el nombre del input y llame a la BBDD
    private void SaveButton()
    {
        playerName = nameText.text;
        SubirABaseDatos();

    }

    //Crea un usuario con los datos del jugador y lo sube a la BBDD, devuelve al main menu
    private void SubirABaseDatos()
    {
        User user = new User();
        RestClient.Put("https://whackamoledb.firebaseio.com/.json", user);
        SceneManager.LoadSceneAsync(GameScenes.MainMenu.ToString());
    }
```

Ilustración 17: SaveScore

4.11 Testing y Pruebas

Varios de los problemas encontrados se han debido a la falta de dominio de la plataforma Unity, la mayoría de las veces cuando había un problema que no conseguía solucionar encontraba algún Thread en el foro de Unity que me mostraba una nueva manera, mas simple , de hacer lo que yo quería conseguir. Esto es lo que me ha impulsado a iniciar el curso de Udemy para el desarrollo

de videojuegos 2D en Unity, gracias a éste he conseguido conocimientos necesarios para que mi aplicación finalmente pudiera funcionar.

Al principio pensaba hacer hoyos y que de ellos salieran topos cada X tiempo, el problema llegó cuando necesite de la ayuda de imágenes de los topos para la creación de sprites que les dieran movimientos, al no tener ninguna imagen gratuita que me pudiera ayudar opté por comprobar si había algún PreFab de Unity que ya fuera un Topo que sube y baja o algo parecido, a mi pesar comprobé que no existía. Por lo que me planteé seriamente el aprender diseño gráfico, pero eran un montón de horas de las que no disponía y ya tenía que completar otro curso que me ayudara con la parte de C# , Escenas, VisualStudio...

5 Conclusiones finales

5.1 Grado de cumplimiento de los requisitos fijados

Objetivos cumplidos :

- Comprensión del motor gráfico Unity y de sus distintas funciones utilizando un curso de udemy presentado por el programador Juan Gabriel Gomila.
- Realización de la base del juego, generando un tablero donde se colocaran los topos así como de un menú principal que nos permita acceder al juego
- Realización de los cálculos necesarios para el cúmulo de puntuaciones dependiendo de los golpes que hayamos conseguido dar a los topos.
- Debugging y pruebas de las partes importantes del proyecto y bugfixing de los errores que surjan al testearlo probando que se pueda jugar y que los componentes funcionen correctamente en distintas situaciones.
- Creación de una base de datos en la que se guarden y reciban los datos del usuario.

Objetivos sin cumplir :

- Creación del objeto Topo que realizará las acciones de subir y bajar.
- Base de datos **online** a la que podrán acceder los jugadores para comparar sus puntuaciones máximas con las de otros jugadores.
- Añadido de imágenes de modelo para los topos, para que no sean objetos sin color y se vea más real.
- Niveles de dificultad con distintas velocidades de aparición de los topos.

5.2 Propuestas de mejora o ampliaciones futuras

En el punto anterior hago un resumen de los objetivos que he conseguido realizar y también de los que no, cuando comencé este proyecto sin conocimiento alguno sobre las posibilidades que tenía ni sobre que dificultades surgirían me planteé unos objetivos básicos mas basados en ambición por aprender que en mis habilidades o experiencia previas, por lo que a la hora de la verdad el proyecto no terminó teniendo todas las funciones que se esperaban, pese a ello, he conseguido aprender un montón de conceptos y de buenas practicas a la hora de construir algo tan complejo y dependiente como es un videojuego así como también un nuevo lenguaje de programación, C Sharp, utilizado en el motor de Unity.

Seguiré informándome y buscando ayuda en los conceptos en los que he fallado dado que son necesarios y es obligado dominarlos así como de mis habilidades de diseño gráfico, otro de los principales problemas que me encontré mientras realizaba el proyecto era que necesitaba que los topos fueran animados, y tuvieran ciertas caras, expresiones etc que les hiciera mas atractivos pero me encontré que al no tener ningún conocimiento de sprites, diseño de niveles, skins, modelling y otros conceptos necesarios para lo que necesitaba, no pude hacer que se viera como yo lo había pensado en mi cabeza,de hecho los pocos sprites que conseguí hacer con imágenes publicas y gratuitas de topos no realizaban la función buscada, que salieran y se escondiesen a la vez que el juego se ejecutaba y que si les pulsabas se viera como habías acertado por lo que al pasar el tiempo y ver la imposibilidad de seguir con esa idea opté por hacer un diseño mas simplista del juego pero que al menos funcionase y no fuera un amasijo gigante de funciones y métodos inconexos.

En el futuro seguiré trabajando en este proyecto, ya que tengo en él mucho código importante que sé que voy a necesitar en mas proyectos y ademas me sirve para probar todos los nuevos conceptos que voy aprendiendo, cuando aprenda sobre como hacer coleccionables en un juego podré añadirle que caigan frutas por ejemplo de unos arboles superiores y que haya varios tipos de fruta y cada una cambie de alguna manera la jugabilidad, o que haya topos con casco que aguantan varios golpes y cosas por el estilo, evidentemente para ello debo anteriormente estudiar bastante diseño gráfico.

También quiero añadir un menú opciones con una selección del modo de juego, para que el mismo cargue con una velocidad de spawn diferente, dependiendo de lo que elija el jugador, y también distintas puntuaciones dependiendo del nivel que elijas, en fácil contarán menos y en difícil más.

6 Guías

6.1 Guía de uso

Al iniciar el juego te encuentras con un menú principal que consta de tres botones, uno te lleva hacia el juego , otro a la pantalla Scores, que muestra la puntuación máxima así como otros datos de interés y el último te hace salir del juego.

Ilustración 18: Guia de uso menú principal

Si pulsas en Iniciar Partida automáticamente comenzarás a jugar y las lucecitas de los topos comenzarán a cambiar indicándote cuales pueden ser golpeados, si consigues golpearles haciendo click en ellos cuando están verdes verás que su luz se pone roja y sale un texto que dice “Golpeado!” lo cuál hace aumentar tu puntuación, indicada abajo a la derecha

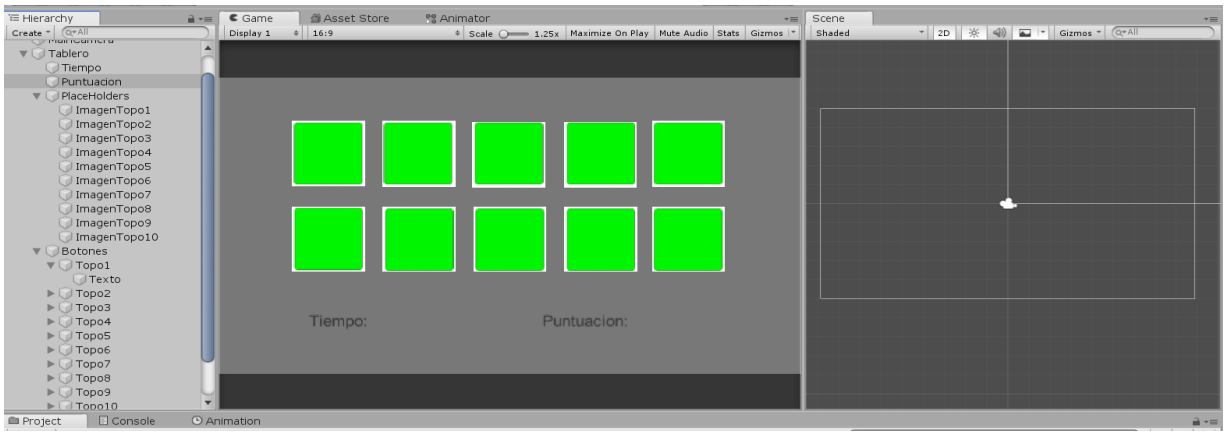


Ilustración 19: Guía de uso escena de juego

Cuando el tiempo haya terminado se mostrará el final de partida y una pantalla con tu información de usuario, golpes, fallos, etc además se te dejará guardar tu puntuación en la base de datos de la aplicación, resetear la partida y volver a jugar, o volver al menú.

Ilustración 20: Guía de uso pantalla con resultados

Si pulsas en Guardar se te llevará a una pantalla en la cual introducir el nombre con el que quieres que se guarde tu puntuación, ahí podrás hacer click en guardar y volver con tu información correctamente salvada.

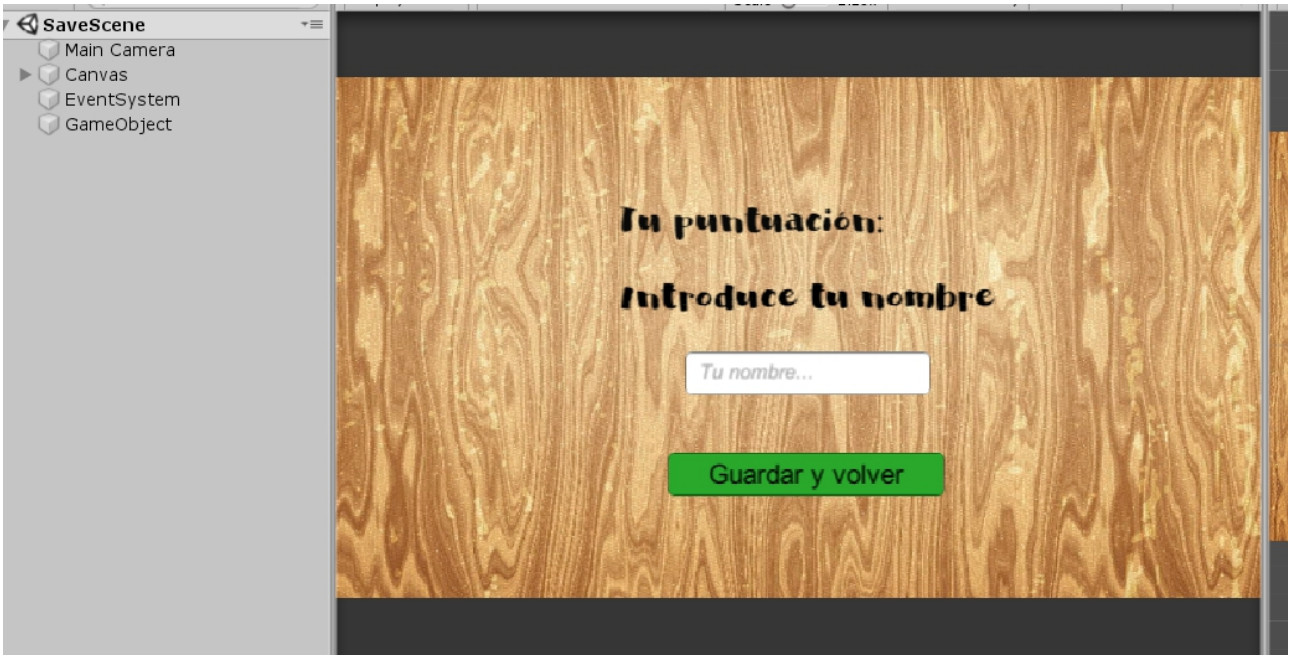


Ilustración 21: Guía de uso Pantalla de guardado

Para comprobar la puntuación guardada en el menú principal podemos pulsar en el botón "Puntuaciones" que nos llevará a una pantalla con nuestro nombre y puntuación máxima así como de los Logros que hayamos conseguido(Aún no implementado).

Ilustración 22: Guía de uso pantalla puntuaciones

7 Referencias bibliográficas

7.1 Curso Udemy Unity2D

(1) Curso de Creación de Videojuegos 2D:
- https://www.udemy.com/crea-un-juego-completo-en-unity-5/
- Canal de Discord propio con mas de 150 alumnos para Q&A.
- Creado por : Juan Gabriel Gomila Salas.
- Profesor universitario, Data Scientist & Game Designer.

7.2 Foro de Unity

(2) Foro de Unity:

- ”forum.unity.com”.

7.3 Referencias multimedia

(3) Como crear un Main Menu en Unity :

- “youtube.com/watch?v=Ygb9j9b4gQU&”

(4) Como utilizar una base de datos Firebase con Unity

- “youtube.com/watch?v=Fz0Sl4tW5O0&”

(5) Como añadir sonido a nuestros juegos

- “youtube.com/watch?v=G1XCadcd0AU”.

(6) Creación de un menú Pause

- “youtube.com/watch?v=JivuXdrIHK0”.

7.4 Páginas Web.

(7) SheWhoCodes tutorial whack a mole:
- “shewhocodes.org/unity-tutorials/building-a-simple-2d-game-in-unity”.
- Utilizado en la creación de la GameScene y SaveScene así como en su comprensión.

8 Índice de ilustraciones