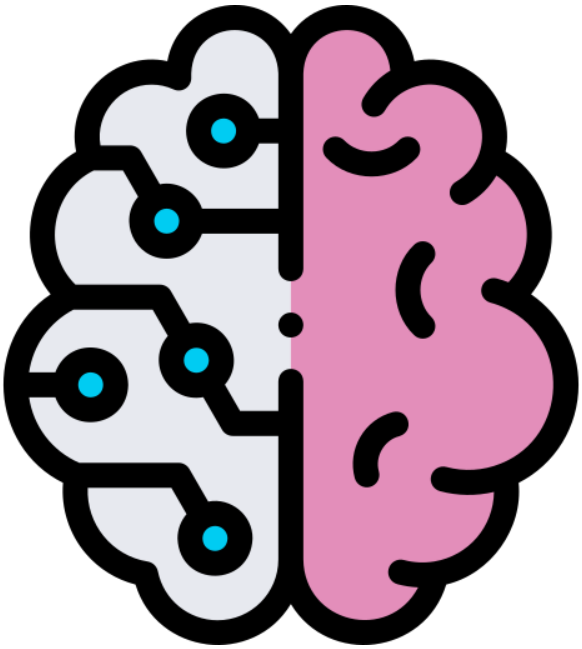
Universidad Nacional Autónoma de México

Facultad de Ingeniería

DetCog



Desarrollador:

Mario Alberto Vásquez Cancino

v. 1.0.0

Índice

Objetivo 2

Plan de Trabajo 2

Alcance del Proyecto 3

Funcionamiento 5

- Instalación y Ejecución 5
- Recursos Físicos 5
  - Tablero 5
  - Cartas 6
- Código Fuente 8
  - Estructura 8
    - Sample Scene 8
    - Directional Light 8
    - ARCamera 9
    - Marcadores 9
    - Display 10
    - Event System 10
  - Código 11
    - Principal.cs 11
    - Tiempo.cs 15
  - Configuración de Unity 17

Costos 21

Conclusión 22

Referencias 22

Objetivo

Tener una aplicación de realidad aumentada que contenga una actividad de categorización de colores enfocada a las personas mayores que sufran de algún deterioro cognitivo, para así estimular su cerebro con estas actividades además de acercarlos a este mundo tecnológico que normalmente desconocen y al adentrarse en él su cerebro se vea más estimulado al experimentarlo.

Plan de Trabajo

Actividades	Fecha de finalización
Buscar y definir los modelos a ocupar.	**6 de noviembre**
Optimizar y adaptar los modelos con sus texturas para la categorización de colores.	**13 de noviembre**
Creación del Layout de la App.	**27 de noviembre**
Creación de las estampas y la detección de las mismas por la app.	**5 de diciembre**
Despliegue de los modelos 3D al detectar las estampas.	**7 de diciembre**
Detección correcta de la categorización de los objetos y funcionalidad del contador de puntaje.	**21 de diciembre**
Periodo de test y posible implementación de los extras.	**6 de enero**

Alcance del Proyecto

Al finalizar este proyecto se debe obtener un aplicación de realidad aumentada que contenga una actividad de categorización orientada a estimular a personas mayores de edad con deterioro cognitivo.

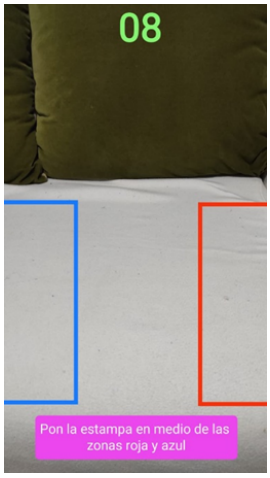
La actividad a realizar es categorización mediante su color, se tendrán dos colores rojo y azul pero la tonalidad va a variar en cada objeto.

A través de la cámara del celular que esté apuntando a una superficie plana (de preferencia a una mesa), se mostrará dos zonas marcadas, una del lado izquierdo (Zona azul) y la otra del lado derecho (Zona roja). En medio de estas estará la zona de “selección” donde el usuario pondrá una pequeña estampa que tendrá un código o imagen que la aplicación detectará para mostrar el objeto 3D que represente con uno de los colores.

Una vez mostrado el objeto el usuario tendrá que poner la estampa en la zona que corresponda para que la app detecte que esté correcta la categorización y contabilice el acierto.

Este proceso se repetirá 10 veces ya que se tendrán en total 10 estampas a categorizar y al completar exitosamente la categorización de todos los objetos se desplegará un mensaje de felicitaciones.

A continuación, se muestra la imagen de referencia.



Como se puede observar se tiene:

- Las zonas de color azul y rojo donde se categorizarán los objetos.
- Zona intermedia donde se pondrán las estampas, así como un pequeño mensaje de instrucciones.
- Un contador de objetos categorizados de manera correcta.

Extras

En caso de que el desarrollo de lo propuesto anteriormente se realice antes de lo planeado se buscará implementar algunas o todas las siguientes propuestas.

- Contador de tiempo para tener una forma de medir la habilidad del usuario.
- Otra actividad de categorización
    - Se tendrá un menú donde se elija que tipo de categorización se desea realizar.
    - La nueva categorización será por tamaño, objeto grande o chico.
    - Se buscará ocupar las mismas estampas de la categorización de color para evitar tener muchas estampas y la aplicación este mas optimizada, en caso de que sea de no poder ocupar las mismas estampas, esta categorización tendrá su propio juego de estampitas con otros modelos.

El proyecto finalizado con su documentación, archivo ejecutable (apk) y binarios deberá ser entregado a más tardar el viernes 6 de enero de 2023.

Limitaciones

Debido a que se están ocupando nuevas tecnologías para la realidad aumentada, así como la constante actualización de las mismas puede que la dinámica de para categorización de color cambie respecto a lo planeado, pero la escencia y la simpleza de la misma se mantendrá intacta. También a que esta aplicación está enfocada hacia adultos mayores se buscará un diseño que sea amigable y no complicado para ellos por lo que el resultado final del diseño de las cartas y aplicación puede cambiar así como la creación de otros elementos extras.

Funcionamiento

Instalación y Ejecución

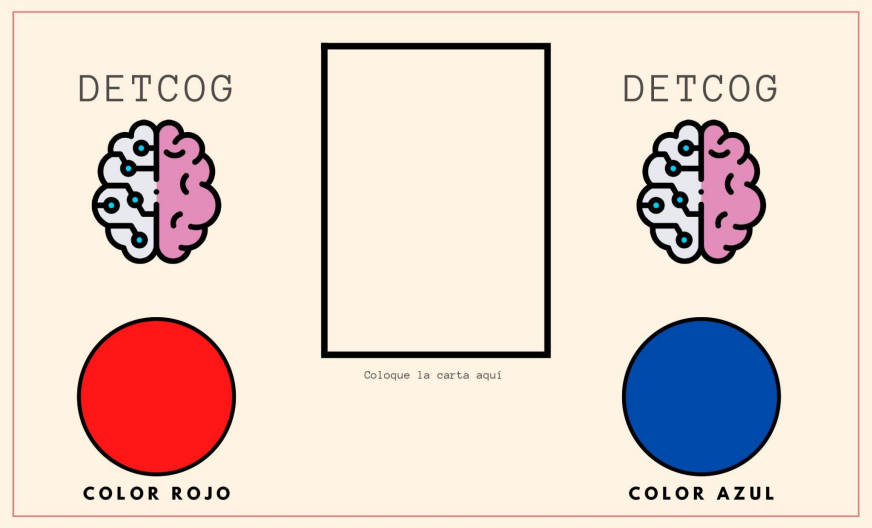
Para el uso de la aplicación véase el Manual de Usuario.

Recursos Físicos

Para el uso de la aplicación necesitaremos los siguientes elementos:

Tablero

Elemento principal de la aplicación ya que este proporciona una guía al usuario para poner las cartas así como para indicarle a la aplicación que opcion se eligio.



El tablero consta de un área central donde se colocarán las cartas así como dos círculos (azul y rojo) para los virtual buttons y que los usuarios puedan ocupar al realizar su selección.

Cartas

Las cartas o marcadores nos servirán para que la aplicación las detecte y muestre diferentes modelos para que el usuario las categorize por color.

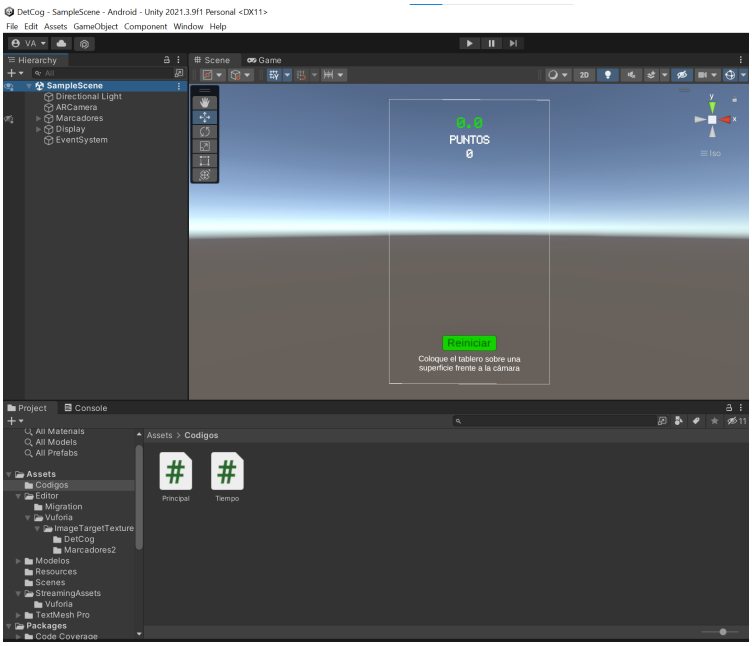
Se eligieron estas cartas ya que aparte proporciona imágenes sencillas y didácticas para el usuario final, también para que la aplicación las pueda detectarlas y diferenciarlas sin problemas.

** **Todos estos recursos se encuentran al final del Manual de Usuario.**

Código Fuente

Estructura

El proyecto se realizó con Unity y Vuforia por lo que encontramos esta estructura al abrirlo.

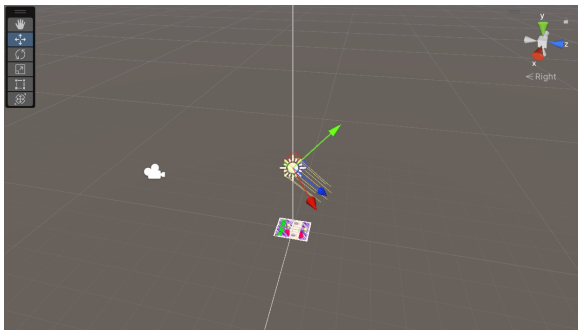


Sample Scene

Escena principal que contiene todo los recursos del proyectos

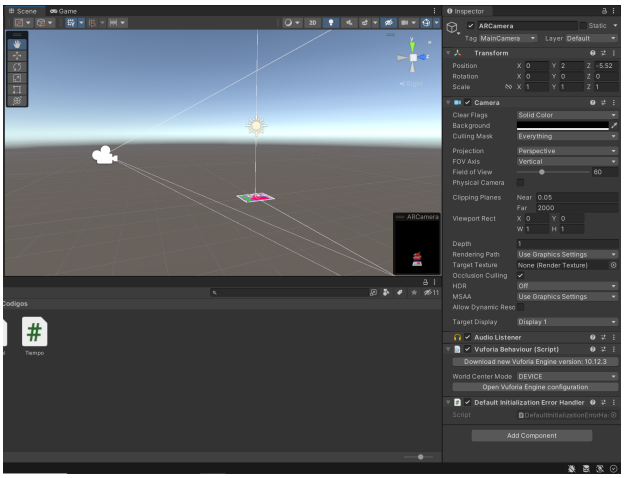
Directional Light

Configuración de la luz de la escena.



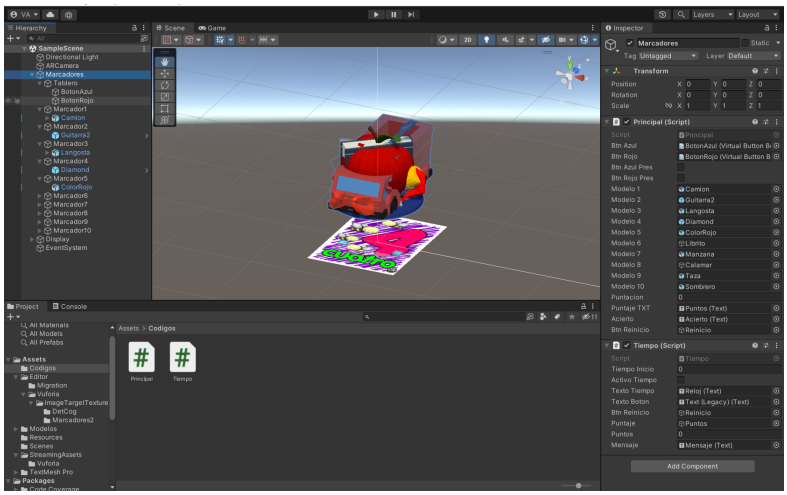
ARCamera

Cámara especial de Vuforia para visualizar los elementos de realidad virtual, está enfoca a los marcadores a ocupar. Dentro de su configuración le indicamos que el centro del mundo (la escena) es el dispositivo.



Marcadores

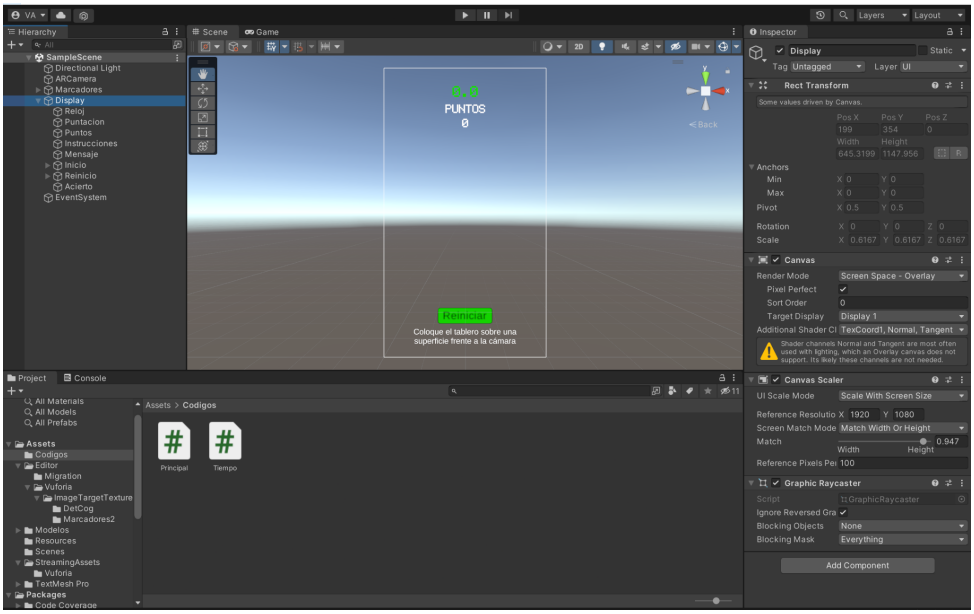
Este es un objeto vacío que contiene todos los marcadores con sus modelos a desplegar, el más destacable es el tablero ya que contiene los Virtual Buttons para seleccionar los colores así como es el marcador principal para usar la aplicación.



También se puede considerar como el elemento principal del proyecto ya que a él están asociados los códigos con las funcionalidades de la app.

Display

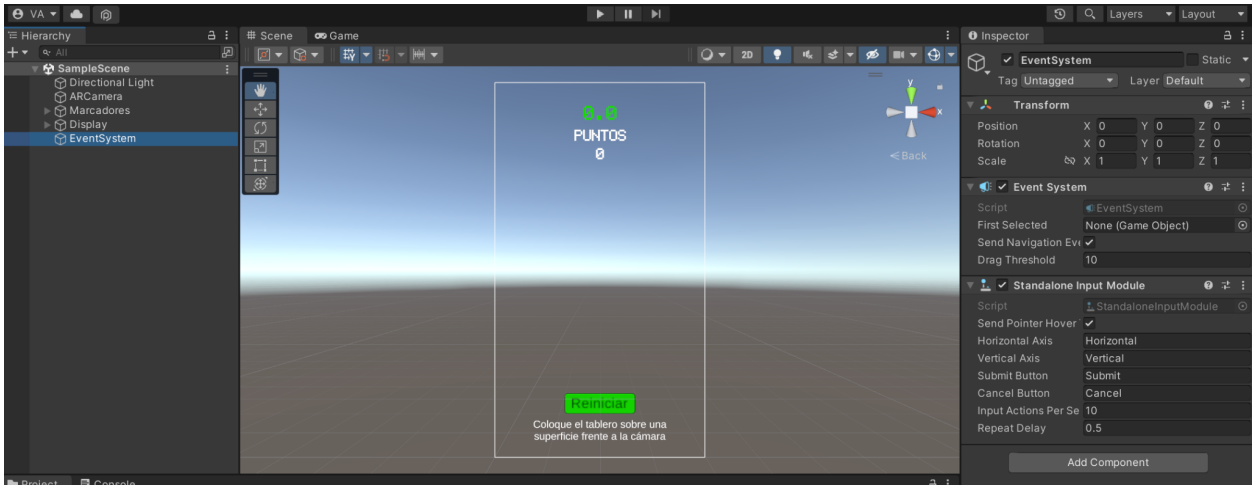
Es la interfaz que mostrará la aplicación, contiene todos los mensajes, cajas de textos y botones que ocupará la aplicación.



Aquí lo importante es el parámetro “UI Scale Mode” ya que se configuró como “Scale With Screen Size” para que la interfaz se adapte a toda pantalla de cualquier dispositivo.

Event System

Contiene los elementos básicos para detectar los eventos físicos que suceden en un dispositivo móvil como la orientación, el salir, regresar, etc.



Código

Principal.cs

Es el código principal de la App ya que este maneja la detección de los marcadores y puntaje.

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.UI;
using Vuforia;

Script de Unity (1 referencia de recurso) | 0 referencias
public class Principal : MonoBehaviour
{
    // Vitual Buttons
    public VirtualButtonBehaviour btnAzul;
    public VirtualButtonBehaviour btnRojo;
    public bool btnAzulPres;
    public bool btnRojoPres;

    // Modelos
    public GameObject modelo1;
    public GameObject modelo2;
    public GameObject modelo3;
    public GameObject modelo4;
    public GameObject modelo5;
    public GameObject modelo6;
    public GameObject modelo7;
    public GameObject modelo8;
    public GameObject modelo9;
    public GameObject modelo10;

    // Juego
    public int puntacion;
    bool [] conteo = new bool[10];
    public Text puntajeTXT;
    public Text acierto;
    public GameObject btnReinicio;

    // Start is called before the first frame update
    Mensaje de Unity | 0 referencias
    void Start()
    {

        btnAzul.RegisterOnButtonPressed(BotonAzulPresionado);
        btnAzul.RegisterOnButtonReleased(BotonAzulSuelto);

        btnRojo.RegisterOnButtonPressed(BotonRojoPresionado);
        btnRojo.RegisterOnButtonReleased(BotonRojoSuelto);

        btnAzulPres = false;
        btnRojoPres = false;
        puntacion = 0;
    }
```

De inicio tenemos la declaración de los Virtual Buttons, variables que nos indican si se presionó un botón o no, la declaración de los modelos de los marcadores, un arreglo de booleanos donde cada casilla representa si una tarjeta ya fue categorizada (sin importar si fue correcto o no) y todo lo necesario para manejar el puntaje.

En la función de start configuramos las funciones de presionado y no presionado de los Virtual Buttons, así como el estado de inicio de la puntuación y variables para saber si el botón se presionó o no.

```
// Update is called once per frame
Mensaje de Unity | 0 referencias
void Update()
{
    juego();
    juegoTermino();
}

1 referencia
public void BotonAzulPresionado(VirtualButtonBehaviour vb)
{
    btnAzulPres = true;
    //mensaje.text = "Color Azul";
}

1 referencia
public void BotonAzulSuelto(VirtualButtonBehaviour vb)
{
    btnAzulPres = false;
    //mensaje.text = "";
}

1 referencia
public void BotonRojoPresionado(VirtualButtonBehaviour vb)
{
    btnRojoPres = true;
    //mensaje.text = "Color Rojo";
}


public void BotonRojoSuelto(VirtualButtonBehaviour vb)
{
    btnRojoPres = false;
    //mensaje.text = "";
}
```

En la función “Update” tenemos que siempre se repiten la función “juego” y “juegoTerminado”.

También tenemos las declaraciones de las funciones cuando se presionan los Virtual Buttons, donde se cambia a “true” sus respectivas variables si el botón es presionado y cambia a “false” si los botones no son presionados.

```
public void juego()
{
    // ---------- Marcador 1 ----------
    if (modelo1.activeSelf && !conteo[0] && btnRojoPres)
    {
        puntacion++;
        puntajeTXT.text = puntacion.ToString();
        conteo[0] = true;
        acierto.text = "Correcto";
    }

    if (modelo1.activeSelf && !conteo[0] && btnAzulPres)
    {
        conteo[0] = true;
        acierto.text = "Incorrecto";
    }

    // ---------- Marcador 2 ----------
    if (modelo2.activeSelf && !conteo[1] && btnAzulPres)
    {
        puntacion++;
        puntajeTXT.text = puntacion.ToString();
        conteo[1] = true;
        acierto.text = "Correcto";
    }

    if (modelo2.activeSelf && !conteo[1] && btnRojoPres)
    {
        conteo[1] = true;
        acierto.text = "Incorrecto";
    }

    // ---------- Marcador 3 ----------
    if (modelo3.activeSelf && !conteo[2] && btnRojoPres)
    {
        puntacion++;
        puntajeTXT.text = puntacion.ToString();
        conteo[2] = true;
        acierto.text = "Correcto";
    }

    if (modelo3.activeSelf && !conteo[2] && btnAzulPres)
    {
        conteo[2] = true;
        acierto.text = "Incorrecto";
    }
```

```
    // ---------- Marcador 4 ----------
    if (modelo4.activeSelf && !conteo[3] && btnAzulPres)
    {
        puntacion++;
        puntajeTXT.text = puntacion.ToString();
        conteo[3] = true;
        acierto.text = "Correcto";
    }

    if (modelo4.activeSelf && !conteo[3] && btnRojoPres)
    {
        conteo[3] = true;
        acierto.text = "Incorrecto";
    }

    // ---------- Marcador 5 ----------
    if (modelo5.activeSelf && !conteo[4] && btnRojoPres)
    {
        puntacion++;
        puntajeTXT.text = puntacion.ToString();
        conteo[4] = true;
        acierto.text = "Correcto";
    }

    if (modelo5.activeSelf && !conteo[4] && btnAzulPres)
    {
        conteo[4] = true;
        acierto.text = "Incorrecto";
    }

    // ---------- Marcador 6 ----------
    if (modelo6.activeSelf && !conteo[5] && btnAzulPres)
    {
        puntacion++;
        puntajeTXT.text = puntacion.ToString();
        conteo[5] = true;
        acierto.text = "Correcto";
    }

    if (modelo6.activeSelf && !conteo[5] && btnRojoPres)
    {
        conteo[5] = true;
        acierto.text = "Incorrecto";
    }
```

```
    // ---------- Marcador 7 ----------
    if (modelo7.activeSelf && !conteo[6] && btnRojoPres)
    {
        puntacion++;
        puntajeTXT.text = puntacion.ToString();
        conteo[6] = true;
        acierto.text = "Correcto";
    }

    if (modelo7.activeSelf && !conteo[6] && btnAzulPres)
    {
        conteo[6] = true;
        acierto.text = "Incorrecto";
    }

    // ---------- Marcador 8 ----------
    if (modelo8.activeSelf && !conteo[7] && btnAzulPres)
    {
        puntacion++;
        puntajeTXT.text = puntacion.ToString();
        conteo[7] = true;
        acierto.text = "Correcto";
    }

    if (modelo8.activeSelf && !conteo[7] && btnRojoPres)
    {
        conteo[7] = true;
        acierto.text = "Incorrecto";
    }

    // ---------- Marcador 9 ----------
    if (modelo9.activeSelf && !conteo[8] && btnAzulPres)
    {
        puntacion++;
        puntajeTXT.text = puntacion.ToString();
        conteo[8] = true;
        acierto.text = "Correcto";
    }

    if (modelo9.activeSelf && !conteo[8] && btnAzulPres)
    {
        conteo[8] = true;
        acierto.text = "Incorrecto";
    }
```

```
    // ---------- Marcador 10 ----------
    if (modelo10.activeSelf && !conteo[9] && btnAzulPres)
    {
        puntacion++;
        puntajeTXT.text = puntacion.ToString();
        conteo[9] = true;
        acierto.text = "Correcto";
    }

    if (modelo10.activeSelf && !conteo[9] && btnRojoPres)
    {
        conteo[9] = true;
        acierto.text = "Incorrecto";
    }

}
```

En la función “juego” encontramos para marcador que si el modelo del marcador se activa, no está marcada su casilla en el arreglo como seleccionada y el botón del color del modelo se presionó, entonces aumenta la puntuación, se muestra en pantalla, se marca su casilla y se muestra el mensaje de “Correcto”. En cambio si el modelo del marcador se activa, no está marcada su casilla en el arreglo como seleccionada y si se presionó el botón incorrecto del color del modelo, se marca su casilla y se muestra el mensaje de “Incorrecto”.

```
1 referencia
public void juegoTermino()
{
    int listos = 0;
    for (int i = 0; i < conteo.Length; i++)
    {
        if (conteo[i])
            listos++;
    }

    if (listos == 10)
        btnReinicio.SetActive(true);
}

0 referencias
public void juegoReinicio()
{
    puntacion = 0;
    puntajeTXT.text = puntacion.ToString();
    btnAzulPres = false;
    btnRojoPres = false;
    acierto.text = "";

    for (int i = 0; i < conteo.Length; i++)
        conteo[i] = false;

    modelo1.SetActive(false);
    modelo2.SetActive(false);
    modelo3.SetActive(false);
    modelo4.SetActive(false);
    modelo5.SetActive(false);
    modelo6.SetActive(false);
    modelo7.SetActive(false);
    modelo8.SetActive(false);
    modelo9.SetActive(false);
    modelo10.SetActive(false);
}
```

Por último la función “juegoTerminado” se encarga de recorrer el arreglo y contar las marcadas para al final checar si todas están marcadas, si todas están marcadas activa el botón de “Reiniciar” lo que indica que acabó el juego, si no el juego continúa.

Por otra parte la función “juegoReinicio” es la funcionalidad del botón “Reiniciar” ya que restablece los valores de todas las variables a su estado original para volver a iniciar el juego.

Tiempo.cs

Este código es el secundario ya que maneja el contador de tiempo así como los botones que lo controlan.

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.UI;
Script de Unity (1 referencia de recurso) | 0 referencias
public class Tiempo : MonoBehaviour
{
    public float tiempoInicio;
    public bool activoTiempo;
    public Text textoTiempo;
    public Text textoBoton;
    public GameObject btnReinicio;

    public GameObject puntaje;
    public string puntos;

    public Text mensaje;

    // Start is called before the first frame update
    // Start is called before the first frame update
    Mensaje de Unity | 0 referencias
    void Start()
    {
        activoTiempo = false;
        textoTiempo.text = tiempoInicio.ToString("F2");
        btnReinicio.SetActive(false);
    }
```

Tenemos la declaración de variables que controlan el contador de tiempo y los textos para mostrarlo en la pantalla.

```
// Update is called once per frame
Mensaje de Unity | 0 referencias
void Update()
{
    if(activoTiempo)
    {
        tiempoInicio += Time.deltaTime;
        textoTiempo.text = tiempoInicio.ToString("F2");
    }


    if (btnReinicio.activeSelf)
    {
        activoTiempo = false;
        mensaje.text = "Juego\nTerminado";
    }
}
```

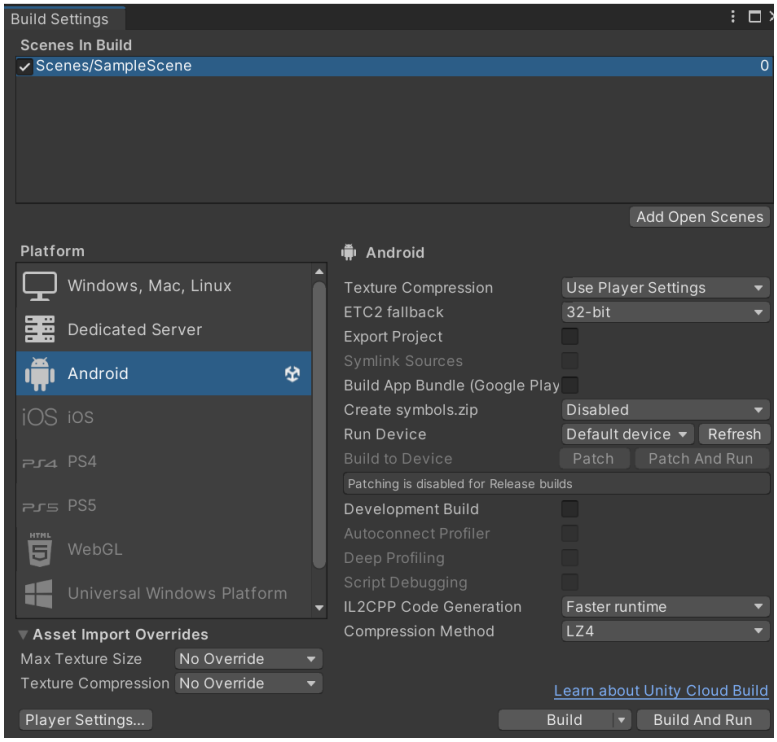
La función “Update” detecta si la variable que controla tiempo es “true”, si es así empieza a contar el tiempo y lo muestra en pantalla, luego revisa si el botón “Reiniciar” esta activado (ya que esto indica que el juego acabó), por lo que si es verdadero entonces para la variable del contador cambia su valor a “false” y manda el mensaje de “Juego Terminado”.

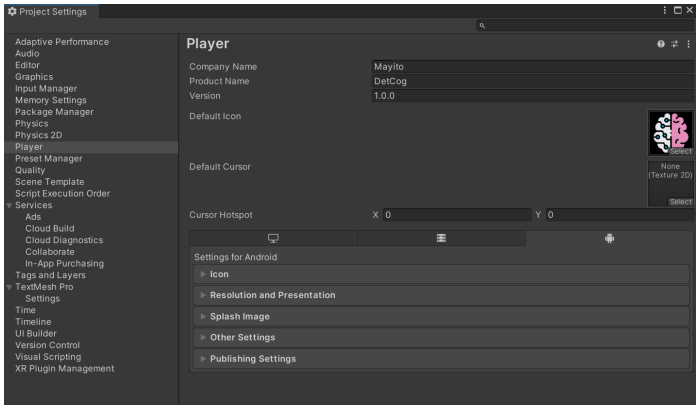
```
0 referencias
public void botonInicio()
{
    activoTiempo = true;
    /*activoTiempo = !activoTiempo;
    textoBoton.text = activoTiempo ? "Pausar" : "Iniciar";*/
}

0 referencias
public void botonReinicio()
{
    tiempoInicio = 0.0f;
    activoTiempo = true;
    mensaje.text = "";
}
```

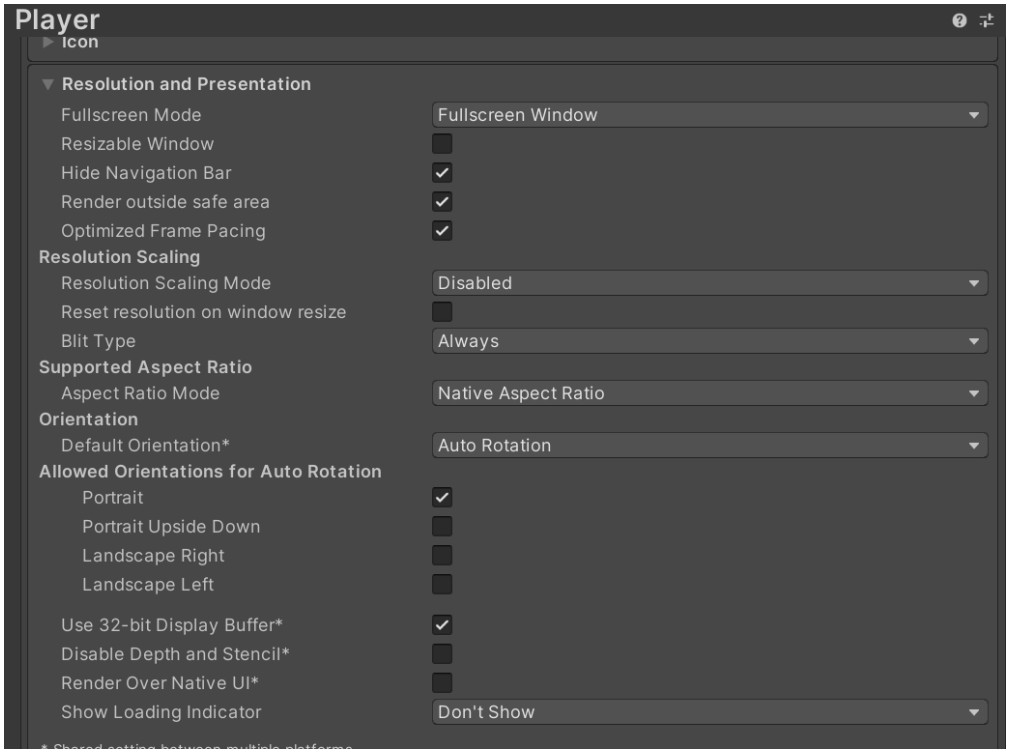
Finalmente están las funciones de los botones, la función del botón “Iniciar” (botonInicio) cambiar el valor de la variable que cuenta el tiempo a “true” cuando este se presiona. Por su parte la del botón “Reiniciar” (botonReiniciar) restablece los valores del contador, la variable del contador y el mensaje de “Juego Terminado” lo quita.

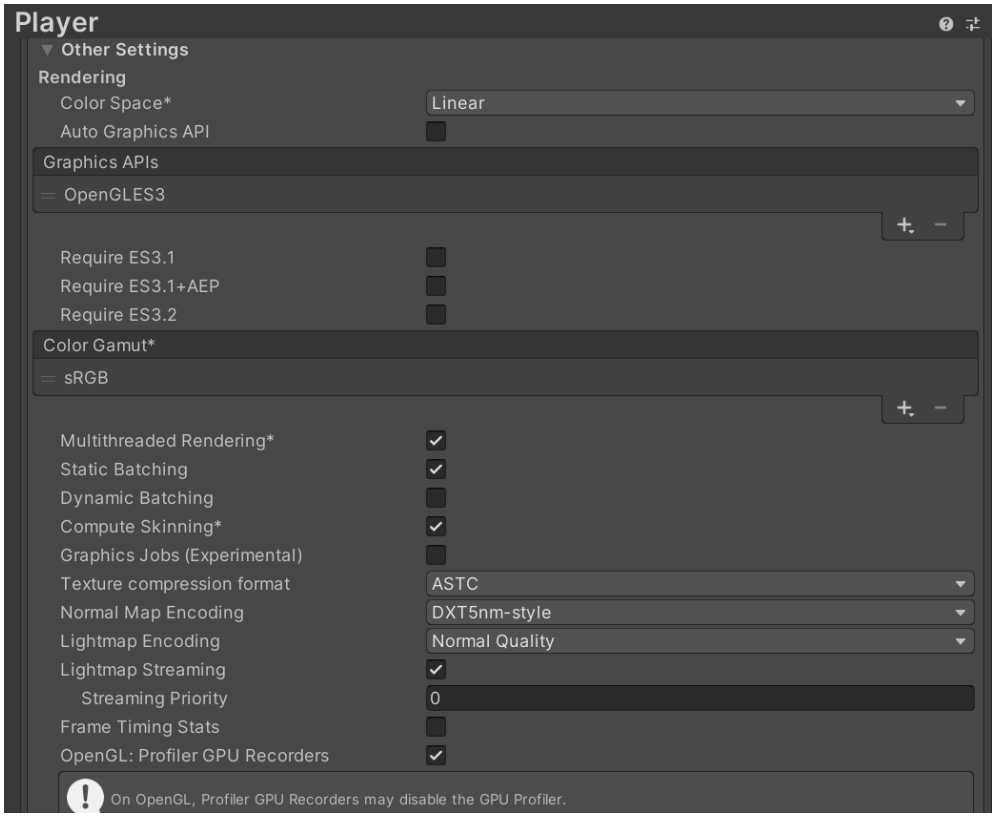
Configuración de Unity

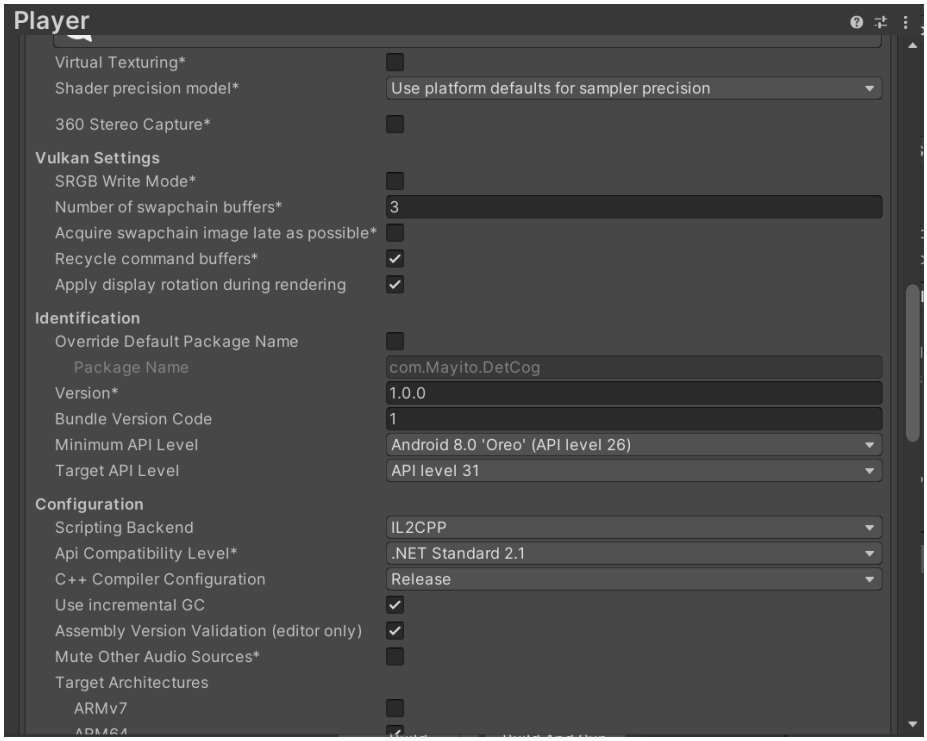
A continuación se muestran capturas de como se configuró Unity para hacer la aplicación.

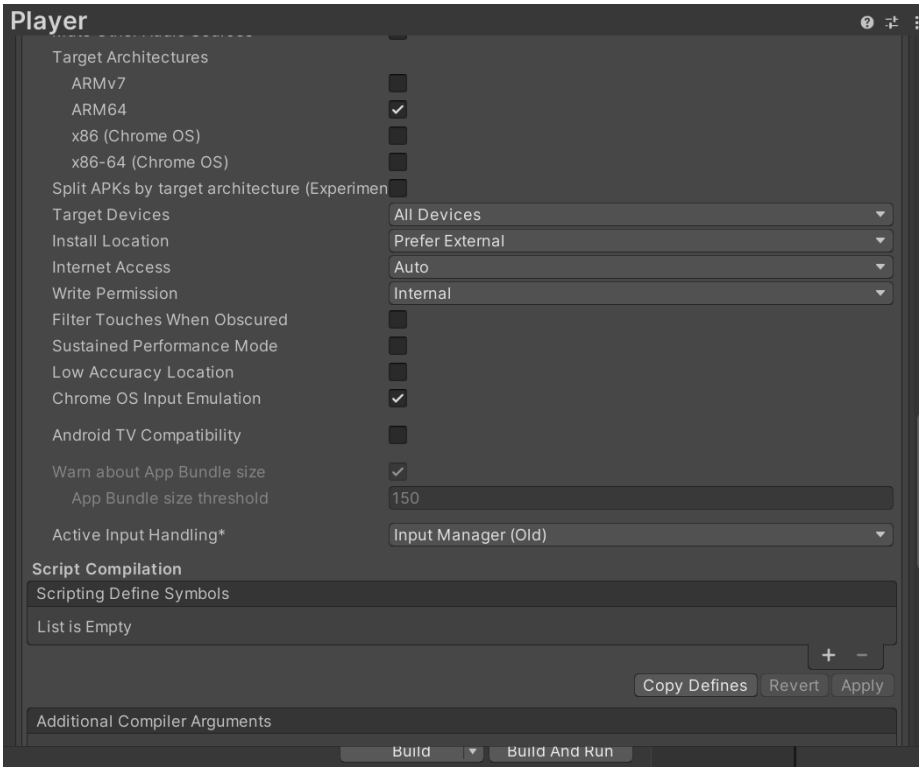


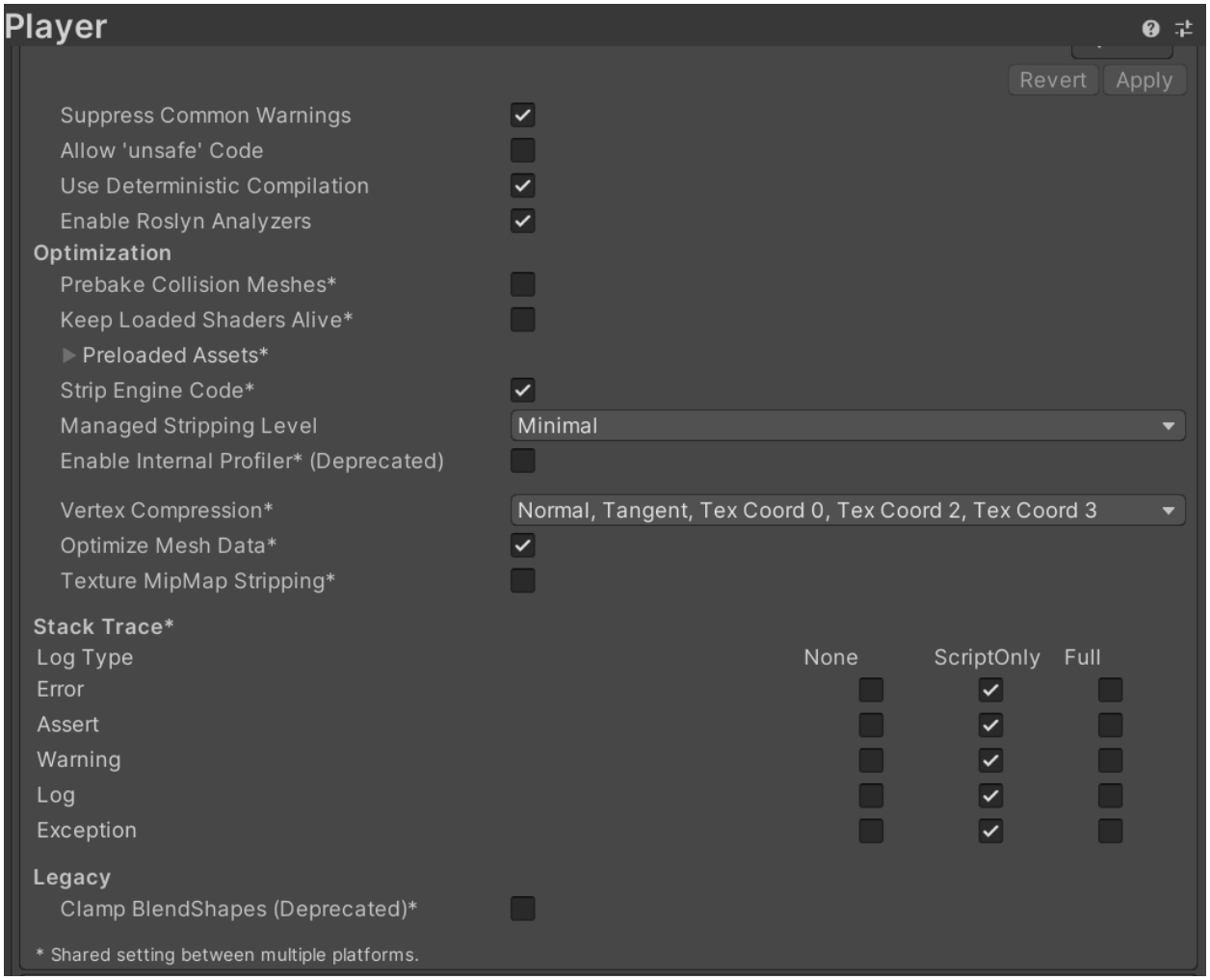



Player
Icon
Resolution and Presentation
Fullscreen Mode
Fullscreen Window
Resizable Window
Hide Navigation Bar
Render outside safe area
Optimized Frame Pacing
Resolution Scaling
Resolution Scaling Mode
Disabled
Reset resolution on window resize
Blit Type
Always
Supported Aspect Ratio
Aspect Ratio Mode
Native Aspect Ratio
Orientation
Default Orientation*
Auto Rotation
Allowed Orientations for Auto Rotation
Portrait
Portrait Upside Down
Landscape Right
Landscape Left
Use 32-bit Display Buffer*
Disable Depth and Stencil*
Render Over Native UI*
Show Loading Indicator
Don't Show


Player
Other Settings
Rendering
Color Space*
Linear
Auto Graphics API
Graphics APIs
OpenGLES3
Require ES3.1
Require ES3.1+AEP
Require ES3.2
Color Gamut*
sRGB
Multithreaded Rendering*
Static Batching
Dynamic Batching
Compute Skinning*
Graphics Jobs (Experimental)
Texture compression format
ASTC
Normal Map Encoding
DXT5nm-style
Lightmap Encoding
Normal Quality
Lightmap Streaming
Streaming Priority
0
Frame Timing Stats
OpenGL: Profiler GPU Recorders
On OpenGL, Profiler GPU Recorders may disable the GPU Profiler.


Player
Virtual Texturing*
Shader precision model*
Use platform defaults for sampler precision
360 Stereo Capture*
Vulkan Settings
SRGB Write Mode*
Number of swapchain buffers*
3
Acquire swapchain image late as possible*
Recycle command buffers*
Apply display rotation during rendering
Identification
Override Default Package Name
Package Name
com.Mayito.DetCog
Version*
1.0.0
Bundle Version Code
1
Minimum API Level
Android 8.0 'Oreo' (API level 26)
Target API Level
API level 31
Configuration
Scripting Backend
IL2CPP
Api Compatibility Level*
.NET Standard 2.1
C++ Compiler Configuration
Release
Use incremental GC
Assembly Version Validation (editor only)
Mute Other Audio Sources*
Target Architectures
ARMv7


Player
Target Architectures
ARMv7
ARM64
x86 (Chrome OS)
x86-64 (Chrome OS)
Split APKs by target architecture (Experimen
Target Devices
All Devices
Install Location
Prefer External
Internet Access
Auto
Write Permission
Internal
Filter Touches When Obscured
Sustained Performance Mode
Low Accuracy Location
Chrome OS Input Emulation
Android TV Compatibility
Warn about App Bundle size
App Bundle size threshold
150
Active Input Handling*
Input Manager (Old)
Script Compilation
Scripting Define Symbols
List is Empty
Copy Defines
Revert
Apply
Additional Compiler Arguments
Build
Build And Run


Player
Revert
Apply
Suppress Common Warnings
Allow 'unsafe' Code
Use Deterministic Compilation
Enable Roslyn Analyzers
Optimization
Prebake Collision Meshes*
Keep Loaded Shaders Alive*
Preloaded Assets*
Strip Engine Code*
Managed Stripping Level
Minimal
Enable Internal Profiler* (Deprecated)
Vertex Compression*
Normal, Tangent, Tex Coord 0, Tex Coord 2, Tex Coord 3
Optimize Mesh Data*
Texture MipMap Stripping*
Stack Trace*
Log Type
None
ScriptOnly
Full
Error
Assert
Warning
Log
Exception
Legacy
Clamp BlendShapes (Deprecated)*
* Shared setting between multiple platforms.

Costos

La siguiente tabla muestra los costos estimados del desarrollo del prototipo realizado por una persona por un periodo de dos meses aproximadamente.

Recurso	Costo Mensual	Costo Total
Desarrollador	$15,000.00 MXN	$30,000.00 MXN
Licencia Unity	$3,700.00 MXN	$7,400.00 MXN
GIMP	Gratuito	Gratuito
Licencia 3ds Max	$3,087.00 MXN	$3,087.00 MXN
Modelos 3D	No Aplica	$2,000.00 MXN
Computadora de escritorio	No Aplica	$30,000.00 MXN
Teléfono Xiaomi Redmi Note 9 PRO o similar	No Aplica	$ 4,500.00 MXN
Internet	$400.00 MXN	$800.00 MXN
Total		**77,787.00 MXN**

Conclusión

Ya con la Demo funcional y despues de probarla puedo concluir que el desarrollo fue algo que me agrado demasiado ya que fueron tecnologías nuevas para mi y el hecho de tener un enfoque real y a la vez que tiene el potencial de ayudar a demas personas me dio mucha motivación para hacerlo lo mejor posible y tener un buen producto final. También me percate de todos los problemas que pueden presentarse cuando no conoces ni las herramientas ni tecnologías que vas a ocupar ya que el resultado final a la idea original son diferentes, comparten la misma lógica pero el proceso es diferente, y el más claro que me tocó solucionar fue que los Virtual Buttons se accionaba con enfocarlos a la cámara y no presionandolos como se suponen que están diseñados, por lo que tuve que cambiar esa mecánica para terminar la Demo, pero pienso en un futuro retomarlo ya que se me ocurrieron varias ideas para solucionarlo o cambiarlo por una mecánica más sencilla.

Referencias

Sueldo: Junior (Agosto, 2022). (s. f.). Glassdoor. Recuperado 23 de agosto de 2022, de https://www.glassdoor.com.mx/Sueldos/junior-sueldo-SRCH_KO0,6.html

¿Cómo ayuda la Realidad virtual a mayores? (2018, 5 noviembre). Solera Asistencial. Recuperado 16 de octubre de 2022, de https://www.soleraasistencial.es/ayuda-la-realidad-virtual-mayores/

https://www.med.ufro.cl/departamentoenfermeria/images/descargas/1.Cuadernillo-estimulacion-cog-fis.pdf

https://fiapam.org/wp-content/uploads/2013/07/muestra.pdf

https://store.unity.com/es/products/unity-pro

https://www.glassdoor.com.mx/Sueldos/encargado-de-realidad-aumentada-y-realidad-virtual-sueldoSRCH_KO0,50.htm#:~:text=%C2%BFCu%C3%A1nto%20gana%20un%20Encargado%20De,de%20MXN%2415%2C176%20en%20M%C3%A9xico.

https://www.turbosquid.com/

https://www.autodesk.mx/products/3dsmax/overview?panel=buy&AID=12741901&PID=8299320&SID=jkp_Cj0KCQjwnvOaBhDTARIsAJf8eVNb4DDnzkD3MbPP4JD_6pGujvyNjZoBE16SR-0agldzWKU7q5nXAaAs56EALw_wcB&cjevent=666b0ad2580d11ed80cbb0790a1c0e10&mktvar002=afc_mx_deeplink&affname=8299320_12741901&term=1-YEAR&tab=subscription&plc=3DSMAX