



Universidad Nacional Autónoma de México

Facultad de Ingeniería



Manual técnico: Build My PC - AR.

Asignatura: Temas selectos de ingeniería en computación III.

Alumno: Ponce Soriano Armando

Grupo: 01

Semestre 2023-2

Profesor: Ing. Arturo Pérez De La Cruz

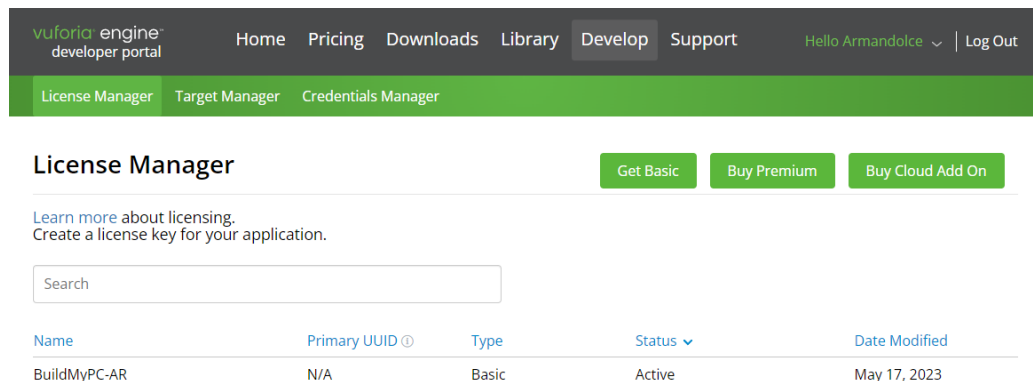
Índice

1. Pasos Previos	3
1.1 Vuforia Engine	3
1.2 Creación de proyecto en Unity e instalación de Vuforia Engine	4
1.3 Otros paquetes relevantes	5
2. Desarrollo de Build My PC – AR.....	7
2.1 Escena: ComponentVisualizer	7
2.1.1 ScriptableObjects	10
2.1.2 Solución al problema de los Múltiples modelos y detalles adicionales	10
2.2 Escena: BuildMyPC-AR	14
2.2.1 UI Manager.....	15
2.2.2 App Manager.....	17
2.2.3 Data Manager.....	17
2.2.4 AR Interaction Manager	18
2.3 Escena: MainMenu.....	19
2.4 Estructura general de los archivos	19
3. Reporte Financiero y esquemas de Tiempo.....	20
4. Funcionamiento de la App	21
5. Referencias	23
Modelos 3D y recursos	23

1. Pasos Previos

1.1 Vuforia Engine

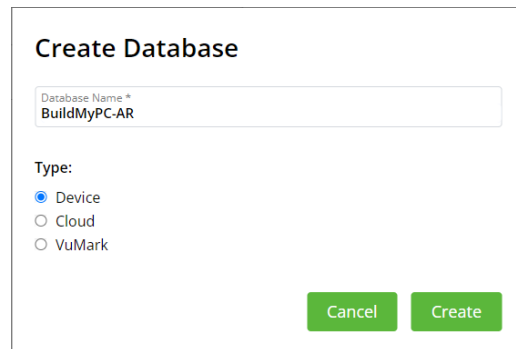
Antes de comenzar a crear nuestra aplicación es importante primero encargarnos de otros pequeños asuntos, lo primero será crear una nueva licencia de vuforia engine para nuestro proyecto.



The screenshot shows the Vuforia Engine Developer Portal. The top navigation bar includes links for Home, Pricing, Downloads, Library, Develop, and Support. A user is logged in as 'Hello Armandolce'. Below the navigation bar, there are tabs for License Manager, Target Manager, and Credentials Manager. The License Manager page has buttons for 'Get Basic', 'Buy Premium', and 'Buy Cloud Add On'. A search bar is present, and a table lists the current license.

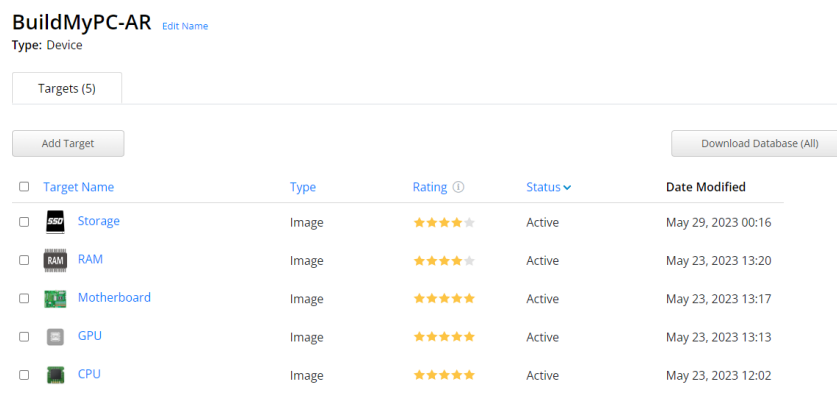
Name	Primary UUID	Type	Status	Date Modified
BuildMyPC-AR	N/A	Basic	Active	May 17, 2023

De igual manera, crearemos una Base de datos nueva para los diferentes Image Target que usaremos en visualización de nuestros componentes.



The screenshot shows the 'Create Database' form. It has a text input for 'Database Name' with the value 'BuildMyPC-AR'. Below it, there are radio buttons for 'Type': 'Device' (selected), 'Cloud', and 'VuMark'. At the bottom right, there are 'Cancel' and 'Create' buttons.

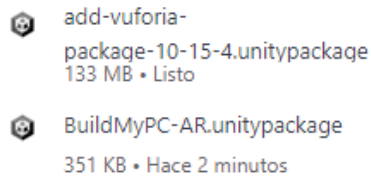
Los targets finales serán los siguientes:



The screenshot shows the 'BuildMyPC-AR' database page. It displays the database name and type (Device). Below, there is a table of targets. The table has columns for Target Name, Type, Rating, Status, and Date Modified. There are buttons for 'Add Target' and 'Download Database (All)'.

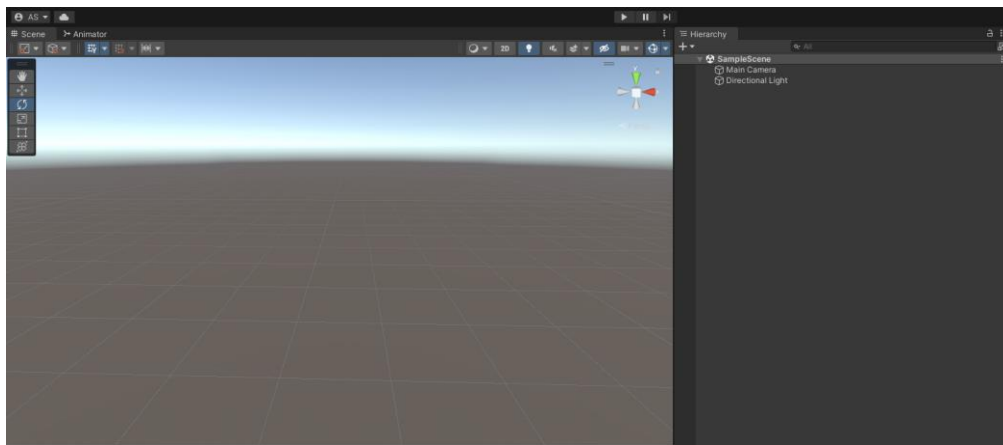
Target Name	Type	Rating	Status	Date Modified
Storage	Image	★★★★★	Active	May 29, 2023 00:16
RAM	Image	★★★★★	Active	May 23, 2023 13:20
Motherboard	Image	★★★★★	Active	May 23, 2023 13:17
GPU	Image	★★★★★	Active	May 23, 2023 13:13
CPU	Image	★★★★★	Active	May 23, 2023 12:02

Una vez realizado todo lo necesario en la plataforma descargamos la base de datos y el paquete de vuforia de la pagina

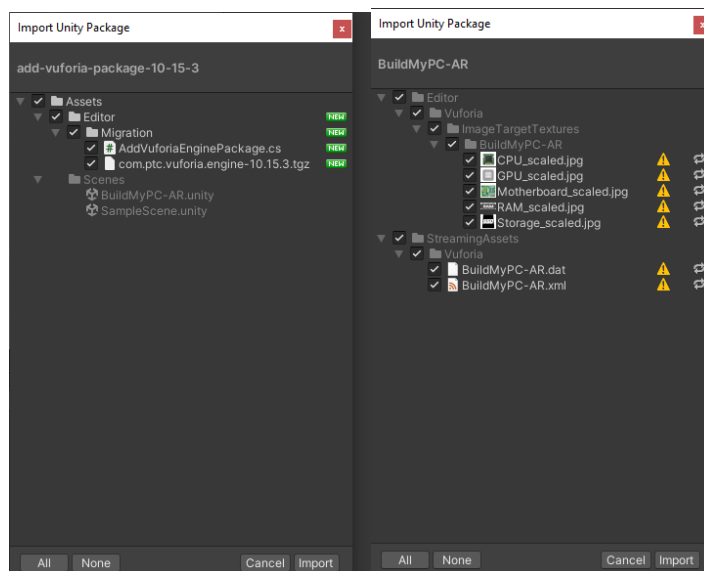


1.2 Creación de proyecto en Unity e instalación de Vuforia Engine

Una vez resuelto el asunto de vuforia engine y creado nuestro proyecto en **Unity Hub** usando la plantilla **3D**, nos encontraremos con el editor de la siguiente manera



Ahora, arrastramos los dos paquetes descargados de Vuforia e importamos a Unity los mismos



A partir de aquí podemos comenzar a crear nuestra aplicación, pero antes de eso revisaremos algunos otros paquetes relevantes para el proyecto.

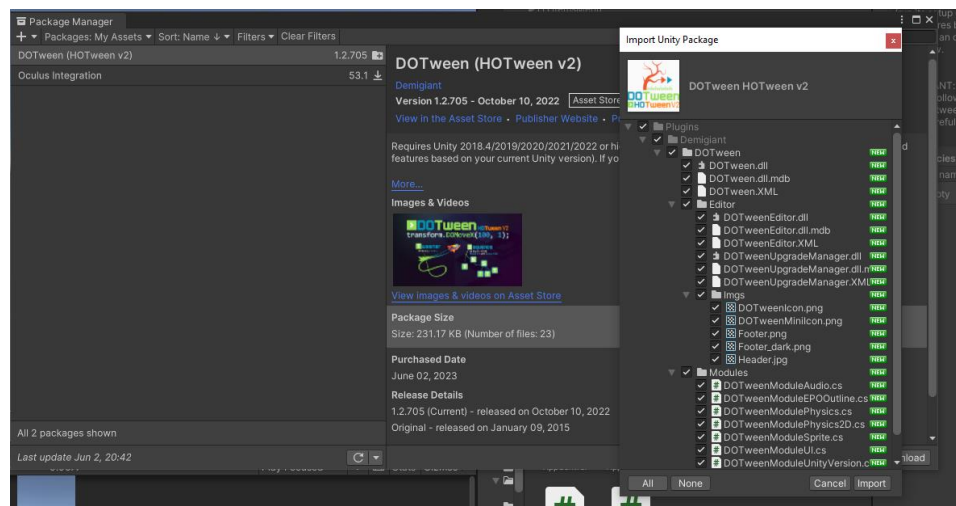
1.3 Otros paquetes relevantes

Para la realización del proyecto fue necesario hacer uso de otros paquetes adicionales como lo son:

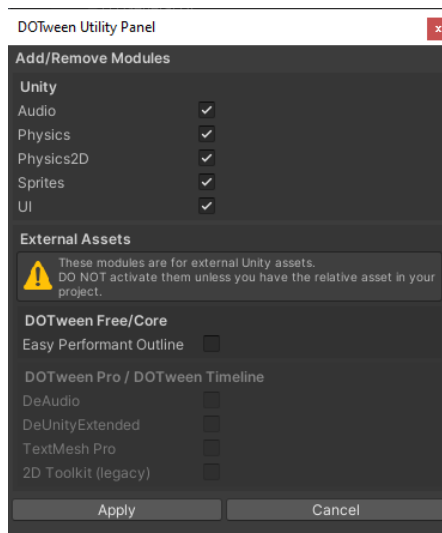
- **AR Foundation**
- **XR Plug-In Management**
- **ARCore XR Plugin**
- **DOTween**

A continuación, capturas de pantalla relevantes de la instalación de algunos de estos paquetes:

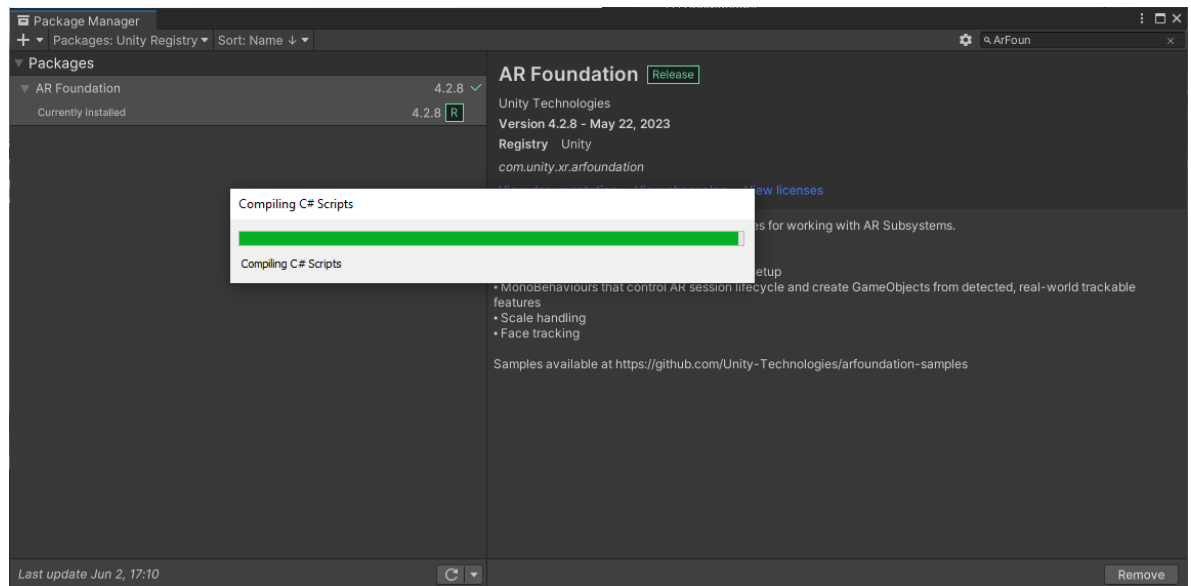
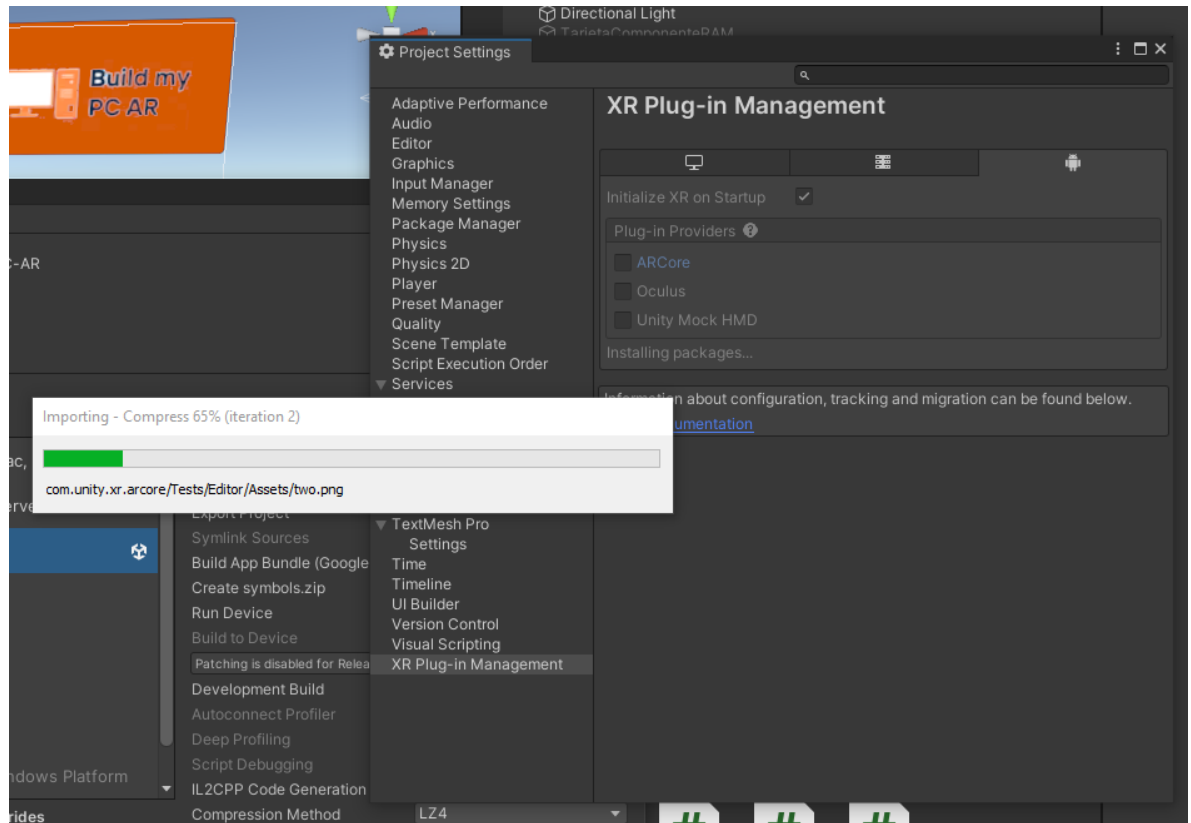
Instalación de DoTween:



Configuración de DOTween:



AR Foundation, ARCore y XR:

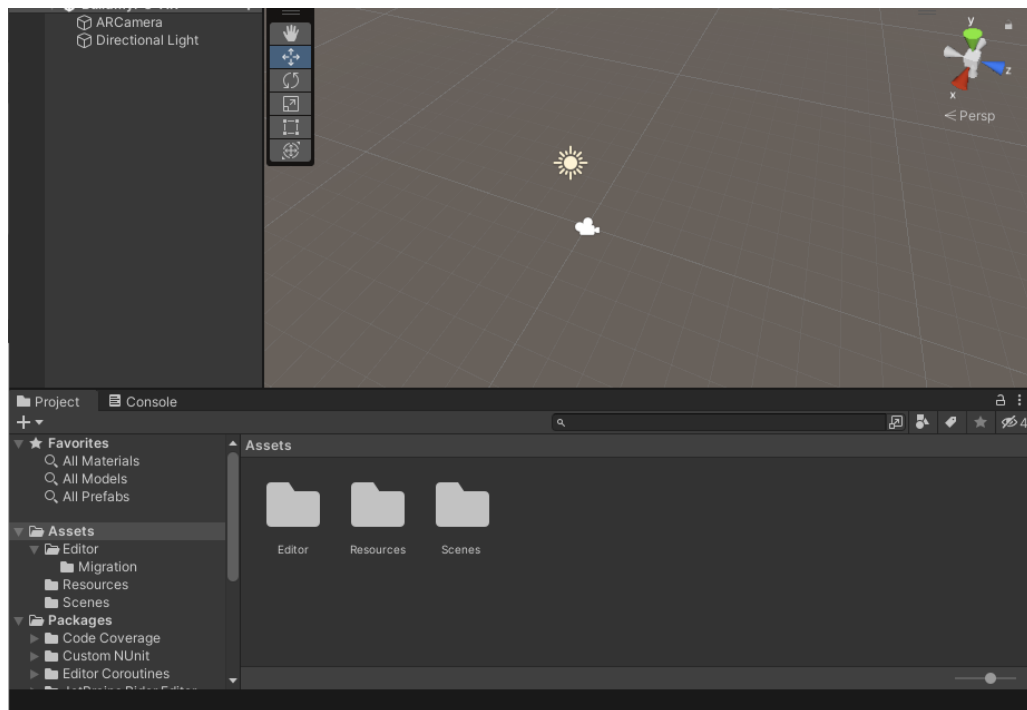


2. Desarrollo de Build My PC – AR

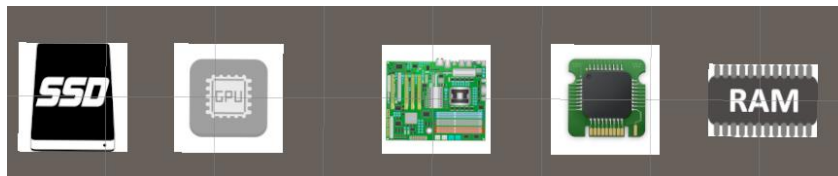
Build My PC – AR en su estado actual se compone de 2 formas de visualizar componentes, una forma implica su posicionamiento a través de superficies, mientras que la que se desarrollo en un principio aprovecha los Image Target de vuforia para la visualización de diferentes componentes de una PC de escritorio, por lo que comenzaremos con esta.

2.1 Escena: ComponentVisualizer

Lo primero en hacerse fue la creación de un objeto ARCamera para la visualización desde el dispositivo móvil, para ello instanciamos una ARCamera en la jerarquía de nuestro proyecto.



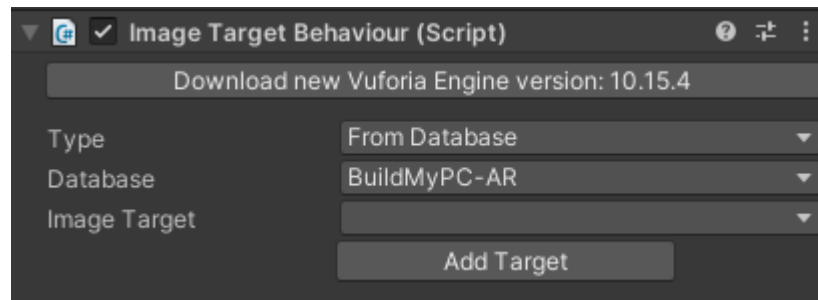
Ahora, crearemos los Image Target con los que visualizaremos los componentes, en total serán 5 y cada uno visualizará componentes diferentes.



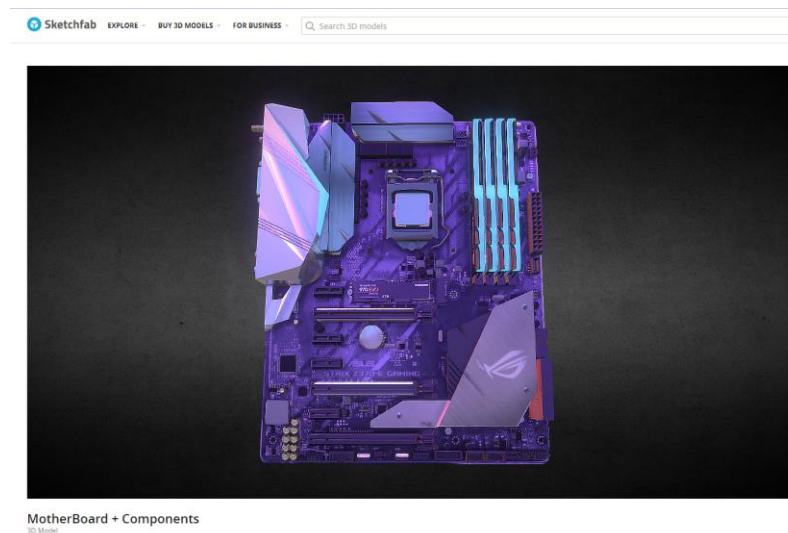
En la jerarquía se miran de la siguiente forma

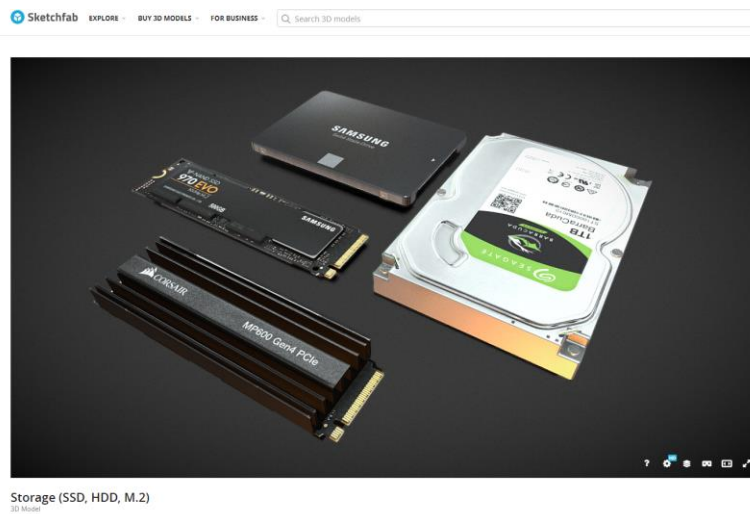
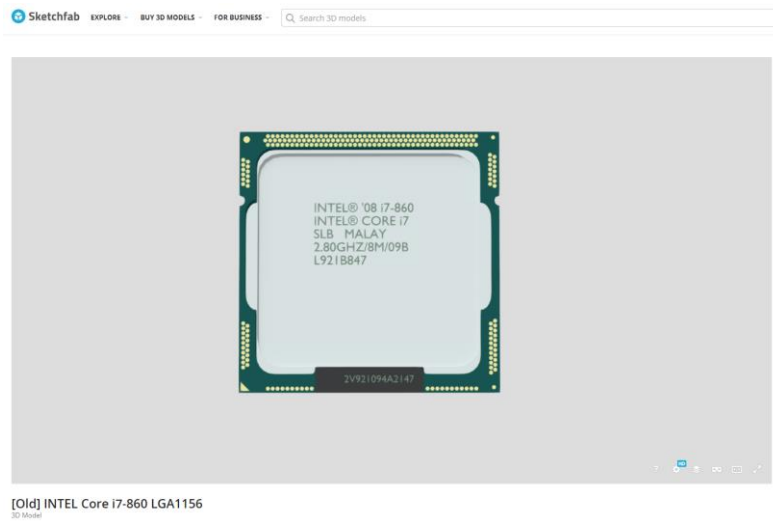


La forma en que se asignó una imagen como Target es mediante el inspector, ya que el objeto Image Target tiene su script por defecto llamado “**Image Target Behaviour**”, en este especificamos la imagen de la base de datos que deseamos que sea el detonante para mostrar contenido.

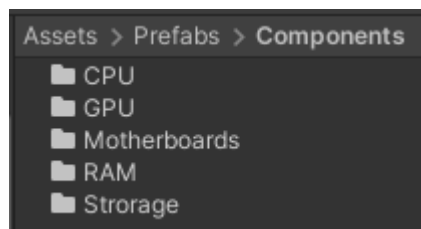


Ahora con nuestros Image Target creados, debemos de buscar componentes de una PC que deseamos mostrar en pantalla, para ello descargamos diferentes modelos en páginas de recursos gratuitos, aquí algunos de ellos como ejemplo:



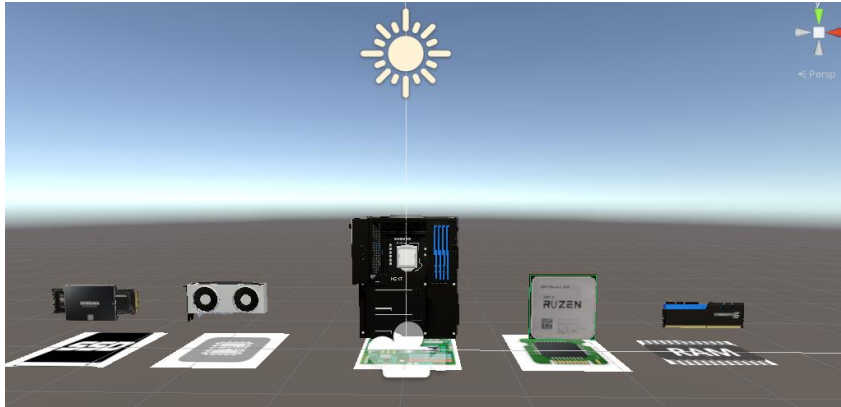


Una vez encontrados e importados los modelos de forma correcta dentro de Unity (Materiales y texturas) podríamos colocar los componentes dentro de sus Image Target correspondientes, sin embargo, como un intento de buena práctica se optó por crear Prefabs de cada uno de los modelos obtenidos, arreglando en algunos casos el pivote con el que venían los modelos. Entonces, en el editor creamos una carpeta llamada **Prefabs** y dentro creamos otra carpeta llamada **Components**. Después de todo el proceso de creación la carpeta se verá de la siguiente manera.



Dentro de cada subcarpeta se encuentran los prefabs de los componentes de cada tipo.

Una vez creados los prefabs podemos instanciar los mismos dentro de la escena y acomodarlos en su Image Target correspondiente.



Como podemos darnos cuenta, al posicionar más de un solo componente dentro del Image Target se comenzarán a empalmar unos con otros, por lo que será necesario crear un Script que se encargue de mostrar un modelo a la vez y de ejecutar un cambio de modelo en caso de que se solicite, para ello haremos un pequeño paréntesis y hablaremos de un recurso muy importante de ahora en adelante.

2.1.1 ScriptableObjects

Antes de hablar de la solución de este problema, me parece coherente hablar de algo primordial para la solución de varios otros problemas que se presentan a lo largo de todo el proyecto: Los **ScriptableObjects**

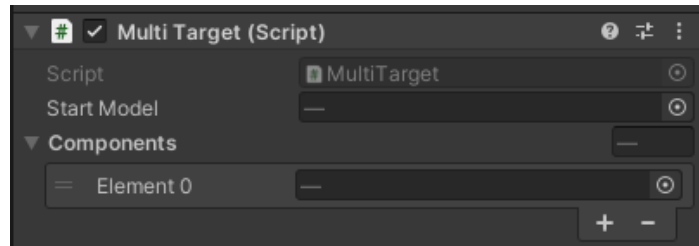
Los **ScriptableObjects** son Objetos que permiten guardar información de forma persistente, sirven al momento de optimizar un programa y en este contexto, nos sirven para guardar el nombre y descripción de nuestros componentes electrónicos.

A lo largo de toda la aplicación se hace uso de diferentes tipos de **ScriptableObjects**, por lo que a partir de este punto se mencionarán de forma explícita sin ningún otro tipo de explicación.

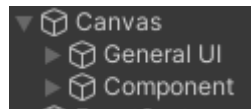
2.1.2 Solución al problema de los Múltiples modelos y detalles adicionales

Regresando al problema en cuestión, para visualizar un componente al mismo tiempo haremos uso de scripts, en este caso de **MultiTarget.cs**. Este recibe el modelo inicial del Image Target así como una lista de **ScriptableObject** de tipo **Component**, con estos dos parámetros se encarga de llevar la visualización de un modelo a la vez, al mismo tiempo de devolver el nombre y descripción del modelo actual.

En el editor el script se ve de la siguiente manera dentro de cada Image Target



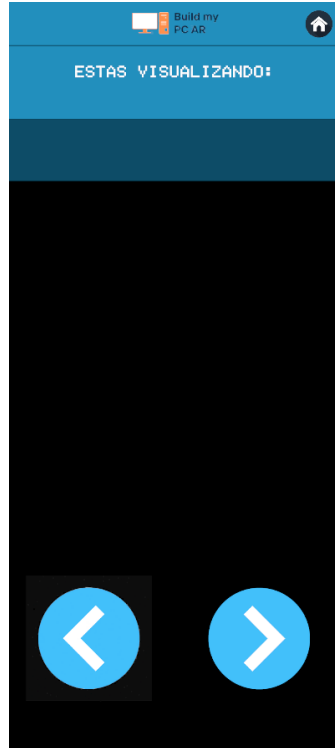
Arriba se mencionó que se visualizaría el nombre y descripción del componente actual, así como la capacidad de cambiar de modelo, para ello haremos uso de un canvas, para ello lo creamos en la jerarquía, se ve de la siguiente manera.



Donde:

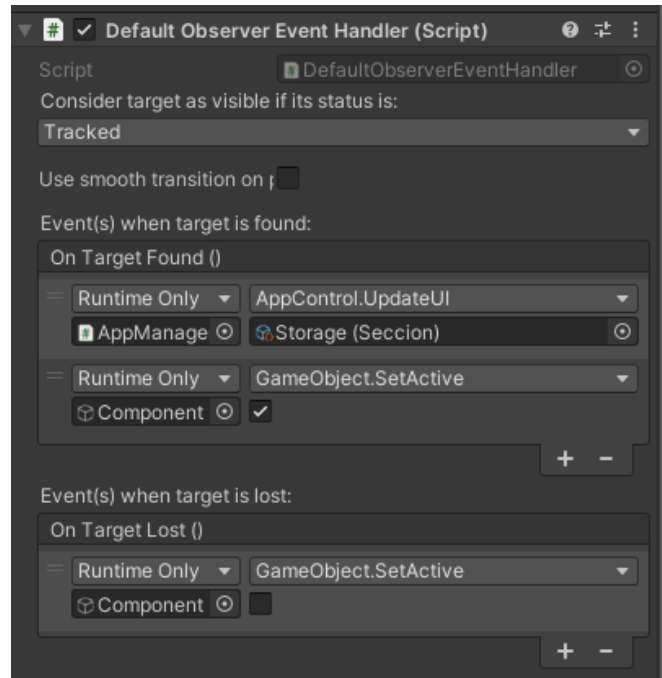
- **General UI:** Es una barra de navegación que te permite volver al menú principal de la aplicación
- **Component:** Es el canvas encargado de la información y cambio de modelos.

A grandes rasgos, el canvas completo de la escena se ve de la siguiente manera:



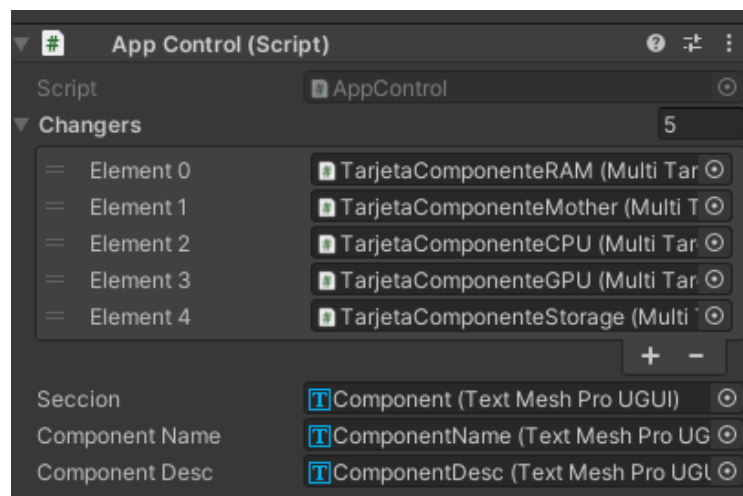
En los espacios vacíos se colocan de forma dinámica el tipo de componentes que estás viendo, el nombre del componente y una pequeña descripción de este.

Hablando de cambios dinámicos, para que la UI se actualice cada que se encuentre un Image Target nuevo, hacemos uso de los eventos **OnTargetFound()** y **OnTargetLost()** que cada Image Target posee. De forma general cada Target se ve de la siguiente manera:



Con la diferencia de que cada Image Target envía un **ScriptableObject** de tipo **Seccion** diferente, siendo este el que concuerda con el tipo de componentes que estamos visualizando. Por dentro, la función **UpdateUI** del script **AppManager** actualiza también el nombre y descripción del objeto en visualización.

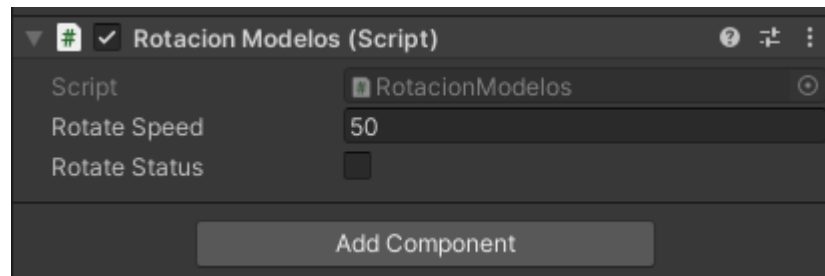
Hablando de **AppManager**, este es un objeto vacío al que se le otorga el Script del mismo nombre y este actúa como controlador de la escena. A este objeto se le pasan los siguientes parámetros.



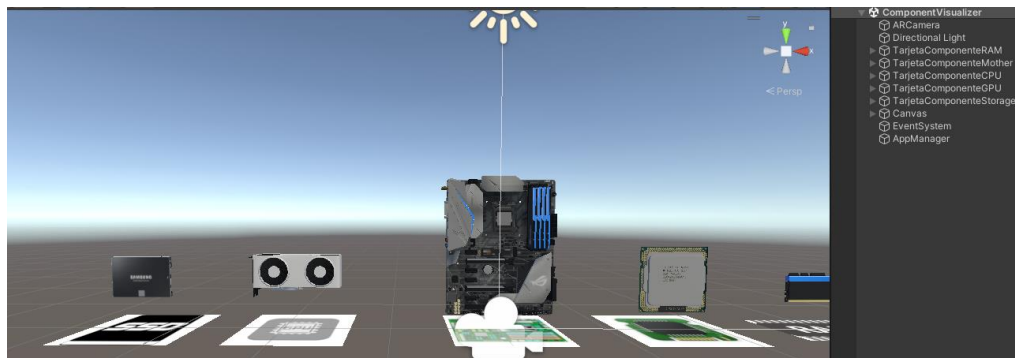
Donde:

- **Changers:** Son los diferentes Image Target que tenemos en la escena
- **Seccion:** Es el texto dentro de la UI que nos dice el tipo de componentes que estamos visualizando
- **ComponentName:** Es el nombre del componente que estamos visualizando
- **ComponentDesc:** Es la descripción corta que tiene este componente.

Por último pero no menos importante, cada modelo cuenta con la capacidad de rotar sobre su propio eje, esto con el fin de brindar algo de dinamismo a la hora de visualizar los componentes, para ello tienen asociado un script llamado **RotacionModelos.cs**.



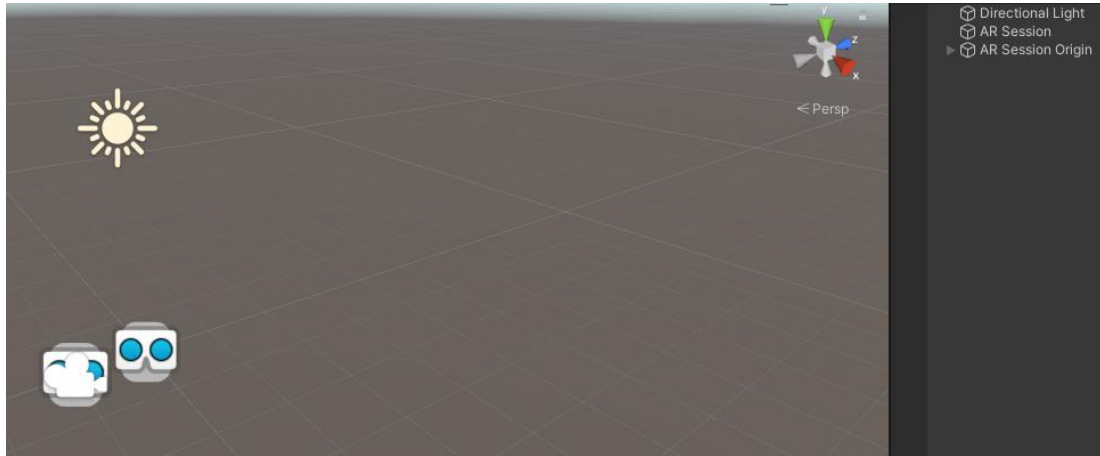
Al final la escena en el editor se mira de la siguiente manera



Con esto podríamos dar por concluida esta escena y damos paso a la siguiente parte de la App: La visualización de componentes en superficies.

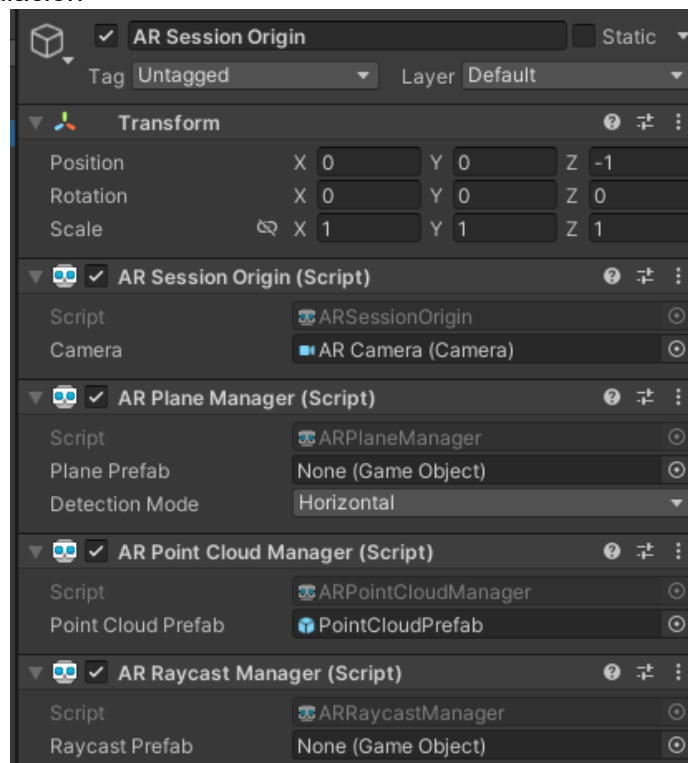
2.2 Escena: BuildMyPC-AR

Para esta escena haremos uso de AR Foundation, por ello importaremos diferentes elementos a la jerarquía, estos son un **AR Session** y un **AR Session Origin**

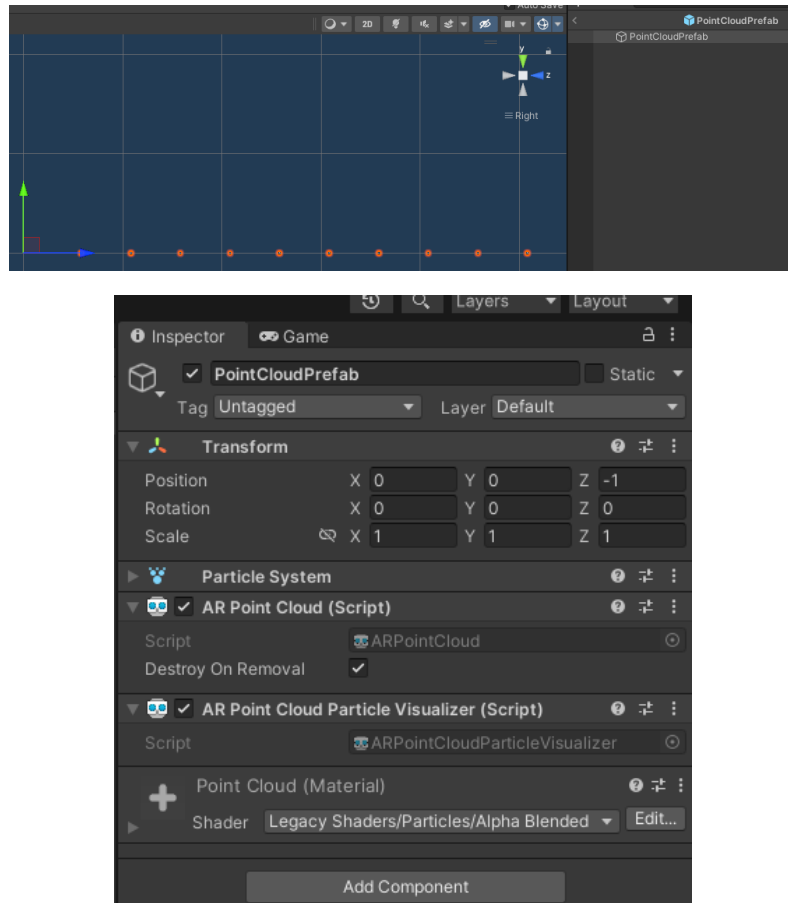


Al objeto AR Session Origin le agregaremos los siguientes componentes:

- **AR Plane Manager:** Para la detección de planos
- **AR Point Cloud Manager:** Para el uso de una nube de puntos como indicadora de detección de superficies
- **AR Raycast Manager:** Para la detección de colisiones con objetos y su manipulación



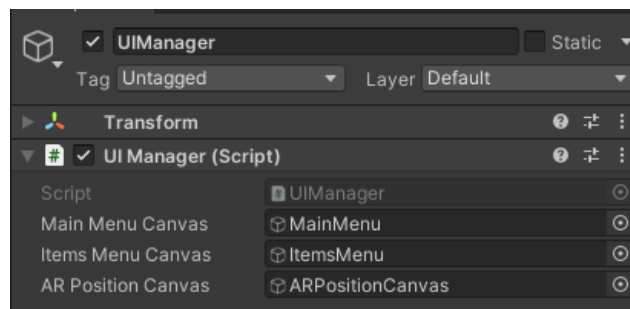
El Prefab **PointCloudPrefab** es una nube de puntos que creamos nosotros, a continuación, muestro el Prefab y sus componentes

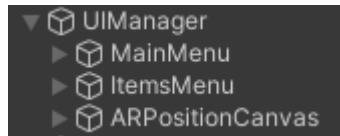


La forma en que se maneja esta escena se encuentra repartida por diferentes componentes, a continuación, explico cada uno de ellos.

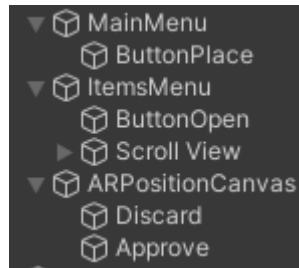
2.2.1 UI Manager

El objeto UI manager se encarga como dice su nombre de manejar los diferentes cambios en la UI de la escena, para ello hace uso de un Script con el mismo nombre en donde se solicitan los Canvas de los 3 menús disponibles en la escena. Internamente el script tiene los comportamientos de las UI en cada caso específico. En el editor el componente se ve de la siguiente manera





Internamente, UI Manager posee los canvas de cada Menú, estos a su vez poseen botones que se encargan de llevar el flujo de ejecución de la aplicación.

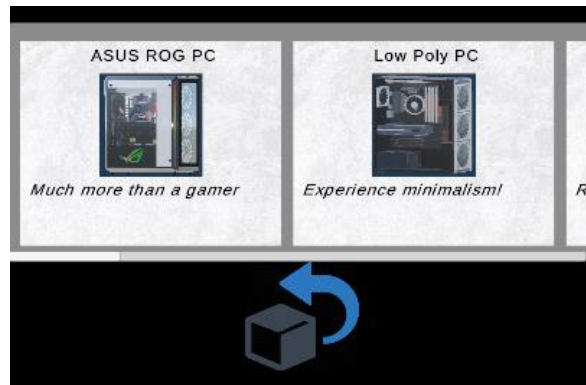


Los Canvas se ven de la siguiente manera:

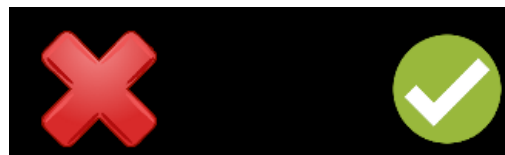
MainMenu



ItemMenu



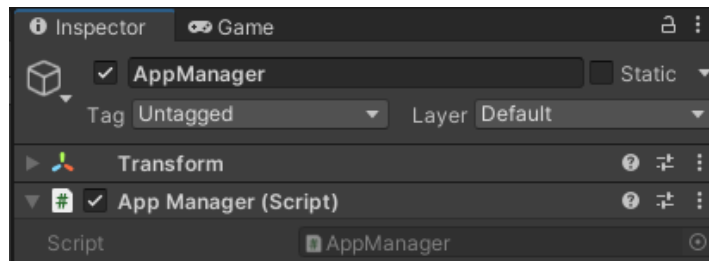
ArPositionCanvas



Cada botón llama a una función del **AppManager**, componente del que hablaremos a continuación.

2.2.2 App Manager

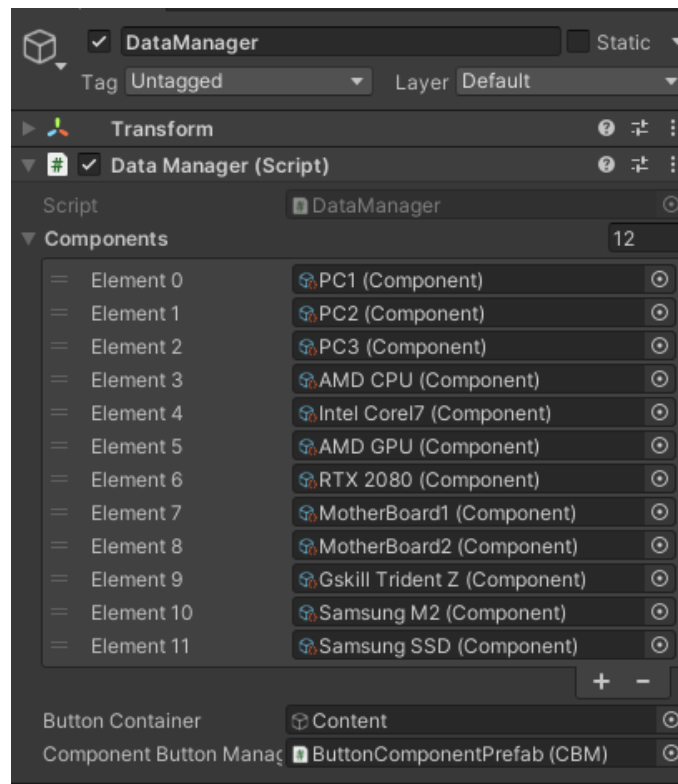
Este objeto se encarga de llamar las transiciones de menús dentro de la escena, el script que tiene asociado posee las funciones para esto, en el editor se ve de la siguiente manera



2.2.3 Data Manager

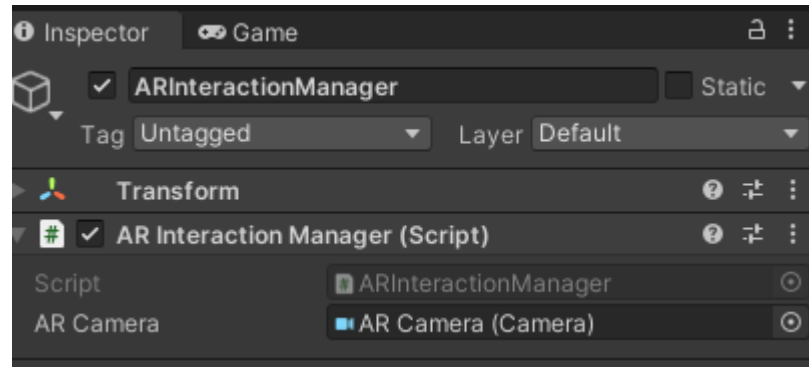
Data Manager se encarga de administrar los botones y los componentes que podremos colocar en las superficies, el script asociado a este objeto solicita una lista de **ScriptableObject** de tipo **component**, estos serán los objetos por mostrar en ejecución. De la misma forma, solicita un canvas para depositar el contenido y un prefab de botón con el script **CBM (Custom Button Manager)** encargado de posicionar el contenido de los ScriptableObjects dentro del botón.

En el editor este objeto se ve de la siguiente manera

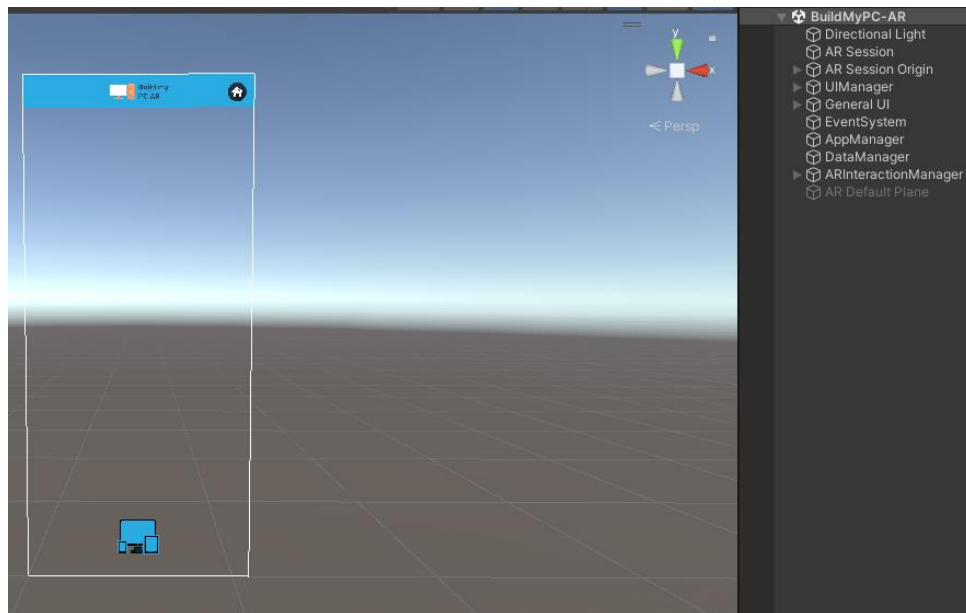


2.2.4 AR Interaction Manager

Este objeto se encarga de las interacciones con los objetos en el entorno de RA, es decir que este objeto es el encargado de que podamos posicionar objetos en las superficies, eliminarlos en caso de desecharlo, reposicionarlos y hasta aplicarles una rotación. En el editor se ve de la siguiente manera



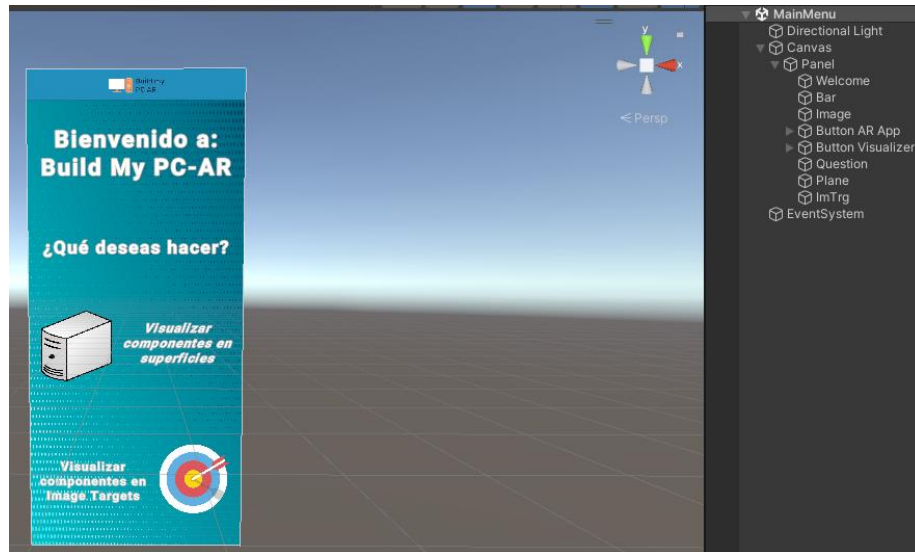
Cubiertos todos los puntos, la escena completa se visualiza de la siguiente manera.



2.3 Escena: MainMenu

Esta escena solo posee un canvas ya que es solo el menú principal de la aplicación, en ella el usuario puede decidir la forma en la que visualizará sus componentes.

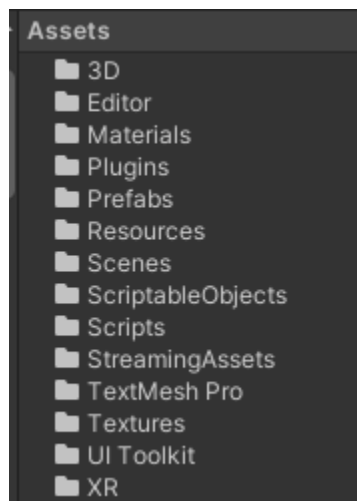
En general se mira de la siguiente manera



Lo único relevante a explicar en esta escena son los botones: Estos tienen en sus funciones **On Click()** una llamada a una función de cambio de escena, estas funciones se encuentran en el Script **StartApp.cs**

2.4 Estructura general de los archivos

La estructura general del proyecto es la siguiente



Donde:

- **3D:** Aquí guardo todos los modelos usados
- **Prefabs:** Aquí se guardan los Prefabs usados en todo el proyecto

- **Scenes:** Aquí se guardan todas las escenas creadas
- **ScriptableObjects:** Aquí se almacenan todos los ScriptableObjects usados
- **Textures:** Aquí se almacenan todos los recursos usados en los modelos y UI

3. Reporte Financiero y esquemas de Tiempo

En comparación al reporte financiero presentado al momento de hacer la proposición de proyecto se realizó una tabla nueva de precios considerando un aspecto nuevo

Tabla de propuesta de proyecto

Componente	Precio
Licencias Unity/ 3DS Max / Editor de Imágenes	\$0.00
PC de escritorio: Ryzen 5 3500 Nvidia GTX1650 Fuente de poder EVGA 400 W 16 GB RAM DDR4	\$16,999.00
Smartphone Android/IOS	\$6,000.00 - \$32,000.00
Uso de electricidad	~\$300.00
Sueldo del programador	\$10,000.00 - \$15,000.00
Materiales para tarjetas	~\$300.00
Modelos 3D	~\$5,000.00
TOTAL	\$38,999.00 - \$69,599.00

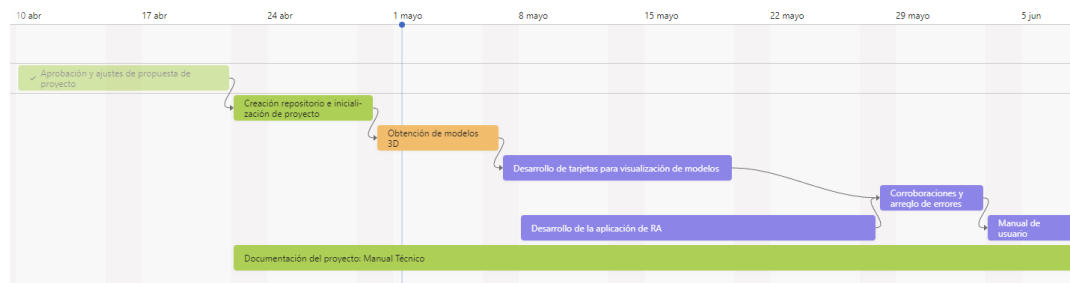
Tabla de precios finales:

Componente	Precio
Licencias Unity/ 3DS Max / Editor de Imágenes	\$0.00
PC de escritorio: Ryzen 5 3500 Nvidia GTX1650 Fuente de poder EVGA 400 W 16 GB RAM DDR4 Laptop Huawei D14s	\$41,399.00
Smartphone Android/IOS	\$6,000.00 - \$32,000.00
Uso de electricidad	~\$300.00
Sueldo del programador	\$10,000.00 - \$15,000.00
Materiales para tarjetas	~\$300.00
Modelos 3D	~\$5,000.00
TOTAL	\$63,499.00 - \$94,099.00

Podemos notar que hubo un aumento de precio total de aproximadamente \$24,500 pesos mexicanos, esto fue debido a una adquisición que se realizó en pro del proyecto, por lo que podemos decir que si terminamos gastando más de lo que esperábamos.

Si consideramos que el precio de venta de nuestro proyecto debe de ser de al menos el doble para obtener una ganancia del 50%, estaríamos frente a un proyecto cuyo valor ronda entre los **\$95,250.00 - \$141,148.50** pesos mexicanos

En cuanto al diagrama de Gantt presentado en la propuesta, es de mi agrado comentar que se cumplieron mayormente los tiempos planificados a lo largo de este.



4. Funcionamiento de la App

A continuación, presento algunas capturas de pantalla de la aplicación corriendo en un dispositivo móvil

Menú:



BuilMyPC-AR



ComponentVisualizer



5. Referencias

- Code Monkey. (2019, March 16). *How to modify the Game Object Transform Pivot (Unity Tutorial)* [Video]. YouTube.
<https://www.youtube.com/watch?v=NsUJDqEY8tE>
- Brackeys. (2017, November 29). *START MENU in Unity* [Video]. YouTube.
https://www.youtube.com/watch?v=zc8ac_qUXQY
- Trucos de diseño. (2020, April 22). *Crea menú, paneles o canvas en realidad aumentada con Unity y Vuforia* [Video]. YouTube.
<https://www.youtube.com/watch?v=epPxxF71zEk>
- *Crea tu aplicación Realidad Aumentada, Curso Completo.* (n.d.). YouTube.
https://www.youtube.com/playlist?list=PLSc07dYXbtBmf2iF6GeWV_lvHaqLMJWdp
- Unity Adventure. (2022, June 24). *ImageTarget Vuforia ¿Cómo cambiar el Modelo de un Image Target?* [Video]. YouTube.
<https://www.youtube.com/watch?v=IjNUBcEDwk0>
- Indierama. (2021, September 1). *SCRIPTABLE OBJECTS en Unity | español* [Video]. YouTube. <https://www.youtube.com/watch?v=WHvJ8RdlOQI>
- Technologies, U. (n.d.). |. Unity. <https://unity.com/es/how-to/architect-game-code-scriptable-objects>
-

Modelos 3D y recursos

- "MOTHERBOARD + COMPONENTS" (<https://skfb.ly/6RDsX>) by Daniel Cardona is licensed under Creative Commons Attribution (<http://creativecommons.org/licenses/by/4.0/>).

- "*MOTHERBOARDS*" (<https://skfb.ly/6VuN9>) by Blue Odyd is licensed under Creative Commons Attribution (<http://creativecommons.org/licenses/by/4.0/>).
- "*STORAGE (SSD, HDD, M.2)*" (<https://skfb.ly/6SNzv>) by Blue Odyd is licensed under Creative Commons Attribution (<http://creativecommons.org/licenses/by/4.0/>).
- "*RED GRAPHICS CARD*" (<https://skfb.ly/o9WUH>) by Cem Gürbüz is licensed under Creative Commons Attribution-NonCommercial (<http://creativecommons.org/licenses/by-nc/4.0/>).
- "*(FREE) GAMING PC*" (<https://skfb.ly/oGSTB>) by Moonway 3D is licensed under Creative Commons Attribution (<http://creativecommons.org/licenses/by/4.0/>).
- "*[OLD] INTEL CORE I7-860 LGA1156*" (<https://skfb.ly/ozw8P>) by Хлюпич is licensed under Creative Commons Attribution (<http://creativecommons.org/licenses/by/4.0/>).
- "*DREAM COMPUTER SETUP*" (<https://skfb.ly/6QW96>) by Daniel Cardona is licensed under Creative Commons Attribution (<http://creativecommons.org/licenses/by/4.0/>).
- "*AMD RYZEN 5 3600 CPU*" (<https://skfb.ly/o76sR>) by Temoor is licensed under Creative Commons Attribution (<http://creativecommons.org/licenses/by/4.0/>).