REALIDAD VIRTUAL

PROYECTO FINAL

MONOPOLY MULTIJUGADOR CON REALIDAD AUMENTADA

ALUMNO: Fragapane Adriel

LEGAJO: 10401

ÍNDICE

1. INTRODUCCIÓN

El proyecto elegido para integrar los conocimientos adquiridos en la materia de Realidad Virtual de la carrera de Mecatrónica, consiste en una aplicación para dispositivos móviles, que replica en principio el clásico juego de mesa ***Monopoly****, el cual integra dos factores fundamentales:*

- *Es* ***multijugador****, por lo que dos, tres, o cuatro personas pueden jugaren la misma partida, interactuando entre ellas mediante pagos/cobros de impuestos y compra/venta de propiedades.*

- *El juego tendrá una modalidad de* ***realidad aumentada****, en donde tanto el tablero como los peones se desplegarán de forma virtual, sobre una superficie real que esté siendo captada por la cámara del dispositivo. Adicionalmente, se podrá desplegar en pantalla información sobre cada uno de los jugadores que participan.*

2. ANTECEDENTES

El concepto de ***realidad aumentada*** *(en inglés Augmented Reality o AR) agrupa aquellas tecnologías que permiten la superposición, en tiempo real, de imágenes, marcadores o información generados virtualmente, sobre imágenes del mundo físico. Se crea de esta manera un entorno en el que* ***la información y los objetos virtuales se fusionan con los objetos reales****, ofreciendo una experiencia tal para el usuario, que puede llegar a pensar que forma parte de su realidad cotidiana, olvidando incluso la tecnología que le da soporte. Es por ello que la realidad aumentada pueda ser entendida como una tecnología que ofrece una nueva lente para ver el mundo, para percibirlo de una manera "aumentada".*

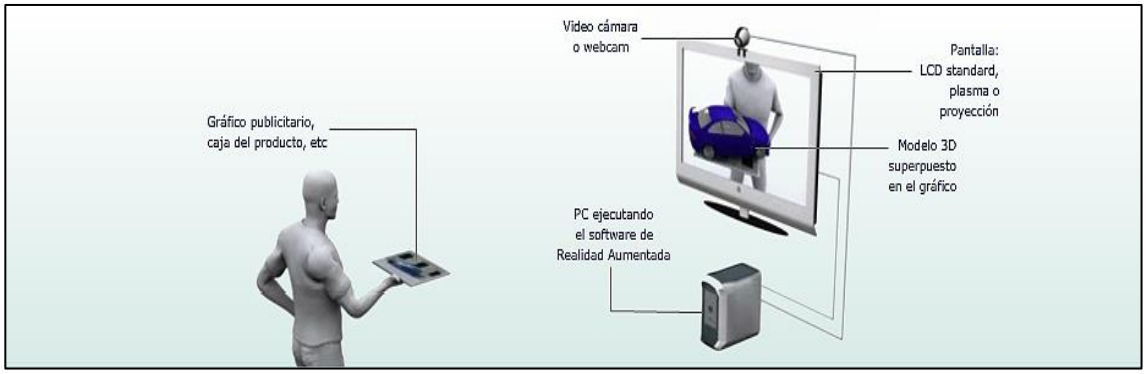

Figura 1. Elementos necesarios para realizar una aplicación de realidad aumentada

De manera simplificada, por lo tanto, para componer un servicio de realidad aumentada son necesarios 4 ingredientes básicos:

1. *Un **elemento que capture las imágenes** de la realidad que están viendo los usuarios. Por ejemplo una sencilla cámara de las que están presentes en los ordenadores o en los teléfonos móviles.*

2. *Un **elemento sobre el que proyectar** la mezcla de las imágenes reales con las imágenes sintetizadas. Para ello se puede utilizar la pantalla de un ordenador, de un teléfono móvil o de una consola de videojuegos.*

3. *Un **elemento de procesamiento**, o varios de ellos que trabajan conjuntamente. Su cometido es el de interpretar la información del mundo real que recibe el usuario, generar la información virtual que cada servicio concreto necesite y mezclarla de forma adecuada. Nuevamente encontramos en los PCs, móviles o consolas estos elementos.*

4. *Un elemento que podría denominarse **activador de realidad aumentada**. En un mundo ideal el activador sería la imagen que están visualizando los usuarios, ya que a partir de ella el sistema debería reaccionar. Pero, dada la complejidad técnica que este proceso requiere, en la actualidad se utilizan otros elementos que los sustituyen. Se trata entonces de elementos de localización como los GPS que en la actualidad van integrados en gran parte de los **smartphones**, así como las brújulas y acelerómetros que permiten identificar la posición y orientación de dichos dispositivos, así como las etiquetas o marcadores del tipo RFID o códigos bidimensionales, o en general cualquier otro elemento que sea capaz de suministrar una información equivalente a la que proporcionaría lo que ve el usuario, como por ejemplo sensores. En un caso ideal, algunos de estos elementos podrían llegar a eliminarse. Esto ocurriría si se consigue, por ejemplo, proyectar la información sintetizada de forma que el ojo sea capaz de verla, bien sobre unas gafas, directamente sobre la retina, o con alguna técnica holográfica avanzada.*

3. APLICACIONES

3.1. JUEGOS

*Un práctico ejemplo de realidad aumentada es **Pokémon GO**. Este popular juego utiliza esta tecnología para colocar a las criaturas virtuales y los objetos que permiten capturarlos en lugares*

reales. Las imágenes digitales de estos elementos virtuales no son más que píxeles en la pantalla del usuario, pero la tecnología de realidad aumentada los hace parecer tangibles.

Figura 2. Ejemplo del juego Pokémon GO.

3.2. TURISMO

La realidad aumentada puede ser utilizada en el sector del turismo para mejorar la experiencia del usuario durante las visitas a museos o lugares de interés. En Burdeos, por ejemplo, es posible ver los monumentos actuales en el contexto del siglo XVIII simplemente descargando una aplicación en una tableta. La Torre Eiffel también tiene su propia aplicación de realidad aumentada en la que es posible ver su construcción, entre otras opciones.

3.3. NEGOCIOS

Hay múltiples usos para la realidad aumentada en los negocios. En algunas tiendas físicas, la realidad aumentada permite a los usuarios probarse la ropa de forma virtual, es decir, sin ir a los probadores. Otras incluso utilizan las aplicaciones de realidad aumentada para atraer clientes. En muchas ciudades del mundo, una gran cantidad de tiendas abrieron sus puertas para que los jugadores de Pokémon GO capturen sus Pokémon dentro de ellas.

3.4. COMERCIO ELECTRÓNICO

En las tiendas online, la realidad aumentada se aplica a muchas áreas. Algunas tiendas de productos oftalmológicos ofrecen a los consumidores la oportunidad de probarse virtualmente gafas y lentes de contacto cargando una foto de su rostro, y muchas tiendas de decoración y muebles (como IKEA) permiten a los clientes visualizar objetos en versiones virtuales de sus hogares antes de realizar la compra.

3.5. AUTOMÓVILES

En algunos modelos de automóviles, la tecnología de realidad aumentada se utiliza para mostrar información directamente sobre el parabrisas. Este modo de visualización proporciona información sobre velocidad, obstáculos, anchura de la carretera y tráfico.

4. REALIDAD AUMENTADA VS. REALIDAD VIRTUAL

Aunque aparentemente son similares, la realidad aumentada y la realidad virtual son tecnologías muy distintas. La realidad virtual es una tecnología informática que replica artificialmente un entorno real (o imaginario) y le proporciona al usuario (apelando principalmente a su visión y audición) la sensación de estar verdaderamente dentro de él. La realidad virtual puede ser experimentada a través del uso de gafas especiales como el ***Oculus Rift****.*

5. HARDWARE UTILIZADO

Debido a que el proyecto consiste en realidad aumentada, el elemento que obtiene la información del mundo real es la misma cámara del dispositivo, ya sea computadora, Tablet o teléfono móvil Smartphone. Por lo tanto el elemento que capture las imágenes de la realidad, el elemento de procesamiento, y el elemento de proyección se encuentran integrados en el mismo dispositivo y no se requiere hardware adicional.

NOTEBOOK BANGHO MAX G0101

Versión de Sistema Operatico: *Windows 10 Home SL*
Procesador
- ***Familia:*** *Intel Core i7*
- ***Modelo:*** *i7-4710MQ 2.50MHz*
- ***Generación:*** *7ma generación*

Memoria RAM
- ***Capacidad:*** *8 GB*
- ***Tecnología:*** *DDR4 2400MHZ*

Almacenamiento
- ***Disco rígido:*** *1 TB*
- ***Tecnología:*** *SATA 3*

Procesador gráfico: *Intel HD Graphics 4600*
Pantalla
- ***Tamaño:*** *15.6”*
- ***Resolución:*** *1920 x 1080 FULL HD*

Webcam: *Incorporada*

Figura 3. Notebook Bangho utilizada para el proyecto.

6. SOFTWARE UTILIZADO

*El programa completo se desarrolló con el programa **UNITY 3D**. Unity es un motor de videojuegos multiplataforma creado por Unity Technologies. Unity está disponible como plataforma de desarrollo para Microsoft Windows, OS X, Linux. La plataforma de desarrollo tiene soporte de compilación con diferentes tipos de plataformas (Véase la sección Plataformas objetivo). A partir de su versión 5.4.0 ya no soporta el desarrollo de contenido para navegador a través de su plugin web, en su lugar se utiliza WebGL. Unity tiene dos versiones: Unity Professional (pro) y Unity Personal. Unity es un programa sumamente versátil, poderoso y relativamente fácil de utilizar. La programación se realiza en lenguaje **C#**.*

*C# (pronunciado **si sharp** en inglés) es un lenguaje de programación orientado a objetos desarrollado y estandarizado por Microsoft como parte de su plataforma .NET, que después fue aprobado como un estándar por la ECMA (ECMA-334) e ISO (ISO/IEC 23270). C# es uno de los lenguajes de programación diseñados para la infraestructura de lenguaje común.*

Su sintaxis básica deriva de C/C++ y utiliza el modelo de objetos de la plataforma .NET, similar al de Java, aunque incluye mejoras derivadas de otros lenguajes.

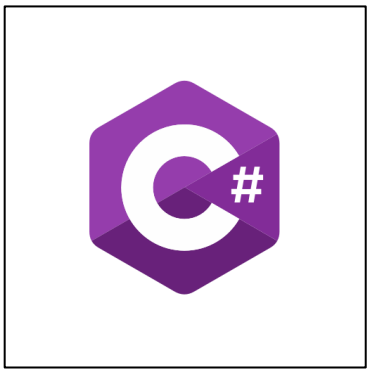

Figura 4. Lenguaje de programación C#

Para comenzar a utilizar el programa se utilizó información y tutoriales de la página de documentación oficial de Unity: https://docs.unity3d.com

Figura 5. Logo de la página de documentación de Unity.

Además se utilizó un tutorial online que se adquirió por US$ 10 en una página www.udemy.com el cual cuenta con 285 clases y 35 horas de contenido en español.

Figura 7. Logo de la página de cursos online Udemy.

Figura 6. Portada del curso tomado.

7. CREACIÓN DEL ENTORNO

Para iniciar a desarrollar el juego, primero se hace una pequeña introducción a la interfaz de Unity con el objetivo de facilitar la comprensión en el resto del informe. En la siguiente figura se observan las diferentes partes del programa, aunque cuando se inicia no necesariamente se presenta con la misma disposición:

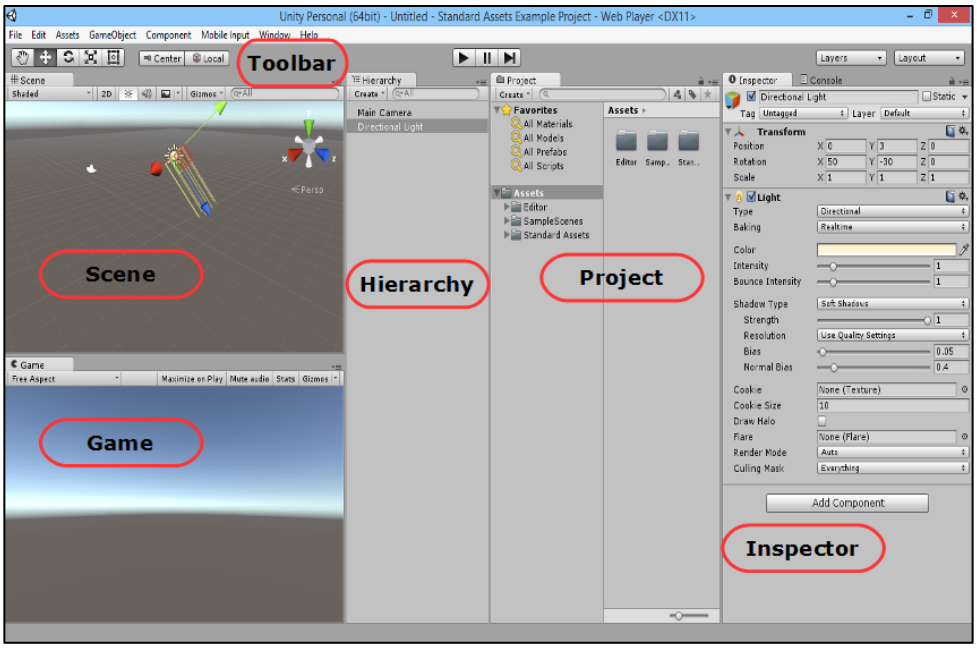

Figura 8. Captura de la pantalla que presenta Unity al empezar un nuevo proyecto.

Las partes principales del programa son:

- ***Barra de Herramientas <Toolbar>:** Al igual que la mayoría de los programas, cuenta con la barra de herramientas, que contiene una gran cantidad de opciones agrupadas en las pestañas.*
- ***Escena <Scene>:** Es donde se ubican todos los objetos **<GameObjects>** que conforman la escena y donde se pueden manipular y editar.*
- ***Juego <Game>:** Es la pantalla que se visualiza al ejecutar el juego. En ella no se pueden modificar los objetos si la configuración del juego no lo permite.*
- ***Jerarquía <Hierarchy>:** En este panel se colocan los nombres de todos los componentes que se encuentran en la escena, estén activos o no. Aquí se puede definir un objeto como hijo de otro (depende de él).*
- ***Proyecto <Project>:** A través de éste panel se puede explorar todos los archivos que conforman el proyecto, y también administrarlos a conveniencia.*

- ***Inspector <Inspector>:*** *En éste panel se observan todas las características de cada objeto. Estas características se agrupan en componentes* ***<Components>*** *los cuales se pueden agregar, quitar, o modificar. Éste es el panel que más se utiliza.*

Al iniciar el programa, se crea una escena ***<Scene>*** *y por defecto coloca dos objetos, una luz direccional* ***<Directional Light>*** *que ilumina todos los componentes de la escena, y una cámara* ***<Main Camera>*** *que captura la porción de la escena que se mostrará en pantalla* ***Game*** *al ejecutarlo.*

En el proyecto se crean dos escenas, una cuando se ingresa al juego, que despliega un menú con las opciones de conexión a la cual se nombra ***Escena Offline****, y otra en la cual transcurre el juego en sí, a la cual se nombra* ***Escena Online****.*

8. ESCENA ONLINE

Primero se procede a elaborar esta escena debido a que es la principal, y en base a sus características se elabora el menú de la ***Escena Offline****.*

El primer paso es colocar el tablero de juego, para lo cual se crea un ***GameObject 3D*** *llamado plano* ***<Plane>*** *que se encuentra en la librería por defecto de Unity. Luego se modifica la posición, rotación y escala en el Inspector, y se le agrega un material* ***<Material>*** *que contiene una textura* ***<Texture>,*** *la textura es la imagen que se cubrirá el plano.*

Además, al objeto tablero se le agrega un archivo de código o ***<Script> C#*** *que posee los datos de cada uno de los casilleros. Estos datos son guardados en una matriz de 40 filas (por los 40 casilleros) por 15 columnas (15 datos específicos):*

1	***Precio:***	***Valor de compra de la propiedad.***
2	***Alquiler:***	***Valor de alquiler sin construir ninguna casa.***
3	***Con1Casa:***	***Valor de alquiler con una casa construida.***
4	***Con2Casas:***	***Valor de alquiler con dos casas construidas.***
5	***Con3Casas:***	***Valor de alquiler con tres casas construidas.***
6	***Con4Casas:***	***Valor de alquiler con cuatro casas construidas.***
7	***ConHotel:***	***Valor de alquiler con un hotel construido.***
8	***Hipoteca:***	***Valor de la hipoteca de la propiedad.***
9	***CostoCasas:***	***Valor de construcción de cada casa en la propiedad.***
10	***CostoHoteles:***	***Valor de construcción de un hotel en la propiedad.***
11	***Ocupado:***	***Indica si la propiedad se encuentra ocupada o no.***
12	***Propietario:***	***Indica qué jugador ocupa la propiedad.***
13	***Construccion:***	***Indica el estado de construcción (cantidad de casas).***
14	***PoseeTodoElGrupo:***	***Indica si el jugador que la ocupa posee el grupo entero.***
15	***Hipotecado:***	***Indica si la propiedad está o no hipotecada.***

Todos los jugadores pueden leer y modificar la matriz para indicar al resto que se ha comprado o vendido una propiedad, que se ha hipotecado, que ha construido una casa u hotel, etc.

Una vez creado el tablero, se colocan dos planos nuevos cuyas texturas tendrán la imagen de ***CARTA DE LA SUERTE*** *y* ***SERVICIO A LA COMUNIDAD*** *que se encontrarán apoyadas sobre el tablero en todo momento. Para ubicarlas se modifica su posición en y escala en el inspector. Como las cartas dependerán en todo momento del tablero, se procede a hacerlas* ***hijas*** *de él en el panel Jerarquía, arrastrándolas sobre el mismo. En la escena se colocan dos cartas de cada tipo, una que representa el mazo, y otra que representa la carta superior y se animará cuando el jugador deba retirar alguna de ellas.*

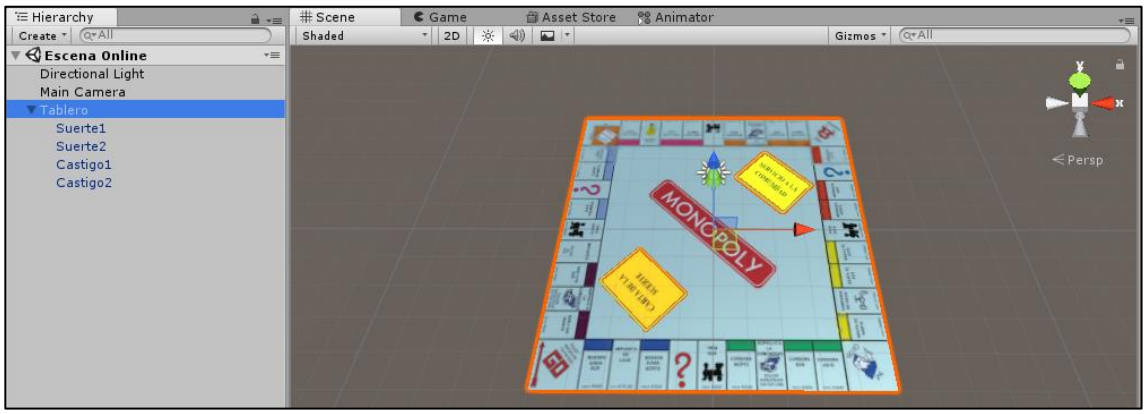

Figura 9. Imagen de la escena con el tablero y cartas en posición adecuada.

A medida que se crean objetos de gran importancia, se los transforma en Objetos Prefabricados ***<Prefabs>*** *arrastrándolos al panel Proyecto. Un Prefab es un GameObject transformado en archivo, para lo cual se puede reutilizar cuantas veces se desee, simplemente arrastrándolo desde el panel Proyecto hasta el panel Jerarquía.*

El siguiente paso es crear los jugadores, con todos sus elementos.

Al conectarse un jugador nuevo a la partida, lo que se agregará a la escena es lo siguiente:

- *Un objeto Peón, que será representado por un cubo (incluido en la librería 3D por defecto) al cual se le dará una forma prismática y un color determinado para diferenciarlo del resto de los jugadores. Este Objeto Peón contiene un Script en C# que define el comportamiento del mismo durante el juego y la forma en la que interactúa con los otros objetos.*
- *Un objeto Canvas* ***<Canvas>*** *que representa una interfaz de usuario para que cada jugador visualice en su pantalla y pueda interactuar con el juego.*

Dentro del Canvas se colocan los botones ***<Button>,*** *cuadros de texto* ***<Text>,*** *imágenes* ***<Image>,*** *campos de entrada de datos* ***<InputField>,*** *y todo tipo de elementos que conforman una interfaz de usuario. A su vez, estos objetos se encuentran agrupados en paneles* ***<Panels>*** *para organizarlos y activarlos o desactivarlos según corresponda. Los dos paneles que agrupan la totalidad de los elementos son dos:*

- ***Panel principal:*** *Es el que se despliega al iniciar la partida y contiene los siguientes elementos:*
  - ***Panel Dinero:*** *Indica el número de jugador y el dinero que posee.*
  - ***Panel Dado:*** *Se mantiene siempre visible y contiene:*
    - *Un botón para tirar un valor aleatorio entre 1 y 6.*
    - *Un cuadro en donde se visualiza el número obtenido.*
    - *Un botón que despliega un teclado numérico para elegir el número manualmente (este botón se utiliza solo para modo de demostración, en el juego real sólo se puede obtener números al azar).*
  - ***Carta Casillero:*** *Es una imagen que se despliega cuando el peón cae en un casillero y representa la carta correspondiente a la propiedad, o en caso de ser un casillero de "suerte" o "servicio a la comunidad", despliega una carta al azar del mazo indicado.*
  - ***Panel Comprar:*** *Se despliega cuando el peón cae en un casillero, y contiene:*
    - *Un cuadro de texto que informa si la propiedad se encuentra ocupada o no y por qué jugador, y el valor de compra o de alquiler según corresponda.*
    - *Dos botones que permiten comprar o no la propiedad si esta se encuentra desocupada.*
    - *Un botón que permite pagar el alquiler si la propiedad se encuentra ocupada.*
  - ***Panel Comprar 2:*** *Se despliega cuando otro jugador desea venderle una propiedad y contiene:*
    - *Un cuadro de texto que informa quién es el vendedor, qué propiedad desea vender y a qué valor.*
    - *Dos botones que permiten comprar o no la propiedad en esas condiciones.*
  - ***Botón Propiedades:*** *Se mantiene siempre visible y a través de él se puede acceder al "Panel Propiedades".*

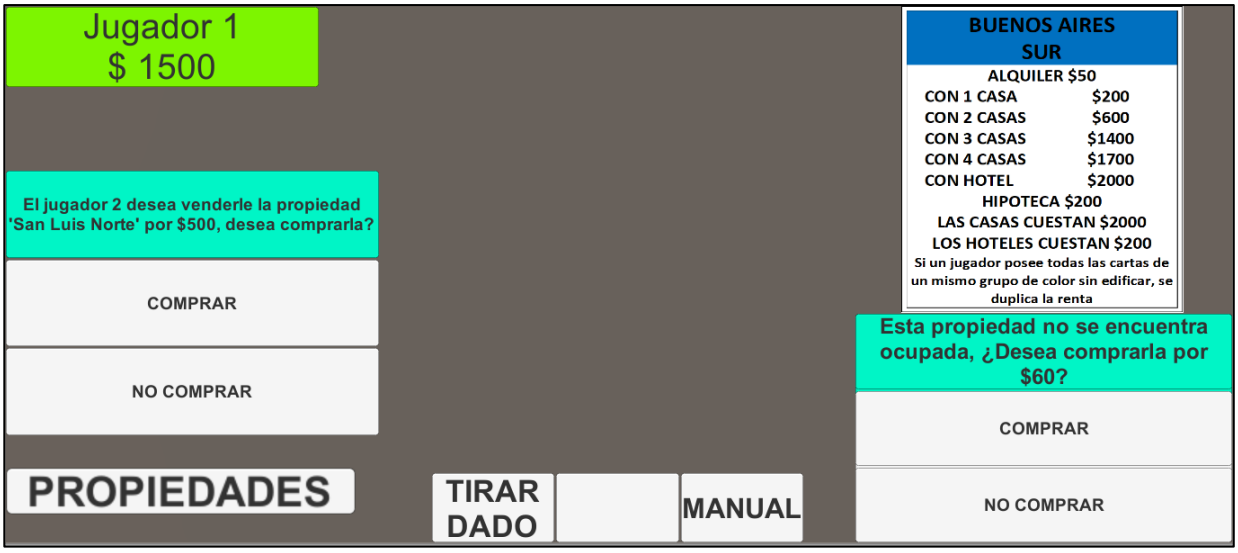

Figura 10. Representación de los objetos que conforman el Panel Principal.

- ***Panel Propiedades:*** *En un principio el panel se encuentra vacío, y a medida que el jugador adquiere propiedades, empiezan a aparecer en la pantalla. Las cartas de la misma provincia siempre se encuentran juntas, sin importar en qué orden se adquirieron.*

  Al tocar cada carta, se despliega un menú que permite hipotecar la propiedad o venderla a otro jugador.

  Cuando el jugador posee todas las propiedades de una provincia (grupo) el color de las cartas de dicha provincia se torna celeste. En esta condición es posible construir casas o un hotel e estas propiedades (se observa que en este caso el menú que se despliega permite además comprar o vender una casa u hotel según corresponda).

  Al final se observan las cartas que representan los casilleros:

  Tren del sur
  Tren del norte
  Tren del oeste
  Tren del este
  Compañía de electricidad
  Compañía de agua

  - ***Botón Salir:*** *Permite cerrar el Panel Propiedades y volver al Panel Principal.*

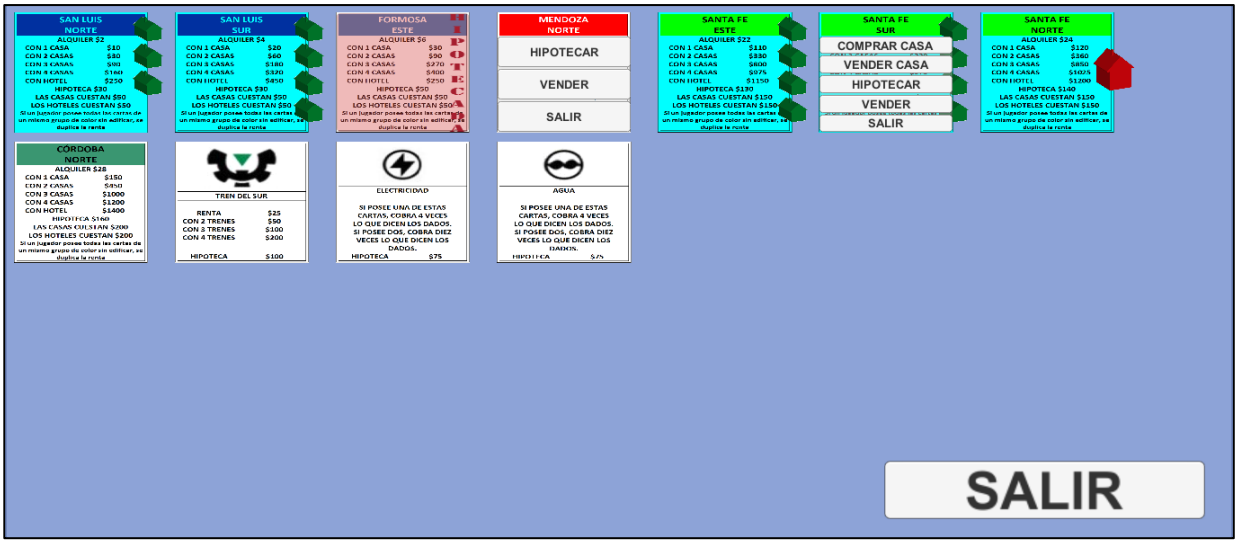

Figura 11. Representación de los objetos que conforman el Panel Propiedades..

Todos los botones que realicen una acción de compra, venta o hipoteca de propiedad, compra o venta de casas u hoteles, pago o cobro de dinero, tienen adjuntado un Script propio que ejecuta todas las acciones necesarias para realizar lo que describe cada botón.

*Los **Scripts** que se encuentran adjuntos a elementos de la Interfaz de Usuario son:*

1. ***Botón Comprar***
2. ***Botón Comprar 2***
3. ***Botón Pagar Alquiler***
4. ***Botón Vender***
5. ***Botón Casillero***
6. ***Botón Comprar Casa***
7. ***Botón Hipotecar***
8. ***Botón Tirar Dado***
9. ***Botón Vender Casa***
10. ***Panel Propiedades***

*Unity permite **dos formas** de visualizar o no objetos en una escena:*

- *Crear y destruir objetos en tiempo de ejecución: Se crea una instancia de un prefab determinado y se lo posiciona en donde se desee mediante código y cuando ya no es necesario se lo destruye y no figura más en la escena ni en la jerarquía.*

- *Activar o desactivar el objeto en tiempo de ejecución: El objeto siempre se encuentra en la jerarquía, y ya se encuentra posicionado y las características deseadas. Entonces éste se activa o desactiva cuando se desee mediante código o algún comando.*

  Por motivos de comodidad, se eligió el segundo método, de forma que todos los objetos figuran en la jerarquía y se puede trabajar sobre ellos más fácilmente, aunque debe trabajarse de forma ordenada para que no existan confusiones.

9. MODALIDAD MULTIJUGADOR

El juego se diseña de modo que puedan jugar e interactuar dos, tres o hasta cuatro jugadores en la misma partida. Para lograr esto los jugadores deben estar conectados en una red.

9.1. SERVIDOR Y ANFITRIÓN

En el sistema de red de Unity, los juegos tienen un servidor y múltiples clientes Dónde no hay un servidor dedicado, uno de los clientes juega el rol del servidor – se le llama a este cliente el ***"anfitrión (Host)"****.*

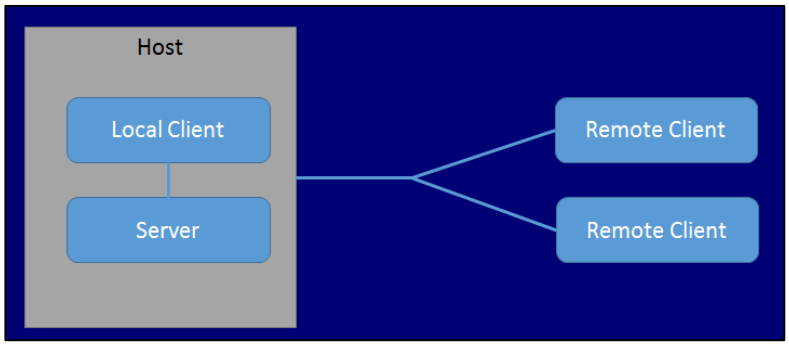

Figura 12. Sistema de red Cliente - Servidor (Host)

El anfitrión es un servidor y un cliente en el mismo proceso. El anfitrión utiliza un tipo especial de cliente llamado ***LocalClient*** *(cliente local), mientras otros clientes son* ***RemoteClientes*** *(clientes remotos). El LocalClient (cliente local) se comunica al servidor (local) a través de llamadas de función directas y colas de mensajes, ya que está en el mismo proceso. En realidad comparte la escena con el servidor. Los* ***RemoteClientes*** *(clientes remotos) se comunican con el servidor sobre una conexión regular de red.*

Una meta del sistema de red es que el código para LocalClients y RemoteClientes es el mismo, entonces los desarrolladores solo tienen que pensar acerca de un tipo de cliente la mayoría de veces.

9.2. JUGADORES, JUGADORES LOCALES Y AUTORIDAD

En el sistema de red, los objetos del jugador son especiales. Hay un objeto jugador asociado con cada persona jugando el juego, y los comandos son enrutados a ese objeto. Una persona no puede invocar comandos en un objeto jugador de otra persona - solamente en él mismo. Por lo que hay un concepto de "mi" objeto jugador. Cuando el jugador es agregado y hay una asociación hecha con una conexión, ese objeto jugador se vuelve un objeto "jugador local" en el cliente de ese jugador. Hay una propiedad ***isLocalPlayer*** *que es configurada a true, y un callback* ***OnStartLocalPlayer()*** *que es invocado en el objeto en el cliente. El diagrama de abajo muestra dos clientes y sus jugadores locales.*

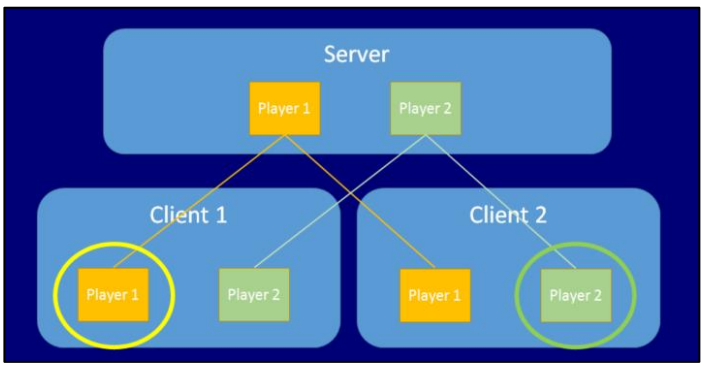

Figura 13. Ilustración de una red con autoridades locales.

Solamente el objeto jugador que es "suyo" tendrá la flag isLocalPlayer configurada. Esto puede ser utilizado para filtrar procesamiento de input, para manejar anexos de cámara, o hacer otras cosas del lado del cliente que solo deberían ser hechas para su jugador.

En adición a isLocalPlayer, un objeto jugador puede tener su local authority (autoridad local). Esto significa que el objeto jugador en el cliente de su dueño es responsable para el objeto - este tiene autoridad sobre este. Esto es utilizado más común para controlar movimiento, pero puede ser utilizado para otras cosas también. El componente ***NetworkTransform*** *entiende esto y va a enviar movimiento del cliente si esté está habilitado. El* ***NetworkIdentity*** *(Identidad de red) tiene una casilla de verificación para configurar* ***LocalPlayerAuthority*** *(Autoridad del jugador local).*

Para objetos no jugadores, tal como enemigos, no hay un cliente asociado, por lo que la autoridad reside en el servidor.

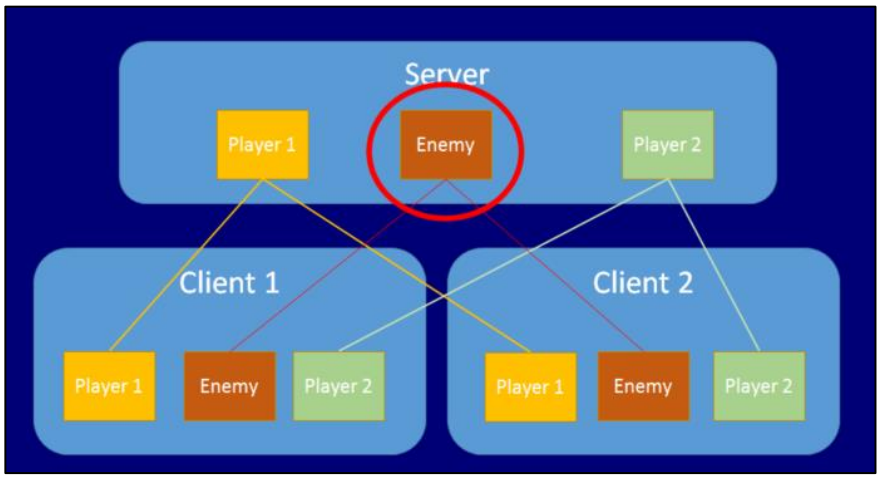

Figura 14. Ilustración de una red con autoridad del servidor sobre objetos no jugadores.

Hay una propiedad isAuthority en el ***NetworkBehaviour*** *(comportamiento de red) que puede ser utilizado para decir si un objeto tiene autoridad. Por lo que objetos no jugadores tienen autoridad en el servidor, y objetos jugadores con* ***localPlayerAuthority*** *configurado tienen autoridad en el cliente de su dueño.*

9.3. PROPIEDADES DEL CONTEXTO DE LA RED

Hay propiedades en la clase NetworkBehaviour que le permite a script saber cuál es el contexto de la red de un objeto en red en cualquier momento.

- ***isServer*** *- true si el objeto está en un servidor (o anfitrión-host) y ha sido generado.*
- ***isClient*** *- true si el objeto está en un cliente, y fue creado por el servidor.*
- ***isLocalPlayer*** *- true si el objeto es un objeto jugador para este cliente.*
- ***isAuthority*** *- true si el objeto es propiedad del proceso local*

10. CONFIGURAR UN PROYECTO MULTIJUGADOR

10.1. CONFIGURACIÓN DEL NETWORKMANAGER

En la escena Offline, se crea un objeto vacío y se le agrega el componente ***<NetworkManager>*** *(Administrador de red) y* ***<NetworkManagerHUD>*** *(Menú para administrar red). En el proyecto se creó un Script que deriva de NetworkManager y se le adicionó un Canvas personalizado compuesto por el menú y el fondo.*

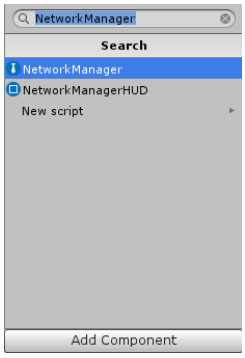

Figura 15. Network Manager.

10.2. CONFIGURACIÓN DEL PLAYER PREFAB

El siguiente paso es configurar el Unity Prefab que representa el jugador en el juego. Por defecto, el NetworkManager instancia (crea) un objeto para cada jugador al clonar el player prefab. En este caso el prefab que se instanciará será el peón acompañado del Canvas, que se describió anteriormente.

Para que el objeto jugador pueda ser identificado en la red, debe agregársele un componente ***NetworkIdentity*** *y debe configurarse la casilla de verificación* ***Local Player Authority*** *en true. Esto le permitirá al cliente controlar el movimiento del objeto jugador.*

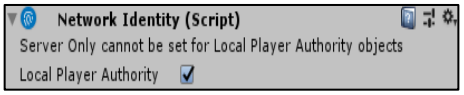

Figura 16. Network Identity.

10.3. REGISTRAR EL PLAYER PREFAB

Se debe indicar al NetworkManager qué prefab es el que se va a crear al iniciar un jugador nuevo. Para eso se debe arrastrar el prefab Peón a la ranura Player Prefab del componente NetworkManager.

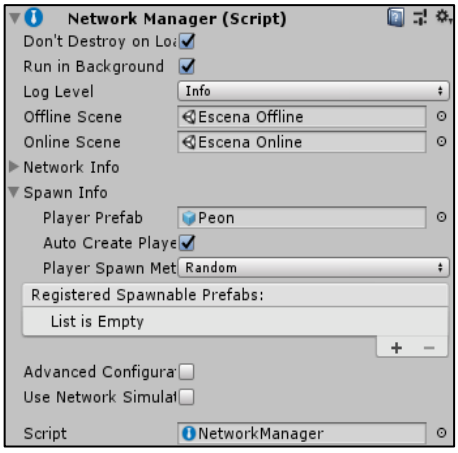

Figura 17. Network Manager.

*Para que el movimiento del peón esté sincronizado en la red, es decir, que los otros jugadores vean que el peón se mueve, se le debe agregar el componente <****NetworkTransform****>. El slider de la variable* ***Network Send Rate*** *se coloca al máximo para que la sincronización es casi inmediata.*

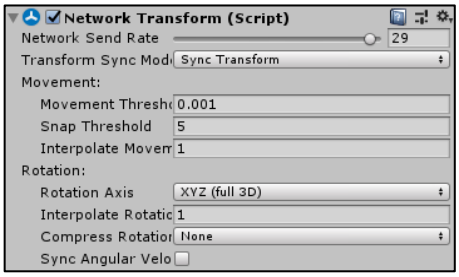

Figura 18. Network Transform.

11. ACCIONES REMOTAS

*El sistema de red tiene maneras de realizar acciones a través de la red. Este tipo de acciones a veces se llama **Remote Procedure Calls**. Hay dos tipos de **RPCs** en el sistema de red, **Commands** (comandos) - que son llamado del cliente y corren en el servidor; y **ClientRpc calls** - que son llamados en el servidor y corren en los clientes.*

El diagrama de abajo muestra las direcciones que las acciones remotas toman:

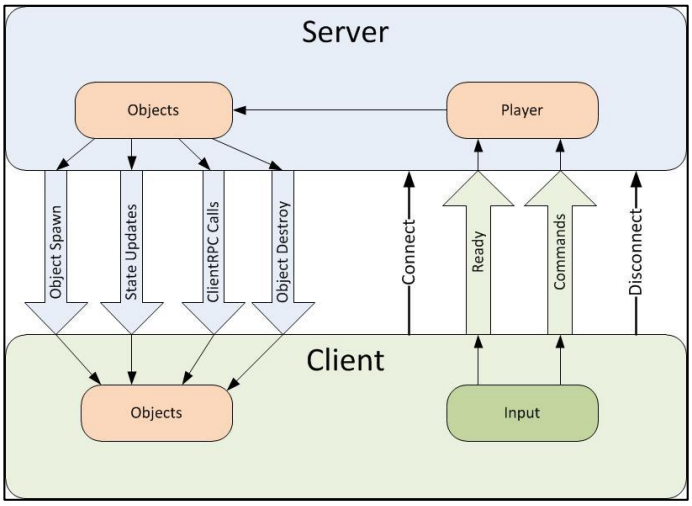

Figura 19. Acciones remotas entre Clientes y el Servidor.

11.1. COMMANDS

*Los comandos son enviados de objetos jugadores en el cliente a objetos jugadores en el servidor. Por seguridad, **Commands** solamente pueden enviarse desde SU objeto jugador, por lo que usted no puede controlar los objetos de otros jugadores. Para hacer una función un comando, agregue el atributo personalizado **[Command]** a este, y agregue el prefijo **Cmd**. Esta función ahora va a correr en el servidor cuando sea llamada en el cliente. Cualquier argumento será automáticamente pasado al servidor con el comando.*

*Las funciones de comandos deben tener prefijos con "**Cmd**". Esta es una pista cuando lea código que llame el comando - esta función es especial y no es invocada localmente como una función normal.*

Se debe tener cuidado en enviar comandos del cliente cada frame. Esto puede causar mucho tráfico en red.

11.2. LLAMADOS CLIENTRPC

*Los llamados del **ClientRpc** son enviadas de objetos en el servidor a objetos en clientes. Estos pueden ser enviados de cualquier objeto servidor con un NetworkIdentity que fue generado. Ya que el servidor tiene autoridad, entonces no hay problemas de seguridad con los objetos del servidor siendo capaces de enviar estas llamadas. Para hacer una función a una llamada **ClientRpc**, agregue el atributo personalizado **[ClientRpc]** a este, y agregue el prefijo **Rpc**. Esta función ahora va a ejecutarse en clientes cuando sea llamado en el servidor. Cualquier argumento va a automáticamente ser pasado a los clientes con un llamado ClientRpc.*

*Las funciones ClientRpc deben tener un prefijo "**Rpc**". Esta es una pista cuando lea código que llame el método - esta función es especial y no es invocada localmente como una función normal.*

Dos de las acciones remotas más importantes son:

- ***Modificar la matriz de información del Script del Tablero:*** *Cuando un jugador adquiere una propiedad, la hipoteca, la vende, o compra o vende una casa, debe informarlo modificando la matriz, para que los otros jugadores puedan saberlo.*

  De esta forma, cuando se realiza una de las acciones, el jugador invoca la función ***CambiarValor(int Casillero , int Parámetro , int Valor)***

  Donde:

  - ***Casillero:*** *Es el casillero al cual se le va a modificar un parámetro.*
  - ***Parámetro:*** *Es el parámetro específico que se desea modificar.*
  - ***Valor:*** *Es el valor que adquirirá el parámetro.*

  Se crean dos funciones en el Script del Peón:

```
[ClientRpc]
public void RpcCambiarValor(int Casillero, int Parametro, int Valor)
{
    Tablero tablero;
    tablero = GameObject.Find("Tablero").GetComponent<Tablero>();
    tablero.Valores[Casillero, Parametro] = Valor;
}
```

Figura 20. Función RpcCambiarValor().

Esta función es enviada del Servidor a los jugadores Clientes (tanto los remotos como el local) para que cada uno de ellos la ejecute y modifique el valor en el objeto Tablero que posee.

```
[Command]
public void CmdCambiarValor(int Casillero, int Parametro, int Valor)
{
    RpcCambiarValor(Casillero, Parametro, Valor);
}
```

Figura 21. Función CmdCambiarValor().

Esta función es enviada desde el jugador Cliente que ejecuta la acción, al servidor. Lo que le indica es que les envíe la misma función a los otros clientes para que de esta forma todos modifiquen el mismo valor.

En la siguiente figura, el Jugador 2 llama a la función Cmd y le dice al servidor que llame a la función Rpc y ejecute la función ***CambiarValor()*** *en todos los clientes.*

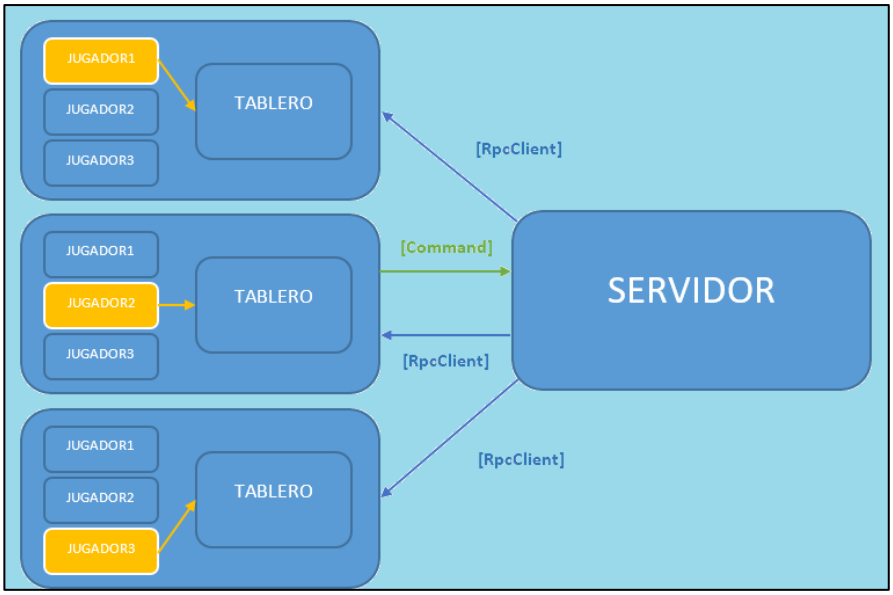

Figura 22. Ejemplo de comunicaciones mediante acciones remotas.

- ***Actualizar dinero:*** *Otra de las funciones fundamentales es que cuando un jugador incremente o disminuya su dinero, le informe al resto para que todos estén actualizados. Esto se realiza de manera similar al caso anterior:*

```
[Command]
public void CmdActualizarDinero(int Peon, int NuevoDinero)
{
    RpcActualizarDinero(Peon, NuevoDinero);
}
```

Figura 23. Función CmdActualizarDinero().

```
[ClientRpc]
public void RpcActualizarDinero(int JugadorCobra, int NuevoDinero)
{
    GameObject.Find("Tablero/Jugador" + JugadorCobra).GetComponent<Jugador>().Dinero=NuevoDinero;
}
```

Figura 24. Función RpcAcualizarDinero().

12. ESCENA OFFLINE

*Una vez terminada la escena Online, se procede a configurar la escena Offline, la cual contendrá una <**Camera**>, una **Directional Light**, y el **NetworkManager** configurado anteriormente. Como hijo del NetworkManager se crea un Canvas, el cual contendrá el menú con el que el usuario se conecta como servidor o como cliente, debiendo ingresar el número de IP y el número de puerto que utilizará. Al crear una partida (servidor) o unirse a una (cliente) automáticamente se desactiva la escena Offline y se activa la escena Online, y se crea el objeto jugador (peón) correspondiente. La siguiente figura representa la pantalla que se visualiza al iniciar el juego:*

Figura 25. Representación de la Escena Offline.

13. REALIDAD AUMENTADA

El objetivo de implementar realidad aumentada es que los jugadores, con un dispositivo móvil, apunten hacia un elemento activador al cual se denomina ***Marcador****, y cuando éste sea detectado se despliegue sobre él el tablero con los peones y las cartas de "suerte" y "servicio a la comunidad". Adicionalmente, se agregarán cuatro marcadores que contendrán información sobre los cuatro jugadores, y al ser detectados por la cámara, desplegarán dicha información, incluyendo el casillero en el que se encuentra, el dinero que posee y las propiedades que ha adquirido (considerando el estado de construcción).*

14. VUFORIA

Vuforia es un kit de desarrollo de software de realidad aumentada (SDK) para dispositivos móviles que permite la creación de aplicaciones de realidad aumentada . [1] Utiliza la tecnología Computer Vision para reconocer y rastrear imágenes planas (objetivos de imagen) y objetos 3D simples, como cuadros, en tiempo real. Esta capacidad de registro de imágenes permite a los desarrolladores ubicar y orientar objetos virtuales , como modelos 3D y otros medios, en relación con imágenes del mundo real cuando se visualizan a través de la cámara de un dispositivo móvil. El objeto virtual luego rastrea la posición y orientación de la imagen en tiempo real para que la perspectiva del espectador en el objeto se corresponde con su perspectiva en el objetivo de imagen, por lo que parece que el objeto virtual es una parte de la escena del mundo real.

Las características adicionales del SDK incluyen detección de oclusión localizada usando 'Botones virtuales', selección de objetivos de imágenes en tiempo de ejecución y la capacidad de crear y reconfigurar conjuntos de objetivos mediante programación en tiempo de ejecución.

Figura 26. Logo de Vuforia.

14.1. ARMANDO UN JUEGO DE REALIDAD AUMENTADA

*Para poder utilizar el SDK de Vuforia, es necesario crear una cuenta en la página https://developer.vuforia.com ya que Unity nos pide una clave de activación. Una vez creada la cuenta, se ingresa y se elige la pestaña **Develop**, luego la pestaña **Target Manager** y se presiona el botón **Add Database**.*

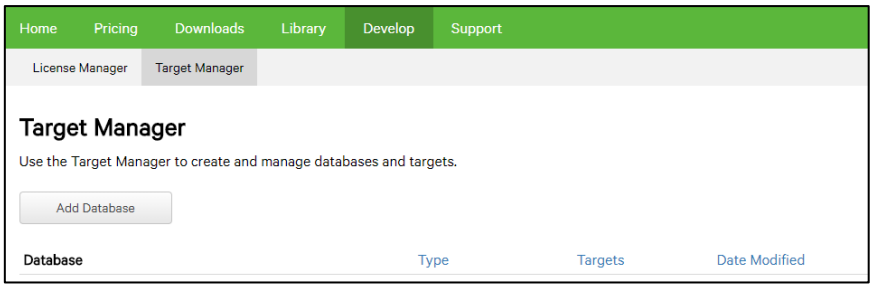

Figura 27. Página de Vuforia.

*Al presionar el botón se despliega un cuadro en donde se asigna el nombre de la Base de datos y se debe seleccionar el casillero **Device**.*

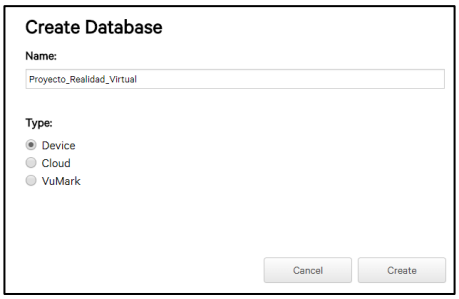

Figura 28. Crear Base de datos.

*Una vez creada la base de datos, se ingresa y se presiona el botón **Add Target** para crear un nuevo marcador. Se despliega un cuadro en donde se puede seleccionar qué tipo de marcador se desea crear (si se desea que la cámara reconozca una imagen simple, una imagen que cubre un cubo, un cilindro, o un cuerpo 3D). Se selecciona la primera opción, luego se busca el archivo de la imagen que tendrá el marcador, y se coloca el tamaño y el nombre que tendrá dentro de Unity.*

Figura 29. Pasos para crear un marcador.

*Hay tres reglas principales que un **marcador** debe cumplir:*

- *Debe ser rico en detalle.*
- *Debe tener un buen contraste, con regiones brillantes y oscuras.*
- *No puede haber patrones repetitivos.*

*En el fondo, **Vuforia** crea un arreglo de la imagen usando sus características, y entonces el algoritmo puede encontrar tales patrones y seguir las blancos. Aproximadamente, una característica de una imagen es un ángulo afilado, como una esquina de la caja o la punta de una estrella. La cantidad de características de una imagen está directamente conectada a su **trackability**.*

14.2. INCORPORACIÓN DE MARCADORES EN UNITY

*Una vez creados los cinco marcadores, se descarga la base de datos presionando el botón **Download Database** y se importa desde Unity ingresando en la barra de herramientas y seleccionado **Assets** → **Import Package** → **Custom Package**.*

Se debe seguir el siguiente procedimiento:

- *Se descarga e instala el complemento de Vuforia para Unity.*
- *Se habilita el complemento ingresando a **File** → **Build Settings** → **Player Setting** y luego en el inspector ingresando a **XR Settings**, se tilda el casillero **Vuforia Augmented Reality**.*
- *Se incorpora una **<AR Camera>** a la escena ingresando en la pestaña de barra de herramientas **GameObject** → **Vuforia** → **AR Camera**. Se debe eliminar la cámara que se encontraba anteriormente para evitar inconvenientes.*
- *En la página de Vuforia, se selecciona la pestaña **Develop**, luego la pestaña **License Manager** y se copia el código que aparece.*
- *Nuevamente en Unity, se selecciona la AR Camera y en el inspector se presiona **Open Vuforia configuration**.*
- *En la ranura **App License Key** se pega el código de la licencia.*
- *Se tilda el casillero **Load Object Targets**.*

*Una vez realizado lo anterior ya se puede colocar los marcadores en la escena. Se incorpora un marcador a la escena ingresando en la pestaña de barra de herramientas **GameObject→Vuforia→Image**.*

*Se selecciona el marcador y en el inspector se debe elegir en **Database** el nombre de la base de datos anteriormente importada, y en **Image Target** el nombre del marcador correspondiente. Luego se debe posicionar el tablero y el marcador en la posición relativa que se desea que aparezcan en el juego, y una vez logrado esto, se debe hacer el tablero hijo del marcador, para que uno siga el movimiento del otro, como lo muestra la figura a continuación:*

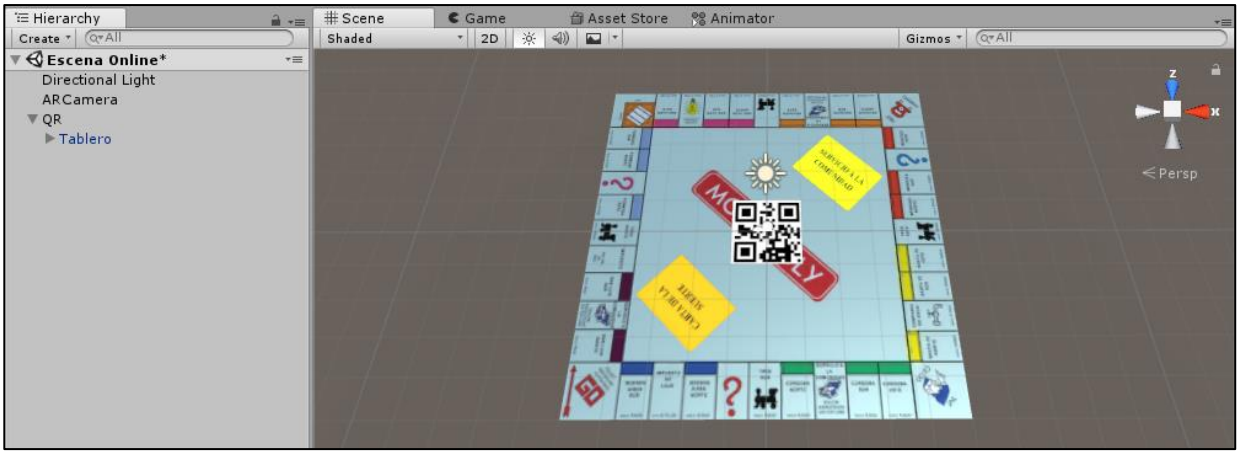

Figura 30. Posicionamiento relativo entre marcador y tablero.

*Por último se crean 4 Canvas con la información de cada uno de los jugadores. En el inspector se debe colocar la variable **Render Mode** en **Screen Space – Override** para que el Canvas esté fijo a la cámara, sin importar la posición del marcador. Luego se agregan los cuatro marcadores de forma análoga al del tablero, y se hace cada Canvas hijo de su correspondiente marcador.*

*La jerarquía que resulta en la Escena Online es la que se observa en la figura, en donde se aprecian los cinco marcadores con su objeto asociado, la AR Camera que es la que se encarga de obtener y procesar la información. El último objeto **<EventSystem>** aparece automáticamente cuando se agrega un Canvas a la escena, y se encarga de captar todos los eventos relacionados con la interfaz de usuario.*

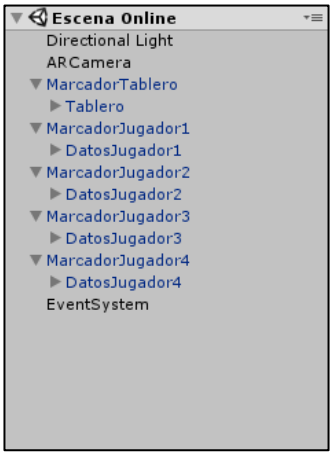

Figura 31. Jerarquía final de la escena Online.

Las imágenes que se utilizan en los marcadores son códigos QR ya que estos cumplen perfectamente con las tres condiciones necesarias planteadas anteriormente. Los marcadores que despliegan la información de los jugadores han sido modificados para lograr diferenciarlos, y el marcador del tablero se diferencia por el tamaño. Como en el juego cada marcador tiene una escala relativa con respecto al objeto que despliega, mientras más grande sea el tamaño del marcador, mayor será el tamaño del objeto que se despliegue.

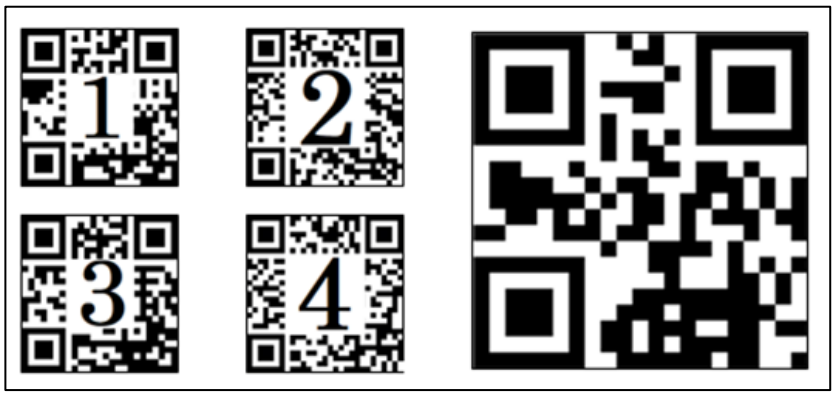

Figura 32. Imagen de los marcadores utilizados.

15. COMPILACIÓN, EXPORTACIÓN E INSTALACIÓN

Una vez terminado el programa, se debe compilar para construir la aplicación y crear el ejecutable que permitirá jugar en la PC fuera del programa y en los dispositivos móviles.

*Se debe ingresar en **File → Build Settings** en donde Unity permite elegir la plataforma a la que queremos exportar la aplicación. Se puede alternar entre ellas seleccionando la deseada y presionando **Switch Plataform**.*

*Antes de construir la aplicación, se debe seleccionar las escenas que se incluirán. Como se observa, en primer lugar se coloca la **Escena Offline** y luego la **Escena Online**. Una vez completado esto, se presiona **Build** y el programa generará la aplicación para la plataforma elegida.*

*Presionando **Player Settings** se puede configurar las características básicas de la aplicación, como el nombre, el ícono, la resolución, etc.*

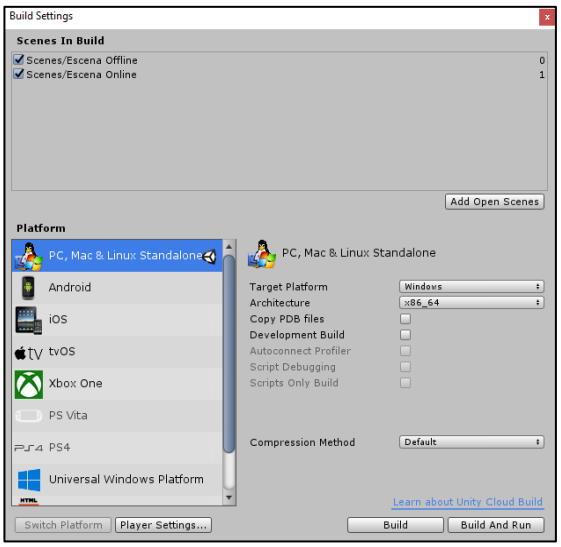

Figura 33. Cuadro de configuración para exportar la aplicación.

16. RESULTADOS

Se presenta a continuación un ejemplo del resultado final que se obtiene. Se han creado dos versiones distintas del juego:

- *Compatible con Android: Esta es la versión oficial que implementa todos los componentes mencionados en el informe.*
- *Compatible con Windows: En esta versión del juego no se utiliza realidad aumentada, de forma que se pueda jugar desde la PC como un juego tradicional. Esto permite a los usuarios elegir en qué plataforma jugar, evitando la necesidad de apuntar en todo momento a los marcadores*

Los jugadores de una misma partida pueden elegir cualquier plataforma y conectarse entre sí sin inconvenientes.

*Todos los jugadores inician en el casillero GO con un monto de $1500. Las siguientes imágenes representan una partida en la que ya se han jugado varios turnos. El **Jugador 1 (host)**, que se encuentra jugando desde la PC, es representado por el peón rojo, que se encuentra en el casillero 9 (El casillero GO no se cuenta) y ya ha adquirido propiedades.*

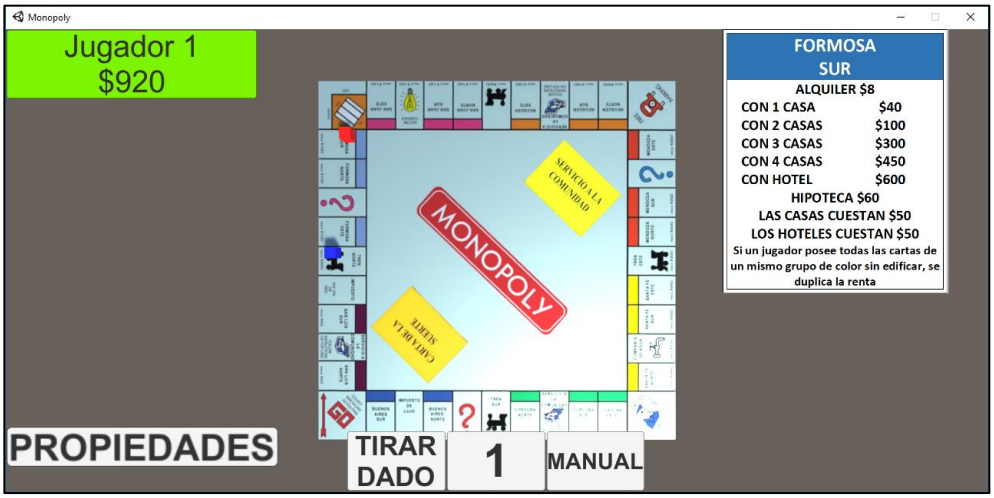

Figura 34. Jugador 1 jugando desde la PC.

*Para observar las propiedades que ha comprado a lo largo del juego, el **Jugador 1** presiona el botón **PROPIEDADES** y se accede a la siguiente pantalla:*

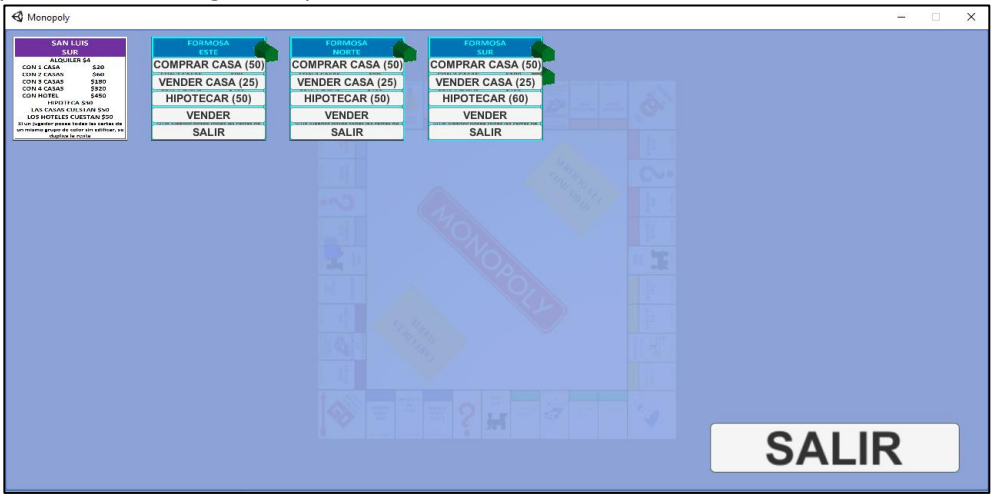

Figura 35. Propiedades del Jugador 1.

*El **Jugador 2 (Cliente)**, que se encuentra jugando desde su Smartphone, es representado por el peón azul, que se encuentra en el casillero 5 (El casillero GO no se cuenta) y también ha adquirido propiedades.*

Figura 36. Jugador 2 jugando desde su Smatphone.

*Igual que en el caso anterior, para observar las propiedades que ha comprado a lo largo del juego, el **Jugador 2** presiona el botón **PROPIEDADES** y se accede a la siguiente pantalla:*

Figura 37. Propiedades del jugador 2.

*A continuación, el **Jugador 2** procede a colocar en pantalla los cuatro marcadores que despliegan la información de los cuatro jugadores en simultáneo con el tablero, iniciando con el marcador del Jugador 1:*

Figura 38. Información disponible sobre el Jugador 1.

De igual manera se procede con el resto de los marcadores:

Figura 39. Información disponible sobre el Jugador 2.

Figura 40. Información disponible sobre el Jugador 3.

Figura 41. Información disponible sobre el Jugador 4.

Toda la documentación necesaria para replicar el proyecto se puede encontrar en el siguiente link:

http://xurl.es/RV-Fragapane

17. BIBLIOGRAFÍA

[1] ¿En qué consiste la realidad aumentada?

https://blogthinkbig.com/en-que-consiste-la-realidad-aumentada

[2] ¿Qué es la realidad aumentada?

http://realidadaumentada.info/tecnologia/

[3] Qué es y cómo funciona la realidad aumentada

https://es.ccm.net/faq/30104-que-es-y-como-funciona-la-realidad-aumentada

[5] Unity (motor de juego)

https://es.wikipedia.org/wiki/Unity_(motor_de_juego)

[6] Conceptos del Sistema de Red

https://docs.unity3d.com/es/current/Manual/UNetConcepts.html

[7]Presentación de la serie - Multijugador en red con Unity

https://www.youtube.com/watch?v=zDVqgUyejmA&list=PLREdURb87ks1vxMHSsM4tRWpE1GTLaz_1

[8] Ejemplo básico de realidad aumentada con Unity3D y Vuforia utilizando un FrameMarker.

https://www.youtube.com/watch?v=XML_AY8FMIk

[9] Unity Unet Commands And ClientRpcs Explained

https://www.youtube.com/watch?v=9VW7ctwvNok

[10] Vuforia License Manager

https://developer.vuforia.com/license-manager

[11] Unity 3D

https://unity3d.com/es