VIDEOJUEGO MULTIJUGADOR BASADO EN REALIDAD AUMENTADA

TRABAJO FIN DE MÁSTER 2012-2014

Con orientación Profesional

Realizado por: Javier López

Dirigido por: Ramón Mollá

rmolla@dsic.upv.es

Valencia, 2014

Agradecimientos

En primer lugar dar gracias a Ramón por su paciencia, sus conocimientos y su buena voluntad

A mis compañeros de Máster por tantos momentos compartidos y el apoyo incondicional que siempre me han brindado.

A mi familia por haber aguantado todo el proceso a mi lado y haber estado apoyándome en todo momento.

TABLA DE CONTENIDO

ÍNDICE DE FIGURAS

1.INTRODUCCIÓN

Los videojuegos siempre han sido un área de gran interés dentro las aplicaciones gráficas en tiempo real. De hecho es uno de los sectores relacionados con el desarrollo de software que ha tenido mayor auge en los últimos años.

Por otra parte, la velocidad con la que se han extendido los nuevos dispositivos móviles, y la potencia de cálculo con la que cuentan han hecho que a día de hoy estemos viviendo una revolución que está influyendo de manera directa tanto en el desarrollo de los videojuegos como en el uso de tecnologías como la realidad aumentada.

En el siguiente trabajo final de máster se van a utilizar y analizar diferentes tecnologías, para posteriormente integrarlas en una aplicación gráfica en tiempo real, en forma de videojuego para dispositivos móviles. El componente fundamental sobre el que se basa la aplicación a desarrollar es la Realidad Aumentada, de manera que será una de las principales tecnologías sobre las que se trabajara en la realización de este trabajo.

Otra parte importante del proyecto es el desarrollo de un módulo que permite la interconexión de dispositivos Android a través de Bluetooth. Este módulo se integrará dentro de la aplicación para que los objetos dentro de la escena de realidad aumentada sean capaces de compartir información a través del envío y recepción de datos. De este modo, se persigue obtener una experiencia utilizando Realidad Aumentada de un modo más dinámico, interactivo y novedoso.

Por otro lado, siempre me he sentido atraído por el desarrollo de videojuegos, y por el planteamiento de la aplicación, es perfectamente factible integrar las diferentes tecnologías implicadas en el desarrollo de un videojuego. Así que en el siguiente trabajo, el desarrollo de la aplicación seguirá las directrices aplicadas al desarrollo de un mini juego, con sus pertinentes mecánicas de juego, etc.

Durante los últimos años son cada más vez los dispositivos que permiten la utilización de tecnologías que hasta ahora se encontraban restringidos a ámbitos de uso menos lúdicos y recreativos. En por ello que he escogido la realizar el desarrollo del videojuego para que sea posible ejecutarlo en la plataforma Android. Está decisión es algo que condicionará el desarrollo de ciertas partes específicas dentro del módulo de interconexión por Bluetooth.

Antes de comenzar con el desarrollo de la aplicación se realizará una etapa de análisis del estado del arte, que será necesario para realizar un planteamiento correcto del desarrollo, así como analizar los requisitos de la aplicación.

La segunda etapa del trabajo comprenderá el desarrollo del módulo de interconexión inalámbrica a través de Bluetooth. Por último, una vez finalizada esta etapa, seguiremos con el desarrollo del videojuego, integrando la tecnología de Realidad Aumentada y el módulo de interconexión dentro de la aplicación.

1.1 – REALIDAD AUMENTADA

La **realidad aumentada** (RA) es el término[20] que se usa para definir una visión directa o indirecta de un entorno físico del mundo real, cuyos elementos se combinan con elementos virtuales para la creación de una realidad mixta en tiempo real. Consiste en un conjunto de dispositivos que añaden información virtual a la información física ya existente, es decir, añadir una parte sintética virtual a lo real. Esta es la principal diferencia con la realidad virtual, puesto que no sustituye la realidad física, sino que sobreimprime los datos informáticos al mundo real.

Con la ayuda de la tecnología (por ejemplo, añadiendo la visión por computador y reconocimiento de objetos) la información sobre el mundo real alrededor del usuario se convierte en interactiva y digital. La información artificial sobre el medio ambiente y los objetos puede ser almacenada y recuperada como una capa de información en la parte superior de la visión del mundo real.

La realidad aumentada de investigación explora la aplicación de imágenes generadas por ordenador en tiempo real a secuencias de vídeo como una forma de ampliar el mundo real. La investigación incluye el uso de pantallas colocadas en la cabeza, un display virtual colocado en la retina para mejorar la visualización, y la construcción de ambientes controlados a partir sensores y actuadores.

Recientemente, el término realidad aumentada se ha difundido por el creciente interés del público en general. La Realidad Aumentada puede ser considerada como una tecnología intermedia entre la Realidad Virtual y la telepresencia[6]. Mientras que en la RV[23] el entorno es completamente sintético y la telepresencia es completamente real, en la RA el usuario ve el mundo real aumentado con objetos virtuales.

Existen tres importantes aspectos involucrados en el diseño de un sistema de Realidad Aumentada: la combinación de el mundo real y el mundo virtual, la interactividad en tiempo real y el registrado en 3D.

A parte de los tres aspectos mencionados, existe otro que también podría incorporarse a la lista, portabilidad. En casi todos los sistemas de realidad virtual el usuario no puede moverse demasiado por el entorno debido a las limitaciones de los dispositivos. Sin embargo, algunas aplicaciones de Realidad Aumentada necesitan que el usuario camine a través de entornos grandes. Por tanto, la portabilidad se convierte en un aspecto importante. En este tipo de aplicaciones, el registrado 3D se vuelve todavía más complejo.

La cantidad de ámbitos en los que el uso de la Realidad Aumentada es verdaderamente amplio, algunos de los sectores en los que es conocido el uso de la Realiadad Aumentada son por ejemplo: publicidad, sector médico y quirúrgico, ocio, juegos, educación y cada día se abren más posibilidades para su aplicación.

Figura 1 Escena de Realidad Aumentada

Gracias a la difusión masiva de dispositivos móviles e inteligentes, se está produciendo un cambio en el paradigma dentro de la computación, lo que se denomina como computación sensible al contexto. Dentro de este nuevo paradigma el uso de la Realidad Aumentada surge como uno de los pilares más importantes para explotar las capacidades de estos nuevos dispositivos. Cada vez salen más dispositivos a la luz cuya principal finalidad es complementar la información del usuario utilizando nuevas interfaces persona-computador, como es el caso de las novedosas gafas que varios fabricantes como Google han puesto a la venta recientemente.

1.2 – VIDEOJUEGOS Y DISPOSITIVOS MÓVILES

Si existe un mercado que se ha visto favorecido por la difusión de los dispositivos móviles ese ha sido el de los videojuegos. La existencia de las tiendas de aplicaciones y la facilidad para distribuir los desarrollos de manera global han convertido el mercado de los videojuegos para dispositivos móviles en un campo de gran atractivo. Es por eso que estamos ante un campo de las ATGR sobre los que esta habiendo un mayor crecimiento. Aunque los primeros dispositivos que aparecieron no gozaban de una potencia de cálculo destacable, actualmente estamos alcanzando un punto en el que estos dispositivos cuentan con unas especificaciones que no tienen nada que envidiar a algunas de las plataformas típicas de videojuegos.

Actualmente dadas las circunstancias favorables del mercado, han emergido numerosas herramientas y motores de videojuegos que son capaces de exportar un mismo proyecto a las diferentes plataformas existentes sin necesidad de cambiar el código. También existe un interés creciente en realizar todos estos videojuegos lo más sociales posibles, de manera que se puedan compartir logros y objetivos con nuestros contactos en las principales redes sociales actuales, lo que hace que la adicción y diversión de este tipo de juegos sea mayor aún si cabe.

1.3 – CONECTIVIDAD EN DISPOSITIVOS MÓVILES

El mercado de los dispositivos móviles comenzó a expandirse velozmente en 2010 y hoy en día todavía sigue creciendo. Las capacidades de estos dispositivos son muy extensas, con ellos se puede mandar emails, navegar por internet, descargar y ejecutar aplicaciones. El cambio sustancial es que estos dispositivos permiten la creación de aplicaciones en conjunción con el sistema operativo y los kits de desarrollo de software (SDK) proporcionados tanto por terceras empresas como por los fabricantes de los dispositivos. A parte de las redes de tercera y cuarta generación (3G/4G) que proporcionan los operadores móviles, los dispositivos móviles incorporan varias tecnologías inalámbricas de corto alcance. Estas capacidades proporcionan gran libertad al usuario en la manera que se comunican con el entorno y mejoran la usabilidad global del dispositivo.

En el siguiente trabajo se van a describir algunas de las tecnologías inalámbricas de corto alcance más importantes y presentes en los actuales dispositivos móviles.

1.3.1 – WI-FI

Wi-Fi es una marca de la Wi-Fi Alliance la organización comercial que adopta, prueba y certifica que los equipos cumplen los estándares 802.11 relacionados a redes inalámbricas de área local.

Esta nueva tecnología surgió por la necesidad de establecer un mecanismo de conexión inalámbrica que fuese compatible entre distintos dispositivos. Buscando esa compatibilidad fue que en 1999 las empresas 3Com, Airones, Intersil, Lucent Technologies, Nokia y Symbol Technologies se reunieron para crear la Wireless Ethernet Compatibility Alliance, o WECA[19], actualmente llamada Wi-Fi Alliance. El objetivo de la misma fue designar una marca que permitiese fomentar más fácilmente la tecnología inalámbrica y asegurar la compatibilidad de equipos.

De esta forma, en abril de 2000 WECA[19], certifica la interoperabilidad de equipos según la norma IEEE 802.11b[7], bajo la marca Wi-Fi. Esto quiere decir que el usuario tiene la garantía de que todos los equipos que tengan el sello Wi-Fi pueden trabajar juntos sin problemas, independientemente del fabricante de cada uno de ellos.

En el año 2002 la asociación WECA[19] estaba formada ya por casi 150 miembros en su totalidad. La familia de estándares 802.11 ha ido naturalmente evolucionando desde su creación, mejorando el rango y velocidad de la transferencia de información, su seguridad, entre otras cosas.

1.3.2 – WI-FI DIRECT

Las redes Wi-Fi domésticas más comunes están formadas por una conexión a un proveedor de acceso a Internet, un punto de acceso o router, y ordenadores u otros dispositivos con capacidad de conectarse a una red (clientes). La mayoría de las redes Wi-Fi están configuradas en "modo infraestructura", en el que el punto de acceso actúa como un hub al que se conectan el resto de dispositivos. Los dispositivos no pueden comunicarse directamente y el punto de acceso siempre hace de intermediario entre ellos.

En cambio, los dispositivos Wi-Fi Direct pueden funcionar como cliente o como punto de acceso. Cuando se establece una conexión por primera vez entre dispositivos Wi-Fi Direct los dispositivos negocian y determinan automáticamente cuál de ellos será el punto de acceso.

Puesto que se han incrementado el número y el tipo de dispositivos que pueden unirse a sistemas Wi-Fi, el modelo básico de un enrutador simple con computadores inteligentes se ha vuelto cada vez más limitante. Al mismo tiempo, la creciente sofisticación de los puntos de acceso ha hecho que surjan problemas de configuración para los usuarios. Para manejar estos problemas, ha habido numerosos intentos de simplificar ciertos aspectos de la tarea de configuración.

Un ejemplo común es el sistema Wi-Fi Protected Setup que se encuentra incorporado en muchos puntos de acceso desde 2007 cuando el estándar fue presentado.2 Wi-Fi Protected Setup permite que un punto de acceso pueda ser configurado introduciendo un número PIN u otra identificación dentro de una pantalla de conexión, o en algunos casos, simplemente presionando un botón. Este sistema utiliza dicha información para enviar datos a un computador, suministrarle la información necesaria para completar la configuración de red y conectarse a Internet. Desde el punto de vista del usuario, un simple clic reemplaza a la configuración de múltiples pasos, distinta para cada dispositivo, y con manejo de tecnicismos que se requería realizar anteriormente.

Mientras que el modelo Protected Setup funcionaba como se había planeado, sólo lo hacía para simplificar la conexión entre el punto de acceso y los dispositivos que hicieran uso de sus servicios, ante todo para acceso a Internet. Proporcionaba escasa ayuda para conexión a redes internas, por ejemplo para encontrar y configurar el acceso a una impresora desde un ordenador. Para manejar estos roles se había desarrollado un número de protocolos diferentes, entre los cuales están Universal Plug and Play (UPnP), Devices Profile for Web Services (DPWS), y Bonjour de Apple. Estos protocolos permiten a los dispositivos buscar a otros dispositivos dentro de la red, obtener sus capacidades, y proveer algún nivel de configuración automática.

1.3.3 – BLUETOOTH

Bluetooth es una especificación industrial para Redes Inalámbricas de Área Personal (WPAN) que posibilita la transmisión de voz y datos entre diferentes dispositivos mediante un enlace por radiofrecuencia en la banda ISM de los 2,4 GHz [9]. Los principales objetivos que se pretenden conseguir con esta norma son:

- Facilitar las comunicaciones entre equipos móviles.
- Eliminar los cables y conectores entre éstos.
- Ofrecer la posibilidad de crear pequeñas redes inalámbricas y facilitar la sincronización de datos entre equipos personales.

Los dispositivos que con mayor frecuencia utilizan esta tecnología pertenecen a sectores de las telecomunicaciones y la informática personal, como PDA, teléfonos móviles, computadoras portátiles, ordenadores personales, impresoras o cámaras digitales.

El estándar Bluetooth está actualmente evolucionando hacia las versiones de nueva generación. La versión 3.0, que obtiene un rendimiento de hasta 24Mb/s, y la versión 4.0 que utiliza la funcionalidad Bluetooth Low Energy, que reduce significativamente el consumo de energía.

La versión 3.0 incluye el "control de energía mejorado", que previene de desconexiones breves y elimina el problema de la pérdida de enlace con los headsets. También soporta "conexión de datos unicast", que simplifica el proceso de negociación entre dispositivos. Esta función reduce el tiempo necesario para establecer una conexión y comenzar a usar el dispositivo. Bluetooth 3.0 elimina varios puntos débiles que mejoran la usabilidad y la fiabilidad.

1.3.4 – NFC

Near field communication (NFC) es una tecnología de comunicación inalámbrica, de corto alcance y alta frecuencia que permite el intercambio de datos entre dispositivos. Los estándares de NFC cubren protocolos de comunicación y formatos de intercambio de datos, y están basados en ISO 14443 (RFID, radio-frequency identification) y FeliCa.1 Los estándares incluyen ISO/IEC 180922 y los definidos por el NFC Forum, fundado en 2004 por Nokia, Philips y Sony, y que hoy suma más de 170 miembros.

Como en ISO 14443, NFC se comunica mediante inducción en un campo magnético, en donde dos antenas de espira son colocadas dentro de sus respectivos campos cercanos. Trabaja en la banda de los 13,56 MHz, esto hace que no se aplique ninguna restricción y no requiera ninguna licencia para su uso.

Soporta dos modos de funcionamiento, todos los dispositivos del estándar NFCIP-1 deben soportar ambos modos:

- Activo: ambos dispositivos generan su propio campo electromagnético, que utilizarán para transmitir sus datos.
- Pasivo: sólo un dispositivo genera el campo electromagnético y el otro se aprovecha de la modulación de la carga para poder transferir los datos. El iniciador de la comunicación es el encargado de generar el campo electromagnético.

El protocolo NFCIP-1 puede funcionar a diversas velocidades como 106, 212, 424 o 848 Kbit/s. Según el entorno en el que se trabaje, las dos partes pueden ponerse de acuerdo de a que velocidad trabajar y reajustar el parámetro en cualquier instante de la comunicación.

1.4 – MOTIVACIÓN

La idea de realizar este trabajo final de Máster surge principalmente de mi inquietud por el desarrollo de videojuegos, además de poder utilizar esta temática como hilo conductor para unificar otras tecnologías de las que se nos ha proporcionado conocimiento durante el transcurso del Máster. Partiendo de esta idea parece interesante incorporar algunas tecnologías que ya son interesantes de manera aislada, y unificarlas para poder ofrecer una experiencia que potencie las capacidades de las AGTR dotándolas de más dinamismo e interactividad.

Aunque el trabajo propuesto está estrechamente relacionado con la línea del Máster de Imagen Digital, hay partes, como por ejemplo la de Realidad Aumentada que también tiene nexos en común con la parte de visión por computador y reconocimiento de formas. Así como el comportamiento de los personajes con la Inteligencia Artificial. Por lo tanto, se deduce que el trabajo también tiene como motivación el uso y aplicación práctica de todos estos conocimientos.

El apostar por enfocar el desarrollo principalmente en dispositivos móviles, y en concreto a la plataforma Android, se debe al gran potencial que ofrecen estos dispositivos para el uso de tecnologías de la denominada computación ubicua. Esto permite utilizar las características de la cámara, conexiones inalámbricas y todo tipo de sensores para mejorar la experiencia y la sensación por parte de los usuarios.

Actualmente se estima que existen alrededor de 1700 millones de tablets y smartphones en todo el mundo, siendo Android la plataforma más extendida también a nivel mundial.

Hoy en día es cada vez más común ver como este tipo de dispositivos están ganando el relevo a los antiguos ordenadores de sobremesa en ciertos ámbitos en los que su uso resulta más rápido e intuitivo. Otro motivo importante es que la idea se plantea como una posible tesina conjunta con otros dos compañeros que compartimos interés por la misma temática. De este modo surge la necesidad de realizar un trabajo en el que la orientación sea de carácter profesional, y de poder realizar un desarrollo a nivel de producto.

1.5 – OBJETIVO

El principal objetivo del trabajo es poder crear una AGTR interactiva, en este caso un videojuego, que haga uso de la Realidad Aumentada como principal escenario donde transcurre la acción, y en el que sea posible realizar interactuar con dos jugadores mediante la conexión inalámbrica de dos dispositivos. Para conseguir este objetivo será también necesario realizar un análisis del estado del arte de cuáles son las herramientas más apropiadas para trabajar desarrollando videojuegos en los que están implicados multitud de factores. Es por ello que se hace una revisión de cuáles son las herramientas dentro de apartados como la conectividad, la realidad aumentada y los motores de videojuegos, entre otros aspectos.

Todo esto nos proporcionará una perspectiva lo más global posible y que hará posible valorar que herramientas y enfoques son los más apropiados para el desarrollo del trabajo.

Un objetivo también importante es que la aplicación resultante sea capaz de ejecutarse en la plataforma Android, lo que permite aportar más dinamismo al resultado final, así como aportar el conocimiento necesario para futuros desarrollos sobre esta plataforma.

Por último otro objetivo a cumplir es el desarrollar y exponer cuáles son los pasos necesarios para integrar ciertas características como la Realidad Aumentada y la conexión inalámbrica de dispositivos utilizando el motor de videojuegos Unity3D[22]. De manera que sirva de ayuda para futuros alumnos que quieran derivar sus trabajos en alguno de los aspectos que se ven implicados en este trabajo.

1.6 - APORTACIONES

La principal aportación es la capacidad de unificar varias tecnologías ya interesantes de por sí, para crear una aplicación con una interacción y un dinamismo en el que el usuario interactúa con su contexto al mismo tiempo que se divierte. Para ello es necesario realizar un análisis profundo de varias tecnologías, revisando las capacidades que puede ofrecer y las necesidades que puede cubrir con función de cumplir el objetivo planteado para el trabajo.

Aunque hay ciertas partes en las que se utilizan partes o herramientas ya desarrolladas, existe una parte importante en algunos de los módulos de la aplicación que han sido desarrollados para este trabajo, de manera que quedan como APIs abiertas a disposición de futuros alumnos para su mejora y el añadido de funcionalidades.

Otra aportación es realizar este proceso pensando en un dispositivo móvil como plataforma destino, y poder analizar hasta que punto un dispositivo de estas características es capaz de manejar todo los requerimientos a nivel de CPU y demás recursos.

2. ESTADO DEL ARTE

2.1 – MOTORES DE VIDEOJUEGOS

La programación de un motor de videojuegos por completo puede convertirse en una tarea realmente compleja. Más aún si queremos que nuestro futuro juego pueda ser ejecutado en el mayor número de plataformas posibles. Hoy en día pocas empresas tienen suficientes recursos como para poder desarrollar un motor propio. Por suerte, en los últimos años han proliferado una gran variedad de motores, algunos de ellos de uso gratuito, que permiten a los desarrolladores centrarse en el diseño y programación del propio videojuego. Esto ha abierto las puertas del desarrollo de videojuegos a tanto pequeñas y medianas empresas como a desarrolladores particulares o "indies".

A continuación se van a revisar algunos de los motores de videojuegos más populares en la actualidad, además de resaltar sus virtudes y defectos respectivamente. Dado que existen multitud de motores, en está revisión nos vamos a centrar en los motores que permiten el desarrollo multiplataforma para dispositivos móviles.

2.1.1 – Cocos2d-x

Cocos2d-x es un motor de videojuegos open source escrito en C++. Puede ser utilizado tanto para crear juegos, aplicaciones o programas interactivos multiplataforma.

El renderizador de Cocos2d-x está optimizado para gráficos 2D con OpenGL. Cocos2d-x adopta un diseño basado en una cola de renderizado.

En cuánto al rendimiento soporta auto-batching, auto-culling y caching de transformaciones, por lo que afirman que los juegos corren a una velocidad de entre 1 y 20 veces más. A parte de esto, la optimización conseguida permite la ejecución más rápida del motor de videojuegos, lo que permite su uso en dispositivos Android de gama baja.

Cocos2d-x también incorpora las últimas características de C++11, por lo que el desarrollador puede hacer uso de las novedades incorporadas a esta versión: expresiones lambda, overrides, threads, inferencia de tipos, smart pointers, etc.

2.1.2 – STARLING

Starling es un motor de videojuegos multiplataforma basado en la tecnología de Adobe Flash y Action Script. Al igual que Cocos2d-x se trata de un proyecto gratuito y de código abierto. Aún así la empresa Adobe soporta el desarrollo de Starling y planea integrarlo en su ecosistema de herramientas de desarrollo. Starling cuenta con una extensa comunidad de desarrolladores que junto con el compromiso de Adobe ofrece buenas garantías a los desarrolladores.

Algunas de las principales características de Starling son las siguientes:

- **Multiplataforma:** Starling permite la publicación de videojuegos en una gran variedad de plataformas. En la actualidad Starling soporta las plataformas Android, iOS, Windows, Mac OS X y navegadores web. Esto es debido a que está basado en Stage3D, la plataforma de Adobe que utiliza aceleración por hardware y principal competidor de WebGL.
- **Rendimiento:** En el desarrollo de Starling se ha considerado la optimización como algo muy importante. Esto hace que se reduzca al mínimo el uso de memoria, CPU y GPU. Todo esto es proporcionado por el propio motor, de manera que el desarrollador no tiene que preocuparse en profundidad de estos asuntos.
- **Organización de la escena jerárquica:** Al igual que en la mayoría de motores y herramientas de desarrollo de aplicaciones gráficas, Starling organiza los objetos de la escena siguiendo una estructura jerárquica.
- **Sistema de eventos:** El sistema de eventos de Starling está basado en el sistema de eventos de ActionScript.
- **Soporte de texturas:** El sistema para carga de texturas de Starling permite la carga de una gran cantidad de formatos (incluyendo el nuevo formato de Adobe ATF), crear subtexturas, transformar coordenadas de texturas o tintarlas de colores. A parte de esto también soporta el uso de atlas de texturas, lo que reduce el coste de GPU de la aplicación.
- **Soporte multiresolución:** Starling es capaz de utilizar el conjunto de texturas en tiempo de ejecución dependiendo de las necesidades del dispositivo en el que se ejecuta la aplicación.
- **Tweens:** Starling contiene su propio sistema de tweening. Lo que permite animar las propiedades de los objetos utilizando diferentes curvas de transición.
- **Extensibilidad:** Starling ha sido construido para que sea fácil de entender, modificar y extender.

A parte de todo lo mencionado, Starling también cuenta con herramientas para tareas concretas como la gestión de servidores, animación por huesos, serialización, etc, que se pueden integrar dentro del desarrollo de las aplicaciones de manera sencilla.

2.1.3 – CORONA SDK

Corona SDK es un motor de desarrollo de videojuegos que se basa en el lenguaje de scripting Lua. La API extensiva de Corona permite hacer prácticamente cualquier cosa, desde animación a interconexión a través de la red, con unas pocas líneas de código. Tanto si se utiliza para el desarrollo de juegos como de aplicaciones de negocios, se pueden ver los cambios al instante con el Simulador de Corona y permite progresar rápidamente.

Corona SDK también permite exportar para las principales plataformas móviles. Corona permite publicar para iOS, Android, Kindle Fire y NOOK utilizando el mismo código base, y próximamente soportará las plataformas Windows Phone 8 y Windows 8.

La plataforma de Corona esta construida utilizando los estándares de la industria, incluyendo OpenGL, OpenAL, Box2D, Facebook, SQLite y muchos más.

El nuevo motor, basado en OpenGL 2.0, permite realizar efectos cinemáticos con unas pocas líneas de código. Algunas de las características más interesantes de este motor es que permite aplicar toda clase de filtros y efectos sobre las imágenes renderizadas, permite simular el efecto de profundidad sobre videojuegos en dos dimensiones utilizando la perspectiva para generar el mismo efecto óptico.

Corona SDK cuenta con multitud de plugins para facilitar tareas específicas como por ejemplo conexión a redes sociales, rankings de puntuaciones, etc. También cuenta con un IDE propio y un simulador que permite la comprobación inmediata del código para facilitar la depuración.

La plataforma Corona SDK, al contrario que otros motores solo se puede utilizar mediante el pago de alguna de las licencias disponibles en su página web. El precio varia desde los 16 a los 200€ mensuales, dependiendo de las necesidades del desarrollador.

2.1.4 – UNITY 3D

Unity 3D[22] es sin duda uno de los motores para videojuegos más populares de la actualidad. Unity es un ecosistema para el desarrollo de videojuegos: un potente motor de renderizado totalmente con un completo conjunto de herramientas y que permite flujos de trabajo ágiles para crear contenido interactivo tanto en 3D como en 2D. Además cuenta con ventajas como: la publicación multiplataforma, multitud de assets disponibles en la Asset Store y una amplia comunidad de desarrolladores. Tanto para los desarrolladores independientes como para los estudios, Unity esta derribando las barreras de tiempo y coste para crear videojuegos únicos.

Unity3D cuenta con un editor propio en el que se pueden crear y gestionar escenas complejas de manera rápida y cómoda. Además permite la iteración rápida gracias a su modo de ejecución instantánea y sus rápidos tiempos de compilación.

Por otra parte Unity3D permite añadir detalles de una gran calidad a los videojuegos: luces, efectos especiales, audio, etc. Esto permite crear juegos que muchas veces alcanzan unas cotas de calidad similares a las de juegos de los denominados AAA o grandes producciones.

Otra importante característica es que Unity no solo permite la creación de videojuegos en 3D, sino que también esta pensando para poder realizar desarrollos 2D sin ningún tipo de problema. El número de formatos que Unity3D acepta a la hora de importar recursos es muy extenso, a parte de que integra ciertos aspectos para facilitar aún más si cabe este proceso.

Unity cuenta también con su propio sistema de animaciones llamado Mecanim. Mecanim permite dotar de vida a personajes de una manera natural y fluida. Debido a que esta integrado de forma nativa dentro del motor, Mecanim evita la necesidad de realizar esfuerzos integrando software de terceros. Mecanim incorpora funcionalidades como máquinas de estados, blend trees, animaciones para músculos entre otros. Otra funcionalidad muy útil relacionada con las animaciones es la capacidad de generar rigs de huesos de manera automática. El sistema de rig automático puede ser utilizado para incluso añadir animaciones a los músculos en tiempo de ejecución.

En la parte de rendimiento Unity también ofrece numerosas características que facilitan la optimización y la depuración de nuestro desarrollos. Unity cuenta con un profiler incorporado con el que controlar por ejemplo la reserva de memoria a nivel de sistema. Este profiler se puede utilizar desde dentro de los scripts, de manera que se puede indagar que secciones del código pueden causar problemas. También permite el realizar análisis durante el transcurso del juego, conectando directamente al dispositivo a través de la red local.

Siguiendo con la parte de rendimiento, Unity incluye algunas mejoras importantes en este aspecto, un ejemplo es su solución de occlusion culling desarrollada junto con Umbra Software, que asegura que solo aquellos objetos vistos por la cámara son mandados a renderizar. El sistema de occlusion culling de unity funciona en móviles, web y consolas con una sobrecarga mínima en tiempo de ejecución.

En el apartado de plataformas compatibles, Unity es probablemente el motor que cuenta con un mayor número de plataformas compatibles. Unity brinda soporte actualmente a las plataformas: PS3, PS4, Xbox360, WiiU, iOS, Android, Blackberry, Windows Phone 8, Windows 8, Linux, Mac y navegadores web.

El proceso de exportación es bastante simple, basta con seleccionar la plataforma deseada y darle a build and run. Además podemos utilizar la vista de juego integrada para visualizar rápidamente los cambios que vamos realizando durante el desarrollo.

Otro aspecto interesante que no todos los motores de videojuegos poseen y que Unity sí, es la capacidad para trabajar de manera colaborativa con su propia herramienta de control de versiones. La herramienta encargada de integrar el la gestión de assets y el control de versiones se denomina Unity Asset Server.

Si a todo lo mencionado anteriormente sumamos la cantidad de plugins que se pueden encontrar en la tienda que integra Unity, tenemos una de las herramientas más potentes y versátiles para el desarrollo de videojuegos de todo tipo. Además con una curva de aprendizaje bastante suave y con multitud de documentación y de soporte por parte de la comunidad de desarrolladores.

Por último mencionar que Unity cuenta con una versión gratuita que está algo limitada, si queremos exportar para determinadas plataformas tendremos que pagar alrededor de unos 1500 para obtener la versión Pro.

2.2 - SISTEMAS DE REALIDAD AUMENTADA

En los últimos años, debido a la versatilidad y el potencial en la utilización de la Realidad Aumentada, han surgido multitud de herramientas que permiten la creación y gestión de sistemas de realidad aumentada. Gran parte de la culpa de este éxito se debe a la proliferación de nuevos dispositivos que permiten aplicar técnicas de Realidad Aumentada en contextos de gran interés para el usuario.

A continuación se va a hacer un recorrido por las principales herramientas para crear contenidos de realidad aumentada que pueden ser utilizadas en Unity3D. En este caso me he centrado en las más importantes, ya que existen herramientas que he considerado no tienen el suficiente interés como para ser mencionadas en este estudio.

2.2.1 – HERRAMIENTAS DE REALIDAD AUMENTADA EN UNITY3D

13thlab Point Cloud

La herramienta de la empresa 13th Lab denominada PointCloud cuenta con soporte para ser utilizada dentro del motor Unity3D. La empresa 13th Lab es una pequeña start up especializada en la visión por computador en dispositivos móviles. Uno de los puntos fuertes de la herramienta Point Cloud es que cuentan con la primera implementación del algoritmo SLAM (Simultaneous Localization And Mapping) para dispositivos móviles. Esta técnica permite detectar características de una escena en tiempo real y hacer uso de las capacidades de la realidad aumentada sin utilizar marcas.

Para hacer uso de la herramienta dentro de Unity es necesario importar el package que se puede encontrar en su página web, eso sí, de momento solo cuentan con una versión compatible para dispositivos iOS.

Metaio SDK

Metaio SDK es otra de las herramientas que cuentan con un plugin para Unity3D entre otras muchas plataformas. Este SDK es uno de los mas completos frameworks entre aquellos que cuentan con plugin para Unity3D.

Éstas son algunas de las principales características de Metaio SDK:

- Calidad de la visualización mejorada mediante shaders.
- Hasta 6 veces mayor velocidad de renderizado
- Mas robustez y velocidad en tracking 2D (SDK Basic), 3D y tracking SLAM (SDK Pro)
- Tracking y renderizado Multi-hilo.
- Soporte para modelos 3D complejos (mas de 32 mil polígonos)
- Depurado y manejo fácil del contenido 3d
- Compatible con iOS 7
- Soporte para Metaio Continuous Visual Search
- Nuevas plantillas y contenido de ejemplos incluído
- Soporte para renderizado estereoscópico
- Nuevos efectos de renderizado: profundidad de campo, blur...

Uno de los puntos fuertes de este software es que cuenta con un gran apoyo por parte de la comunidad de desarrolladores y empresas, que junto con los creadores de la aplicación ponen a disposición de los desarrolladores guías, seminarios y multitud de ejemplos que ayudan a sacar el máximo partido a la herramienta. Además también cuenta con la posibilidad de realizar tracking sin utilización de marcas con la opción Metaio SLAM. En este caso van un paso mas allá y dado que estos mapas 3d se pueden guardar y cargar dentro de cualquier aplicación, también disponen de una herramienta interactiva para crear los mapas de puntos 3d que pueden ser luego utilizados en nuestras aplicaciones. También cuenta con reconocimiento en la nube, donde podemos subir nuestras imágenes para ser utilizadas como marcas. Por otra parte, el punto negativo es que la licencia pro, que permite eliminar la marca de agua y acceder a las funciones realmente interesantes como el SLAM, cuesta alrededor de los 5000 €.

Qualcomm Vuforia

Esta es sin duda la herramienta mas popular dentro del desarrollo de aplicaciones y juegos compatibles con Unity3D. Vuforia es un software desarrollado por la empresa de fabricación de microprocesadores Qualcomm. Vuforia cuenta con algunas características muy interesantes como los "Virtual Buttons", que permiten utilizar partes de la imagen utilizada como marca a modo de botones. También dispone de un Manager para gestionar los targets y para poder utilizar el reconocimiento de cualquier imagen que subamos a la plataforma. Esto lo explicaré con mas detalle mas adelante.

Dada la calidad y robustez de esta herramienta y que su utilización es totalmente gratuita, esta herramienta es la que cuenta con una mayor comunidad de desarrolladores y empresas que la respaldan y hacen uso de ella. Además de todos los ejemplos disponibles también cuenta con un foro de preguntas y respuestas muy útil para compartir y solucionar problemas. Quizás la única cosa que se hecha en falta es la implementación del algoritmo SLAM para el tracking sin utilización de marcas, algo con lo que si cuentan otras herramientas.

Al igual que Metaio permite el uso de shaders, gestión de la oclusión de objetos, marcas cilíndricas, marcas múltiples, playback de vídeo, etc.

Vuforia cuenta con soporte para los entornos de desarollo Eclipse,[21] Xcode y Unity y se encuentra actualmente en su versión 2.87. Existen multitud de juegos que han tenido una gran repercusión que han sido desarrollados con Vuforia, como por ejemplo Logo Connect, la aplicación del libro de los Record Guinness entre otros.

ARToolkit

La conocida biblioteca ARToolkit cuenta actualmente con una extensión para ser integrada en Unity. Actualmente la licencia pertenece a ARToolWorks, que dispone de una breve documentación sobre como utilizar la herramienta bajo Unity. ARToolkit siempre ha sido un software muy utilizado en entornos académicos, pero en este caso parece que se han quedado un poco atrás con respecto a los demás competidores.

Tabla Comparativa

	13th Lab Point Cloud	Metaio	Vuforia	ARToolkit
Documentación	Escasa	Extensa	Muy extensa	Muy escasa
Comunidad	Pequeña	Grande	Muy grande	Pequeña
Robustez	Alta	Alta	Alta	Baja
SLAM	Si	Si	No	No
Licencia	Pago	Pago (5000€)	Gratuita	Pago

2.2.2- SISTEMAS DE RECONOCIMIENTO SIN MARCAS

En el siguiente apartado se van a exponer algunas de las técnicas más actuales en lo referente a la posibilidad de reconocer una escena obteniendo las características directamente del entorno, sin necesidad de utilizar marcas durante ninguna de las partes del proceso de tracking y reconocimiento.

Existen trabajos al respecto, la mayoría de ellos utilizan técnicas de visión artificial basadas en técnicas S.L.A.M (Simultaneous Lozalization and Mapping). La principal idea detrás de esta técnica es la generación de mapas de entornos de desconocidos mientras que se mantiene un tracking de la posición de la máquina simultáneamente en el entorno físico. Es una técnica que ha sido utilizada típicamente para facilitar el movimiento de robots de manera autónoma como rovers[5] ,etc.

En lo que respecta al uso de variantes de técnicas SLAM aplicadas al reconocimiento y tracking de escenas de Realidad Aumentada existen trabajos, en los que por ejemplo se utilizan variantes que utilizan threads separados[1] para poder separar el tracking y el mapping, el método de mapping se basa en la obtención de líneas epipolares de las fotogramas claves obtenidos por la cámara. Posteriormente los mismos autores realizaron una versión simplificada que utiliza menos puntos en el mapa generado para poder ejecutar el algoritmo en un iPhone [3], ya que uno de los principales obstáculo a la hora de utilizar las técnicas SLAM en dispositivos móviles es que tienen un coste computacional elevado.

Aunque aparentemente la idea de no utilizar parece una solución que otorga gran flexibilidad al usuario, al no tener que depender de un soporte físico como las marcas, todavía existen algunos impedimentos a la hora de implementar estas técnicas en productos para el usuario final. Como se menciona en [4] muchas veces resulta demasiado complicado para el usuario el hecho de calibrar y posicionar la cámara de manera correcta en una escena compleja.

Por tanto se podría concluir en este sentido que aunque incluso soluciones comerciales que integran las técnicas SLAM para la detección y el tracking sin marcas, el sistema no es lo suficientemente robusto y rápido como para integrarlo en determinadas aplicaciones sin necesidad de afrontar problemas de usabilidad,etc.

2.3 CONECTIVIDAD: HERRAMIENTAS EN UNITY3D

Hoy en día uno de los puntos que se ha vuelto muy importante dentro de los videojuegos para dispositivos móviles es su carácter social. Las opciones con las que un usuario cuenta a la hora de jugar con otros usuarios y compartir los sus logros y puntuaciones son cada vez más amplias.

Unity3D cuenta con diferentes herramientas que pueden ser utilizadas a modo de plugin para facilitar la integración de las capacidades multijugador y on-line dentro del desarrollo de videojuegos.

A continuación se va a hacer un análisis de algunas de las herramientas más interesantes a la hora de plantear el desarrollo de un videojuego multijugador utilizando el motor de videojuegos Unity3D.

2.3.1 QUALCOMM ALLJOYN

AllJoyn es un proyecto open source inicialmente desarrollado por Qualcomm Innovation Center, Inc y alojado por AllSeen Alliance, que proporciona un conjunto de software y librerías universales para la interconexión de dispositivos indistintamente de la plataforma y del hardware sobre el que se desarrolle.

Los fabricantes pueden, de este modo, ofrecer productos y servicios interoperables que permitirán a los usuarios conectar diferentes dispositivos desde un punto de vista diferente y novedoso. El rango de productos en los que AllJoyn puede funcionar esta limitado únicamente por la innovación y volutad de las empresas.

Desde dispositivos móviles hasta electrodomésticos y productos multimedia en los hogares, hasta la electrónica de los coches o el equipamiento de oficina.

Permite la interoperabilidad horizontal

- Multiplataforma, independiente del fabricante y de las capas físicas.
- Interoperable entre los principales sistemas operativos para sistemas embebidos.
- Los dispositivos pueden interaccionar independientemente de cómo estén conectados, Wi-Fi, Ethernet, etc.

En la actualidad AllJoyn dispone de un SDK para Unity, pero también está disponible para las plataformas Android, iOS, Mac OS X, Windows y Linux.

Una de las mayores ventajas de AllJoyn frente a otros frameworks es que permite la interconexión de dispositivos de manera muy sencilla y prácticamente transparente para el programador.

2.3.2 PHOTON CLOUD

Photon Cloud es uno de los frameworks más populares para el desarrollo de videojuegos multijugador. Cuenta con soporte para varios motores de videojuegos, entre ellos Unity3D. La plataforma de Photon Cloud, gracias a su escalabilidad, es apropidada tanto para desarrolladores indies como para desarrolladores de juegos AAA.

Algunas de las características más interesantes de este plugin son las siguientes:

- Tiempo real: Photon Cloud está disponible de manera global para garantizar una baja latencia y el menor tiempo de ida y vuelta en videojuegos multijugador alrededor del mundo.
- Multijugador: Photon ofrece la fiabilidad necesaria en los ratios de envío y recepción de mensajes. Su sistema SaaS es una buena elección, independientemente de que protocolo utilices: TCP, HTTP, Websockets o RUDP.
- Multiplataforma: El SDK de Photon Cloud permite ser utilizado en varias plataformas, ya que Unity3D tiene soporte para la mayoría de ellas: Android, iOS, Windows Phone, etc.
- Alta escalabilidad: La escalabilidad automática de distribución mundial de Photon Cloud permite la conexión de miles de usuarios sin ningún tipo de problema.
- API de emparejamiento: Photon permite emparejar usuarios tanto de manera aleatoria como utilizando filtros en los juegos. Se pueden crear habitaciones a las que los usuarios pueden decidir unirse de manera intuitiva.
- Personalizable: El SDK facilitado por Photon Cloud permite la adaptabilidad y personalización para los mas variados casos de uso que el desarrollo de un videojuego multijugador pueda tener.

Photon Cloud cuenta con diferentes tipos de licencia dependiendo de los requerimientos de cada aplicación. Por una parte parte existe la cuenta gratuita, que permite la utilización del servicio para desarrollo de pruebas y pequeños juegos indie. Por otro lado, debido a su enfoque cloud y a su escalabilidad, permite que el plan de precios se ajuste al uso real que los usuarios hacen de la aplicación o videojuego bajo demanda.

2.4 - CONCLUSIONES

En los apartados anteriores se ha realizado un análisis de herramientas y tecnologías que será de utilidad a la hora de afrontar el desarrollo del videojuego.

Una vez analizados los diferentes motores de videojuegos que proporcionan características compatibles con la idea del desarrollo inicial, he decidido utilizar Unity3D como motor de videojuegos. Cuenta con muchas ventajas con respecto a otros competidores, documentación, plugins, extensibilidad, entre muchas otras.

En lo que respecta a la parte de la conectividad se han analizado varias herramientas. Dado que el dentro del objetivo de la aplicación existe el requisito de que la aplicación sea compatible con el mayor número de dispositivos posible, he decidido implementar el módulo multijugador haciendo uso de la API del bluetooth nativa que proporciona el SDK de Android.

El motivo de esta decisión es que haciendo uso de la herramienta AllJoyn existían importantes restricciones a la hora de utilizar la conexión Bluetooth en Android, ya que el estándar del protocolo que utilizan necesita que el dispositivo sobre el que se ejecute la aplicación tenga permisos root. Esto implica tener que desbloquear el gestor de arranque del teléfono, a parte de otras operaciones engorrosas. La mayoría de usuarios de Android no tiene su teléfono rooteado, es por eso que decidí desestimar la opción de usar AllJoyn como herramienta. Alljoyn proporciona la alternativa de utilizar WiFi, el problema es que también restringe la conexión únicamente a dispositivos que estén conectados a la misma red. En el caso del Wifi punto a punto o Wifi-Direct, no es necesario que ambos dispositivos estén conectados dentro de la misma red, pero es necesario que ambos dispositivos incluyan esta característica a parte de tener instalada la versión 4.0 Android Ice Cream Sandwich como mínimo.

Por otra parte, en el caso de utilizar Photon Cloud se requiere que los dispositivos que se quieren conectar estén conectados a una red de datos 3G. Es por esto que la implementación se realizará finalmente utilizando la plataforma proporcionada por Android a nivel nativo. De esta manera el número de dispositivos que pueden utilizar la aplicación es mucho más amplio, ya que no es necesario tener conexión de datos o Wi-fi y prácticamente la mayoría de dispositivos hoy en día incorporan bluetooth.

En cuánto a la parte que implica el uso de Realidad Aumentada, he optado por finalmente utilizar Vuforia. Después de hacer una revisión de las diferentes herramientas, Vuforia es sin duda la que cuenta con una comunidad, documentación y robustez mayor. La única alternativa que puede hacer algo de sombra es Metadio SDK, pero el precio de su licencia resulta prohibitivo. Otras opciones como ARToolkit son interesantes pero no son integrables con Unity3D a través de plugins, lo que hubiera supuesto un incremento del tiempo muy grande dentro del desarrollo de la aplicación.

2.5 – PROPUESTA DE MEJORA

Una vez analizados todas las tecnologías se han visto cuales son las herramientas que mas se adecuan al objetivo perseguido por el trabajo. Por tanto se ha decidido utilizar el plugin que proporciona Vuforia para Unity3D para realizar la parte de visualización y Realidad Aumentada, Unity3D como motor de videojuegos para toda la parte de mecánica e interacción y por último realizar un plugin utilizando las librerías de Android nativas e integrándolo dentro del plugin de Vuforia.

Para ello será necesario conocer técnicamente a fondo las plataformas sobres las que se va a realizar el desarrollo del trabajo. En este caso será necesario investigar y dominar el uso de la programación de aplicaciones nativas en Android y de conocimientos generales de programación de videojuegos como pueden ser gestión de colisiones, operaciones vectoriales etc.

La principal propuesta del proyecto consiste en realizar una integración de tecnologías que muy pocas veces suelen complementarse para lograr una experiencia de uso novedosa. Uno de los aspectos que se propone es la mejora de la experiencia de un entorno de realidad aumentada, haciendo uso de la conectividad con otros dispositivos. Por otro lado este enfoque proporciona una mejora en la interacción con el entorno y con otros usuarios, ya que la información se comparte entre varios dispositivos.

3. METODOLOGÍA Y DESARROLLO

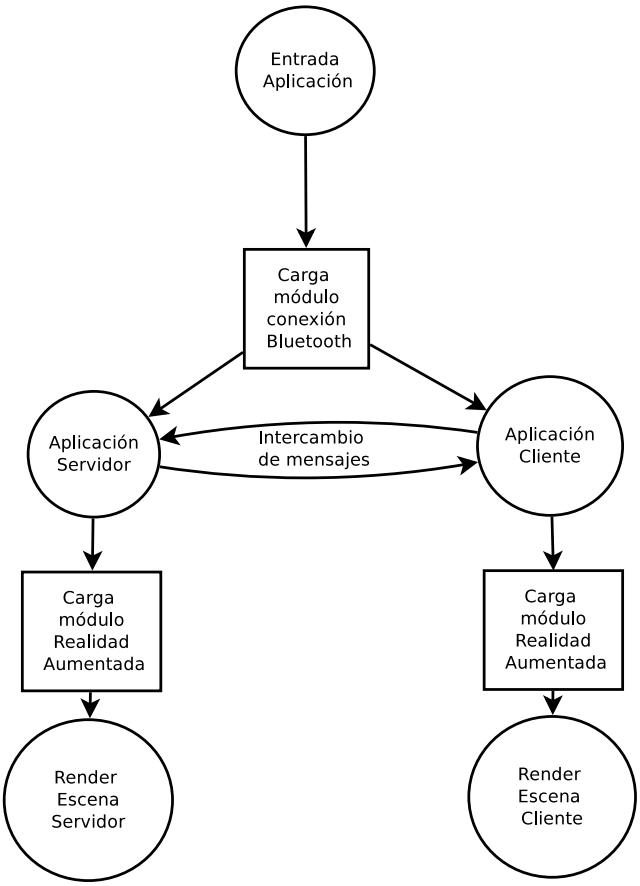
En el siguiente apartado se van a tratar los principales puntos relaciones con el proceso de desarrollo de la aplicación así como los detalles de la implementación.

El desarrollo del videojuego sigue la idea de poder instanciar dos personajes animados en un escenario que será renderizado en función de la marca que visualicemos a través de la cámara de nuestro dispositivo. Una vez los dos jugadores están conectados pueden mover sus avatares por el escenario virtual que se acaba de generar y combatir contra su adversario.

El diseño de la aplicación se ha planteado como una arquitectura cliente-servidor, en la que ambas partes conectan de modo directo. Esta separación viene impuesta debido a que el hecho de que estemos utilizando la interconexión para compartir información hace necesario que existan los roles de cliente que conecta y servidor que queda escuchando peticiones de clientes para conectar.

A partir de aquí y siguiendo este planteamiento inicial, también se divide la aplicación en dos escenas diferentes aunque de contenido visual idéntico, para que dependiendo de si vamos a ejecutar el código del cliente o del servidor sea sencillo separar los diferentes comportamientos de cada uno de las partes.

A continuación se muestra un gráfico de cuál sería el diseño resultante para la aplicación:



El gráfico representa una idea a grandes rasgos de la estructura y el flujo de ejecución, en los siguientes apartados se desarrollan los detalles de cada una de las partes implicadas y como se relacionan entre si.

3.1 MÓDULO MULTIJUGADOR

Como se ha comentado en apartados anteriores, el desarrollo del módulo multijugador será desarrollado utilizando la plataforma nativa que proporciona Android a través de su SDK.

Android proporciona un framework que permite construir aplicaciones y juegos innovadores en un entorno que hace uso del lenguaje de programación Java. Aunque el SDK está disponible en versión aislada para poder ser utlizado en el IDE que el desarrollador estime oportuno, Android proporciona su propio paquete de descarga del framework, integrado ya dentro del IDE Eclipse[21]. Este paquete denominado ADT (Android Development Tools) incluyendo todo lo necesario para comenzar a crear aplicaciones para Android:

- Eclipse + plugin ADT
- Herramientas Android SDK
- Android Platform-tools
- Una versión de la plataforma Android
- Una imagen del sistema Android para el emulador

La arquitectura de Android se ejecuta sobre una máquina virtual similar a la de Java, denominada Dalvik. La máquina virtual Dalvik[13] se basa en la JVM pero está pensada para el desarrollo de aplicaciones nativas de Android, teniendo en cuenta la arquitectura de los dispositivos donde se va a ejecutar, como por ejemplo son los procesadores ARM, energía a través de batería,etc.

En la plataforma Android, el código fuente Java también se compila en ficheros .class. Pero una vez estos ficheros .class son generados, se convierte a ficheros ejecutables por la máquina virtual Dalvik .dex. Mientas que un fichero .class contiene únicamente una clase, un fichero .dex contiene múltiples clases. En cuánto a la plataforma del sistema operativo está basada en Linux y cuenta con su capa de abstracción para los drivers del dispositivo como GPS, cámara, acelerómetro.

En la siguiente imagen se describe un poco la arquitectura global del sistema:

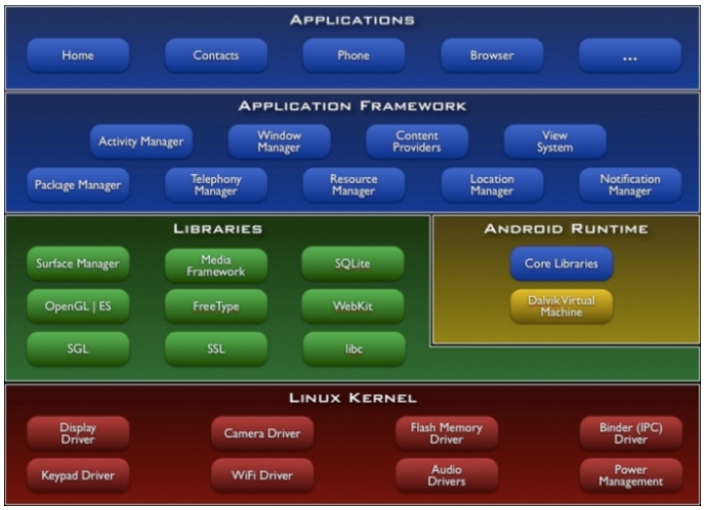


Figura 2 Arquitectura Android

La plataforma de Android también permite crear aplicaciones utilizando código en C/C++, haciendo uso de lo que ellos denominan NDK o Native Development Kit. Esto permite optimizar aplicaciones con una carga elevada de recursos como pueden ser videojuegos, aplicaciones de visión artificial, de manera que se le pueda sacar el máximo rendimiento al hardware sobre el que corre la aplicación.

Android permite realizar llamadas a código Java desde el código en C/C++ del NDK y viceversa, para ello se utiliza el denominado JNI (Java Native Interface). En el siguiente libro[8] se puede encontrar información detallada de cómo crear aplicaciones nativas para el NDK de Android, así como ejemplos de gran ayuda para comenzar a probar la plataforma.

Durante el desarrollo del módulo será necesario utilizar la JNI para poder acceder a los métodos escritos en Java desde el código C# que utiliza Unity3D.

3.1.1- DESARROLLO DEL PLUGIN BLUETOOTH PARA UNITY

Unity permite el desarrollo de plugins que pueden ser embebidos para poder ser utilizados dentro del propio motor. Esto permite añadir funcionalidades que pueden estar restringidas a ciertas plataformas, en el siguiente proyecto vamos a utilizar esta característica para añadir la conectividad a través de Bluetooth al videojuego.

Unity ofrece varias alternativas a la hora de crear plugins de Android, ambas maneras obligan a crear el plugin empaquetando el código o bien en Java creando una librería .jar o creando una librería .so con el NDK e incluyéndolo en la carpeta ubicada en la ruta **Assets->Plugins->Android,** dentro del proyecto en el que queramos utilizar el plugin. A partir de aquí podemos realizar llamadas al código java haciendo uso de la JNI

directamente desde código en C++ y cargando la librería en el código C#. La otra opción es utilizar la clase auxiliar que contiene Unity llamada **AndroidJNIHelper,** que permite realizar llamadas a métodos escritos en Java, así como mandar y recibir parámetros.

En mi caso me he decantado por la segunda opción, ya que permite un desarrollo algo más simple, evitando ciertos aspectos de bajo nivel que supone programar completamente en C++.

Una vez descargado el Eclipse[21] junto con las versiones del SDK de Android necesario podemos comenzar con el desarrollo de la aplicación. Para poder realizar el desarrollo e integrarlo con Unity es necesario heredar de la clase UnityPlayerActivity y declarar la nueva actividad dentro del fichero AndroidManifest.xml. El fichero AndroidManifest es el que se encarga de gestionar cuáles son los nombres de las Activities que vamos a declarar junto con los permisos que serán necesarios en la aplicación, como por ejemplo escritura, acceso a la cámara, GPS, etc.

A esto hay que añadirle el hecho de que nuestra aplicación requiere del plugin de Realidad Aumentada proporcionado por Vuforia, por lo que el proceso se vuelve algo más complejo.

En este caso el proceso consiste en importar al proyecto de Eclipse varias dependencias. En primer lugar es necesario añadir al proyecto los ficheros .jar correspondientes al plugin de Vuforia y el fichero classes.jar que es el correspondiente a las clases de Unity.

En la siguiente imagen se muestra cuál es la jerarquía del proyecto resultante.

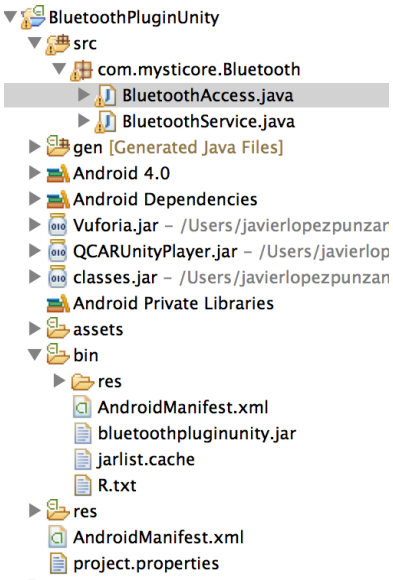


Figura 3 Estructura proyecto eclipse

Una vez preparado y configurado el proyecto, ya podremos comenzar con el desarrollo de la librería para Unity.

En primer lugar será necesario heredar de la clase QCARPlayerNativeActivity, esto permite que el código que vamos a desarrollar pueda ser ejecutado simultáneamente junto al código del propio plugin de Vuforia.

Otro punto necesario es añadir dentro del fichero AndroidManifest.xml el permiso para hacer uso del Bluetooth. A partir de aquí ya podemos empezar a plantear cual será el funcionamiento de la aplicación. Antes de que la aplicación intente comunicarse por Bluetooth, es necesario verificar que la conectividad Bluetooth está soportada por el dispositivo, y en caso afirmativo, asegurarse de que está activado.

En el caso de que el Bluetooth esté soportado en el dispositivo, pero este desactivado, se puede realizar una petición al usuario para que active el Bluetooth sin necesidad de salir de la aplicación. Este proceso es realizado en dos pasos, utilizando la clase **BluetoothAdapter.**

Una vez que tenemos el Bluetooth activado en el dispositivo, podemos comenzar a buscar la lista de dispositivos con el Bluetooth activado. La API de Android provee de un proceso asíncrono que retorna inmediatamente un booleano indicando si el proceso de descubrimiento ha comenzado sin problemas. El proceso de descubrimiento de dispositivos normalmente implica un escaneo de alrededor de unos 12 segundos, seguido de un escaneo de página por cada dispositivo para recuperar su nombre.

En el caso de que nuestro dispositivo sea el encargado de recibir las conexiones del server socket, es necesario poner el dispositivo en modo escucha. Por defecto el dispositivo en el que se active el modo de escucha permanecerá a la espera de conexiones durante 120 segundos por defecto, aunque este parámetro se puede configurar en caso necesario.

3.1.2 – CONEXIÓN ENTRE DISPOSITIVOS

Para poder crear una conexión entre la dos dispositivos, es necesario implementar los mecanismos tanto del lado cliente como del lado del servidor, ya que un dispositivo debe crear un socket servidor y el otro debe iniciar la conexión (utilizando las direcciones MAC del servidor. El servidor y el cliente se consideran como conectados el uno con el otro cuando cada uno dispone de un BluetoothSocket en el mismo canal RFCOMM. En este punto cada dispositivo puede obtener flujos de datos de entrada y salida y la transferencia puede comenzar.

El dispositivo cliente y el dispositivo servidor pueden obtener el BluetoothSocket requerido de diferentes maneras. El servidor lo recibe cuando una conexión entrante es aceptada. El cliente cuando abre un canal RFCOMM con el servidor.

Una técnica para implementar este comportamiento es prepara automáticamente cada dispositivo como un servidor, donde cada uno tiene un socket abierto y escuchando conexiones. De este modo, cualquiera de los dispositivos puede iniciar la conexión con el otro y convertirse en cliente. Alternativamente, un dispositivo puede albergar explícitamente la conexión y abrir un socket bajo demanda y el otro dispositivo puede simplemente iniciar la conexión.

En el caso de que los dispositivos no hayan sido emparejados ninguna vez anteriormente, el sistema mostrará automáticamente una notificación de petición al usuario durante el procedimiento de conexión.

3.1.3. – CONEXIÓN COMO SERVIDOR

Cuando se quiere conectar dos dispositivos, es necesario que uno de ellos actúe como servidor manteniendo un server socket abierto. El propósito del socket del servidor es escuchar peticiones de conexión entrantes, y cuando una es aceptada, lanzar otro socket de conexión. Una vez aceptada la conexión el primer socket debe ser descartado, a menos que se quieran aceptar más conexiones.

A continuación se detalla cual es el procedimiento para lanzar un server socket y aceptar una conexión:

- Obtener el BluetoothServerSocket llamando a la función listenUsingRfcommWithServiceRecord(String,UUID). No se ha mencionado anteriormente pero el UUID corresponde al identificador único del servicio al cual queremos conectarnos. Este identificador tiene que ser igual en todos los dispositivos que se quieran conectar a la aplicación.
- Comenzar a escuchar conexiones entrantes. Esta llamada es bloqueante y por lo tanto finalizará cuando una conexión sea aceptada o ocurra alguna excepción. Una conexión es aceptada únicamente cuando un dispositivo remoto haya mandado una petición de conexión con un UUID igual al que esta registrado por el socket que está a la escucha. Una vez aceptado correctamente, la función retorna un BluetoothSocket conectado.

- Al contrario que TCP/IP, RFCOMM solo permite tener un cliente conectado al mismo tiempo, así que en la mayoría de los casos tiene sentido cerrar el socket que estaba a la escucha una vez establecida la conexión.

Otra consideración importante, es que la llamada a la función encargada de aceptar las conexiones entrantes no debe ser ejecutada en el hilo de la activity principal porque es bloqueante. Así es que la implementación lógica es crear un thread para el socket de escucha y otro thread para el socket que se crea una vez conectada la aplicación.

3.1.4.- CONEXIÓN COMO CLIENTE

En el caso del cliente el primer paso para iniciar la conexión con el dispositivo servidor, es necesario en primer lugar obtener el objeto BluetoothDevice que representa el dispositivo remoto.

A continuación se describe el procedimiento básico:

- Utilizando el dispositivo bluetooth device obtener un socket llamando a la función createRfcommSocketToServiceRecord(UUID). Esto inicializa el socket que conectará con el otro dispositivo bluetooth. De igual modo que en el caso del servidor, es necesario que identificador UUID sea igual en los dos dispositivos que van a establecer la comunicación. Esto es algo que simplemente se codifica con una cadena de texto dentro de la aplicación, por lo que resulta bastante trivial.
- El siguiente paso es llamar al método de conexión *connect(),* de está manera se buscarán los dispositivos con el mismo UUID y se llevará caso la conexión utilizando el mismo canal RFCOMM.

Del mismo modo que en el caso del servidor, en el caso del cliente también es muy conveniente realizar todas las llamadas bloqueantes en hilos corriendo a parte del de la interfaz de usuario.

3.1.5. IMPLEMENTACIÓN FINAL

Por último falta crear todas las llamadas a los métodos que queramos utilizar desde Unity, por lo que he creado todas las funciones necesarias, algunos ejemplos son: llamadas para consultar el estado del bluetooth, activarlo, lanzar el servidor y ponerlo en escucha, lanzar el cliente, poner el dispositivo en modo de descubrimiento, etc.

También he creado una pila en la que se almacenan los mensajes recibidos por la aplicación y funciones para mandar y recibir mensajes de texto, de manera que desde Unity se podrá acceder a los mensajes enviados y recibidos por los dispositivos.

También he añadido llamadas a el Wrapper de Unity desde el código de la aplicación, ya que es la única manera de contestar cuando se termina algún proceso asíncrono. Unity permite mandar mensajes a los gameobjects con la función UnitySendMessage. De esta manera podemos establecer una comunicación bilateral, podemos consultar datos del lado de Java desde Unity y podemos avisar a Unity de la finalización de eventos como la finalización de la conexión etc.

3.2. VISUALIZACIÓN E INTERACCIÓN

En el siguiente apartado vamos a ir detallando cuales son los requerimientos y pasos necesarios para poder a comenzar el plugin realizado en la plataforma nativa de Android y como se van a unir todas las tecnologías, así como cuales son los apartados en los que se realiza el reconocimiento y puesta en marcha de la aplicación.

Como ya explicamos anteriormente, una vez hemos finalizado el plugin basta con compilarlo y generar el fichero .jar correspondiente. Este fichero lo ubicaremos dentro de la ruta de nuestro proyecto, en la carpeta **Assets->Plugins->Android,** de este modo ya podremos acceder a los métodos declarados en la librería creada para Android.

3.2.1 DESARROLLO DE LA INTERFAZ

En el siguiente vamos a abordar el desarrollo de la interfaz principal del juego y los menús**.** Aunque Unity permite crear elementos para la GUI, para el desarrollo de los menús he decidido un plugin específico denominado NGUI. Este plugin cuenta con una versión gratuita y de pago. La ventaja de este plugin frente a las herramientas de creación de GUIs que ofrece Unity es que es una herramienta mucho más completa, ofrece más funcionalidades de una manera mas sencilla y optimiza el rendimiento minimizando los DrawCalls de la aplicación.

El plugin de NGUI también está pensado para poder crear interfaces que sean compatibles con dispositivos móviles de diferentes resoluciones sin alterar el resultado obtenido en cuanto a proporciones, dimensiones, etc.

NGUI permite crear todo tipo de elementos tales como botones, paneles, etiquetas de texto, barras, sprites, etc.

La aplicación contiene simplemente dos menús, en el primero se nos permite escoger si queremos unirnos a un combate, o bien, crear un combate a modo de servidor. Dependiendo de que botones pulsemos se lanzarán los métodos del plugin de lanzar el servidor en el cliente respectivamente. Aunque lo que primero realiza la aplicación es asegurarse de que el dispositivo dispone de bluetooth y que lo tiene activado, en caso de tenerlo desactivado muestra un diálogo para que el usuario lo pueda activar sin necesidad de salir de la aplicación.

En el caso de que queramos crear una partida como servidor, al pulsar sobre la opción crear un combate se nos abrirá un diálogo que permanecerá a la espera hasta que el segundo jugador se conecte a la partida. Por el contrario, si lo que queremos es unirnos a una partida y conectarnos como cliente, al pulsar sobre el botón de unirse a un combate pasaremos al menú de selección de dispositivo. En este menú irán apareciendo todos los dispositivos bluetooth que están disponibles para poder conectarnos. Una vez seleccionado el dispositivo se mandará un mensaje al dispositivo, se establecerá la conexión y se comenzará a renderizar el escenario de combate.

3.2.2.- FASE DE RECONOCIMIENTO

Una vez desarrollado todo el protocolo de comunicación y la interfaz que permite a los usuarios seleccionar las opciones pertinentes, vamos a comenzar con la fase encargada de reconocer y renderizar en función de las marcas.

Una vez ya hemos importado al proyecto el plugin de Vuforia, el primer consiste en crear una marca o ImageTarget que nuestra aplicación sea capaz de reconocer y sobre la que añadir la información a nuestro escenario de Realidad Aumentada.

Para poder crear la marca, en primer lugar debemos elegir una imagen, a poder ser muy rica en textura para que el reconocimiento sea lo más rápido y robusto posible. Vuforia cuenta con una plataforma online denominada Vuforia Target Manager, en la que podemos registrarnos e ir subiendo nuestras imágenes y poder descargar los paquetes generados para realizar el reconocimiento dentro de Unity. Además la plataforma cuenta con un sistema para valorar la idoneidad de la imagen para ser utilizada en el reconocimiento, puntuándola en una escala de 1 a 5 estrellas, siendo las imágenes con 5 estrellas las más fiables para realizar el reconocimiento. En mi caso, he diseñado una imagen específicamente para el trabajo utilizando un programa de retoque de imágenes. Obtener una calificación no es sencillo, una de las recomendaciones que señalan en la página web de Vuforia es realizar una mejora del contraste local de la imagen. En mi caso apliqué esta técnica y conseguí mejorar la clasificación de cuatro a cinco estrellas.

Una vez que hemos descargado el paquete ya podemos comenzar a crear la escena de realidad aumentada. Los principales elementos que hay añadir a la escena son los prefabs proporcionados por el plugin de Unity. Estos prefabs son la ARCamera y el Image Target, la cámara lleva incorporados los scripts necesarios para realizar el reconocimiento, y en el caso del Image Target es donde añadiremos nuestra marca descargada desde la plataforma Target Manager.

Hay que decir que el sistema de reconocimiento de Vuforia es bastante robusto e incluye numerosos parámetros de configuración y funcionalidades. A pesar de ello hay situaciones en las que se puede observar cierto comportamiento nervioso en los modelos renderizados cuando se realiza el reconocimiento. Esto sucede sobre todo en los casos en los que el modelo a renderizar es bastante más grande que la marca sobre la que se realiza el reconocimiento.

Para solucionar este posible problema y hacer el diseño de la aplicación más robusto frente a los distintos tamaños de marcas que utilizar he decidido implementar un filtro de paso bajo para la cámara principal. Este filtro va guardando la posición y rotación de la cámara y realiza una media para aplicar un suavizado a estas transformaciones. El número de frames es un parámetro configurable aunque los mejores resultados se obtienen con valores entre 5 y 10 frames.

Por otro lado, Vuforia cuenta funciones que permiten realizar acciones personalizadas en los momentos en los que se comienza y se finaliza el tracking de las marcas. Haciendo uso de la función OnTrackingFound(), que se llama cuando nuestra marca es reconocida, he implementado una animación en la que se realiza una transición gradual de la opacidad de

los objetos que aparecen en la escena. Para poder implementar esta característica es necesario que todos los objetos de la escena dispongan de un shader que permita la utilización de transparencia. Una vez cambiado el tipo de shader de todos los objetos, lo siguiente es añadir una interpolación de la transparencia de los objetos, de manera que vaya aumentando progresivamente durante un tiempo hasta que todos los objetos se vuelvan opacos definitivamente.

3.2.3.-MOVIMIENTO Y COLISIONES

En el siguiente apartado se va a hablar en detalle de cuáles son las posibilidades que ofrece el motor utilizado y que características han sido las utilizadas para lograr el comportamiento deseado dentro de la aplicación.

En Unity3D existen dos elementos importantes con relación a la gestión de las colisiones y el uso de fuerzas. Estos elementos son por una parte los denominados colliders. Se podría decir que los colliders son propiedades que se pueden agregar a cualquier GameObject de la escena para que cuando intersecte con otro objeto que también cuente con un collider puedan utilizarse las llamadas definidas por el motor. Estas llamadas o eventos permiten detectar los eventos de entrada, salida y duración de la colisión. Esto ofrece una gran potencia a los desarrolladores ya que no es necesario codificar las colisiones realizando cálculos algebraicos demasiado complicados, además de que garantiza que los elementos funcionan de manera óptima en la mayoría de los casos. Unity cuenta con diferentes tipos de colliders, que pueden ser utilizados en función de los requerimientos de la aplicación, desde muy básicos para detección de colisiones con un gasto computacional bajo para efectos simples, hasta colliders que se ajustan perfectamente a las mallas de los objetos de la escena.

En lo que se refiere al uso de fuerzas dentro de la aplicación Unity proporciona el elemento denominado como RigidBody. Esto permite definir los cuerpos físicos de los objetos, añadiéndoles masa, material, fricción,etc. El sistema de físicas que utiliza Unity3D esta basado en la tecnologías Physx de Nvidia, por lo que los comportamientos obtenidos son bastante realistas. Combinando los mencionados colliders y los rigidbodies podemos realizar simulaciones con un alto grado de precisión y realismo, en la siguiente referencia[14] se muestra una tabla muy útil con cuáles son las eventos que podemos consultar dependiendo de que combinación de rigidbody y collider estemos utilizando.

En el caso del desarrollo de este trabajo he utilizado un collider específico denominado CharacterController, que como su nombre indica es el más apropiado para realizar interacciones entre personajes animados como en el caso del presente trabajo. Este collider no utiliza fuerzas por lo que permite evitar comportamientos inesperados cuando los personajes chocan entre sí o intentan salir del escenario.

En cuánto a la parte del movimiento, Unity cuenta con muy diversas maneras para poder aplicar movimiento a los objetos de una escena. Se pueden utilizar técnicas de interpolación a bien utilizar las funciones que Unity implementa como por ejemplo Translate o Move. También es posible mover bien el nodo de transformación del personaje o el nodo de transformación del collider asociado al modelo del personaje. En mi caso me he decantado por mover

directamente el nodo de transformación del collider, ya que si mueves simplemente el nodo transform del personaje pero no su collider, el comportamiento obtenido no es el deseado, ya que no se notifican los eventos de colisión.

3.2.4.- DESARROLLO DE LA LÓGICA PRINCIPAL DEL JUEGO

En el siguiente apartado vamos a tratar el desarrollo de todo la parte relacionada con la parte del desarrollo de la lógica y el transcurso del juego.

En primer lugar, como se ha comentado anteriormente, la aplicación se ha separado en *dos escenas completamente idénticas en el apartado visual, pero que funcionarán de* forma distinta en función de si estamos ejecutando la aplicación cliente o la aplicación del servidor.

A continuación se muestra una imagen de cómo a quedado estructurada la jerarquía de GameObjects de ambas escenas:

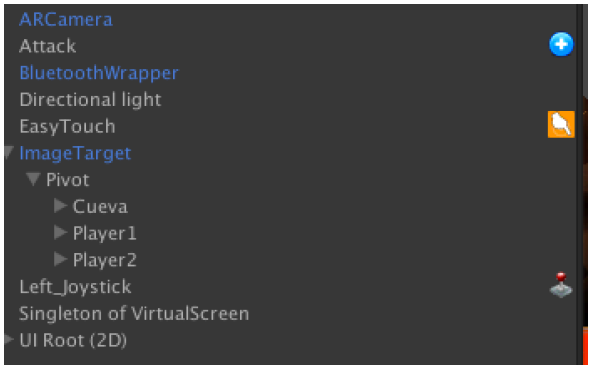


Figura 4 Jerarquía Proyecto Unity

Un punto bastante importante es añadir un GameObject que se llame BluetoothWrapper a la escena y añadir ahí los scripts asociados al módulo de conectividad. Esto es necesario porque como ya se ha explicado previamente, hay momentos en los que la librería del módulo manda notificaciones a Unity y es necesario pasar el nombre del GameObject destinatario como parámetro. Como se ha mencionado en el apartado anterior los GameObjects importantes en el apartado de Realidad Aumentada son el ARCamera e ImageTarget. Todo lo que queramos que se muestre cuando se realice el reconocimiento deberá ir situado como hijo del objeto ImageTarget dentro de la jerarquía.

En cuanto a los modelos utilizados para el desarrollo del juego he de resaltar que han sido descargados de la tienda de assets de Unity. Esto es así porque el desarrollo de personajes animados requiere de mucho tiempo y es algo más relacionado con el arte del juego, por lo que no cuenta con una relación directa con el objetivo perseguido con el proyecto.

Los elementos principales de la escena son los dos personajes controlables y el escenario, que se trata de una cueva con algún efecto de luz. La mecánica del juego es muy sencilla, consiste en moverse por dentro del escenario e ir a buscar al jugador oponente para golpearle hasta dejarle sin vida y proclamarte vencedor de la pelea.

Para poder controlar a los personajes se ha incorporado un joystick virtual en la interfaz de la pantalla, de manera que en función de si movemos el joystick más o menos, nos desplazaremos a una velocidad u otra. También he tenido que añadir un botón para poder realizar los ataques. En lo que respecta al control de los personajes, la metodología que he seguido ha sido dejar en las dos escenas dejar al jugador rival totalmente controlado por los mensajes que recibe desde la comunicación bluetooth y nuestro jugador es controlado por el joystick. De manera que mandamos nuestra posición, rotación y animación que estamos ejecutando a la otra aplicación, y esta recoge los datos y los aplica sobre el jugador para que se muestre que está haciendo durante el transcurso de la partida.

Este es un enfoque bastante clásico en los juegos multijugador, de hecho a mí es el que mejor me ha funcionado finalmente, después de haber probado otras opciones. La funcionalidad básica consiste por una parte en detectar los eventos de movimiento tanto del joystick, como del botón de ataque y envíar los mensajes para que el otro jugador reciba esta información y aplique las transformaciones correspondientes al otro personaje. Al mismo tiempo que se mandan los mensajes hay que estar escuchando cuando llegan los mensajes del otro jugador para hacer lo mismo en nuestra escena.

Otro importante dentro del desarrollo de la lógica de la aplicación, ha sido como tratar y detectar las colisiones de los personajes. Tanto cuando colisionan con el entorno como cuando atacan o son atacados por el otro jugador. Para tratar la colisión de los personajes he añadido un character controller a los personajes controlados por el jugador, de manera que la colisión se detecta automáticamente cuando colisionan con las paredes del escenario o con el adversario. En el caso de detectar las colisiones cuando un jugador ataca al otro el proceso es más complejo. Para ello hay que en primer lugar comprobar que el jugador esta ejecutando la animación de atacar. Después hay que calcular que el jugador que ataca tiene en su ángulo de alcance al jugador rival y por último que esta a una distancia determinada del jugador. Si se cumplen estas tres condiciones, podemos afirmar que el otro jugador ha sido atacado y generar las acciones correspondientes, como por ejemplo aplicar la animación de daño en el otro jugador, etc.

El apartado de colisiones y sincronización ha sido el más costoso de desarrollar, ya que hay situaciones en las que el jugador puede no recibir un mensaje y es necesario corregir las transformaciones que se van aplicando constantemente a los personajes.

Por último se han añadido las barras de vida, para que se muestre en todo momento cuales son los niveles de vida de cada jugador y cuando agoten su vida caigan y reproduzcan la animación correspondiente.

4. RESULTADOS

En la siguiente sección se van a mostrar y a evaluar los resultados obtenidos en el desarrollo de este trabajo. Dado que el tipo de trabajo tiene una clara orientación profesional, se va a intentar realizar un análisis del producto y de las métricas que se suelen tener en cuenta dentro del mercado profesional.

Por lo que respecta a la interfaz a continuación se muestran algunas imágenes en las que se puede apreciar el diseño de los componentes con los que el usuario puede interactuar.

Figura 5 Menu Principal de la aplicación

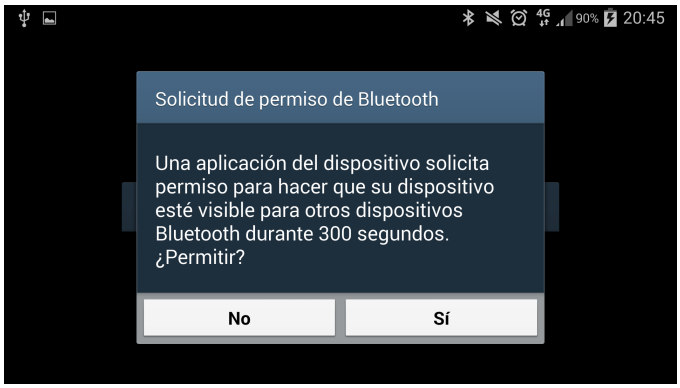


Figura 6 Diálogo conexión Bluetooth

Las anteriores ilustraciones muestran las partes de la interfaz que resultan comunes tanto a la ejecución de la aplicación cliente como a la del lado del servidor.

A continuación se muestran las figuras de las partes restantes y el resultado de la escena final.

Figura 7 Menú selección de rival

Figura 8 Escena principal de combate

Como se puede apreciar en la última figura, en la captura aparecen los fotogramas por segundo a los que se esta ejecutando la aplicación. Para realizar el cálculo se ha añadido un script que realizar el conteo de los fotogramas y los muestra por pantalla.

Para comprobar el comportamiento de la aplicación dependiendo de la plataforma en que se ejecuta he realizado mediciones de la media de fotogramas por segundo en dos dispositivos con hardware diferente.

En primer lugar he realizado mediciones de los fotogramas por segundo en un dispositivo con las siguientes características:

Modelo y fabricante:	Samsung Galaxy Nexus
Chipset:	TI OMAP 4460
CPU:	PowerVR SGX540
GPU:	Dual-core 1.2 GHz Cortex-A9
RAM:	1 GB RAM
Bluetooth:	v3.0, A2DP

En este dispositivo la tasa de frames por segundo obtenido ha sido de aproximadamente una media de **20fps**. Estamos hablando de un dispositivo con alrededor de dos años y medio de antigüedad y que cuenta con una potencia de cálculo algo desactualizada con respecto a la mayoría de los modelos que se encuentran en el mercado actualmente.

El otro dispositivo en el que ha sido testeada la aplicación es un dispositivo que cuenta con las siguientes características:

Modelo y fabricante:	Samsung Galaxy S4+
Chipset:	Qualcomm MSM8974 Snapdragon 800
CPU:	Quad-core 2.3 GHz Krait 400
GPU:	Adreno 330
RAM:	2 GB RAM
Bluetooth:	v4.0, A2DP, EDR, LE

Este dispositivo si que cuenta con un hardware más actual, un procesador de cuatro núcleos y el doble de memoria RAM, la tasa de frames por segundo obtenida con este dispositivo ha sido aproximadamente de unos **55 a 60fps.** También es apreciable la calidad de la cámara con respecto a la del otro dispositivo, ya que también ayuda a reconocer la marca en menor tiempo.

Por último mencionar que la aplicación como se puede apreciar necesita de una potencia de cálculo bastante elevada para poder disfrutar de una experiencia suficientemente satisfactoria para el usuario, aunque también es cierto que se podría aplicar optimizaciones y realizar una fase de testing en profundidad para mejorar este aspecto.

5. CONCLUSIONES

5.1 PROBLEMA PLANTEADO

Hoy en día existen multitud de ATGR que hacen uso de características y tecnologías muy novedosas, como por ejemplo el reconocimiento de patrones, la Realidad Aumentada, Realidad Virtual entre otras.

En el caso concreto de los videojuegos, existen todavía pocos desarrollos comerciales en los que se integren estas técnicas de manera satisfactoria. Sobre todo en lo que respecta a desarrollos de videojuegos para dispositivos móviles, las capacidades técnicas que ofrecen estos dispositivos son realmente sorprendentes, y si se cuenta con una idea general de estas capacidades, es fácil pensar en multitud de ideas para integrarlas dentro de un videojuego.

Además en el caso concreto de la Realidad Aumentada, no es habitual encontrar entornos en los que el jugador puede compartir la información que se esta renderizando con otros jugadores a través de cualquier tipo de conectividad y en tiempo real.

El problema es que en algunas ocasiones y casos concretos integrar estas tecnologías bajo el desarrollo de un videojuego no es algo tan sencillo como cabría esperar. De aquí surge la idea y la solución propuesta en el desarrollo de este trabajo final de Máster.

5.2 SOLUCIONES PROPUESTAS

Como solución al problema planteado se propone realizar un videojuego, que se ejecutará sobre un dispositivo móvil de la plataforma Android, integrando la tecnología de Realidad Aumentada y añadiendo la posibilidad de interactuar con otro jugador en el mismo entorno generado mediante marcadores. Todo el desarrollo esta enfocado desde un punto de vista muy cercano a lo que podría ser un desarrollo de un producto que pudiera ser puesto a la venta en un futuro, utilizando las herramientas que utilizan la mayoría de profesionales independientes dentro del sector de los videojuegos.

5.3 TECNOLOGÍA UTILIZADA

En el proyecto desarrollado se han multitud de tecnologías diferentes, de hecho una de las características principales del desarrollo se centra en la integración de estas diferentes tecnologías. Se podría decir que el motor de videojuegos utilizado, Unity3D es el pilar sobre el que se incorporan el resto de las tecnologías utilizadas. Unity3D cuenta con la ventaja de que es un motor de videojuegos que cuenta con unas extensibilidad realmente grande, y es por ello que ha sido la opción por la que he apostado para el desarrollo de este proyecto.

En primer lugar se ha hecho uso de la Realidad Aumentada, técnica que permite ampliar la información de nuestro entorno con información añadida de manera artificial. Para el desarrollo del reconocimiento y el tracking de las marcas se ha hecho uso del framework

desarrollado por la empresa Qualcomm, denominado Vuforia, que cuenta con soporte para poder ser utilizado en proyectos del engine de juegos Unity3D. Por otro lado se ha realizado un plugin ad-hoc para poder incorporar la posibilidad de interconectar los dispositivos en los que se ejecutará el juego a través de Bluetooth. Por último mencionar que para el desarrollo del código se han utilizado los IDEs MonoDevelop y Eclipse.

5.4 Relación de conocimientos aplicados

En el siguiente apartado se van a analizar cuáles son los conocimientos que están relacionados de algún modo con materias y conocimientos adquiridos durante el desarrollo del Máster.

En primer lugar, el componente principal sobre el que desarrolla el trabajo está basado en la utilización de Realidad Aumentada como entorno en el que se ubica el escenario donde se ejecuta la aplicación, lo que tiene una relación totalmente directa con el bloque de especialización de Imagen Digital del Máster. Tanto es así que existe una asignatura específicamente dedicada al desarrollo y puesta en práctica de conocimientos relacionados con la realidad aumentada. Por otra parte, el resto de la aplicación se centra en el desarrollo de un videojuego, algo que también se ha trabajado con detalle durante el transcurso del plan de estudios. En concreto en las asignaturas de Animación por Computador y Videojuegos, Programación Gráfica y Avances en Informática gráfica. Asimismo también existe una clara relación entre la parte de estudio y análisis de técnicas de reconocimiento y tracking sin utilización de marcas con las técnicas de Visión por Computador vistas en algunas asignaturas de Reconocimiento de Formas y la asignatura de Visión 3D y movimiento.

Por tanto cabe decir que gracias a los conocimientos adquiridos durante el plan de estudios del Máster ha sido posible afrontar el desarrollo de este trabajo con una base sólida sobre la que poder aplicar herramientas y ampliar ciertos aspectos, todo ello enfocado a los campos que en mi caso me han parecido más estimulantes. De hecho, aunque obviamente no todo lo ha aprendido en el Máster ha sido aplicado en el desarrollo del trabajo, es complicado pensar en algún campo de los estudiados que no pudiera ser incorporado para alguna de las funcionalidades del trabajo en futuras ampliaciones.

Por todo ello me siento satisfecho de haber podido formar parte como alumno y haber podido acceder a los conocimientos impartidos en el Máster. Bajo mi punto de vista que me ha sido de mucha utilidad no sólo para el desarrollo de este trabajo, sino como una gran mejora de mi currículo y por lo tanto en mi futuro profesional.

6. TRABAJO FUTURO

El trabajo realizado permite pensar en multitud de ampliaciones que podría resultar interesantes, así como la aplicación de algunas técnicas utilizadas en proyectos de diferente ámbito. A continuación se describen algunas de las ampliaciones que pienso podrían resultar más interesantes:

- En primer lugar hubiera sido interesante realizar una fase de testing sobre diferente hardware, ya que solo se han medido resultados en un número de dispositivos limitado. Esto permite realizar una validación mayor y la posible optimización de partes problemáticas desde el punto de vista de la gestión de recursos.
- Otro apartado a tratar sería el realizar un estudio sobre la usabilidad final de la aplicación, ya que sería interesante medir de que manera interactúan los usuarios con la aplicación y el grado de satisfacción obtenido por los mismos.
- Por lo que respecta a las ampliaciones relacionadas con la mecánica de juego existen multitud de características que enriquecerían la experiencia y la diversión durante el desarrollo de la partida. Algunas de ellas podrían ser añadir la posibilidad de que aparecieran ítems equipables por los jugadores que pudieran potenciar el daño a la defensa de los personajes. También podría ser muy interesante añadir nuevos botones para permitir el uso de otras animaciones, realizar ataques especiales o defenderse de los ataques realizados por el oponente.
- Por otro lado otra mejora interesante sería poder cambiar el escenario en función del reconocimiento, o bien reconociendo características del entorno o bien utilizando diferentes marcas, lo que además permitiría añadir características que influyeran en el transcurso de la partida.
- También resultaría interesante haber implementado una capa de persistencia de datos, para que los jugadores pudieran guardar el estado de su personaje y así poder ir mejorando sus características y habilidades a lo largo del tiempo. Para ello hubiera sido necesario generar algún tipo de base de datos que se alojara en el propio dispositivo.
- Dado que el origen de este trabajo era la propuesta del desarrollo de una tesina conjunta, junto con dos compañeros más del Máster, hubiera sido interesante combinar los desarrollos dentro de alguna de las partes del juego. En concreto podría ser interesante utilizar la IA desarrollada por uno de ellos para poder añadir personajes no controlables o utilizar el tracking de la cabeza para mover la cámara e incluso incorporar este movimiento a algunas mecánicas del juego.
- Otra mejora muy interesante sería poder eliminar la marca en la parte del reconocimiento, implementando un plugin o algoritmo que pudiera ser capaz de reconocer características del entorno.

Estas son algunas de las posibilidades planteadas como posibles mejoras y ampliaciones, aunque en mi opinión el enfoque del proyecto y la base desarrollado puede aplicarse a ámbitos muy diferentes, como por ejemplo educación, muestras de arte, etc.

7. REFERENCIAS

[1] G. Klein and D. Murray. Parallel Tracking and Mapping for Small AR Workspaces. *Proc. IEEE/ACM ISMAR'07*, November 2007.

[2] Brian Williams, Georg Klein and Ian Reid, Department of Engineering Science, University of Oxford, UK. Real-Time SLAM Relocalisation, ICCV 2007.

[3] G. Klein and D. Murray. Parallel Tracking and Mapping on a camera phone. In IEEE *ISMAR*, October 2009.

[4] P. Martin, E.Marchand, P.Houlier and I. Marchal. Decoupled Mapping and Localization for Augmented Reality on a Mobile Phone . Institut de Recherche en Informatique et Systèmes Aléatoires, ICIP 2014.

[5] Andrew Howard, NASA Jet Propulsion Laboratory. Multi-robot Simultaneous Localization and Mapping using Particle Filters, RSS, 2010.

[6] *Wendy E. Mackay* , Université de Paris-Sud Augmented Reality: Linking real and virtual worlds A new paradigm for interacting with computers. Advanced Visual Interfaces 1998.

[7] Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications. IEEE Std 802.11TM-2007

[8] Feipeng Liu, Android Native Development Kit Cookbook, Packt-Publishing. ISBN 978-1-84969-150-5

[9] Bluetooth Specifications, IEEE Standard for 802.15.1- 2002.

[10] Chatschik Bisdikian, "An Overview of the Bluetooth Wireless Technology", IBM Research Report, June 2001.

[11] Onur Cinar, Android Apps with Eclipse, Apress. ISBN 978-1-4302-4435-6

[12] Essential facts about the computer and video game industry. Entertainment Software Association.

[13] David Ehringer, The Dalvik Virtual machine architecture, 2010.

[14] Collision detection Unity. http://docs.unity3d.com/Manual/CollidersOverview.html

[15] Official AllJoyn website: https://www.alljoyn.org/about

[16] Android developer page: http://developer.android.com

[17] Corona SDK website. http://coronalabs.com/products/corona-sdk/

[18] Starling website. http://gamua.com/starling/features/

[19] Wireless Ethernet Compatibility Alliance (WECA). http://www.wi-fi.org/who-we-are

[20] Azuma, R. T., A Survey of Augmented Reality. Presence: Teleoperators and Virtual Environments, Vol. 6, N. 4, pp. 355 – 385, 1997

[21]Eclipse official website https://www.eclipse.org/

[22] Unity3D offical website http://unity3d.com

[23] Milgram, P., Kishino, F., A taxonomy of mixed reality visual displays, IEICE (Institute of Electronics, Information and Communication Engineers) Transactions on Information and Systems, Special issue on Networked Reality, Dec., 1994

8. ANEXO I – GUÍA DE USUARIO DEL API

Dado que el desarrollo del plugin del Bluetooth para Unity quizás pueda resultar de interés para futuros desarrollos, en el siguiente capítulo se va a explicar cuáles son los pasos necesarios para poder integrar y hacer uso de la librería.

En primer lugar hay que coger la librería del paquete .jar y añadirla a la carpeta plugins->Android de Unity. En el caso de querer cambiar el código tendríamos que preparar un proyecto Android en Eclipse y coger el fichero .jar después de modificar el código fuente generado. Es importante que para poder utilizar el plugin se cree un GameObject en la escena que queramos utilizarlo con el nombre BluetoothWrapper, y que todos los scripts que hagan llamadas a la librería del Bluetooth estén asociados a ese GameObject. Una vez hecho esto y haciendo uso de la clase Android JNI Helper ya podemos hacer llamadas a las funciones de la librería en java:

Las principales llamadas en la parte Java son:

```
//Comprobar si el bluetooth está activado en el dispositivo
public boolean isBluetoothEnabled() {}

//Activar el bluetooth, devuelve true si todo ha ido bien y false en caso de error

public boolean enableBluetooth() {}

//Pone el servidor en modo descubrimiento y lanza el thread para que se inicie el socket
//de escucha del socket

public void launchServer(){}

//Activa la fase de búsqueda y descubrimiento de dispositivos en el cliente.

private void launchClient(){}

//Conectar el dispositivo con la dirección mac pasada por parámetro en modo
seguro/inseguro

public void connectDevice(String address,boolean secure) {

//Manda un mensaje al dispositivo conectado

public void sendMessage(String message) {}

//Lee el último mensaje almacenado en la pila de mensajes

public String getMessage(){}

//Muestra un mensaje informativo sobre la interfaz principal

public void showDialog(){}

//Activa el modo de descubrimiento en el dispositivo seleccionado

public void enableDiscover(){}

//Muestra un spinner dialog mientras se espera la conexión del jugador rival

public void showDialog(){}
```

El resto de funcionalidad es interna a la clase y es la que se encarga de gestionar los threads en una clase separada, de todos modos cualquier método público puede ser accedido sin ningún problema haciendo uso de JNI.

EJEMPLO DE USO DE LA LIBRERÍA CON UNITY3D

En el siguiente apartado se va a detallar un caso de uso haciendo uso de la librería implementada en Android y de Unity3D como enlace. Mencionar que antes de nada habría que seguir los pasos detallados previamente para incorporar el plugin dentro de nuestro proyecto en Unity. Es decir, crear un nuevo proyecto y asociar un GameObject vacío a la escena con el nombre de BluetoothWrapper.

Por otra parte sería muy conveniente separar la parte del cliente y el servidor en diferentes escenas, tal y como yo he hecho para el desarrollo de este trabajo. Esto permite encapsular y aislar de manera sencilla tanto la parte gráfica como la lógica de las partes cliente y servidor respectivamente.

En el siguiente ejemplo vamos a ver como mandar un mensaje de Hola mundo entre una aplicación cliente y una aplicación servidor en Unity. Para el desarrollo del ejemplo vamos a hacer uso de una clase singleton que se encarga de realizar la conexión con el código Java de la librería. Este código se encuentra incluido dentro del código del proyecto, por lo que es una buena base para poder comenzar ya que incorpora las partes que son comunes tanto para la parte cliente como para la parte del servidor.

Esta clase se encarga básicamente de llamar a la clase Java contenida en el paquete de la librería, además proporciona la llamada a la función que permite activar el bluetooth, ya que esta parte es necesaria en los apartados del cliente y el servidor.

CLASE COMÚN

```
public class BluetoothAndroid {
   private AndroidJavaClass jc ;
   private AndroidJavaObject jo;
   public bool connected;
   private bool benabled, bconnected;

   private static BluetoothAndroid instance = null;

    public static BluetoothAndroid GetInstance(){
      if(instance == null){
         instance = new BluetoothAndroid();
      }
      return instance;
   }
   public AndroidJavaClass JC
   {
      get { return jc; }
      private set { jc = value; }
   }
   public AndroidJavaObject JO
   {
      get { return jo; }
      private set { jo = value; }
   }
   public bool Benabled
   {
      get { return benabled; }
      private set { benabled = value; }
   }
   public bool BConnected
   {
      get { return bconnected; }
      set { bconnected = value; }
   }
   public void enableBluetooth() {
      benabled = jo.Call<bool>("isBluetoothEnabled");
      if (benabled) {
      }
      else {
         bool activebluetooth = jo.Call<bool>("enableBluetooth");
         benabled = jo.Call<bool>("isBluetoothEnabled");
      }
   }
   public void Init () {
      connected = false;
      AndroidJNI.AttachCurrentThread();
      jc = new AndroidJavaClass("com.mysticore.Bluetooth.BluetoothService");
        jo = jc.GetStatic<AndroidJavaObject>("currentActivity");
      if (jc != null) {
         Debug.LogError("The class is loaded");
      }
   }
```

El siguiente código es necesario ubicarlo dentro de un script que vaya asociado al GameObject BluetoothWrapper creado anteriormente para la escena cliente.

APLICACIÓN CLIENTE

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

public class BluetoothClient : MonoBehaviour {
   BluetoothAndroid bAnd;
   private bool connected;

   void Start(){
      connected = false;
      bAnd = BluetoothAndroid.GetInstance();
   }
   //Este método es llamado cuando se cambia de estado en la librería, a la inversa del procedimiento normal
            //desde Java a Unity para notificar que los dispositivos se encuentran conectados
   void setConnected(String con) {
      if (con == "true"){
         connected = true;
         bAnd.BConnected = true;
      }
      else {
         connected = false;
         bAnd.BConnected = false;
      }
   }
   void FixedUpdate(){
      if (connected){
         string message = bAnd.getMessage();

         Debug.LogError("Server: " + message);

   }
```

APLICACIÓN SERVIDOR

En este caso deberíamos utilizar el mismo código que en el cliente pero simplemente reemplazar la función FixedUpdate por la siguiente:

```
  void FixedUpdate(){
     if (connected){
        bAnd.sendMessage("Hola Mundo");

}
```

De este modo la aplicación del servidor se limita a ir mandando mensajes de "Hola Mundo" mientras que la clase cliente se limita a imprimir los mensajes por la consola dentro del bucle principal.

9.ANEXO II - DOCUMENTO DE DISEÑO DEL VIDEOJUEGO FINAL

9.1 - DISEÑO DE LA HISTORIA

Esto es un documento de diseño relativo a una tesis conjunta llevada a cabo por tres personas. La parte común a los tres es la creación de un videojuego para dispositivos móviles con sistema operativo Android; por otra parte, cada uno de los integrantes del proyecto desarrollará un módulo que se aplicará en la versión final del juego.

9.2 VISIÓN GENERAL DEL JUEGO

El fin del proyecto, es ofrecer un producto que consideramos novedoso en el mercado de las aplicaciones, tanto por la tecnología que utiliza como por el juego en sí. El proyecto consta de cuatro partes: tres partes son módulos que introducirán algunas novedades en el desarrollo y funcionalidad del producto, la última parte será el propio videojuego, trabajo común a todos los desarrolladores. Nuestro juego funcionará en dispositivos Android. Esto se debe a la accesibilidad al mercado en dispositivos con este sistema, y a la tecnología a emplear.

Consideramos más prudente iniciarnos en el mundo del desarrollo de videojuegos con estas especificaciones. El objetivo final será ofrecer un producto de calidad e ir incrementando nuestro conocimiento en el área del desarrollo de videojuegos.

9.3 PREGUNTAS FRECUENTES

9.3.1¿DE QUE TRATA EL JUEGO?

El juego consiste en el entrenamiento de una criatura o varias (en un futuro personalizable/s) y en el combate con otras criaturas de otros usuarios mediante una experiencia basada en la realidad aumentada y la visión 3D.

9.3.2¿CUÁL ES EL MOTIVO PARA CREAR ESTE JUEGO?

Consideramos interesante su creación especialmente por dos motivos. Primero, este tipo de juego todavía no se ha llevado a cabo en los dispositivos móviles. Si bien es cierto que existen juegos que guardan algunas similitudes conceptuales, no hay ningún juego importante en el género. Buscamos que el usuario se divierta con un producto sencillo. Segundo, pretendemos llevar a cabo la acción mediante una experiencia basada en la Realidad Aumentada, pero sin el uso de marcadores adicionales. Si bien la RA todavía no está demasiado extendida en los videojuegos, todavía lo está menos en aquéllos para dispositivos móviles, además de llevar a cabo la RA sin marcadores.

9.3.3 ¿DÓNDE TIENE LUGAR EL DESARROLLO DEL JUEGO?

Nuestro juego hace uso de la Realidad Aumentada, así pues, tiene lugar allá donde queramos lo tenga. Pretendemos que el usuario pueda jugar en cualquier parte con otros usuarios y competir.

9.3.4 ¿CUÁNTOS PERSONAJES PUEDO CONTROLAR?

En un primer momento, el usuario creará su mascota, a la cual entrenará y hará competir. Se está estudiando ampliar esto a simulaciones de lucha más grandes, con más personajes controlables a nivel grupal.

9.3.5 ¿CUÁL ES EL OBJETIVO PRINCIPAL?

Más allá del aprendizaje, únicamente crear un juego sencillo y adictivo.

9.3.6 ¿QUÉ TIENE DE DIFERENTE?

La experiencia como tal es diferente, por lo comentado anteriormente, y porque su pone una novedad a nivel jugable.

9.4 CONJUNTO DE CARACTERÍSTICAS

9.4.1 CARACTERÍSTICAS GENERALES

El juego no desarrolla una historia. Se basa en el entrenamiento de una criatura (configurable) mediante una serie de retos y desafíos, para hacerla aumentar sus habilidades y stats con el fin de competir contra otras de otros usuarios.

Estas criaturas poseen ciertos poderes físicos y mágicos, acorde a su tipo (Anfibio, Arácnido, Ave, Insecto, Mamífero, Pez, Reptil) y elemento natural (a priori: Fire, Water, Earth, Wind), además del nivel de entrenamiento (que determinará el nivel de poder).

El usuario podrá entrenar su criatura en una serie de niveles de entrenamiento (a priori virtuales), así como educarla para que le haga caso y darle cuidados (lavar, dar de comer, equiparla). Luego podrá hacerla competir con otra criatura en el escenario en el que los usuarios se encuentren, haciendo uso de la ***detección de área***. El fin del juego es el combate, así pues, la competición es el principal atractivo. La jugabilidad se basa en las habilidades físicas y mágicas de una criatura y la acción es similar a la de los videojuegos de lucha convencionales. Una criatura podrá golpear a su adversario con su físico o con sus poderes mágicos.

9.4.2 CARACTERÍSTICAS MULTIJUGADOR

En lo referente al multijugador, es el aspecto más importante y que por tanto debe cuidar más detalles. Es el principal atractivo del juego, hacer competir a nuestra criatura con otras de otros jugadores de manera offline (la gran baza del juego es estar basado en la Realidad Aumentada y por tanto no se contempla una partida online, pues no tiene sentido no estar en el mismo escenario de juego). Está pensado para combates 1 vs 1 en tiempo real, aunque es probable que el juego incluya alguna otra modalidad para añadir más jugadores.

La conexión se establece mediante el protocolo *Bluetooth*. Un jugador detecta a otro cercano y le manda una **petición de combate**. Cuando se crea una partida para competir, se establecen una serie de parámetros necesarios en lo referente al escenario (detección de área) y a la partida en sí (tiempo de partida, ¿número de jugadores?, calibración, etc...).

9.4.3 CARACTERÍSTICAS EDITABLES

Escenarios

El juego es poco editable a nivel de escenarios de combate; son los jugadores los que deciden dónde jugar.

Creación de la criatura

Un aspecto configurable es la apariencia de nuestra criatura. Al inicio de la experiencia, es necesario crear una criatura desde cero. Eso implica dar al jugador una serie de opciones entre las que elegir, relativas a la apariencia física del monstruo, y mediante la combinación de éstas, formarlo en su totalidad. Las diferentes elecciones del usuario determinarán **la raza**. Para la demo se utilizarán dos razas predefinidas como modelos disponibles.

Por otro lado, tendremos una serie de **puntos de habilidad extra** que tendremos que repartir entre una serie de **características innatas** (*Fuerza física, Fuerza especial, Resistencia física, Resistencia especial y Velocidad*) para determinar **la clase** de la criatura.

Para la creación de nuestro monstruo tenemos un total de 15 puntos de habilidad que definen las características de la criatura (5 de ellos son elegibles por el jugador). Al confirmar la asignación de estos puntos, se procederá a asignar el cristal de clase inicial en el tablero de clases. En este tablero solo serán seleccionables una de las cinco clases base: Beast, Wizard, Tank, Guardian y Flash. Cada una corresponde a una habilidad que destaca en la asignación de puntos de habilidad: Fuerza, Fuerza especial, Resistencia física, Resistencia especial y velocidad (respectivamente). En el caso de que el jugador tenga varias habilidades a igual nivel, podrá elegir una entre las clases correspondientes a cada habilidad.

Equipo de la criatura

La criatura puede editarse mediante el equipo de una serie de armamento o protección para el combate, llamado **equipo de combate**. Mediante dinero que el jugador conseguirá en los niveles de entrenamiento o vendiendo objetos, podrá comprar diferentes vestimentas de combate para su criatura. Además, estos objetos aparecerán en los entrenamientos o combates con una frecuencia baja como reliquias de batalla.

Por otro lado, existen los **potenciadores de combate**, que son habilidades relativas a la clase de la criatura y que podrán ser equipadas y añadirán una protección/potencia adicional (invisibilidad, fuerza bruta, velocidad extrema, magia doble, etc...), todos ellos adquiribles por clase y/o dinero (según la clase se desbloquearán para ser comprados).

Existen dos tipos de potenciadores: aquellos que se basan en la fuerza física y otros que utilizan la fuerza especial. Los potenciadores conseguidos se pueden asignar libremente en los dos slots disponibles en el menú de equipo de la criatura. Los potenciadores se diferencian en su potencia, duración y tiempo de recuperación.

Por otro lado, los **objetos de combate** aparecerán a lo largo del enfrentamiento al azar, y podrán ser recogidos y almacenados en un slot especifico de objeto. Estos objetos se podrán activar a lo largo del combate y mejoraran algunas características de la criatura (en el caso de medicinas, pociones…), activarán ataques especiales (espadas, bastones...) o cambiaran el estado de la criatura (invisibilidad, invencibilidad…) o cualquier combinación de los anteriores. Cada objeto tiene su tiempo de objeto, que delimita el tiempo que el objeto está operativo una vez el jugador lo haya activado (pulsando sobre el icono de objeto recogido). Una vez consumido ese tiempo, el objeto desaparecerá, devolviendo a la criatura a su estado anterior a la activación de éste.

9.5 GAMEPLAY

El flujo normal de juego que sigue un jugador es el siguiente:

Creación de la criatura
Resumiendo lo especificado en la sección de *Editable Features*, el jugador crea la mascota definiendo ***la raza*** *(en la DEMO hay dos razas predefinidas)*, y las ***características innatas*** en base al reparto de una serie de ***puntos de habilidad extra***.

Definición de su clase
Según el reparto de los puntos de habilidad iniciales, se define la **clase base** de la criatura. A medida que el jugador vaya consiguiendo cristales de clase, podrá ir asignando nuevas y desbloqueará de esta manera más potenciadores.

Cuidado, Entrenamiento y/o combate (gameplay más extenso)
La parte más importante de nuestra experiencia jugable es la relativa a los cuidados, entrenamiento y a la competición con otras criatura, de manera que a este nivel, el juego tiene que ser lo más atractivo y robusto posible.

Como realizar el juego al completo es una tarea larga, partimos desde cero y el tiempo máximo de desarrollo no es ilimitado, se pretende crear una DEMO que cumpla en la experiencia jugable en todos los apartados anteriores **(Al menos 2 o 3 razas, objetos limitados, cuidados mínimos, un nivel de entrenamiento y el combate)**.

A continuación se explican al detalle cada uno de los tres puntos anteriores:

1) La creación de la criatura

A priori, el jugador podrá modelar su criatura en base a una serie de elecciones. Para el juego desarrollado para esta tesina, no se dispone de las herramientas completas de edición y creación de la criatura, así pues, para la versión DEMO únicamente se podrá elegir entre dos criaturas de razas diferentes y con sus 10 puntos de habilidades base ya definidos, pudiendo cambiar su clase inicial mediante la asignación de los 5 puntos de habilidad adicionales.

1.1 Historia

Desde el origen de la vida en la Tierra, cada ser humano tiene asociado un Mystical, un alter ego que habita cerca del núcleo terrestre y que se encarga, junto a sus semejantes, de mantenerlo estable. Una tormenta solar ha producido que estos Mysticals hayan perdido su misticismo, y esto ha generado un desequilibrio natural en el núcleo. Este suceso ha provocado que las criaturas salgan a la superficie para advertir a los humanos de que este cambio puede ser catastrófico para la vida en el planeta. Además, dicha tormenta, ha alterado los dispositivos electrónicos de la superficie, y los humanos son capaces de ver a través de algunos de ellos a sus alter ego. Los Mysticals deben recuperar su poder ancestral con ayuda de los humanos entrenando y combatiendo entre ellos para devolver el equilibrio perdido.

1) Definición de su clase

En esta sección veremos todo lo relativo a la clase de un Mystical.

2.1 Reparto de puntos de habilidad adicionales

En total hay 5 características por cada criatura: *Fuerza física, Fuerza especial, Resistencia física, Resistencia especial y Velocidad*. La fuerza determina el daño directo infligido mediante el botón de ataque físico. La fuerza especial determina el daño que causan los potenciadores. Resistencia es el valor de defensa contra ataques físicos y resistencia especial contra ataques realizados mediante potenciadores. La velocidad determina la agilidad de movimiento de nuestra criatura sobre el mapeado.

Al inicio se obliga al jugador a repartir 5 puntos adicionales sobre los puntos de habilidad base que tiene nuestro Mystical. Esta escala está limitada hasta un máximo de 10 en este menú, siendo la escala final en el juego hasta 100.

2.2 Clases

Al configurar las características innatas añadiendo los puntos adicionales comentados, se especifica la clase base de nuestro Mystical según su parámetro más poderoso. Así pues, al comienzo se puede clasificar a nuestra criatura en una de las 5 clases iniciales: ***Beast***, ***Wizard***, ***Tank***, ***Guardian*** y ***Flash***.

El Mystical pertenecerá por tanto a una clase base derivada de todo lo anterior, y esto no variará, pero entrenando o combatiendo, podrá desbloquear nuevas clases. Al realizar ciertas actividades con la criatura, se consigue experiencia de clase, que otorgará al usuario **cristales de clase**. Estos cristales permiten subir de nivel en la clase actual o cambiar a otras clases disponibles. Al desbloquear una clase se desbloquean además nuevos potenciadores relativos a dicha clase.

Utilizando el ejemplo anterior, nuestro Mystical pertenece a la clase base Flash, pues su característica innata más poderosa ha resultado ser la Velocidad. En este caso, la criatura siempre será de clase Flash, y esto no cambiará, teniendo repercusión en su desarrollo y en algunos aspectos más, pero podrá desbloquear otras. Es decir, tendrá una clase base que será fija (**Flash**) y podrá aprender todas las demás clases (por ejemplo, la clase **Ghost**).

Al realizar un cambio de clase, se inicia la carrera en esta nueva clase desde el nivel uno, pudiendo acceder a los potenciadores correspondientes a ésta y su nivel. Cuanto más elevado es el nivel de clase, más fuertes son los potenciadores disponibles. Para acceder a nuevas clases se necesita tener varias clases básicas superadas: para los dobles se necesitan dos clases básicas, para los triples, 3, etc. En teoría estas combinaciones de clase son mas fuertes respecto a su equivalente en nivel a una clase pura, ya que necesitan de mas tiempo para ser accesibles. Además de los requerimientos para acceder a nuevas clases, se necesita cumplir con condiciones para desbloquear nuevas clases.

Para acceder a una clase doble, es necesario haber superado 2 clases base; para acceder a una clase triple, a parte de tener 3 clases base superadas, también se necesitan 3 clases dobles superadas; para los cuádruples se necesitan 4 triples, 4 dobles y 4 base y así sucesivamente.

Hay una serie de potenciadores de clase exclusivos de tienda, que se tendrán que comprar para ser utilizados. Deberemos desbloquear la clase correspondiente en nuestro tablero de clases para poder acceder a ellos en la tienda.

Tabla de razas y clases por stats:	**Clase**
Fuerza	**Beast**
Fuerza mágica	**Wizard**
Defensa física	**Tank**
Defensa mágica	**Guardian**
Velocidad	**Flash**
Fuerza = Fuerza mágica	**Paladin**
Fuerza = Defensa física	**Warrior**
Fuerza = Defensa mágica	**Hero**
Fuerza = Velocidad	**Raptor**
Fuerza mágica = Defensa física	**Monk**
Fuerza mágica = Defensa mágica	**Sorcerer**
Fuerza mágica = Velocidad	**Thunder**
Defensa física = Defensa mágica	**Adamantai**
Defensa física = Velocidad	**Fighter**
Defensa mágica = Velocidad	**Ghost**
Fuerza = Fuerza mágica = Defensa física	**Giant**
Fuerza = Fuerza mágica = Defensa mágica	**Boss**
Fuerza = Fuerza mágica = Velocidad	**Hunter**
Fuerza = Defensa física = Defensa mágica	**Conqueror**
Fuerza = Defensa física = Velocidad	**Ninja**
Fuerza = Defensa mágica = Velocidad	**Saviour**
Fuerza mágica = Defensa física = Defensa mágica	**Demon**
Fuerza mágica = Defensa física = Velocidad	**Druid**
Fuerza mágica = Defensa mágica = Velocidad	**Bearer**
Defensa física = Defensa mágica = Velocidad	**Genius**
Fuerza = Fuerza mágica = Defensa física = Defensa mágica	**Colossus**
Fuerza = Fuerza mágica = Defensa física = Velocidad	**Knight**
Fuerza = Fuerza mágica = Defensa mágica = Velocidad	**Protector**
Fuerza = Defensa física = Defensa mágica = Velocidad	**Meteor**
Fuerza mágica = Defensa física = Defensa mágica = Velocidad	**Sage**
Fuerza = Fuerza mágica = Defensa física = Defensa mágica = Velocidad	**Emperor**

La criatura estará asociada desde el principio a su clase base inicialmente seleccionada. Cada una de estas clases iniciales conllevan unas bonificaciones especiales que se aplicarán sobre el combate. Con cada intensificación de la clase inicial se mejora un pequeño porcentaje esta bonificación:

Beast:	Bonificación sobre la fuerza de los potenciadores.
Wizard:	Bonificación sobre la duración de los potenciadores.
Tank:	Objetos de combate caen más cerca.
Guardian:	Objetos de combate tienen mayor duración.
Flash:	Recompensa de dinero mayor.

2.3 El tablero de clases

El origen de las criaturas afecta al diseño de la forma del tablero de clases; tiene una apariencia circular (emulando al planeta), en la cual aparecen las clases de nivel 1 en la superficie, y el objetivo es llegar a la clase central (el núcleo). Al seleccionar una clase de nivel 1, se desplegarán los niveles superiores de dicha clase. Si el jugador selecciona "Ver todo", todas las clases se desplegarán y se podrá ver el tablero completo, viendo iluminadas aquellas clases desbloqueadas mediante cristales de clase. El objetivo es ir desbloqueando las clases con cristales de clase, que se podrán obtener tras el combate o el entrenamiento.

3) Cuidado, entrenamiento y/o combate

A continuación se explica el apartado que determina la experiencia principal nivel jugable: cuidados de la criatura, entrenamientos y/o combates.

3.1 Cuidar a un Mystical

Aunque estas criaturas encierran poderes ancestrales, la desaparición de éstos los hace muy dependientes de los humanos. El jugador podrá (y deberá) cuidar a su Mystical para que presente un estado saludable. Los cuidados influirán sobre la capacidad de la vida de la criatura, en el entrenamiento y en el combate.

La siguiente tabla muestra los parámetros ocultos de un Mystical, que se verán afectados por los cuidados.

Cuidado	**Stats ocultos**	**En entrenamiento y/o combate**
Alimentar	Sobre la *capacidad de vida* (ampliable a comida específica)	Podría incidir en los stats principales (*Fuerza y Defensa*)
Lavar	Sobre la *capacidad de vida*	Podría incidir en los stats principales (*Velocidad*)
Jugar	*Afinidad* (Pobre, Baja, Media, Alta, Muy Alta)	% de suerte y poder sobre los golpes
Acariciar	*Afinidad* (Pobre, Baja, Media, Alta, Muy Alta)	% de suerte y poder sobre los golpes

3.2 Entrenar a un Mystical

Realmente, un Mystical no tiene por qué combatir con otro para recuperar su misticismo. Aunque con el combate aprende más rápido, con el entrenamiento también puede ir mejorando sus habilidades y su nivel (niveles de 1 a 100).

La siguiente tabla muestra los tipos de entrenamiento.

3.2.1 Entrenamiento general (DEMO)	Modo *Survival*
3.2.2 Entrenamiento de Fuerza	Por determinar
3.2.3 Entrenamiento de Fuerza especial	Por determinar
3.2.4 Entrenamiento de Resistencia	Por determinar
3.2.5 Entrenamiento de Resistencia especial	Por determinar
3.2.6 Entrenamiento de Velocidad	Por determinar

3.2.1 Entrenamiento general

El entrenamiento general será el único entrenamiento que se incluirá en la DEMO. En él podremos subir nuestro nivel místico y conseguir cristales de clase. Haremos luchar a nuestra criatura contra hordas de enemigos hasta que decidamos parar o hasta que las fuerzas le fallen. Se basa en un Modo *Survival* en donde nuestro Mystical tendrá que aguantar el ataque de un número de enemigos que irá haciéndose mayor con el paso del tiempo.

A continuación se explica el transcurso de un combate contra otro Mystical. Aunque con el entrenamiento también es posible, la competición con otra criatura es una manera más rápida de aumentar el nivel de misticismo y el nivel de clase de nuestro Mystical.

3.3.1 Comienzo del combate

El primer jugador creará una partida (y especificará las opciones) y el segundo podrá unirse a ella. Tras la unión de ambos jugadores se podrá calibrar la partida.

A continuación se muestra una tabla con las opciones posibles para el combate (especificadas al **crear** la partida).

Parámetros	Opciones
Duración del combate	30 segundos a 10 minutos
Aparición de objetos consumibles	Sí/No
Frecuencia de aparición de objetos consumibles	Baja, Normal, Express
Recarga de energía	Normal, Express

3.3.2 El combate

Una vez iniciado el combate, los dos Mysticals lucharán utilizando sus diferentes habilidades. El final del combate lo determinará la derrota de uno de ellos o bien que se agote el tiempo. Salvo que lo indique el jugador en el modo de batalla elegido en Colliseum, la lucha estará calibrada de manera que el Mystical de más nivel tenga más posibilidades de ganar. Obviamente, si compiten dos Mysticals cuyo nivel de misticismo es muy diferente, el combate será mucho más sencillo para el de nivel más alto. Consideramos que es una manera de premiar al jugador que ha dedicado más tiempo a entrenar a su criatura.

En el combate pueden aparecer objetos consumibles y reliquias de batalla. Al coger los objetos consumibles en el área de combate, éstos se añadirán a la ranura de objeto recogido situada a la izquierda de la pantalla, y dicho objeto se activará cuando la toquemos, otorgando al Mystical una potencia adicional en algún parámetro, rellenando su barra de vida o equipando un arma/escudo durante un tiempo determinado. Podremos coger y activar más de un objeto consumible, aunque solamente se podrá recoger y activar uno antes de poder coger otro. Respecto a las reliquias de batalla, podrán ser objetos vendibles o equipo de combate coleccionable y equipable tras la batalla.

3.3.3 Al terminar la batalla

Al terminar el combate, cada jugador recibirá experiencia de nivel, experiencia de clase y quizá algún objeto equipable y/o reliquia (esto dependerá de los logros en combate). Los puntos de experiencia de nivel permiten subir el nivel místico al Mystical, aumentando de esta manera sus características según su clase. Se trata de una barra tradicional que con cada punto de experiencia va aumentando. Si se alcanza un número suficiente de experiencia, la criatura aumenta en una unidad su nivel y los stats base aumentan acorde a la clase base y la segunda clase.

La experiencia de clase determina los cristales de clase conseguidos y su funcionamiento se puede comparar con el de la barra de experiencia para subir de nivel místico. Una vez se llena la barra se consigue un cristal de clase que se podrá utilizar en el tablero de clases. Los dos jugadores reciben todos sus puntos de experiencia calculados mediante un multiplicador que se aplica a la experiencia base. La experiencia base depende del nivel de la criatura rival. El multiplicador conseguido depende de la valoración del combate.

Realizando ciertos logros en el campo de batalla el multiplicador podrá aumentar. Obviamente el jugador ganador del combate tendrá un multiplicador especial de victoria y su experiencia obtenida siempre será superior al perdedor, es decir, se premiará ganar el combate por encima de cualquier otro logro.

9.5 EL MUNDO DEL JUEGO

El juego se desarrolla en el mundo real. No se hace uso de un mundo paralelo ni ficticio. Como se ha comentado antes, los Mysticals son seres ancestrales que viven cerca del núcleo terrestre. Su existencia está ligada a la existencia de los humanos, es decir, cada humano que nace crea un alter ego.

LOS MYSTICALS

Como se ha comentado ya, son unos seres muy parecidos a los animales, que viven cerca del núcleo terrestre de manera paralela a los humanos. Desde la creación de la tierra, dichos seres eran los encargados de mantener la estabilidad en el núcleo del planeta. Eran seres cuyos poderes ancestrales les permitían, sólo con su mera presencia, controlar la energía del núcleo, pero una tormenta solar ha mermado sus habilidades innatas y se han debilitado, dejando a la Tierra en grave peligro.

Ahora, los Mysticals deben recordar de alguna manera ese poder latente que encierran, y han salido a la superficie para pedir ayuda a sus "hermanos".

Los humanos

En el juego, los humanos son tal y como los conocemos. De hecho, se podría decir que son el personaje principal del juego, ya que gracias a ellos, los Mysticals pueden ir recuperando su poder. No conocían la existencia de estos seres hasta que fueron avisados por ellos. Pueden ver a sus Mysticals gracias a algunos dispositivos electrónicos alterados por la tormenta solar.

La catástrofe

La tormenta solar ha debilitado a los Mysticals, y en consecuencia, ha desestabilizado el núcleo del planeta, haciendo que se vuelva imprevisible y peligroso. Los Mysticals han salido a la superficie para que los humanos les ayuden a recuperar su misticismo ancestral, de manera que se pueda retornar al estado de calma en el que se encontraba el planeta antes de la tormenta.

9.6 INTERFAZ DE USUARIO

A continuación se muestra todo lo relativo a la interacción Humano-Máquina, así como los diferentes menús del juego.

Menus

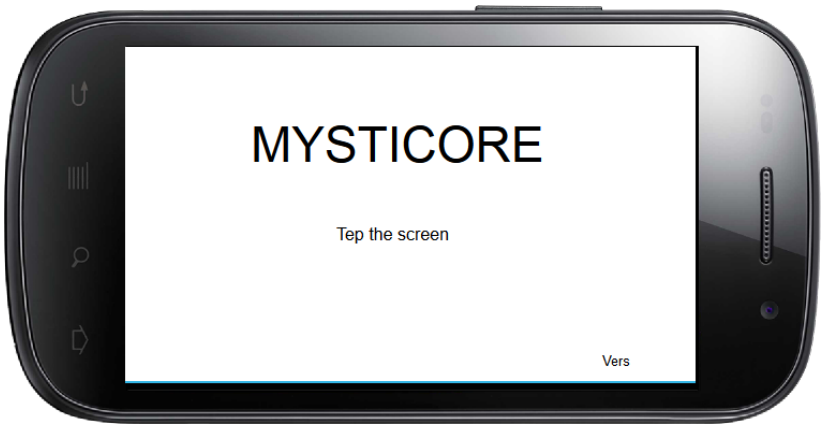
A continuación se muestran los diferentes menús del juego, con una explicación al detalle de cada uno de ellos.

Mapa de menús

El mapa de menús muestra un esquema general de navegación entre los diferentes menús del juego:

Con las flechas se muestra el orden de navegación posible entre los diferentes menús, de manera que se pueda volver o no a los anteriores, o pasar a los siguientes. En la figura, todos los módulos que no tienen descendencia, son menús de opciones (Options, Shop…) o bien son acciones directas que pueden realizarse (Feed, Wash…), es decir, no tienen submenús en su interior.

A continuación se analizará cada uno de los menús mostrados en el árbol. **No presentan el diseño final**; únicamente muestran un boceto de la información a mostrar. La pantalla principal que se muestra al ejecutar el juego. Muestra el nombre del juego y comunica al usuario que toque la pantalla para avanzar. Además, muestra la versión del juego actualmente instalada.



Como comentábamos antes, en el diseño final es posible que el título y el fondo sean animados. Avanzando desde este menú, vamos a la pantalla **Main Menu**.

Main Menu

Desde **Main Screen** llegamos a este submenú. Se mantiene el nombre del juego y la versión, lo único que cambia son las dos opciones que se ofrecen: Play y Options.

Si pulsamos en **Play** o en **Options** accederemos a otras ventanas. Desde la opción Play se va al menú **Play,** desde la opción Options se acceden a las opciones generales del juego.

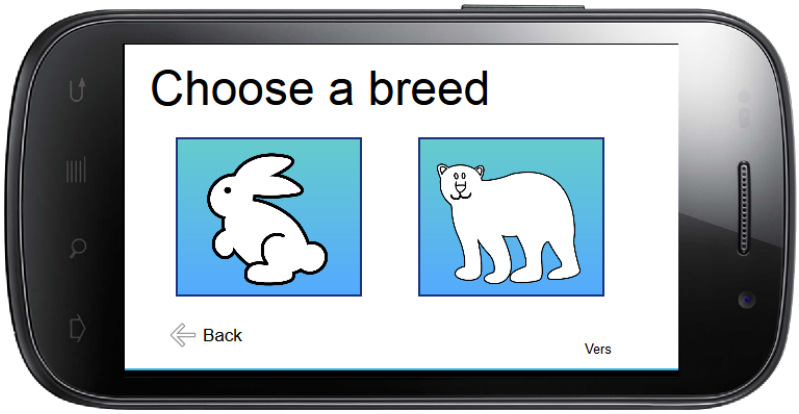
Play

Desde la opción Play, accedemos a este menú. Se mantiene el título del juego y la versión, y se añade una opción de retorno al menú anterior (Main Menu) y dos opciones más para seguir con el juego: Continue y New Game.

Si tenemos datos del juego guardados, podremos pulsar en la opción Continue, y accederemos al menú **Standard Screen**. Si pulsamos en la opción New Game, pasaremos al menú **Choose a breed** para seleccionar una raza y empezar un juego nuevo.

Choose a breed

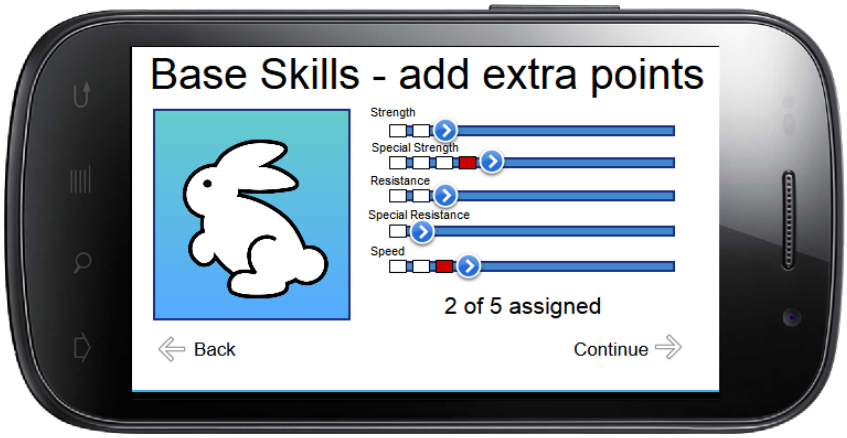
En este menú podremos seleccionar la raza de nuestro Mystical. Para la versión final, es posible que se incluya la opción de editar nuestra criatura una vez seleccionada su raza, pero para la DEMO se darán dos predefinidas. Se muestra el título del menú, las razas disponibles, una opción de retorno y la versión instalada del juego.



Una vez seleccionada una raza, podremos acceder al siguiente menú, llamado **Base Skills**, donde podremos establecer las habilidades base de nuestro Mystical.

Base Skills

En este menú podremos ver los stats base de nuestro Mystical (varían según la raza) y añadir cinco puntos extra. Se muestra el título del menú, la imagen del Mystical, sus parámetros con los puntos base + extra, una opción de retorno y otra para continuar y el total de puntos asignados.

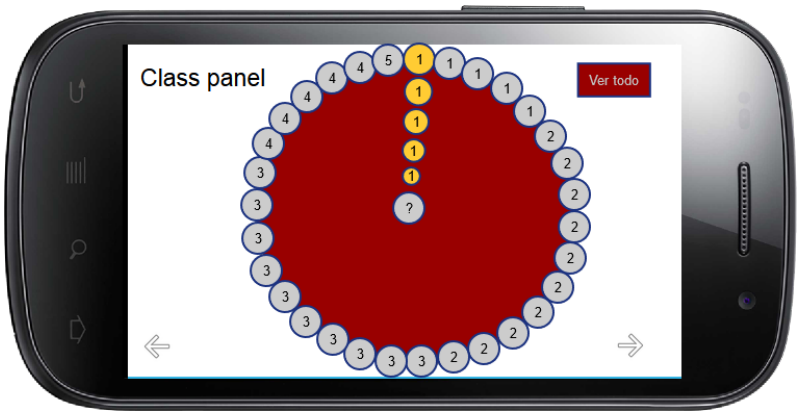


Si pulsamos en la opción Back, volveremos al menú anterior (**Choose a breed**). Por el contrario, si repartimos los cinco puntos podremos pulsar en Continue y acceder al siguiente menú, llamado

Class Panel.

La acción en este menú depende de las elecciones en el menú anterior (**Base Skills**). Si hemos repartido los puntos base de manera que dos o más características innatas (Fuerza, Resistencia…) queden al mismo nivel, en este menú se nos dará a elegir en qué clase queremos que nuestro Mystical active el primer Cristal de clase. Por lo contrario, se activará automáticamente la clase correspondiente.

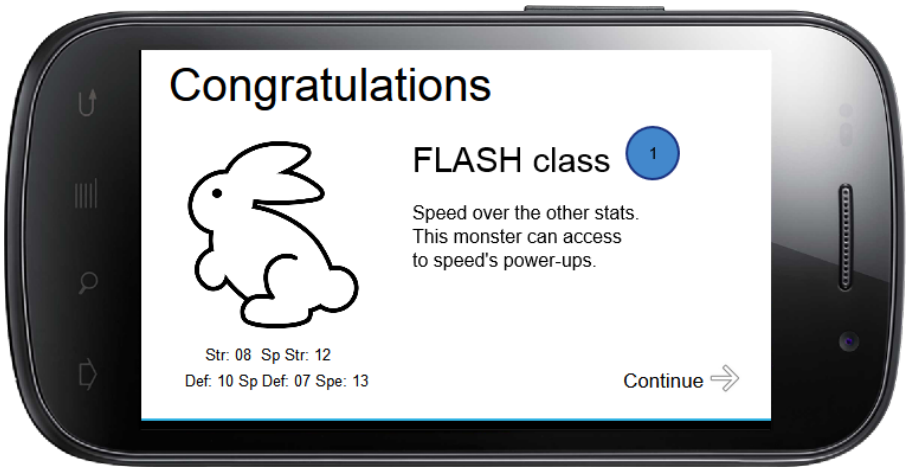
Se muestra el título del menú, el tablero de clases (una circunferencia que emula la apariencia de la tierra, con pequeñas bolas activables), un botón con la opción *Ver todo*, que muestra todas las clases activadas (en este momento todavía no habrá ninguna), y dos opciones más, de retorno y para continuar.



Si pulsamos en la opción Continue, pasaremos al menú **Base Skills Congrat**.

Base Skills Congrat

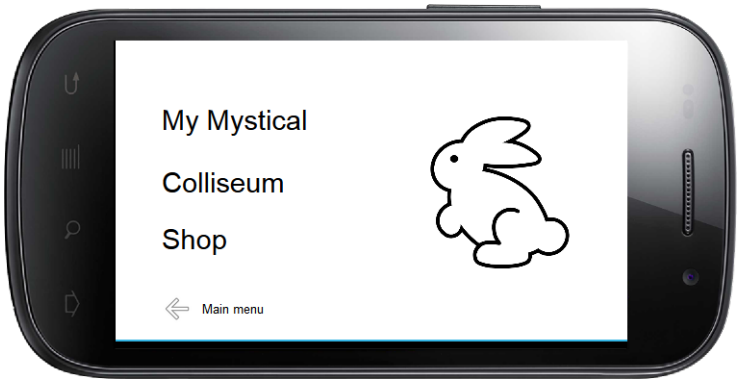
A este menú se accede desde **Class Panel** (sólo al crear el juego). Se muestra un resumen de lo creado: una imagen del Mystical, los stats iniciales definidos, y la clase a la que pertenece con una pequeña explicación.



Desde este punto no se puede volver atrás. Si pulsamos sobre la opción Continue, pasaremos al menú **Standard Screen**.

Standard Screen

A este menú se llega desde **Base Skills Congrat** (en el caso de haber empezado un nuevo juego) o desde la opción Continue en el menú **Play**. En este menú se muestran varias opciones: My Mystical, Colliseum y Shop, además de una opción de retorno al menú **Main Menu**.



Si pulsamos en la opción *My Mystical* accederemos al menú **My Mystical**, si pulsamos en la opción *Colliseum* accederemos al menú **Colliseum** y si pulsamos en Shop, accederemos a **la tienda**.

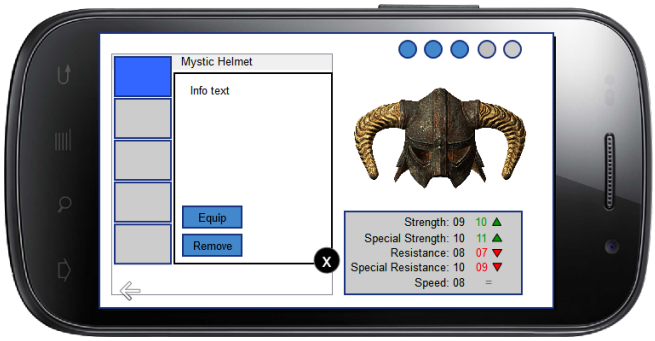
My mystical

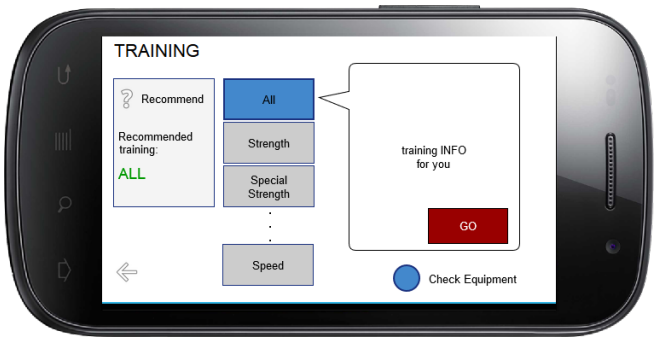
A este menú se accede desde el menú **Standard Screen**. Se muestran varias acciones y opciones. Las acciones son aquellas que podemos ejecutar en la misma pantalla, como son**: lavar, alimentar, jugar y acariciar**. Las opciones disponibles son las de Equipo (En la imagen se muestra como Equip.) y la de retorno al menú anterior (**Standard Screen**).

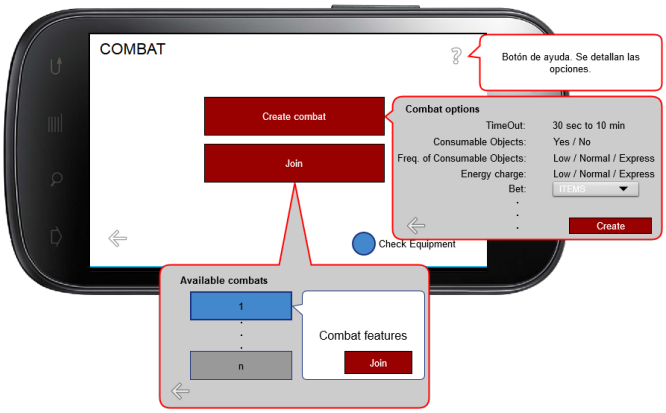
Si pulsamos sobre la opción Equip, podremos acceder al menú de **CombatEquip**.

CombatEquip

A este menú se accede desde **My Mystical**. Se muestran los diferentes objetos equipables, información sobre ellos y las mejores que tienen sobre nuestro Mystical. Además, se indica cuáles de ellos están equipados, dando las opciones de equipar o quitar. Aparece también de nuevo la opción de volver al menú anterior **(My Mystical).**







La ventana de combate (HCI)

La ventana de combate está compuesta por un joystick digital (3) que permite el manejo de la criatura en el entorno de la realidad aumentada. Además existe un botón de ataque básico (1) y un botón de defensa (2).

Aparte de estos dos botones se dispone de dos slots para almacenar dos potenciadores (4 y 5). Estos ataques se basan en la fuerza especial y la resistencia especial respectivamente, y suelen ser ataques sobrenaturales y mágicos. Los potenciadores se asignan también antes de iniciar el combate. Según las características del potenciador, este tardará un tiempo determinado en recargarse para volver a utilizarse. Potenciadores devastadores tardan más tiempo en recargarse que ataques menos potentes.

Durante el desarrollo del combate aparecerán objetos especiales (obj) que afectan directamente a las características de las criaturas (pociones, espadas etc...). Estos objetos se almacenan hasta su activación en una casilla en la pantalla de combate (6), desde la cual se podrán activar o desechar con un movimiento determinado.

Gráficamente: