Escuela Politécnica Superior

Control remoto de realidad aumentada mediante un robot

Grado en Ingeniería Multimedia

Trabajo Fin de Grado

Autor:
Miguel Soriano Sanz

Tutor/es:
Mireia Luisa Sempere Tortosa

Universitat d'Alacant
Universidad de Alicante

Septiembre 2016

Resumen

Hoy en día estamos experimentando un cambio drástico en la manera en la que funciona y se desarrolla la tecnología. Explotar nuevos campos tecnológicos y desarrollar proyectos en ellos ya no está al alcance de unos pocos, internet y ciertos dispositivos han conseguido que usuarios básicos sean capaces de crear sus propios proyectos sin necesidad de un gran presupuesto.

Este trabajo es una muestra de ello al juntar en un solo proyecto individual varias tecnologías como son la robótica, redes, modelado, animación y tecnologías web. Dispositivos como arduino o Raspberry Pi y comunidades de desarrolladores en internet como GitHub o Stack Overflow han sido utilizados en este proyecto y lo han hecho posible sin necesidad de invertir una gran cantidad económica.

En este proyecto en cuestión se ha realizado el prototipo de una plataforma de juegos que permite integrar la realidad aumentada en los videojuegos de una manera nunca vista. El prototipo se basa en un modelo 3D que será controlado por el usuario en el mundo real a través de una aplicación web. Esto es posible gracias a un robot dirigido mediante WiFi y la aplicación web, el cual posee una cámara que enviará imágenes en tiempo real a la aplicación dando la sensación de inmersión del modelo 3D en el mundo real. Además se ha elaborado un ejemplo de minijuego en realidad aumentada accionado mediante un marcador.

Justificación y objetivos

Cualquier trabajo de fin de grado debería recoger al menos una o más aptitudes desarrolladas durante la carrera para demostrar los conocimientos que se han adquirido en la misma. Tener esto en mente permite dar salida a diferentes proyectos muy variados entre sí que no se limitan a una aplicación móvil o web o a un simple videojuego. Mezclar varías ramas de conocimiento permite además dar una sensación de utilidad de la carrera en el sentido de la diversidad de conocimientos que ofrece.

La elección de este trabajo de fin de grado buscaba encontrar un proyecto donde plasmar lo aprendido tanto por mi cuenta como en la carrera en un solo trabajo con el que me sintiese a gusto y que de verdad me apeteciese desarrollar. Me alegra haber encontrado un proyecto donde poder incluir desarrollo web, realidad aumentada, robótica y modelado y llevarlo a cabo en forma de prototipo de videojuego.

El **objetivo principal** de este proyecto consiste en la realización de un videojuego en realidad aumentada. En la mayoría de juegos del mercado la realidad aumentada se usa para añadir NPC que el usuario debe capturar, disparar o enfocar con la cámara. En este juego el usuario controlará al personaje principal a modo de juego en 3ª persona con la cualidad de que el mundo por el que se va a mover será el mundo real. Por lo tanto se pretende dar la sensación al jugador de que está controlando un personaje virtual por su entorno y en definitiva convertir su casa en el mundo del juego.

Esta idea se ha hecho posible con la ayuda de un robot que será lo que verdaderamente controle el jugador y que irá "siguiendo" al personaje virtual para permitir al usuario explorar su mundo desde una nueva perspectiva. Dicho control se realizará desde una aplicación web habilitada para usar tanto en ordenadores como en dispositivos móviles permitiendo al usuario jugar como más le apetezca.

Agradecimientos

Me gustaría agradecer en este documento a todas las personas que de una manera u otra han hecho posible tanto este trabajo como mi paso por la carrera.

En primer lugar agradecer a mi familia el apoyo que siempre me ha dado permitiéndome hacer lo que de verdad quería y dejándome tomar mis propias decisiones. En especial a mi madre por suplir todos los gastos que mi paso por los estudios han conllevado y el apoyo que siempre me ha dado.

En segundo lugar a mis amigos y compañeros de clase los cuales me han proporcionado unos cuatro años inolvidables llenos de momentos increíbles y nuevas amistades. Sobre todo a mis compañeros de piso y del ABP que han sido los que más se repiten en dichos momentos. También agradecer a mi pareja que siempre me apoyase en todo y el haber estado junto a mi todo este tiempo.

Por ultimo agradecer a aquellas personas que han ayudado directamente en el desarrollo de este trabajo como es mi tutora Mireia Luisa Sempere Tortosa y a toda la comunidad y foros de internet que no paran de crecer y ofrecer su ayuda a cambio de nada haciendo posible que cada vez más y más personas puedan adquirir conocimientos de manera gratuita y ser autodidactas.

Citas

"No hay nada noble en ser superior a tus semejantes, la verdadera nobleza está en ser superior a tu antiguo yo"

Ernest Hemingway

Índice de contenidos

Índice de figuras

1 Introducción

Hoy en día vivimos en un mundo donde la tecnología avanza cada vez más y cada vez más rápido en la mayoría de los sectores de la sociedad. Esto se debe en parte a que cada vez que se desarrolla una nueva tecnología esta es usada en varios ámbitos, así que el ocio, la medicina o la educación entre otros se pueden favorecer de tecnologías desarrolladas para un fin concreto en otro sector.

La realidad aumentada es una de las nuevas tecnologías que están teniendo una gran repercusión en la sociedad y es un ejemplo de esto. La medicina, el turismo o el ocio son algunos de los sectores que la están empezando a usar con resultados sorprendentes. Puede servir tanto para entretener como para añadir información al mundo que nos rodea y está siendo cada vez más utilizada a la vez que mejorada, favoreciendo su integración en nuestro mundo.

Así mismo el sector del ocio es uno de los que más se aprovechan de las nuevas tecnologías y está en constante cambio, sobre todo el ocio digital y tecnológico. Podemos ver como en los últimos 50 años los ordenadores han evolucionado sorprendentemente y como lo juegos han ido cambiando, adaptándose a esta evolución. De ver simples píxeles por pantalla a una imagen que no podríamos decir claramente si es real o creada por computador.

Pero sin duda cuando más partido se le saca a una nueva tecnología es cuando se combina con otras para hacerla más completa y útil. Es el caso de la realidad virtual que usa acelerómetros y otras herramientas para ser así mucho más inmersiva. Y es en este sentido donde la robótica entra en juego ya que la mayoría de herramientas necesitan de la robótica para ser desarrolladas.

La robótica a nivel de usuario ha avanzado de tal manera que cualquier persona con conocimientos básicos es capaz de crear un robot gracias a Arduino y Raspberry Pi. Esto hace que cada día podamos encontrarnos con sorprendentes inventos como la domotización de una casa o el internet de las cosas llevado hasta límites increíbles, y todos ellos desarrollados por usuarios normales, sin una gran multinacional o un importante equipo de desarrollo detrás.

Este trabajo de fin de grado intenta juntar todo lo visto anteriormente y hacer uso de la realidad aumentada orientada al ocio con ayuda de herramientas robóticas para mejorar las opciones de las que disponemos y obtener el prototipo de lo que puede ser un videojuego innovador que cambie la forma de jugar con la realidad aumentada.

2 Marco teórico

En este proyecto, al tocar tantas áreas tan diversas como son la electrónica, la programación web, la realidad aumentada y el modelado, hay que tener en cuenta una gran cantidad de conceptos. A continuación se dará un repaso a los más importantes y los motivos por los que han sido seleccionados para el desarrollo del mismo.

2.1 Lenguajes de programación Web

Hoy en día existen una gran variedad de alternativas en lenguajes web, no obstante son los básicos y estandarizados los que a su vez son más potentes y más utilizados, por lo que la mayor variedad se encuentra a la hora de elegir librerías y frameworks de estos lenguajes.

2.1.1 HTML 5

HTML es el lenguaje de programación web basado en etiquetas más utilizado. En su versión actual, la 5, ofrece grandes posibilidades y nuevas etiquetas que ayudan a estructurar la página y a añadir contenido.

La nueva etiqueta que más nos interesa de HTML 5 es la etiqueta <canvas> que permite dibujar en ella tanto en 2D como en 3D haciendo uso de OpenGL en su versión para web, WebGL.

2.1.2 JavaScript

JavaScript es el lenguaje más famoso de programación web del lado del cliente. Permite agregar interacción al usuario con las páginas web creadas con HTML.

Debido a lo extendido que está este lenguaje, hay muchas librerías y frameworks distintos por los que se puede optar para programar en JavaScript. Aunque vanilla JavaScript, que es como se llama a JavaScript básico sin ninguna librería, es más rápido al no tener ninguna capa intermedia, las librerías existentes como JQuery o AngularJS son tan potentes que merece la pena su utilización. Además los ordenadores de hoy en día están preparados para soportar la posible carga extra que puede suponer el uso de estas librerías.

En este proyecto se ha optado por la utilización de JQuery ya que permite acceder y utilizar elementos de la página de una manera muy rápida, además de que JQuery ofrece un cómodo sistema para realizar llamadas AJAX y que Three.js, la librería utilizada para facilitar el uso de WebGL, también hace uso de JQuery.

2.1.3 PHP

PHP es un lenguaje muy popular especialmente para el desarrollo web del lado del servidor. Compite directamente con ASP utilizado en servidores web de Microsoft. Hoy en día está aumentando la popularidad de usar JavaScript también del lado del servidor ya que así un mismo desarrollador podría dedicarse tanto al frontend como al backend con solo un lenguaje, no obstante sigue sin ser la opción más popular por el momento.

La mayoría de servidores web como Apache o Nginx son frecuentemente utilizados con PHP y será el usado en este proyecto ya que es el lenguaje que más domino para programación del lado del servidor de los anteriormente mencionados.

2.2 Realidad aumentada

La realidad aumentada (RA) es un término que se está popularizando mucho y que está teniendo un gran impacto últimamente gracias al avance de esta tecnología en conjunto de su hermana, la realidad virtual, a veces confundidas y a juegos que la emplean tan populares como el nuevo Pokémon GO.

La RA consiste en mezclar en un entorno real objetos virtuales en 3D dando la ilusión al usuario de que dichos objetos verdaderamente se encuentran en la realidad o aportando información extra sobre el entorno. Por el contrario la realidad virtual utiliza un entorno completamente virtual, intentando que el usuario se sienta como si estuviese en dicho entorno ficticio y se abstraiga de la realidad.

Las aplicaciones de la RA van desde el ocio y el entretenimiento a aplicaciones utilizadas en la medicina, la educación o la ingeniería.

2.2.1 RA en web y WebGL

La manera más común de mostrar la RA en web es usando la etiqueta canvas de HTML5 junto con WebGL. WebGL se podría definir como una API desarrollada en

JavaScript que permite el uso de la implementación nativa de OpenGL en los navegadores.

Para el uso de WebGL se han desarrollado diversas librerías o motores gráficos que facilitan su implementación, la más popular hasta el momento es Three.js, la cual todavía está en desarrollo. Al comienzo del proyecto la versión de Three.js era la R77 mientras que en el momento de escribir estas líneas la versión actual es la R79.

2.2.2 RA en videojuegos

Hoy en día el uso de RA en videojuegos está creciendo rápidamente y cada vez más a menudo vemos nuevos juegos que hacen uso de esta tecnología. La RA no simplemente añade nuevas funcionalidades a los juegos de siempre si no que permite cambiar la manera en la que los usuarios juegan. Prueba de ello es el nuevo Pokémon GO que está haciendo que millones de usuarios salgan a la calle a jugar a un videojuego estableciendo un antes y un después en la historia de los videojuegos.

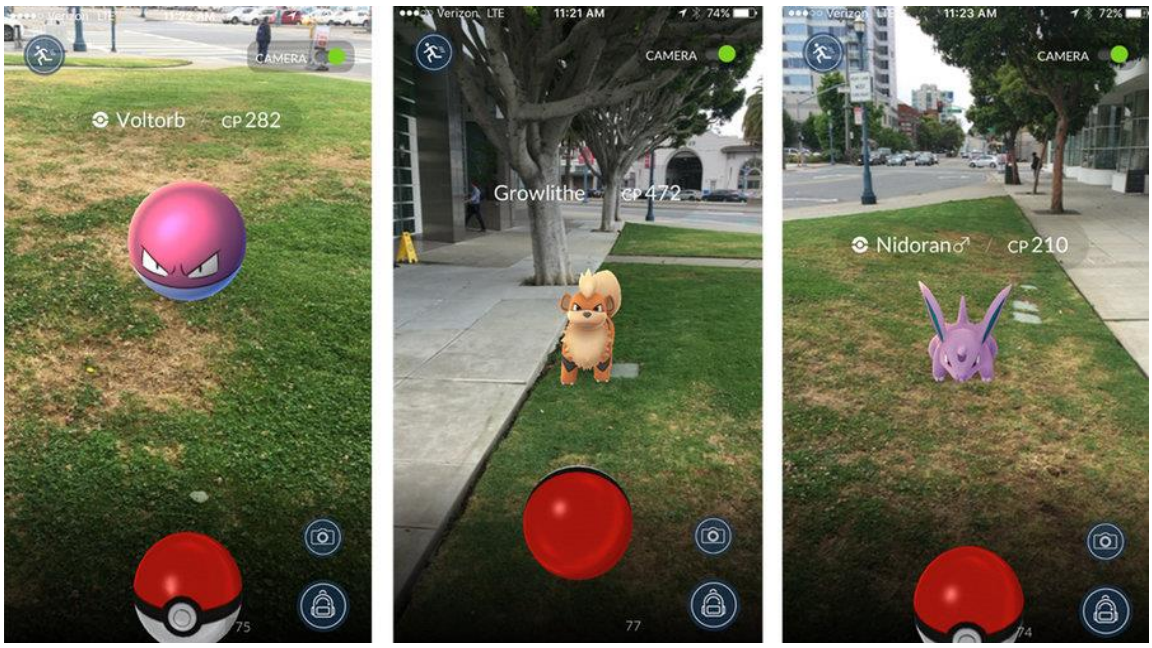

Figura 1 - Juego de Pokémon Go
Fuente: Niantic Labs

La mayoría de los juegos que emplean RA la utilizan para añadir ciertos elementos que el usuario debe encontrar, disparar o capturar, pero el usuario nunca tiene el control de dichos elementos por lo que este proyecto resulta ser una innovación en ese

sentido en cuanto al uso de la RA en los videojuegos, permitiéndonos tomar el control del personaje en RA y moverlo libremente por el mundo, en este caso, el mundo real.

2.3 Modelado y animación

El mundo del modelado y la animación por computadora ha cambiado drásticamente en la última década con nuevos métodos y tecnologías que permiten realizar cada vez trabajos más reales en cuanto a gráficos y físicas. Los videojuegos han evolucionado de simples píxeles pintados en pantalla a enormes entornos virtuales modelados completamente por ordenador.

Figura 2 - Evolución de los videojuegos.
Fuente: Taringa.

Son varias las herramientas y programas que nos permiten modelar en 3D. Las más populares por el momento son 3Ds Max, Maya y Blender. La elección de un software 3D u otro depende de cada usuario y de a lo que esté acostumbrado ya que aunque se puede llegar a un mismo resultado con los tres, cada software cambia el estilo de trabajo, las herramientas y el modo de emplearlas. Este proyecto ha sido desarrollado con Blender ya que no solo es del que más conocimientos se disponían sino que además la licencia de Blender es gratuita para uso comercial además de ser de código abierto lo que permite a la comunidad desarrollar sus propias herramientas y plugins.

2.4 Arduino

Arduino es una plataforma electrónica open source basada en easy-to-use que permite la realización de prototipos y proyectos electrónicos a un bajo coste. Al ser open source cualquiera puede consultar su diseño y modificarlo o crear sus propias versiones de arduino orientadas a su proyecto.

El lenguaje de programación de arduino es muy parecido a processing y está basado en C++. Tanto el lenguaje de programación como el IDE de arduino se idearon para poder poner en marcha proyectos de la manera más fácilmente posible, sin gastar mucho tiempo en aprendizaje.

Hay varios modelos de arduino para elegir según la envergadura del proyecto a realizar. Entre los distintos modelos varía el tamaño, el precio, la capacidad de procesamiento y los posibles componentes que pueda incorporar desde un principio la placa. Así por ejemplo podemos encontrarnos con el Arduino UNO, el más común de todos y con un tamaño medio, con 14 pines digitales y 6 analógicos ideal para iniciarse en arduino y para cualquier proyecto medio que se tenga en mente. El Arduino MICRO es una versión a pequeña escala que nos permite realizar proyectos donde se requiere un tamaño menor y dispone de 20 pines digitales de los cuales 12 pueden ser usados como analógicos, en contra posición debido a su tamaño es necesario soldar los cables a los pines o usar una protoboard, además de que carece de entrada para la alimentación. En el otro extremo tenemos Arduino MEGA 2560, diseñado para proyectos más complejos ya que dispone de 54 pines digitales y 16 analógicos, además de una mayor capacidad de procesamiento, es el recomendado para crear robots o impresoras 3D.

Para sacarle el máximo partido a arduino disponemos de una serie de módulos y sensores que amplifican las posibilidades. Entre los más comunes están el módulo WiFi o de bluetooth que nos permiten tener conexión inalámbrica con nuestro arduino. Respecto a los sensores hay un sinfín de posibilidades y los más comunes son los de proximidad o movimiento, de temperatura, de humedad, de luminosidad y muchos más sensores oficiales, además de los que cada uno desarrolle gracias a la filosofía open source de arduino.

En este proyecto en concreto se ha usado la placa Arduino UNO por su facilidad de montaje y potencia adecuada, no teniendo necesidad de usar una placa más potente. Respecto a los sensores se ha optado por el sensor de distancia basado en ultrasonidos HC-SR04 ya que es el más común para este propósito. Respecto al módulo encargado de controlar los motores se ha elegido el controlador L298N frente al L9110S ya que este último utiliza la misma fuente para alimentación y control mientras que el L298N puede hacer uso de una fuente de alimentación externa para darle más potencia a los motores.

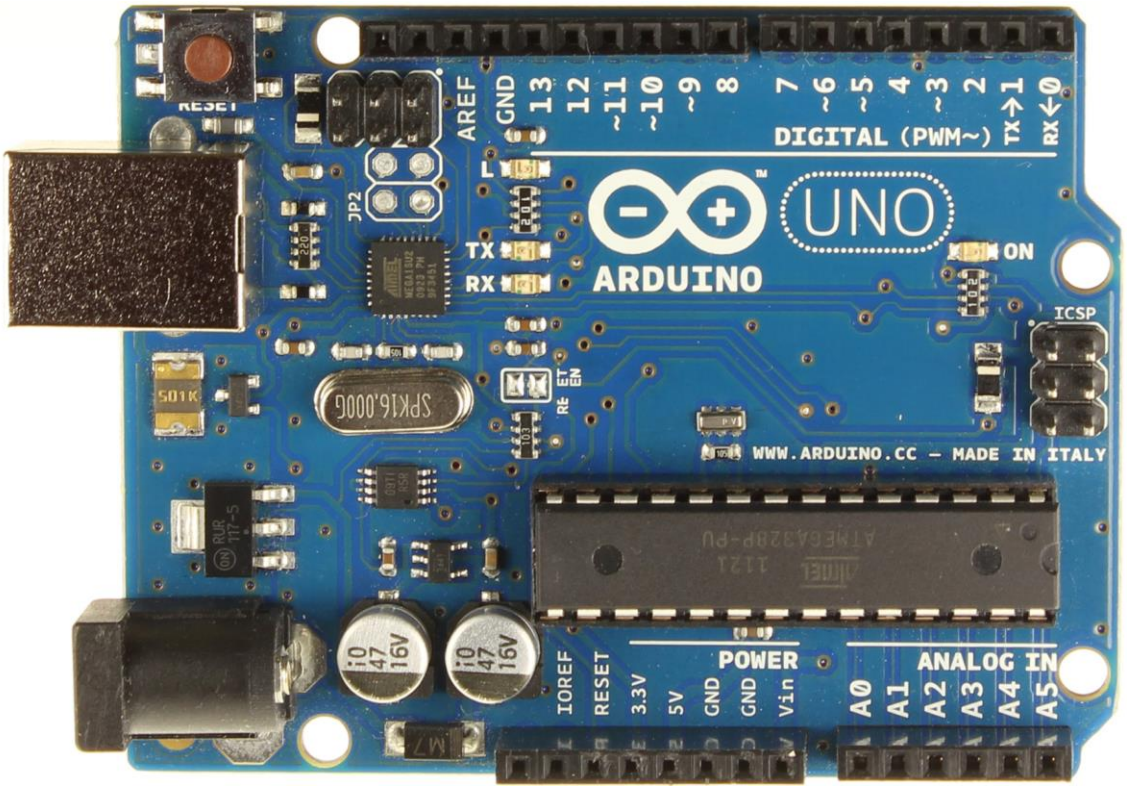

Figura 3 - Arduino UNO
Fuente: http://www.electronicaestudio.com/

2.5 Raspberry Pi

El proyecto de Raspberry Pi nació con la idea de poder hacer llegar un ordenador económico y funcional a todos los lugares del mundo, en especial para su empleo en educación. De esta manera podríamos decir que Raspberry Pi es un mini ordenador al que solo le hace falta instalarle un sistema operativo y conectarle un teclado y un ratón. Según el modelo podemos encontrar que lleva puertos USB, una salida HDMI para la

pantalla, una entrada para alimentación y según el modelo una entrada de Ethernet o antena de WiFi.

Aunque no se ha extendido demasiado en el ámbito de la educación, la Raspberry se ha hecho muy popular en cuanto a proyectos tecnológicos se refiere y ha conseguido que un usuario cualquiera pueda llevar a cabo proyectos de domótica o servidores multimedia con un coste muy bajo, ya que permite tener las capacidades y el procesamiento de un ordenador por el precio de alrededor de 50€. Es muy usual ver proyectos en los que la Raspberry (como servidor) y el Arduino (como actuador) se complementan, por ello se ha popularizado el término Raspduino para denominar estos trabajos.

Al igual que pasaba con arduino, la Raspberry dispone de diferentes modelos a elegir según nuestras necesidades. Podemos encontrar modelos más caros con más capacidad de procesamientos, 4 puertos USB e incluso antena WiFi hasta modelos más sencillos con solo un puerto USB y sin conexión Ethernet.

En este caso se ha elegido el modelo Raspberry Pi 3 B ya que cuenta con WiFi incluido por lo que no es necesario comprar un módulo a parte y su precio es solo 5€ más cara que el modelo anterior. Además este modelo es el más potente del mercado. En cuanto a la cámara seleccionada se ha optado por la oficial de Raspberry Pi, la Raspberry Pi Camera V2. Hay dos modelos posibles a seleccionar, la NoIR con visión infrarroja y la IR. Para el proyecto se descarta la NoIR ya que no es necesario visión infrarroja y los colores de la IR son más reales.

Figura 4 - Raspberry Pi 3 Model B
Fuente: www.modmypi.com

2.5.1 Raspbian como SO

La Raspberry Pi viene sin ningún sistema operativo instalado por lo que deberemos quemar la imagen del sistema operativo que deseemos tener en una tarjeta de memoria y conectarla a la Raspberry Pi para empezar.

Se pueden elegir varios sistemas operativos para la Raspberry y se pueden descargar desde la web oficial de Raspberry en el apartado descargas. https://www.raspberrypi.org/downloads/. Entre las posibilidades tenemos Ubuntu, sistemas operativos de terceros e incluso una versión de Windows 10 para internet de las cosas. No obstante el sistema operativo recomendado y más famoso es Raspbian.

Raspbian es el sistema operativo oficial de la Raspberry. Basado en Debian viene con software matemático y su licencia para ser usado de manera no comercial. Además Raspbian dispone de una opción para principiantes llamada NOOBS. NOOBS es un

instalador de sistemas operativos que contiene Raspbian entre otros sistemas operativos y permite al usuario elegir entre uno y otro fácilmente.

2.5.2 Servidores web para Raspberry Pi

Si se quiere utilizar la Raspberry Pi como servidor web deberemos elegir primero qué servidor es el que más nos interesa. A la hora de elegir hemos de tener en cuenta que al estar basado en Debian, podremos instalar cualquiera de los servidores que se pueden usar en Linux pero que la Raspberry Pi no tiene la misma capacidad de procesamiento que un ordenador potente, por lo tanto la velocidad y lo ligero que sea el servidor deberán anteponerse a la potencia del mismo.

Nginx parece una buena elección ya que aunque es algo menos potente que Apache es más ligero y más rápido, ideal para este proyecto en cuestión. Lighttpd también puede ser una alternativa y en ciertas pruebas[1] incluso supera en velocidad a Nginx, pero en general es menos estable.

[1] https://www.jeremymorgan.com/blog/programming/raspberry-pi-web-server-comparison/

3 Objetivos

El objetivo principal del proyecto sería la creación de un videojuego en realidad aumentada. El videojuego se caracterizará por dirigir a nuestro personaje en 3ª persona y donde el mundo de juego es en este caso el mundo real. Esto es posible gracias a la elaboración de un robot que será lo que realmente controle el usuario, pero con la sensación de estar controlando un modelo 3D.

Para conseguirlo se tienen que desarrollar y cumplir diferentes objetivos bien diferenciados por separado y después unirlos e integrarlos en un proyecto conjunto donde cada una de las partes se comunique entre sí de manera fluida. Por lo tanto vamos a ver qué se intentará conseguir con cada parte del proyecto para que el resultado final sea el esperado.

3.1 Desarrollo del personaje

Como en cualquier videojuego, el desarrollo y modelado de los personajes es fundamental en la finalización del mismo. En esta parte se intentará dar forma y vida a nuestro personaje principal teniendo en cuenta ciertas características para que su integración sea lo más sencilla posible.

- **Número de polígonos no excesivos:** Ya que nuestro personaje se moverá en tiempo real en nuestro navegador web no se podrá abusar del número de polígonos usados para su creación, ya que esto provocaría un trabajo extra del ordenador pudiendo ocasionar ralentizaciones en la animación y movimiento del personaje que harían de su control algo tedioso.
- **Modelado orgánico:** Como sabemos que el personaje será posteriormente animado deberemos tener cierto cuidado a la hora de modelar extremidades y articulaciones ya que serán las que tendrán que aguantar el movimiento de la malla. Para esto hay que poner especial atención en que los vértices estén bien ordenados en estas zonas y darles suficiente densidad de vértices para permitir deformar esta parte de la malla sin afectar a otras zonas. La ropa también tendrá que ajustarse a la malla del personaje para que en el momento de animarlo el cuerpo no traspase la ropa.

- **Fácil texturizado:** Sabemos que va a haber un trabajo de texturizado y para facilitarlo habrá que seguir diferentes pautas a la hora de modelar. Se intentará mantener una organización de vértices adecuada para que al hacer el *unwrap*[2] de la malla nos salgan estructuras bien diferenciadas. Para esto habrá que evitar el añadir vértices aleatorios que puedan romper el *loop* de vértices en la malla y puedan dar lugar a caras irregulares.
- **Animaciones simples:** Hay que tener en cuenta que al final el personaje se integrará en el videojuego completo y que el usuario podrá controlarlo haciendo que salte, corra o se pare en cualquier momento. Para esto las animaciones tienen que ser simples y estar sincronizadas para evitar cortes entre ellas y facilitar la transición de una a otra de la manera más limpia posible. Para ello al crear las animaciones todas deberán empezar y acabar con el personaje en la misma posición, además de animar en *loop* haciendo que el principio de una animación coincida con su final para que se pueda correr el tiempo que el usuario quiera sin ver cortes ni saltos en la animación cada dos pasos.

3.2 Desarrollo de la aplicación web

La aplicación web es otra de las partes que se deberá implementar y además con especial cuidado ya que será la que junte cada una de las otras partes. Los objetivos en su desarrollo deberán estar enfocados a favorecer tanto al usuario como al proyecto que hay detrás.

- **Usabilidad:** Se deberá tener en cuenta la usabilidad en todo momento. En cualquier videojuego hay que facilitar el control del personaje al usuario haciendo que la pantalla sea adecuada y los controles intuitivos.
- **Aplicación móvil y *responsive*:** Se pretende que el videojuego se adapte a todo tipo de pantallas para favorecer su uso en cualquier dispositivo sin penalización al usuario por cortes de pantalla en pantallas pequeñas o demasiadas zonas vacías en pantallas grandes. Además también se pretende ajustar la aplicación para su uso móvil, para ello se tendrán que usar

[2] La técnica de unwrap se basa en plasmar en un plano 2D una malla 3D. El resultado es similar a aplastar la malla 3D contra un plano.

controles alternativos al teclado que consistirán en botones táctiles sobre la pantalla. Se deberá evitar que dichos controles tapen el mundo al usuario y le permitan jugar de manera cómoda e intuitiva.

- **Aplicación ligera:** Se intentará crear una aplicación lo más ligera posible para que cualquier dispositivo pueda controlar el juego de manera fluida sin penalización por descargas grandes del servidor o por velocidad de procesamiento de imágenes. Para ello habrá que limitar las descargas y el uso de recursos lo máximo posible e intentar que cada llamada al servidor sea útil y necesaria. También será necesario ajustar la recarga del video conseguido de la cámara para tener un video fluido y en tiempo real pero sin excesivas peticiones al servidor.
- **Comunicación ágil con las demás partes del proyecto:** La aplicación web deberá controlar el robot mediante llamadas al servidor y deberá gestionar la recepción de datos para el correcto funcionamiento de todas las partes de manera sincronizada. Para ello se deberá implementar una API ligera y rápida que permita mandar ordenes al servidor de manera asíncrona para no perjudicar el resto de la ejecución y que deberá controlar llamadas repetitivas e innecesarias para ~~para~~ evitar saturar el servidor.
- **Control de acceso:** El servidor deberá controlar el acceso y darle los mandos del robot solo a un usuario a la vez ya que sería inmanejable de manera correcta por de usuarios al mismo tiempo enviando ordenes contrarias.

3.3 Desarrollo del robot

El robot será la parte que permita al usuario manejar su personaje por el mundo real. Para su implementación se deberán cumplir unos objetivos que aseguren su funcionamiento y fiabilidad durante el juego. El robot deberá ser capaz de procesar la información entrante y ejecutarla en un tiempo de respuesta adecuado.

- **Velocidad óptima para el juego:** El robot tendrá que ser capaz de moverse a una velocidad adecuada para el manejo del personaje en realidad aumentada. Si esta velocidad es muy lenta su manejo será tedioso y es posible que los motores no sean capaces de mover el robot a una velocidad

tan baja debido a su peso. De igual manera si la velocidad es muy alta no dará tiempo de reacción al usuario además de que el video se verá distorsionado por el movimiento rápido de la cámara.

- **Disposición de todos los componentes:** Los componentes se tendrán que colocar de manera ordenada no solo para ofrecer una buena impresión visual y no caótica de cables y componentes, sino también para distribuir el peso de forma adecuada y teniendo en cuenta de poner cerca los componentes que tienen que estar conectados entre sí para evitar cables demasiado largos. La distribución del peso es importante ya que cualquier descompensación podría hacer que el robot se ladeara hacia los lados impidiendo su control en línea recta.
- **Capacidad de "ver":** El robot dispondrá de un sensor de distancias además de la cámara para ser capaz de frenar ante una pared y no dejar al jugador seguir avanzando hacia la misma dando la sensación de que al frenar la pared quede justo en frente del personaje. Además también tiene que ser capaz de gestionar la captura de imágenes y su envío al servidor de manera fluida sin saturar ni el servidor ni el procesador de la Raspberry Pi.

3.4 Realidad aumentada

Para dar sensación de inmersión al usuario, la realidad aumentada deberá seguir unas pautas en su desarrollo e implementación. Estos objetivos están enfocados a la correcta visualización de la realidad aumentada.

- **Correcta disposición en el escenario:** Ante un escenario cambiante a deseo del usuario es difícil controlar cada situación y la disposición de los elementos virtuales en el mundo real. Estos elementos deberán colocarse de tal manera que no traspasen paredes o estén inaccesibles. Para ello habrá que usar la posición del personaje que si estará controlada en el entorno gracias al robot y sus sensores. También sería posible usar algoritmos de detección de esquinas para procesar las imágenes y poder diferenciar suelo y paredes. Esto permitiría saber la anchura de un pasillo para limitar el

rango de movimiento del personaje como de otros elementos de realidad aumentada al espacio disponible en la realidad.

- **Sensación de realidad:** Para dar una mejor sensación de realidad los elementos virtuales tendrán que comportarse como si estuviesen en un entorno real. Para ello la generación de sombras en el suelo que nos ayude a posicionarlos visualmente respecto al suelo puede ayudar a dar sensación de realidad. De igual manera habrá que tener en cuenta el uso de la cámara, la visión humana y el movimiento del personaje. De esta manera si el personaje se encuentra en un lateral de la cámara y está desplazándose en línea recta hacía delante, no deberá aparecer completamente de espaldas

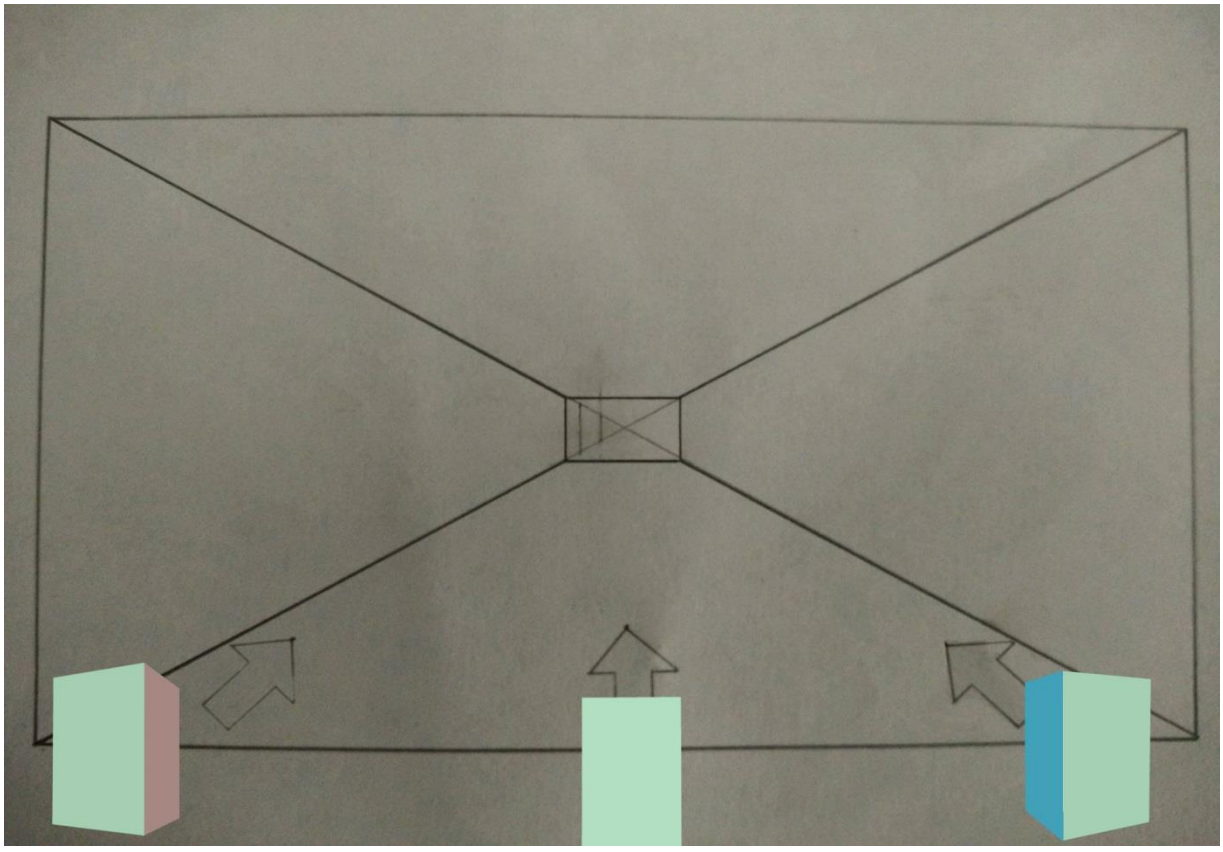

Figura 5 - Perspectiva cónica en un pasillo.
Fuente: Propia.

sino orientado hacia el punto de fuga creado por la perspectiva cónica asemejada a la visión humana.

- **Ser intuitiva:** Hay que tener en cuenta que el usuario no estará acostumbrado a ver el mundo real desde la perspectiva que se le va a ofrecer en el videojuego. Por lo tanto el movimiento del personaje y de los

elementos han de ser intuitivos. El usuario deberá ser capaz de reconocer cuando un elemento se le está acercando o subiendo y a qué velocidad. Por lo que si hay un objeto acercándose al personaje dicho objeto deberá ir aumentando de tamaño conforme se acerca y dicho aumento deberá ser a mayor velocidad si el personaje está corriendo hacía el objeto a la vez que el objeto va hacia el personaje.

4 Metodología

En este apartado se explicarán los métodos seguidos para desarrollar el trabajo así como las herramientas empleadas para gestionar el control de versiones y las tareas realizadas y por hacer.

4.1 Metodología del desarrollo del software

En el desarrollo de este trabajo de fin de grado se ha usado una metodología similar a las metodologías ágiles para el desarrollo del software. Las metodologías ágiles se basan en la realización de entregables funcionales periódicos que permiten ver el estado del proyecto y tomar decisiones respecto a posibles cambios, ya que estas metodologías aceptan cambios en las funcionalidades del sistema incluso en las etapas finales del proyecto.

Para poder seguir una metodología ágil hay que dividir el proyecto en diferentes iteraciones con diferentes etapas en cada una. Descomponer un problema en varios problemas pequeños es una metodología de resolución de problemas que permite centrarse en un punto en concreto y todas sus posibles soluciones que mirando el problema global podríamos haber obviado. Al final de cada iteración se entregará un software funcional y se pasará a la siguiente iteración hasta la finalización del proyecto.

Se ha elegido esta metodología ya que se adapta perfectamente al tipo de proyecto. Al tener distintos apartados tan diferentes entre sí, la organización en diferentes entregables no ha sido nada difícil. De esta manera se ha dividido el proyecto entero en 4 apartados: El personaje, la aplicación web, el robot y la realidad aumentada. Prácticamente cada uno de estos apartados podría desarrollarse de manera totalmente aislada de todos los demás, lo que facilita la implementación de las metodologías ágiles. Cada apartado se ha descompuesto a su vez en diferentes entregables que pudiesen ir desarrollándose de manera iterativa como es los bocetos, modelado de cada parte del personaje, su animación, su integración, etc.

4.2 Gestión del proyecto

Para la realización del proyecto se ha hecho uso de aplicaciones externas que ayudan en la gestión y planificación del mismo. A continuación se expondrán dichas herramientas.

4.2.1 Repositorios y GitHub

Para el control de versiones se ha usado GitHub. Gestionar las diferentes versiones del proyecto es una tarea importante pero que puede resultar tediosa. Normalmente es totalmente imprescindible en proyectos en los que trabajan más de una persona ya que el control de versiones permite estar trabajando simultáneamente y después unir las dos versiones de manera casi automática.

Tener un control sobre las diferentes versiones permite además volver atrás en caso de que alguna versión fallase y localizar el error y el punto y momento exacto de su aparición. Por lo tanto GitHub también es usado como copia de seguridad en caso de que la versión local se borrase o el ordenador fallase, ya que con GitHub se almacena todo nuestro proyecto en la nube y puede ser descargado desde cualquier ordenador. Para la realización de este trabajo se han realizado copias de seguridad en diferentes plataformas como Dropbox y Drive antes de cualquier cambio importante o sustancial, aunque las copias de seguridad más redundantes se han hecho con GitHub.

Hay que añadir que GitHub no solo te permite controlar tus versiones si no que se basa en la filosofía de código abierto y código colaborativo, así que cualquiera puede ver y descargar los proyectos públicos de los demás. Por lo que es una manera estupenda de que la comunidad pueda colaborar en grandes proyectos aportando sus propios plugins y soluciones de bugs.

4.2.2 Trello

Trello es una plataforma online que permite la creación de tableros virtuales en los que crear tus propios proyectos y asignar tareas y subtareas a cada uno. Es especialmente útil cuando se trabaja en grupo ya que cada uno puede ver en lo que está trabajando el otro y añadir tareas o fallos tanto en su propia tarjeta como en las tarjetas de los demás. Sería el equivalente virtual a las pizarras con pósit.

En mi caso, Trello ha sido utilizado para tener en cualquier momento a la vista las tareas realizadas y por realizar. Además Trello permite agregar comentarios en las tareas por lo que cualquier problema surgido o información buscada sobre la tarea puede ser comentado en la misma tarjeta para tener dicha información accesible de manera rápida y ordenada.

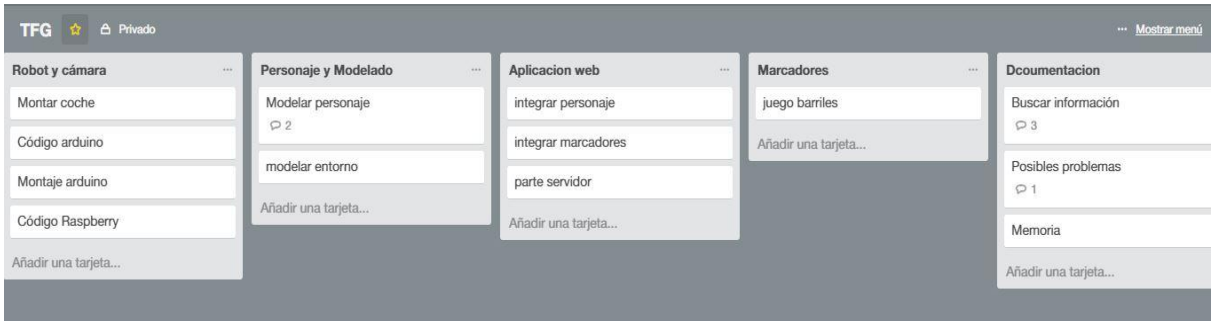

Figura 6 - Vista del tablero en Trello.
Fuente: Propia.

4.2.3 Toggle

Toggle es una herramienta de cronometro que dispone de una página web donde gestionar todos los tiempos recogidos según proyectos y tareas. Además permite crear equipos y gestionar los tiempos de cada uno en un mismo proyecto. Dispone de una extensión para google Chrome que facilita su uso, además de estar integrado en Trello, por lo que puedes iniciar el contador del tiempo desde el propio tablero del Trello simplemente dándole a *start* y Toggle añade automáticamente una nueva tarea con el mismo título que en Trello y comienza a contar el tiempo empleado hasta que se pause el cronometro.

Permite ver el tiempo que se ha empleado en cada tarea con un filtro de rango en el que se puede elegir el periodo que se desea ver, diario, semanal, mensual o personalizado.

Pese a que no es necesario dar cuentas del tiempo empleado en el trabajo de fin de grado me pareció curioso llevar un contador para poder organizarme mejor y saber el tiempo empleado en cada una de las tareas.

5 Desarrollo

En esta parte del documento se explicará el desarrollo del trabajo desde principio a fin hasta lograr la realización del proyecto. El desarrollo se ha dividido en cuatro partes bien diferenciadas para facilitar su implementación y posterior explicación.

5.1 Desarrollo del personaje

Para crear el personaje se han seguido los pasos básicos en cualquier creación de contenido digital visual: bocetos e idea, modelado, texturizado y por último animación.

Esta parte del desarrollo del trabajo ha sido de las que más tiempo ha llevado ya que el proceso de modelar un cuerpo orgánico es lento y meticuloso. Ya que el prototipo de videojuego desarrollado no tiene historia propiamente dicha podríamos pensar que el personaje principal no es importante en el desarrollo y se podría haber simplificado su creación. No obstante me parecía importante conseguir un personaje con una calidad mínimamente buena para que la sensación de estar controlando un videojuego por un entorno real no se viese mermada por la insuficiencia gráfica del personaje.

5.1.1 Bocetos

Para la creación de cualquier personaje es necesaria una serie de bocetos que guíen en el proceso de modelado qué es lo que se quiere conseguir. Estos bocetos no son solo imágenes sino también personalidad y actitud del personaje a modelar, ya que eso influirá en el posterior modelado y animado.

En mi caso la creación de bocetos se ha resumido en una búsqueda de imágenes y referencias teniendo en cuenta el perfil de personaje al que quería llegar. Este perfil es el siguiente: Mujer, joven, aventurera o exploradora y con ropa no sexualizada. Este perfil se escogió ya que aunque el juego carezca de una historia principal, el trasfondo es el de una exploradora que tendrá que superar pruebas en diferentes entornos. Este trasfondo se conseguirá gracias a los minijuegos en realidad aumentada.

Finalmente los bocetos seguidos fueron los siguientes:

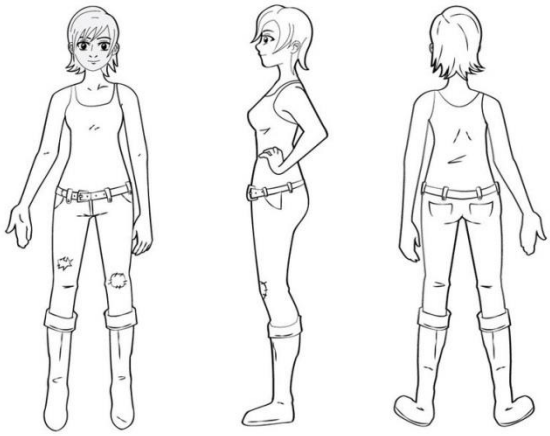

Figura 7 - Primer boceto, chica.
Fuente: Deviantart de zerocrack21 (http://zerocrack21.deviantart.com/art/Model-Sheet-Girl-361628106

Figura 7 - Segundo boceto, Sintel.
Fuente: Deviantart de noahsummers (http://noahsummers.deviantart.com/art/Sintel-Lowpoly-Model-324398958)

Figura 9 - Tercer boceto, exploradora.
Fuente: http://dibujando.net/dib/diseno-personaje-chica-exploradora-53828

Cabe mencionar que dichos bocetos no se copiaron tal cual, si no que sirvieron de referencia para la creación del modelo final, por lo que el personaje tiene un poco de todos ellos.

5.1.2 Modelado

En el proceso de modelado se siguieron las pautas establecidas en el punto sobre los objetivos para que el proceso fuese lo más sencillo y a la vez eficaz posible. Por lo tanto se tuvo en cuenta mantener la organización de los vértices, no romper los *loop* de vértices y llevar especial cuidado en las articulaciones para facilitar la animación. Todo el proceso se llevó a cabo en Blender.

Para llevar a cabo el modelado primero se hizo el cuerpo y luego la cabeza, y finalmente se unieron en una misma malla.

Para el proceso de modelar el cuerpo se usaron los bocetos como fondo para tener una guía. Se empezó con un rectángulo simple a base de torso. Primero se eliminó la parte izquierda del rectángulo y se usó el modificador de espejo para que al editar los vértices de la parte derecha, los de la izquierda se editaran de igual manera y por lo tanto conseguir modelar solo un lado del cuerpo mientras que el otro era automáticamente construido por simetría. Al rectángulo inicial se le fueron añadiendo vértices y mediante extrusión se crearon las piernas y el resto del cuerpo. El resultado una vez modelado fue el siguiente.

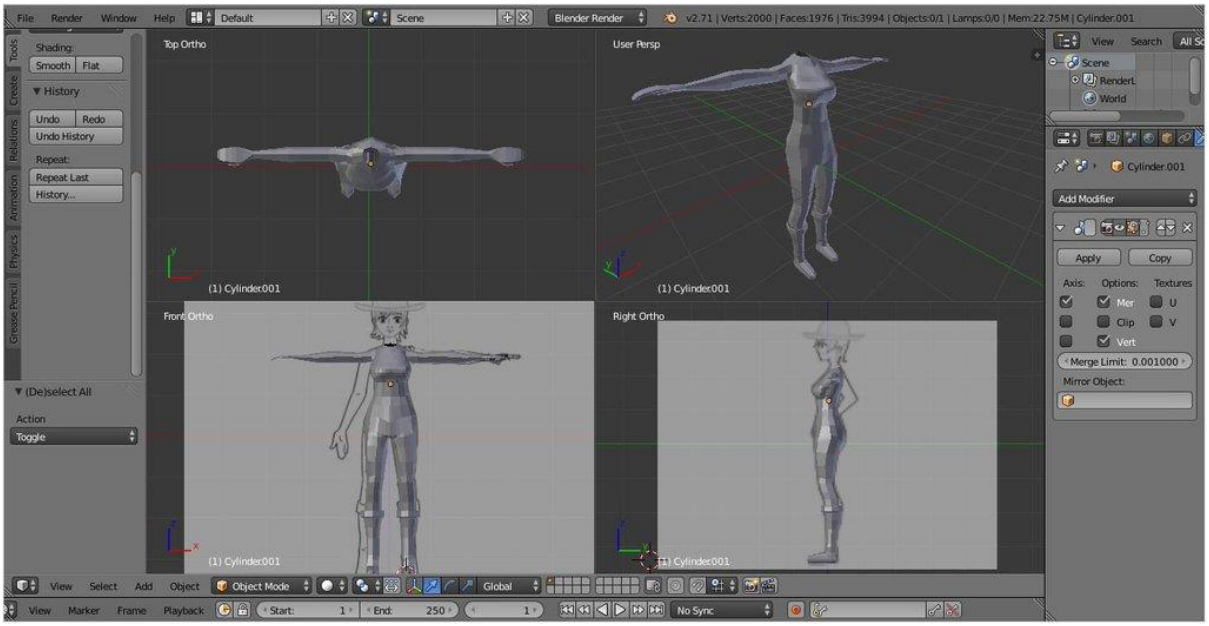

Figura 8 - Cuerpo modelado.
Fuente: Propia.

El siguiente paso fue modelar la cabeza. Para ello se empleó el mismo modificador de espejo para conseguir simetría pero esta vez se comenzó mediante vértices sueltos puestos a mano siguiendo la línea de la nariz. Desde ahí se siguió hacia el mentón y la boca y luego los laterales de la cara junto con los ojos. Por último se crearon los laterales de la cabeza y la parte de arriba. Fueron los ojos y la boca lo que más complicaciones supusieron ya que rompen con la organización normal de los vértices y obligan a añadir vértices adicionales para completar las zonas con más detalle. Finalmente se añadieron las orejas y el cuello y se terminó la parte de la cabeza. Faltaría unirlo con el resto del cuerpo y añadirle el pelo.

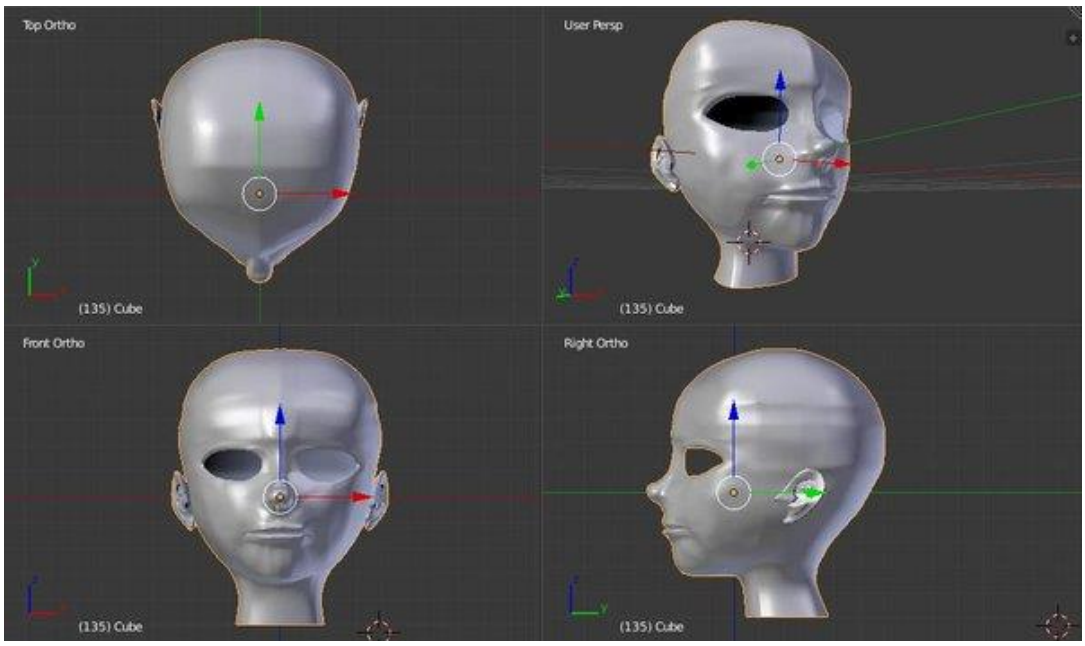

Figura 9 - Cabeza terminada.
Fuente: Propia.

Una vez unidas las mallas de la cabeza y el cuerpo se trabajó en realizar la ropa. Para ello se seleccionaron las caras sobre las cuales iba a estar la ropa, el torso para el caso de la camiseta y las piernas y cintura para el caso del pantalón. Las caras seleccionadas se separaron de la malla y se extruyeron para darles más volumen. Después se ajustaron al cuerpo en forma de la ropa que se quería conseguir y se agregaron detalles como el cinturón. Para que no hubiese problemas a la hora de animar se aplicó el modificador de máscara para ocultar las partes del cuerpo que estuviesen bajo la ropa. De esta manera al moverse la malla, los vértices del cuerpo que sobrepasasen la ropa no se verían.

Con la ropa creada se pasó al pelo. Se intentó modelar varias veces un cabello que quedase bien, pero resulta ser una de las partes más difíciles en el modelado, y más el cabello de mujer. Finalmente se optó por recurrir a modelos de terceros con licencia Creative Commons.

Con el cabello añadido se separaron los vértices en grupos y se colorearon para que fuese más fácil su identificación y la texturización.

Figura 10 - Modelo con pelo y ropa.
Fuente: Propia.

El último paso fue el texturizado. Añadir texturas de forma correcta puede dar una sensación de realismo enorme, aunque el modelado no esté del todo pulido. Para texturizar se usó el método UV Mapping que crea una imagen 2D con toda la malla la cual tiene referencias al modelo 3D, por lo tanto lo que se coloque encima del 2D se traducirá a la posición real del vértice en 3D. Primero hay que colocar la malla 3D en un plano 2D, este proceso se denomina *unwrap* y el resultado es parecido a aplastar la malla contra un plano. Para que el *unwrap* se haga de manera correcta hay que indicarle a Blender los puntos de corte, ya que resulta imposible aplastar una esfera en un plano de manera que se vean todas sus caras sin hacerle un corte a la espera.

Por lo tanto el primer paso llevado a cabo en el texturizado fue marcar las costuras en los puntos adecuados teniendo en cuenta que depende de lo bien que salga el *unwrap* será más sencillo o trabajoso texturizar después. Una vez con las costuras marcadas se procede al *unwrap*.

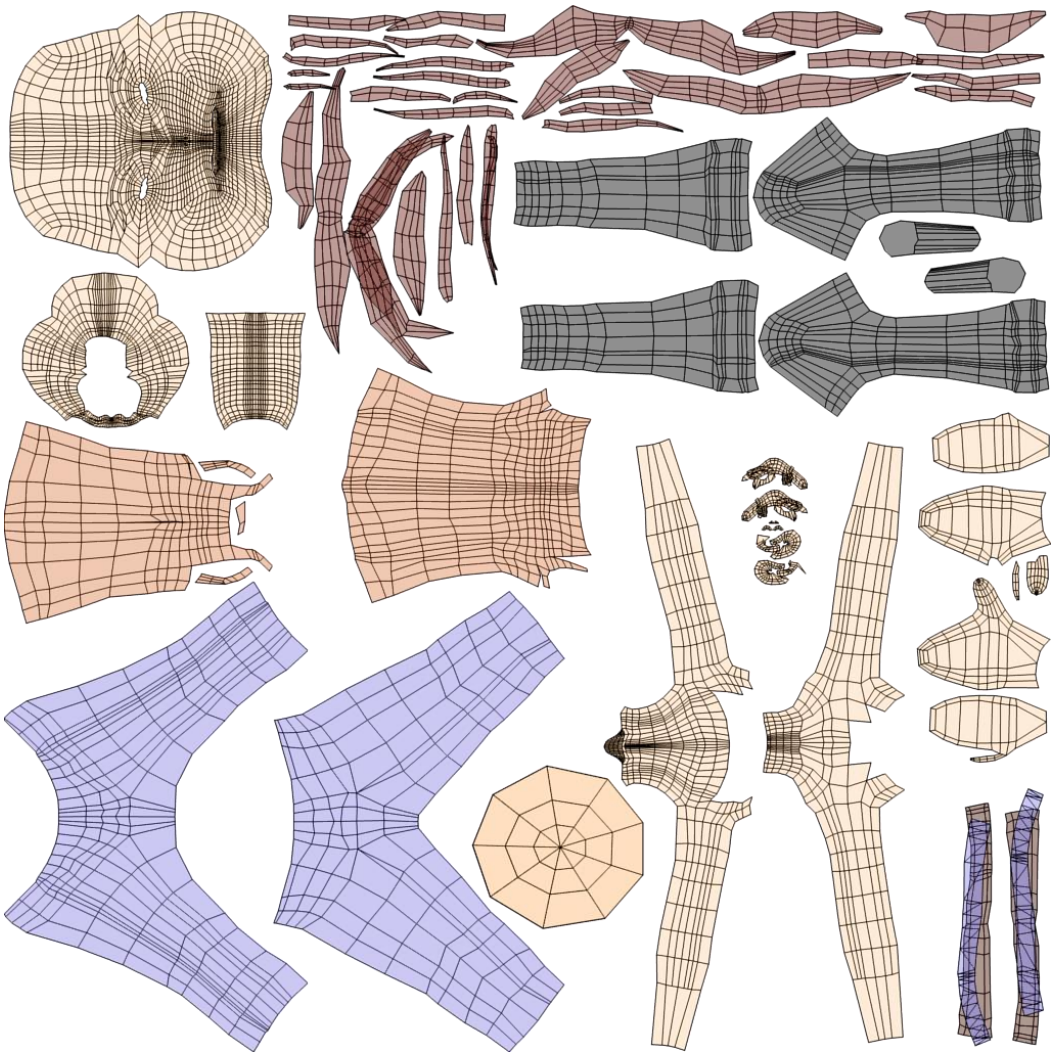

Figura 11 - Unwrap de la malla sin texturizar.
Fuente: Propia.

Con el *unwrap* ya hecho el próximo paso es colocar las texturas encima con un editor de imágenes. Después de eso simplemente hay que asignarle la textura al material del personaje y automáticamente los vértices en 3D se colorearán igual que los del plano en 2D.

Figura 12 - Unwrap texturizado.
Fuente: Propia.

Figura 13 - Modelo texturizado.
Fuente: Propia.

5.1.3 Animación

Para animar nuestro modelo lo primero que necesitamos es un esqueleto asociado con el modelo que nos permita deformar la malla moviendo los huesos del esqueleto. También son necesarios los archivos de animaciones a utilizar.

Para este proyecto se ha utilizado el siguiente banco de animaciones. De dicho banco se han utilizado las animaciones: 127_07 para correr, 127_26 para salto, 132_22 para andar, 137_28 para esperar. Al ser extraídas del mismo banco de animaciones tenemos la ventaja de que todas las animaciones tienen los mismos huesos asociados por lo que será fácil cambiar de una a otra.

Estas animaciones tienen una pose en T que será la utilizada para referenciar el modelo al esqueleto. Simplemente hay que ajustar el tamaño del esqueleto al del modelo y hacer que los huesos coincidan con la parte del modelo a la que van a controlar. Una vez ajustado hay que referenciarlos.

Figura 14 - Esqueleto referenciado al modelo.
Fuente: Propia.

Esto hará que cada hueso tenga un peso determinado sobre cada parte de la malla. Ahora cada vez que movamos un hueso la malla se deformará con él, la cantidad de movimiento de la malla depende de la cantidad de peso del hueso sobre la malla

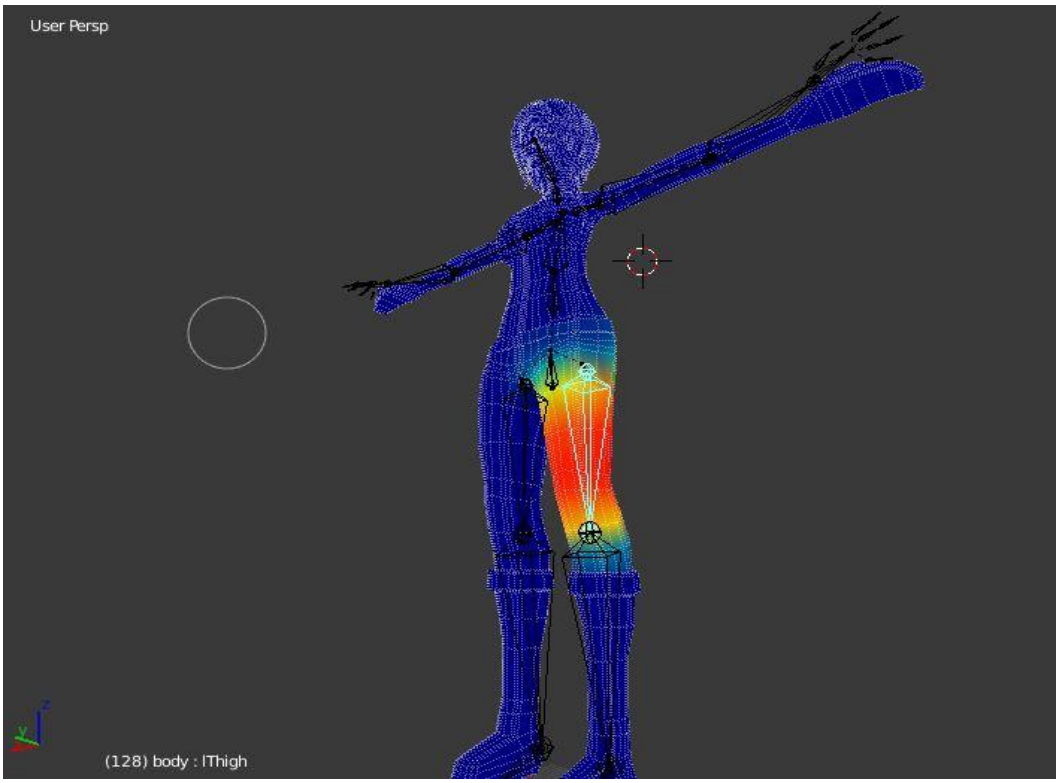

Figura 15 - Peso del hueso del fémur sobre la malla.
Fuente: Propia.

Una vez que el esqueleto esté referenciado al modelo el siguiente paso es preparar las animaciones para su uso. Las animaciones tienen principio y final, esto hace que la animación de correr empiece en parado, por lo que si se reproduce en bucle en una carrera larga cada 3 pasos habría un salto a la posición de parado. Por lo tanto hay que recortarlas para conseguir que el principio coincida con el final. Además cada animación está orientada hacia una posición y en las que son de movimiento el esqueleto avanza junto con el movimiento, hay que conseguir que haga la animación de correr pero sin moverse de la posición en el plano.

Para la edición de los archivos .bvh se ha usado el programa bvhacker, un editor gratuito de archivos bvh, con el que se puede editar el movimiento de los huesos para rotarlos o conseguir que no se desplacen en ningún eje. Además dispone de una herramienta muy potente para modificar el *loop* de la animación, que permite

seleccionar el *frame* inicial y el final y coserlos de manera que la posición de los huesos coincida en ambos *frames*.

Con las animaciones preparadas lo que queda por hacer es importar las

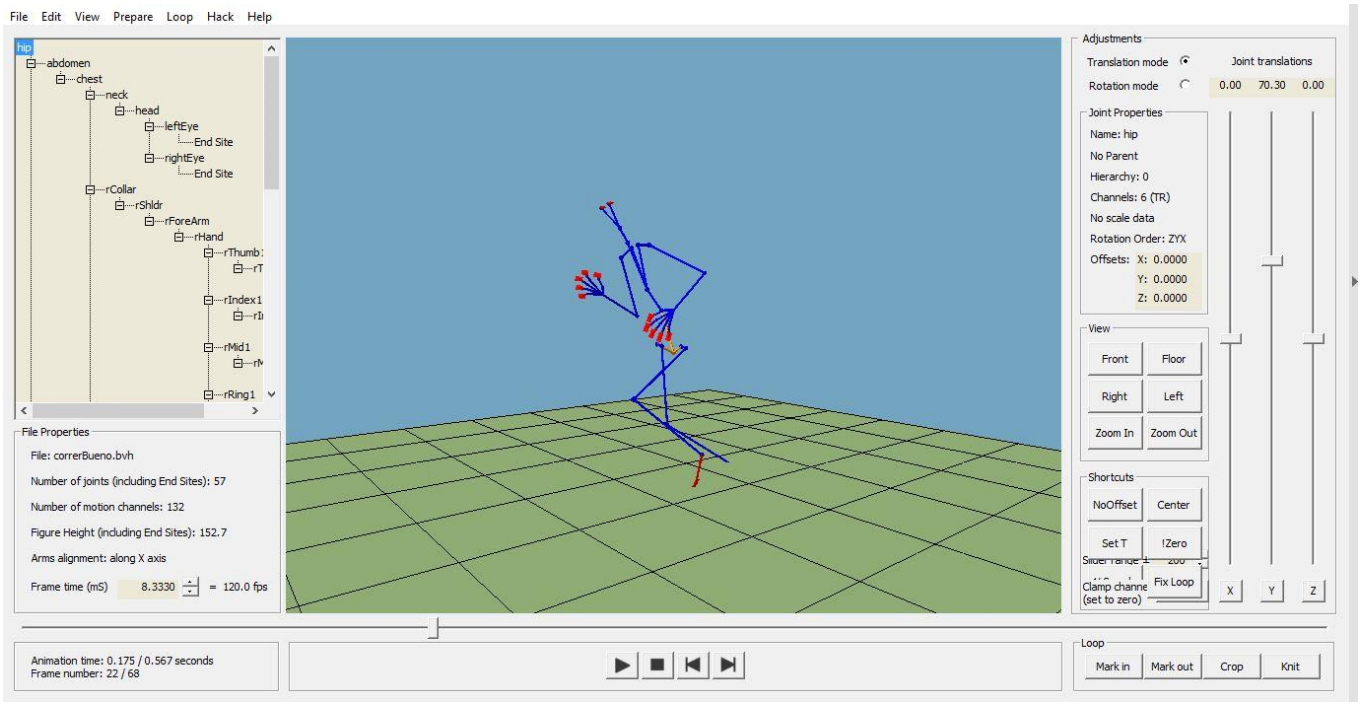

Figura 16 - Interfaz de bvhacker.
Fuente: Propia.

animaciones en Blender y hacer que el modelo las siga. Para ello es muy importante que todas las animaciones tengan los mismos huesos para que el proceso sea sencillo. Simplemente hay que importar una animación y al ser del mismo tipo que nuestro esqueleto referenciado en T simplemente tendremos que seleccionar nuestro esqueleto, abrir la biblioteca de animaciones y elegir la que queremos. De esta manera nuestro esqueleto adoptará la forma de la animación recién importada.

Figura 17 - El modelo (derecha) en posición de la animación importada (izquierda).
Fuente: Propia.

El último paso en el desarrollo del personaje es exportarlo de tal manera que podamos leerlo con Three.js para incluirlo en nuestra aplicación web. Para esto la librería de Three.js incluye un exportador de Blender a JSON. En el momento de exportarlo la versión de Three.js era la r77 y el exportador aún estaba en desarrollo por lo que hubo ciertos problemas al principio. Una vez exportado y cargado en la aplicación el esqueleto se cargaba correctamente pero la malla se deformaba. Finalmente este problema se arregló modificando el código fuente del exportador cambiando una línea de código en donde las coordenadas Y de la malla se exportaban como coordenadas Z.

5.2 Desarrollo de la aplicación web

En este proyecto la aplicación web es el núcleo que une todas las demás partes. Mientras que en el lado del cliente se usa el personaje desarrollado y se ofrece al usuario una interfaz con la que jugar, el lado del servidor se comunica con el robot y envía las órdenes para controlarlo. A continuación se verá cómo se han ido desarrollando cada una de las partes de la aplicación web y se entrará en detalle cómo funciona cada una y las tareas que realiza.

5.2.1 Cliente

Para la integración del personaje modelado y el uso del WebGL se ha utilizado Three.js. Three.js es una potente librería que facilita el uso de WebGL y todo lo que se refiere a gráficos en el navegador. Se basa en el uso de un canvas con un contexto en 3D para mostrar toda la información.

5.2.1.1 Carga del personaje

Cargar el personaje con sus animaciones y controlar las animaciones al gusto fue una tarea relativamente sencilla gracias a los archivos de ejemplo de los que dispone Three.js. En especial el ejemplo "*webgl_animation_skinning_blending*" que ya carga un modelo 3D en JSON y permite jugar con sus animaciones cambiando de una a otra de manera fluida. Por lo tanto partiendo de este ejemplo hubo que cambiar el modelo cargado por el nuestro y se estableció unos controles para manejar las animaciones con el teclado y mediante pesos.

Además de cargar el personaje se crea un plano a nivel del suelo para reflejar las sombras del personaje. El mostrar las sombras, aunque sean muy diáfanas, da una sensación de realidad que de otra manera no se podría conseguir, además de que nos ayuda a situar la posición en el eje Y del personaje respecto al suelo.

La función *animar* es la que se encarga de pintar al personaje en el estado actual, dicha función es llamada 60 veces por segundo gracias al método "requestAnimationFrame" que se encarga de automatizar estas llamadas siempre y cuando la ventana esté activa, por lo tanto la aplicación no consumirá recursos cuando no se encuentre en el primer plano del navegador. La función *animar* llama a su vez a la función *render* que es la encargada de limpiar el canvas y volver a pintar la escena actualizada en él en cada pasada.

5.2.1.2 Controles

La clase que el ejemplo usa para cargar el modelo 3D se llama "BlendCharacter.js". En el proyecto se ha renombrado esta clase y se ha ampliado añadiendo propiedades y métodos necesarios para controlar al personaje a nuestro gusto. En particular se han establecido una serie de estados en los que puede estar el personaje como moviéndose hacia adelante, saltando o parado. Además se usa un *array* con el peso que cada

animación tiene en cada momento. Cuando se pulsa la tecla de avanzar por ejemplo cambia el estado de parado a avanzando. En el estado avanzando la animación de correr empieza a adquirir peso mientras que la animación de parado lo pierde gradualmente, esto ocurre hasta que la animación de correr tiene el máximo peso. Si se libera el botón de avanzar vuelven a cambiar los estados con el cambio de animación pertinente.

Además el control de los estados en cada momento permite usarlos y comprobar si realizar o no ciertas acciones según el estado actual. De este modo si el estado activo es avanzando se podrá mover el personaje en el eje horizontal para dar la impresión de movimiento lateral, mientras que si el estado es parado este movimiento no se permite.

De igual manera cada vez que un estado está activo se llama a la API para controlar al robot. Esta API se verá mejor en la parte del servidor.

Para permitir el uso de la aplicación en móviles se ha habilitado unos controles adicionales para dispositivos con pantalla pequeña. Estos controles se basan en botones sobre el fondo que al activarlos mandan la misma señal que se mandaría al pulsar una tecla, por lo que la ejecución funciona exactamente igual.

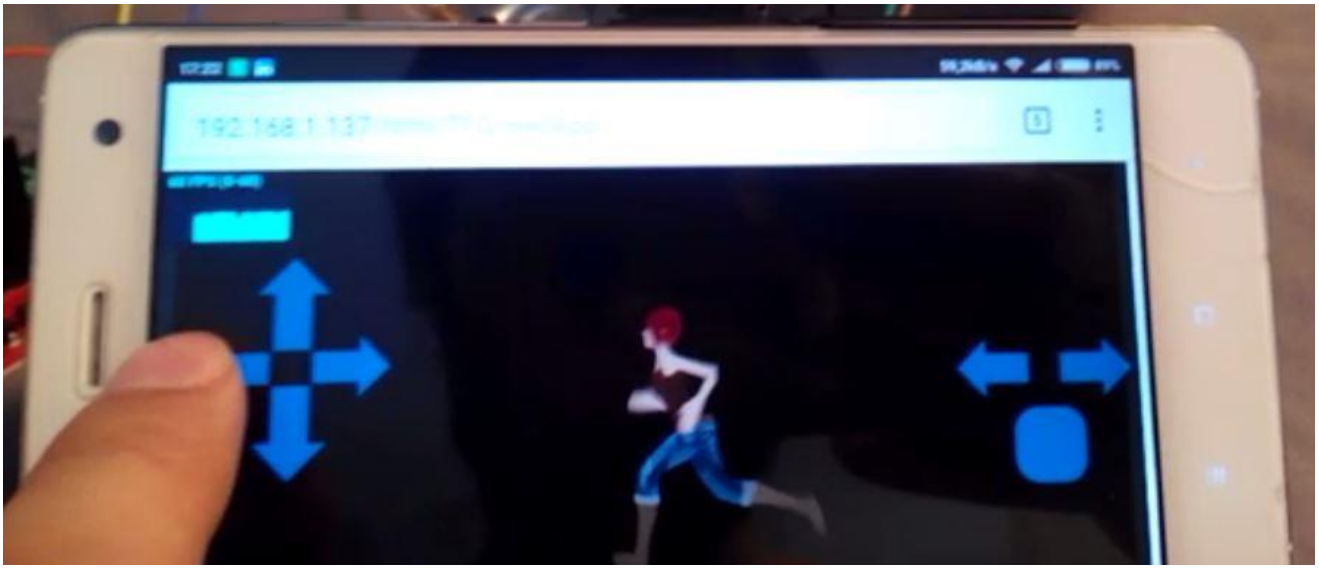

Figura 18 - Controles para dispositivos móviles.
Fuente: Propia.

5.2.1.3 Cámara y fondo

Para dar la sensación de que nuestro personaje se encuentra en el mundo real, la aplicación rellena el fondo de la misma con imágenes en tiempo real captadas por la cámara del robot. El método para tomar estas imágenes y gestionarlas se verá más adelante.

El canvas cuando usa un contexto en 2D te permite usar la propiedad *background* para establecer una imagen o video de fondo. Sin embargo cuando se usa un contexto 3D esta propiedad está desactivada, por lo que para aplicar un fondo hay que usar otros métodos.

En este proyecto se ha usado un plano vertical, dicho plano se ha colocado de tal manera que ocupe todo el canvas para que no queden huecos vacíos. A este plano se le asigna una textura con la imagen tomada desde la cámara. De esta manera cada pasada de *render* se llama a la función que cambia esta imagen por la siguiente tomada. Las imágenes se van sustituyendo a medida que son tomadas por el script raspiMJPEG. La función encargada de cambiar las imágenes realiza petición POST a PHP que abre el archivo de la nueva fotografía. Aunque la función es llamada cada pasada de *render*, tiene un manejador para solo realizar la petición cada 4 pasadas del *render*, es decir que la imagen se actualizará 60/4 = 15 veces por segundo. Este manejador se implementó ya que realizar la petición al servidor de cambio de imagen 60 veces por segundo resultaba muy costoso y hacia que a mayoría de peticiones se perdiesen.

La carga de imágenes estáticas 15 veces por segundo crea la sensación de video y por lo tanto es suficiente para nuestro objetivo.

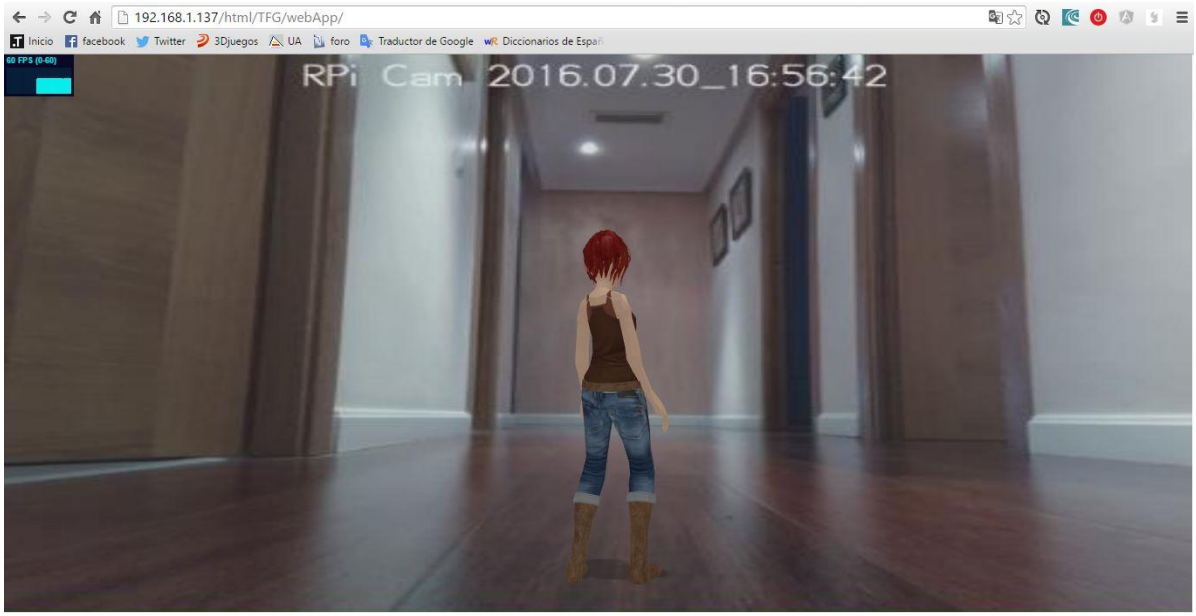

Figura 19 - Aplicación web con la cámara funcionando.
Fuente: Propia.

5.2.2 Servidor

En la parte del servidor se organiza toda la comunicación con el robot. Como se ha mencionado antes, cuando un estado está activo se llama a la API del robot para hacer que éste responda a las órdenes. Las llamadas se hacen con cada paso del *render* pero la petición al servidor para controlar el robot solo se envía si el robot no está ejecutando ya la orden. De esta manera al controlar las llamadas se libera el servidor de carga con todas las peticiones irrelevantes que se pudiesen hacer.

La API del control del robot está desarrollada en PHP y se la llama mediante peticiones AJAX POST. Estas peticiones se ejecutan en JavaScript y al ser asíncronas permiten a la aplicación seguir su funcionamiento sin tener que esperar una respuesta del robot. Por lo tanto si una llamada falla no se paralizará la ejecución si no que al no haberse ejecutado el estado del robot no ha cambiado por lo que a la siguiente pasada de *render* se la volverá a llamar hasta que se ejecute correctamente y el estado del robot cambie. En la petición POST se pasa un comando en un *String*, la API lee el comando y ejecuta a un script en Python con los argumentos adecuados en forma de caracteres según el comando recibido. Este script se comunica con el arduino mediante el puerto serial y le envía las órdenes.

Por lo tanto la comunicación completa para el paso de órdenes es la siguiente: AJAX -> PHP -> Python -> Arduino.

Las órdenes dadas por usuario no son las únicas llamadas que el servidor controla. Además para usar el sensor de distancia del arduino, cada pasada del render se llama a una función que envía una petición al servidor para comprobar la distancia. De nuevo mediante un contador esta función solamente se ejecuta cuando se ha llamado 120 veces, por lo que se comprobará la distancia cada dos segundos. Dicha función tiene un parámetro para forzar la ejecución, cuando el parámetro esté a *true* la función hará la petición al servidor sin importar la última vez que la hizo, y el contador volverá a cero.

Cuando se envía la petición para comprobar la distancia se recibe una respuesta de parte del arduino que pasa por Python y se devuelve al PHP y finalmente al código JavaScript. Si la distancia resulta haberse leído mal se fuerza otra llamada inmediatamente y se repite el proceso, si la distancia es menor que un número determinado, 27 centímetros actualmente, se bloquea la opción de avanzar tanto al robot como al personaje, por lo que lo único que podría hacer es rotar hacia los lados para sortear el obstáculo de delante.

La API del robot maneja los siguientes comandos:

Orden recibida	**Argumento enviado al arduino**	**Acción realizada por el robot**	**Opción disponible**
avanzar	W	Enciende los dos motores hacia delante.	SI
parar	X	Apaga los motores y se para.	SI
girarIz	Q	Enciende el motor derecho hacia delante y el izquierdo hacia atrás.	SI
girarDe	E	Enciende el motor izquierdo hacia delante y el derecho hacia atrás.	SI
retroceder	S	Enciende los dos motores hacia atrás.	NO
distancia	M	Llama al sensor de distancia y espera la respuesta en forma de *String*.	SI

5.3 Desarrollo del robot

El desarrollo del robot ha sido la parte más técnica y a la vez desconocida para mí. Aunque ya había trabajado anteriormente con arduino nunca había sido a este nivel y con un proyecto tan serio.

El montaje del robot podría separarse en dos partes muy diferenciadas, la Raspberry Pi y el Arduino. Se podría decir que la Raspberry Pi funciona como el cerebro del robot mientras que el arduino son los músculos que lo mueven.

El robot se ha montado en un sencillo chasis al que se le ha añadido un segundo nivel para poder distribuir los componentes más fácilmente. En el nivel de arriba se encuentran el Arduino y la Raspberry Pi, mientras que en el nivel de abajo está el controlador de motores, los motores, la cámara, la batería, las pilas y el sensor de distancia.

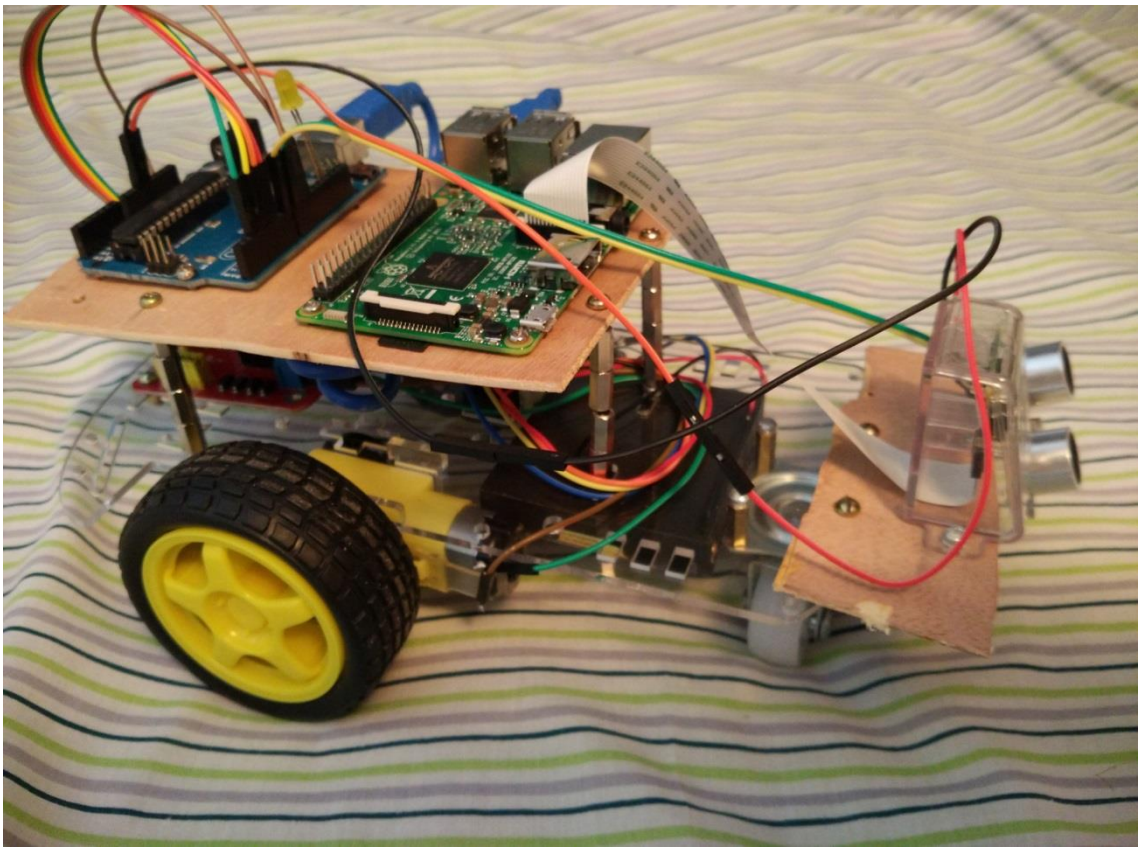

Figura 20 - Vista lateral del robot.
Fuente: Propia.

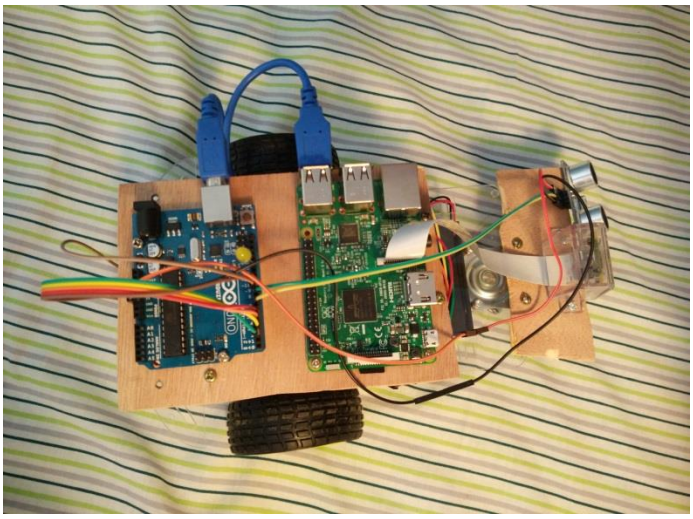

Figura 21 - Vista superior del robot, (Arduino a la izquierda y Raspberry Pi a la derecha).
Fuente: Propia.

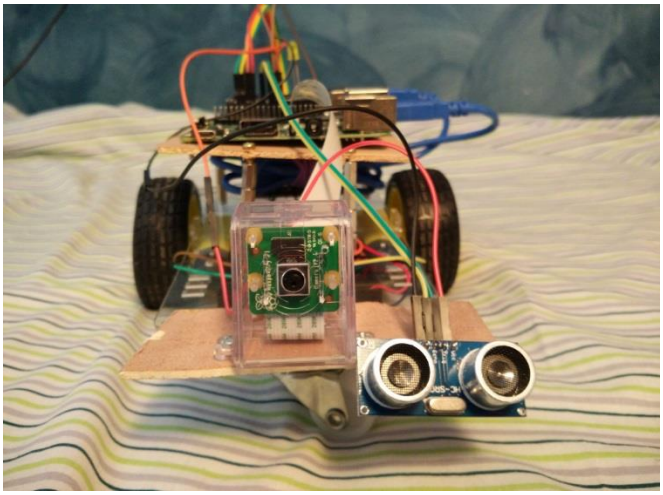

Figura 22 - Vista frontal del robot.
Fuente: Propia.

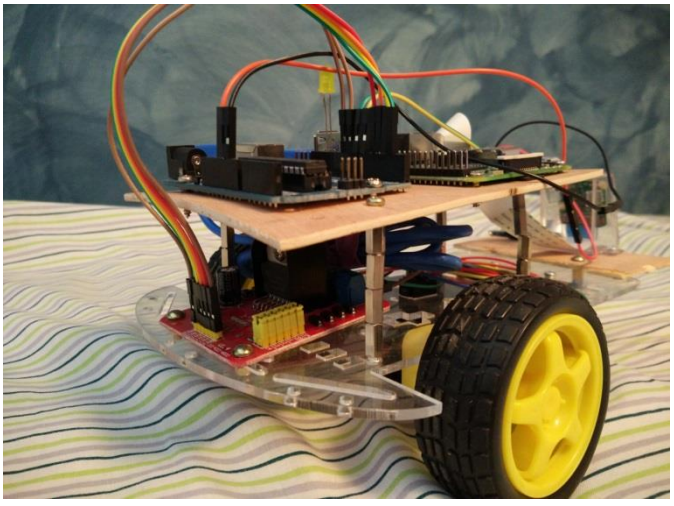

Figura 23 - Vista trasera del robot.
Fuente: Propia

El robot tiene dos ruedas traseras conectadas a motores individuales y una rueda "loca" delantera que ayuda en los giros y la estabilidad.

Se usan dos fuentes de alimentación distintas en el robot. La primera es una batería externa que se sitúa entre la cámara y el segundo piso. Esta fuente alimenta la Raspberry Pi que a su vez alimenta al Arduino. La segunda es una base para 4 pilas AA situada en la parte inferior del robot, en la cara de abajo. Esta segunda fuente alimenta la placa de los motores así como los mismos motores, si se queda sin pilas los motores no tendrán suficiente fuerza para avanzar, pero la aplicación web seguirá funcionando.

5.3.1 Raspberry pi

Como se ha comentado anteriormente, la Raspberry Pi es el cerebro del robot. Es la encargada de capturar las imágenes de la cámara, de controlar el arduino y es a la vez el servidor web que hospeda la aplicación.

Para preparar la Raspberry Pi se tuvieron que hacer una serie de pasos y configuraciones iniciales que se van a explicar a continuación:

- Para controlar la Raspberry Pi desde el ordenador sin tener que usar un ratón, teclado y pantalla supletoria se instaló un servidor VNC[3] en la Raspberry Pi que se ejecuta nada más enchufarla gracias a un script. El servidor VNC nos permite ver y controlar el escritorio de la Raspberry Pi desde otro ordenador usando simplemente el cliente VNC.
- Se instaló el servidor web NginX con PHP[4]
- Se asignó una IP privada estática a la Raspberry Pi para que la conexión al servidor VNC como al servidor web fuese siempre en 192.168.1.137.
- Se instaló el IDE de arduino y se configuró para usar el puerto ttyACM0 de la Raspberry Pi a una velocidad de 9600 baudios.
- Se dieron permisos al usuario www-data [5]de ejecución del directorio *index* donde se encuentran los archivos web. Además se agregó el usuario www-

[3] Se siguieron las siguientes instrucciones para la instalación del servidor VNC.
[4] Se siguieron las siguientes instrucciones para la instalación del servidor web.
[5] Es el usuario que usa el servidor web NginX

data al grupo "dialout[6]" para poder controlar los puertos seriales desde el servidor web. Es importante reiniciar la Raspberry Pi después de cambiar los permisos.

Una vez que la Raspberry Pi estuviese preparada se hospedó la aplicación web en ella y se pudo comenzar a hacer las pruebas de comunicación entre el Arduino y el servidor.

Para el uso de la cámara se usó la aplicación raspiMJPEG. Esta aplicación permite configurar los parámetros de la cámara y tomar fotografías cada *x* tiempo. Hay que tener en cuenta que la cámara no capturaba un video y después lo enviaba a la aplicación web, si no que se dedica a capturar imágenes cada pocos milisegundos y guardarlas en la misma carpeta de tal manera que cada nueva imagen sobrescriba a la anterior. Entonces cada vez que la aplicación web carga la imagen con el mismo nombre resulta que la imagen cargada es diferente a la anterior, lo que provoca una sensación de vídeo. Para que este proceso de guardado y apertura de archivos pueda realizarse a una velocidad adecuada sin perjudicar la velocidad de ejecución lo que se hace es guardar las imágenes en el directorio "/dev/shm/" de la Raspberry Pi, lo que equivale a guardarlas en memoria *RAM* en vez de en la tarjeta sd y por lo tanto permite su uso de manera más veloz.

5.3.2 Arduino

Arduino conforma la parte más electrónica del proyecto, es el encargado de dar las órdenes a los motores y de manejar el sensor de distancia.

El código de Arduino se basa en una función inicial llamada *setup* donde se inicializan todas las variables y estados de los pines de Arduino y una función *loop* que se ejecuta continuamente. Esta función *loop* lo que hace es mantenerse a la escucha en el puerto serial y en el momento en el que llegue algún dato comprueba a qué orden de entrada se refiere y hace al arduino actuar de una manera o de otra arrancando los motores o leyendo la distancia por el sensor.

[6] Es el grupo de usuarios que tiene permisos en el uso de los puertos de la Raspberry Pi.

Las conexiones del Arduino se explicarán en el apartado de cada sensor más detalladamente. El circuito completo puede verse a continuación:

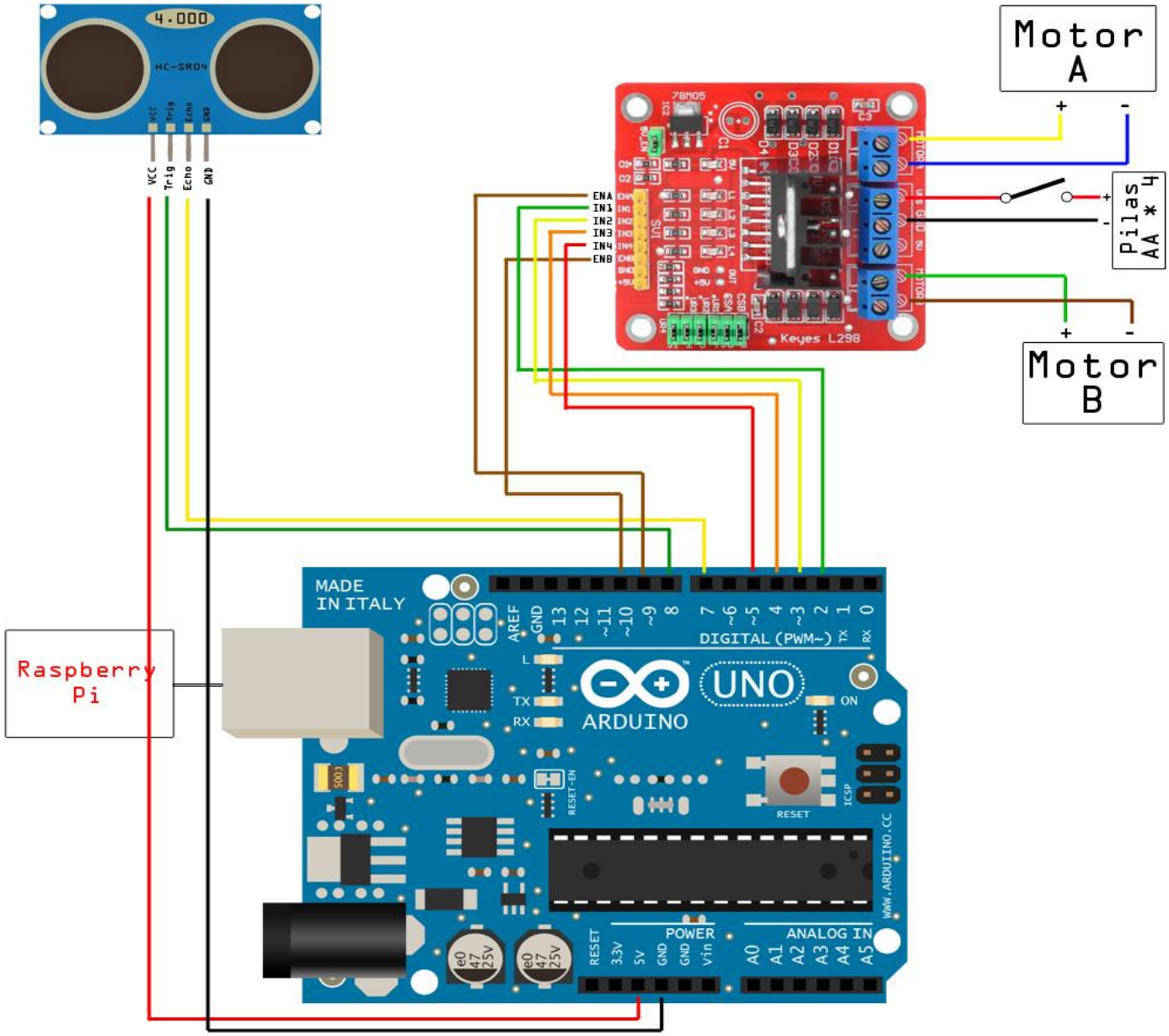

Figura 24 - Circuito completo del arduino y sus sensores.
Fuente: Propia.

5.3.2.1 *Sensor de distancia*

La conexión del sensor de distancia al arduino fue sencilla. Para comenzar simplemente hay que conectar el pin de 5V del Arduino al pin VCC de alimentación del sensor y el pin de GND del Arduino al pin GND del sensor. Por lo tanto el sensor es alimentado por el Arduino. El pin TRIG del sensor va conectado al pin número 8 del Arduino. Este pin se pone en modo *output* y mandará una señal analógica al sensor cada vez que se quiera medir la distancia y lo pondrá en funcionamiento enviando una señal de ultrasonido. El pin ECHO del sensor se conecta al pin 7 del Arduino, iniciado en modo *input* el pin 7 leerá el tiempo que ha transcurrido desde que el sensor envió la señal de ultrasonido hasta que la ha recibido.

Cuando se recibe el tiempo transcurrido en milisegundo hay que hacer una conversión para pasarlo a distancia. Sabiendo que la velocidad del sonido en el aire es de 340 *m/s* aproximadamente y que la fórmula de la distancia es $d=v*t$ podemos hacer el siguiente cálculo: 340 *m/s* es lo mismo que 0.034 centímetros cada microsegundo. Teniendo en cuenta que el tiempo corresponde al tiempo de ida y de vuelta hay que dividirlo entre 2, lo que nos queda 0.017. Por lo que hay que multiplicar el tiempo que recibimos por 0.017.

5.3.2.2 Shield *de motores L298n*

La *shield* de motores L298n nos permite controlar tanto la velocidad como la dirección con salida para 2 motores, con la posibilidad de usar 4 motores si se colocan por parejas. Utilizar la *shield* de motores es necesario ya que el Arduino no es capaz de gestionar la intensidad de salida suficiente para alimentarlos.

Nuestra *shield* estará alimentada mediante 4 pilas AA lo que proporcionará cerca de 5 voltios. Suficientes para mover los dos motores. Esta placa además permite alimentar también al arduino pero en nuestro caso no usaremos esta opción ya que el Arduino es alimentado por la Raspberry Pi, y los 5 voltios que nos proporcionan las pilas no son suficientes para gestionar todos los componentes.

Las conexiones de la *shield* son las siguientes: Las 2 salidas de motores están conectadas cada una a un motor, la salida A al motor izquierdo y la B al motor derecho. La entrada VCC está conectada a un interruptor que continua hacia las pilas

mientras que la salida GND conecta directamente con el otro extremo de las pilas. Esta *shield* dispone de dos pines (INA e INB) que son los encargados de activar y desactivar los motores mediante el envió de un pulso digital. Que un motor esté activo no quiere decir que esté en funcionamiento, simplemente que se puede poner en funcionamiento. Estos dos pines también nos permiten enviarles un pulso analógico en vez de digital para controlar la velocidad entre 0 y 250. Los pines IN1 e IN2 son los encargados de controlar el motor A y los IN3 e IN4 los del motor B. Depende cuál de ellos se active mediante pulsos digitales el motor rotará en un sentido o en otro.

Motor A truth table

ENA	IN1	IN2	Description
0	N/A	N/A	Motor A is off
1	0	0	Motor A is stopped (brakes)
1	0	1	Motor A is on and turning backwards
1	1	0	Motor A is on and turning forwards
1	1	1	Motor A is stopped (brakes)

Motor B truth table

ENB	IN3	IN4	Description
0	N/A	N/A	Motor B is off
1	0	0	Motor B is stopped (brakes)
1	0	1	Motor B is on and turning backwards
1	1	0	Motor B is on and turning forwards
1	1	1	Motor B is stopped (brakes)

Figura 25 - Tabla con las posibilidades de los motores según los pines activos.
Fuente: https://www.bananarobotics.com/shop/How-to-use-the-L298N-Dual-H-Bridge-Motor-Driver

A la hora de usar los motores resultó que a velocidad máxima el coche avanzaba y giraba demasiado rápido, por lo que le control de la aplicación era difícil y el video borroso debido a la velocidad de movimiento. Para resolver este problema se varió la velocidad de los motores con el inconveniente de que al rotar más lentamente no eran capaces de poner en movimiento a todo el robot. Finalmente se solucionó arrancando los motores a plena potencia y 125 milisegundos después poniéndolos a una velocidad de 100. De esta manera se consigue poner en movimiento el robot y una vez en movimiento los motores ya son capaces de controlarlo a menor velocidad.

Para realizar giros se optó por el movimiento de ambas ruedas en sentido contrario. De esta manera cuando se desea girar a la derecha la rueda izquierda rota hacia delante mientras que la derecha rota hacia atrás. Esto permite un giro más cerrado y preciso que si simplemente girase una rueda mientras la otra permanece parada.

5.4 Realidad aumentada

En la parte de realidad aumentada se ha desarrollado el prototipo de un minijuego el cual consiste en que cada vez que se detecte un marcador específico se situará una trampilla en dicho marcador que soltará barriles que seguirán al personaje el cual deberá esquivar o saltar dichos barriles.

Para la implementación de la RA se ha usado la librería JSARToolKit la cual permite hacer uso de las tecnologías de RA en navegadores web usando JavaScript, HTML5 y WebGL.

El primer paso fue reconocer el marcador. Para ello se utilizó un marcador proporcionado por la propia librería, en especial el marcador con id 8.

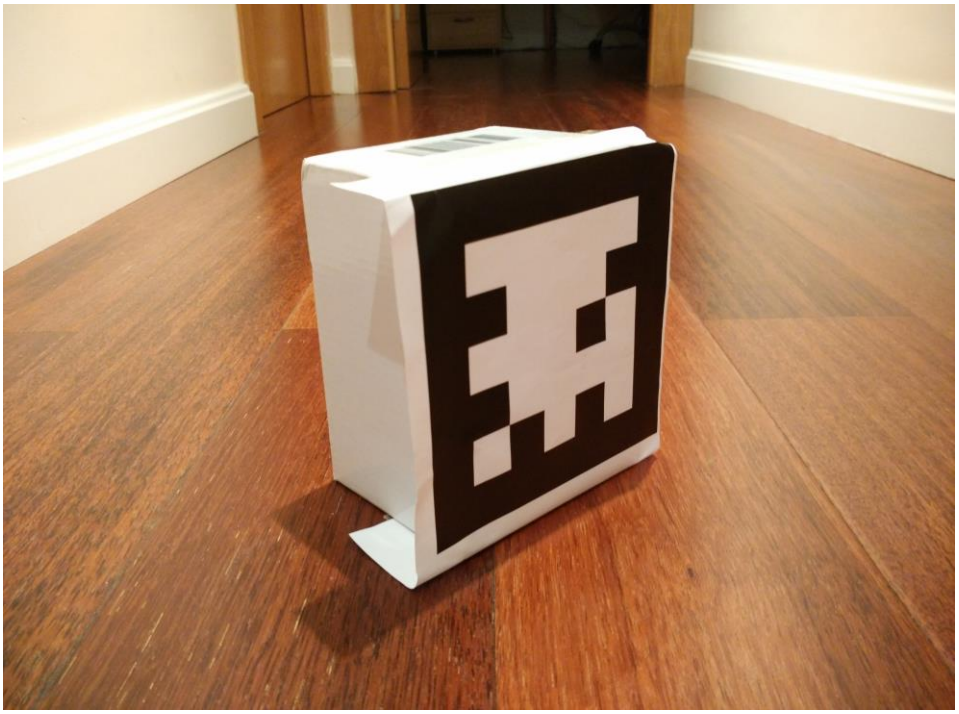

Figura 26 - Marcador con id 8 proporcionado por JSARToolKit.
Fuente: Propia.

Una vez que se consiguió detectar marcadores y obtener su matriz de transformación para posteriormente aplicársela a nuestros modelos se pasó a crear el minijuego en cuestión. Inicialmente, en la fase de programación, se usaron cubos como guía. Las fases del juego son las siguientes:

1) Al cargar la página se carga el juego y se crea la puerta y los barriles. Los barriles se introducen como hijos de la puerta y la puerta a su vez es hija del objeto al que se aplica la matriz de transformación obtenida del marcador.

2) El detector recibe la imagen a analizar en cada *frame* y detecta todos los marcadores de la imagen obteniendo su id y su matriz. Si alguno de los marcadores detectados tiene la id 8 se inicia el juego.

3) Cuando se inicia el juego empieza un contador que hará que cada 2 segundos salga un barril de la compuerta. Cada vez que un barril sale sigue unos ciertos pasos: Se hace rotar localmente el modelo de la puerta para dar la sensación de que se abre; Se cambia el padre del barril de la puerta a un objeto auxiliar que empieza con la misma matriz de transformación que la puerta. Así si la puerta se mueve el barril ya no se moverá con ella al haber sido soltado; El barril comienza a avanzar hacia el personaje a la vez que rota sobre sí mismo. Para avanzar hacia el personaje el movimiento que realiza tanto en el eje X como en el eje Y depende del eje Z. Se calcula la distancia del barril con el personaje en el eje Z y según cuanto aumente cada vez en el eje Z se calcula en cuantas pasadas de render va a llegar al personaje. Sabiendo cuantas pasadas de render se van a dar se calcula la velocidad necesaria tanto en el eje X como en el eje Y para que al final de dichas pasadas el barril esté en las coordenadas deseadas.

4) Cuando el detector ya no encuentre el marcador con id 8 se dará un margen de 1 segundo hasta que éste desaparezca y se reinicie el juego. Esto se hace para contrarrestar posibles demoras en el detectado de marcadores y darle tiempo a hacer más detecciones antes de reiniciar el juego.

5) Al reiniciar el juego se vuelven a colocar los barriles en su posición inicial y se vuelve a cambiar su padre por la puerta.

Figura 28 - Cubos utilizados en la fase de programación.
Fuente: Propia.

Una vez terminada la programación se pasó a la fase de modelar y cargar los objetos utilizados finalmente. Para ello se modeló y texturizó una trampilla y un barril.

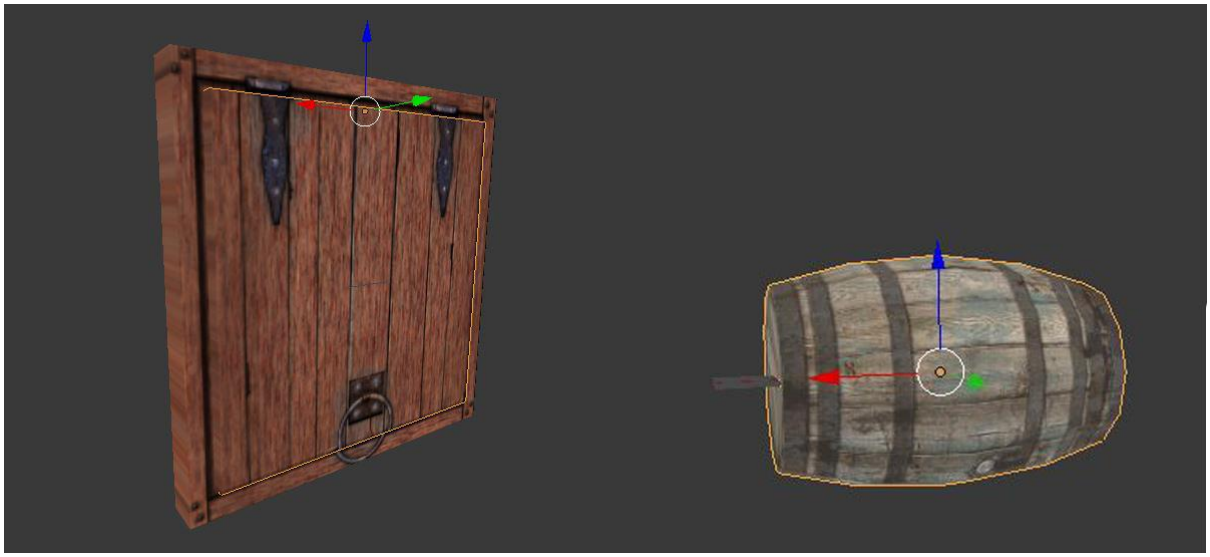

Figura 27 - Puerta y barril usados en el minijuego.
Fuente: Propia.

Figura 29 - Minijuego de los barriles implementado.
Fuente: Propia.

6 Trabajo futuro

Lo realizado en este Trabajo de Fin de Grado no es más que un prototipo de lo que podría llegar a ser una nueva manera de jugar. Por lo tanto se pretende mostrar la idea pero dejando claro que el trabajo entregado no muestra todo el potencial que tendría una versión final ya que se requeriría mucho más tiempo para su realización además de un equipo más amplio.

Echemos entonces una vista al futuro y a lo que se podría realizar con los recursos suficientes.

6.1 Mejoras

Para que el jugador tenga ganas de seguir usando el juego es necesario desarrollar una serie de puntos para engancharlo y hacer que quiera seguir usando el sistema.

6.1.1 Historia

Uno de estos puntos sería el desarrollo de una historia. En el prototipo simplemente se enseña un minijuego para mostrar las capacidades de jugabilidad, sin embargo se podría añadir una historia detrás para formar un juego completo y no a simples minijuegos separados entre sí.

6.1.2 Rejugabilidad

Hay que tener en cuenta que tenemos dos proyectos distintos entre manos. Uno es el robot como plataforma de juego y otro el juego en sí mismo. Por lo tanto sería posible que, usando el mismo robot, el juego fuese diferente y por lo tanto en vez de mostrar por pantalla unos gráficos mostrasen otros diferentes con una finalidad distinta.

Ahora mismo como juego se tiene simplemente la visualización y control de un personaje femenino además de una compuerta colocada mediante un marcador de RA la cual lanza barriles hacia el personaje cuando es detectada. Sin embargo se podría visualizar un coche controlado por nosotros además de los controlados por la IA y hacer una carrea o crear un RPG con eventos aleatorios y gestión de recursos.

6.1.3 Chasis de 4 ruedas

Actualmente el chasis del robot dispone de 2 ruedas traseras motorizadas y una "rueda loca" delantera que ayuda a mantener el equilibrio y girar. Una vez finalizado el proyecto se ha comprobado que esta rueda hace ladearse al robot y no le permite ir recto. Este efecto se acentúa si se disminuye la velocidad de las ruedas traseras ya que no tienen la suficiente potencia para poner la "rueda loca" recta lo que deriva en que cuando se pretende ir recto el robot gira hacia el último lado girado. Esto se podría solucionar sustituyendo el chasis por uno en el que se puedan colocar 4 ruedas dos a cada lado. De esta manera se podría disminuir la velocidad del robot todo lo deseado para permitir un mayor control en el momento de los juegos sin ningún tipo de problema de este tipo.

6.1.4 Mayor control del entorno

Está claro que para poder tener más opciones a la hora de desarrollar un juego es necesario tener un mayor control del entorno. Tanto para situar decorado del escenario, npcs o limitar el paso al jugador por ciertas zonas.

Para conseguir un control del entorno adecuado sería necesario avanzar en el tema de visión por computador y detección de esquinas. Pudiendo distinguir las paredes del suelo se podría por ejemplo crear un río solo en el suelo o añadir decoración en las paredes que sumerjan más al usuario en el juego.

6.2 Control de acceso

Debido a que se controla un robot y que este control no se puede dar al mismo tiempo por dos usuarios distintos sería necesario establecer un sistema de seguridad el cual otorgue el control del robot al primer usuario que se conecte siempre y cuando no haya un usuario ya conectado.

Dicho sistema podría implementarse mediante sesiones de html5 ya que las maneja el servidor. La sesión del usuario controlador quedaría registrada de tal manera que antes de enviar órdenes al robot se comprobaría que proceden de dicha sesión. Cuando se libere la sesión controladora el sistema le otorgará el control al siguiente usuario que se conecte.

6.3 Multijugador

Sería posible establecer un sistema multijugador si se consiguiese que desde un robot se pudiese ver el personaje o gráficos generados por otro robot. Hay que tener en cuenta que actualmente los gráficos generados en WebGL son creados en el cliente, por lo tanto necesitaríamos establecer una manera de pasar el estado de los gráficos al servidor para que otros clientes pudiesen servirse de ellos y pintarlos en sus pantallas también.

El establecer un sistema multijugador podría abrir el abanico de posibilidades y crear juegos multijugadores sociales o competitivos. Un ejemplo podría ser una habitación o entorno con varios robots controlados remotamente a distancia, por lo tanto en dicho entorno se podría establecer un juego del tipo *second life* pero sabiendo que se está jugando en un entorno real o un juego competitivo en el que los jugadores fuesen siendo eliminados en un entorno preparado para el caso con barricadas o trincheras sabiendo el jugador en todo momento que eso que él está controlando está ocurriendo de verdad en algún punto del mundo y que se está conectado de esa manera con otros jugadores.

7 Conclusiones y valoración personal

La realización general de este trabajo ha sido satisfactoria en cuanto a que se han conseguido completar todos los objetivos propuestos. Al finalizar el proyecto se dispone de un prototipo funcional de lo que podría ser un nuevo sistema de juego basado en realidad aumentada y un robot dirigido mediante una aplicación web. Todo ello se ha realizado y entregado en el tiempo correspondiente a la realización de este TFG por lo que ha rasgos generales se ha cumplido con lo propuesto desde un principio.

El robot se realizó en un tiempo menor al esperado y no surgieron muchas complicaciones en cuanto a su montaje y desarrollo corresponden. Respecto a este punto lo que más tiempo consumió fue la realización del código y su implementación con la aplicación en general.

Con lo que a la aplicación web se refiere las complicaciones fueron ocasionadas debido a que la tecnología empleada aún está en desarrollo y hay partes que siguen sin ser estables como es el caso del exportador de Blender a JSON. Además fue complicado el conseguir que el canvas se ajustase a la pantalla correctamente sin cortar demasiada imagen. Por todo lo demás la implementación de la aplicación web fue sencilla y simplemente hubo que adaptar el ejemplo de Three.js al proyecto e integrar la parte del control del robot.

La parte del modelado fue la que realmente llevó más tiempo del esperado y del necesario ya que se realizó un modelado demasiado exhaustivo de partes que o no se ven o se podrían haber suplido con las texturas. Es el caso de la cara del personaje en la cual se empleó mucho tiempo siendo que el personaje se visualiza de espaldas. Además los detalles de la cara implementados mediante el modelado se obvian al aplicar una textura por lo que se podría haber realizado un modelado más sencillo dejando la adición de detalles en la parte de la textura. También se gastó mucho tiempo en la animación del personaje al tener que pasar por un periodo de aprendizaje previo. No obstante el resultado final es satisfactorio y estoy satisfecho con él.

Sin embargo hay una faceta del proyecto que no ha resultado ser satisfactoria. Una vez se consiguió implementar el robot y su control se vio que en algunas ocasiones las órdenes mandadas no llegaban correctamente o llegaban tarde. Esto es debido al medio de comunicación WiFi, el cual no es el más adecuado cuando se trata de controlar un robot debido a su latencia y pérdida de información. Si se habla de velocidad y persistencia, otro medio de comunicación como es el Bluetooth o el radio control habrían sido más efectivos pero habría sido más difícil el crear una aplicación para su control que se pudiese usar tanto en ordenadores como en móviles de manera sencilla. Por lo tanto el hecho de que la comunicación entre la aplicación y el robot no sea del todo efectiva es el único punto con el que no he quedado satisfecho y el que podría ser el mayor inconveniente a la hora de avanzar y ampliar el proyecto.

Dicho esto podemos concluir en que la parte más satisfactoria del proyecto ha sido el poder usar todos los conocimientos aprendidos durante el grado estudiado y avanzar en los nuevos adquiridos para la realización de este TFG. El hecho de tener entre manos un proyecto propio me ha hecho ver la importancia de trabajar en algo que realmente te haga ilusión y como todo cambia de hacer algo por obligación a levantarte por las mañanas con ganas de seguir trabajando. Considero que en un mundo tan globalizado es importante encontrar lo que realmente te apasiona, y que es una suerte que eso sea la tecnología ya que es un ámbito en constante crecimiento y que te permite aprender de una manera autodidacta con la simple ayuda de un ordenador y tus ganas por seguir avanzando. Además el mundo de la tecnología no solo te permite estar al tanto de los últimos inventos de nuestro presente, sino que también te permite construir el futuro.

8 Bibliografía

[1] Tutorial sobre modelado, http://cgi.tutsplus.com/es/categories/modeling

[2] Fuente de descarga del pelo, http://www.blendswap.com/blends/view/63151

[3] Banco de animaciones, https://sites.google.com/a/cgspeed.com/cgspeed/motion-capture/daz-friendly-release

[4] Documentación de Raspberry Pi, https://www.raspberrypi.org/documentation/

[5] Comparación de velocidad de distintos servidores web, https://www.jeremymorgan.com/blog/programming/raspberry-pi-web-server-comparison/

[6] Documentación de RaspiMJPEG, http://elinux.org/RPi-Cam-Web-Interface#RaspiMJPEG

[7] Documentación sobre la shield L298N, https://www.bananarobotics.com/shop/How-to-use-the-L298N-Dual-H-Bridge-Motor-Driver

[8] Crear una red Ad-Hoc en la Raspberry Pi, http://slicepi.com/creating-an-ad-hoc-network-for-your-raspberry-pi/

[9] Conectarse a una red Ad-Hoc con Windows 10, https://globalcache.zendesk.com/entries/82172789-FAQ-Windows-8-1-and-Windows-10-AdHoc-network-support-solution

[10] Documentación de Three.js, http://threejs.org/docs/

[11] GitHub de Three.js, https://github.com/mrdoob/three.js/

[12] GitHub de JSARToolKit, https://github.com/kig/JSARToolKit

[13] Dudas en general, http://stackoverflow.com/

[14] Tutorial de JSARToolKit, http://www.html5rocks.com/es/tutorials/webgl/jsartoolkit_webrtc/