UNIVERSIDAD DE  VALLADOLID

E.T.S.I. TELECOMUNICACIÓN

TRABAJO FIN DE GRADO

GRADO EN INGENIERÍA DE TECNOLOGÍAS ESPECÍFICAS DE TELECOMUNICACIÓN, MENCIÓN EN INGENIERÍA TELEMÁTICA

Juego Serio para entrenamiento de personas con discapacidad física mediante tecnología Kinect

Autor:
Ana Isabel Hervás Torío
Tutor:
Mario Martínez Zarzuela
Valladolid, Diciembre de 2014

TÍTULO:	**Juego Serio para entrenamiento de personas con discapacidad física mediante tecnología Kinect**
AUTOR:	**Ana Isabel Hervás Torío**
TUTOR:	**Mario Martínez Zarzuela**
DEPARTAMENTO:	**Departamento de Teoría de la Señal y Comunicaciones e Ingeniería Telemática**

TRIBUNAL

PRESIDENTE:	**Dña Míriam Antón Rodríguez**
VOCAL:	**D. Mario Martínez Zarzuela**
SECRETARIO	**D. David González Ortega**
SUPLENTE	**D. Francisco Javier Díaz Pernas**
SUPLENTE	**D. Carlos Gómez Peña**

FECHA:	**de Diciembre de 2014**
CALIFICACIÓN:	

A mis amigos, familia y profesores que me acompañaron en estos últimos años; por apoyarme, aguantarme y mostrarme que de todo se puede sacar una enseñanza positiva.

Resumen de TFG

El Trabajo Fin de Grado desarrollado consiste en la realización de un videojuego serio el cual a través de la tecnología que ofrece la Kinect de Microsoft® sirva como mecanismo de rehabilitación para personas con problemas de movilidad, mayormente causados tras sufrir un ictus.

Inicialmente se hará un estudio de los trastornos causantes de estos problemas de movilidad y de las tecnologías a utilizar, para después explicar en profundidad el juego hecho con el motor de juego Unity3D y el *plugin* Zigfu.

Este proyecto pretende ofrecer una herramienta de ayuda para fisioterapeutas y pacientes para facilitar la rehabilitación y la independencia de estos últimos.

Palabras clave

Realidad Virtual, Kinect, Rehabilitación Física, Juego Serio.

Abstract

This developed master thesis consists on making a serious videogame which, through the tecnology that is offered by Microsoft®'s Kinect, could be an important instrument of rehabilitation of people with movility problems, which mainly are caused by the ictus.

At first, we made a research about the diferent issues that cause these movility problems and the technologies we are going to use and, then, we deeply explain the game made by the game engine called Unity3D and the Zigfu plugin.

This proyect will try to offer some help to physiotherapist and patients in order to ease the rehabilitation and independency processes.

Keywords

Virtual Reality, Kinect, Physical Rehabilitation, Serious Game.

Índice de contenidos

Índice de figuras

Índice de tablas

Capítulo 1. Introducción

En este primer capítulo se comentarán los aspectos más generales del proyecto, como son la motivación por la cual me decidí a hacer un trabajo de este estilo, los conocimientos previos de los que disponía para llevar a cabo dicha tarea, los objetivos que se pretendían lograr y las necesidades que se querían cubrir a través de la aplicación para quienes ésta iba destinada, así como también se dará resumido cómo ha sido el desarrollo del trabajo.

1.1. Motivación

A lo largo de la carrera hemos realizado múltiples proyectos a través de la programación, como han sido aplicaciones de seguridad, aplicaciones para móviles, pequeños juegos al estilo "el ahorcado" o "sopa de letras" u otros con algo de más peso como el "3 en raya", pero sin embargo la parte de la programación que quizás más llame la atención, sobretodo de la gente joven, es la de los videojuegos. Y el hecho de no haber realizado algo tan amplio y complejo como un videojuego a lo largo de mi proceso de aprendizaje, ha sido mi primera motivación para escoger la realización de este juego serio para entrenamiento de personas con discapacidad física mediante tecnología Kinect.

Por otro lado, la inmensa mayoría de los videojuegos, y los de más éxito a la par, están dedicados únicamente al entretenimiento sin tener en cuenta algún otro fin constructivo; pero debido a que en pleno siglo XXI hay un convencimiento de que cualquier expresión del medio digital es parte de nuestra cultura y por lo tanto debe formar parte de la educación de las personas, son cada vez más las aplicaciones o videojuegos educativos que van surgiendo al mercado (Gros, 2008).

Pero, ¿qué ocurre con los videojuegos destinados a la rehabilitación? Pues es un hecho muy reciente el que cada vez haya más proyectos e investigaciones acerca de este asunto, que vienen de la mano con el desarrollo en campos como la robótica o la realidad virtual, en los cuales gracias a dispositivos robóticos se puede practicar ejercicios orientados a una tarea específica de manera repetitiva e intensiva o simplemente se pueden realizar mediante juegos de ordenador o técnicas de realidad virtual (Espinosa, Arroyo, Martín, Ruiz y Moreno, 2010).

Todas estas son técnicas muy prometedoras ya que se pueden integrar los procesos cognitivos y emocionales en la rehabilitación motora mediante la interacción directa del paciente con objetos virtuales, y esta realidad virtual

proporciona una correspondencia espacial en tres dimensiones entre el nivel de movimiento en la realidad y el visto en la pantalla, favoreciendo el aprendizaje motor (Espinosa et al., 2010). Además, anteriormente esto parecía algo bastante costoso debido a la necesidad de usar dispositivos complejos y poco comunes, pero hoy en día se necesita poco más que un ordenador personal y una videoconsola conectable a un monitor junto con algún accesorio como puede ser el mando de Wii, guantes de realidad virtual o la Kinect de Microsoft®, entre otros.

Junto a todo esto está el hecho evidente de que para un niño que necesite rehabilitación, siempre se le va a hacer más ameno y divertido que sea en frente de un videojuego y por lo tanto van a aumentar su motivación e interés por realizar los ejercicios de manera intensa y repetitiva.

Siguiendo con mis motivaciones, el hecho de hacer algo con la tecnología que pueda ayudar a personas con algún tipo de problema (en este caso de movilidad) es otro punto que debo mencionar. Ya que creo que eso le ha sumado cierto nivel de esfuerzo por mi parte a la realización de este trabajo, el hecho de poder hacer algo que fuera realmente útil para estas personas y no simplemente algo para mostrar y que quizás no se le diese ningún uso.

En concreto, con este proyecto se pretende ayudar a aumentar el nivel de movilidad a personas que hayan sufrido cualquier tipo de enfermedad, trastorno o accidente que les haya dejado secuelas, como pueden ser un ictus, que se estén rehabilitando de un accidente de tráfico o relacionado, con lesiones medulares, ataxias, esclerosis múltiples, distintas patologías infantiles, etc.

1.2. Conocimientos Previos

Para poder realizar este proyecto, eran necesarios unos conocimientos previos que fueron adquiridos a lo largo de la carrera. Estos conocimientos se basaban principalmente en la programación en diferentes lenguajes. En concreto, a lo largo de la carrera hicimos varios programas utilizando lenguajes de diferentes niveles como son C, C++, ensamblador, Java, Verilog y HTML.

Teniendo este nivel de soltura en lo que a la programación se refiere, no resultó difícil aprender a programar en los nuevos lenguajes que el programa Unity3D requería, que son Javascript y C#. Ambos son orientados a objeto, al igual que Java, con el cual se realizaron varios trabajos en la carrera.

Aun así, la primera parte del proyecto se basó en visualizar tutoriales y ejemplos de Javascipt y C# para aprender a manejar correctamente estos lenguajes. La utilización de estos es necesaria para aplicar un comportamiento a los elementos que construiremos y que van a formar parte del videojuego y para que puedan interactuar entre ellos, por lo tanto podríamos decir que es una de las partes más importantes.

Aparte de esto, el dispositivo que se iba a utilizar para la interacción entre el usuario y el videojuego también era conocido por mi. Este dispositivo es la Kinect de Microsoft®, del cual sabía que captura el movimiento corporal y nos permite

jugar con nuestro propio cuerpo sin la necesidad de ningún mando, por lo tanto es un elemento muy adecuado para nuestro propósito de la rehabilitación. Pero de este dispositivo continuaremos hablando profundamente más adelante.

Independientemente de estos conocimientos acerca de la programación y del dispositivo a utilizar, no disponía de ninguno más sobre otros aspectos como es el motor de juegos que se iba a usar, por lo tanto también se dedicaron unas semanas a aprender a manejar Unity3D, del cual también hablaremos más adelante en profundidad. Tampoco sobre programas para la creación y animación de personajes, como son MakeHuman, Blender o Mecanim, del cual sólo se utilizó este último ya que viene integrado de forma nativa en el propio Unity3D y fue suficiente para lo que lo necesitábamos realizar.

1.3. Objetivos

A la hora de plantear este trabajo fin de grado se tenían en consideración diferentes objetivos que se querían conseguir con el proyecto. El primero y fundamental era el de ayudar a las personas que tuvieran problemas de movilidad en general, por medio de la rehabilitación, para que pudieran aumentar esos niveles de movimiento por medio de un entrenamiento más o menos constante y que resultase sencillo de realizar.

Además de los estos beneficios del juego en los usuarios, también se buscan otro tipo de ventajas, entre las que destaca la comodidad de poder realizar todo esto desde la propia casa del paciente y por sí mismo. Es decir, el fisioterapeuta controlará la evolución del paciente y le indicará los ejercicios más apropiados para él, pero nuestro videojuego ayudará a continuar este proceso desde casa y sin necesidad de profesionales que supervisen al jugador. Aunque el juego tendrá unas opciones en las cuales el fisioterapeuta pueda modificar ciertos valores acorde con las necesidades del paciente, por lo tanto se pretende mezclar la independencia del paciente con cierto control por parte de profesionales.

Por otro lado, se pretende borrar los conceptos de “aburrida” o “costosa” que puede tener la rehabilitación, haciendo de esta algo ameno, que en definitiva es la finalidad de un videojuego. Así introducimos el hecho de mejorar mientras nos divertimos, y ya no sólo eso, si no que además generemos un cierto grado de adicción que produzca constancia en el entrenamiento, ya que la constancia es la base de toda mejora.

Todos estos objetivos son aplicables para todo tipo de personas, tanto para ancianos como jóvenes, pero quizás la parte que más beneficiada puede salir son los niños que sufren estos problemas, ya que son los que más disfrutan con los videojuegos y posiblemente sean las personas a las que más les cuesta ser constantes con un tratamiento y quizás, los que peor llevan tener estas discapacidades ya que les puede impedir hacer cosas tan básicas como jugar a un deporte con sus amigos, pintar, comunicarse con soltura e incluso jugar a otro tipo de videojuegos.

1.4. Necesidades a cubrir

Este proyecto desde su origen iba a ser un proyecto de rehabilitación destinado a la asociación ASPAYM, la cual nos hizo llegar una propuesta con unas condiciones muy claras de lo que tenían en mente para este trabajo.

La asociación Castellano-Leonesa ASPAYM nació en 1992 en Valladolid y es una asociación que trabaja para mejorar las condiciones de vida del colectivo de personas con discapacidad en general y lesionados medulares en particular (Fundación ASPAYM Castilla y León, 2014). A partir de ese momento de su creación, ha ido creciendo poco a poco a través de sus delegaciones y mediante programas o actividades en las que participan sus socios y usuarios. Además cabe destacar su inversión y creencia en la tecnología como medio de ayuda a estas personas, como se ve en su interés por este proyecto llevado a cabo.

La mayoría de las necesidades de la asociación básicamente coincidían con nuestros objetivos iniciales y se pretendía mayormente la rehabilitación de personas que hubieran sufrido un ictus y por lo tanto tuvieran trastornos en la movilidad, a la hora de comunicarse, en la percepción, visuales o de su imagen corporal, que son las secuelas que éste puede dejar (Ictussen, 2014).

Junto con personas pertenecientes a la asociación estuvimos debatiendo sobre qué juego podía ser más adecuado para nuestro propósito, y llegamos a la conclusión de hacer el juego del que hablaremos en profundidad en unos apartados más adelante. Este tenía que cubrir las siguientes necesidades:

- Independencia del usuario con respecto al fisioterapeuta.
- Capacidad por el fisioterapeuta de modificar valores o parámetros del juego en función de las necesidades del usuario.
- Mejora de la movilidad tanto en brazos como piernas y proporcional a las capacidades de cada usuario.
- Mejora de la coordinación y de la propia percepción de su cuerpo que tiene el usuario.
- Generación de competitividad constructiva por medio de un sistema de puntuaciones.
- Ser claro y jugable para un gran rango de edades.
- Tener incentivos, como puntos o vidas extra, que animen al usuario a jugar y así lo haga más a menudo.

Bajo todas estas bases y encaminado por las pruebas con pacientes reales que hacíamos periódicamente, elaboramos un proyecto que fuera satisfactorio para la asociación y que resultase efectivo en la rehabilitación.

1.5. Desarrollo del trabajo

El trabajo fin de grado realizado ha estado ligado a las prácticas en empresa que he hecho en la misma Universidad de Valladolid, y las cuales han consistido en la formación en las tecnologías implicadas y la impartición de un curso acerca del manejo del programa de desarrollo y de los lenguajes de programación utilizados.

Este trabajo en cuestión se ha basado en el desarrollo de un videojuego serio para la rehabilitación de personas de movilidad reducida. Por lo tanto, la finalidad de éste es hacer que el proceso de rehabilitación de una persona pueda llegar a ser algo divertido e incluso adictivo, al añadir el factor de competitividad entre unos jugadores y otros según el número de pruebas que superen, y que sea un proceso que tenga cierto grado de independencia del paciente con respecto al fisioterapeuta o médico que le lleve la rehabilitación y así poderlo realizar de forma cómoda en su casa, evitando el desplazamiento a otro lugar, los costes que eso pueda conllevar e incluso evitando inconvenientes a personas que no puedan desplazarse por si mismas.

Para hacer todo esto vamos a utilizar el programa Unity3D, que es un ecosistema de desarrollo de juegos, o mejor explicado, un poderoso motor de juegos totalmente integrado con un conjunto completo de herramientas intuitivas y flujos de trabajo rápido para crear contenido interactivo para casi cualquier plataforma de las más conocidas (Unity Technologies, 2014a). Por lo que mi primer paso para la realización de estas prácticas fue aprender a manejar este programa mediante el visionado de tutoriales y la realización de diversidad de ejemplos.

Una vez que pasó poco más de mi primera semana de prácticas, consideré que tenía cierta soltura en el manejo de este programa, por lo que comencé a elaborar unas diapositivas con la finalidad de dar un curso básico sobre Unity3D. Este curso se basaba en introducir lo que era Unity3D, explicar conceptos básicos del programa, elaborar algunos de los elementos más importantes que aparecerán en un videojuego, explicar la utilización los lenguajes de programación JavaScript y C# (los cuales se utilizan en este programa junto con Boo, el cual es menos utilizado) y la realización de una serie de ejemplos básicos, mostrando los resultados en el momento. Este curso tuvo lugar a mediados del mes de mayo.

Justo antes de la impartición de este curso, asistí junto a mi tutor en la universidad y otros compañeros (los cuales trabajan en la universidad en el desarrollo de videojuegos) a una reunión con representantes de la fundación ASPAYM de Castilla y León. En esta reunión establecimos el camino que tenía que tomar el videojuego que yo iba a realizar y cómo hacerlo de una forma que resultase efectivo tanto en resultados como en entretenimiento. Así se propuso que inicialmente se hiciera un registro de los ángulos de movimiento que el paciente era capaz de articular y que según esto se le propusieran una serie de posiciones en una silueta que se le iría acercando y que él tendría que adoptar para poder ir ganando puntos, y que todo esto pudiera ser regulado de alguna forma por el fisioterapeuta.

Para la realización de este videojuego, necesitaremos por lo tanto algún elemento que capture los movimientos del usuario, por lo que utilizaremos la máquina Kinect de Microsoft® junto a un PC en el cual se ejecutará el juego. Utilizaremos Kinect debido a que permite a los usuarios controlar e interactuar con la consola sin necesidad de tener contacto físico con un controlador de videojuegos tradicional, mediante una interfaz natural de usuario que reconoce gestos, comandos de voz y objetos e imágenes (Kinect for Windows, 2014).

Por lo tanto, a partir de aquí mis prácticas en empresa se basaron en acudir al laboratorio del grupo GTI (Grupo de Telemática e Imagen) y orientar mi formación hacia las tecnologías para llevar a cabo todas estas ideas establecidas en la reunión, utilizando el programa Unity3D y eligiendo el lenguaje de programación orientado a objetos C# para la programación del mismo; y acudiendo cada cierto tiempo a la fundación ASPAYM para mostrar mis avances en el proyecto y hacer pruebas con personas que tuvieran problemas de movilidad reales y así acertar en qué líneas futuras se debían seguir para mejorar.

Capítulo 2. Enfermedades o problemas causantes de problemas de movilidad

La rehabilitación que se propone a través de nuestro videojuego es para las personas que hayan sufrido cualquier tipo de enfermedades o trastornos que les hayan dejado secuelas físicas y les limite la movilidad y coordinación de las extremidades o tronco.

Así se pueden producir disfunciones en el sistema esquelético, alteraciones en la longitud o fuerza muscular, en los mecanismos modulares del sistema nervioso, en el elemento biomecánico o alteraciones múltiples de los integrantes del movimiento; los cuales encontramos desarrollados en el libro de Sahrmann (2005).

A continuación desarrollaremos ciertas de estas enfermedades o trastornos. En concreto, desarrollaremos con las que la fundación ASPAYM está más familiarizada a tratar, como ya mencionamos anteriormente, y que son de las más conocidas por la sociedad en general.

2.1. Ictus

El ictus, también conocido como enfermedad cerebrovascular, es cualquier trastorno de la circulación cerebral, que generalmente es brusco. Como nos muestra la *figura 2.1*, puede producirse porque se interrumpa el flujo sanguíneo a una parte del cerebro (Ictus isquémico, 75% de las veces) o por la rotura de una arteria o vena cerebral (Ictus hemorrágico, 25% de las veces) (Agencia Valenciana de Salud, 2007).

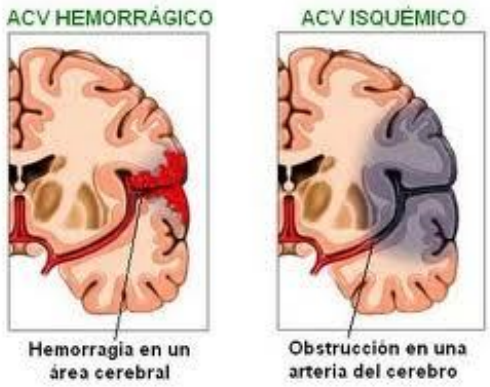


Figura 2.1: Tipos de ictus

El cerebro es un órgano muy sensible que aunque suponga solamente el 2% del peso humano, necesita casi el 20% de la circulación. Además, necesita constantemente nutrientes y oxígeno, ya que no tiene reservas, por ello es tan sensible ante la falta de flujo cerebral y es enorme la cantidad de vasos sanguíneos que llegan hasta él.

Generalmente, las personas mayores son más propensas a sufrir un ictus, aunque los jóvenes tampoco están exentos de ello, por lo que hay que tener cuidado y evitar los llamados factores de riesgo que hacen que este trastorno pueda aparecer, como son ciertas circunstancias ambientales, personales o sociales y otras inevitables como el sexo, edad, raza, historia familiar o haber sufrido un ictus con anterioridad (Grupo de Estudio de las Enfermedades Cerebrovasculares, 2003). Al fin y al cabo, el mejor "tratamiento" para el ictus es la prevención.

A pesar de que cada vez hay mejores técnicas de prevención, el ictus es la tercera causa de muerte en los países desarrollados y la primera en España por entidades específicas (Montaner, 2009). También es la segunda causa de demencia, justo después del Alzheimer, y el primer motivo de incapacidad en la población adulta.

Tal es el número de personas que han sufrido este trastorno, que en muchos hospitales está la conocida como Unidad de Ictus (UI), que es una zona específica dentro del hospital la cual se dedica al cuidado de pacientes con patologías cerebrovasculares, por parte de personal especializado y de forma multidisciplinar (Montaner, 2009). El objetivo de las UIs es que la atención médica de los pacientes mejore y que así se disminuyan los niveles de mortalidad, además de enseñar a familiares y a los propios pacientes sobre los cuidados sanitarios de éste y la rehabilitación que tiene que seguir.

2.1.1. Síntomas

Sobretodo en las personas que poseen un riesgo muy alto de sufrir un ictus, es muy importante saber cuáles son los conocidos como síntomas de alarma que preceden a este trastorno.

El mayor problema de estos síntomas es que no traen el "dolor", que es lo que más nos puede llamar la atención a la hora de tomar precauciones, y esto es muy peligroso ya que cuanto antes se vaya al hospital y se reciba atención, menores serán las consecuencias del ictus (Grupo de Estudio de las Enfermedades Cerebrovasculares, 2003).

Así pues, los síntomas que hay que tener en cuenta son la pérdida de fuerza o la sensación de "hormigueo" de un lado de cualquier parte del cuerpo, ambas de forma brusca; la pérdida súbita de visión parcial o total en uno o ambos ojos; alteración repentina en el lenguaje, como problemas al articular palabras o en la expresión; dolor de cabeza repentino y de mayor intensidad que lo habitual; y la sensación de vértigo o pérdida del equilibrio (Grupo de Estudio de las Enfermedades Cerebrovasculares, 2003). Si aparecen cualquiera de estos, y con

más razón si son más de uno los que nos encontramos en la misma persona, es un motivo de alarma para acudir rápidamente a un profesional.

2.1.2. Trastornos

Una vez que el ictus ha sido inevitable, éste conlleva una serie de trastornos que pueden aparecer en mayor o menor grado. Estos pueden ser trastornos motores, de la comunicación, de la percepción o emocionales, psicológicos y de conducta (Grupo de Estudio de las Enfermedades Cerebrovasculares, 2003).

En los trastornos motores nos encontramos con una plejía o parálisis si es una extremidad la que ha perdido la fuerza y el paciente es incapaz de moverla, una paresia si, aunque con trabajo, puede realizar ciertos movimientos con la extremidad, una hemiplejía si la debilidad ha afectado a toda una mitad del cuerpo y otros trastornos basados en que se produzcan movimientos involuntarios(Grupo de Estudio de las Enfermedades Cerebrovasculares, 2003). Una curiosidad es que si la lesión se ha producido en un lateral del cerebro, los problemas de movilidad afecten al lado contrario del cuerpo. Esto es así porque a la altura del cuello, en la médula espinal, se produce un cruce entre el conjunto de fibras nerviosas (vía piramidal) de la izquierda el de la derecha. Otra curiosidad es porqué a veces la falta de movilidad sólo se manifiesta en ciertas extremidades, y es porque como hay diferentes neuronas en cada lado del cerebro dedicadas a ciertas extremidades, puede que sólo una parte de ellas sea la que se vea afectada y no su totalidad.

Los trastornos de la comunicación pueden ser en el lenguaje, llamadas afasias, en el habla debido a la debilidad de los músculos de la cara, las disartrias, o que se produzca mutismo y no se pueda emitir ningún sonido (Grupo de Estudio de las Enfermedades Cerebrovasculares, 2003). Además, éstas acostumbran a traer dificultades también a la hora de leer, escribir o comprender. Muchos pacientes de los cuales tienen estos problemas de expresarse, consiguen emitir palabras que sean repetición de lo que han oído.

Por otro lado, los trastornos de la percepción son de varios tipos. En primer lugar encontramos los visuales, que son muy frecuentes debido a que la arteria que lleva sangre al ojo procede de la arteria carótida interna, que es la que lleva la mayoría de sangre al cerebro, por lo que si esta se obstruye se dañará tanto el cerebro como el ojo (Grupo de Estudio de las Enfermedades Cerebrovasculares, 2003). En la pérdida de visión se produce el mismo fenómeno de inversión que en la pérdida de movilidad de las extremidades, ya que el hemisferio derecho del cerebro se ocupa del ojo izquierdo y viceversa. También encontramos los trastornos de la sensibilidad, ya que la temperatura, el tacto, el dolor, etc., se recogen por unos receptores que llegan hasta la médula espinal por los nervios sensitivos, estas vías de sensibilidad pueden lesionarse, provocando sensación de "hormigueo" o acorchamiento. Y finalmente encontramos trastornos en la percepción de la imagen corporal, que vienen provocados por la pérdida de sensibilidad en un hemisferio del cuerpo hasta el punto de hacer creer al paciente que están moviendo alguna extremidad y que esto no sea cierto.

Para terminar, los trastornos emocionales, psicológicos y de conducta son muy comunes debido a los sentimientos de impotencia, depresión o ansiedad que se dan después de sufrir un ictus (Grupo de Estudio de las Enfermedades Cerebrovasculares, 2003). También se producen por el daño en ciertas zonas del cerebro que hacen perder el control emocional o la memoria, por lo tanto la comunicación y el reaprender a realizar tareas rutinarias es la mejor solución.

2.1.3. Tratamientos

Antes de nada cabe destacar que cada persona necesita un tratamiento especializado debido a que el ictus en cada una se produce por causas diferentes. A pesar de esto, hay unos tratamientos a seguir.

Lo primero sería desatascar la arteria carótida, que es la principal vía de transporte de la sangre al cerebro, ya que el bloqueo de ésta es la causante de un 20% de ictus isquémicos (Agencia Valenciana de Salud, 2007). Se puede hacer una endarterectomía, que se basa en cirugía, o una angioplastia, siempre y cuando el grado de estrechamiento de la arteria sea menor al 50%.

El ictus también puede producirse por una arritmia cardiaca llamada fibrilación auricular, por lo tanto hay que evitar los coágulos cardiacos mediante anticoagulantes orales, siempre que el paciente no tenga alergia o contradicciones para el uso de ellos. El más usado es el Sintrom (Agencia Valenciana de Salud, 2007).

Pero ante todo, lo que hay que levar siempre es un tratamiento médico con medidas farmacológicas para evitar el riesgo de volver a sufrir un ictus (Agencia Valenciana de Salud, 2007). Así se mantienen regulados la hipertensión, diabetes y colesterol.

La rehabilitación además es imprescindible para que los pacientes puedan llegar a llevar una vida más o menos normal. Ésta debe ser dirigida por un médico rehabilitador y tendrá una duración variable según el grado de afectación en el paciente, aunque se compone de unas fases fijas que son la de aceptación por parte del paciente de su situación, la movilización o automovilización de ciertas partes del cuerpo, la realización de ciertos ejercicios sencillos controlados por un especialista y si la recuperación es favorable, más adelante se desarrollaran de la misma manera actividades más complejas (Agencia Valenciana de Salud, 2007). Cuanto antes empiece esta movilización, mejor será para el paciente.

2.2. Accidentes de tráfico

Los accidentes de tráfico son uno de los principales problemas en cuanto a mortalidad de la población en casi todo el mundo, además del peligro que suponen por las secuelas que conllevan aunque se sobreviva a ellos.

En nuestro país en concreto se producen al año al menos 2.000.000 de accidentes con daños materiales, que ocasionan importantes pérdidas económicas, según la Unión Española de Entidades Aseguradoras y Reaseguradoras (UNESPA).

A pesar de la gran importancia que tienen las pérdidas económicas, lo más importante por su alcance para la salud de la población, es ser conscientes tanto del número de accidentes que finaliza con alguna víctima y la gravedad de las lesiones, como también de los factores que han influido en que se produjese dicho accidente.

Si observamos los datos del año 2013 ofrecidos por la Dirección General de Tráfico (2014), el número de accidentes con víctimas mortales fue de 994, ocasionando unos 5.206 heridos graves y 1.128 fallecidos (computo realizado a 30 días: la victima fallece en el acto o durante los próximos 30 días a contar desde que se produce el accidente). El número promedio de personas que mueren al día ha disminuido considerablemente como muestra la *figura 2.2*, aunque aún hay que trabajar en ello.

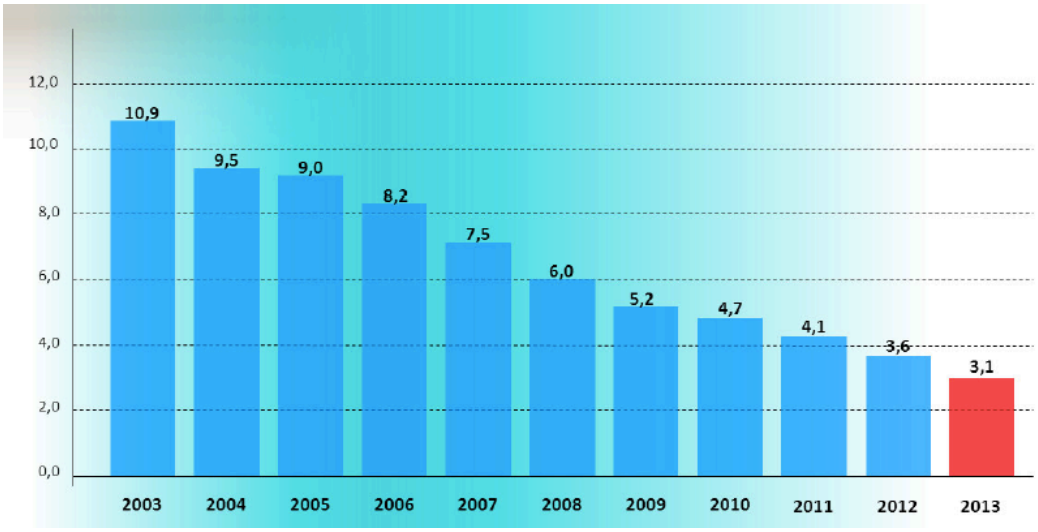
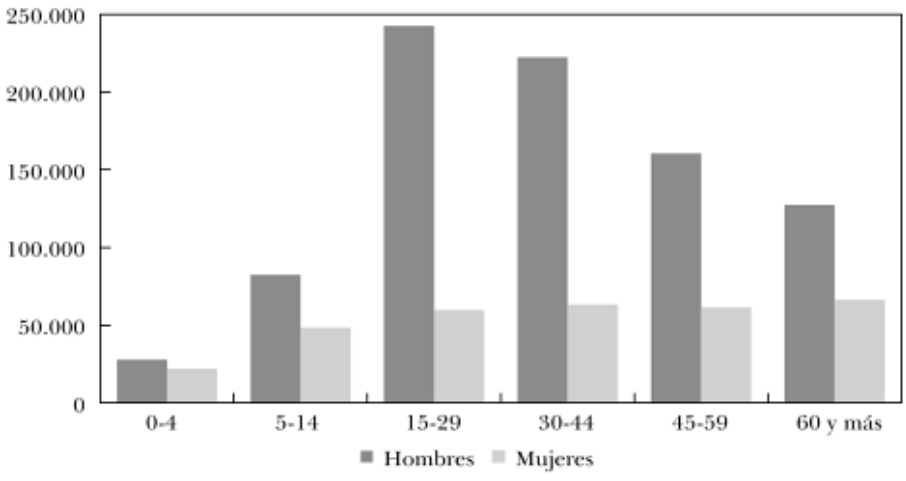


Figura 2.2: Promedio de personas que mueren al día por accidentes de tráfico en España

Además la cantidad de accidentes de tráfico que se producen varían considerablemente de una franja de edad a otra, como se aprecia en la *figura 2.3*, la cual nos muestra que entre los 15 y los 49 años es dónde más mortalidad se encuentra.



Fuente: OMS (2002).

Figura 2.3: Mortalidad en España en accidentes de tráfico según la franja de edad

Pero no sólo estos son los conductores del vehículo, si no que también son los acompañantes o peatones los que están bajo peligro cuando viajan, viendo en la *figura 2.4* cómo ambos ocupan un porcentaje importante dentro de las víctimas mortales que se producen en nuestro país.

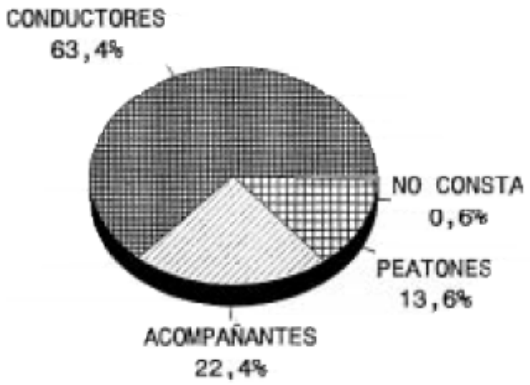


Figura 2.4: Porcentaje de mortalidad en España en accidentes de tráfico según el puesto que ocupa la víctima

Para que una persona sobreviva a un accidente de tráfico, es crucial el momento justo después de que se produzca éste (Dirección General de Tráfico, 2014). Así en los primeros segundos o minutos después del accidente mueren el 10% de las personas, pero en la primera o dos horas primeras, el porcentaje aumenta hasta el 75%. Semanas tras el traumatismos es cuando se producen el 15% de fallecimientos.

Pero no sólo hay que tener en cuenta a estos accidentes mortales, si no que son miles de personas en España las que al año sufren accidentes de tráfico, quedando en muchas de ellas graves secuelas que requieren un proceso más o menos largo de recuperación.

2.2.1. Secuelas

Según la Dirección General de Tráfico (DGT) (2014), *la secuela es un daño estable, definitivo e irreversible, consecuencia directa o resultado de una o varias lesiones originadas por el accidente de circulación. La secuela a menudo no es más que la prolongación de la propia lesión original*. Éstas van a requerir tratamiento médico durante diferentes periodos de tiempo, pudiendo ser incluso de por vida.

Las secuelas más frecuentes (en el 90% de las ocasiones) se producen en la columna vertebral y pelvis, en las extremidades superiores e inferiores, en la cabeza, cráneo o cara, u ocasionan un perjuicio estético ligero o medio. También un pequeño 0,5% afectan al sistema nervioso central.

Aparte de todo esto, también hay que tener en cuenta otras secuelas secundarias, que son de carácter personal y socio-económicas, como son los días de hospitalización, de baja laboral, la incapacidad profesional, la necesidad de una

tercera persona y de la adaptación de elementos de la vida cotidiana a las nuevas capacidades (DGT, 2014).

2.2.2. Rehabilitación

Para poder devolver al accidentado la mayor independencia y capacidad funcional posible, se necesita pasar por un proceso de rehabilitación (DGT, 2014). Ésta tiene que cubrir muchos campos por la cantidad de lesiones o trastornos que pueden producirse, por lo tanto se va a necesitar un amplio abanico de profesionales, como un logopeda, fisioterapeuta, terapeutas ocupacionales, auxiliares de enfermería, psicólogos, neuropsicólogos, trabajador social, etc...

Es apropiado que el proceso de la rehabilitación se inicie mientras el paciente continúa hospitalizado, y básicamente está formado por un médico rehabilitador, el cual asume la responsabilidad realizando el historial clínico, diagnóstico, pronóstico y el plan terapéutico de cada paciente; un fisioterapeuta, que utiliza tratamientos físicos para mejorar la movilidad, la potencia, la resistencia muscular, la coordinación o la sensibilidad, entre otras cosas; un terapeuta ocupacional, que se ocupa de dar la mayor autonomía al paciente tanto a nivel personal como con su ambiente; y un logopeda, que tratará los trastornos en el área de la comunicación y deglución (DGT, 2014).

2.3. Lesiones medulares

La médula espinal se encuentra discurriendo entre nuestras vértebras, protegida por ellas, ya que es una parte muy importante a la par que sensible, ya que transporta e integra la información sensitiva y motora entre el cerebro y las estructuras somáticas y viscerales, por lo que si ésta se lesiona, se verían afectadas las funciones motoras, sensitivas y autónomas además de otras discapacidades secundarias (Harvey, 2010).

Esta lesión se da sobretodo en la franja de edad de entre los 15 a los 25 años, siendo poco frecuente a partir de los 45 años. Esta se refiere al daño neurológico de la médula espinal que va a continuación del traumatismo, que ha podido ser causado por un terremoto, accidente del tráfico, en el trabajo, practicando algún deporte o por heridas de bala o puñaladas, entre otras posibles (Harvey, 2010).

Según la zona de la columna vertebral en la que se sufra esta lesión, se la denomina de una manera y las consecuencias serán diferentes. Así el 55% de las ocasiones, la lesión se produce en la zona cervical, la llamada tetraplejía, la cual afecta a las cuatro extremidades. Si la lesión se produce en la zona lumbar, torácica o sacra, será una paraplejía, la cual afecta sólo a las extremidades inferiores (Harvey, 2010).

2.3.1. Recuperación

Cuando se produce una lesión medular, los esfuerzos se centran en minimizar el daño en la médula espinal, gestionar las discapacidades asociadas y optimizar la recuperación neurológica (Harvey, 2010). El proceso que se sigue en pacientes los

cuales nos han sufrido inestabilidad vertebral o lesión es diferente al que siguen los que sí, ya que a los del primer grupo son movilizados los primeros días de la lesión, sin embargo, con los segundos hay que actuar de diferente forma.

Para éstos, el proceso de recuperación empieza por tener inmovilizada la zona de la columna vertebral afectada entre 6 y 12 semanas, ya sea con fuertes protecciones en una silla de ruedas (para permitirles movimiento) o en una cama. Las terapias de recuperación se hacen con un elevado control y supervisión médicas, para no mover nada de la zona de la lesión, y el tiempo de duración de éstas varía de un hospital a otro (Harvey, 2010). Cuando se considera que la columna está estable, se pasa a la movilización a silla de ruedas.

Otra forma mayoritaria de recuperación es la de acudir en primer lugar a la cirugía, tras la cual se procede más rápidamente a la movilización del paciente que en el caso anterior (Harvey, 2010). Esto es beneficioso también debido a que se evitan problemas que vienen derivados de la inmovilización como complicaciones respiratorias, úlceras, debilidad por desuso y contracturas; aunque conlleva el riesgo de la anestesia.

Para hacernos una idea del tiempo que se gasta en la recuperación, en la *tabla 2.1* vemos un estudio realizado a 80 personas con lesiones medulares en la Cínica de Afecciones Raquimedulares, Neuromusculares y Esclerosis Múltiple, del Centro Internacional de Restauración Neurológica (CIREN) en la Ciudad de la Habana, Cuba, en 2002. Como podemos apreciar, es pequeño el porcentaje de personas que tienen una pronta recuperación, si no que los dos mayores porcentajes los encontramos en los pacientes que emplean entre 1 y 3 años o más de 3 años.

Tiempo de evolución.	No. Pacientes.	
	No	%
1 mes	2	2.5
1-3 meses	1	1.25
3-6 meses	11	13.75
6 meses-1 año	15	18.75
1-3 años	26	32.5
+ 3 años	25	31.25
Total	80	100

Tabla 2.1: Tiempo de recuperación en personas con lesiones medulares

2.4. Ataxias

Las ataxias cerebelosas crónicas (ACC) provocan un desequilibrio lento y progresivo de la marcha y la incoordinación de los movimientos de los miembros. La mayoría de ellas son hereditarias, por los que se las suele conocer como ataxias hereditarias, y provocan le degeneración del cerebelo o de las vías cerebelosas aferentes y eferentes (Kelley, 1992).

Éstas aparecen en muchas ocasiones antes de los 20 años y cuanto antes aparezcan, de manera más rápida suelen progresar y llevar al paciente a una silla

de ruedas (Kelley, 1992). La ataxia de Friedeich es la que suele presentarse con más frecuencia antes de los 25 años y la mayoría de estos pacientes fallecen en la tercera o cuarta década de su vida.

2.4.1. Clasificación

Hay varias clasificaciones de las ataxias, pero la más aceptada es la de Harding, que las separó por primera vez en 1983 en las de inicio temprano (antes de los 20 años) y las de inicio tardío (Jiménez, 2003). Esta se basa en los siguientes tipos:

I. Trastornos congénitos de causa desconocida.
II. Trastornos atáxicos con causa metabólica desconocida.
    a. Trastornos metabólicos: debidos a la deficiencia de alguna enzima metabólica.
    b. Trastornos con déficit en la reparación del DNA.
III. Trastornos atáxicos de etiología desconocida.
    a. Ataxias cerebelosas de inicio precoz (antes de los 20 años):
        i. Ataxia de Friedreich.
        ii. Ataxia cerebelosa con reflejos conservados.
        iii. Otras formas raras.
    b. Ataxias cerebelosas de inicio tardío (después de los 20 años):
        i. ACAD tipo I: Ataxias autosómicas dominantes con atrofia óptica, oftalmoplegia, demencia y signos extrapiramidales.
        ii. ACAD tipo II: Ataxias autosómicas dominantes con degeneración retiniana, oftalmoplegia y con o sin signos extrapiramidales.
        iii. ACAD tipo III: Ataxia autosómica dominante pura.
        iv. ACAD tipo IV: Ataxia cerebelosa dominante con temblor esencial y ataxia autosómica dominante periódica.

Por el momento no existe ningún tratamiento efectivo para las ataxias. Según el análisis genético se puede adivinar el origen de la enfermedad y así proporcionar el tratamiento más adecuado (Jiménez, 2003).

2.5. Esclerosis múltiple

La esclerosis múltiple es una enfermedad desmielinizinante del sistema nervioso central con la cual surgen lesiones inflamatorias que producen la destrucción de la mielina (Terré-Boliart, Orient-López, 2007). Es la enfermedad neurológica más frecuente y de las principales causas de invalidez en adultos jóvenes de Europa y Norteamérica.

Es una enfermedad que se presenta de forma heterogénea, ya que en el 80-90% de los pacientes presenta una fase inicial remitente-recidivante y a partir de los 10 o 15 años de evolución, la mitad de personas pasan a un curso secundariamente progresivo (Blanco, Saiz, Graus, 2005). Además, entre un 10 y un 20% de los pacientes no mantiene secuelas importantes 15 años más tarde de la esclerosis, pero hay entre un 1 y un 3% en los cuales al poco tiempo ya tienen desarrollada una gran discapacidad.

Un gran problema es que no hay un tratamiento claramente efectivo para este trastorno, pero se ha demostrado el papel del sistema inmune en la patogenia de la esclerosis múltiple, por lo que el tratamiento más utilizado son los fármacos inmunodepresores e inmunomoduladores, los cuales reducen el número de brotes, la carga lesional en la resonancia y la progresión de la enfermedad a largo plazo (Blanco, Saiz, Graus, 2005).

2.5.1. Síntomas

Los síntomas de la esclerosis múltiple, al igual que todos los relacionados con lesiones en la médula espinal, son de carácter urinario (Micheli, 2003). Así, el síntoma urinario de consulta más común en pacientes con esclerosis múltiple es la urgencia miccional, que es provocada por las dificultades motoras que los pacientes suelen padecer.

Esta disfunción urinaria progresa de forma paralela con el deterioro neurológico, pero esto rara vez provoca daño en el tracto urinario superior.

2.5.2. Tratamiento de rehabilitación

La esclerosis múltiple produce un complejo patrón de discapacidad que suele ser progresivo, por lo que el tratamiento se basará en integrar los objetivos terapéuticos de mejorar los episodios agudos, frenar la progresión de la enfermedad y el tratamiento de los síntomas y complicaciones (Terré-Boliart, Orient-López, 2007).

Debido a que, como ya dijimos anteriormente, la esclerosis múltiple es una enfermedad heterogénea, el tratamiento será personalizado para cada paciente y será llevado por un equipo multidisciplinar y en el cual se tengan que efectuar diferentes intervenciones terapéuticas según como vaya evolucionando la enfermedad (Terré-Boliart, Orient-López, 2007).

Por lo tanto, la estrategia de tratamiento común que se puede seguir es la de prevenir déficits secundarios, entrenar nuevas habilidades para que el paciente potencie los sistemas sanos y la capacidad funcional de los afectados, compensar con ayudas técnicas funciones que no se pueden reeducar, instruir a una tercera persona para que realice por el paciente las actividades que éste ya no pueda, modificar el entorno social y laboral y técnicas psicológicas para el paciente, familia y cuidadores (Terré-Boliart, Orient-López, 2007).

Pero previo a este tratamiento rehabilitador, lo primero es efectuar una evaluación del déficit, discapacidad y minusvalía del paciente que se puede estimar según la clasificación elaborada por la Organización Mundial de la Salud.

2.6. Patologías infantiles

Las patologías infantiles son abundantes y muy diversas, y no sólo se tratan de trastornos físicos o motores. También encontramos trastornos psicológicos, en el habla, en ciertos comportamientos, etc… Por lo tanto hablaremos de todas estas

patologías posibles y para cada una de las cuales habrá algún tipo de terapia o tratamiento, ya sea para el que las padezca o incluso para su entorno, ya que a veces éstas pueden surgir por las alteraciones en el ritmo de vida del niño, ciertos comportamientos externos o quizá tomar como patología algo que no tiene mayor importancia (Palazón, 1998).

La clasificación de éstas se hace según las más frecuentes a las distintas edades, y así las presentamos a continuación (Palazón, 1998):

-En torno a los 12 meses de edad las patologías son de tipo funcionales (no come, no duerme…) o cuadros psicopatológicos de la depresión, trastornos del desarrollo o retraso mental. Las consultas más frecuentes suelen ser por problemas con la alimentación, trastornos en el sueño (por molestias físicas o autismo) y problemas en el desarrollo psicomotor respecto a la edad.

-Sobre los 24 meses, además de los problemas anteriores, se añaden problemas de ansiedad y conductas disruptivas, y la mayor cantidad de consultas son por trastornos alimentarios y de sueño, como en el caso anterior, y además en la conducta, que pueden ser por trastornos en el desarrollo, y en el ámbito social al sentir ansiedad por las nuevas experiencias.

-Ya por los 4 años, de nuevo incluyendo los trastornos anteriores, se añadirían los de ansiedad y conductas perturbadoras. Los problemas más frecuentes en esta edad son también relacionados con la alimentación, el sueño, la conducta y en el ámbito social, como en el caso anterior, además de problemas en el control de esfínteres que pueden ser síntoma de ansiedad o depresión, y en el ámbito familiar con fobias relacionadas con la oscuridad o la soledad.

-Finalmente, a partir de los 11 años hay muchos más trastornos posibles, pero se destacan las conductas problemáticas que puedan surgir bruscamente.

Capítulo 3. Revisión de tecnologías

En este capítulo desarrollaremos la información acerca de las tecnologías que se han utilizado en el desarrollo del proyecto, las tecnologías similares a las finalmente utilizadas y explicaremos porqué se han elegido esas y no alguna de sus competidoras.

La primera parte tratará de introducirnos al mundo de los videojuegos contando una breve historia del inicio y evolución de ellos. A continuación, pasaremos a hablar de poderosos motores de juego que nos permiten el desarrollo de videojuegos para diferentes plataformas. Después comentaremos los programas de modelado y animación 3D más adecuados para nuestra finalidad. Y para terminar, describiremos los dispositivos de captura de movimiento que se pueden usar y, en el apartado final, los *plugins* que permiten que este dispositivo funcione con nuestro equipo.

3.1. Historia de los videojuegos

El invento de estos podría remontarse a los años 40, en concreto a cuando los americanos desarrollaron un simulador de vuelo para el entrenamiento de pilotos, pero no fue hasta el 1962 cuando éstos experimentaron un ascenso continuado, de la mano de la tercera generación de ordenadores que traía reducción de tamaño y costes (Etxeberria, 2009).

Poco más adelante, en 1969 se creó el microprocesador y a partir de ese momento comenzó a constituir el corazón de los aparatos electrónicos y produciendo grandes cantidades de información en un espacio mucho más reducido que el que se había ocupado anteriormente (Etxeberria, 2009).

Ya en los años 70, apareció el disco flexible y la empresa *Atari* lanzó al mercado el que se considera por los expertos como el primer videojuego de la historia, el PONG, que consistía en una partida muy simple de tenis (Gil y Vida, 2007). Éste se vendía con una terminal o máquina recreativa. Y fue a través de estas máquinas recreativas como los videojuegos de la época como el *Space Invaders* o *Asteroids*, se hicieron populares y generaron las primeras preocupaciones sobre cómo afectarían los videojuegos a la conducta de los niños (Etxeberria, 2009). También en esta época comenzaron a surgir las primeras consolas, como la *Magnavox Odyssey*, y nuevos géneros de videojuegos, hasta el punto de que a mediados de la década, más de 20 compañías desarrollaban estos juegos.

Más adelante en los 80, encontramos el considerado *boom* de los videojuegos (Gil y Vida, 2007). Las mejoras que aparecieron en las máquinas recreativas, consolas y ordenadores personales, tanto en potencia de los microprocesadores como en memoria, permitieron la presentación de juegos de gran calidad en movimiento, color y sonido, impensables años atrás (Etxeberria, 2009). Entre los juegos más destacables de esta época encontramos el *PacMan*, *Mario Bros* y el *Tetris*.

Ya en los 90, se extendieron masivamente los videojuegos gracias a que hubo aun más mejoras y a la incorporación del CD-ROM y de entornos en 3 dimensiones. Fueron *Nintendo* y *Sega* las que desde un principio hicieron videojuegos de forma masiva, aunque esta última, junto con *Atari*, terminaron desapareciendo en esta época. Consolas tan conocidas como la *Nintendo64* y la *PlayStation* fueron creadas en esta década (Etxeberria, 2009) (Gil y Vida, 2007).

El principio siglo XXI comenzó con el dominio de las consolas *PlayStation* de *Sony*, *Xbox* de Microsoft® y *GameCube* de Nintendo, pero según se va avanzando en el siglo, van aumentado el número de plataformas para las que los videojuegos están disponibles (teléfonos móviles o *Smartphones*, Internet, nuevas consolas portátiles…) y se amplia el rango de usuarios que se quieren captar (Gil y Vida, 2007). Así pues, los videojuegos se han convertido en una de las industrias más poderosas del mercado audiovisual.

3.1.1. Clasificación de los videojuegos

Es enorme la variedad de videojuegos existentes hoy en día, por lo tanto se hace casi obligatoria una clasificación de todos ellos. Sin embargo, debido a esta gran abundancia no hay una sola clasificación, si no que investigando en diferentes fuentes encontramos las muchas clasificaciones existentes.

La elegida para este proyecto es la realizada por la Fundación de Ayuda contra la Drogadicción (2002), ya que viene junto con la *figura 3.1* que muestra el porcentaje de uso de estos diferentes tipos de videojuegos según un estudio que ellos realizaron.

Así encontramos estos diferentes tipos de videojuegos (Instituto de la mujer/CIDE, 2004):

- Juegos de plataforma: Gracias a nuestra precisión y destreza se van pasando diferentes plataformas (p.e., Pokemon, Super Mario Bros…).

- Simuladores: Ejemplifican situaciones realistas, como controlar vehículos (p.e., GT2, Driver…).

- De práctica de algún deporte: Consisten en jugar a algún deporte, pero sin que esto sea una simulación (p.e., FIFA, Snowboard Supercross...).

- De estrategia deportiva: Son de temática deportiva como los anteriores, pero resaltando el aspecto de la estrategia (p.e., PC Futbol, Manager…).

- De estrategia no deportiva: Basados en controlar y planificar situaciones (p.e., Sims, Age of Empires…).

- De disparos: En los que hay que alcanzar diferentes objetivos en movimiento disparando con un arma (p.e., Quake, Halflife…).

- Lucha: Los cuales consisten en peleas entre personajes, ya sea cuerpo a cuerpo o con armas (p.e., Tekken, Mortal Kombat…).

- Aventura gráfica: Se recrea una aventura con diferentes pruebas y personajes (p.e., Lara Croft, La fuga de Monkey Island…).

- Rol: En los que el usuario es un personaje que tiene que superar aventuras o combates (p.e., Final Fantasy, Diablo…).

Y después de describir en qué consiste cada uno, mostramos los porcentajes de uso de estos videojuegos por los usuarios, viendo que destacan los que se tratan de practicar deporte en detrimento de los videojuegos de rol, que están menos extendidos.

Figura 3.1: Porcentaje de uso de los diferentes tipos de videojuegos

Por otro lado, se encuentran clasificaciones más generales, como la que muestra la *figura 3.2*, en la cual se agrupan en la categoría de "arcade" los videojuegos deportivos, de lucha, de disparos y en general los más tradicionales de ordenador. Además, vemos agrupados todos los juegos de estrategia, que son los que tienen en mayor porcentaje de usuarios (aunque seguidos muy de cerca por los arcade). Por otro lado tenemos los juegos de simulación, con un porcentaje de

utilización algo menor que los arcade y los de estrategia, y finalmente los videojuegos basados en juegos de mesa, con una utilización bastante baja.

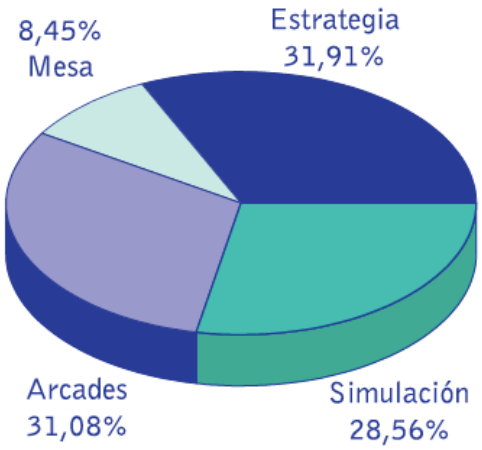


Figura 3.2: Porcentaje más global de uso de videojuegos

3.1.2. Tipos de jugadores

Cada persona que juega a videojuegos lo hace por diferentes motivos, juega a cosas distintas, a diferentes horas, etc. Así, existe una gran variedad de tipos de usuarios, entre las que destaca la diferencia entre los *hardcore gamers* y los *casual gamers*, en donde para los primeros la actividad principal de ocio son los videojuegos y para los segundos, los videojuegos son un complemento en su tiempo de ocio y no les importa dejar a medias una partida o desafíos (Gil y Vida, 2007).

Pero no toda la población juega a videojuegos, así que no todo el mundo entra en esta clasificación. Aunque según un estudio realizado por la FAD en el 2002, más de la mitad de adolescentes en España juegan a videojuegos y otro gran porcentaje afirma que jugaba antes o al menos los ha probado, como se muestra desglosado en la *figura 3.3* que vemos a continuación.

Figura 3.3: Porcentajes de la frecuencia de juego y edad media de inicio

Además de esto, en la *tabla 3.1* vemos como por lo general, las personas que juegan a videojuegos lo hacen de una manera más o menos constante, de manera

que la mayoría lo hacen con un ritmo de una o dos veces por semana o tres o más días.

RARA VEZ	3.1
ALGUNA VEZ AL MES	19.8
UNO O DOS DÍAS POR SEMANA	34.7
TRES O MÁS DÍAS POR SEMANA	22.8
TODOS LOS DÍAS	19.6

Tabla 3.1: Porcentajes de la frecuencia de juego de usuarios de videojuegos

De todos estos jugadores que hay, encontramos una clasificación más detallada según el estudio de *Electronic Gaming in the Digital Home* (FAD, 2002). En ella se diferencian los *Power gamers*, los cuales son jugadores avanzados y representan el 11% del mercado, los *Social gamers*, que utilizan los videojuegos como medio para interactuar con otras personas, los *Leisure gamers*, que juegan en su tiempo de ocio y se interesan por las novedades relacionadas con los videojuegos, los *Dormant gamers*, para los cuales los juegos tienen que representar un reto, los *Incidental gamers*, que simplemente juegan por aburrimiento, y los *Occasional gamers*, que suelen jugas a juegos similares a los de mesa.

Por lo tanto, puede decirse que los videojuegos son un aspecto muy presente en la vida de las personas y sobretodo de la juventud y algo muy influyente, por lo que podría ser un muy buen medio para guiar a los jóvenes y no dejarlo sólo como un método de diversión. Así pues habría que tener en cuenta los factores que más llaman la atención de los jóvenes (*figura 3.4*) para poder crear cosas que realmente les atraigan y utilizarlo para su propio beneficio.

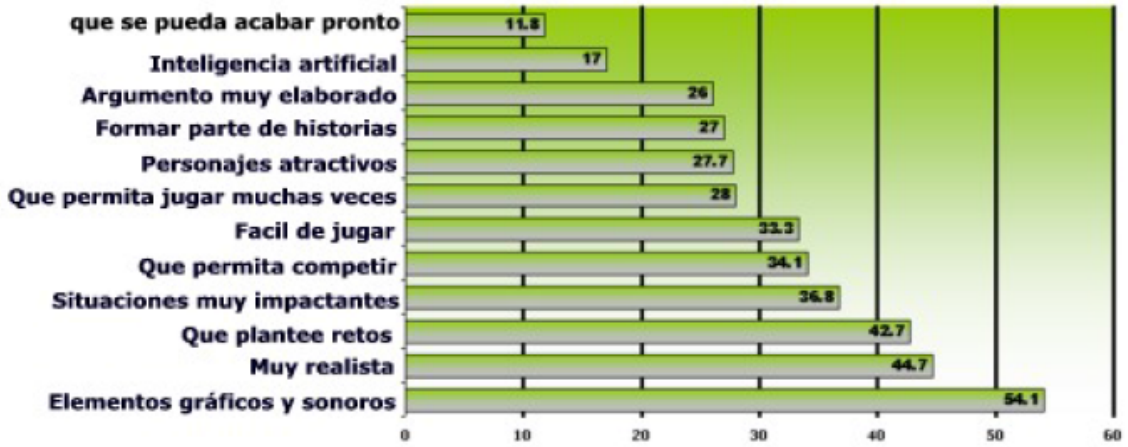


Figura 3.4: Porcentajes de factores que más gustan a los usuarios de videojuegos

3.1.3. Áreas de uso de los videojuegos

En este apartado nos metemos ya de lleno en lo que son los videojuegos serios, que es en lo que se basa nuestro proyecto. Éstos son una serie de juegos que aprovechan la penetración de los videojuegos en la sociedad para obtener beneficios para el jugador y servir como método educativo o de formación (Marcano, 2008).

Estos están destinados para la educación, entrenamiento de habilidades y comprensión; están vinculados con algún aspecto de la realidad y así poder recrear escenarios propicios para desempeñar un rol concreto; permiten que los aprendices puedan realizar prácticas seguras en un entorno bastante real si se lleva a cabo con tecnología como la 3D; y suelen encontrarse intereses manifiestos en sus contenidos, de índole político, religioso, psicológico, etc. (Marcano, 2008).

El problema que hay con los juegos serios surge a la hora de hacer que estos sean divertidos, y de ellos mayormente va a depender que el jugador juegue más o menos a éste o que lo cambie por otro (Marcano, 2008). Por lo tanto es muy importante incluir este factor de diversión, ya sea en un videojuego serio para un entrenamiento a piloto como para otro que sirva para la propia rehabilitación, y así conseguir la acción voluntaria de jugar del usuario.

Entre las áreas de uso de estos juegos serios, encontramos las siguientes (Marcano, 2008):

- Militar: Esta área ha sido pionera en el uso de videojuegos y simuladores para el entrenamiento de ciertas habilidades y en lanzar al mercado juegos basados en este aspecto. Estudios han demostrado que realmente resultan efectivos, siendo los videojugadores los que mejor capacidad de reacción, manejo y desenvolvimiento en situaciones de guerra poseían.

- Política: Muchos entes gubernamentales, e incluso no gubernamentales, han utilizado los videojuegos para entrenar, informar y persuadir a la gente, así como para que el jugador representara a dirigentes políticos de la oposición o que generasen debates.

- Empresarial o corporativo: También las empresas y corporaciones privadas se aprovechan de los videojuegos para el entrenamiento o para publicitarse. Es una forma de llegar a mucha gente de forma efectiva y a bajo coste. Además, utilizan estos juegos como forma de "entrenar" a los trabajadores para que tengan visión de futuro, capacidad de tomar riesgos y obtener beneficios.

- Educación: Cada vez son más los videojuegos serios de este tipo y son muchas las instituciones y universidades que colaboran para la implementación y evaluación de los resultados didácticos de estos.

- Religión: Actualmente los videojuegos de esta área se consideran como un medio efectivo de transmitir las lecciones y el contenido bíblico.

- Arte: Aquí han sido desarrollados videojuegos que estimulan la creatividad, simulan posibilidades de diseño o permiten crear nuevas melodías.

- Salud: Aparte de los usos para la práctica del personal paramédico y como medio para superar el estrés postraumático y ansiedad, entre otras cosas, el uso que más nos interesa es el del terreno de la rehabilitación para la recuperación de habilidades motoras o cognitivas de los pacientes y de una forma distractora.

3.2. Motores de juego o *Game Engines*

Los juegos y las aplicaciones gráficas interactivas en tiempo real en general, están viviendo uno de los momentos más importantes de su historia, debido a que el volumen de la industria del videojuego ha sobrepasado a otras tan importantes como el cine, generando un ecosistema de empresas en todo el mundo (Linares, Martínez, Candela, 2012).

Todo esto viene de la mano con la "reciente" aparición de los dispositivos móviles iOS, Android o Windows Phone, que han multiplicado este gran mercado haciendo que casi cualquier persona sea un cliente (Linares, Martínez, Candela, 2012).

Por lo tanto, el desarrollo de aplicaciones gráficas que exigen altas prestaciones, como son los videojuegos, necesita herramientas de desarrollo especializadas y más o menos complejas (Linares, Martínez, Candela, 2012). Así pues, lo que deben de hacer principalmente es:

- Obtención del máximo rendimiento gráfico de los dispositivos sobre los que se ejecuta el videojuego o la aplicación.

- El aprovechamiento de las características más avanzadas de las unidades de procesamiento gráfico que permita los mejores resultados visuales y de realismo.

- Ofrecer un conjunto de herramientas que permitan a los desarrolladores aprovechar las características anteriores con el menor esfuerzo posible, con un potente editor, un poderoso lenguaje de programación, y con una amplia gama de complementos como la importación de modelos, imágenes, sonidos y otros medios, motor de físicas, edición de terrenos, sistemas de partículas, herramientas de inteligencia artificial, etc.

- Multiplataformidad tanto en las herramientas de desarrollo como en su capacidad de exportar la aplicación final a diversas plataformas: escritorio, web, dispositivos móviles y consolas.

Ante este complejo escenario, el desarrollador que está interesado en generar aplicaciones gráficas en tiempo real y más específicamente videojuegos, debe elegir convenientemente las herramientas más adecuadas, con características muy dispares e incluso costes muy variados.

Así pues, aparecieron los motores de juegos prácticamente desde el mismo momento en el que empezó el desarrollo de juegos. Este concepto surgió a mediados de los 90 con la aparición del famosísimo juego de acción en primera persona *Doom*, desarrollado por la compañía *id Software* bajo la dirección de John Carmack (Vallejo, Martín, 2012). La definición de motor de juego es el conjunto de herramientas software que permitan asistir a los desarrolladores en la creación de juegos y aplicaciones gráficas interactivas en tiempo real (Linares, Martínez, Candela, 2012).

Encontramos gran variedad de estos motores, por lo que describiremos algunos de ellos que se han tomado como los más importantes debido a la importancia que tienen en el mercado por su nivel de penetración y por el número de desarrolladores que lo utilizan, la potencia de las herramientas que ofrecen a los desarrolladores y las plataformas sobre las que estos desarrolladores pueden distribuir sus productos al utilizar dicho motor.

3.2.1. Unity3D

Unity3D (logo en la *figura 3.5*), según su página oficial (2014), es un ecosistema de desarrollo de juegos, es decir, un poderoso motor de renderizado totalmente integrado con un conjunto completo de herramientas intuitivas y flujos de trabajo rápido para crear contenido 3D interactivo, publicación multiplataforma sencilla, miles de activos de calidad, listos para usar en la "Tienda de Activos" (la *Assets Store*) y una Comunidad donde se intercambian conocimientos.

Figura 3.5: Logo de Unity3D

Éste puede ser usado tanto por empresas como por desarrolladores independientes. En concreto para estos segundos es una herramienta maravillosa a la hora de reducir el tiempo de desarrollo del juego y los costes de éste, ya que Unity3D está disponible de manera gratuita, aunque con leves limitaciones, pero aun así ofreciendo grandes resultados. Igualmente, la versión de "Unity3D PRO 5.x"

se obtendría por $1.500 o por $75 al mes (a Octubre de 2014, ya que estas cantidades podrían variar). También encontramos gran cantidad de versiones "PRO" para iOs, Android, Windows Phone, Blackberry, etc.

Unity3D se usa para la creación de contenido 2D y 3D multiplataforma, de manera sencilla, y es un motor de juego el cual, aunque su desarrollo empezó en el 2001, fue creado por la compañía Unity Technologies en el 2004. Esta empresa fue fundada por David Helgason (CEO), Nicholas Francis (CCO) y Joachim Ante (CTO) en Copenhague, Dinamarca, después de su primer juego, CooBall, el cual no tuvo ningún éxito. No obstante, los tres reconocieron que el motor y las herramientas de desarrollo tenían un gran potencial, por lo que decidieron crear el motor que pudiese ser usado por quien quisiera. Esta empresa tiene actualmente un gran éxito, y ha recibido financiación de la talla de Sequoia Capital, Capital WestSummit y Socios iGlobe (Rowland, 2014).

Su notable éxito viene dado por la cantidad de desarrolladores independientes y sin apenas recursos que hacen uso de esta plataforma, gracias a que en el 2009 Unity3D se empezó a ofrecer de forma gratuita, justo un año después del boom que tuvo el iPhone, ya que se convirtió en uno de los primeros desarrolladores de motores en empezar a apoyar a la plataforma. Y así fue desde el principio, ya que en el 2005 Unity3D lanzó su primera versión en la *WorldWide Developers Conference* de Apple (Unity Technologies, 2014a).

Desde el momento de su presentación en la *WWDC* de Apple, nuestro motor de juego no hizo más que crecer, ya que cada vez permitía realizar aplicaciones gráficas para más plataformas, se iba expandiendo por países de todo el mundo abriendo oficinas y cada vez conseguía más inversiones de empresas como las mencionadas anteriormente.

A día de hoy, Unity3D ha sacado su versión 5, ha ganado en los *Develop Awards 2014* el premio al mejor motor de juego y permite realizar juegos para las siguientes plataformas:

- iOS
- Android
- Windows Phone 8
- Blackberry 10
- Windows
- Aplicaciones de la Windows Store
- Mac
- Linux
- Reproductor web
- PlayStation 3, 4, VITA y Mobile
- Xbox 360 y ONE

- Wii U
- Tizen

A la hora del diseño gráfico de los objetos que van a formar parte del videojuego, este motor de juego nos permite la importación de gran cantidad de elementos desde diferentes programas de diseño como son Blender, Cinema 4D, 3DS Max, Cheetah3D, Modo, Maya, Softimage, ZBrush, Adobe Photoshop, Adobe Fireworks y Allegorithmic Substance.

A nivel de motor gráfico, Unity3D hace uso de Direct3D (Windows), OpenGL (Mac, Windows), OpenGL ES (Android, iOS) y APIs propietarias (Wii).

Para desarrollar y depurar el código fuente del juego (los *Scripts*), desde la versión 3.0 encontramos por defecto el entorno de desarrollo MonoDevelop. Además, Unity3D es muy flexible a la hora de dejarnos elegir el lenguaje de programación a utilizar, ya que nos da a elegir entre C#, JavaScript y Boo (éste tercero el menos utilizado).

Podemos afirmar que la página oficial de Unity3D es una de las más completas en cuanto a las páginas de motores de juego, ya que contiene numerosa documentación, muy completa y muy bien estructurada, por lo que resulta sencillo aprender por nuestra cuenta a manejarnos con la programación y el entorno. Si con esta documentación no somos capaces de realizar algunas tareas, aún así podemos encontrar infinidad de tutoriales y videotutoriales, aparte de un enorme foro en el que podemos consultar dudas más concretas.

Es en esta misma página donde también encontramos que Unity3D es de lejos el software de desarrollo de juegos dominante a nivel mundial. Más juegos se crean con Unity3D que con cualquier otra tecnología para juegos, más jugadores juegan juegos creados con Unity3D, y más desarrolladores confían en sus herramientas y servicios para hacer crecer su negocio. Así encontramos que posee un 45% de participación en el mercado, un 47% de desarrolladores frente a sus competidores, 3,3 millones de desarrolladores registrados, 600 millones de jugadores y con él se realizan entre un 70 y un 50% de los juegos para móviles de países tan importantes como China, Japón, Corea y Estados Unidos.

Entre la gran variedad de la que hemos hablado de juegos hechos con Unity3D, destacamos Temple Run (juego muy adictivo de los últimos tiempo que vemos en la *figura 3.6*), Among the Sleep, Game of Thrones: Seven Kingdoms, Elite Squad, War Arm, No Heroes, Line of Fire, Rust, Endless Space o Prime World.

Figura 3.6: Imagen del Temple Run: juego hecho con Unity3D

3.2.2. Unreal Engine

El siguiente motor de juego más importante que encontramos es Unreal Engine (logo en la *figura 3.7*), que es un profesional conjunto de herramientas y tecnologías usados para construir videojuegos en alta calidad a través de un amplio rango de plataformas. La arquitectura de renderizado de este motor permite a los desarrolladores lograr impresionantes efectos visuales y también escalar con elegancia a los sistemas de gama baja (Unreal Engine, 2014).

Las nuevas y revolucionarias características de flujo de trabajo y su conjunto profundo de herramientas permiten a los desarrolladores implementar sus ideas rápidamente y ver resultados inmediatos, mientras que el acceso completo al código fuente C ++ trae la experiencia a un nivel completamente nuevo.

La tecnología de Unreal Engine ha permitido hacer cientos de juegos, así como películas 3D en tiempo real, simulaciones de entrenamiento, visualizaciones y más. En los últimos 15 años, miles de personas y equipos y han construido carreras y empresas sobre las habilidades desarrolladas utilizando el motor.

UNREAL
ENGINE

Figura 3.7: Logo de Unreal Engine

Este motor de juego fue creado por la compañía Epic Games y fue implementado inicialmente en el *shooter* en primera persona Unreal, en el año 1998. Tras esto fue la base de juegos como Unreal Tournament, Tom Clancy's Rainbow Six, Gears of War o la saga de BioShock, y desde el 2004 hasta ahora ha recibido prácticamente todos los años el premio de "Mejor Motor de Juego" por la revista "Game Developer" y en el 2013 por parte de la "Excelencia de la Industria del Desarrollo".

Con este motor se pueden realizar videojuegos o aplicaciones con el lenguaje C++, aunque es más conocido como Unreal Script por sus particularidades, y para varias plataformas que son:

- iOs
- Android
- HTML5
- Linux
- Mac
- Oculus Rift
- PlayStation 4 y Morpheus
- SteamOS
- Windows
- Xbox One

Actualmente se distribuye la versión 4 de Unreal Engine por 19€ al mes, o un 5% de las ganancias obtenidas con el juego desarrollado con el motor, aunque puede conseguirse gratis para uso académico. Para esta versión se requiere al menos la versión de Windows 7 de 64 bits o de Mac OS X 10.9.2, procesador Intel Dual-Core o AMD de 2.5 GHz o más rápido, GPU compatible con DirectX 11 y 8 GB de RAM.

Además ya está disponible una actualización a la versión 4.1 que incorpora soporte para SteamOS y Linux, flujo de trabajo refinado para iOs y Android, nuevas plantillas de juego para programadores y diseñadores C ++ usando Blueprint y más de 100 mejoras basadas en los comentarios de la comunidad.

Unreal Engine es considerado uno de los motores de juego más potente que hay. Pero su *boom* vino a raíz del videojuego Gears of War, hecho por la versión 2 de este motor de juego con los gráficos que vemos en la *figura 3.8*, y el cual salió en el 2006 al mercado y en sus dos primeras semanas a la venta vendió más de un millón de copias. Consiguió más de 30 nominaciones, entre las que destacan las de "Juego del año 2007" y "Mejor juego para Xbox 360", así como el de "Mejor juego internacional de acción", ganado en los premios 1UP WON la cual se presentaron el 18 de octubre de 2007. Tal fue su éxito que en los años 2008 y 2011 salieron Gears of War 2 y 3, respectivamente. Finalmente, en el presente año Microsoft® adquirió los derechos de este videojuego (Microsoft®, 2014b).

Figura 3.8: Imagen del Gears of War: juego hecho con Unreal Engine

Aparte de estos juegos, con Unreal Engine se han hecho otros tan conocidos como Spider-man 2, Star Wars: Republic Commando, America´s Army, la saga de Tom Clancy´s, Medal of Honor… Por empresas desarrolladoras y por distribuidoras tan importantes como Atari, Ubisoft, Electronics Arts, Microsoft® Game Studios o Activision, entre otras.

3.2.3. Corona SDK

Entre los motores de juego más importantes, nos encontramos también con Corona SDK (logo en la *figura 3.9*). Éste es un motor de juego mucho más sencillo que los anteriores ya que sólo permite videojuegos y aplicaciones en 2D. Es 10 veces más rápido que el resto de motores debido a que su código es mucho más simple, pudiendo realizar en un par de líneas de *script* lo que otros lenguajes hacen en muchísimas más (Corona Labs, 2014).

Figura 3.9: Logo de Corona SDK

Corona SDK es un producto de Corona Labs, originalmente creado por Walter Luh y Carlos Icaza, ambos empleados de Adobe. Su primer lanzamiento fue en Diciembre del 2009 y era soportado solamente por una única plataforma: iOs. Por los tanto sólo era soportado por el iPhone hasta que el iPod Touch y el iPad llegaron al mercado (Zammetti, 2013).

En Abril de 2010 llegó la versión 2.0 de Corona SDK, y fue a partir de aquí cuando este motor cogió impulso al volverse multiplataforma, ya que además de

para los dispositivos que ya estaba disponible, comenzó a ser soportado por el sistema operativo Android.

Otra fecha importante a destacar es Enero de 2011, ya que ahora también se podía desarrollar en Windows y no sólo en plataformas Unix y MacOS como hasta el momento. Y más adelante, en Abril de ese mismo año, se introdujo otra nueva plataforma para la cual Corona SDK podía desarrollar material: NOOK. Por lo tanto, hoy en día podemos realizar juegos con este motor para iOs, Android, NOOK y además para Kindle y próximamente para Windows Phone.

El lenguaje de programación que utiliza Corona SDK se denomina Lua, que es un lenguaje en la capa alta de C ++ y OpenGL. Éste utiliza un motor de renderizado propiedad de OpenGL ES que provee a las aplicaciones escritos con él aceleración *hardware* automática. Por lo que, como mencionamos anteriormente, los juegos construidos con Corona SDK serán más rápidos en la mayoría de los dispositivos.

Este motor también nos permite incorporar a nuestro juego OpenGL, OpenAL, Box2D, Facebook, SQLite y Game Center entre otras.

Para instalar Corona SDK, necesitamos un MacOS 10.7 o superior con un sistema basado en Intel, una arquitectura de 64 bits y OpenGL 1.4 o superior; o un Windows XP o superior con un procesador de al menos 1 GHz, al menos 1 GB de RAM y una versión de OpenGL igual o mayor de la 1.3 (Williams, 2013). Esto nos permite desarrollar aplicaciones para iOs 4.5 o superior y para Android 2.2 o superior.

Corona SDK no tiene versión gratuita, aunque ofrece descuentos para fines educacionales. La versión básica puede obtenerse por 16$ al mes, la PRO por 49$ al mes, y la de empresas puede ser para pequeñas empresas por 79$ o la ilimitada por 199$, todas estas cantidades al mes (Corona Labs, 2014).

También vemos que su web oficial trae una documentación muy completa con más de 1000 APIs, *plugins*, guías y tutoriales.

Como podemos ver después de todo lo descrito anteriormente, con Corona SDK se realizarán básicamente juegos para *smartphones*, entre los que reinan Fun Run, Bubble Ball, Blast Monkeys, Major Magnet, The Lost City, Freeze! o Streetfood Tycoon. Entre ellos destacamos la curiosidad de que Bubble Ball (mostrado en la *figura 3.10*) fue hecho por un niño de 14 años y ha sido uno de los juegos más populares del momento, formado por diferentes puzles basados en físicas y teniendo que conseguir que pequeños monstruos lleguen a su meta.

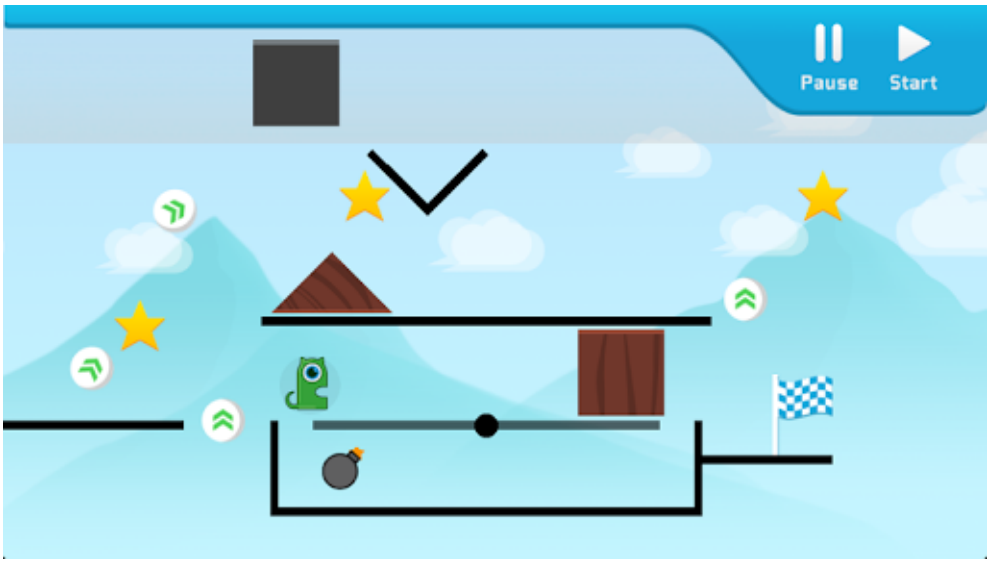


Figura 3.10: Imagen del Bubble Ball: juego hecho con Corona SDK

3.2.4. Shiva3D

El motor de juego ShiVa3D, cuyo logo vemos en la *figura 3.11*, es una herramienta para los desarrolladores que les permite crear aplicaciones y videojuegos 2D y 3D de tiempo real. Es muy poderoso y un editor WYSIWYG 3D y servidor MMO (*Massive Multiplier Online*) multiplataforma (ShiVa Technologies, 2014).

Figura 3.11: Logo de Shiva3D

Su editor es un software WYSIWYG para el diseño de juegos y aplicaciones. Éste nos permite importar diseños hechos con 3ds Max, Maya o XSI, o crear con el propio editor sistemas de partículas, senderos, animaciones, HUDs, materiales, terrenos y océanos, entre otras cosas, y aplicarles un comportamiento por medio de los scripts escritos en LUA (que es más sencillo, como vimos anteriormente) o directamente en C ++.

ShiVa3D nos permite realizar proyectos para gran variedad de plataformas. Así podemos realizar juegos para un navegador web, para ordenadores con sistemas operativos Windows, MacOS o Linux, para *smartphones* o *tablets* iOs, Android, BlackBerry, HP WebOS, Bada, Symbian o Marmalade, y para videoconsolas como Wii, xBox 360 y PlayStation 3, Portable o Vita.

Además a este motor de juego se le pueden añadir *plugins* de seguridad ssl como el motor PhysX, la librería de sonidos Fmod, ARToolkit y el motor Scaleform HUD.

Actualmente, se distribuye la versión 1.9.2 de ShiVa3D, la cual incluye las plataformas de Windows Phone 7.5 y 8, la exportación gratis a Adobe Flash 11 de nuestra edición en el editor web y las arquitecturas de 64 bits en Linux, Windows y Mac. También podemos encontrar una Beta de la versión 2.0 que salió en Julio de este mismo año.

ShiVa3D se puede utilizar de forma gratuita gracias a su editor web y permite realizar todo tipo de videojuegos y aplicaciones para sitios web, aplicaciones 3D para Facebook y previsualizaciones de juegos para móviles y consolas. La versión básica se puede adquirir por 400$ y permite a desarrolladores independientes o pequeñas empresas comercializar su aplicación. Otra versión es la avanzada, que es para empresas desarrolladoras más grandes y está disponible por 2000$. Finalmente, hay una versión educacional que es la avanzada pero por el precio de la básica.

Como curiosidad, comentaremos que ShiVa3D fue el primer motor de juego que permitió el desarrollo para la plataforma Android (Agencia Francesa por los Videojuegos, 2014).

Con este motor de juego se han realizado gran cantidad de aplicaciones y videojuegos, entre las que podemos destacar Judge Dee y Voodoo Dice en el campo de las videoconsolas; Non Flying Soldiers, ArchVIZ 3D Interactive, JAN 3D Interactive y Corvin Office (entre otras) en el de las aplicaciones o juegos de escritorio; y Monkey Drum Deluxe, Paper Racing y Block Rockin' en el de los *smartphones*. Pero hay un juego que destaca en estos tres campos: Babel Rising, del cual vemos una captura en la *figura 2.12*, considerado uno de los mejores juegos del 2012.

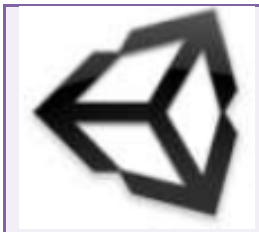
Figura 3.12: Imagen del Babel Rising: juego hecho con ShiVa3D

3.2.5. Comparación de motores de juego

Todos los motores que hemos descrito anteriormente son muy importantes y comúnmente utilizados, por ello a la hora de elegir uno había que realizar una comparativa a fondo, fácil de realizar tras los análisis anteriores.

Para la comparación (*tabla 3.2*) vamos a tener en cuenta:

- Si tiene soporte 2D o 3D.
- El editor y sus herramientas que permitan la edición de escenas en 2D o 3D.
- El lenguaje de programación que se utiliza.
- Plataformas sobre las que se puede distribuir el juego o producto utilizando el motor.
- Motor de físicas.
- Herramientas de Inteligencia Artificial.
- Características avanzadas, como *occlusion culling*, *lightmapping*, *shaders*, editor de terrenos, sistemas de partículas, etc.
- Licencias de las que dispone.

Motor de juego				
2D/3D	2D y 3D	2D y 3D	2D	2D y 3D
IDE	Sí (Mac y Windows)	Sí	No	Sí
Lenguaje	C#, BOO, JavaScript	C++ (Unreal script)	LUA	LUA, C++
Plataformas de distribución	iOS, Android, Windows Phone 8, Blackberry 10, Windows, Windows Store, Mac, Linux, Reproductor web, PlayStation 3, 4, VITA y Mobile,	iOs, Android, HTML5, Linux, Mac, Oculus Rift, PlayStation 4 y Morpheus, SteamOS, Windows Xbox One	iOs, Android, NOOK, Kindle	Navegador web, Windows, Mac, Linux, iOs, Android, BlackBerry, HP WebOS, Bada, Symbian, Marmalade, Wii, xBox 360,

	Xbox 360 y ONE, Wii U, Tizen			PlayStation 3, Portable o Vita
Motor físico	PhysX	PhysX	Box2D	PhysX
Herramientas de IA	NavMesh y Path Finding	Path Finding	No (Sólo programados desde cero en LUA)	AIModel
Características avanzadas	Forward y Deferred rendering, Occlusion Culling, light mapping, light probing, LOD discreto, edición de terrenos, sistema de partículas	Deferred rendering, occlusion culling, parallax mapping, Per Object Motion Blur	Interfaz simple de LUA para la programación de física	Per-pixel lighting, terrains, vegetation, oceans, particles, Video & Data streaming
Licencia	Comercial. Versión básica gratuita (desktop y web) y versiones PRO y plataformas móviles y consolas con un pago único	Comercial. Versión de pago y únicamente gratuita para uso educacional	Comercial. Versión Indie para desarrollar para una única plataforma (iOS o Android) y versión Pro para todas las plataformas	Comercial. Versiones: Web, Basic, Advanced, Educational
Datos de interés	Motor con mayor número de desarrolladores registrados del momento	Motcon el motor grafico más potente del momento según los expertos	OpenGL, OpenAL, Google Maps, Facebook Connect, Game Center, in-app purchase	Herramientas específicas MMO (Massive Multiplayer Online)

Tabla 3.2: Comparación de los cuatro motores de juego más importantes

Por lo tanto, a la hora de elegir un motor de juego para desarrollar nuestro videojuego, la elección no es nada sencilla debido a las buenas características de los cuatro motores.

En primer lugar descartaremos al motor Corona SDK, ya que queremos trabajar con tecnologías 3D y éste solo tiene soporte 2D, por lo tanto es más apropiado para aplicaciones o juegos para *smartphones*, no para nuestro proyecto con la Kinect.

A continuación, revisamos los otros 3 motores que nos quedan. Todos nos permiten desarrollar proyectos 3D tanto para Windows como para Mac, por lo que

ese factor no nos ayuda a decidirnos. Y los 3 nos permiten conseguir unos resultados de un nivel muy alto.

Lo que finalmente nos hace decidirnos es la licencia, debido a que Unity3D se nos ofrece con casi todas sus características de forma gratuita, además de que ofrece una programación orientada a objetos que se ha dado a lo largo de la carrera. En los otros dos motores hay que pagar para tenerlos o al menos para disponer de un editor de calidad.

Además de esto, Unity3D está en auge y se considera el motor más importante, en parte por la gran cantidad de desarrolladores que lo usan hoy en día, por lo que nos decantamos por él.

3.3. Creación y animación de objetos

A la hora de crear los gráficos del videojuego, tenemos que añadir una serie de objetos o "*gameobjects*" los cuales tienen que haber sido creados previamente.

Muchos motores de juego traen incorporada un especie de "tienda" donde poder adquirir estos objetos ya hechos por otras compañías o por usuarios independientes. Así en Unity3D tenemos la *Assets Store* (https://www.assetstore.unity3d.com/en/), la cual incorpora una enorme cantidad de objetos, escenarios y personajes que se pueden adquirir de forma gratuita o pagando.

Sin embargo, estos modelos han sido hechos por medio de programas de diseño específicos para ello, y además nosotros mismos podemos realizar modelos propios adaptados a nuestras necesidades. Esto se hace por medio de programas de diseño gráfico o modelado 3D, entre los que destacan Blender, Maya, Autodesk 3ds Max, MakeHuman, Lightwave, Softimage, SketchUp o Rhinoceros.

El modelado 3D se puede dividir en cinco partes, que son el modelado, que es la forma que se da a los objetos que se van a utilizar, el sombreado, con la cual daremos el comportamiento que tienen que seguir las caras de un objeto cuando reciben luz, el texturizado, que añade imágenes a las caras de los objetos para dar realismo mediante estas texturas, la animación, si queremos que el objeto tenga algún tipo de movimiento, y finalmente el renderizado, que consiste en generar mediante una gran capacidad de cálculo una imagen 2D o animación a partir del objeto o escena creados.

Todos estos programas de diseño gráfico o modelado 3D tienen unas cualidades y una calidad final muy elevadas, por lo tanto esto no van a ser unos factores demasiado relevantes a la hora de escogerlos, ya que todos van a ser muy buenos. Por lo tanto habría que fijarse en cuál se nos da mejor manejar o tiene un manejo más sencillo y, además, el campo para el cual va destinado nuestro proyecto, ya que no es lo mismo el diseño para el mundo del cine que para proyectos relacionados con la arquitectura o diseño de edificios, ni que para el mundo de los videojuegos.

Aparte de diseñar a los personajes u objetos, también necesitamos programas que les den movimiento, como por ejemplo un personaje que nos hable o que haga algún tipo de movimiento. Así se usan los programas de animación 3D.

Estos suelen estar ligados con los programas de diseño 3D que vimos anteriormente, es decir, que muchos de los programas que nos permiten diseñar personajes u objetos, además nos permiten aplicarles algún tipo de animación. Así los mencionados Maya, 3ds Max, Blender, SketchUp o Rhinoceros, permiten realizar ambas tareas.

Por si habría que diseñar algún objeto en nuestro proyecto, se valoraron varios de los programas de modelado y/o animación 3D más conocidos o más utilizados sobretodo por los usuarios de Unity3D. Así de entre todos destacaba Maya, pero fue descartado inmediatamente por su elevada complejidad y coste. También se valoraron SketchUp, Blender, 3ds Max y Rhinoceros debido a su importancia y que los cuatro permiten tanto el diseño como la animación, pero de los cuales estos dos últimos fueron descartados por no ser gratuitos y el primero, por no permitir animaciones más complejas, cosa que Bender sí nos permite.

También se valoraron otras opciones más simples como MakeHuman, para crear diseños humanos, y Mecanim, que viene integrado con Unity3D.

3.3.1. MakeHuman

Según la página oficial del equipo de MakeHuman (2014), éste es un software gratuito, innovador, profesional y con licencia de código abierto (AGPL) para crear personajes humanos en 3D con la finalidad de usarlos en ilustraciones, animaciones, juegos o esculturas Zbrush o Mudbox (logo en la *figura 3.13*).

Figura 3.13: Logo de MakeHuman

En el año 2001, fue Manuel Bastioni quien fundó MakeHuman y en el 2004 él y su equipo recibieron el premio internacional Suzanne por el mejor *script* Bender Pyton en gráficos 3D. Actualmente él es el director de MakeHuman y lo desarrolla como investigador y como modelador. Otros líderes del equipo son Jonas Hauquier, que es el desarrollador del *software* y el que apoya el código abierto y las licencias creativas comunes siempre que sean posibles, y Joel Palmius, el cual lleva las "construcciones nocturnas" y la infraestructura del servidos web de MakeHuman.

Actualmente, la versión disponible de MakeHuman es la 1.0.2, la cual hemos dicho que es completamente gratuita, debido a que la filosofía de este equipo es que "desarrollar es su pasión y así contribuyen a un mundo mejor". Ésta es

soportada por las tres plataformas principales, que son Mac, Windows y Linux. También se pueden descargar otros tres elementos curiosos:

- Construcciones nocturnas o "Nightly Builds", los cuales son *scripts* que según el equipo de MakeHuman se crean cada noche y producen ejecutables que incluyen los últimos cambios. Estos pueden que tengan fallos, por lo tanto por los reportes de *bugs* de los usuarios se pueden arreglar.
- El código fuente de MakeHuman, el cual está programado en Python.
- Herramientas de Blender para importar modelos de MakeHuman a Blender o crear nuevos activos para MakeHuman con Blender. Para ello se necesita la versión 1.0.2 de MakeHuman y la 2.7x de Blender.

Por lo tanto, este programa de modelado 3D es una herramienta diseñada para simplificar la creación de humanos virtuales usando una interfaz gráfica de usuario (GUI), permitiendo que uno de éstos pueda ser creado con pocos clips y con gran facilidad, consiguiendo a la vez resultados de elevada calidad. Los atributos que se pueden dar a nuestro humanoide se dividen en macro o al detalle. Estos primeros tratan de las características generales como son la edad, género, altura, peso y etnia. Los segundos permiten refinar al personaje y centran en los detalles a bajo nivel como la forma de los ojos o la longitud de los dedos.

Pero a pesar de que este *software* incluye herramientas que aun no han sido creadas en otras plataformas, no consigue unos resultados tan logrados como programas más complejos de utilizar como Maya o Blender (así su preocupación por que modelos de MakeHuman puedan ser exportados a Blender, como hemos visto anteriormente), por lo que será útil si necesitamos modelos de humanos simples.

Otro gran inconveniente es que sólo permite el diseño y modelado 3D, pero no la animación, por lo que de nuevo habría que recurrir a la exportación si queremos dar algún tipo de comportamiento a nuestra creación, lo cual será muy común si son humanos lo que estamos creando.

Sin embargo son 12 años los que llevan investigando la topología del cuerpo humano, lo que ha llevado a este programa a ser premiado en Septiembre de este mismo año, en el 12º "Brazilian Congress of Forensic Dentistry", por su presentación del "Protocolo para la reconstrucción facial del forense con *software* libre: Simplificación del método usando MakeHuman".

Finalmente, en la web de MakeHuman encontramos una amplia galería de fotos de diferentes proyectos creados por varios autores y de las cuales hemos sacado la *figura 3.14*, que nos da un pequeño ejemplo de todo lo que se puede crear con esta herramienta.

Figura 3.14: Cara al detalle hecha con MakeHuman

3.3.2. Blender

Este programa del que hablamos a continuación, cuyo logo está en la *figura 3.15*, es una herramienta gratuita multiplataforma de modelado y animación 3D que permite diseñar personajes, elementos y escenas en tres dimensiones con diversas técnicas. Estos elementos pueden ser animados con la técnica del *keyframing* o animación por fotogramas clave (Blender.org, 2014.).

Figura 3.15: Logo de Blender

Sus orígenes se comenzaron a vislumbrar cuando en 1988, Ton Roosendaal cofunda el estudio de animación holandés NeoGeo, que se convierte en uno de los más importantes en Holanda. Él era responsable de la dirección artística y el desarrollo del *software* interno, y decidió que el sistema de herramientas 3D que usaban tenía que ser reescrito. Así en 1995 comenzó esta reescritura, que pasó a formar el hoy en día conocido Blender. En 1998 Ton fundó una compañía que se dedicaba a desarrollar Bender. En el 2000 la empresa siguió adelante gracias a las muchas inversiones que se hicieron en ella, pero en el 2002 el clima económico se volvió difícil y estas inversiones se retiraron, y así este mismo año Ton comenzó la Fundación Blender, la cual no recibía beneficios, y Blender se convirtió en un *software* libre.

Blender se distribuye actualmente con la versión 2.72, que fue lanzada en Octubre de 2014, y de manera gratuita, lo que ha hecho que sea muy popular entre desarrolladores independientes. Además, aunque no fue así desde un principio, este programa tiene la licencia Pública General GNU, o lo que es lo mismo, quiere decir que es de *software* libre, por lo que podemos usar Blender libremente y estudiar como éste trabaja y va cambiando y por ello éste ha ganado terreno ante sus competidores más cercanos, como 3D Studio.

Este programa además es compatible con todas las versiones de Windows, MacOS X, GNU/Linux, Solaris, FreeBSD e IRIX.

Blender dispone por defecto de tres motores gráficos de pre-renderizado y uno de tiempo real. Los motores de pre-renderizado pueden dividirse en dos motores realistas con la finalidad de crear imágenes de aspecto fotorrealista y otro de representación de dibujo a mano.

Los dos realistas son el llamado motor interno de Blender, que es el seleccionado por defecto cuando se ejecuta por primera vez el programa. Cycles es un motor más reciente y basado en el trazado de rayos de luz. A partir de la versión 2.67, Blender incorpora un nuevo motor gráfico, denominado FreeStyle, enfocado a crear trazos que simulan dibujos hechos a mano, y que en la versión actual (la 2.72) ha sido integrado a Cycles.

El motor gráfico para tiempo real está basado en OpenGL, y Blender lo emplea tanto para la edición del escenario 3D a través de su editor 3D View, como para su motor de juegos.

Blender también facilita la creación de un flujo de trabajo con otros motores externos. Por este motivo es posible exportar las escenas a dichos motores. Existen *plugins* que permiten trabajar en Blender usando esos motores como si estuviesen integrados en el propio programa.

Este programa no está limitado a ser usado para el modelado, diseño y animación de gráficos 3D, sino que entre otras cosas también incluye un potente software para la creación de juegos, basándose en el lenguaje de programación Python.

Además, ofrece herramientas de simulación avanzada, tales como dinámicas de cuerpos rígidos y suaves, animación de personajes, un sistema de simulación física avanzado y un sistema de composición se materiales.

Por todo ello, Blender también tiene cierta importancia en la industria del cine y así se ha utilizado en la creación de películas de animación, en concreto en la creación de películas independientes como Elephants Dream (mostrada en la *figura 3.16*) o la argentina Plumíferos. Hollywood ha empezado también a fijarse en él y, aparte, ya ha sido usado en algunos proyectos de apoyo en películas con renombre como Spiderman 2 (Suau, 2011).

Figura 3.16: Captura de la película Elephants Dream, hecha con Blender

La única pega que se le pone a Blender es la complejidad de su interfaz, ya que se considera poco intuitiva. Por lo tanto se han hecho comunes y casi obligatorios los atajos por teclado para poder exprimir sus capacidades al máximo.

Sin embargo, son más los puntos positivos que los negativos, ya que aparte de todo lo ya nombrado, cuenta con gran cantidad de tutoriales explicativos y una guía oficial dentro de su propia página web. Además cabe destacar que permite editar audio y sincronizar video, aceptando formatos gráficos como JPG, PNG, TGA o TIFF. Y por todas estas cosas está considerado como uno de los mejores programas de modelado y animación del momento, sobretodo para desarrolladores independientes.

3.3.3. Mecanim

"Mecanim Animation System" es el sistema de diseño y animación singularmente potente y flexible de Unity3D, que hace que los personajes humanos y no humanos cobren vida con movimientos naturales y fluidos (Unity Technologies, 2014b). Mecanim es una solución integrada en forma nativa con el motor de Unity3D, por lo que es un software muy sencillo de utilizar con él ya que no hay necesidad de desarrollar *middelware* con terceros y luego importarlo a Unity3D, sino que todas las herramientas y flujos de trabajo necesarios para crear y construir clips musculares, árboles de mezcla, máquinas de estado y controladores se encuentras directamente aquí.

Además, ofrece la ventaja de que se puede invocar desde dentro de la reproducción de animaciones cualquier función de un *script* que se desee, con AnimationEvents.

El uso de este elemento no es muy complicado como lo es el uso de terceros para la misma finalidad, pero aun así encontramos en la propia página web de Unity gran cantidad de tutoriales en video, imágenes y un manual de uso que hacen más fácil el entendimiento de esta herramienta.

Mecanim ofrece las siguientes características:

- Flujo de trabajo sencillo y configuración de las animaciones de los personajes humanoides.
- El uso del *retargeting* en las animaciones, que permite aplicar la misma animación que tiene un carácter a otro diferente.
- La simplicidad del flujo de trabajo para alinear clips de animación.
- Previsualización de clips de animación, transiciones e interacciones entre ellos. Esto permite que los animadores tengan más independencia de los programadores, permitiéndoles hacer un prototipo y una previsualización de sus creaciones antes de que el código del juego esté.
- Gestión de complejas interacciones entre animaciones y una herramienta de programación visual.
- Animación de diferentes partes del cuerpo que tienen diferente lógica.

El flujo de trabajo de Mecanim se puede dividir en tres etapas generales que son la de preparar e importar los activos o *assets*, el cual se hace con programas externos como Max o Maya, la configuración de caracteres para Mecanim, que se puede hacer creando un humanoide (*retargeting* y definición de músculos) o un personaje genérico, y finalmente la de dar vida a los personajes configurando clips de animación, máquinas de estados y árboles de mezcla.

Los árboles de mezcla (*figura 3.17*) permiten al usuario crear una amplia variedad de movimientos con pocos clips. En el *Blend Tree Editor* se definen los parámetros de mezcla y se obtiene una vista previa de las animaciones combinadas con una vista 3D, y con los nodos de mezcla 2D se pueden mezclar animaciones basadas en dos parámetros en un solo nodo.

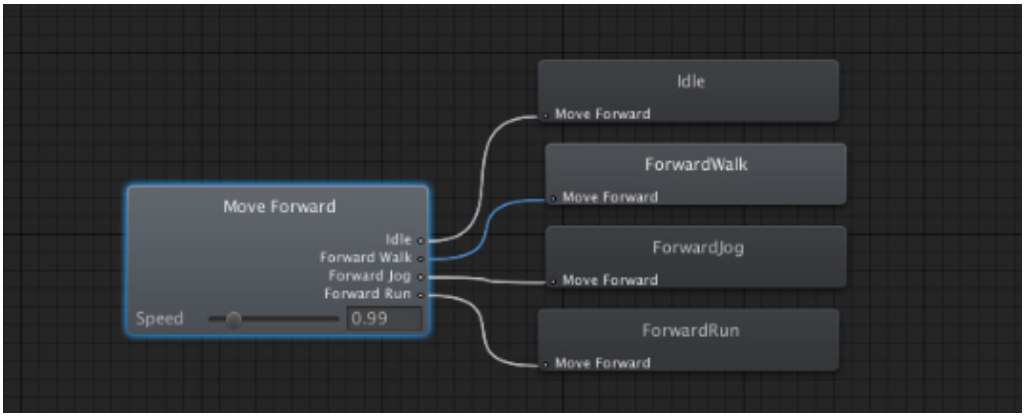


Figura 3.17: Ejemplo de árbol de mezcla de Mecanim

Las máquinas de estado (*figura 3.18*), jerárquicas y multicapa, se crean en la *Animator Tool*. Un controlador puede definir un número arbitrario de capas y cada capa puede usar su propia máquina de estados o compartirla con una "capa maestra". Las capas pueden ser eliminadas o agregadas y el uso de una máscara de cuerpo define a qué parte del cuerpo van a afectar. Por último, se puede utilizar

una jerarquía de máquinas de estado para descomponer un controlador complejo en pequeños módulos reutilizables.

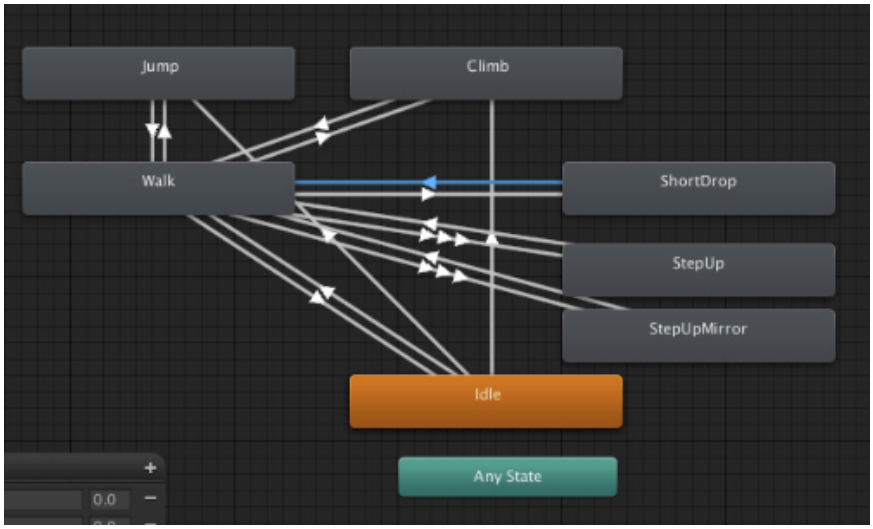


Figura 3.18: Ejemplo de máquina de estados de Mecanim

Por lo tanto concluimos con que Mecanim es una herramienta que ofrece muy buenos resultados de animación de personajes, de reutilización de estas animaciones y que tiene la grandísima ventaja de que viene integrado con Unity3D.

3.3.4. Comparación de programas de creación y animación

Como hemos visto en las descripciones detalladas de cada programa de modelado y/o animación 3D, cada uno de ellos tiene características muy dispares. Por lo tanto parece que la decisión de qué programa deberemos de utilizar va a venir dado por las necesidades que tengamos para nuestro proyecto.

Para ayudarnos a elegir el programa adecuado, veremos la *tabla 3.3* que compara diferentes aspectos importantes para nosotros.

Programas de modelado y/o animación 3D			
Permite el diseño y modelado 3d	Sí	Sí	No
Permite la animación 3D	No	Sí	Sí
Para cualquier tipo de objeto	No (sólo humanoides)	Sí	Sí
Sencillez de manejo	Sí	No	Sí
Importación a un motor de	Sí	Sí	Sí (recomendado para

renderizado externo			Unity3D)
Gratuito	Sí	Sí	Sí
Biblioteca con creaciones gratuitas	No	No	Sí (Assets Store)
Software libre	Sí	Sí	No
Sistemas operativos que lo soportan	Windows, MacOS, Linux	Windows, MacOS, Linux, Solaris, FreeBSD, IRIX	Windows, MacOS

Tabla 3.3: Comparación de los tres programas de modelado y/o animación 3D

Por lo tanto, vemos que si queremos unificar en un mismo programa el modelado y la animación 3D, Blender sería nuestro programa, además de sus otras grandes cualidades. Pero tiene el inconveniente de que es un programa de difícil manejo, ya que requiere ciertos conocimientos previos de modelado 3D, por lo que nos llevaría bastante tiempo aprender a manejarlo y posiblemente sea algo que no compense, ya que para nuestro tipo de proyecto no llegamos a necesitar en ningún momento unos objetos determinados, si no que eran cosas comunes.

Por otro lado, MakeHuman se limita mucho con sólo permitir la creación de humanoides y además del hecho de no poderles dar movilidad, pero sería un gran complemento junto con Mecanim, ya que uno nos permitiría crear personajes de una calidad bastante alta y el otro nos permitiría "darles vida".

Visto lo anterior, y puesto que el diseño 3D no era una parte realmente importante en este proyecto, se decidió por lo tanto usar simplemente Mecanim para dar animación a los personajes que lo necesitaran. En caso de haber necesitado algún humanoide, se hubiera realizado con MakeHuman, pero fue suficiente con los que ya venían de manera gratuita en la *Assets Store* o junto con el *plugin* que utilizamos. Si hubiéramos necesitado algún diseño más complejo, Blender hubiera sido nuestro programa.

En nuestro proyecto en concreto no se llegó a necesitar la animación, pero sí para las prácticas en empresa anteriormente mencionadas, ya que realicé varios ejemplos de animación de humanoides para enseñar en las clases que impartí, por lo tanto se utilizó Mecanim.

3.4. Dispositivos de captura de movimiento

La captura de movimiento es una técnica utilizada para digitalizar movimientos reales, con los cuales de una manera fácil e intuitiva se anima a los objetos y personajes (Alvarado, Barría, Cordella y Lechuga, 2007).

La gran mayoría de los programas de desarrollo 3D facilitan esta tarea incluyendo herramientas para transcribir la información del dispositivo de entrada que estemos utilizando para así poder aplicárselo a un personaje virtual.

Los orígenes de la captura de movimiento se remontan al año 1915, cuando Max Fleishcher hizo el primer intento real de copiar el movimiento para animar personajes mediante el "rotoscoping", técnica que consistía en animar directamente sobre un movimiento filmado cuadro por cuadro. Un ejemplo de uso de éste fue la producción de "Blanca Nieves y los 7 Enanitos" (Rincón y Zambrano, 2012).

A finales de los años 70 fue cuando empezó a ser posible animar a personajes computarizados. Entonces aun se usaba el ya mencionado "rotoscoping" mediante la utilización de unos espejos especiales que se ponían en frente de la pantalla del ordenador reproduciendo videos de actores y creaban una pose del personaje digital, que era utilizada por el ordenador como referencia para la creación de una animación fluida.

A principios de los 80, el profesor de universidad Tom Calvert añadió potenciómetros a un cuerpo y mediante las salidas de este sensor manejaba figuras animadas para estudios coreográficos o el diagnóstico de problemas de movilidad. Más adelante, surgen los primeros sistemas de monitoreo visual que utilizan marcadores especiales en los cuerpos de los actores monitorizados, y que estaban limitados por la velocidad con la que se pueden examinar los marcadores, la resolución de las cámaras y la oclusión de los marcadores en el cuerpo.

Ya en 1988, Brad deGraf y Michael Warhman presentaron en una conferencia a "Mike the Talking Head", demostrando la actuación en tiempo real de la captura de movimiento. Este mismo año, Rick Lazzarinis, comisionado de la Pacific Data Images creó un dispositivo mecánico llamado *exoeskeleton*, capaz de capturar el movimiento de la parte alta del cuerpo y la cabeza y basado en el uso de potenciómetros en cada articulación.

Algo más adelante, a principios de los 90, compañías como MediaLab o Windlight Studios, pioneras en toda esta tecnología, comenzaron a prosperar, ya que los proyectos que usaban captura de movimiento (*Motion Capture* o MoCap) en computación gráfica comenzaban a tener éxito.

A partir de ese momento hasta la actualidad, se han creado gran cantidad de equipos que utilizan diferentes métodos para grabar los movimientos de personas y animales. Además que cada vez se las exige a las empresas dedicadas al MoCap que ofrezcan mejores prestaciones y una representación del movimiento casi perfecta, debido al auge de esta tecnología en la industria del cine y de los videojuegos. Así pues, esta técnica se ha utilizado para proyectos tan importantes como la película de *El Seños de los Anillos* o los últimos videojuegos de *FIFA*.

Los dispositivos de captura de movimiento son muy diferentes unos de otros, y así tenemos unos tan simples como un *joystick*, un ratón o un teclado en entornos no profesionales, y otros mucho más avanzados, y por lo tanto más caros, tales como los dispositivos ópticos que incluyen trajes y sensores para recopilar movimientos más complejos con mucha más precisión (Alvarado et al., 2007). Por lo tanto la elección entre unos y otros va a depender del entorno de trabajo.

Así encontramos los sistemas ópticos o visuales, los cuales capturan la información utilizando una o más cámaras sobreponiendo la proyección del objeto grabado y luego procesando estos datos para triangular la posición del objeto; los sistemas no visuales, que para triangular la posición utilizan otro tipo de sensores como giroscopios o acelerómetros (sistemas de inercia); la utilización de indumentaria electromecánica, la cual se adapta al cuerpo humano y recoge los movimientos mediante barras metálicas o de plástico unidas por potenciómetros que se encuentran en las articulaciones; los sensores magnéticos, que miden la relación espacial con un transmisor cercano; y finalmente encontramos el uso de fibra óptica, utilizando un conjunto de estas en un cuerpo ya que al doblarse atenúan la luz transmitida, permitiendo ver dónde se encuentran las extremidades.

Para nuestro proyecto evaluaremos una serie de dispositivos que ofrecen MoCap mediante la tecnología óptica o visual y que pueden considerarse como intermedios en cuanto al precio y a su avance tecnológico.

3.4.1. Kinect

La idea de desarrollo de un dispositivo de entrada sensible al movimiento por parte de Microsoft® fue anunciada bajo el nombre de "Project Natal" el 1 de Junio de 2009. Pero finalmente la Kinect que conocemos (mostrada en la *figura 3.19*) fue lanzada el Noviembre del 2010 y rápidamente de convirtió en el aparato electrónico más rápido comprado por los consumidores, consiguiendo por ello entrar en el Libro Guinness Mundial de los Records (Catuhe, 2012).

Figura 3.19: Primera versión de Kinect para Windows

No fue hasta Junio de 2011 cuando Microsoft® anunció el lanzamiento del *Software Developer Kit* o SDK para la Kinect para Windows, ya que anteriormente sólo se podía utilizar para "jugar" con ella en la consola Xbox 360. Ya a principios del 2012, Microsoft® lanzó la versión comercial de SDK, permitiendo a los desarrolladores de todo el mundo el uso del poder del sensor de la Kinect en sus propias aplicaciones.

Kinect permite a los usuarios interactuar con el PC o la videoconsola y controlarlos sin la necesidad de ningún contacto físico con algún periférico de

entrada, gracias a la interfaz natural de usuario (NUI) que posee reconocimiento de voz y de movimientos.

Para explicar los componentes internos de Kinect, mostramos en la *figura 3.20* la máquina sin carcasa, pudiendo ver cada una de sus partes señaladas. De izquierda a derecha, en primer lugar encontramos la fuente de luz infrarroja. A continuación está el indicador LED que se enciende cuando la Kinect está conectada por USB al PC de forma que ésta funciona. A su lado está la cámara de color usada para recoger los datos RGB, que tiene una resolución máxima de 1280 x 960. Y finalmente a la derecha del todo vemos la cámara de infrarrojos que trabaja en conjunto con la fuente de luz infrarroja para capturar los datos de profundidad y tiene una resolución máxima de 640 x 480 (Webb y Ashley, 2012). Esta tecnología de imagen de profundidad utilizada fue desarrollada por PrimeSense.

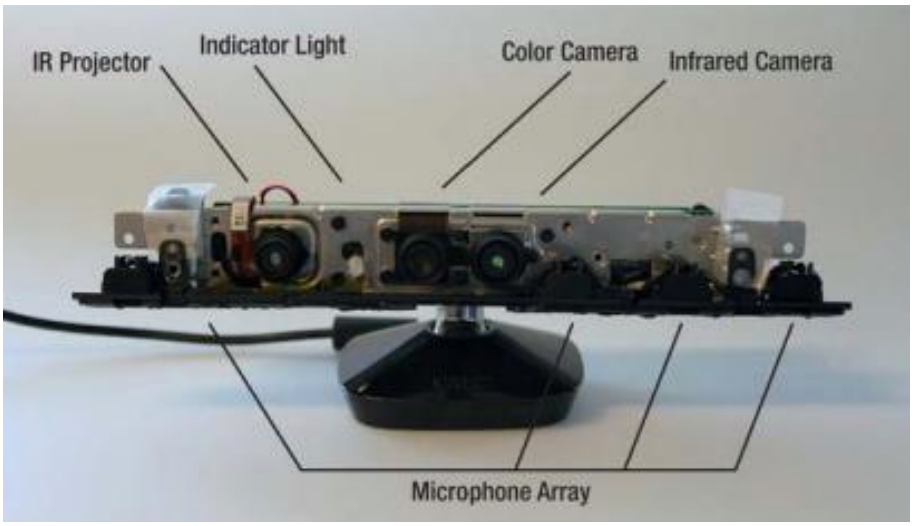


Figura 3.20: Interior de Kinect para Windows

También encontramos un *array* de micrófonos en la parte inferior, compuesto por 4 micrófonos diferentes situados en las 4 zonas señaladas por la flechas de la imagen.

El sistema de escaneo que se utiliza para capturar datos de profundidad, se conoce como *Light Coding* y es una variante de la reconstrucción 3D. Su funcionamiento se basa en la codificación del volumen de la escena con luz de tipo infrarrojo cercano, invisible para el ojo humano. Mediante un sensor CMOS estándar se lee la luz codificada de la escena, para que instantáneamente se ejecute un algoritmo de computación en paralelo para descifrar y generar la imagen de profundidad. Esta solución es inmune a la luz ambiental.

Kinect también dispone de un mecanismo de inclinación motorizado. Por este motivo, y para poder cubrir el consumo de energía del sensor, necesita más corriente de la que le proporciona el puerto USB 2.0 convencional, por lo que para disponer de la corriente óptima hace uso de un conector propietario especial (Catuhe, 2012).

No hay una CPU dentro del sensor, sólo un DSP que procesa la señal del *array* de micrófonos, por lo que el procesamiento de datos se ejecuta en el PC gracias al *driver* de Kinect.

Este *driver* puede ser instalado en las versiones Windows 7 y Windows 8 (que son las que soportan Kinect para Windows) y sobre un procesador de 32 o 64 bits. También se necesitarán al menos 2 GB de RAM y un dual-core de 2.66 GHz.

Actualmente, se distribuye la versión 2 de Kinect para Windows, la cual ha mejorado en muchos aspectos como la calidad a 1080p de la cámara de color, la orientación del esqueleto, mano y *joints*, el seguimiento facial y mayor campo de visión, entre otras cosas (Microsoft®, 2014a).

3.4.2. DepthSense

DepthSense es un sensor de profundidad patentado por SoftKinetic, la cual es una compañía belga líder ofreciendo tecnologías de visión 3D para PCs, aparatos electrónicos portables, coches y máquinas, para así permitir crear las interacciones entre el hombre y la máquina de la forma más natural e intuitiva (SoftKinetic, 2014).

Actualmente, se comercializan los modelos DS311 y DS325 (*figura 3.21*), siendo éste segundo el más común. Su sensor mide el tiempo que toma la luz infrarroja emitida por DepthSense en volver, y transforma estos datos conocidos como "tiempos de vuelo" en datos en "tiempo real", imágenes RGB o en escala de grises y en imágenes del mapa de profundidad.

Figura 3.21: Cámara DepthSense DS325 de SoftKinetic

Ésta cámara ofrece datos de profundidad a una distancia como mínimo de 15 cm y hasta 60 fps. Tiene una resolución de profundidad por QVGA de 320 x 240, una resolución RGB de 720p y necesita luz de ambiente de interiores (desventaja ante Kinect, la cual no necesita luz de ambiente).

Además, este dispositivo posee una sola entrada USB y junto con el sensor de profundidad y la cámara RGB, combina 2 micrófonos.

Por otro lado, SoftKinetic nos permite trabajar con un middleware para el desarrollo de aplicaciones denominado "iisu SDK". Dispone de interfaces y patrones de gesticulación predefinidos para reducir los ciclos de desarrollo, y compatible con Windows 7 y 8, y aunque con más dificultades también con Linux y Android. El problema es que no permite el reconocimiento de cara y voz, si no de las manos y dedos.

La tecnología de Softkinetic la hemos visto presente en iSec, una videoconsola de Lenovo que salió a la venta en China a finales de 2011 y de la que se dice que puede ser un gran competidor en la próxima generación de videoconsolas (Conneally, 2014).

3.4.3. Minoru 3D

La Minoru 3D Webcam fue lanzada en enero de 2009 y fue la primera cámara con 3D incorporado. Ésta se conecta mediante USB al ordenador, de forma que sus dos lentes se iluminan y comienza a grabar videos anáglifos, es decir, videos en 3D, así como en otros formatos estereoscópicos (Zone, 2013). Para conseguir este tipo de imagen con efecto estereoscópico, el dispositivo cuenta con dos cámaras situadas a una distancia similar a las de los ojos en una persona, que podemos ver en la *figura 3.22* (RobotShop Distribution, 2014).

Figura 3.22: Cámara Minoru 3D

El *software* de Minoru tiene procesamiento anaglífico nativo y la imagen en 3D puede ser vista por cualquiera que lleve unas gafas 3D de lentes rojas y azules, aunque también permite ser utilizada como una cámara convencional de imagen en 2D (Zone, 2013).

Esta cámara nos permite también sacar fotos en 3D y funciona para programas como Skype, el antiguo Windows Live Messenger, AOL Instant Messenger, Oovoo o incluso para subir estos vídeos 3D a YouTube.

Según su página oficial (2014), las especificaciones que posee son un sensor VGA con tecnología CMOS, una resolución máxima de 800 x 600, una tasa máxima de 30 fps, enfoque manual, un ampo de 42 grados y que los dos obturadores de

izquierda y derecha no se sincronizan. Además, es solamente compatible con el sistema operativo Windows y sus versiones XP, Vista y 7.

Actualmente, ésta se encuentra en el mercado por tan sólo 29,99 € en tiendas online como Amazon, en las cuales, cuando se lanzó el producto, tenía un precio inicial de 89.95 $. En aquel entonces, fue lanzada para todo el mundo justo después de ser presentada en la Asociación de la Electrónica de Consumo (CES) y ganar el premio de "Favorito para los Fans".

3.4.4. Xtion

Xtion es el dispositivo creado por ASUS para introducirse en este mercado y competir directamente con Kinect. Este producto está destinado específicamente para PCs (ASUS, 2014). En un primer momento, consistía únicamente en un sensor de profundidad. Unos meses más tarde se integró también una cámara RGB junto con este sensor de profundidad.

3.23: Dispositivo Xtion de ASUS

La acogida de este dispositivo no ha sido la esperada, ya que sus prestaciones no superan a Kinect. Además el precio de Xtion es superior, y su competidora puede utilizar varios kits de desarrollo diferentes.

Ofrece una resolución de imagen RGB de 1280 x 1024, y el sensor de profundidad puede trabajar por VGA a 640 x 480 y 30 fps o permite reducir a la mitad la resolución para doblar la tasa de fotogramas por QVGA (320 x 240 a 60 fps). Funciona tanto en Windows (XP y 7) como en Linux Ubuntu 10.10: X86, y utiliza como kit de desarrollo OpenNI.

3.4.5. Comparación de dispositivos de captura de movimiento

Una vez explicados todos estos dispositivos de captura de movimientos, procedemos a hacer la correspondiente comparación para comprobar cuál se adapta mejor a las necesidades del proyecto a realizar.

Realmente la idea inicial fue siempre realizar este proyecto para Kinect, ya que se disponen de varias de éstas en el laboratorio y además es la cámara más común y más conocida entre los usuarios debido a que es utilizada para gran cantidad de juegos de la videoconsola Xbox de Microsoft®, pero siempre es útil explorar otros dispositivos por si pudiera haber una opción mejor.

Así pues, se muestran las características principales comparadas en la *tabla 3.4*.

Dispositivos	KINECT for Windows	DepthSense	minoru	Xtion
Consumo energético	Requiere alimentación adicional a la del USB	< 2.5 W	< 1.5 W en modo operativo, < 2 mW en *standby*	< 2.5 W
Precio	149.99 €	143.28 €	29.99 €	179.15 €
Resolución	640 x 480	320 x 240 720 p	800 x 600 máximo	1280 x 1024
Tipo	Luz estructurada	TOF	Estereoscópica	Luz estructurada
Compatibilidad	Windows (7 y 8)	Windows (7 y 8), Linux y Android	Windows (XP, Vista y 7)	Windows (XP y 7), Linux y Android
FPS	30	25 - 30 50 - 60	Hasta 30	30 o 60
Software	OpenNI Kinect for Windows SDK	iisu	Librerías para OpenCV	OpenNI
Rango	1.2 – 3.5 m	15 cm – 1 m	> 10 cm	0.8 – 3.5m

Tabla 3.4: Comparación de los dispositivos de captura de movimiento

Por consiguiente tendremos que comprobar que Kinect es perfectamente válida para nuestra finalidad. Así pues, el rango de distancia es completamente válido, ya que es la suficiente distancia para que una persona se vea de cuerpo entero en la pantalla, capturado por nuestro dispositivo, y que pueda tener una distancia propicia para los movimientos propios de la rehabilitación, la cual este rango nos ofrece.

Por otro lado la tasa de fotogramas por segundo es mejorable, pero 30 fps no suponen un problema en nuestro videojuego, y lo mismo ocurre con la resolución.

Aparte, como puntos a favor podemos decir que además de ser compatible con las últimas versiones de Windows, según el *plugin* que utilicemos (de los cuales hay una amplia variedad) también lo puede ser con MacOS, y también la gran ventaja de que la luz ambiental no influya en la captura de los cuerpos debido a que se produce por luz infrarroja.

3.5. *Plugins* para Kinect

Un *plugin* es un complemento informático que relaciona a dos o más elementos para que estos puedan interactuar o, más bien, para que uno pueda añadirle funcionalidades al otro. De esta forma los *plugins* suelen ser usados por desarrolladores para evolucionar un programa o dispositivo o para crear aplicaciones novedosas para éste.

Así, en nuestro caso, necesitamos un *plugin* para que Unity3D funcione con la Kinect de Microsoft® y así poder desarrollar juegos para Kinect mediante la tecnología que Unity3D nos ofrece.

Para algunos de ellos, en primer lugar necesitaremos instalar el *Software Depelopment Kit* (SDK) oficial de Microsoft® para que la Kinect sea compatible con nuestro PC con sistema operativo Windows y a partir de ahí instalaremos el *plugin* que nos dará nuevas funciones con Unity3D para desarrollar nuestro videojuego. Sin embargo, para otros *plugins* no es necesario instalar la SDK previamente, si no que estos hacen la función de ésta e incluso permitiendo que la Kinect funcione con otros sistemas operativos.

3.5.1. Kinect for Windows SDK

El *Sofware Development Kit* o SDK de Microsoft® permite a los desarrolladores crear aplicaciones que soporten reconocimiento de voz y gestual usando en ordenadores la tecnología que ofrece Kinect (Microsoft®, 2014a). Esta SDK provee herramientas y APIs necesarias para el desarrollo.

Las versiones para la Kinect que nosotros estamos utilizando han llegado hasta la actualidad hasta la 1.8, la cual es compatible con Windows 7, 8 y 8.1. A partir de aquí se pasó a la versión 2.0 que es para la Kinect V2 que recientemente acaba de salir y de la cual hablaremos en las líneas futuras.

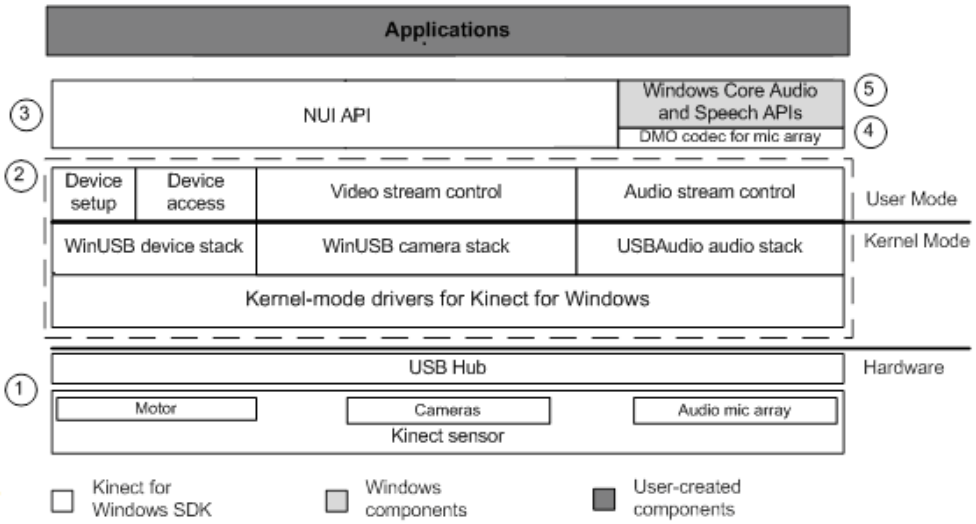
Lo que exactamente contienen estas SDK son *drivers* para que el sensor de la Kinect funcione con nuestro PC, interfaces del dispositivo e interfaces de programación de aplicaciones (APIs), e incluso podemos descargar un kit de herramientas denominado *Kinect for Windows Developer Toolkit* que contiene ejemplos de código fuente, la SDK del seguimiento de cara, cantidades de recursos para el desarrollo, Kinect Studio y Kinect Fusion; es decir, con estas herramientas podremos crear nuestro propio código.

Para poder instalar la SDK 1.8 necesitaremos un procesador de 32 o 64 bits que al menos sea Dual-Core de 2.66 GHz, entrada para UBS 2.0 y 2 GB de RAM, además de Visual Studio 2010 o 2012, .NET Framework 4 o 4.5 y, si queremos desarrollar el reconocimiento de voz, instalar Microsoft® Speech Platform SDK v11.

Además de todo esto que Microsoft® nos ofrece para el desarrollo, en su web (Microsoft®, 2014c) viene una librería donde nos explica al detalle diferentes aspectos de la SDK que vayamos a utilizar, desde las novedades que contiene esa

versión hasta una explicación de los ejemplos de código que contiene, pasando por una guía para el programador y las clases de código ya creadas, lo que hace mucho más fácil el proceso de aprendizaje y manejo de todas estas herramientas.

Es allí mismo donde encontramos la arquitectura de la SDK mostrada en la *figura 3.24*. Así, la librería se encuentra entre medias del flujo de imágenes, profundidades y audios que captura la Kinect y de las aplicaciones, es decir, que es la que hace posible esta interacción. Por lo tanto los componentes de la SDK son los siguientes:



3.24: Arquitectura de la SDK de Microsoft®

1- Es el *hardware* de la Kinect que va conectado al ordenador.

2- Los *drivers* de Kinect que tienen soporte para el array de micrófonos, los controles para audio y vídeo en *streaming* y funciones para poder usar la aplicación con más de una Kinect.

3- Los componentes de audio y vídeo para el seguimiento del cuerpo humano, audio y la captura de la imagen de profundidad y en color.

4- Objeto DirectX Media para la formación del haz del array de micrófonos y la localización de la fuente de audio.

5- APIs estándar de Windows 7 u 8.

3.5.2. Kinect Wrapper

Éste es el primer *plugin* para Kinect de los dos que vamos a ver. Es muy simple, ya que necesita de la instalación de la SDK de Microsoft® previa, ya que si no la Kinect no funcionaría con nuestro PC.

La mayor ventaja de este *plugin* es que es completamente gratis y no añade ninguna marca a tu aplicación, es decir, que no te obliga a hacer propaganda de ello por medio de alguna imagen o marca de agua en el videojuego.

Lo que contiene este paquete son dos *prefabs*. El primero es un "PointMan" que muestra las articulaciones que se capturan a través de puntos en la pantalla. El segundo está vacío y simplemente contiene todos los *scripts* que el usuario necesita para empezar a usar Kinect.

También encontramos diferentes *scripts* entre los cuales destacan los que muestran una imagen de la profundidad o en colores RGB, el controlador al que se "engancha" el usuario detectado, el que captura datos de la Kinect física y el que toma datos del esqueleto capturado.

Y finalmente encontramos un modelo, que es llamado "rainbowMan", y el cual será controlado directamente por nuestro cuerpo. En la única escena que viene de ejemplo con este *plugin* podemos ver esto junto con una imagen en profundidad y otra en RGB.

3.5.3. ZigFu

Este *plugin* es mucho más completo e independiente que el anterior. En primer lugar, nada más descargarlo nos vienen varias escenas de ejemplo, las cuales podemos analizar con Unity3D para ver los *scripts, prefabs* y materiales utilizados. Éstas son muy variadas y nos muestran cómo se hace el seguimiento de la mano del usuario, cómo se detecta a más de un usuario y se hace seguimiento de su cuerpo, cómo se captura nuestro esqueleto, las pantallas de la captura de Kinect en profundidad, RGB y mostrando sólo la silueta del usuario y cómo se seleccionan objetos con la mano, todo ello con Zigfu.

Otro aspecto que destaca ampliamente de este *plugin* es el hecho de que posee su propio *Software* de desarrollo llamado ZDK, por lo tanto no necesitará de la previa instalación de la SDK de Microsoft® (Motion Arcde Inc., 2014).

La ZDK que nos interesa es la que nos permite crear un videojuego con Unity3D, pero también la encontramos para poder desarrollar aplicaciones web con Javascript o HTML5 y para hacerlo con Flash (la cual aún está en desarrollo).

Esta está disponible tanto de manera gratuita como pagando una licencia de 200$ si es para un desarrollador individual, de 1500$ si es para una compañía de menos de 10 empleados o de 3000$ si la compañía tiene menos de 25. Ambas versiones son del 1 de Enero del 2013 y tienen características idénticas como las últimas actualizaciones de la ZDK, soporte para OpenNI2, resolución de errores y mostrando las APIs de seguimiento que subyacen. Pero la de pago puede tener uso comercial, posee ciertas funcionalidades extras y no contiene la marca de agua de Zigfu durante todo el videojuego.

Además, esta ZDK cuenta con una página web de lo más completa, la cual trae ejemplos, tutoriales y la API de esta ZDK, facilitando enormemente el desarrollo con ésta y asegurándonos una fuente de información fiable y completa.

Por otro lado, después de instalar esta ZDK habría que instalar el *plugin* el cual permite que nuestra Kinect funcione tanto para Windows como para Mac, lo cual es una gran ventaja frente a la SDK de Microsoft® que sólo lo hacía con Windows.

Aunque si ésta ya se tiene instalada en el PC, o se tiene OpenNI/NITE, Zigfu nos ofrece la posibilidad de descargar este *plugin* sin *drivers*, es decir, sin necesidad de instalar su ZDK.

3.5.4. Comparación de *plugins*

De nuevo tenemos que valorar cuál de los dos *plugins* nos proporciona mejores soluciones para lo que nosotros necesitamos a la hora de desarrollar el proyecto con Unity3D. Así hacemos una comparación directa en la *tabla 3.5* que nos ayudará a tomar una decisión:

Plugins	**KINECT WRAPPER**	zigfu
Necesidad de Microsoft® SDK previa	Sí	No (sobre ZDK propia, SDK o OpenNI/NITE)
Sistemas operativos que lo soportan	Windows	Windows y Mac
Web con información y API	No	Sí
Coste	100% gratuita	Versión gratuita con ciertas limitaciones. Licencia básica de 200$.
Marca de agua	No	Sí (versión gratuita)
Escenas ejemplo	1	9

Tabla 3.5: Comparación de los *plugins* para que Unity3D funcione con Kinect

En términos generales vemos que Kinect Wrapper es bastante más simple que Zigfu, quizás debido a que su "base" es la SDK oficial de Microsoft® y allí está la información necesaria de ésta, por lo tanto se consideró que no era necesario ofrecer demasiado material de apoyo sobre el *plugin*.

Por lo tanto, a nivel de material extra y de ejemplos en los cuales apoyarnos, nos quedamos con Zigfu, debido a su completa API, sus tutoriales, sus ejemplos y el gran número de escenas ejemplo, las cuales además son muy variadas. Además, en estas escenas podemos encontrar *prefabs* y objetos muy interesantes los cuales podemos utilizar, como son sobre todo los personajes masculino y femenino.

Aparte de todo esto, cada vez más gente dispone de un ordenador Mac, entre los que me incluyo, por lo que el poder desarrollar y probar el videojuego también en esta plataforma además de en la extendidísima Windows, es otro punto muy favorable. Mencionando que Zigfu nos permite también usar la SDK de Microsoft® u OpenNI/NITE, lo cual lo hace aún más modular.

La única pega que podríamos ponerle sería la permanente marca de agua que sale en la parte inferior izquierda de la pantalla que veremos en las imágenes del juego más adelante. Todo esto si queremos que sea gratuita al igual que lo es Kinect Wrapper, la cual sólo ofrece esta versión. Aunque consideramos esto como un mal sin importancia y las anteriores ventajas mencionadas merecen la pena a cambio de este pequeño inconveniente que, en todo caso, puede ser solventado con un precio para nada desorbitado de 200$.

Por otro lado, la SDK posee ciertas funciones extras que la ZDK no, como puede ser el reconocimiento de voz. Pero de momento para el proyecto desarrollado no se necesitaba ninguna de estas funciones extra de la SDK, por lo que no se cambió la idea de elegir Zigfu.

Capítulo 4. Juego serio desarrollado

En este capítulo ya nos vamos a meter en profundidad con el videojuego diseñado, después de evaluar los diferentes campos que influían en su creación, por lo que puede considerase la parte con más peso de este informe.

Así pues, comenzaremos con un enfoque general acerca del juego y su finalidad, explicando los mecanismos que utilizamos para conseguir el propósito deseado. A continuación, hablaremos en detalle de cada una de las escenas que componen el videojuego total, explicando la utilidad de cada una y detallando las partes más importantes mediante el código fuente y/o capturas del juego. Después de esto, explicaremos cómo funciona el videojuego como si fuera un manual para el usuario. Y finalmente, hablaremos sobre las pruebas que se realizaron y según las cuales se decidió seguir unos caminos en detrimento de otros.

4.1. ¿En qué consiste?

Como ya hemos comentado anteriormente en varias ocasiones, nuestro juego es un videojuego serio cuya finalidad es ayudar a las personas con movilidad reducida a llevar mejor su día a día mejorando sus capacidades, es decir, mejorando la calidad de vida de estas personas incluso llegando a darles la movilidad que en su día perdieron por culpa de una enfermedad o trastorno.

De estas enfermedades o trastornos que pueden causar estos problemas de movilidad ya hablamos anteriormente en el *capítulo 2*, aunque nuestro videojuego irá sobretodo centrado a las personas que padecen secuelas provocadas por un ictus, pero será completamente extrapolable a personas con secuelas causadas por otro tipo de trastornos.

Así pues, el ictus produce trastornos en el movimiento impidiendo que ciertas partes del cuerpo se muevan con soltura, que se realicen ciertos movimiento más complicados e incluso provocan la parálisis total de algunas extremidades.

Además, junto a estos problemas van asociados otros relacionados con la coordinación, la percepción del propio cuerpo que hace que el paciente crea que se encuentra en cierta posición diferente a la que en realidad tiene adaptada y problemas en el habla.

Ya que el grado de daño causado a cada persona depende de cómo le haya afectado la enfermedad, cada uno necesitará diferentes niveles de rehabilitación y diferentes ejercicios relacionados con sus capacidades. Por lo tanto, era importante

hacer un videojuego que englobara a un amplio rango de problemas en el movimiento y que además fuera adaptativo a cada usuario.

Los aspectos principales que se tuvieron en cuenta que incluyera el juego eran:

- Coordinación.
- Movilidad de extremidades con menor rango de movimiento.
- Equilibrio.
- Ampliación del rango de movimiento.
- Mejora de la percepción del propio cuerpo.
- Adicción saludable.
- Entretenimiento.
- De fácil manejo para el paciente y fisioterapeuta.

Con estos principios establecidos, surgió la idea de crear un videojuego simulando unas siluetas las cuales vendrían de frente y el usuario tenía que adaptar su cuerpo a ellas para conseguir puntos.

Pero antes de esto había que hacer una especie de "análisis" al usuario para detectar cuál es su grado de movilidad, ya que no se podrá proponer imitar las mismas posiciones a diferentes pacientes, ya que habrá algunos que no puedan mover un brazo, otros una pierna e incluso otros que se encuentren en una silla de ruedas. Por lo cual, la idea fue la de realizar un test inicial en el cual se calculasen los ángulos de movimiento en las articulaciones del cuerpo y a partir de esos valores recogidos se pudieran adaptar de forma apropiada las siluetas que dicho usuario tendrá que imitar dentro del propio videojuego.

Por otro lado, el juego tendría que contar con varios niveles para que no fuera siempre el mismo grado de entrenamiento, si no que el usuario cada vez tuviera que esforzarse más para poder conseguir mayor movilidad, ya que la finalidad de la rehabilitación es la mejora de la calidad de vida del propio paciente y de la gente que lo rodea, gracias a que éste tuviera que ejercer cada vez menos esfuerzo para realizar tareas cotidianas tales como recoger algo de un armario, ducharse o desplazarse por sí mismo.

Además se requería de un juego que crease cierto grado de adicción, de forma que aparte de que fuera entretenido, se le añadió el factor de la competitividad creando un ranking de las mejores puntuaciones, de forma que además de superarse a ellos mismos, buscasen superar a los demás.

En el campo de los trastornos de la percepción visual Kinect es una aparato altamente útil, ya que captura el cuerpo humano y así el paciente puede verse en la pantalla y no sólo imaginarse cómo sería su posición, lo que significa el llegar a corregir poco a poco los fallos asociados a este campo.

Finalmente, otro aspecto muy importante del proyecto es que el fisioterapeuta pudiera gestionar la rehabilitación por medio de una configuración, de forma que

pudiera introducir posiciones de interés, regular la precisión de los movimientos, la velocidad de reacción del paciente, etc... De esta forma se añadió una escena de configuración al proyecto la cual puede ser controlada tanto por el fisioterapeuta como por el paciente, ya que también interesa que este videojuego pueda ser un complemento en casa de la rehabilitación de la clínica.

Todo este juego se ha intentado hacer de la forma más simple posible para el paciente, ya que debía de ser interesante para un amplio rango de edad que va desde los más jóvenes hasta gente de edades avanzadas, y para estos últimos puede ser a veces muy complicado adaptarse a la tecnología.

4.2. Estructura del juego

El videojuego está dividido en 7 escenas de diferente complejidad. La primera que vemos al iniciar el juego es la escena de inicio, a partir de la cual podemos acceder a las demás. Lo ideal es que si somos nuevos usuarios entremos a la escena del test para que nuestra capacidad de movimientos sea recogida, y una vez que esto esté completado, juguemos a la partida en la escena con el nombre de escena pasarela.

Una vez hecho todo esto, tenemos una escena con el ranking de las mejores puntuaciones en la que el usuario puede aparecer si ha llegado a estar entre ellas. También encontramos la escena de la configuración para cambiar ciertos parámetros relacionados con el juego, y las escenas de instrucciones y créditos, las cuales nos indican todo este proceso y niveles del juego y por quiénes ha sido hecho, respectivamente.

A continuación, describiremos al detalle cada una de ellas.

4.2.1. Escena inicio

La primera escena que vemos nada más se inicia el juego es la escena de inicio. Como se ve en la *figura 4.1*, ésta tiene 3 opciones principales que son:

- Hacer test de movilidad: La cual nos llevará a la escena del test que calculará los rangos de movimiento del usuario.
- Empezar partida: Que nos lleva a la escena de la partida, en la cual el usuario practicará diferentes movimientos que formarán parte de su proceso de rehabilitación.
- Ver ranking: Cuya elección cargará la escena con el ranking de las 10 mejores puntuaciones.

Aparte de ellas, vemos abajo a la izquierda de la imagen que hay otras 3 opciones que son:

- Instrucciones
- Configuración
- Créditos

Todas estas escenas se irán describiendo a lo largo de este apartado 4.2, pero ahora nos centraremos solamente en la escena inicio, la cual también denominaremos como el menú principal.

Figura 4.1: Captura de la escena inicio del videojuego

Esta escena está formada por un plano principal que forma el fondo de la imagen y engloba todo lo demás. Sobre éste se encuentran varios planos que muestran el logo del Grupo de Telemática e Imagen (GTI) perteneciente a la Universidad de Valladolid, otro con unas breves explicaciones de lo que hace cada opción y las teclas de acceso rápido para las instrucciones, configuración y créditos, y otros 3 con las opciones principales que nos llevarán a cada una de las escenas.

Cada una de las 3 opciones principales del menú contienen un *box collider* con la opción *is trigger* deseleccionada, lo que hace que pueda ser detectado cada vez que otro *collider* entra en este territorio y así poder realizar cualquier acción en respuesta.

En nuestro caso, queremos que se detecte cuando el cursor está seleccionando esa opción, por lo tanto el *gameObject* de la mano producirá un rayo hacia el frente de forma que cada vez que esté en colisión con alguna opción del menú durante al menos 3 segundos, se cargará la escena que tenga asignada. Vemos cómo se crea y se gestiona este rayo por medio del código de la *figura 4.2*, siendo éste el *Raycast*, y cómo se carga la escena seleccionada mediante *Application.LoadLevel()* al cabo de los 3.0 segundos.

```
18          RaycastHit hit;
19
20          if (Physics.Raycast (transform.position, transform.forward, out hit)) {
21              Debug.DrawLine(hit.transform.position, hit.point, Color.red);
22              if (hit.collider.gameObject.tag == "menu1") {
23                  hit.transform.renderer.material.mainTexture = select1;
24                  if(c1 == 0){
25                      tiempo = Time.time;
26                      c1++;
27                  }else if (Time.time >= tiempo + 3.0F){
28                      Application.LoadLevel("escenaTest");
29                  }
30              }else if (hit.collider.gameObject.tag == "menu2") {
31                  hit.transform.renderer.material.mainTexture = select2;
32                  if(c2 == 0){
33                      tiempo = Time.time;
34                      c2++;
35                  }else if (Time.time >= tiempo + 3.0F){
36                      Application.LoadLevel("escenaPasarela");
37                  }
38              }else if (hit.collider.gameObject.tag == "menu3") {
39                  hit.transform.renderer.material.mainTexture = select3;
40                  if(c3 == 0){
41                      tiempo = Time.time;
42                      c3++;
43                  }else if (Time.time >= tiempo + 3.0F){
44                      Application.LoadLevel("escenaRanking");
45                  }
```

Figura 4.2: Parte del código fuente del cursor de la escena inicio

Por otro lado, estas opciones se pueden seleccionar también haciendo clic con el ratón sobre ellas o presionando en el teclado las letras que nos indica el plano inferior izquierdo de la *figura 4.1* ("T" para iniciar el test de movilidad, "P" para comenzar la partida y "R" para ver el ranking de puntuaciones). Esto ofrece una gran ventaja para las personas que tengan problemas de movilidad en los brazos y no puedan manejar el cursor descrito anteriormente, o para cuando esta rehabilitación se realice con el fisioterapeuta, ya que éste puede ser quien maneje desde su sitio el teclado y el ratón para guiar al paciente.

Para entrar a las otras 3 opciones de las que dispone este menú principal, se hace también por medio del teclado, presionando en este caso la "I" para mostrar las instrucciones, la "C" para configurar el juego o la "Q" para ver los créditos. Esto es debido a que la clase Input es una interfaz del sistema de entrada (Unity Technologies, 2014c) y detecta la entrada que haya, para que según cuál sea ésta se pueda realizar una acción determinada (en nuestro caso, la de *Application.LoadLevel()*).

4.2.2. Escena test

Esta es la escena que compone la primera opción del menú. Su finalidad es filtrar en la partida las posiciones que el usuario no pueda realizar y tener un registro de los grados de movimiento del paciente para que puedan ser utilizados más adelante.

Al inicio de ésta se nos indica por pantalla que vamos a realizar el test de movimientos y que introduzcamos nuestro nombre de usuario. Este nombre de usuario sirve para dar nombre, junto con la fecha a la que se hizo el test, a un

fichero XML el cual guardará los datos correspondientes a la movilidad del paciente y del cual vemos un ejemplo en la *figura 4.3*.

Estos datos consisten en los ángulos máximos de movimiento alcanzados por el paciente con las diferentes partes de su cuerpo. Por lo tanto, en el fichero estarán organizados en diferentes campos , en los cuales el primero será el nombre del usuario, el último será su puntuación máxima (la cual inicialmente se crea con un valor 0 y se actualizará con cada partida jugada) y entre medias encontraremos estos ángulos organizados según la parte del cuerpo a la que pertenezca esa medida y el plano al que corresponde, siendo los planos estipulados por dos de las tres coordenadas X, Y o Z.

```
▼<UserData xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xsd="http://www.w3.org/2001/XMLSchema">
  ▼<Usuario>
     <nUsuario>anahervas</nUsuario>
     <angBrazoDrchXY>163.5331</angBrazoDrchXY>
     <angBrazoIzdaXY>153.191971</angBrazoIzdaXY>
     <angBrazoDrchYZ>90</angBrazoDrchYZ>
     <angBrazoIzdaYZ>90</angBrazoIzdaYZ>
     <angBrazoDrchYZatras>49.3280144</angBrazoDrchYZatras>
     <angBrazoIzdaYZatras>45</angBrazoIzdaYZatras>
     <angRodillaDrchYZ>74.08558</angRodillaDrchYZ>
     <angRodillaIzdaYZ>68.90153</angRodillaIzdaYZ>
     <angPiernaDrchXY>62.3813133</angPiernaDrchXY>
     <angPiernaIzdaXY>60.96273</angPiernaIzdaXY>
     <angTorsoDXY>32.07505</angTorsoDXY>
     <angTorsoIXY>29.0372715</angTorsoIXY>
     <puntuacion>2</puntuacion>
   </Usuario>
 </UserData>
```

Figura 4.3: Captura del fichero .../proyectorehabilitacion/Assets/datos_usuarios/anahervas24-7-2014.xml

Para guardar estos datos en principio se consideró utilizar una base de datos, pero en seguida se descartó la idea debido a que eran datos simples y estas otras están orientadas a sistemas más complejos, además del coste computacional que suponen.

Por lo tanto, para obtener estos datos hay que hacer que el usuario realice una serie de movimientos marcados por nosotros. Estos movimientos tienen que englobar todo el rango de posiciones que pueda adquirir la silueta de la cual tendrán que imitar la postura en la partida, para que nunca les salga una imposible para ellos.

Así, como vemos en la *figura 4.4*, la interfaz de esta escena estará compuesta por un plano en la parte izquierda de la pantalla que nos mostrará con imágenes las posiciones que el paciente deberá imitar y a partir de las cuales se tomarán las medidas de los ángulos. En la parte derecha, se muestra la imagen en tiempo real de lo que la Kinect está capturando, para que el propio paciente se vea a sí mismo. Como ya dijimos anteriormente, esto es algo muy útil para mejorar la propia percepción que tiene el usuario de su cuerpo, ya que algunos que han sufrido un ictus pueden tener esta visión distorsionada. En cualquier momento esta escena se puede pausar presionando la tecla “P” o podemos volver al menú inicial presionando “espacio”.

Figura 4.4: Captura de la escena Test del videojuego

Al terminar de realizar este test, se nos mostrará una pantalla como la de la *figura 4.5* con las capturas de todas nuestras posiciones para que el paciente o el fisioterapeuta vea cuál era la postura que se estaba realizando cuando el sistema calculó el ángulo de movimiento. Si alguna posición ha sido incorrecta, se podrá repetir simplemente haciendo clic con el ratón sobre ella, y a continuación volveremos a esa posición en el test.

Figura 4.5: Captura del final de la escena Test del videojuego

Será una vez que estemos conformes con todas las posiciones cuando terminará esta escena y se activará la opción de guardar los datos, creando así el fichero XML anteriormente mencionado o sobrescribiendo el ya existente si elegimos esta opción al principio de la escena.

-Creación del fichero XML

Inicialmente, para crear el fichero XML que mostramos en la *figura 4.3*, crearemos en el código la estructura que vamos a seguir. En nuestro caso no es más que la mostrada anteriormente, definiendo el tipo de datos que van a ir en cada campo y qué campos van dentro de los otros, como vemos en la *figura 4.6*:

```
10 //Datos que vamos a almacenar en el XML
11 public class UserData {
12     public DemoData Usuario;
13     public UserData() { }
14 
15     public struct DemoData
16     {
17         public String nUsuario;
18         public float angBrazoDrchXY;
19         public float angBrazoIzdaXY;
20         public float angBrazoDrchYZ;
21         public float angBrazoIzdaYZ;
22         public float angBrazoDrchYZatras;
23         public float angBrazoIzdaYZatras;
24         public float angRodillaDrchYZ;
25         public float angRodillaIzdaYZ;
26         public float angPiernaDrchXY;
27         public float angPiernaIzdaXY;
28         public float angTorsoDXY;
29         public float angTorsoIXY;
30         public int puntuacion;
31     }
32 }
```

Figura 4.6: Código fuente de la estructura que formará el XML en la escena test

Por lo tanto, nuestro código será sencillo, conteniendo dentro de la estructura "UserData" el campo principal de "Usuario", dentro del cual se encuentran cada uno de los datos mencionados ya anteriormente.

A continuación, comprobaremos que este usuario no es el primero en tener un registro de sus movimientos. En el caso de que lo sea, crearemos el directorio que contendrá todos los registros, ya que este no existirá previamente. Y en cualquiera de los dos casos, guardaremos el fichero cuando la opción *save* se active (al final del test) llevando éste de título el día, mes, año y nombre del usuario. Todo esto lo vemos en e código de la *figura 4.7*:

```
48     void Start () {
49         if(!Directory.Exists(_FileLocation +"/datos_usuarios")){ //Si el directorio no existe
50             Directory.CreateDirectory(_FileLocation +"/datos_usuarios"); //Se crea
51         }
52         datosUsuario = new UserData();
53         obj = GameObject.FindWithTag("codigoEscena");
54     }
55 
56     void Update () {
57         if (save){ //Se guardan los datos
58             save=false;
59             Dia=DateTime.Now.Day;
60             Mes=DateTime.Now.Month;
61             Ano=DateTime.Now.Year;
62             nombreUsuario=obj.GetComponent<obtencionAngulosMov>().nUser;
63             _FileName=nombreUsuario+Dia+"-"+Mes+"-"+Ano+".xml";
64                 //Se asigna la hora y la fecha actual como nombre de fichero
65             CreateXML();
66         }
```

Figura 4.7: Código fuente de la creación del directorio y el titulo del fichero

Finalmente, se creará el XML en la ruta establecida (dentro del directorio "datos_usuarios") justo después de ir obteniendo cada uno de los ángulos recogidos por otro *script* y poner el nombre de usuario y la puntuación inicial a 0, como vemos en la función de *ObtenerDatos()* de la *figura 4.8*. Una vez que tenemos esto, se procederá a serializar los datos para así poderlos escribir en el fichero con el formato XML deseado, ya que la serialización de datos consiste en convertir un objeto en una secuencia de bytes para almacenarlo o transmitirlo a memoria, una base de datos, o en un archivo y cuyo propósito principal es guardar el estado del objeto para poder crearlo de nuevo cuando se necesite (Microsoft®, 2014d).

```
79      void CreateXML() {
80          StreamWriter writer;
81          String path = _FileLocation+"/datos_usuarios/"+ _FileName;
82
83          if(!File.Exists(path)) { //Si el fichero no existe
84              ObtenerDatos(); //Se llama a la función donde se asignaran todos los datos al array de estructuras
85
86              // Serializacion
87              _data = SerializeObject(datosUsuario);
88              writer = File.CreateText(path);
89              writer.Write(_data);
90              writer.Close();
91          }
92      }
93
94      void ObtenerDatos() {
95        if (datosUsuario != null) {
96          datosUsuario.Usuario.nUsuario = nombreUsuario;
97      //  POSTURA 1
98          datosUsuario.Usuario.angBrazoDrchXY = obj.GetComponent<obtencionAngulosMov>().angElevacionBrazoD;
99          datosUsuario.Usuario.angBrazoIzdaXY = obj.GetComponent<obtencionAngulosMov>().angElevacionBrazoI;
100     //  POSTURA 2
101         datosUsuario.Usuario.angBrazoDrchYZ = obj.GetComponent<obtencionAngulosMov>().angElevacionBrazoDFront;
102         datosUsuario.Usuario.angBrazoIzdaYZ = obj.GetComponent<obtencionAngulosMov>().angElevacionBrazoIFront;
103     //  POSTURA 3
104         datosUsuario.Usuario.angBrazoDrchYZatras = obj.GetComponent<obtencionAngulosMov>().angElevacionBrazoDAtras;
105         datosUsuario.Usuario.angBrazoIzdaYZatras = obj.GetComponent<obtencionAngulosMov>().angElevacionBrazoIAtras;
106     //  POSTURA 4
107         datosUsuario.Usuario.angRodillaDrchYZ = obj.GetComponent<obtencionAngulosMov>().angElevacionRodillaD;
108     //  POSTURA 5
109         datosUsuario.Usuario.angRodillaIzdaYZ = obj.GetComponent<obtencionAngulosMov>().angElevacionRodillaI;
110     //  POSTURA 6
111         datosUsuario.Usuario.angPiernaDrchXY = obj.GetComponent<obtencionAngulosMov>().angElevacionPiernaDLado;
112     //  POSTURA 7
113         datosUsuario.Usuario.angPiernaIzdaXY = obj.GetComponent<obtencionAngulosMov>().angElevacionPiernaILado;
114     //  POSTURA 8
115         datosUsuario.Usuario.angTorsoDXY = obj.GetComponent<obtencionAngulosMov>().angTorsoD;
116     //  POSTURA 9
117         datosUsuario.Usuario.angTorsoIXY = obj.GetComponent<obtencionAngulosMov>().angTorsoI;
118         datosUsuario.Usuario.puntuacion = 0;
119       }
120     }
```

Figura 4.8: Código fuente de la recolección de datos y escritura en el XML

Hecho esto, ya tendríamos nuestro fichero permanente guardado para ser utilizado por cualquier otra escena del videojuego o para ver los datos siempre que lo deseemos.

-Mecanismo de cálculo de los ángulos de movilidad

Para el cálculo de los ángulos de movimiento conseguidos por el paciente, lo primero es capturar el cuerpo de éste mediante un proceso denominado *tracking* o seguimiento que se realiza en tiempo real, lo que nos permitiría ver en todo momento los ángulos que estamos calculando. Gracias a esto la Kinect va a recoger las posiciones de cada parte del cuerpo y las rotaciones de las articulaciones que se producen en cada momento.

En nuestro caso, esto está reflejado en una figura que contiene un bloque en cada una de las articulaciones del cuerpo capturado, pero a la cual hemos hecho invisible para que no molestase en la visibilidad y estética de la escena. En consecuencia, en las características de cada uno de los bloques nos vendrán reflejadas las posiciones que van adquiriendo, que hemos hecho que sean relativas a un mismo punto para que sea más sencillo a la hora de compararlas unas con otras para calcular los ángulos deseados.

Antes de explicar el propio mecanismo de cálculo de los ángulos, mostramos las diferentes posturas que se requieren del paciente para poder tener unos límites establecidos dentro del juego según sus posibilidades, y los datos que se recogen de cada posición:

- POSTURA 1ª:
  - Posición de T.
    - Ángulo de elevación del brazo derecho con respecto al plano X-Y.
    - Ángulo de elevación del brazo izquierdo con respecto al plano X-Y.
  - Brazos hacia arriba.
    - Si llega con el brazo derecho a subirlo hasta arriba.
    - Si llega con el brazo izquierdo a subirlo hasta arriba.

- POSTURA 2ª: Brazos hacia delante.
  - Ángulo de elevación del brazo derecho con respecto al plano Y-Z.
  - Ángulo de elevación del brazo izquierdo con respecto al plano Y-Z.

- POSTURA 3ª: Brazos hacia atrás.
  - Ángulo de elevación del brazo derecho con respecto al plano Y-Z.
  - Ángulo de elevación del brazo izquierdo con respecto al plano Y-Z.

- POSTURA 4ª: Doblar el torso hacia la derecha.
  - Ángulo de torsión del torso hacia la derecha en el plano X-Y.

- POSTURA 5ª: Doblar el torso hacia la izquierda.
  - Ángulo de torsión del torso hacia la izquierda en el plano X-Y.

- POSTURA 6ª: Levantar rodilla derecha.
  - Ángulo de elevación de la rodilla derecha en el plano Y-Z.

- POSTURA 7ª: Levantar rodilla izquierda.
  - Ángulo de elevación de la rodilla izquierda en el plano Y-Z.

- POSTURA 8ª: Levantar pierna derecha hacia el lateral.
  - Ángulo de elevación de la pierna derecha en el plano X-Y.

- POSTURA 9ª: Levantar pierna izquierda hacia el lateral.
  - Ángulo de elevación de la pierna izquierda en el plano X-Y.

Las posturas de elevación de piernas se cancelan si se detecta al inicio del test que el usuario está sentado, es decir, que usa silla de ruedas, por medio de la detección de que la altura de las rodillas y de la cadera es similar.

Así, en cada postura se tomarán dos articulaciones entre las cuales se creará un vector imaginario, del cual se extraerán sus dos componentes necesarias de las tres posibles (X, Y o Z) para calcular el ángulo por medio de la trigonometría. Para contar este proceso lo haremos explicando cómo se realiza en la postura 1, es decir, en la elevación de los brazos en el plano XY.

El ángulo que vamos a calcular en este caso va desde la línea imaginaria vertical que pasa por el hombro (eje Y) hasta la línea imaginaria que va del hombro hasta la altura del codo (eje X), por lo que podría decirse que es el ángulo que forma la axila, como vemos en la *figura 4.9*.

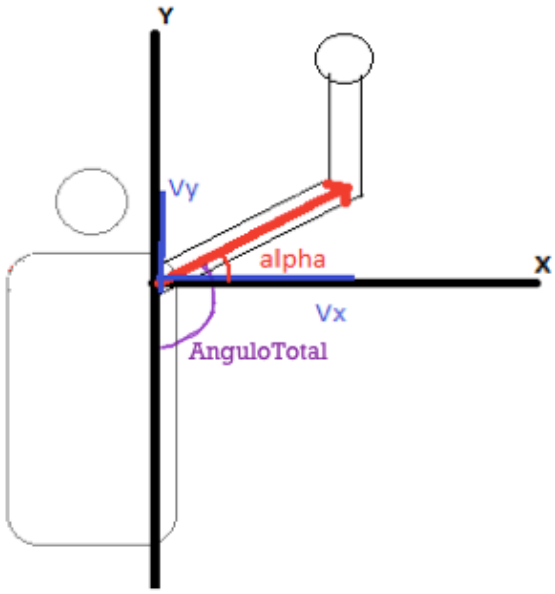


Figura 4.9: Componentes del ángulo de elevación del brazo derecho por encima del hombro

En este caso querremos calcular el ángulo *alpha* formado por el vector X (Vx) y el vector Y (Vy) y a éste sumarle los 90º pertenecientes al cuadrante inferior derecho para así obtener el ángulo "AnguloTotal". Por lo tanto, haremos las siguientes comparaciones:

- Si posición X del codo > posición X del hombro = El brazo es el derecho.
- Si posición Y del codo >= posición Y del hombro = El codo está hacia arriba (se eleva más de 90º).

- *alpha* = arctan (Vy, Vx)
- AnguloTotal = 90º + *alpha*.

Sin embargo, mientras detectamos el brazo derecho, puede que éste se encuentre por debajo de los 90º grados como vemos en la *figura 4.10*. Entonces a partir del primer paso anterior (el brazo es el derecho), seguiremos los siguientes:

- Si posición Y del codo < posición Y del hombro = El codo está hacia abajo (se eleva menos de 90º).
- *alpha* = arctan (Vy, Vx)
- AnguloTotal = 90º - *alpha*.

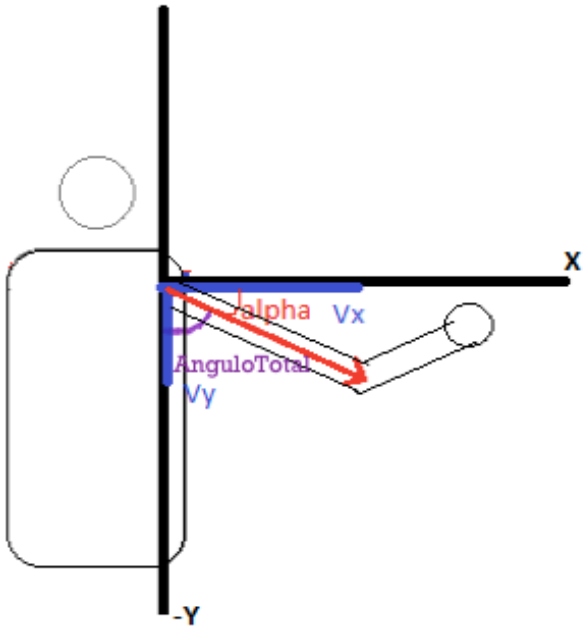


Figura 4.10: Componentes del ángulo de elevación del brazo derecho por debajo del hombro

Si por el contrario es el brazo izquierdo el que se quiere analizar, la forma de calcular los ángulos será igual que con el derecho pero cambiando la comprobación inicial de forma que si la posición X del codo es mayor que la posición X del hombro, el brazo es el izquierdo. La forma de calcular esto con código en C# es la de la *figura 4.11*.

Lo mismo se hará con los demás ángulos del cuerpo, sólo que en las piernas no habrá que hacer diferencia entre si la analizada será la derecha o la izquierda, ya que no se pueden elevar las dos a la vez, si no que la medida de cada una se toma en una posición diferente. Por lo tanto, en todas se calculará el ángulo de movimiento respecto a un eje vertical imaginario.

```
236     public void brazosArriba(Boolean repetir) {
237         timer -= Time.deltaTime;
238         if(timer < tiempoConfig/2f) {
239             c=4;
240         }
241 
242         vXD = Mathf.Abs (hombroDX - codoDX);
243         vYD = Mathf.Abs (hombroDY - codoDY);
244         vXI = Mathf.Abs (hombroIX - codoIX);
245         vYI = Mathf.Abs (hombroIY - codoIY);
246 
247         ALPHAD = Mathf.Atan2 (vYD, vXD) * Mathf.Rad2Deg;
248         ALPHAI = Mathf.Atan2 (vYI, vXI) * Mathf.Rad2Deg;
249 
250         if (codoDY >= hombroDY)
251                 angElevacionBrazoD = 90 + ALPHAD;
252         else
253                 angElevacionBrazoD = 90 - ALPHAD;
254         if (codoIY >= hombroIY)
255                 angElevacionBrazoI = 90 + ALPHAI;
256         else
257                 angElevacionBrazoI = 90 - ALPHAI;
258         
259         Posicion1.SetPixels32 (ZigInput.Image.data);
260         Posicion1.Apply (true);
261 
262         if (timer < 0) {
263             if (repetir) {
264                 c = 16;
265             } else {
266                 c = 5;
267             }
268             timer = tiempoConfig;
269         }
270     }
```

Figura 4.11: Código fuente del cálculo de los ángulos de elevación de los brazos

4.2.3. Escena pasarela

Ésta es la escena que compone la segunda opción del menú. Su finalidad es la de la propia rehabilitación del paciente por medio de una serie de siluetas que van a ir apareciendo por la pasarela y de las cuales habrá que imitar su postura con la mayor fidelidad posible.

Es una escena compleja formada por dos personajes iniciales, entre los cuales el paciente elegirá uno para jugar, un escenario con una pasarela y gradas con público en donde se desarrollará la "competición" y una silueta que irá tomando las posiciones indicadas y se irá aproximando al jugador a través de esta pasarela.

La escena comenzará pidiendo al jugador que introduzca su nombre de usuario y su género para poder elegir entre los dos personajes. El introducir su nombre será necesario para poder cargar los datos que haya en el fichero XML de dicho usuario, por lo que si no existe dicho usuario, no se podrá jugar la partida y deberá crearlo realizando el test de movilidad desarrollado en el apartado anterior.

Como ya mencionamos, el test es imprescindible para conocer el rango de movimiento de cada paciente, por lo que es personal de cada jugador. Esto sirve para aplicar las posiciones adecuadas a la silueta blanca que se aproxima al paciente, como vemos en la *figura 4.12*, las cuales se aplicarán de manera semi-aleatoria, lo que quiere decir que serán posiciones calculadas aleatoriamente pero las cuales estarán limitadas para que no salgan posiciones naturalmente imposibles, como por ejemplo las dos piernas levantadas a la vez, posiciones

imposibles para el paciente, como levantar una extremidad más de lo establecido en el test, o que las piernas estén fijas para pacientes en silla de ruedas.

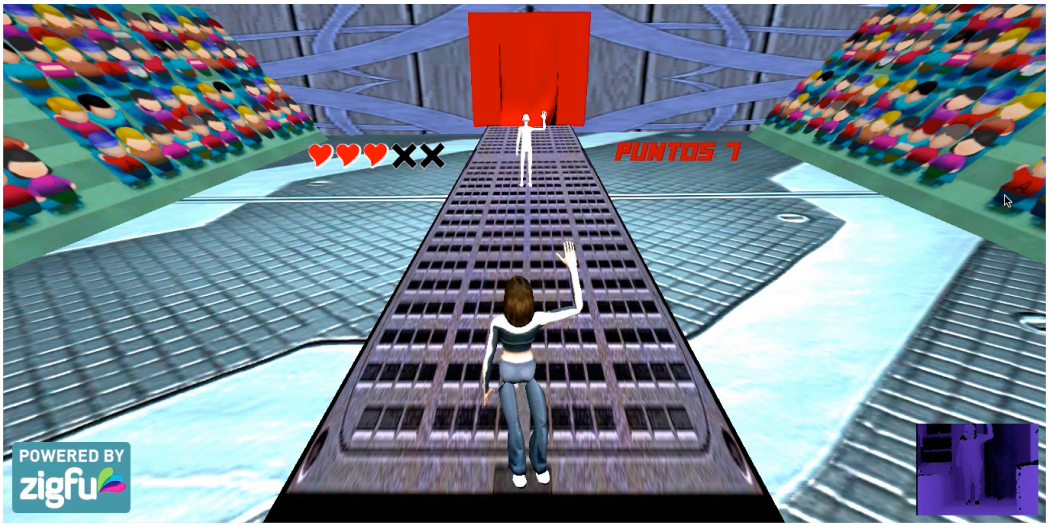


Figura 4.12: Captura de la escena Pasarela del videojuego

Para valorar la cantidad de fallos o aciertos del jugador, usamos un sistema de vidas y puntos los cuales se rigen por los elementos denominados *colliders*. Estos serán unos cubos invisibles cuyo área encerrado será el área de colisión y se encuentran en cada articulación tanto del personaje del usuario como de la silueta. Por lo tanto, se contabilizará el número de colisiones correctas, es decir, que cada vez que la silueta atraviese el cuerpo del usuario se comprobará con cada *collider* que el de la silueta ha "chocado" con el correspondiente del usuario. Según esto, sumaremos 3 puntos si se han acertados todos (la posición ha sido perfecta), sumaremos 1 punto si se ha fallado alguno dentro de los permitidos por la configuración o restaremos una vida si se fallan más de los permitidos. Vemos esta parte desarrollada en código en la *figura 4.13*.

```
347     if (objCaderaDPersonaje.GetComponent<colisionCaderaDrch>().colCaderaD==false) error += 1;
348     if (objCaderaIPersonaje.GetComponent<colisionCaderaIzqd>().colCaderaI==false) error += 1;
349     if (objHombroDPersonaje.GetComponent<colisionHombroDrch>().colHombroD==false) error += 1;
350     if (objHombroIPersonaje.GetComponent<colisionHombroIzqd>().colHombroI==false) error += 1;
351     if (objManoDPersonaje.GetComponent<colisionManoDrch>().colManoD==false) error += 1;
352     if (objManoIPersonaje.GetComponent<colisionManoIzqd>().colManoI==false) error += 1;
353     if (objPieDPersonaje.GetComponent<colisionPieDrch>().colPieD==false) error += 1;
354     if (objPieIPersonaje.GetComponent<colisionPieIzqd>().colPieI==false) error += 1;
355     if (objRodillaDPersonaje.GetComponent<colisionRodillaDrch>().colRodillaD==false) error += 1;
356     if (objRodillaIPersonaje.GetComponent<colisionRodillaIzqd>().colRodillaI==false) error += 1;
357     if (objCodoDPersonaje.GetComponent<colisionCodoDrch>().colCodoD==false) error += 1;
358     if (objCodoIPersonaje.GetComponent<colisionCodoIzqd>().colCodoI==false) error += 1;
359
360
361     if (error > jointsFallidos) { //Cuando + de los joints que se pueden fallar no se han ajustado a la postura,-1vida
362             vidas--;
363             audio.PlayOneShot(fallo, 1F);
364             extra = 0;
365     }else if (error == 0) { //Cuando todo el cuerpo se ha ajustado a la postura, 3ptos
366             puntos += 3;
367             audio.PlayOneShot(aplausos, 1F);
368             extra++;
369             if (extra == 5){
370                 if(vidas < 5)   vidas++;    //Si se hacen bien 5 posiciones seguidas, vida extra
371             }else if (extra == 8){
372                 puntos += 5;    //Si se hacen bien 3 posiciones sequigas despues de ganar una vida, puntos extra
373                 extra = 0;
374             }
375     } else { //Si se fallan como mucho los joints permitidos segun la config., +1pto
376             extra = 0;
377             if (noFallos){
378                 vidas--;
379                 audio.PlayOneShot(fallo, 1F);
380             }else   puntos++;
381     }
```

Figura 4.13: Captura del código del sistema de vidas y puntos de la escena Pasarela

Por lo tanto, para grabar estas posiciones de las que hemos hablado que se asignan a la silueta, se grabaron los valores de las rotaciones de las posturas de cada parte de nuestro cuerpo que nos interesaran para que el paciente las imitara, cada una con un rango de amplitud y, por lo cual, de dificultad.

Cada una de estas rotaciones se guardó en un array de valores y con un orden determinado, que a continuación se proceden a guardar en un XML permanente para después poder acceder a ellas, de manera que a la hora de programar cada nivel de dificultad del juego o a la hora de eliminar posiciones que el paciente no podía realizar, se fuera limitando el rango de valores aleatorios para excluir a estas posiciones de esta aleatoriedad. Así, el valor aleatorio de cada extremidad será uno diferente en cada posición, por lo que irán surgiendo posturas completamente diferentes en cada nueva iteración.

Como la escena consta de diferentes niveles de juego, en cada uno de ellos se harán ciertas limitaciones para ir aumentando la dificultad y ofrecer las posiciones deseadas según de lo que trate cada nivel.

Finalmente, la partida termina cuando el usuario se queda sin ninguna de las 5 vidas con las que empieza. Entonces se indica el final de ésta y se va automáticamente al menú principal, guardándose la puntuación obtenida en el XML del usuario en el caso de que haya sido la más alta que ha obtenido.

-Coger datos de un XML

Anteriormente explicamos cómo creábamos un fichero XML para almacenar unos datos que queríamos conservar de manera permanente para poder utilizar después. Pues bien, ahora será el momento en que queramos utilizar varios datos almacenados en estos ficheros.

En este caso querremos obtener todos los valores posibles de las rotaciones de las articulaciones para aplicar a la silueta (los valores que grabamos) y los ángulos máximos de movimiento de cada usuario para poder hacer los filtros pertinentes en las posiciones.

Así, en primer lugar obtendremos los datos que obtuvimos en el test de movilidad, que se encuentran en el XML con la estructura mostrada en la *figura 4.3* y los obtendremos de la forma que vemos en la *figura 4.14*.

```
79      void LoadXML()
80      {
81          XmlDocument xDoc = new XmlDocument();
82          xDoc.Load(_FileLocation+"/datos_usuarios/"+nFichero);
83
84          XmlNodeList user = xDoc.GetElementsByTagName("Usuario");
85          XmlNodeList name = ((XmlElement)user[0]).GetElementsByTagName("nUsuario");
86          foreach (XmlElement nodo in name) {
87              User.nUsuarior = nodo.InnerText;
88          }
89          XmlNodeList brazoDXY = ((XmlElement)user[0]).GetElementsByTagName("angBrazoDrchXY");
90          foreach (XmlElement nodo in brazoDXY) {
91              User.angBrazoDrchXYr = System.Convert.ToSingle(nodo.InnerText);
92          }
93          XmlNodeList brazoIXY = ((XmlElement)user[0]).GetElementsByTagName("angBrazoIzdaXY");
94          foreach (XmlElement nodo in brazoIXY) {
95              User.angBrazoIzdaXYr = System.Convert.ToSingle(nodo.InnerText);
96          }
```

Figura 4.14: Captura del código de recolección de datos de un XML

Como vemos, la función de la que hemos hecho la captura lo primero que hace es cargar el fichero XML con la ruta indicada. A partir de aquí iremos creando los llamados nodos de lista, los cuales pueden contener diferentes elementos o datos, aunque en este caso sólo habrá un elemento en cada nodo (en la imagen, nombre del usuario, ángulo de elevación del brazo derecho o ángulo de elevación del brazo izquierdo).

Por otro lado, también podemos encontrarnos documentos XML que contengan varios elementos en cada nodo, como es el caso del fichero que contiene todas las posiciones posibles de la silueta y que vemos en la *figura 4.15*.

Figura 4.15: Captura del fichero .../proyectorehabilitacion/Assets/PosicionesSilueta.xml

En este caso, tendremos que acceder a cada uno de ellos con un bucle *for* y almacenar cada dato posible del mismo nodo (en la *figura 4.16* las rotaciones del brazo derecho en el eje X) en un array de elementos a los que, como explicamos anteriormente, accederemos uno a uno de manera semi-aleatoria.

```
529 /*BRAZO DRCH*/ XmlNodeList brazoDrX = ((XmlElement)silueta[0]).GetElementsByTagName("brazoDrotX");
530         XmlNodeList valoresBDRX = ((XmlElement)brazoDrX[0]).GetElementsByTagName("float");
531         for (int BDRX=0; BDRX < valoresBDRX.Count; BDRX++) {
532             posSilueta.Silueta.brazoDrotXr[BDRX] = System.Convert.ToSingle(valoresBDRX[BDRX].InnerText);
533         }
```

Figura 4.16: Captura del código de recolección de datos de un XML con varios elementos en un nodo

-Niveles del juego

Esta escena cuenta con varios niveles de juego, los cuales exigirán unas actividades diferentes al paciente de forma que si las va superando, accederá al siguiente nivel que será más complicado. Así encontramos los 6 siguientes:

-Nivel 1: Este nivel se encargará de dar una posición aleatoria a tan sólo un brazo o al torso. Su finalidad es la de hacer realizar al usuario una especie de calentamiento antes de pedirle que adopte posiciones más complejas y le servirá como refuerzo positivo al permitirle conseguir puntos de manera relativamente sencilla al inicio del juego, mientras entrena la movilidad de sus brazos.

-Nivel 2: Este nivel se encarga de evaluar el equilibrio del usuario, por lo que su finalidad se consigue haciendo que éste ponga sus piernas en diferentes posiciones.

-Nivel 3: Este nivel se encarga de que el usuario se adapte a posiciones sencillas. Su finalidad es la de mejorar la coordinación del paciente de manera progresiva, ya que se darán solamente valores aleatorios a dos de sus extremidades.

-Nivel 4: Este nivel ofrecerá posiciones completamente aleatorias al usuario. Su finalidad es la de continuar mejorando la coordinación que se comenzó en el nivel anterior, de un modo más complicado a dar la aleatoriedad a todas las extremidades.

-Nivel 5: Este nivel será igual que el anterior pero añadiendo la dificultad de que puede que alguna extremidad tenga un ángulo de movimiento levemente mayor que el conseguido en el test. Su finalidad es la de continuar con la mejora de la coordinación del usuario pero forzándole a que haga algo más allá de lo que está dentro de sus capacidades anteriores a la partida.

-Nivel 6: Este nivel ofrecerá posiciones aleatorias y en las que no se puede tener ninguna imprecisión. Su finalidad es la de continuar mejorando la coordinación del usuario, al igual que cómo se hacía en el nivel 4 pero obligándole a acertar con absolutamente todas las partes de su cuerpo, ya que si no se le restará siempre una vida. Además, se añade la dificultad de que cada 10 posiciones el nivel de precisión va a ser menor (disminuye el tamaño de los *collider*).

4.2.4. Escena ranking

Esta es la escena que compone la tercera opción del menú. Su finalidad es mostrar las 10 mejores puntuaciones de entre todos los usuarios que están registrados en el juego, creando así un ambiente de competitividad sana, un pequeño grado de adicción y un impulso a la autosuperación.

Su interfaz es la que se muestra en la *figura 4.17* y, en primer lugar, está formada por un plano superior que incluye el título de esta escena. También encontramos la mano que selecciona el botón inferior de "OK" cuando queramos salir de la escena, cuyo mecanismo es exactamente igual a la mano de la escena inicio, y aunque de nuevo esto lo podemos hacer presionando la tecla de "espacio" para mayor facilidad.

Figura 4.17: Captura de la escena Ranking del videojuego

Pero lo más interesante de esta escena son las puntuaciones que se muestran. Éstas están ordenadas de mayor a menor junto con el nombre del usuario al que corresponden y su posición en el ranking. El algoritmo de ordenación implementado lo vemos en la *figura 4.18*, y está basado en ir comparando los valores uno a uno y en cada comparación poner el mayor en la primera posición del array de puntuaciones.

```
59    void ordenarPuntuacion () {
60        for(i=0; i<(ficheros.Length-1); i++){
61            for(int j=i+1; j<ficheros.Length; j++){
62                if(puntos[i]<puntos[j]){    //Pomenos en primera posicion del array la puntuacion mas alta
63                    int aux = puntos[i];
64                    puntos[i] = puntos[j];
65                    puntos[j] = aux;
66                    string naux = nombre[i]; //Nombre de usuario a la misma posicion del array que su puntuacion
67                    nombre[i] = nombre[j];
68                    nombre[j] = naux;
69                }
70            }
71        }
72    }
```

Figura 4.18: Código fuente de la de la ordenación de puntuaciones

Estos valores se han obtenido previamente accediendo a los ficheros XML de cada usuario existentes, concretamente al último apartado que es el que contiene la puntuación, y al primer apartado para los nombres de usuario que van en la clasificación, y se muestran por pantalla gracias a la función *OnGUI()* en 3 columnas diferentes: posición, nombre de usuario y puntos.

4.2.5. Escena configuración

Esta escena es una de por las cuales está compuesto el submenú del menú principal. Su finalidad es modificar ciertos parámetros del juego de manera que éste se pueda volver más "jugable", se aumente o disminuya la dificultad según las capacidades o necesidades del paciente o se pueda hacer más adaptable.

Así pues, esta escena mostrada en la *figura 4.19* surgió por la necesidad de que el fisioterapeuta pudiera tener mayor influencia sobre el juego y así adaptar los ejercicios a la rehabilitación necesaria para el paciente. Por otro lado, con la configuración se podría utilizar para disminuir o aumentar la dificultad que ya el juego tiene de por sí, para hacerlo apto tanto para pacientes que les cueste más imitar las posiciones que van apareciendo como a los que lo hacen con facilidad. Este aspecto es muy importante ya que nuestro objetivo es que el videojuego sea extensible para el mayor público posible.

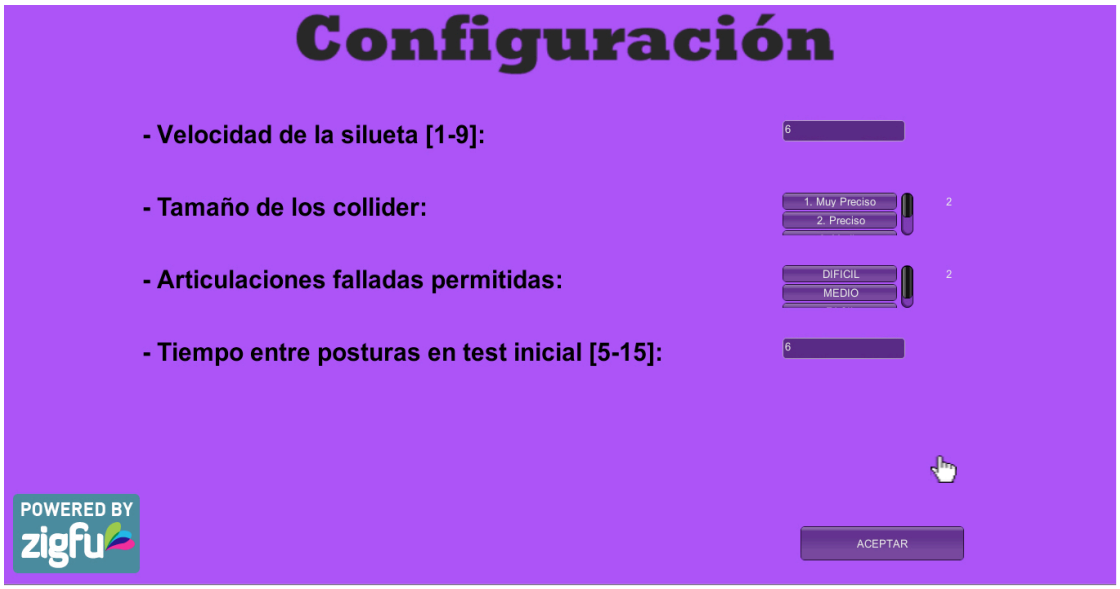


Figura 4.19: Captura de la escena Configuración del videojuego

Como vemos, es una escena simple formada por un plano con el título, un botón de "Aceptar" para cuando estemos conformes con los valores escogidos, y cuatro elementos a la derecha de cada línea de texto, compuestos por dos cajas llamadas "TextArea" para introducir los valores que queramos escoger y dos cuadros con un *scroll* para elegir entre las diferentes opciones posibles a través de un botón. Para seleccionar todo esto contamos con el cursor, en este caso con una textura de una mano con el dedo índice extendido adjunta.

Dicho esto, la configuración puede llegar a contener un montón de parámetros para variar, pero nos centramos en los cuatro implementados que consideramos principales:

- Velocidad de la silueta: La velocidad a la cual se acerca la figura de la escena pasarela, la cual tendrá que ser imitada por el paciente, puede ser modificada. Para los pacientes que les cueste más interpretar esta posición, por mala visión o por problemas en la percepción, o que tengan que realizar un trabajo mayor para adaptar su cuerpo, son algunos ejemplos de usuarios para los que sería ideal reducir la velocidad de aproximación de la silueta. Sin embargo otros que no tengan estos problemas o que quieran aumentar el nivel de su entrenamiento podrán, por el contrario, aumentarla. Esta velocidad ha sido limitada del valor 1 al 9, siendo el 1 la velocidad de aproximación más lenta y el 9 la más rápida.

- Tamaño de los *collider*: Concretamente esto se refiere a la precisión de la postura que el usuario quiera o deba adaptar. Así el nivel de mayor precisión obliga al paciente a calcar la silueta con gran exactitud para dársela por buena, y el nivel menor, permite algo de separación entre las partes del usuario y las de la figura. Los niveles de precisión serán cinco.

- Articulaciones falladas permitidas: Esto quiere decir el número de *collider* que el usuario podrá fallar como máximo para que no se le resten puntos, ya que cada uno de estos se encuentra en una articulación. Así pues, en cada posición se podrán fallar las articulaciones que permitamos con este valor y sumaremos un punto y no se nos quitará vida. Ésta se nos restará cuando fallemos más, a la vez que no se nos sumará ningún punto. Los niveles permitidos son fácil (hasta 2 articulaciones fallidas), medio (como mucho 1) y difícil (no se permite ningún fallo).

- Tiempo entre posturas del test inicial: De nuevo para ayudar a las personas que les cueste mayor esfuerzo entender lo que el juego les pide o adaptar su cuerpo. Así cada posición en el test de movimientos permanecerá más o menos tiempo en pantalla, permitiendo variar desde 5 hasta 15 segundos de duración.

Finalmente, cuando estemos satisfechos con los valores, pulsaremos el botón de aceptar. A partir de aquí se comprobará si todos los valores son correctos para de nuevo volver al menú principal. En caso contrario, se nos mostrará un mensaje de error y no saldremos de la pantalla hasta que los datos introducidos no estén dentro de los rangos establecidos.

Una vez hecho esto con éxito, todos estos valores se van a actualizar en el fichero XML que los va a guardar de forma permanente para que se puedan recoger y utilizar en las escenas adecuadas para aplicar estos valores a los objetos correspondientes.

4.2.6. Escenas de instrucciones y créditos

Estas dos escenas forman parte del submenú del menú principal. La finalidad de la primera es la de explicar al paciente el procedimiento para poder jugar de manera óptima y mostrarle la consistencia de cada uno de los 6 niveles de juego. La segunda, es una escena muy simple la cual indica por quién ha sido creado el videojuego.

En la primera de estas dos escenas, que vemos en la *figura 4.20*, se recogen las instrucciones de unos ficheros de texto que están guardados en una carpeta del juego. Éstas se sacarán por pantalla poco a poco, permitiendo al usuario ir avanzando por medio de la gesticulación o con las flechas del teclado (de nuevo ofreciendo más posibilidades a los usuarios, ya que recordamos que tratamos con pacientes con problemas de movilidad).

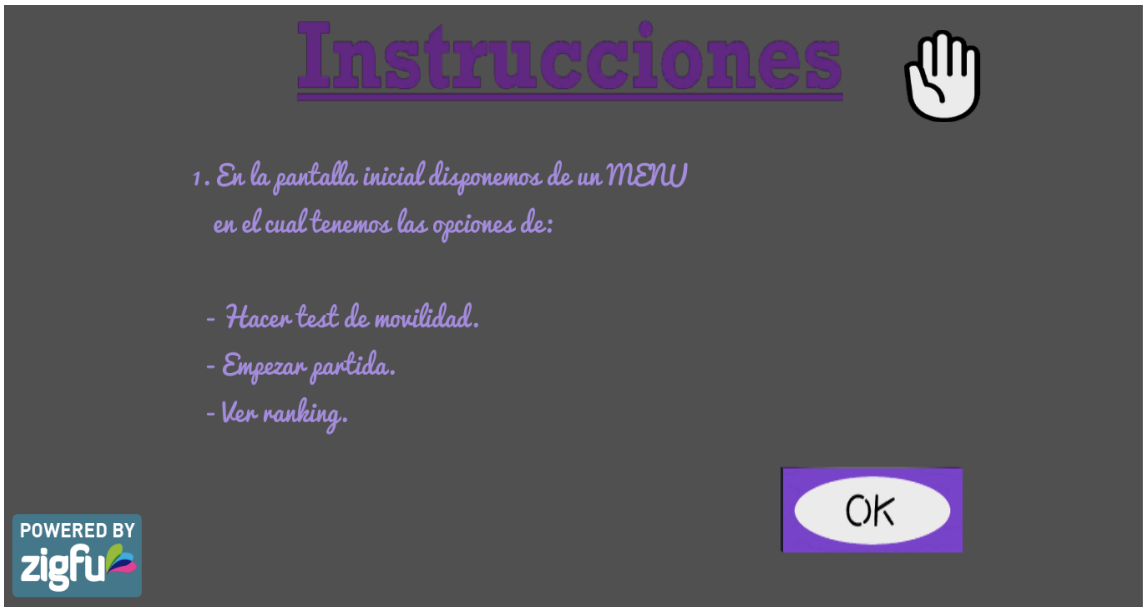


Figura 4.20: Captura de la escena Instrucciones del videojuego

La gesticulación consiste en mover la mano de derecha a izquierda para avanzar y de izquierda a derecha para retroceder en las instrucciones, de la misma forma que si pasásemos las hojas de un libro. Esta detección del movimiento es posible gracias a la función de Zigfu *DoSwipe(string direction)*, la cual detectará si la mano se mueve hacia la izquierda o la derecha (parámetro contenido en la variable *direction*) y así avanzar o retroceder. Es decir, cada fichero que contiene una parte de las instrucciones del juego tiene un número en su nombre en sentido ascendente y, a su vez, según avancemos (movamos la mano hacia la izquierda) habrá una variable que contenga un entero que irá aumentando. Así, según el valor de esta variable se cargará el fichero que la contenga en su título, manteniendo siempre un orden coherente. Esta misma variable irá aumentando cada vez que presionemos la tecla "avanzar" o disminuyendo con la tecla "retroceder".

La segunda escena es aun más simple que la primera de éstas, y es la escena de créditos que se muestra en la *figura 4.21*. Ésta muestra el nombre de la creadora de este Trabajo Fin de Grado por medio de la función *OnGUI()*, junto a otros dos planos inferiores que contienen el escudo de la Escuela Técnica Superior de Ingenieros de Telecomunicación y del Grupo de Telemática e Imagen de la Universidad de Valladolid.

Figura 4.21: Captura de la escena Créditos del videojuego

Ambas escenas de nuevo contarán con la mano que seleccionará el botón de "OK" para salir de la escena al menú principal o mediante la presión de la tecla "espacio".

4.3. Funcionamiento

Si nos disponemos a jugar al videojuego, es la primera vez que lo vamos a hacer y tenemos este documento en nuestras manos, éste es el apartado que necesitamos revisar. Si no disponemos de este documento, ya dijimos anteriormente que dentro del propio juego hay una opción en el submenú que nos muestra una pantalla con las instrucciones del juego, las cuales procedemos a desarrollar a continuación.

En primer lugar, desde el menú principal, procedemos a elegir la opción que deseemos de las 6 posibles (3 principales y 3 secundarias).

Si es la primera vez que usamos el videojuego, no tenemos un registro de usuario o tenemos uno pero queremos sobrescribirlo, debemos elegir la opción primera de "HACER TEST DE MOVILIDAD", la cual estamos probando en la *figura 4.22*. En ésta el paciente tiene que representar las posiciones que le indican las figuras que van apareciendo por pantalla, siempre que pueda lograrlas. Estas posturas se utilizarán para mostrar las figuras adecuadas a imitar durante la partida y para tener un registro de los ángulos de movimiento del usuario.

Al finalizar esta escena, se podrá seleccionar la postura de la cual no estemos satisfechos y así repetirla. Para ello simplemente habrá que hacer clic encima de la captura de esa posición. Cuando estemos satisfechos, pinchamos con el ratón sobre "Test completado" y esto nos llevará de nuevo al menú principal.

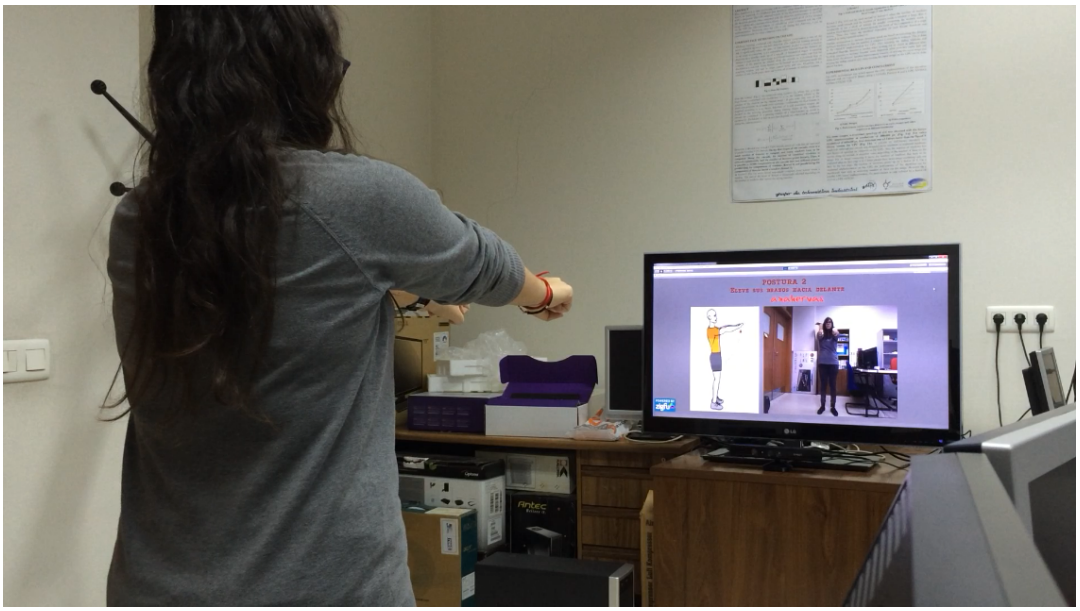


Figura 4.22: Probando la escena Test del juego

Una vez que estemos en el menú principal y tengamos nuestro registro de usuario, elegiremos la segunda opción llamada "EMPEZAR PARTIDA". Ésta nos pedirá introducir nuestro nombre de usuario y a continuación se comenzará a jugar imitando con la mayor fidelidad las posiciones que toma la silueta, posiciones las cuales tomarán valores adecuados al paciente según su registro.

Este juego consta de los 6 niveles siguientes:

-Nivel 1: Brazos y torso. Posiciones con tan solo un brazo o el torso aleatorios.

-Nivel 2: Equilibrio. Posiciones aleatorias de las piernas.

-Nivel 3: Coordinación. Posiciones en las que hay 2 o 3 extremidades aleatorias.

-Nivel 4: Posiciones aleatorias. Posiciones completamente aleatorias.

-Nivel 5: Mayor movilidad. Posiciones con posibles ángulos mayores de movimiento que los del test.

-Nivel 6: Posiciones precisas. Posiciones aleatorias con precisión cada vez mayor y no se puede fallar ninguna articulación.

Además de estos niveles, el videojuego posee un sistema de vidas y puntos que es el siguiente:

- Se suman 3 puntos si se aciertan todas las articulaciones con la silueta.
- Se suma 1 punto si se falla alguna articulación pero la configuración elegida lo permite.
- Se resta una vida si se fallan más articulaciones de las permitidas en la configuración.

- Se suma una vida extra si se aciertan 5 posturas seguidas perfectas, hasta un máximo de 5 vidas, que son las iniciales.
- Se suman 5 puntos extra si se aciertan 8 posturas seguidas perfectas, hasta un máximo de 5 vidas.
- La partida se termina cuando se nos acaban las 5 vidas, y la puntuación que hayamos obtenido se guarda si es la mayor de las conseguidas.

La *figura 4.23* nos muestra jugando con esta escena, exactamente en el nivel 1, imitando la posición del brazo izquierdo estirado hacia delante y el resto del cuerpo en posición de reposo.

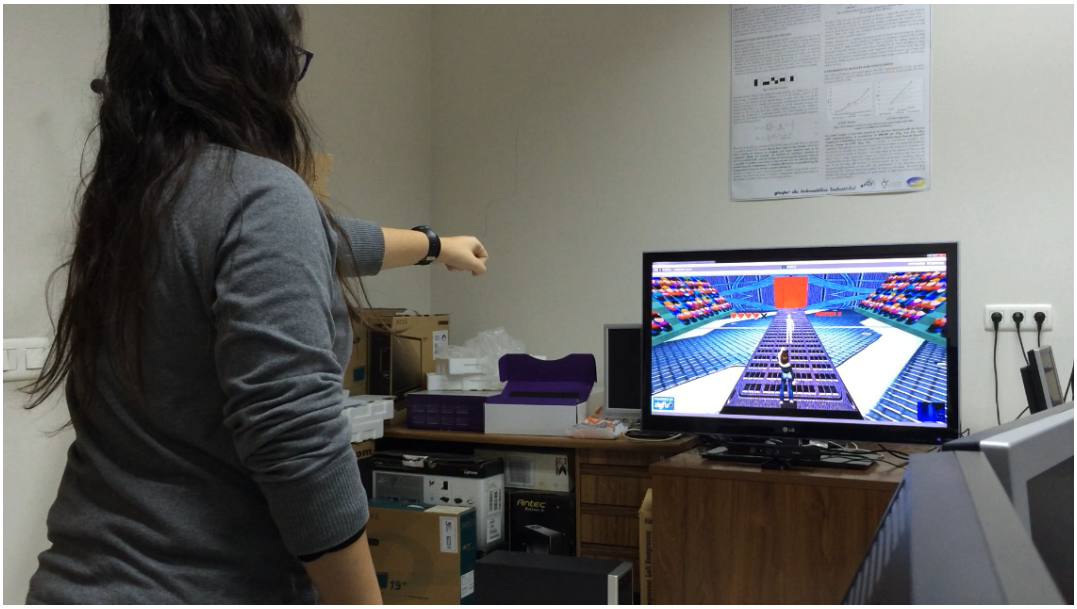


Figura 4.23: Probando la escena Partida del juego

La puntuación máxima que se guarda de cada partida podemos verla entrando a la tercera opción del menú principal, que es la de "VER RANKING", siempre que esté entre las 10 mejores. Aquí veremos a los 10 mejores usuarios junto con su puntuación máxima y la posición en el ranking que ocupan.

De nuevo en el menú principal, tenemos otras 3 opciones a las cuales accederemos presionando la letra correspondiente. Si presionamos la tecla "C" entraremos a la configuración del juego. Son 4 los parámetros que podemos cambiar en ella. En primer lugar la velocidad a la cual se nos acerca la silueta en la escena pasarela, teniendo un rango de 1 (muy lenta) a 9 (muy rápida). En segundo lugar tenemos el tamaño de los *collider*, es decir, la precisión a la hora de imitar a la silueta (a mayor tamaño, menor precisión), teniendo 5 niveles posibles desde muy poco preciso a muy preciso. En tercer lugar encontramos el número máximo de articulaciones o *collider* que podemos fallar, pudiendo escoger que sean 2 (nivel fácil), 1 (nivel medio) o ninguna (nivel difícil). Y como última opción tenemos para elegir los segundos entre las posturas del test inicial, para que se puedan adaptar a la capacidad de reacción de cada paciente, pudiendo variarlos entre 5 y 15 segundos. Cuando tengamos los valores deseados, pulsaremos con el ratón en el botón aceptar.

Si, de nuevo en el menú principal, presionamos la tecla “I”, entraremos en la pantalla de instrucciones del juego en las cuales podremos avanzar moviendo la mano de derecha a izquierda, como si pasásemos de página, o retroceder con el movimiento inverso. También podremos hacerlo con las flechas de “avance” y “retroceso” del teclado.

Finalmente, si presionamos la tecla “Q” en el menú principal, entraremos a la escena que muestra los créditos del juego, mostrando el nombre de la autora y a las instituciones o grupos a los que pertenece y para los que se realizó el videojuego.

En todas las escenas se nos permitirá volver al menú principal con tan solo presionar la tecla de “espacio” en el teclado, o seleccionando con nuestra mano o con el ratón el botón de “OK” o “Aceptar” siempre que aparezca en la pantalla.

4.4. Pasos seguidos y pruebas realizadas

Todo proyecto debe ser probado para saber por qué camino seguir a la hora de diseñarlo, para saber qué aspectos hay que cambiar, introducir o mejorar y para comprobar si produce los efectos deseados. Por lo tanto, nuestro proyecto no fue una excepción.

En primer lugar, después de buscar toda la información necesaria acerca de Zigfu y del desarrollo para Kinect, hubo que plantearse la estructura que iba a tener el juego a partir de lo acordado con la fundación ASPAYM y siguiendo la forma que ha de tener un videojuego.

Lo primero que se planteó fueron las escenas de las que iba a constar el juego, haciendo un imprescindible menú de inicio, y las principales escenas para realizar el test de movilidad y para jugar a la partida con el perfil de usuario correspondiente.

La escena del test era imprescindible debido a que cada persona tiene un tipo de problema en la movilidad y unas capacidades diferentes, así pues se decidió tomar muestras de los ángulos de torsión, elevación de brazos, piernas, etc., que puede adquirir el jugador y guardarlos en un registro con el nombre del usuario y la fecha de realización del test.

Para la escena partida se tenía en mente que fuesen una especie de muros con unas siluetas de persona recortadas las que fueran hacia el jugador y que éste tuviera que encajar el cuerpo en ella, pero debido a que no nos centramos en la parte de diseñar y modelar objetos, finalmente se decidió utilizar a un personaje que hiciera de la silueta que el paciente tendría que imitar. Aquí vino una de las dificultades, al buscar cuál sería la forma más adecuada de dar las posiciones deseadas a las siluetas. Finalmente se optó por “grabar” las rotaciones de las articulaciones en ciertas posiciones fijadas a propósito y a continuación asignar éstas de forma semi-aleatoria a la silueta, filtrando por rangos de movimiento las posiciones que no serían posibles para cada usuario según su registro (a una persona con el brazo derecho paralizado no se le pueden pedir posturas

levantando dicho brazo) y las que sean imposibles para todo el mundo (como levantar ambas piernas a la vez).

Por otro lado, se pensó en cómo se podría comprobar si el usuario realmente imitaba la postura correctamente, y se llegó a la conclusión de usar *collider* o zonas de colisión en cada articulación de la silueta y cada articulación del personaje del usuario, por lo tanto si un *collider* específico de la silueta (por ejemplo, el situado en la rodilla derecha) entra en contacto con el *collider* correspondiente del usuario (el de la rodilla derecha del paciente), esto será un acierto. Si se falla por lo tanto al menos uno de los *collider*, sería un fallo en la postura.

La forma de puntuación inicial era que se restara una vida por cada fallo en la posición (al menos un *collider* fallido) empezando con 5 vidas y que en cada acierto se fuera sumando un punto.

Aparte de estas dos escenas principales, se sugirió por parte de ASPAYM el añadir el aspecto competitivo al juego, para que generase más adicción, por lo tanto se optó por hace un ranking de puntuaciones. También se pensó en que el decorado tuviera público y que hubiera música y efectos sonoros de fondo, como aplausos por cada postura acertada, para añadir un aspecto motivador, y que se pudiera elegir entre un personaje masculino o femenino para empatizar con el paciente.

Hasta aquí era la idea inicial, la cual se fue ampliando y transformando según se hacían pruebas tanto con personas sin ningún trastorno en la movilidad como con pacientes en rehabilitación. Uno de los aspectos que se ampliaron fue el hecho de diferencias también entre personas que están en silla de ruedas y personas que no, por lo tanto añadimos la detección de ésta al principio del test para así evitar las siluetas con distintas posiciones de piernas.

Además, pudiera ser que el paciente quisiera hacer un nuevo test a medida que va mejorando sus rangos de movimiento para que las posiciones a imitar le sigan suponiendo un esfuerzo proporcional a sus capacidades, por lo tanto se añadió la opción de poder sobrescribir el test de un usuario ya existente si éste así lo deseaba.

También se necesitaba que de alguna forma el nivel a lo largo del juego fuera aumentando, por lo que se dividió la escena principal en los 6 diferentes niveles explicados en la descripción de la escena pasarela.

La estructura del juego ya estaba completamente definida, pero se quería que el fisioterapeuta tuviera un mayor grado de implicación pudiendo variar ciertas constantes según el paciente que procedería a la rehabilitación, por lo que se optó por añadir una escena de configuración. De esta manera también se añadiría jugabilidad al juego al poder permitir que ciertos pacientes quizás más “patosos” tuvieran un rango mayor de error, o que por el contrario a los más aplicados se les pudiese añadir ciertos puntos de dificultad.

Así, se hizo que la configuración contuviera los parámetros de velocidad de la silueta, tamaño de los *collider*, articulaciones falladas permitidas y tiempo entre las posturas del test inicial, y con todo ello se estableció el sistema de vidas y puntos del que actualmente dispone el videojuego.

Capítulo 5. Conclusiones y líneas futuras

En este apartado final concluiremos con los objetivos que se han llegado a alcanzar con este proyecto y si se ha conseguido la finalidad deseada desde el inicio, y la proyección de futuro que tiene, ya que no es algo que haya tocado techo, sino que es el comienzo de un proyecto mayor para la rehabilitación y del cual pretendemos que llegue a ser usado en diferentes centros de fisioterapia o en casa de los pacientes.

5.1. Conclusiones

Todo este trabajo desarrollado hasta la fecha ha tenido como principal objetivo ser un complemento sólido en la rehabilitación de personas con problemas de movilidad, centrándose especialmente en las que han sufrido ictus. Y recalcamos el término de complemento, ya que pretende ser una herramienta más de la que se sirva el fisioterapeuta para realizar su trabajo con los pacientes y que además motive a estos al ser un instrumento portable, entretenido y fácil de utilizar por un solo usuario en entornos más familiares, como puede ser su propia casa.

Por lo tanto, esto es un claro ejemplo de que las técnicas de rehabilitación están en continuo desarrollo y cada vez se pueden usar elementos más comunes para dicho fin, provocando esto el abaratamiento de los costes y por lo tanto favoreciendo la expansión de estas técnicas pioneras.

Es muy probable que el éxito de estas tecnologías comunes usadas para nuevos fines venga, aparte de por la reducción de costes comentada, de que constituyen un elemento que genera diversión y adicción. Estos conceptos favorecen la concentración en cada ejercicio y por lo tanto los resultados que se obtendrán.

Es por ellos por lo que la realidad virtual aplicada a la rehabilitación está en auge, creando un nuevo concepto conocido como “Rehabilitación Virtual” (LabHuman, 2014). Al no ser real, podremos generar entornos atractivos e inspiradores, adaptarlos a cada paciente y hacer que cada tarea cobre sentido, al contrario que los tradicionales ejercicios llevados a cabo, los cuales no poseen estas ventajas a pesar de resultar efectivos.

Por otro lado, al llevar la rehabilitación por medio de un programa, se puede monitorizar la actividad, llevando registros de la evolución de cada paciente y de

los programas realizados y/o que debería realizar, incluso pudiendo hacer todo esto desde casa y de una forma totalmente objetiva.

Hoy en día existe un enorme abanico de posibilidades en cuanto a la realidad virtual se refiere, pero hay que ser cuidadoso con el sistema elegido, ya que tendremos que tener en cuenta que las personas mayores están menos familiarizadas con las nuevas tecnologías, por lo que evitaremos sistemas muy complejos, y que todos tienen problemas al moverse, así que tampoco sería buena idea usar tecnologías que haya que llevar sobre el cuerpo y dificulten los movimientos.

Así se llegó a la conclusión de que el mejor sistema emplearía la Kinect de Microsoft®, con la cual ya se han realizado numerosas pruebas en pacientes con este tipo de problemas con resultados beneficiosos como se ha demostrado en diferentes estudios que se pueden encontrar fácilmente buscando en internet.

A medida que evolucione este proyecto, se realizarán más pruebas con pacientes en rehabilitación, por lo que podrá hacerse un estudio más serio. De momento el videojuego se ha probado con ciertos pacientes que han respondido muy favorablemente.

5.2. Líneas futuras

Hecho todo esto, hay que explicar que este trabajo pretende ser la base de un gran proyecto que se continúa desarrollando. Esto es debido a que si se pretende que el videojuego sea una gran aplicación capaz de mejorar muchos aspectos relativos a la movilidad de las personas con problemas, requiere muchas más características que por la limitación de tiempo de un Trabajo Fin de Grado no se han llegado a alcanzar.

En primer lugar, sería ideal aumentar la cantidad de aspectos configurables para que el fisioterapeuta pudiera modificar o adaptar el juego a cada paciente de una manera más personal. Así, por ejemplo, en nuestro proyecto podemos cambiar el tamaño de los *colliders* en la configuración, es decir, la precisión. Pero, ¿y si un paciente apenas posee movilidad en una extremidad y queremos que desarrolle las capacidades de la opuesta? Igual sería conveniente asignar una precisión baja a su extremidad casi inmóvil y una precisión elevada a la opuesta para que la entrene mucho más. Lo que quiere decir, poder cambiar el tamaño a cada *collider* de manera independiente.

Además, bajo este supuesto también sería útil el poder elegir entre diferentes programas de entrenamiento para ejercitar, por ejemplo, una extremidad específica, o incluso poder elegir el nivel al que queremos acceder directamente, ya que en el caso de una persona que ha evolucionado notablemente en su rehabilitación sería una pérdida de tiempo que tuviera que empezar por el primer nivel en cada partida.

Por otro lado, además de la dificultad y los movimientos establecidos en cada nivel, sería muy beneficioso para el paciente que se pudiera elegir el porcentaje de

movilidad que se pretende conseguir con respecto a la lograda en el test inicial. Así se puede pretender que el paciente complete un nivel con un porcentaje de un 80%, por lo que no tendría que llegar a forzar el movimiento de sus partes del cuerpo hasta la totalidad de lo logrado en el test, o se podría exigir un porcentaje del 110%, teniendo que mover algunas partes más allá del máximo logrado.

Actualmente, el videojuego se centra en rehabilitar aspectos como la percepción del cuerpo, movilidad y equilibrio, pero en un proyecto mayor se podrían llegar a valorar otros aspectos a mayores como puede ser el habla que, como comentamos en el apartado de trastornos o enfermedades que causan problemas de movimiento, también es una capacidad que se puede deteriorar. Sin embargo, aunque por la SDK de Microsoft® se puede capturar la voz, no es algo que esté optimizado para el español, por lo que se decidió no implementarlo por el momento.

También comentamos a lo largo de este informe que la gran ventaja del videojuego frente a otros métodos de rehabilitación es la diversión y motivación que supone para el paciente, por lo tanto, añadir tantos elementos como podamos para aumentar estos valores será beneficioso. Así por ejemplo, se podría hacer que ciertas posiciones fueran para alcanzar algún objeto, como por ejemplo un sombrero, y que al conseguirlo se le añadiera al personaje como parte de su atuendo.

Junto con todo esto, podemos aprovecharnos también de más elementos ya implementados para añadirles más funciones, como por ejemplo, tener un mayor seguimiento de la evolución del paciente mediante el test no sólo al principio, sino también al final de la partida y comparando los ángulos de movimiento logrados en ambos. Quizás como ahora haya que guardar más datos que los que ya guardábamos, sería entonces útil usar la base de datos descartada inicialmente, y así hacer un historial de seguimiento al guardar los test de diferentes partidas del inicio y del final.

Aparte de lo meramente interno referente al videojuego que hemos comentado hasta ahora, también cabe plantearse el uso del videojuego junto con otros elementos externos útiles para la rehabilitación, como pueden ser resistencias en muñecas y tobillos que añadan peso a estas extremidades y se aumente levemente el nivel de dificultad, o que el fisioterapeuta pudiera estar cerca o junto al usuario sin que interfiera en los movimientos de éste frente a la Kinect, ya que ciertos pacientes se sienten más seguiros con alguien de apoyo cerca de ellos por miedo a perder el equilibrio. Esto se podría lograr mediantes elementos como máscaras que hacen que no se detecten a las personas a partir de cierta distancia de la Kinect.

Finalmente, cabe destacar de nuevo que actualmente ya se encuentra disponible la Kinect v2 la cual posee la última tecnología en computación humana permitiendo la interacción entre la máquina y la persona de una forma más natural a la permitida hasta el momento (Microsoft®, 2014e). Por lo tanto, gracias a estas mejoras que la versión 2 incorpora, que encontramos en el reconocimiento de voz, la captura del cuerpo humano de manera más precisa y eficiente y los sensores,

podremos ser más precisos en las tareas que nuestro videojuego ya realiza e incorporar otras nuevas.

Por todo ello, este proyecto y su futuro desarrollo va a permitir seguir con los estudios acerca de la efectividad de este tipo de terapias para la rehabilitación, los cuales ya se estaban realizando con resultados favorables para el paciente.

De momento se ha comprobado la buena acogida que tiene nuestro proyecto en el Museo de la Ciencia de Valladolid celebrado en noviembre de 2014, en el cual se presentaban todo tipo de soluciones innovadoras para personas dependientes por medio de productos de elevado componente tecnológico (DICYT, 2014). Tanto ancianos como niños se acercaban a jugar con nuestro videojuego e incluso algunos venían a probarlo porque les había llegado el rumor a través del boca a boca.

En resumen, lo que se pretende con nuestro trabajo es constituir una base sólida a partir de la cual se cree un gran proyecto que ayude a personas que realmente lo necesitan, tratando de que puedan llevar una vida lo más común e independiente posible, y que se haga tan grande gracias que se invierta en él para que se utilicen tecnologías más potentes que ofrezcan aún mejores resultados.

Bibliografía

Agencia Francesa por los Videojuegos. (Octubre de 2014). *ShiVa 3D Becomes First 3D Dev Kit for Android*. Recuperado de http://www.afjv.com/press1003/100319_shiva_stonetrip_android.php

Agencia Valenciana de Salud (2007). *Guía de información al paciente de ictus*. Valencia, España: Conselleria de Sanitat de la Generalitat Valenciana.

Alvarado, D., Barría, S., Cordella, R. y Lechuga, A. (2007). *Métodos de optimización para proyectos 3D*. Recuperado de http://www.optimizacion3d.info/descargas/metodos_de_optimizacion_para_proyectos_3d_version_web.pdf

ASUS. (Octubre de 2014). *Xtion PRO LIVE*. Recuperado de http://www.asus.com/es/Multimedia/Xtion_PRO_LIVE/

Blanco, Y., Saiz, A., Graus, F. (2005). Evolución clínica de la resonancia magnética y de las bandas oligoclonales en los pacientes con esclerosis múltiple receptores de un trasplante autólogo de progenitores hematopoyéticos. *Revista Española de Esclerosis Múltiple, 1,* 6-15.

Blender.org. (Octubre de 2014). *Blender*. Recuperado de http://www.blender.org

Catuhe, D. (2012). *Programming with the Kinect for Windows Software Developmente Kit*. Redmond, WA: Microsoft® Press.

Conneally, T. (Octubre de 2014). *Lenovo: Next-generation videogame console competitor?*. Recuperado de http://betanews.com/2011/05/09/lenovo-next-generation-video-game-console-competitor/

Corona Labs. (Octubre de 2014). *Corona SDK*. Recuperado de http://coronalabs.com

DICYT: Agencia Iberoamericana para la Difusión de la Ciencia y la Tecnología. (Noviembre de 2014). *Una muestra con soluciones innovadoras para personas dependientes*. Recuperado de http://www.dicyt.com/noticias/una-muestra-con-soluciones-innovadoras-para-personas-dependientes

Dirección General de Tráfico (Octubre de 2014). *El número de muertos por accidente de tráfico en 2013 registra un mínimo histórico*. Recuperado de http://www.dgt.es/es/prensa/notas-de-prensa/2014/20140103-balance-2013-seguridad-vial-2013.shtml

Dirección General de Tráfico (Octubre de 2014). *Tema 33: Tratamiento post-accidente. Protocolos de actuación de los servicios de emergencias en caso de accidentes. Descripción del problema. Índices de supervivencia después de un accidente. Secuelas y costes. Proyecto e-call. Tratamiento pre-hospitalario, hospitalario y post-hospitalario. Resumen de medidas y efectos*. Recuperado de

http://www-cdn-org.dgt.es/es/la-dgt/empleo-publico/oposiciones/2013/20131126-temario-grupo-de-materias-comunes-de-moviliad-segura-convocatoria-promocion-interna-2013.shtml

Espinosa, J., Arroyo, O., Martín, P., Ruiz, D. y Moreno, J. A. (2010). *Guía esencial de rehabilitación infantil*. Madrid, España: Editorial médica Panamericana.

Etxeberria, F. (2009). Videojuegos y educación. *Teoría de la Educación: Educación y Cultura en la Sociedad de la Información, 10.*

Fundación ASPAYM Castilla y León. (Septiembre de 2014). *ASPAYM, ¿quiénes somos?*. Recuperado de http://www.aspaymcyl.org/index.php/quienes-somos/presentacion

Fundación de Ayuda contra la Drogadicción. (2002). *Jóvenes y videojuegos: Espacio, significación y conflictos*. Madrid, España: FAD, INJUVE.

Gil, A., Vida, T. (2007). *Los videojuegos*. Barcelona, España: Editorial UOC.

Gros, B. (2008). *Videojuegos y aprendizaje*. Barcelona, España: Editorial Graó.

Grupo de Estudio de las Enfermedades Cerebrovasculares (2003). *Después del ictus: Guía práctica para el paciente y sus cuidadores*. Barcelona, España: EdiDe.

Harvey, L. (2010). *Tratamiento de la lesión medular : guía para fisioterapeutas*. Barcelona, España: Elsevier.

Ictussen. (Septiembre de 2014). *Grupo de estudio de enfermedades Cerebrovasculares de la SEN*. Recuperado de http://www.ictussen.org/?q=node

Instituto de la mujer/CIDE. (2004). *La diferencia sexual en el análisis de los videojuegos*. Madrid, España: CIDE, Instituto de la mujer.

Jiménez, A. (2003). *Manual de neurogenética*. Madrid, España: Ediciones Díaz de Santos.

Kelley, W. N. (1992). *Medicina interna, Volumen 1*. Philadelphia, Pennsylvania: Editorial Médica Panamericana.

LabHuman. (Diciembre de 2014). *Realidad virtual en neurorehabilitación: 5 años de experiencias clínicas*. Recuperado de http://www.imserso.es/InterPresent1/groups/imserso/documents/binario/4_holrlvirtual.pdf

Linares, J., Martínez, J. V., Candela, A. (2012). Gráficos a la máxima potencia: Una comparativa entre motores de juegos. *3 Ciencias, 2*, 52-65.

MakeHuman Team. (Octubre de 2014). *MakeHuman*. Recuperado de http://www.makehuman.org

Marcano, B. (2008). Juegos serios y entrenamiento en la sociedad digital. *Teoría de la Educación: Educación y Cultura en la Sociedad de la Información, 9,* 97-101.

Micheli, F. (2003). *Tratado de neurología clínica*. Buenos Aires, Argentina: Editorial médica Panamericana.

Microsoft®. (Septiembre de 2014a). *Kinect for Windows*. Recuperado de http://www.microsoft.com/en-us/kinectforwindows/default.aspx

Microsoft®. (Octubre de 2014b). *Microsoft® Studios acquires rights to Gears of War franchise*. Recuperado de http://news.xbox.com/2014/01/games-microsoft-studios-gears-of-war

Microsoft®. (Noviembre de 2014c). *Kinect for Windows SDK*. Recuperado de http://msdn.microsoft.com/en-us/library/hh855347.aspx

Microsoft®. (Noviembre de 2014d). *Microsoft® API and reference catalog*. Recuperado de http://msdn.microsoft.com/es-es/library/ms233843.aspx

Microsoft®. (Noviembre de 2014e). *Kinect for Windows features*. Recuperado de http://www.microsoft.com/en-us/kinectforwindows/meetkinect/features.aspx

Minoru3D. (Octubre de 2014). *Say Hi to the Minoru 3D Webcam*. Recuperado de http://www.minoru3d.com

Montaner, J. (2009). *Tratamiento del ictus isquémico*. Barcelona, España: IGC Marge, SL.

Motion Arcde Inc. (Noviembre de 2014). *Zigfu*. Recuperado de http://zigfu.com

Palazón, I. (1998). Salud mental infantil: Patologías más frecuentes en la Atención Primaria. *Revista Asociación Española de Neuropsiquiatría, 18 (67)*, 545-553.

Rincón, P., Zambrano, E. (2012). *Motion Capture*. Recuperado de http://123seminarsonly.com/Seminar-Reports/2013-02/87126810-Motion-Capture.pdf

RobotShop Distribution. (Octubre de 2014). *Minoru 3D Webcam*. Recuperado de http://www.robotshop.com/en/minoru-3d-webcam.html

Rowland, A. (Octubre de 2014). *Unity Technologies Lands $12 Million in Series B Funding Led by WestSummit Capital and iGlobe Partners*. Recuperado de http://www.marketwired.com/press-release/unity-technologies-lands-12-million-series-b-funding-led-westsummit-capital-iglobe-partners-1540593.htm

Sahrmann, S. A. (2005). *Diagnóstico y tratamiento de las alteraciones de movimiento*. Barcelona, España: Paidotribo.

ShiVa Technologies. (Octubre de 2014). *ShiVa3D*. Recuperado de http://www.stonetrip.com

SoftKinetic. (2014). *SoftKinetic*. Recuperado de http://www.softkinetic.com/en-us/softkinetic.aspx

Suau, P. (2011). *Manual de modelado y animación con Blender*. Alicante, España: Textos Docentes.

Terré-Boliart, R., Orient-López, F. (2007). Tratamiento rehabilitador en la esclerosis múltiple. *Revista NEUROL, 44*, 426-431.

Unity Technologies. (Septiembre de 2014a). *Unity3D*. Recuperado de http://unity3d.com

Unity Technologies. (Octubre de 2014b). *Mecanim - Tecnología de animación simple y poderosa*. Recuperado de https://unity3d.com/es/unity/animation

Unity Technologies. (Noviembre de 2014c). *Scripting API*. Recuperado de http://docs.unity3d.com/ScriptReference/

Unreal Engine. (Octubre de 2014). *Unreal Engine*. Recuperado de https://www.unrealengine.com

Vallejo, D., Martín, C. (2012). *Desarrollo de Videojuegos: Arquitectura del Motor*. Ciudad Real, España: Bubok.

Webb, J, Ashley, J. (2012). *Beginning Kinect programming with the Microsoft® Kinect SDK*. New York, NY: Apress.

Williams, D. (2013). *Corona SDK Application Design*. Birmingham, Reino Unido: Packt Publishing.

Zammetti, F. (2013). *Learn Corona SDK Game Development*. New York, NY: Apress.

Zone, R. (2013). *3DIY: Stereoscopic Moviemaking on an Indy Budget*. Burlington, MA: Focal Press.