Curso 2019/2020
Trabajo de Fin de Grado en
Ingeniería Informática e Ingeniería de Software

Desarrollo de un videojuego de acción para personas con discapacidad visual

Realizado por:

Enrique Maestro Mañanes
Adrián Menéndez López
Jose Mª Gómez-Trabadela García
Jorge Gómez Baraibar

Dirigido por:

María Guijarro Mata-García
Borja Manero Iglesias

Palabras clave

- Videojuego
- Unity
- Acción
- Sonido
- Accesible
- Shooter
- Laberinto
- Discapacidad Visual
- Enemigo
- Audiojuego

Keywords

- Videogames
- Unity
- Action
- Sound
- Accesible
- Shooter
- Maze
- Visual Impairment
- Enemy
- Audiogame

Índice

1. Introducción

1.1. Motivación

Según las estimaciones realizadas por la Organización Mundial de la Salud (OMS) aproximadamente cerca de 1.300 millones de personas padecían algún tipo de discapacidad visual. De todas estas personas, cabe destacar que cerca de 188 millones de personas tienen una deficiencia visual moderada, 217 millones tienen una deficiencia visual moderada-grave y cerca de 36 millones estaban ciegas totalmente, lo que supone un 0,7% de la población total (OMS, 2018).

Desde hace años la tecnología ha dejado de ser sólo un medio de comunicación y una herramienta de trabajo. Cada vez compramos más por internet, consumimos más entretenimiento online y jugamos cada vez más a videojuegos complejos que requieren de agudeza visual o basan sus pruebas a los reflejos en percibir sonidos. Las grandes empresas de videojuegos no invierten recursos en hacer juegos accesibles y acercar así este mundo a las personas con discapacidad, por lo que este colectivo no es capaz de disfrutar al máximo de este pasatiempo, en el que hoy en día es considerado un arte como la literatura o el cine. (Javi Andrés, 2020).

El perfil de jugador con algún tipo de discapacidad visual no es tenido en cuenta por los creadores de videojuegos, aunque muchos de estos no son tan difíciles de adaptar. Los pocos videojuegos diseñados específicamente para personas con ceguera casi siempre son demasiado genéricos, infantiles. Algunos juegos presentan una dificultad añadida para que lo puedan disfrutar jugadores que no pueden recibir estímulos visuales a la hora de jugar.

Esto lo aprendimos gracias a una visita al Centro de Tiflotecnología e Innovación (CTI) de la Organización Nacional de Ciegos Españoles (ONCE), donde tuvimos la oportunidad de escuchar testimonios de personas con ceguera con relación a los videojuegos: cómo juegan, qué opinión tienen de los juegos adaptados actuales, a qué les gustaría jugar, etc.

Nos pareció impresionante la habilidad que realmente demuestran estas personas a la hora de manejarse con la tecnología y la accesibilidad que ofrecen hoy en día sistemas como Android o iOS para móviles.

Con esta información decidimos hacer un videojuego que estuviera adaptado desde el principio del desarrollo y que resultara entretenido. Queríamos salir del prototipo de los videojuegos simplones que habíamos visto en el transcurso de esa reunión. Como, por ejemplo:

- Feer, para dispositivos móviles, el cual consiste ir esquivando enemigos que aparecen enfrente tuyo moviéndote a la izquierda o a la derecha, mediante sonidos detectas por qué lado se acercan hacia a ti.

Ilustración 1. Feer.

- Diversión accesible, que se compone de cuatro minijuegos que hacen uso de la vibración del móvil.

Ilustración 2. La imagen de la izquierda es el menú, y las otras dos son dos de los minijuegos. (Arias, n.d.)

- A Blind Legend, que es un audio juego que cuenta una historia y mediante unos controles simples en la pantalla puedes ir avanzando y tomando decisiones.

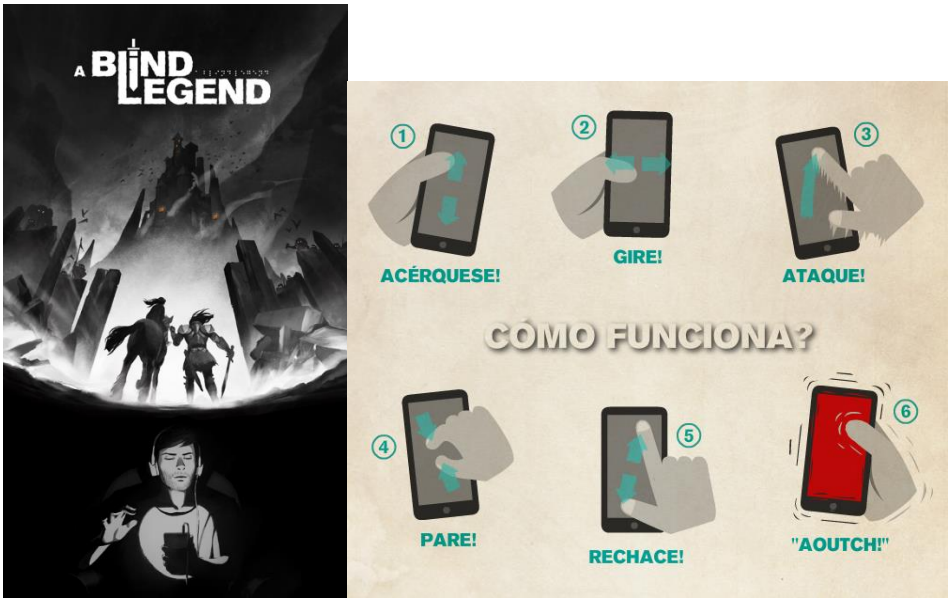

Ilustración 3. Blind Legend. A la izquierda vemos una imagen del juego, y a la derecha los controles que se usan. (DOWiNO, n.d.)

Nos gustaba la idea de hacer un juego en primera persona, en realidad virtual, donde los jugadores pudieran disfrutar de una experiencia inmersiva en un entorno virtual de la forma más parecida posible a la de un videojuego actual.

Con la llegada de la pandemia mundial del COVID-19, que llegó a mitad del curso, la opción de hacer el proyecto en realidad virtual se vio truncada, ya que ninguno de los integrantes del equipo disponíamos personalmente del equipamiento necesario para probar el proyecto en un entorno de realidad virtual. No obstante, continuamos el proyecto, modificándolo para que fuese jugable con un mando de consola conectado al ordenador, con el que podíamos utilizar la vibración como estímulo para comunicar algunas situaciones dentro del juego, como se verá más adelante en el documento.

En las últimas fases del desarrollo, contactamos con el antiguo director del CTI de la once, de 2001 a 2005, que se ofreció a probar el juego y ayudarnos. Gracias a su ayuda tenemos la perspectiva de una persona ciega y con experiencia en videojuegos para personas invidentes, por lo que tenía una idea bastante formada sobre el uso de los controles desde un inicio, y se presentó voluntario para probar el juego, darnos *feedback* de qué mecánicas le parecían positivas para la adaptabilidad y cuáles necesitaban revisarse para hacer el juego más accesible. Gracias a esta persona, pudimos hacer al menos una iteración de pruebas sobre el desarrollo del videojuego, consiguiendo mejorar la accesibilidad.

1.2. Objetivos

El principal objetivo de este proyecto es desarrollar un videojuego que sea entretenido, tanto para personas videntes como para personas con algún tipo de problema visual; que siga los estándares y mecánicas de los juegos más modernos, y que sirva de precedente a otros desarrolladores para mejorar la accesibilidad en este entorno.

A nivel práctico, para conseguir este objetivo deberemos plantearnos otros objetivos más específicos que nos ayuden a guiarnos durante el desarrollo. Estos objetivos son:

- Análisis en profundidad del estado del arte, tanto en los videojuegos en general como en el desarrollo de videojuegos adaptados para personas invidentes y tipos de discapacidades visuales.
- Desarrollar una idea de videojuego que, siendo simple en sus mecánicas, sea lo suficientemente atractivo y desafiante para demostrar que una persona con ceguera es completamente capaz de jugar a un videojuego como los que se desarrollan hoy en día.
- Verificar la accesibilidad. Mediante experimentos con personas con discapacidad visual podremos verificar que el juego puede jugarse por personas invidentes y que cumple con los estándares para ser accesible.

Dichos objetivos son los esenciales, independientemente de los cambios que ha sufrido la idea inicial debido a la pandemia del COVID-19. En ambos casos el objetivo principal es el desarrollo de un videojuego accesible, que sea atractivo tanto para videntes como para personas que tengan algún problema de visión. Los cambios que hemos realizado son más visibles en las mecánicas que hemos introducido, ya que es la parte más afectada por la situación actual.

Se podría considerar que el objetivo que más cambios ha sufrido ha sido la accesibilidad, ya que al no disponer de los controles que habíamos pensado inicialmente, hemos tenido que cambiar algunas mecánicas para adaptarlas a los recursos que teníamos disponibles. En un principio íbamos a usar los mandos de realidad virtual como bastón blanco, en su lugar hemos tenido que hacer uso del motor de vibración del mando de una consola.

1.3. Estructura del documento

Este documento seguirá una estructura que va desde lo más genérico y externo a lo más preciso.

En el capítulo 3, “Estado del Arte”, vemos el desarrollo y los tipos de videojuegos que hay actualmente, con algunos ejemplos. Explicamos el tipo de juego que hemos escogido, sus mecánicas y jugabilidad.

Por último, vemos ejemplos de juegos adaptados para personas con algún tipo de discapacidad visual.

El capítulo 4, “Plan de Desarrollo”, es el desarrollo e implementación del videojuego, en el cual detallamos los puntos importantes del diseño e implementación del producto final.

El capítulo 5, “Fases de experimentación”, explica cómo al comienzo del curso teníamos pensado probarlo con el mayor número de gente, pero debido a los acontecimientos nos hemos visto obligados a modificar este apartado. Por este motivo, este capítulo detallará la experimentación con un trabajador de la ONCE, que posee la experiencia necesaria como para valorar de manera crítica y objetiva el juego que le presentaremos y las mecánicas implementadas en éste.

En el capítulo 6, “Conclusiones y trabajo a futuro”, explicamos las conclusiones que hemos sacado de todo este proceso, las lecciones aprendidas y las posibles mejoras del proyecto en un futuro.

2. Introduction

2.1. Motivation

According to estimates made by the World Health Organization (WHO) approximately 1.3 billion people had some kind of visual impairment. Of all these people, it is worth noting that nearly 188 million have a moderate visual impairment, 217 million have moderate-severe visual impairment and nearly 36 million were totally blind, representing 0.7% of the total population (OMS, 2018).

From years, technology has ceased to be just a communication media and a work tool. We buy more and more on the internet; we consume more online entertainment and we play more and more complex video games that require visual acuity or base their reflex tests on perceiving sounds. The large video game companies do not invest resources in making games accessible and thus bringing this world closer to people with disabilities, so this group is not able to enjoy this pastime to the fullest, in which it is now considered an art like literature or film. (Javi Andrés, 2020)

The profile of a player with some kind of visual impairment is not taken into account by video game creators, although many of them are not so difficult to adapt. The few video games designed specifically for people with blindness are almost always too generic and childish. Some games present an added difficulty for players who cannot receive visual stimuli when playing.

We learned this thanks to a visit to the Center for Tiflotechnology and Innovation (CTI) of the Spanish National Organization for the Blind (ONCE), where we had the opportunity to heard testimonies from people with blindness in relation to video games: how they, what opinion have about the actual adapted games, what they would like to play, etc.

We were impressed by the ability that these people actually demonstrate when it comes to dealing with the technology and accessibility that systems like Android or iOS for mobiles offer today.

With this information we decided to make a video game that was adapted from the beginning of the development and that would be entertaining. We wanted to get out of the prototype of simple video games we had seen during that meeting. Like, for example:

- Feer, for mobile devices, which consists of dodging enemies that appear in front of you by moving left or right, using sounds to detect which side they are approaching you from.

Ilustración 4, Feer.

- *Diversión accesible*, consisting of four mini-games that make use of the vibration of the mobile phone.

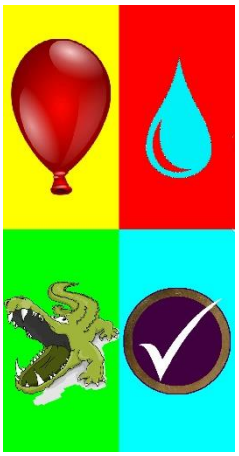

Ilustración 5, The image on the left is the menu, and the other two are two of the mini-games.(Arias, n.d.)

- A Blind Legend, which is an audio game that tells a story and through simple on-screen controls you can move forward and make decisions.

Ilustración 6, Blind Legend. On the left we see an image of the game, and on the right the controls that are used.(DOWiNO, n.d.)

We liked the idea of making a game in first person, in virtual reality, where players could enjoy an immersive experience in a virtual environment as close as possible to that of a current video game.

With the arrival of the COVID-19 global pandemic, which came mid-year, the option of doing the project in virtual reality was cut short, as none of us on the team personally had the necessary equipment to test the project in a virtual reality environment. However, we continued the project, modifying it to make it playable with a console controller connected to the computer, with which we could use the vibration as a stimulus to communicate some situations within the game, as we will see later in the document.

In the last phases of development, we contacted the former director of the CTI of the eleven, from 2001 to 2005, who offered to test the game and help us. Thanks to his help we have the perspective of a blind person with experience in video games for blind people, so he had a pretty good idea about using the controls from the beginning, and volunteered to test the game, give us feedback on what mechanics he found positive for adaptability and what needed to be revised to make the game more accessible. Thanks to this person, we were able to do at least one iteration of testing on the development of the game, and we were able to improve accessibility.

2.2. Objectives

The main objective of this project is to develop a video game that is entertaining, both for sighted people and for people with some kind of visual problem; that follows the standards and mechanics of the most modern games, and that serves as a precedent for other developers to improve accessibility in this environment.

On a practical level, to achieve this goal we will have to set other more specific objectives to help guide us during development. These objectives are:

- In-depth analysis of the state of the art, both in video games in general and in the development of adapted video games for blind people and types of visual disabilities.
- To develop an idea of a video game that, being simple in its mechanics, is attractive and challenging enough to demonstrate that a person with blindness is fully capable of playing a video game like those developed today.
- Check accessibility. Through experiments with visually impaired people we will be able to verify that the game can be played by blind people and that it complies with the standards to be accessible.

These are the essential objectives, regardless of the changes that the initial idea has undergone due to the COVID-19 pandemic. In both cases, the main objective is the development of an accessible video game that is attractive to both sighted and visually impaired people. The changes we have made are more visible in the mechanics we have introduced, as this is the part most affected by the current situation.

It could be considered that the objective that has suffered more changes has been the accessibility, since not having the controls that we had thought initially, we have had to change some mechanics to adapt them to the resources that we had available. At the beginning we were going to use the virtual reality controls as a white cane, instead we have had to use the vibration engine of a console control.

2.3. Document structure

This document will follow a structure ranging from the most generic and extern of the project to the most precise.

In chapter 3, "State of the Art", we see the development and types of video games that are currently available, with some examples. We explain the type of game we have chosen, its mechanics and playability.

Finally, we see examples of games adapted for people with some kind of visual disability.

Chapter 4, "Development Plan", is the development and implementation of the video game, in which we detail the important points of the design and implementation of the final product.

Chapter 5, "Experimental Phases", explains how at the beginning of the course we had planned to test it with the largest number of people, but due to events we have been forced to modify this section. For this reason, this chapter will detail the experimentation with an ONCE worker, who has the necessary experience to critically and objectively evaluate the game that we will present, and the mechanics implemented in it.

In chapter 6, "Conclusions and future work", we explain the conclusions we have drawn from this whole process, the lessons learned and possible improvements to the project in the future.

3. Estado del arte

3.1. Definición de Videojuego

Según la R.A.E. Un videojuego puede definirse de la siguiente forma (Real Academia Española, n.d.):

1. m. Juego electrónico que se visualiza en una pantalla.

2. m. Dispositivo electrónico que permite, mediante mandos apropiados, simular juegos en las pantallas de un televisor, una computadora u otro dispositivo electrónico.

Añadiremos a esta definición que en la parte de "mandos apropiados" se refiere a un dispositivo de entrada para manipular lo que sucede en la simulación. Este dispositivo puede ser una palanca, un botón, un dispositivo que mezcla botones y palancas, un teclado o un ratón.

3.1.1. Tipos de videojuego

Los videojuegos se suelen categorizar a través de géneros. Estos géneros catalogan los juegos en función de su jugabilidad y mecánicas internas. Comúnmente se clasifican los videojuegos 4 campos principales, siendo estos:

- Juegos de acción: Aquellos que requiere una participación muy activa y mucha atención a los eventos que suceden por pantalla para poder actuar en consecuencia.
- Juegos de rol: Aquellos en los que se toma el papel de un personaje preconstruido o que el propio jugador construye desde un inicio.
- Juegos de estrategia: Aquellos juegos más reflexivos, que se basan en la toma de decisiones y gestión de recursos para lograr una meta.
- Juegos de simulación: Aquellos que intentan emular alguna situación de la vida real de manera más o menos realista.

Los juegos pueden ser un híbrido de varios de estos géneros y tomar características de varios de ellos en mayor o menor medida.

Como esta categorización por géneros queda demasiado amplia, pudiendo recoger muchos juegos sin llegar a especificar mucho, se pueden observar varios subgéneros con unas características más concretas.

En estos subgéneros se puede diferenciar algunos como (Adams, 2014):

- **Shooters**: Juegos principalmente de acción. Llamados así porque su principal mecánica es la lucha a través de disparos. Por ejemplo: Counter Strike, Call Of Duty, Battlefield. Entraremos en más detalle con este subgénero en su propio apartado dentro de este documento.
- **Plataformas**: Juegos principalmente de acción, basados en el movimiento del personaje por un escenario con obstáculos, alturas y enemigos que hay que sortear con habilidades que se van desbloqueando a lo largo del juego. Por ejemplo: Super Mario, Spyro, Meat Boy.
- **Lucha**: Juegos de acción, basados en el combate de varios personajes con diferentes habilidades especiales. Por ejemplo: Street Fighter, Tekken, Skull Girls.
- **RTS o estrategia a tiempo real**: Basados en la toma de decisiones y control de recursos de manera continua. Por ejemplo: Warcraft, Age of Empires, Imperium.
- **MOBA**: Acrónimo del término inglés Multiplayer Online Battle Arena. Que siendo un derivado de los juegos RTS, se diferencia de ellos en que controlas a un solo personaje y formas equipo con otros jugadores para enfrentarse a otros jugadores. Por ejemplo: League of Legends, DOTA, Smite.
- **TBS o estrategia por turnos**: Basados en los turnos como espacio de tiempo en el que el jugador piensa y toma una cantidad limitada de decisiones en función de los eventos que han sucedido desde su último turno. Por ejemplo: Sid Meier's Civilization, Fire Emblem, X-COM.
- **Deportivos**: Juegos de simulación que intentan emular el funcionamiento de un deporte en concreto. Por ejemplo: FIFA, NBA2K, PES.
- **Carreras**: Juegos que tratan de emular la conducción ya sea de coches, naves o cualquier otro vehículo. El objetivo suele ser completar una serie de carreras. Por ejemplo: DIRT, Mario Kart, Need For Speed.
- **Aventuras**: Juegos con mecánicas varias que pueden contener características de rol, dado que tienes que tomar el control de un personaje y sus decisiones; de estrategia, ya que dependiendo de las elecciones que se tomen, sucederán unos eventos u otros, y de acción, pues muchos de estos juegos tienen escenas de acción en las que hay que pulsar una serie de tecla de manera repetida o realizar acciones con poco tiempo de reacción. Por ejemplo: Heavy Rain, The Last of Us, Uncharted.
- **Construcción de ciudades**: Juegos con mecánicas de estrategia y simulación, basados en la gestión de recursos para construir asentamientos o ciudades. Estas ciudades se componen de módulos (edificios) que tienen ciertas sinergias entre ellos o tasas de generación y consumo de recursos. Por ejemplo: Sim City, Saga ANNO, juegos Tycoon.
- **Puzles**: Juegos puramente de estrategia, basados en la resolución de un escenario a través de las mecánicas que nos ha ido presentando el juego (Adams, 2010; Apperley, 2006). Por ejemplo: Cut the Rope, The Talos Principle, Portal.

3.1.2. Shooters

Los Shooters o Juegos de disparos es un género de videojuego centrado en el combate con armas de fuego. Existen 2 tipos según la perspectiva que se da al jugador a la hora de manejar al personaje:

- **Primera persona (First Person Shooter o FPS):** Es aquella en el que el jugador experimenta la acción a través de los ojos del protagonista. El primer videojuego que podemos ver que usó este tipo de experiencia fue el Maze War (1974) (Colley, n.d.; Olivetti, 2012). En este juego el jugador se desplaza por un laberinto, lleno de enemigos que se mueven hacia atrás y adelante, y que giran a la izquierda y a la derecha. Los jugadores van ganando puntos conforme aciertan con sus disparos a otros jugadores y los pierden cuando son abatidos.

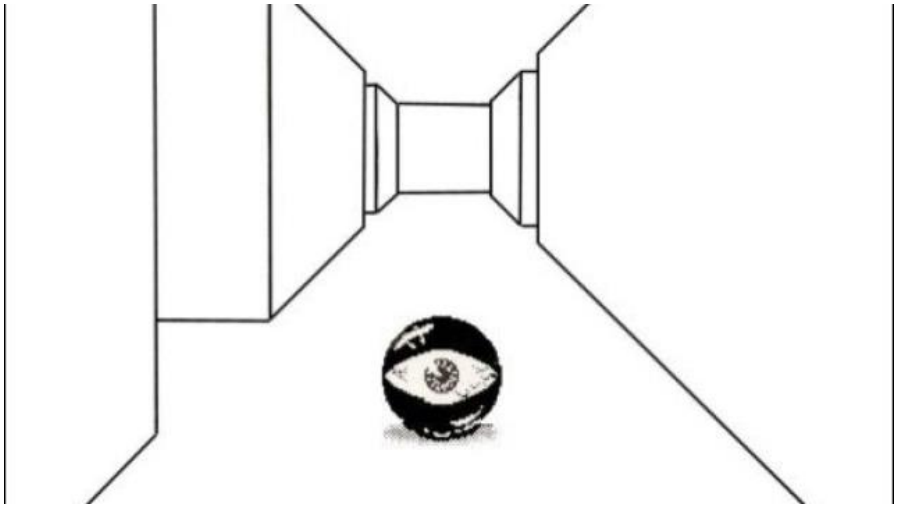

Ilustración 7. Maze War (Olivetti, 2012)

- **Tercera persona**: Es aquella en el que el jugador experimenta la acción desde una perspectiva en tercera persona. El primer videojuego que podemos ver que usó la tercera persona y que además se considere shooter es el "Spacewar!" (1962) (Cabacas, 2012) que fue también uno de los primeros videojuegos que se crearon.

Ilustración 8. Spacewar! ("Spacewar!," n.d.)

Aparte de estos existen también unos subtipos de shooters dependiendo del realismo, el número de personajes, la temática y las limitaciones del personaje (“Shooter game,” n.d.).

- Hero shooters: Se puede considerar como un híbrido entre Primera persona (FPS) y MOBA. Se juegan seleccionando un personaje específico de un plantel que dispone el juego. Cada uno de los personajes dispone de unas habilidades específicas, unas características de daño y vida y un equipo concreto. La finalidad es que el jugador, con la ayuda de su equipo (otros jugadores), cumpla unos objetivos: capturar una zona específica, escoltar una carga, etc. (Alonso, 2016) Ejemplos de estos juegos son Overwatch, Lawbreakers, Valorant.

Ilustración 9. Overwatch (Wawro, 2016)

- Shooters tácticos: Aquellos que mezclan los componentes básicos del shooter con un estilo de juego más pausado, dando más peso a la estrategia y el trabajo en equipo a través de mecánicas, introducidas por equipamiento y roles de personaje específicos, que se usan para desarrollar un plan a tiempo real con la finalidad enfrentarse a la estrategia del equipo contrincante (“Tactical shooter,” n.d.). Ejemplos de esto son juegos como Counter Strike o Rainbow Six.

Ilustración 10. Rainbow Six Siege (Matas, 2019)

- Loot Shooters: Juegos que mezclan las mecánicas de juego de los juegos de disparos y los juegos Hack & Slash, subgénero de los juegos de rol-acción en el que la jugabilidad se centra en el combate. Se basan en que al acabar con los enemigos éstos dejarán caer equipo, para mejorar al personaje (armas, accesorios, armaduras), el cual tendrá unas características aleatorias. Juegos de este tipo son Borderlands o Destiny (Kionay, 2019).

Ilustración 11. Borderlands 3 (Tassi, 2019)

- Shoot'em up o Mata-marcianos: Es un subgénero específico de los shooters donde el jugador tiene un control limitado del movimiento. Su nombre se debe a la influencia que tuvo Space Invaders (Ortega, 2018; Velasco, 2011).

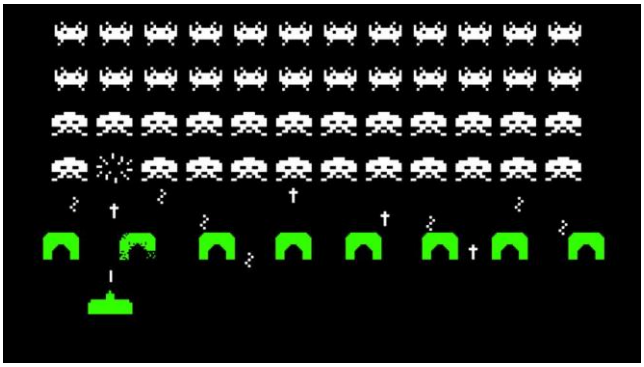

Ilustración 12. Space Invaders (García, 2018)

- Shooter sobre raíles: Se denominan así porque el jugador no tiene control sobre el movimiento, moviéndose el personaje sobre "raíles" o caminos prefijados. El jugador lo único que tiene que hacer es disparar a los enemigos que aparezcan en pantalla mientras es dirigido por el juego a través de pantallas. Juegos de este tipo son Doom Rails, Over Touch ("Definición de Shooter On Rails," 2020).

Ilustración 13. Doom Rails (Bouzo, 2009)

- Arcades con periféricos: Con este género nos referimos a las máquinas recreativas que tenían una pistola de plástico conectada, la cual servía como mando y permitía disparar a través de su gatillo a objetivos que mostraba el juego.

Ilustración 14. Shooters arcade (Sanchez, 2015)

3.2. Presencia y Telepresencia

La Presencia (Gaggioli, Bassi, & Fave, 2003; Witmer & Singer, 1998)(Steuer, 1992) puede ser considerada como la experiencia del entorno físico de uno; no se refiere a los alrededores tal como existen en el mundo físico, sino a la percepción de estos por parte de nuestras capacidades físicas. (Ilustración 9)

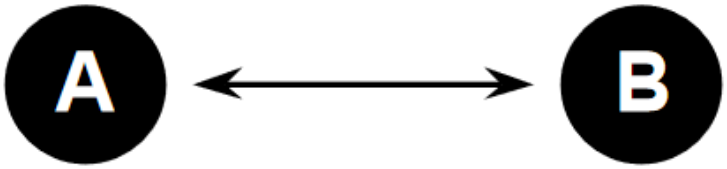

Ilustración 15. La vista tradicional. A es el observador y B es el entorno (Steuer, 1992)

Muchos factores contribuyen a crear o generar esta experiencia. Los sentidos como el oído nos permiten generar esa presencia mediante estímulos sensoriales, pero hay otros procesos, tanto perceptivos como mentales que asimilan estos estímulos y los mezclan con las experiencias presentes y pasadas generando una presencia distinta en cada persona. La presencia está relacionada con el fenómeno de la atribución distal o externalización, que se refiere a la percepción del entorno desde un punto de referencia distinto al que nuestro cuerpo físico está ocupando.

Sin embargo, cuando la percepción está mediada por la tecnología uno se ve obligado a percibir dos entornos separados a la vez: un entorno físico en el que uno está realmente presente y otro entorno presentado por la tecnología en el que se encuentra uno presente digitalmente. Lo segundo es lo que se denomina Telepresencia.

La Telepresencia (Steuer, 1992) es la medida en la que uno se siente presente en un mundo mediado, más que en un entorno físico inmediato.

El uso de "telepresencia" fue usado en sus orígenes por Marvin Minsky (Minsky, 1980) para hacer referencia a la manipulación remota de objetos físicos a través de sistemas de tele operación. Esta primera definición no está reñida con la definición actual ya que al manipular ese objeto físico uno se siente presente sin estarlo.

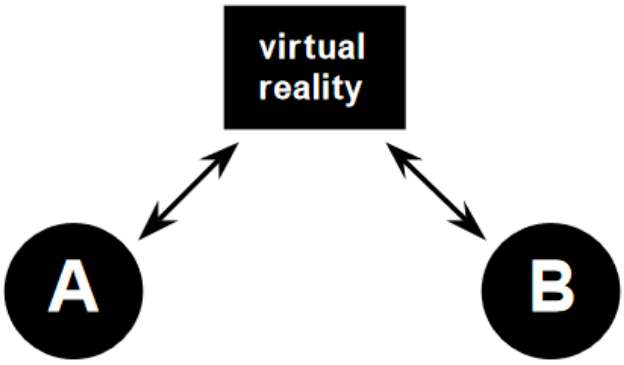

Ilustración 16. La Vista en la Telepresencia

Podemos tener ese mismo sentimiento de telepresencia al leer un libro, al tener una llamada telefónica o con una videollamada, escuchando música grabada en una sala o al aire libre, mirando a través de una pantalla lo que ve una cámara de video o al jugar a un videojuego. En todas estas experiencias tenemos a la vez una experiencia de Presencia, ya que todos estos ejemplos pueden darse sin salir de una habitación, y una experiencia de Telepresencia cuando experimentamos una presencia distinta a la presencia de nuestro cuerpo físico.

Por esto podemos definir la Telepresencia como la experiencia de presencia en un entorno a través de un medio de comunicación. (Steuer, 1992)

3.3. Variables tecnológicas que influyen en la telepresencia (Ilustración 11)

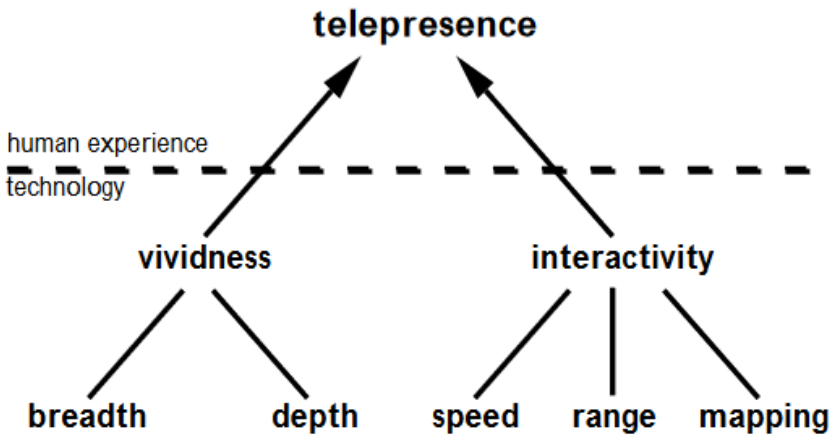

Ilustración 17. Variables tecnológicas que influencian la telepresencia

Vividez

La vividez es la riqueza representativa de un entorno mediado según se define por sus características formales. La vividez es impulsada por los estímulos que puede aportar la tecnología. Muchos factores contribuyen a la vividez, pero en este documento nos centraremos en la amplitud y en la resolución.

La amplitud es una función de la capacidad de un medio de comunicación para presentar información a través de los sentidos; oído, tacto, gusto, olfato, vista y equilibrio. Todas las aportaciones que pueda dar el medio pueden servir a formar una mejor experiencia, pero una activación simultánea de estos sentidos puede producir una redundancia que refuerza una experiencia concreta.

La resolución es la nitidez con que se percibe una imagen observada por un entorno mediado. La evolución de la tecnología ha conseguido que la resolución sea mejor año tras año aportando mejor vividez (Steuer, 1992).

Interactividad

La interactividad se define como la medida en que los usuarios pueden participar en la modificación de la forma y el contenido de un entorno mediado en tiempo real. Examinaremos brevemente los tres factores que contribuyen a la interactividad: la velocidad de interacción, el rango de interactividad y el mapeo (Steuer, 1992).

La velocidad de interacción, o tiempo de respuesta, es una característica importante de un sistema interactivo. La interacción en tiempo real representa claramente el valor más alto posible para esta variable: Las acciones de un usuario alteran instantáneamente el entorno mediado.

El rango de interactividad está determinado por el número de atributos del entorno mediado que pueden ser manipulados y por la cantidad de variación posible dentro de cada atributo. En otras palabras, el rango se refiere a la cantidad de cambio que se puede efectuar en el entorno mediado. Cuanto mayor es el número de parámetros que pueden modificarse, mayor es el rango de interactividad de un medio determinado.

El mapeo se refiere a la forma en que las acciones humanas están conectadas a las acciones dentro del entorno mediado. El mapeo es, pues, una función tanto de los tipos de controladores utilizados para interactuar con un entorno mediado como de las formas en que las acciones de estos controladores están conectadas con las acciones dentro de ese entorno (Steuer, 1992).

3.4. Videojuegos adaptados a personas con discapacidad visual

A la hora de crear videojuegos pensados para personas con discapacidad visual, lo primero para tener en cuenta es que diseñar juegos que funcionen para personas con discapacidad visual es un pequeño reto debido a que el principal canal de retroalimentación es, generalmente, visual.

Los videojuegos tienen que dar una buena sensación a los jugadores. La forma de describir esta sensación positiva es a través de la presencia y la zona (o flow). Ambos describen la sensación de inmersión por parte del usuario. (Archambault, Ossmann, Gaudy, & Miesenberger, 2007)

La zona está más orientada a la sensación de satisfacción que a una ilusión de estar en cualquier otra parte. Se puede definir como un estado de concentración, disfrute y absorción por parte de una actividad. Es el resultado del balance entre la ansiedad y el aburrimiento, producidas por dos aspectos del juego: el reto que supone y las habilidades del jugador (Chen, 2007; Johnson & Wiles, 2003).

3.4.1. Alternativas para hacer accesible un juego

Audio

De esta forma de hacer accesible un juego, surgen los audiojuegos (Audiogames). Estos juegos se diferencian de los videojuegos más comerciales en que su principal canal de información es el auditivo, quedando el canal visual relegado a un papel secundario.

Dentro de los audiojuegos, la adaptación puede ser mediante lenguaje no verbal (sonidos, música, pasos...) o por Text-To-Speech (TTS) que es una forma de sintetizar una lectura de un texto a través de software.

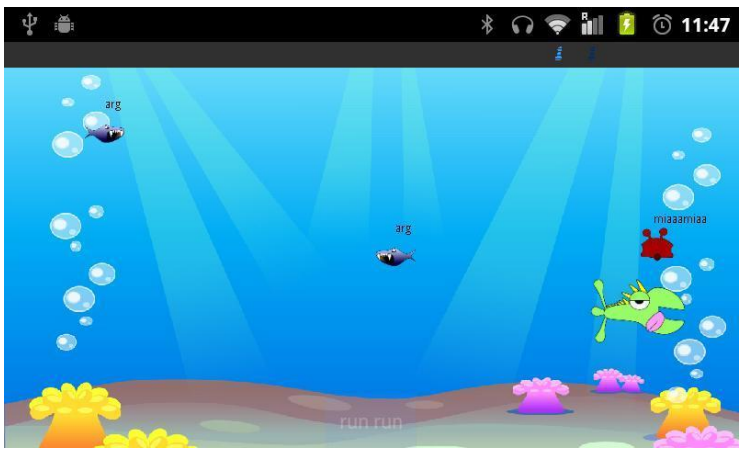

Ilustración 18, Imagen del audio juego Zadrodnik. El pez verde es el oyente y los otros pezes emiten un sonido. Este sonido solo es perceptible cuando el jugador se hacerca a la fuente del sonido (Pozuelo & Álvarez, n.d.)

Táctil

Pueden estar diseñados para ser utilizados a través de teclados de braille, dispositivos con una cartulina que represente la pantalla en función del tacto que tenga esa zona de la cartulina.

Háptico

Uno de los dispositivos a nombrar dentro de esta accesibilidad el The Phantom, utilizado para varios experimentos en torno a juegos (Archambault et al., 2007).

Ilustración 19, The Phantom Premium("The Phantom," n.d.)

3.4.2. Audio juegos (Audiogames)

La principal fuente de videojuegos adaptados para la gente con discapacidad visual se basa en la primera vía de adaptación.

Los audiojuegos están fuertemente influenciados por los juegos textuales (aventuras narrativas), de tal manera que aquellos audiojuegos más complejos, requerirán de texto para enseñar al jugador y para ponerle en contexto. Por otro lado, los juegos más simples, se pueden llevar a través de comunicación no verbal. (Archambault et al., 2007)

Actualmente, los audiojuegos pueden estudiarse en base a tres factores:

- Su dependencia a la comunicación verbal.
- Su dependencia a los mecanismos basados en el momento.
- Su dependencia en los mecanismos de exploración.

Existen páginas especializadas en la recopilación de estos juegos, principalmente desarrollados con flash. ("AudioGames Games List," n.d.)

Categorización

Los audiojuegos se categorizan siguiendo la categorización de los videojuegos no adaptados. De estos géneros de los videojuegos, se derivan sus equivalentes a los de los audiojuegos, recortando aquellas partes de la experiencia que no es posible adaptar.

Se distinguen cuatro géneros principales (Gaudy, Natkin, & Archambault, 2006; Natkin, 2006):

- Juegos de acción: Inspirados por los arcades. Requieren de destreza. Un ejemplo podría ser DDR (Recreativa de baile), pero se basa en la anticipación mediante canal visual. La forma de adaptarlo sería que el usuario tuviera que reconocer varios tipos de sonidos y actuar en función a ellos (Archambault et al., 2007). La diversión en estos juegos se basa no en completar un objetivo como tal si no en repetir la misma acción una y otra vez con una dificultad en constante crecimiento hasta llegar al *Game Over* y batir la puntuación. Un ejemplo es Tampokme.

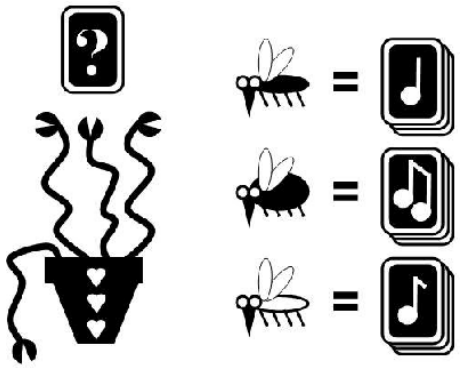

Ilustración 20. Pequeña guia del juego Tampokme (Gaudy, Natkin, Prado, Dilger, & Archambault, 2007)

- Juegos de exploración: Heredan de los videojuegos de aventuras, solo que el diseño del escenario y los puzles se tienen que minimizar o recortar por ser muy complicados. Generalmente, se mantiene algún pequeño puzle o recolección de objetivos, pero de manera secundaria, siendo lo principal la exploración del entorno (razón por la que se llaman de exploración) (Archambault et al., 2007). Un ejemplo es Super Egg Hunt. (Gaudy et al., 2006; “L-Works,” n.d.; Natkin, 2006)

- Juegos de simulación: Surgen de los juegos de estrategia, solo que a estos se les quita la parte de cartografía (descubrimiento del mundo). Los audiojuegos de simulación se centran en la experiencia de gestión (Archambault et al., 2007). Ejemplos de estos juegos son Simcarriere, Galaxy Ranger.

Ilustración 21. Captura del juego Simcarriere ("SimCarriere.com : jeu en ligne & online gratuit. Gérez une carrière d'avocat," 2009)

- Juegos de puzles: Son adaptaciones de juegos de mesa como el ajedrez, el monopoly, etc. Se centran en que el juego de la información al usuario para que este pueda tomar decisiones, sustituyendo la capacidad de ver el tablero por información suministrada a petición del usuario mediante Text-To-Speech (TTS).

3.4.3. Métodos para hacer accesible un videojuego

Se pueden conseguir indicaciones auditivas, tan buenas como la localización basada en el visual, a través de señales auditivas completas y bien fundamentadas que jueguen con las características del sonido para transmitir la información. Por ejemplo, se puede utilizar el volumen para representar si un objeto está cerca, si el volumen del sonido está alto, o si está lejos, si el volumen del sonido está bajo; o la posición en el eje Y del objeto haciendo ver que el objeto está más alto que nuestra posición, si el sonido es agudo, o está por debajo de nosotros, si el sonido es más grave (Balan, Moldoveanu, & Moldoveanu, 2015; Lessard, Paré, Lepore, & Lassonde, 1998). Además, se representaría con sonidos naturales sucesos que sean reconocibles, como unos pasos, el sonido de una puerta, golpearse con la pared, etc.

Las técnicas principales para hacer accesible un juego para ser jugado mediante sonido:

- Iconos auditivos: Sonidos simples y fácilmente reconocibles que representan objetos, acciones, etc. Proporcionan una gran variedad de tipos de información de manera simultánea.

- Auriculares (Earcons): Pequeñas piezas musicales que aportan información no verbal. Por ejemplo, un cambio de música puede representar la entrada a un combate, un cambio de sala, el descubrimiento de un tesoro.

- Text-To-Speech (TTS): Ya definido anteriormente, audios de voz pregrabados o reproducidos por un programa de reconocimiento de texto.

Estas técnicas dependen de la memoria auditiva del usuario para recordar qué representa cada sonido. Además, cabe decir que el lenguaje verbal no es universal, por lo que sirve como adaptación solo para los usuarios que dominan esa lengua (Dingler, Lindsay, & Walker, 2008; Ossmann & Miesenberger, 2006).

Tipos de sonido

Hay tres tipos de escucha aplicables a los videojuegos:

- Escucha causal: El jugador escucha el sonido y busca determinar su origen, características, etc.

- Escucha semántica: El jugador escucha el sonido y atiende a la información que este contiene. En nuestro caso sería principalmente proveniente de Text-To-Speech (TTS).

- Escucha reducida: El jugador escucha el sonido y se centra en sus características, como pueden ser el tono, timbre, intensidad, etc. (Balan et al., 2015; Chion, 2012)

Y a su vez hay dos maneras de ubicar el sonido en función del jugador. Se puede concebir el entorno de una manera alocéntrica, de tal forma que se describa a través del sonido las características de los objetos que hay alrededor; o de manera egocéntrica, si se centra la descripción del espacio de manera subjetiva y centrada en el usuario, priorizando el dar conocimiento a las personas con discapacidad visual de lo que les rodea desde una perspectiva en primera persona. (Balan et al., 2015)

3.4.4. Guías de creación Accesible

Las guías de adaptación de un videojuego para ser accesible se basan en el sistema de prioridades que bien se usa en software para la priorización de Historias de Usuario (HU):

- Must Have: Aquellos requisitos que son necesarios para que el juego sea accesible.
- Should Have: Aquellos requisitos que ofrecen una gran ayuda al usuario sin llegar a ser necesarios.
- May Have: Aquellos requisitos que suponen una ayuda para un grupo de jugadores.

Además, se tiene en cuenta las posibles discapacidades a las que se pueden adaptar: visual, audición, movilidad y capacidades cognitivas (Archambault et al., 2007; Ossmann & Miesenberger, 2006).

A pesar de esta categorización y guías de adaptación, existen medios que se dedican a evaluar la adaptabilidad de un juego en función de las diversas discapacidades, siendo estas las anteriores mencionadas excepto las cognitivas (“Dager System,” n.d.).

Además, se busca en un audiojuego que cumpla las siguientes especificaciones (Balan et al., 2015; Gaudy, Natkin, & Archambault, 2009):

- Que sea inmersivo y anime al usuario a avanzar.
- Que las pistas auditivas sean diferentes entre sí.
- Que no existan o haya pocas situaciones de Game Over que desanimen al usuario a seguir jugando. (Siendo una excepción juegos arcades, cuya diversión se basa justo en evitar constantemente el Game Over).
- Una forma, única y clara, de resolver el escenario.
- Un tutorial integrado en el juego que sea entendible para usuarios con discapacidad visual y que no use o limite el uso del Text-To-Speech (TTS).

Por último, las guías de creación accesible de las interfaces son las recogidas en el estándar WCAG 2.0. (Universidad de Alicante, 2008). Las Pautas de Accesibilidad al Contenido en la Web (WCAG) 2.0 abarcan una amplia gama de recomendaciones para hacer más accesible el contenido. Seguir estas pautas hará que el contenido sea accesible a una gama más amplia de personas con discapacidades, incluyendo la ceguera y la baja visión, la sordera y la pérdida de audición, las discapacidades de aprendizaje, las limitaciones cognitivas, la limitación del movimiento, las discapacidades del habla, la fotosensibilidad y combinaciones de éstas. Seguir estas pautas también hará que el contenido de su Web sea más utilizable para los usuarios en general.

Estás pautas se basan en cuatro principios:

- Perceptible: La información y componentes de la interfaz de usuario debe ser presentada de manera que el usuario la pueda captar por los sentidos.
- Operable: Los componentes de la interfaz de usuario y la navegación deben ser manejables de forma sencilla.
- Comprensible: La información y el manejo de la interfaz de usuario se debe poder entender.
- Robusto: El contenido debe ser suficientemente robusto para ser interpretado por una variedad de aplicaciones de usuario, incluyendo las ayudas técnicas (“Guidelines Page,” 2009).

3.5. Plataformas de creación de videojuegos

Construct3

Permite crear juegos de todo tipo mediante un sistema Drag & Drop apoyado en un asequible lenguaje de scripting y varias herramientas de apoyo como un motor de físicas o recursos gráficos de apoyo (“Game Making in Education,” n.d.).

Un sistema Drag & Drop (“Drag & Drop,” n.d.) se basa en poder ir eligiendo elementos preconstruidos o creados por el mismo desarrollador e ir colocando dentro del espacio del proyecto arrastrándolos desde una especie de inventario hasta la ventana que representa el proyecto en el entorno de desarrollo, pudiendo especificar así qué lugar ocupa cada objeto.

Funciona mediante HTML5 y puede extenderse a través de JavaScript. Es una herramienta más manejable pues puede usarse sin necesidad de instalación, a través de la página de desarrollo (“Construct 3,” n.d.).

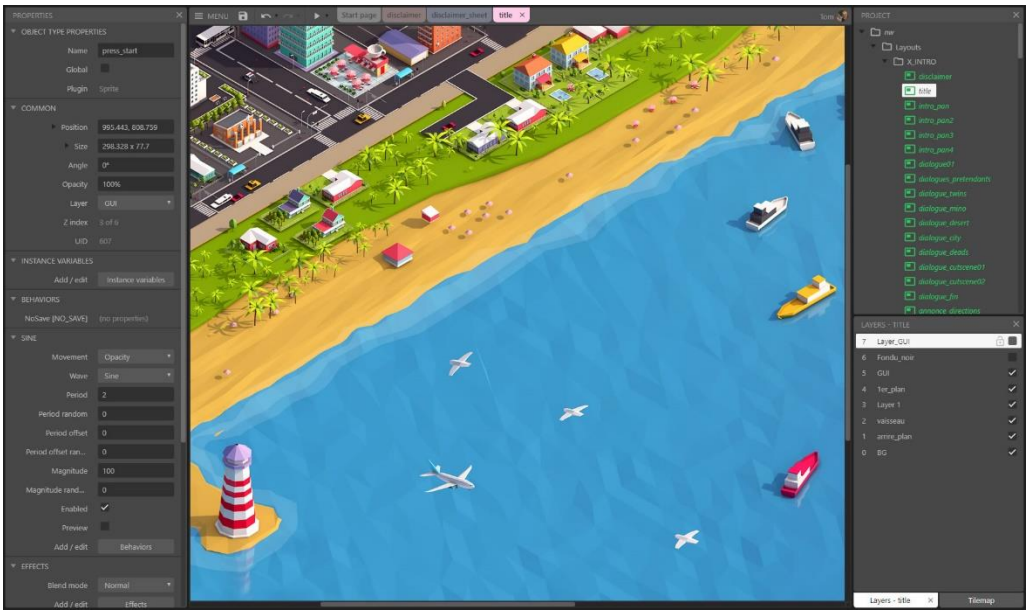

Ilustración 22, Captura de pantalla de Construct 3 (Construct 3, n.d.)

Game Maker Studio

Uno de los más completos y asequibles. Permite crear casi cualquier tipo de juego (incluso 3D) y existe muchísima documentación en Internet. Su lenguaje de scripting es opcional y hace que gane mucha profundidad. Perfecto tanto para novatos como profesionales.

Es bastante accesible a nuevos desarrollos pues tiene, como Construct2, un sistema de Drag & Drop que permite a personas con pocos conocimientos de programación utilizar la herramienta sin problemas. Se ha utilizado para desarrollos de varios juegos indies como “Cook, serve, delicious”, “Risk of Rain” o “Spelunky” (Moore, 2014).

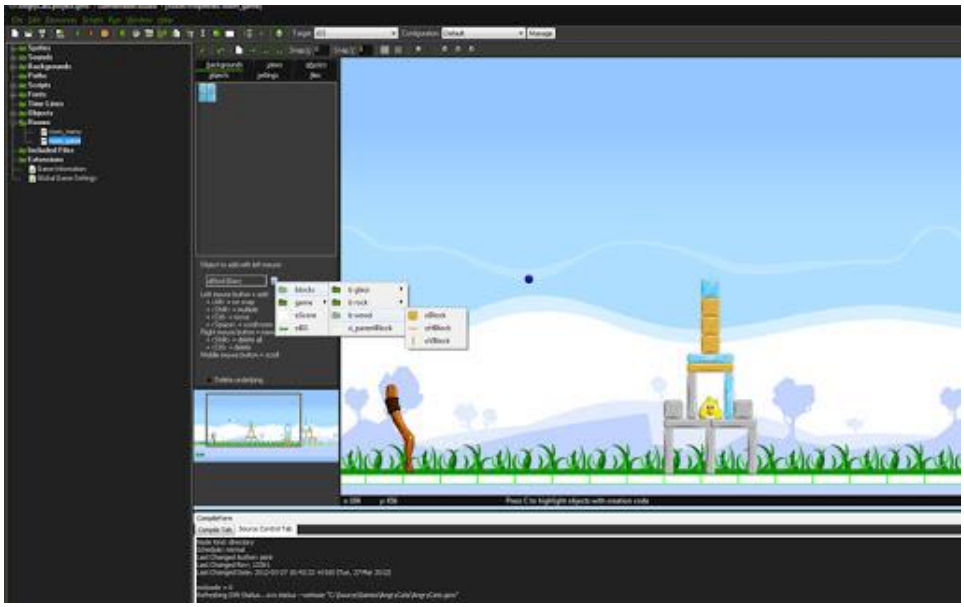

Ilustración 23, Captura de pantalla de Game Maker Studio(“Game Maker Studio,” n.d.)

Unreal Development Kit

El motor profesional para la creación de videojuegos 3D ha pasado recientemente a ser completamente gratuito (con un pequeño porcentaje para el estudio si hay beneficios). En sucesivas versiones se ha hecho tan asequible que es posible crear juegos sin la necesidad de escribir código.

Utilizado en muchísimos desarrollos, un par de ejemplos pueden ser “Killing Floor” o “Rock of Ages”.

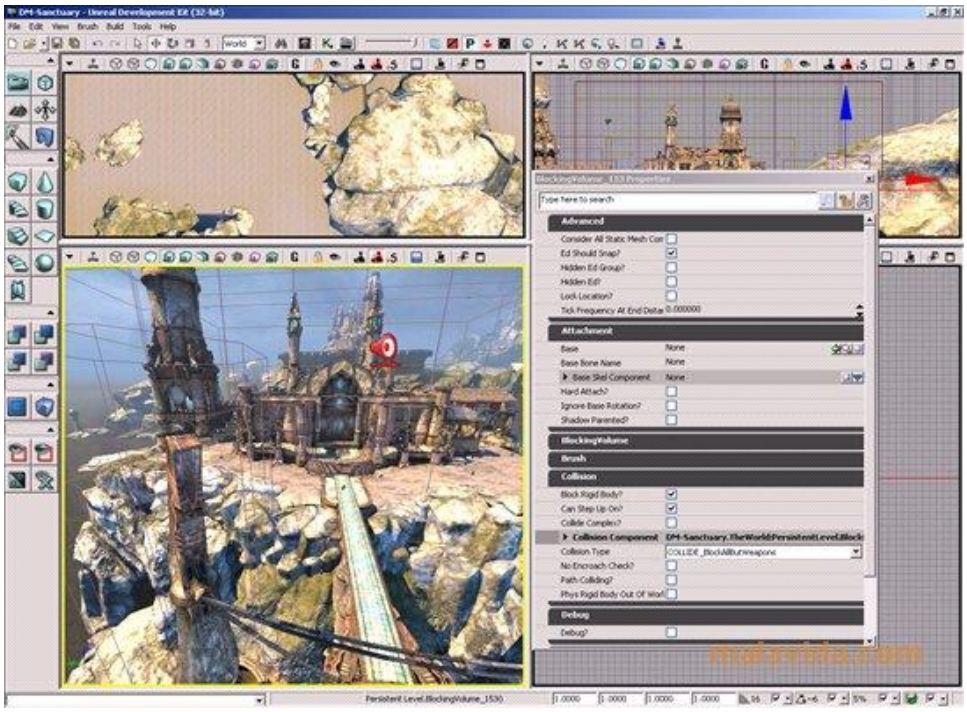

Ilustración 24, Captura de pantalla del Unreal Engine (“Unreal Engine,” n.d.)

Torque 3d

Completo entorno de desarrollo de videojuegos 3D de código abierto (Garage Games, 2012).

Se utiliza en el desarrollo de juegos, principalmente indies, con una relativa popularidad y cuyo desarrollo de videojuegos ha quedado un poco obsoleto, aunque su código sigue renovándose hoy en día (“Best of Torque,” n.d.).

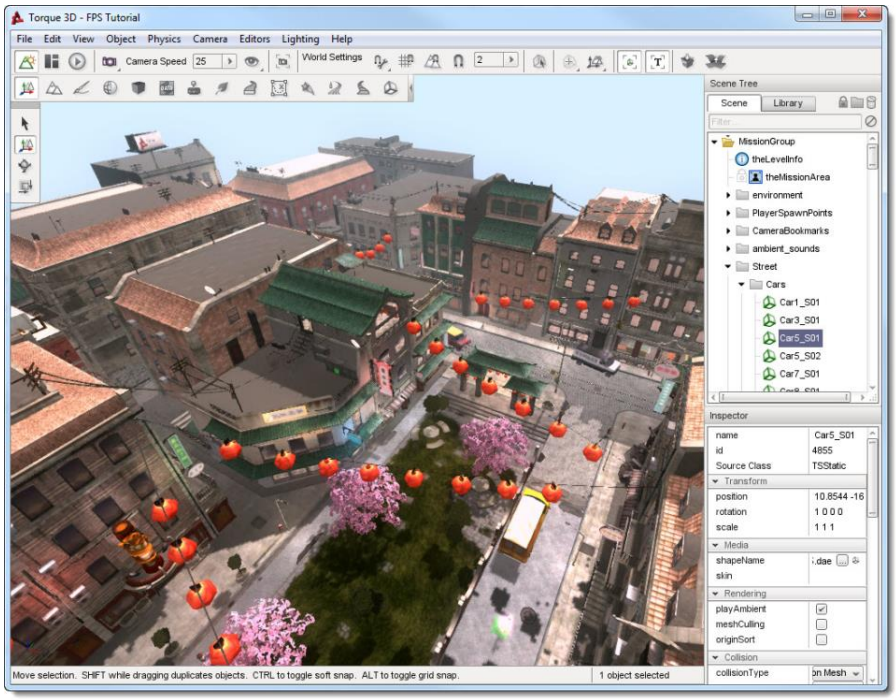

Ilustración 25, Captura de pantalla del Torque 3(Garage Games, 2012)

StencylWorks

Fantástico entorno de desarrollo integrado (IDE) en la línea de Game Maker o Construct2 con el aliciente de tener un sistema de objetos modular que permite descargar recursos para nuestros proyectos a través de una plataforma propietaria.

No requiere de programación real, si no que funciona principalmente a través de scratch. Sirve para cualquier tipo de plataforma, sin embargo, solo da soporte a juegos 2D. Se suele utilizar en el desarrollo de minijuegos, juegos flash y juegos para Smartphones (Armor Games, Kongregate y Newgrounds) (“Minigames make with Stencyl,” n.d.; “Stencyl,” n.d.).

Ilustración 26, Captura de pantalla de StencylWorks (“Stencyl,” n.d.)

Unity 3D

Permite crear todo tipo de juegos, pensada mayormente para el desarrollo 3D. Muy versátil, y su versión básica es totalmente funcional y permite exportar nuestras creaciones en Windows o mediante un reproductor web, aunque está soportado en cualquier plataforma de juego, incluso en las de realidad virtual. Algunos ejemplos de juegos creados con Unity son “Temtem”, “Risk of Rain 2”.

Ilustración 27, Captura de pantalla de Unity 3d (“Unity 3d,” n.d.)

3.6. Transformación de un juego no adaptado

Es importante recordar que no es lo mismo adaptar un juego desarrollado para un público más convencional que realizar un desarrollo de un juego cuya jugabilidad está planteada para ser jugada sin necesidad de la vista.

Mientras que un juego específicamente desarrollado para un público con discapacidad visual tendrá mecánicas más orientadas al resto de sentidos, un juego adaptado tratará de poner un parche a los estímulos visuales, sustituyéndolos por otros que intenten dar la misma información.

Un ejemplo de esta adaptación de un juego convencional a algo más adaptado puede ser el juego de lucha SkullGirls que, a través del sonido envolvente (Galvez Enrique, 2019), la diferencia entre las voces de los personajes y los iconos auditivos de los movimientos principales; señala la posición del otro jugador en el escenario, especificando donde se encuentra y si está realizando una acción. De esta manera, si el oponente está a la izquierda, sus sonidos sonarán más a la izquierda que los del jugador, y lo respectivo pasará si está a la derecha (Straub, 2014).

Ilustración 28. Skullgirls (López, 2012)

Además, es importante que los juegos que se adapten para el público con discapacidad visual contengan interfaces de usuario que faciliten la exploración del usuario por las mismas y adapten la experiencia de navegación en primera persona.

Para poder adaptar un videojuego a un público con discapacidad hay que afrontar los siguientes retos (Bierre et al., 2005):

- A la hora de tener que solucionar un puzle o tarea, ¿cómo se ofrece la información que en el juego original se ofrecería a través de textos o imágenes?
- Antes de empezar el juego, ¿cómo aprenderá el usuario a jugar?
- Si el tutorial inicial se hace a través de texto hablado, perdemos alcance de usuarios, ya que se queda limitado al idioma, ¿qué se puede hacer para evitar o reducir estos textos hablados?
- En función del hardware que necesite el juego se limita la cantidad de plataformas que pueden usarlo.
- En las situaciones de urgencia o peligro, ¿cómo se le notifica al jugador lo que está ocurriendo en detalle?

Un ejemplo de cómo hacer adaptable un juego que inicialmente no lo era es Audio Quake (“AGRIP,” n.d.; Matatk, 2012), una versión adaptada del FPS Quake que está en código abierto y libre para la descarga que hace retoques en el audio del juego y el Text-To-Speech (TTS) para que las personas con discapacidad visual puedan jugarlo.

Interfaces de Usuario accesibles

A la hora de crear un menú para un juego, lo mejor es hacer un menú accesible en todo momento basado en lógica de árboles que se use con unas teclas concretas. Un menú en lógica de árbol puede contener objetos, por ejemplo, que se pueda acceder a las distintas ramas u opciones dentro del menú moviéndonos de izquierda a derecha y seleccionando con una tecla, de tal manera que con tres botones se mapeara el control del menú. Un ejemplo de este menú es la selección de objetos en Monster Hunter (Biggs, Yusim, & Coppin, 2018; Krygier, 1994).

A la hora de crear la interfaz de usuario, se puede elegir entre 5 tipos de mapas en función de cómo se comportan con respecto a la entrada y salida, el WIMM (la ventana, los iconos, los menús y los mensajes) y con el mundo del juego (Balan et al., 2015; Krygier, 1994):

- MUD (Multi-user dungeon): Juego basado en texto. Los usuarios no interactúan con el mundo como tal, sino que van tecleando comandos con las acciones que tiene que hacer el personaje. Son audiojuegos, pues se puede reproducir el texto por Text-To-Speech (TTS) y escribir los comandos en lenguaje natural. El problema de estas interfaces es que limita el juego completamente a aquellos usuarios que controlan la lengua en la que está escrito (Balan et al., 2015; Biggs et al., 2018).

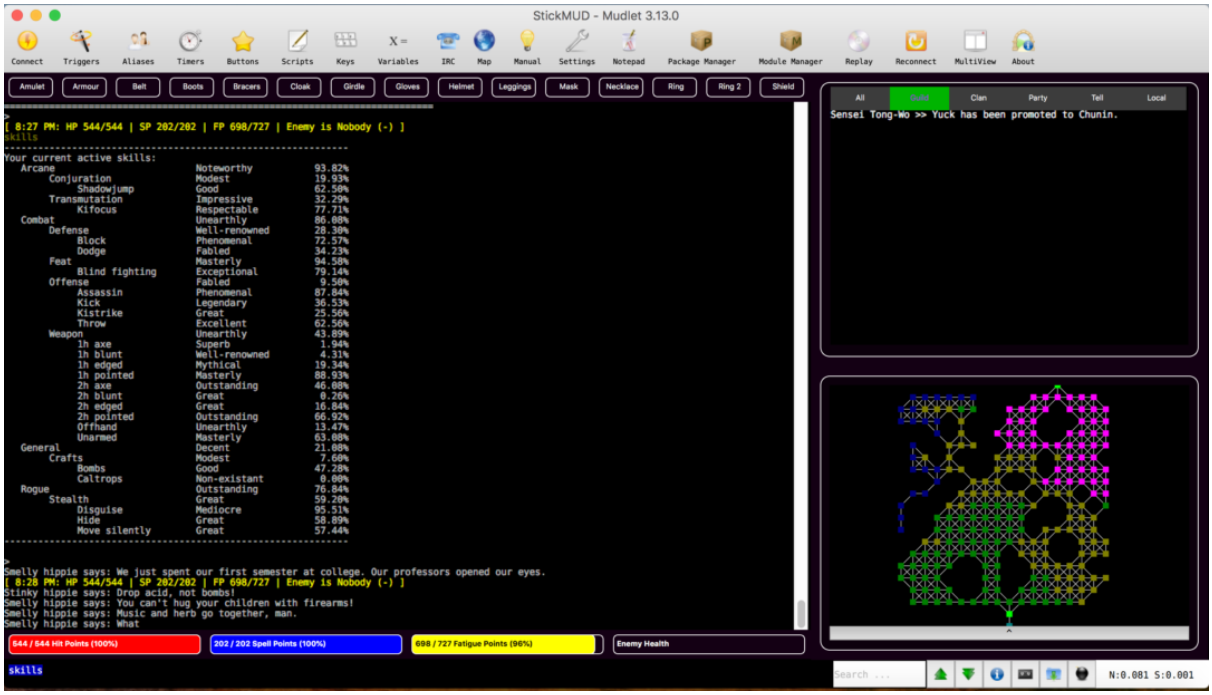

Ilustración 29, captura de pantalla del MUD ("StickMUD," n.d.)

- Tree-based: Se basan en moverse adelante, atrás, arriba o abajo en una estructura de nodos. Es la estructura por defecto en los menús, ya que los jugadores no tienen que acordarse de la posición de cada opción, si no de aquello que buscan (Balan et al., 2015).

Ilustración 30, Ejemplo de un menú Tree-Based del juego Minecraft ("Tree Node," n.d.)

- Basado en celdas: Se basa en coordenadas XY. El jugador se mueve en vertical y horizontal por un mapa de celdas conectadas. Las acciones se realizan con ciertas teclas cuando estás en la celda correcta (Balan et al., 2015; Zhao, Plaisant, Shneiderman, & Lazar, 2008).

Ilustración 31, Dos ejemplos de los distintos Monkey Island 2 en los que pulsando en una coordenada de la escena se despliegan las opciones ("Monkey Island 2: LeChuck's Revenge Complete Walkthrough," n.d.)

- Scroller: Como pudiera ser un juego del género de plataformas. Usa principalmente controles direccionales y teclas que permiten navegar entre menús o dan información específica, como el objetivo o la vida (Balan et al., 2015).

Ilustración 32, Ejemplo de menú tipo Scroller en el juego Mad Max ("Scroller Menú," n.d.)

- Primera persona 3D: Los componentes del mundo se colocan en torno al personaje y se utilizan sonidos para indicar la posición. También se utilizan sonidos concretos para detallar la superficie sobre la que nos movemos (Balan et al., 2015).

Ilustración 33, Ejemplo de Primera persona 3D de Last of Us 2. Receptor en azul y emisor del sonido en amarillo (Javi Andrés, 2020)

Navegación en primera persona

Una dificultad de los audiojuegos en primera persona es que los usuarios puedan detectar el origen de un sonido a la hora de jugar. Swamp es un juego que utiliza una vista de águila, de tal manera que no requiere de rotar los movimientos con la posición del usuario. Se tiene comprobado que de esta manera son capaces de ubicar a los enemigos y a los objetivos (Flowers, Turnage, & Buhman, 2005; Krygier, 1994).

A la hora de hacer la rotación del personaje, el usuario se puede desubicar por no saber o no tener calibrada la sensibilidad con la que el personaje rota y cuánto rota. Para evitar esto, sería conveniente que hubiera siempre un sonido que sirva de referencia.

3.6.1. Grado de discapacidad visual

Lo primero que debemos tener presente a la hora de hacer un videojuego adaptado es que no todas las personas con discapacidad visual la tienen en el mismo grado. Existen diferentes pérdidas de la capacidad visual que tienen unas u otras características y que impiden la realización de unas u otras actividades. Estos diferentes grados se pueden catalogar en tres términos: ceguera, visión reducida y daltonismo o discromatopsia (Asociación D.O.C.E., 2015; Bierre et al., 2005; Willings, 2019).

Dado que queremos orientar el proyecto para que llegue al máximo número posible de jugadores, tendremos que adaptarlo a una pérdida de visión total. Esta decisión no está reñida con abrir el alcance del proyecto a aquellos jugadores con discapacidades menos restrictivas y que puedan disfrutar de los gráficos de un videojuego y de las mecánicas que requieran de los mismos. Por lo que no se trata de hacer un proyecto exclusivo para personas con discapacidad visual, se trata de hacer un juego inclusivo que pueda disfrutar cualquiera.

Lo primero a diferenciar es que las capacidades visuales pueden ser relativas a un solo ojo, con lo que se denominaría uniocular, o afecta a ambos ojos, con lo que se le denominaría binocular.

Una buena visión uniocular es compatible con el desempeño de las actividades cotidianas, afectando principalmente a la percepción de la profundidad. Esta percepción de la profundidad no es necesaria para la visualización de una pantalla, por lo que no sería un obstáculo para poder disfrutar de un videojuego.

Además de eso, existen las isópteras, que son las áreas de igual agudeza visual, es decir, con la misma capacidad de enfoque en todos los puntos interiores de esa área circundante. A medida que nos alejamos del punto central de enfoque del ojo o mácula, la agudeza visual va disminuyendo isóptera a isóptera.

Aparte de las siguientes discapacidades visuales, el principal problema visual es la pérdida de la ya definida agudez visual. Esta pérdida de agudeza visual no tiene por qué ser total, en cuyo caso se considera ceguera, si no que puede ir marcándose por porcentajes de agudeza visual.

También hay que diferenciar que cada ojo tiene una agudez visual y unas isópteras diferentes, por lo que la visión del jugador, la que está marcada por la binocular, es la que define qué grado de discapacidad visual tiene. Cuanto peor es la agudez visual más le costará al usuario enfocar y visualizar la pantalla, por lo que necesitará de fuentes de letra más grandes, formas más descriptivas e iconos más diferenciados entre sí.

Por un lado, tenemos la pérdida del campo visual que proporcionan los ojos, también llamado isóptera periférica. Para comprender esto tenemos que saber que el campo visual de un ojo se compone de una zona central (campo visual central), que a su vez se compone de la isóptera central de unos 30 grados y de una zona periférica que va de los 60º a los 90º. Esta pérdida puede ser un problema relativo a la hora de jugar. Estos problemas vienen del tamaño de la pantalla y la distancia con la misma que con las técnicas y la tecnología de adaptación ("Ceguera y pérdida de la visión," n.d.).

Y, por otro lado, tenemos los escotomas que son la aparición de manchas, de zonas borrosas, etc. que suponen la pérdida de áreas del campo visual. Pueden estar ubicados en la zona periférica o en la central, suponiendo un obstáculo para la visualización de la pantalla (Asociación Mácula Retina, 2019).

Por último, tenemos la alteración de la visión de los colores o discromatopsia. Existen varias clases de esta alteración que hacen que se confundan distintas gamas cromáticas. Ésta alteración visual supone un problema a la hora de diferenciar las entidades en pantalla, por lo que se necesitan medidas para hacerlas más diferenciables, como códigos de colores de distinta tonalidad (que no se lleguen a confundir en caso de sufrir dicha alteración) o elección de paletas de colores diferenciadas para cada variante de discromatopsia (Merino, 2016; Romagosa, 2017).

3.6.2. Importancia de esta adaptación

De todas formas, es importante contar con videojuegos convencionales adaptados a un público con discapacidad visual, pues permite incluir en la experiencia de estos juegos a un público que, de otra manera, quedaría apartado.

Uno de los retos importantes que nos planteamos es hacer que la capacidad de ver la pantalla y distinguir lo que hay en ella no suponga una gran ventaja para aquellos que puedan hacerlo contra los que no pueden, las personas con discapacidad visual, parcial o total.

La principal manera de lograr esto es reducir la cantidad de información obtenida a través de la vista. Por ejemplo: creando escenarios donde se quedan en completa oscuridad de manera inesperada, a través de objetos y enemigos invisibles, etc.

Y por el contrario aumentar la información del entorno de la interacción con este. Por ejemplo: llevar la información de su posición, su movimiento y sus acciones a unas claves acústicas o iconos auditivos.

3.7. Conclusiones

En el mercado de los videojuegos hay una extensa y amplia diversidad de videojuegos que podrían con mayor o menor esfuerzo adaptarse o prepararse inicialmente para abarcar a un público con discapacidades visuales. Esta adaptación debe hacerse pensando en el tipo de juego que se va a hacer y en lo que conlleva dicha adaptación.

En el mercado ya hay juegos que han conseguido adaptarse de una manera u otra, por tanto, es posible.

Por otro lado, hay juegos que se han centrado en las personas con discapacidad visual, pero estos no resultan atractivos para los que no tienen discapacidades debido a varios factores; el choque inicial que tiene el pasar de un juego a otro, la falta de interfaz...

Con esto llegamos a la conclusión de que es posible adaptar los juegos ya creados para que puedan jugarlos las personas con discapacidades visuales, pero ¿y si se hicieran los juegos desde sus cimientos pensando en la adaptabilidad?

Si se hiciera desde sus cimientos pensando en la adaptación se rompería la frontera que hay ahora mismo entre los juegos más comunes y los juegos pensados para personas con discapacidad. Hay que trabajar en aunar estos dos grupos de juegos para conseguir que tanto personas sin discapacidad como personas con discapacidad puedan disfrutar en igualdad de condiciones, que puedan experimentar y que puedan compartir este pasatiempo sin la barrera que puede ser el tener una minusvalía.

4. Plan de Desarrollo

En este capítulo se describirá el plan de desarrollo que ha seguido el equipo para la realización del videojuego. Una demo final del mismo puede verse a través de este enlace https://www.youtube.com/watch?v=gy6cgysASGo

4.1. Fase de investigación

La primera fase de desarrollo se ha dedicado a estudiar la tecnología que se usará para el desarrollo del proyecto y a hacer un prototipo de la idea principal del proyecto.

Este prototipo se denomina "UME", siglas de "Unidad Mínima de Experiencia", y es una demostración que contiene las mecánicas y el contenido más básico para que sea jugable. Esto ayuda a generar un '*core*', o base, sólido desde el que empezar a desarrollar para lo que será el producto final.

Se decidió utilizar Unity 3D como plataforma de creación de videojuegos. Esto es porque Unity ofrece mucha documentación, tutoriales y contenido de manera gratuita. Es uno de los motores más usados actualmente y es el recomendado para personas nuevas en el sector de la creación de videojuegos.

Como se ha mencionado con anterioridad, se desarrollará un juego acorde al género *First Person Shooter* (FPS), con dos mecánicas principales:

- Movimiento en 3D de un personaje en primera persona.
- Combate contra enemigos.

Durante esta primera fase, el equipo se dividió en dos subgrupos. Dos personas investigarían la plataforma (Unity), sus características y cómo utilizarla. Los otros dos empezarían recolectando información, a través de artículos de investigación y blogs divulgativos, sobre accesibilidad en videojuegos (ejemplos de juegos adaptados, técnicas y estándares de la industria para la accesibilidad, etc.).

Objetivos iniciales

En este punto inicial teníamos decididas las bases del videojuego, centrándonos en los siguientes puntos:

- El juego final ha de ser entretenido tanto para personas videntes como personas ciegas de tal manera que unifique el ocio de ambos colectivos.
- Queríamos un juego de acción, rápido y que supusiera un reto para el jugador.
- El sistema de combate debía ser entretenido, que no se base en turnos, movimiento por celdas o cualquier otra cosa que diera la sensación de ser demasiado cuadriculado. Queríamos mantener la libertad de movimiento característica de los *FPS*.

- El aspecto gráfico tendría que ser aceptable, lo suficiente para que llamara la atención a personas videntes o cuya discapacidad visual no es total, pero que no nos llevara la mayoría del tiempo de desarrollo del proyecto ya que nuestro público principal seguía siendo personas invidentes y el aspecto gráfico no suponía una prioridad tan importante como la accesibilidad.
- Queríamos que las mecánicas del juego se basaran en la realidad virtual y tratar de explorar qué posibilidades de adaptabilidad nos podía llegar a ofrecer esta tecnología, transmitiendo información en función de la posición y orientación de la cabeza del jugador, el movimiento de los mandos y su posición en el juego.

Por otro lado, dudábamos entre varias posibilidades en torno a:

- Dificultad del juego, había que intentar encontrar el punto medio entre dificultad y entretenimiento, de tal manera que no fuese frustrante si no conseguían los objetivos, pero sin ser muy fácil.
- Qué armas podríamos usar, pues barajamos la opción de usar armas cuerpo a cuerpo manejadas a través del movimiento de las propias manos del jugador, usar armas de fuego que tuvieran un sistema de ayuda de apuntado o usar armas mágicas que permitían adaptar la experiencia a las necesidades con proyectiles que fueran directos a los enemigos o facilitaran la experiencia de otra forma.

Prototipo

Para el desarrollo de la UME, el equipo se centró en la creación de un entorno simple y cerrado, una habitación de cuatro paredes en las que periódicamente se generaban enemigos simples. Con este prototipo teníamos cubiertas varias mecánicas esenciales para el núcleo del juego:

- Movimiento del jugador con el teclado y cámara en primera persona controlada por el ratón.
- Ataque del jugador con arma a distancia, apuntar y disparar.
- Enemigos. Generados automáticamente y con un comportamiento simple (seguir al jugador).
- Entorno 3D con paredes y suelo que encierren la acción.

Todo esto se hizo desde cero, tanto el movimiento del personaje y de la cámara como el comportamiento de los enemigos.

Aunque muy simple, este desarrollo nos servía como toma de contacto con Unity para comprender el entorno de la herramienta, comprender cómo funcionan sus ficheros, su lógica y poder desarrollar nuestro juego desde una pequeña base abstracta, que nos sirviera para poder ir avanzando poco a poco con cada iteración, añadiendo contenido y mecánicas mejoradas al juego.

Todos los recursos utilizados, tanto gráficos como sonidos, para realizar esta primera versión se han obtenido de repositorios de uso libre, accesibles desde Unity a través de su “Asset Store” una tienda donde se publican modelos, scripts, imágenes etc.

En la Ilustración 31 se puede ver el aspecto del prototipo inicial que desarrollamos durante esta fase.

Ilustración 34 Captura del prototipo creado

Investigación

El equipo de investigación se centró en la búsqueda inicial de la información básica que nos serviría para comprender el estado del arte dentro de la creación de los videojuegos accesibles, centrándose en obtener información para poder responder preguntas como:

- ¿Cómo suplir la información visual que ofrece la pantalla y las imágenes a través de métodos alternativos?
- ¿Qué tecnología existe actualmente para poder suplir esta incapacidad de recibir información de la pantalla?
- ¿Qué son los audiojuegos? ¿Cómo se desarrollan? ¿Qué posibilidades tenemos dentro del desarrollo de estos?
- ¿Qué tipos de juegos orientados a personas invidentes existen en la actualidad y cuáles son las mecánicas que hacen entretenido el juego para personas con discapacidad visual?
- ¿Cómo funcionan las señales auditivas dentro de los videojuegos? ¿Qué pueden indicar y en qué medida?
- ¿Qué tipos de señales auditivas existen para poder proveer al jugador de la información necesaria para poder entender el entorno del juego?
- ¿Qué información puede aportar cada tipo de sonido?
- ¿Qué patrones de desarrollo tenemos que seguir para la creación de un juego que sea accesible para una persona invidente?
- ¿Qué requisitos tienen que cumplir las funcionalidades que incluimos para que sean accesibles?
- ¿Cómo desarrollar una interfaz de usuario que sea accesible?
- ¿De qué manera podemos desarrollar una navegación que permita jugar a un jugador invidente?

El día 11 de octubre de 2019 el equipo asistió a una reunión, en el Centro de Tiflotecnología e Investigación de la Organización Nacional de Ciegos Españoles (ONCE-CTI) en Camino de las Hormigueras 172, Madrid. En esta reunión se dio una charla sobre la utilización de nuevas tecnologías por parte de personas con ceguera.

Esta reunión sirvió para orientarnos en temas de integración a través del videojuego y mostrarnos los precedentes de desarrollo de videojuegos adaptados. Durante esta reunión, se mostraron algunos videojuegos orientados a personas con discapacidades visuales y pudimos notar que estos eran de una extremada simpleza en las mecánicas y el aspecto gráfico era pobre, usando imágenes creadas con herramientas simples.

Ejemplos de estos juegos son Feer o Blind Legend, descritos previamente en la introducción del documento. Tenían en común que eran juegos de móvil, que se servían de la vibración de este o del sonido envolvente que se podía obtener conectando unos auriculares al dispositivo y cuya forma de interactuar con el videojuego es deslizando el dedo por la pantalla en la dirección correcta.

En un inicio, el aspecto gráfico del videojuego puede no parecer importante, pero hay que contar con que no todas las discapacidades visuales requieren necesariamente de una pérdida total de la visión, es decir, que algunos de nuestros usuarios objetivo sí que podían disfrutar de este apartado y era importante cuidarlo un mínimo. Además, uno de los objetivos iniciales era hacer que el juego fuera inclusivo y eso pasaba por hacerlo atractivo también a personas videntes, que están muy acostumbradas a tener un estímulo visual.

Tras la investigación sobre estas preguntas y la reunión con la ONCE, pudimos concluir que:

- La mejor manera de suplir los estímulos visuales era con estímulos auditivos, lo suficientemente detallados, intuitivos y diferenciados los unos de otros como para que el usuario pueda entender qué representan.
- La tecnología de sonido envolvente era la mejor manera de implementar estos estímulos auditivos, jugando con los tonos y las intensidades para darles una posición en el espacio que rodea al jugador.
- Los audiojuegos son aquellos juegos desarrollados centrando la información que se le aporta al jugador en el canal auditivo y que suponen un modelo para nuestro desarrollo.
- Los juegos adaptados se basan en la exploración, en la estrategia más calmada, en la repetición de acciones simples a su debido momento y en la adaptación de juegos de mesa o puzles.
- Las señales auditivas son sonidos que se pueden oír en la vida real, que describen una acción o un objeto específico; pequeñas composiciones musicales que se repiten al suceder un evento, o textos narrados, para instrucciones más extensas.
- Hay unas guías y patrones con los requisitos que tienen que cumplir las interfaces accesibles y que hay tipos de menús para un videojuego que permiten un acceso más sencillo.
- Hay sistemas de guía que permiten a los usuarios invidentes facilitar la tarea de exploración a lo largo de un nivel, un laberinto o una sala. Son ejemplos los sonidos del entorno, sistemas de guía que dicen por dónde hay que ir de manera aproximada, balizas que señalicen las salidas de una sala, etc.

Por último, se hizo una rápida investigación sobre las diferentes mecánicas de realidad virtual (VR) que podíamos utilizar para nuestro desarrollo.

Existían múltiples opciones:

- Uso de gafas de VR con un controlador tradicional (mando de consola). De esta manera las gafas proporcionarían una interfaz para el movimiento de la cámara, pero el usuario realizaría las acciones a través del mando. Esta opción era poco interesante, ya que no explota las posibilidades actuales de la tecnología VR.
- Uso de gafas y mandos de VR. Los mandos de VR se controlan con las manos, teniendo la posibilidad de replicar el movimiento que el jugador haga con los brazos. Además, poseen elementos compartidos con los mandos tradicionales, como botones y joysticks.
- Uso de controladores hápticos, estos son mandos con forma de guantes, que imitan en el juego el movimiento dactilar del jugador. Muy interesante para hacer juegos en los que el personaje no se mueva, pero sí tenga que interactuar con muchas interfaces.

Finalmente se decidió la segunda opción ya que nos brindaba la posibilidad de que el jugador pudiera usar el joystick del controlador para mover a su personaje en el espacio sin necesidad de utilizar mecánicas de teletransporte, que podrían resultar muy confusas y difíciles de explicar a una persona ciega. Además, el jugador también podría utilizar los mandos para orientarse “tocando” lo que hubiese a su alrededor.

4.2. Fase de implementación

Tras la primera fase de investigación y el desarrollo de la UME llegamos a la conclusión de que queríamos que nuestro juego fuese un FPS con las mecánicas de combate propias del juego. Pero también decidimos meter como elemento principal laberintos, para explorar las formas de orientación en un entorno 3D de realidad virtual.

4.2.1. Las bases del juego.

Para crear una base para el juego decidimos centrarnos inicialmente en 5 puntos iniciales:

- **Laberinto**: El juego debería generar de manera procedural un laberinto con una entrada y una salida. Este algoritmo, explicado más adelante en el documento, se repetiría en diversas posiciones para generar varios laberintos a lo largo del nivel.

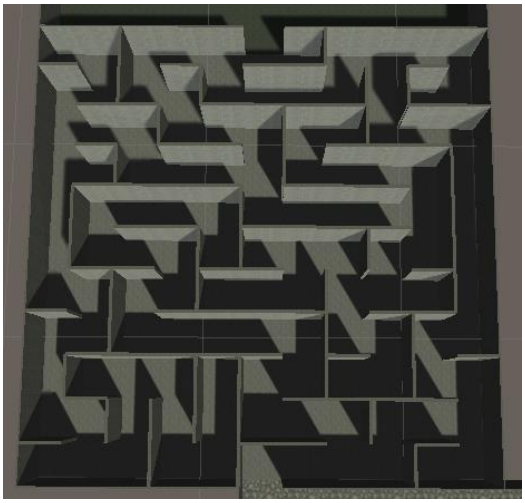

Ilustración 35. Laberinto ya creado.

- **Enemigos**: Creación de un enemigo simple, con el que el jugador pueda interactuar y combatir. Con una generación de sonidos reconocibles para el usuario y con los siguientes estados:

    - **Idle**: O “esperando”. Es el estado inicial del enemigo. El enemigo realiza periódicamente un sonido calmado y descriptivo en función del tipo de criatura de la que se trata, con el fin de que el jugador pueda oírlo y detectar su presencia. Además, este estado es al que el enemigo tiene que regresar desde los estados de caminar o correr antes de ejecutar un ataque al jugador, es decir, el enemigo tendrá que quedarse quieto para poder realizar un ataque al jugador.

    - **Walk**: O “caminar”. El enemigo permanece en este estado en el caso de que el jugador no se encuentre dentro del área de detección del enemigo. En este estado, el enemigo permanece paseando por la sala sin rumbo fijo, sin hacer nada más. Durante este estado realiza el mismo sonido que en el estado “idle”.

- **Run**: O “correr”. Es el estado en el que entra cuando el jugador entra en el área del enemigo. En este estado, el enemigo persigue el jugador, usando un camino formado por una línea recta entre él y el jugador. El sonido que emite el enemigo en esta fase es similar al de los estados anteriores, pero sonará más alto, más agresivo y más frecuentemente. Es importante que el sonido sea similar al del resto de estados para que el jugador pueda entender lo que está sucediendo con el enemigo.

- **Attack**: “Ataque”. Es el estado en el que el enemigo entrará cuando esté lo suficientemente cerca del jugador como para atacar. En este estado, el enemigo realizará una animación de ataque y se ejecutará el comportamiento de ataque, restando vida al jugador. En este estado, el enemigo realizará otro sonido característico para informar al jugador de lo que está pasando.

- **Muerte**: No es un estado como tal ni realiza un sonido, es más el evento en el que el personaje enemigo desaparece ya que el jugador pulsa el botón para atacar, se acciona la lógica del ataque y, si detecta un enemigo justo en frente de él, (dependiendo del arma, lo detectará a más o menos distancia) hará que se llame a este proceso.

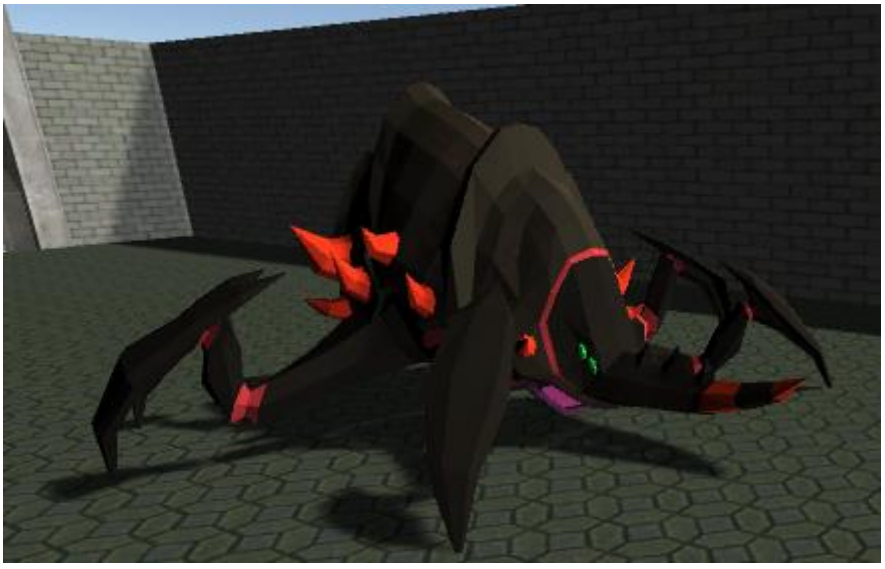

Ilustración 36 Enemigo del juego

- **Interactuar con objetos en el suelo**: En el planteamiento inicial del juego se pensó en recompensar al jugador en distintas partes del laberinto con bonificaciones que afecten a su fuerza de ataque, a su vida o a sus herramientas, por lo que vimos necesario empezar a implementar una forma de interactuar con estas mejoras. Para esto, se tuvo que implementar una lógica que almacenará los objetos dropeados (aquellos que están en el suelo, habilitados para cogerlos) en una lista interna y que se comprobará periódicamente su posición relativa al jugador. Si estaba lo suficientemente cerca del jugador, aparecerá un mensaje que notificará que ese objeto se podía recoger y qué tenía que hacerse para poder interactuar con él.

Ilustración 37. Interacción con objetos del suelo.

- **Arma cuerpo a cuerpo**: Un arma inicial que llevaría el jugador y que usaría desde el comienzo para defenderse o para atacar a los enemigos antes mencionados. Lo primero que hicimos es importar un hacha prediseñada desde repositorios de uso libre para evitar el largo proceso de diseñarla desde cero.
  También se creó un controlador de animaciones para el arma con tres estados:
    - Espera: El jugador tiene el arma seleccionada, pero no realiza ninguna acción.
    - Ataque: El jugador pulsa sobre el botón de disparo("FIRE1"), y el hacha se mueve simulando un ataque.
    - Desenfundar: Animación que serviría para el cambio de arma.

  Tras tener todos los movimientos controlados pasamos a crear un script que nos permitirá atacar con el arma. El script sencillamente detectará que un input llamado "*Fire*" ha sido disparado y disparará otro evento que hará que el hacha pase al estado de ataque activando así la animación. Si hay un enemigo delante del jugador, este muere. Para esto, aprovecharemos el sistema de etiquetas que tiene Unity para clasificar todos los objetos de la escena. Si cuando se realiza la acción de ataque hay una entidad con la etiqueta de 'enemigo' a una distancia relativamente corta, se ejecutará el comportamiento del enemigo de ser herido.

- **Movimiento a través de un periférico**: Queríamos afrontar en la base del programa esta decisión para poder tener una base para las posteriores implementaciones con los mandos de VR. Así que inicialmente intentamos añadir un mando al juego para controlar el movimiento del personaje, el control de cámara y la acción de atacar. Unity permite una conexión rápida del mando de Xbox360 por lo que solo había que añadir los inputs necesarios a Unity para que mapear los botones, los gatillos y los joysticks del mando para su uso en el juego. Al configurar los controles a través de inputs se hizo más fácil la implementación de los controles del juego tanto a través del teclado como a través del mando.

4.2.2. Segunda fase

En este punto del desarrollo, la UME estaba acabada, y comenzamos a darle protagonismo a la parte de accesibilidad del proyecto. Teníamos un videojuego, el objetivo era convertirlo en un videojuego accesible.

Para ello nos fijamos varios objetivos:

- **Control VR**: En el control de la realidad virtual tenemos que diferenciar dos conceptos diferentes, la posición del jugador y su orientación.
  Para controlar su posición en el espacio, los jugadores utilizarán joysticks que, ya que serán el mejor método dentro de los estándares de movimiento actuales en la realidad virtual, frente al movimiento por tele portación, el cual, como ya hemos mencionado, puede ser desconcertante para los usuarios invidentes.
  Para poder orientarse en el espacio queremos rehusar de todo tipo de movimiento de cámara por controladores que puedan hacer desorientarse al jugador. El movimiento de cámara irá completamente manejado por la orientación de las gafas de realidad virtual. De esta manera, el jugador sabrá en todo momento hacia qué punto está mirando y avanzando.

- **Sonido 7.1**: Se usará un sistema envolvente de audio que haga distinguible el origen del sonido. De esta manera tanto al jugador vidente como invidente le será más inmersiva y adaptable la experiencia de juego. El audio tendrá que generarse con una posición relativa a la cabeza del personaje para poder precisar mejor la detección de la dirección de los sonidos que se originen en el juego.
  En esta área Unity nos brinda la opción de generar puntos de sonido, o asignar un controlador de sonidos a cualquier objeto dentro de la escena. Y con un simple *tick*, hacer que ese sonido se convierta en "envolvente" y se comporte de distinta manera con respecto a la situación del jugador.
  Todos los sonidos del juego utilizan esta característica, siguiendo el patrón logarítmico que por defecto se encuentra en Unity, que hace que el sonido crezca rápidamente en volumen (de forma logarítmica) cuando el usuario se acerca al origen.

- **Boss**: Con esto nos referimos con el enemigo al final del juego. El *boss* será distinto al resto de enemigos; tendrá más vida, se encontrará en la última sala, será más grande, tendrá un patrón de movimiento distinto y producirá sonidos distintos.
  Empezamos creando un *boss* con colores distintos para diferenciarlo del resto de enemigos, lo escalamos para que sea más grande y lo posicionamos en el centro de la última sala.

Con este primer paso empezamos a configurarlo para que siga un patrón de movimiento. En este patrón tendremos distintos estados en los que puede estar:

- **Idle**: O "esperando" es el estado inicial después de su construcción dentro del juego, al entrar en este estado cargará sus características y se preparará para su siguiente estado. En el transcurso de esta fase detectará en cada ciclo la posición del jugador y a qué distancia está de él, si el jugador se encuentra dentro de su radio de acción pasará al estado "*Run Forward*". Si en este estado recibe daño pasará al estado "*Hurt*" y volverá a este estado. En cada ciclo también se ejecutará una animación de espera y producirá un sonido característico.

- **Run Forward**: Al principio de este estado el *boss* registra la última posición del jugador y en cada ciclo el *boss* avanzará hacia esa posición. De igual forma el *boss* realizará una animación de movimiento y generará un sonido de pasos para indicar que ha cambiado de estado y que se está moviendo. Cuando el *boss* alcance el punto en el que se encontraba el jugador al principio de la transición o tenga cerca al jugador o cuando alcance su límite de rango el *boss* pasará al azar al estado de "*Smash Attack*" o al "*Stab Attack*" y atacará. Si en este estado recibe daño pasará al estado "*Hurt*" y volverá a este estado.

- **Smash Attack**: En la fase inicial el *boss* realizará un sonido avisando al jugador de que va a realizar un ataque. Después en este estado el *boss* se queda momentáneamente en la posición y ejecuta una animación en el que realiza la acción de golpear el suelo. Este golpe al suelo se traduce en daño si el jugador se encuentra dentro del rango de acción y se produce un sonido que permite identificar el tipo de ataque que ha realizado. Tras esto el *boss* pasará automáticamente al estado de "*Return*".

- **Stab Attack:** En la fase inicial el *boss* realizará un sonido avisando al jugador de que va a realizar un ataque. Después en este estado el *boss* se queda momentáneamente en la posición y ejecuta una animación en el que realiza la acción de embestir hacia delante. Esta embestida se traduce en daño al jugador si se encuentra delante del *boss* y se produce un sonido que permite identificar el tipo de ataque que ha realizado. Tras esto el *boss* pasará automáticamente al estado "*Return*".

- **Return**: En este estado el *boss* se moverá a su posición inicial y pasará de nuevo al estado de "*Idle*".

- **Hurt**: Es un estado momentáneo en el que estará cuando reciba daño en los estados de "*Idle*", "*Run Forward*" o "*Return*". Al iniciar este estado el *boss* hará un sonido y una animación que nos indicará que ha recibido daño de tus ataques.

- **Die**: El estado final que experimentará el *boss*. El *boss* pasará a este estado cuando sus puntos de vida lleguen a cero. Cuando llegue a este estado realizará una animación y un sonido que dejará claro que el *boss* ha muerto.

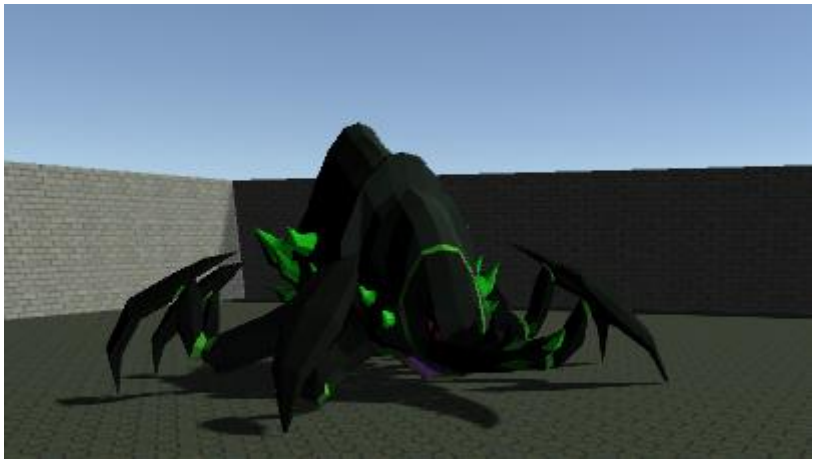

Ilustración 38 Boss del juego.

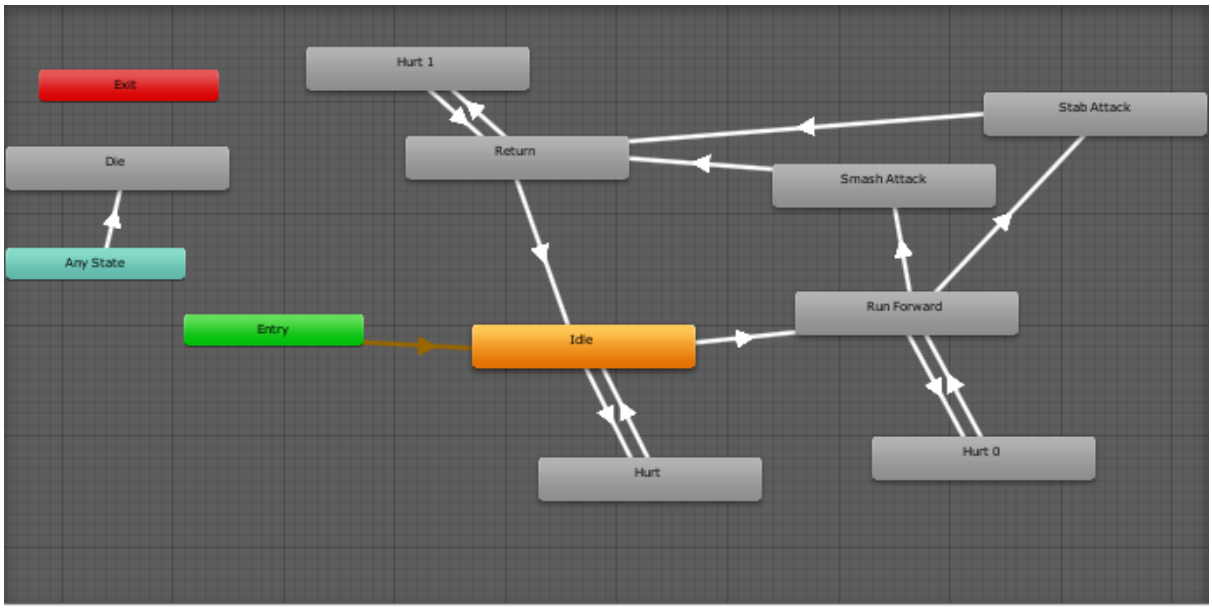

Ilustración 39 Diagrama de estados del Boss

- **Vida**: El sistema de vida es totalmente dependiente de la audición, evitando usar interfaces visuales que haya que traducir de alguna manera al usuario invidente cuando su vida empieza a bajar. Para ello se desarrollará un contador dentro de la entidad del jugador que lleve la cantidad de vida que tiene, de tal manera que, cuando se acerca demasiado a un enemigo y este le ataca, el contador baje hasta que llegue a cero y termine el juego.
  La manera de hacer que este sistema estándar de vida dependa del oído y no de la vista es a través de un ritmo cardíaco. En función de la cantidad de vida actual, el ritmo cardíaco (unos sonidos estándar de latidos de corazón) se irán acelerando o ralentizando en función de si se pierde o gana vida, respectivamente. De esta manera, si se está perdiendo vida, se notará en el aumento del periodo con el que suenan los latidos y, si se está en un nivel de salud crítica, el ritmo será muy acelerado.

- **Puertas**: Las puertas delimitan las salas y abren camino a distintas partes del mapa. Las puertas se abrirán por cercanía del jugador a ellas, teniendo que calcular constantemente la distancia de estas al jugador. No se abrirán si hay enemigos en la sala. Comprueban la existencia de entidades enlazadas (enemigos en este caso) y, en el caso de que exista alguna entidad la puerta no se abre al pasar junto a ella y, en el caso de que no exista, se abrirán emitiendo un sonido de escotilla al acercarse el usuario y volverán a cerrarse al alejarse.

Ilustración 40 Aspecto de las puertas en el juego

4.2.3. Tercera fase

En esta fase el equipo se ha centrado en pulir las mecánicas, arreglar bugs, y seguir implementando mecánicas y contenido accesibles, que mejoren la experiencia del juego:

- **Chocando con las paredes**: El laberinto ya es un reto de por sí para personas con plenas capacidades visuales, pero es un reto aún mayor para personas con discapacidad visual. Hablando con algunos potenciales usuarios finales nos comentaron que el método que usan para recorrer el laberinto era ir dibujándolo en su mente y por ende debíamos darle algún medio para poder detectar las paredes y que ellos mismos pudieran formarse ese dibujo.

  Finalmente llegamos a la conclusión de que la mejor forma de transmitirles el laberinto era a través de la vibración del mando.

  Añadimos una etiqueta "Wall" a las paredes. De esta forma, si se detectaba una colisión en el script del jugador con un objeto que tuviera esta etiqueta se generaba un sonido y se hacía vibrar el mando.
  Unity tiene muchas herramientas, pero está poco enfocado en los periféricos por lo que a través de las herramientas estándar de Unity no podíamos hacer que el mando vibre cada vez que el jugador estaba cerca de una pared.
  Para solucionar esto, usamos una solución de código abierto que nos ofrece una solución fácil para tener mayor control de los periféricos a través de Unity. "**XinputDotNet"** (Gillig, Hymers, & Wang, 2009) es una librería de código abierto que nos permite manejar los mandos conectados al PC, en nuestro caso el Mando de XBOX 360.

  Una vez hecho esto en el script del movimiento del jugador añadiremos una función "*OnCollisionStay*" que se ejecutará de manera cíclica mientras exista una colisión, por lo que el jugador cada vez que se acerque a una pared notará la vibración del mando y podrá detectar tan solo con sus manos la ubicación de la pared relativa al personaje y a su vez empezar a dibujar un mapa mental.

- **Sonido del arma de cuerpo a cuerpo**: Una de las ideas que teníamos en torno a la implementación en realidad virtual era la opción de poder usar el arma, ya fuera cuerpo a cuerpo o a distancia, como objeto para poder generar un sonido si chocaba con un obstáculo. De esta forma, el arma serviría a forma de bastón blanco para el jugador, que podría moverlo a su alrededor para detectar si hay obstáculos o enemigos dado que generarían iconos auditivos cuando el arma chocase con ellos.
  Debido al imprevisto surgido con la pandemia y a la incapacidad de poder hacer una implementación tal en torno a los controles de la realidad virtual, decidimos hacer una versión alternativa de esta mecánica de tal manera que, si se ataca con el arma cuerpo a cuerpo, se emitirá un sonido distinto dependiendo del objeto golpeado.
  De esta manera, el jugador podrá discernir lo que hay a su alrededor antes de continuar.
  Está mecánica está principalmente orientada al laberinto, pues puede ser bastante complicado completarlo sin ningún tipo de pista auditiva y resulta excesivamente complicado implementar un sistema de guía auditiva por raíles debido a su generación procedural.

- **Menú**: Se ha implementado un menú sencillo con tres opciones: “Jugar”, que comienza la partida normal, “Tutorial” que tras una larga locución genera la escena “tutorial” explicada en el siguiente punto, y “Salir” que cierra el juego.

- **Tutorial**: Se trata de una escena que comienza con una locución extensa que explica las diferentes mecánicas y sonidos del juego, posteriormente carga una sala vacía y cuadrada, donde el jugador puede moverse libremente sin peligro, para que el usuario pueda probar los controles. Además, se le ofrece un input con el que poder generar enemigos, para que pueda probar a su ritmo el combate del juego.

- **Locuciones**: En esta fase se generaron las distintas locuciones para la navegación y explicación del juego, estas son:

    - **Locuciones de los botones del menú:** Describen el texto de los botones “Jugar”, “Tutorial” y “Salir”. Estas locuciones se escuchan cuando el botón está activo.

    - **Tutoriales**: Existen dos, el más corto es un pequeño tutorial que explica cómo navegar en el menú después de que detecte una inactividad del usuario durante 2 segundos aproximadamente. El otro es un tutorial extenso donde se explican todas las mecánicas del juego, así como los sonidos y el objetivo.

    - **Introducción al juego**: Locución con efectos para generar ambiente antes de empezar la partida, introduce al usuario en el lugar donde aparece y le explica lo que tiene alrededor, así como el objetivo del juego.

    - **Sonido fin del juego**: Existen dos sonidos para acabar el juego, uno de victoria, donde suena una música victoriosa, y uno de derrota, donde una voz dice “Game Over”.

4.2.4. Implementación

En este punto desarrollaremos las mecánicas descritas previamente, de manera más detallada y usando imágenes y código.

- **Laberintos**: La generación del laberinto se hace mediante un algoritmo el cual lo crea de manera aleatoria. Usa un modelo (celda), el cual tiene suelo y cuatro paredes (definidas como un enumerado: oeste, norte, este, y sur), las cuales se pueden activar o desactivar.

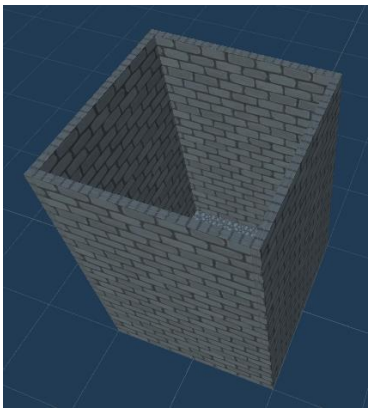

Ilustración 41. Vista previa de la celda.

```
32 referencias
public enum WallOrientation
{
    WEST = 0, NORTH = 1, EAST = 2, SOUTH = 3
}
24 referencias
public class Cell : MonoBehaviour
{
    public GameObject[] walls;

    public bool isVisited = false;

    private bool[] wallsActive = { true, true, true, true };

    18 referencias
    public void HideWall(WallOrientation orientation)
    {
        int index = (int)orientation;

        if(walls[index] != null)
        {
            Destroy(walls[index]);
            this.walls[index] = null;
            this.wallsActive[index] = false;
        }
    }

    0 referencias
    public bool IsWallActive(WallOrientation orientation)
    {
        int index = (int)orientation;
        return this.wallsActive[index];
    }
}
```

Ilustración 42. Código de la celda.

El algoritmo recibe por parámetros, la posición de inicio, el tamaño, la entrada, y la salida del laberinto.

Se crea un tablero, en el cual se introducen una celda en cada posición, el tablero tiene dos variables, *i* y *j* (coordenadas *x* e *y*). Empieza a crear el laberinto desde una posición aleatoria del tablero (dentro del tamaño del laberinto), desde esta posición, vamos avanzando por las casillas, sacando un número aleatorio entre cero y tres, el cual indica la dirección que tomaremos teniendo en cuenta el enumerado previamente explicado. Dependiendo del aleatorio anterior, se mira la siguiente celda y se comprueba que no haya sido visitada, si no se ha visitado se eliminan dos paredes, una de la celda actual y otra de la siguiente, por ejemplo, si se avanza hacia delante, se quita la pared norte de la actual y la pared sur de la siguiente, de tal manera que queden conectadas. Si ya ha sido visitada (la siguiente celda) no se realiza nada.

Para que el algoritmo entre en todas las celdas, se crea una pila con cada celda, en el *update* (método que se llama a la hora de generar el laberinto) cada vez que se modifica una celda, se extrae de la pila mediante un pop. Cuando se han recorrido todas las celdas (la pila está vacía), se pone a true un booleano que indica que la generación ha sido finalizada (generationFinished) y posteriormente se añade la entrada y la salida del laberinto, las cuales son valores fijados previamente (mediante los parámetros).

```
int posX = posXini, posZ;
for(int i = 0; i < this.width; i++)
{
    posZ = posZini;
    for (int j = 0; j < this.height; j++)
    {
        GameObject cellGO = Instantiate(this.cellPrefab, new Vector3(posX, 0, posZ), Quaternion.identity);
        this.cellGrid[i, j] = cellGO.GetComponent<Cell>();
        posZ += 2;
    }
    posX += 2;
    yield return null;
}
```

Ilustración 43. Código para la creación de las celdas en su posición.

```
case WallOrientation.WEST:
    if(current_x > 0)
    {
        Cell next = this.cellGrid[current_x - 1, current_y];

        if (!next.isVisited)
        {
            current.HideWall(WallOrientation.WEST);
            next.HideWall(WallOrientation.EAST);

            next.isVisited = true;
            this.stack.Push(next);
            valid = true;
        }
    }
    break;
```

Ilustración 44. Código para comprobar si se tienen que ocultar las paredes de la celda.

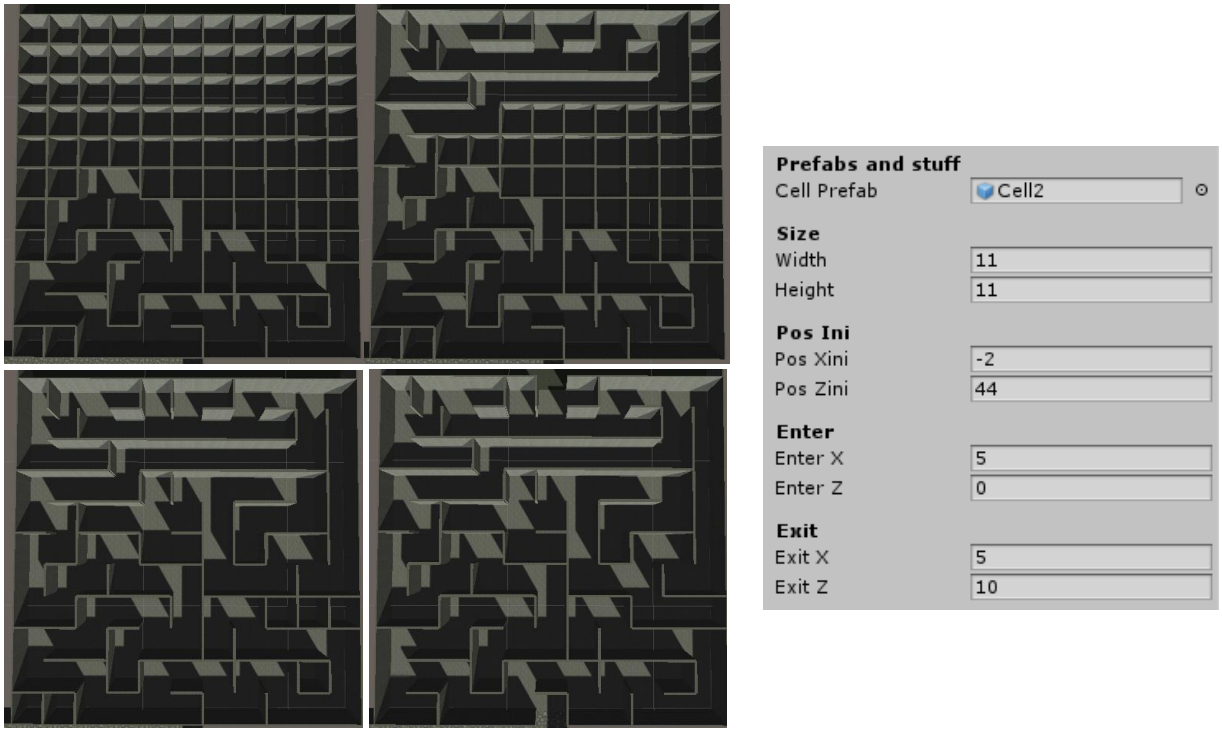

Ilustración 45. Transcurso de la creación del laberinto. A la derecha, los parámetros de entrada.

- **Comportamiento del enemigo** (EnemyBehaviour.cs): El comportamiento del enemigo es relativamente simple y se puede describir en una serie de comprobaciones y estados por cada *frame*.

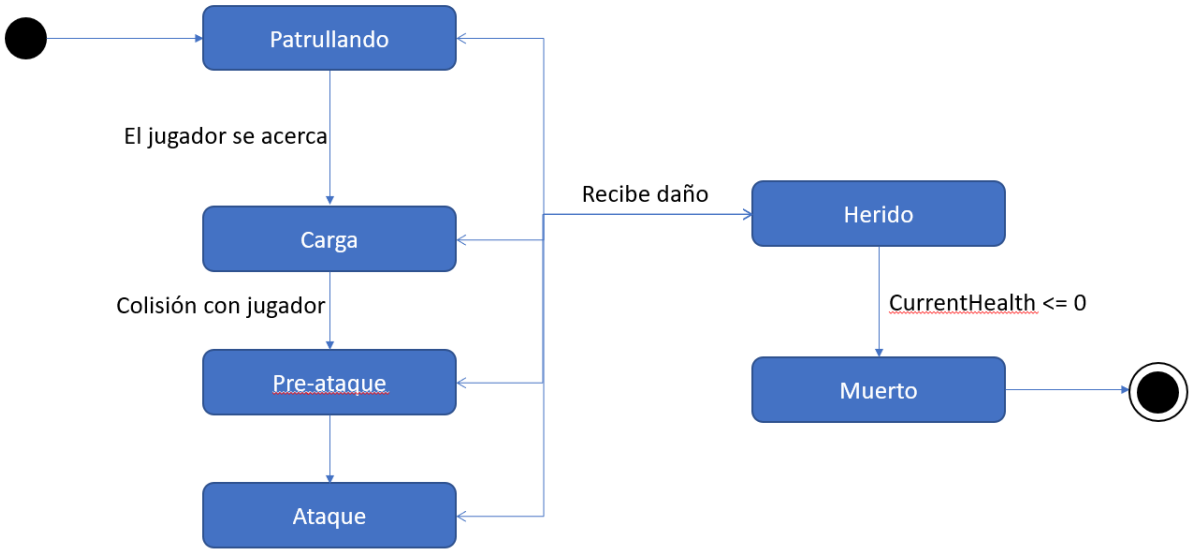

Ilustración 46 Diagrama de estados del enemigo.

Por cada *frame* el enemigo:

- Comprueba si está muerto con una variable booleana, que se pondrá a true en el método "*Dead*", que puede invocarse desde el script de control de la vida. Si está muerto, obviamente, deja de comprobar más cosas y no hace nada.
- Comprueba si el Jugador está a un rango de distancia:
  - Si el jugador está cerca, inmediatamente rota para enfrentarse a él, y comienza a avanzar. Cada *frame* además comprobará si ha sido dañado, para detenerse y realizar el sonido de enemigo dañado. Cuando colisione con la *hitbox* del jugador, se detiene.
  - Si está colisionando con el jugador (Player), pasado un tiempo hará un sonido de pre-ataque, y luego atacará, emitiendo daño hacia el jugador.
- Si el jugador está lejos, se mueve en una dirección aleatoria, hasta que choca con una pared o pasa un tiempo (nextDirectionTime)
- Si está muerto se autodestruye después de reproducir el sonido correspondiente.

- **Vida** (HealthScript.cs): El sistema de vida es relativamente simple. Contiene dos variables públicas configurables desde el editor, tanto para la vida máxima como para la vida actual. Este script se utiliza tanto para el jugador principal como para los enemigos. Se compone de varios puntos:

  - **Método ApplyDamage**: Método para restar vida a la vida actual, dependiendo del daño que se envíe como argumento. Este método se llama desde el *player* cuando este dispara, o desde el enemigo cuando ataca. También comprueba si la vida ha llegado a cero, y si es así:

    - Si muere el enemigo, llama al método "*Dead*" del enemigo, que se encargará de destruirlo.

    - Si muere el Boss, nos hemos pasado el juego, así que cambiaremos de escena al menú principal comunicando, haciendo uso de la PlayerPrefs, que hemos ganado, para que reproduzca el sonido correspondiente

    - Si muere el jugador haremos lo mismo que en caso anterior, pero pasando por PlayerPrefs el *string* que indica que la partida se ha perdido.

  - Por cada *frame* comprueba la cantidad de vida que queda, si es el jugador principal. Si es menor que la mitad, empieza a reproducirse el sonido de latidos, si es menor que un cuarto se acelera ese sonido al doble de velocidad.

4.2.5. Control de versiones

Para la realización del proyecto barajamos varias opciones para el control de versiones y para poder trabajar a la vez en el proyecto de manera remota, cada uno haciendo una parte de la implementación.

En un principio, empezamos usando Drive (“Drive,” n.d.) como sistema de control de versiones, pero nos llevaba a complicaciones porque no gestiona los cambios automáticamente, obligándonos a añadir los cambios a la última versión si el archivo divergía. Por este motivo, tratamos de investigar cómo utilizar GitHub (“GitHub,” n.d.) en Unity y como integrarlo al entorno de desarrollo.

Github

Decidimos utilizar GitHub porque era lo estándar, ya estábamos más o menos familiarizados con el entorno y nos aportaba facilidades gracias a las cuentas de la universidad.

Necesitábamos que GitHub guardara código, modelados, recursos sonoros, relaciones entre entidades y scripts, etc. Vimos que lo más correcto era usar un plugin que se acopla al entorno de desarrollo de Unity y se ocupara de la gestión de los repositorios locales y en remoto.

El problema de este plugin es que, al hacer *commit* de los cambios que había hecho una persona en local, si alguien actualizaba su repositorio local para obtener los cambios de la otra persona, surgían errores de divergencia de todo tipo.

Los errores que había con cada *update* nos obligaban a tener que revisar todas las entidades con cada cambio, asegurarnos de que no se había perdido nada y corregir aquellas relaciones que habían desaparecido, los scripts que se habían eliminado de las entidades y los valores de los atributos que se habían eliminado de los scripts.

Google Drive

Todas estas correcciones a cada cambio eran muy engorrosas y llevaban a la pérdida de mucho tiempo y, si no se revisaba todo bien, la pérdida de datos en los *commits* del resto de los compañeros. Por este motivo, el equipo decidimos no usar GitHub y seguir utilizando Drive.

Por otro lado, Drive funciona como una nube de archivos, pudiendo guardar las presentaciones e información además del código y hacer varias versiones de un mismo archivo. De esta forma, podíamos trabajar sobre un fichero comprimido que hiciera las veces de lo que en GitHub sería la rama de ‘*develop’* y usar otros ficheros como ramas alternativas para ir implementando las funciones asignada a cada uno y al terminar, añadirlo al fichero ‘*develop’*.

Además, Drive nos permitía ir guardando diferentes ficheros comprimidos con las versiones de cada iteración, para poder guardar una copia por si se corrompían los datos o los cambios no nos convencían y teníamos que volver a una versión previa.

5. Experimentación

El objetivo final de la fase de experimentación es poner a prueba el videojuego con un jugador con discapacidad visual, ver si realmente el proyecto es accesible, y ver los posibles fallos o mejoras con respecto a este tema.

En un principio la experimentación se iba a realizar gracias al equipo que nos donaría la universidad de realidad virtual y con voluntarios que pudiésemos conseguir a través de la ONCE. Debido a la pandemia del Covid-19, la experimentación planeada no fue posible debido al cierre de las universidades y el estado de alarma decretado el 14 de Marzo de 2020 (“BOE,” 2020)

La experimentación ha sido llevada a cabo por una persona ciega, Jose Luís Fernández Coya antiguo director del CTI de la ONCE.

5.1. Diferencia de estándares

En este punto comentaremos las cosas más interesantes que han surgido de la experimentación, que son las diferencias de concepto que podemos tener nosotros, como personas sin ceguera, sobre los videojuegos, frente a una persona con esta discapacidad.

Cuando uno mueve a su personaje en un entorno 3D el movimiento se hace de una manera continua, ya que el contexto nos permite esa libertad.
Por ejemplo, si controlamos a un personaje y queremos que éste se mueva hacia adelante, pulsamos la tecla indefinidamente, hasta que decidimos que pare. Esta decisión la hacemos usando coordinación mano-ojo. Nuestros ojos ven que hemos llegado al destino y le comunican a la mano que deje de pulsar el botón

En las conversaciones con el *tester* pudimos ver que no es así como piensa él en el movimiento. Lo piensa por medio de lo que llama “pulsaciones” es decir, pulsar y soltar una tecla. Con eso él espera que el personaje haga lo que tenga que hacer, en este caso moverse hacia adelante, y recibir un *feedback* sonoro de que realmente lo ha hecho.
Además, en este caso, si el usuario pulsa el botón/tecla de desplazarse a la derecha, el personaje hará exactamente eso, un desplazamiento, pero sin mover la cámara. Esta interacción parecía muy extraña para el *tester*, ya que él asumió que ese input giraría al personaje.

En este caso el error está en la concepción previa que nosotros tenemos de los juegos FPS del mercado actuales, en los que el estándar es: desplazamiento del personaje con el teclado, pero giro de la cámara con el ratón. En este caso hay que tener en cuenta que las personas ciegas no usan ratón, por lo que esta interacción es completamente errónea.

5.2. Pruebas con usuario objetivo

Previo a la prueba por parte del usuario final, se decidieron unos puntos importantes que debía tener el juego antes de ser probado, para asegurar que no habría bloqueos ni cosas externas que afectasen, estos fueron:

- **Integración con JAWS**: Este punto choca directamente con nuestro objetivo, ya que JAWS es una aplicación para hacer accesibles aplicaciones que no lo son. Mientras que nosotros pretendemos que nuestro juego sea 100% accesible "de caja". No obstante, el juego ha de ser completamente neutral a JAWS, esto quiere decir que por mucho que JAWS esté activo, no debe de chocar con el audio (sonidos y/o locuciones).
- **Controles**: En nuestro caso el juego es 100% jugable con un mando de consola convencional (probado con mando de XBOX y PS3), y también con el teclado y ratón. El usuario indicó que no estaba cómodo con el uso del mando.

Tras comprobar estos puntos el usuario procedió a probarlo.

Los conflictos encontrados se han clasificado dentro de varios tipos:

- **Leve**: No afecta a la jugabilidad. Fallos que no afectan a la jugabilidad, errores y bugs pequeños. Descritos en la Tabla 1
- **Medio**: Conflictos que provoquen que el juego no sea una experiencia positiva. Por ejemplo, algo que provoque una dificultad excesiva. Descritos en la Tabla 2
- **Grave**: este tipo de conflictos impide que el juego sea jugable, rompe la experiencia de usuario y resulta frustrante. El juego no avanza. Descritos en la Tabla 3

Para cada conflicto, buscaremos la razón que lo provoca, y propondremos una solución.

Conflictos leves

Conflicto	Razón	Solución
No hay sonido de bienvenida al juego cuando se ejecuta	El menú reproduce las instrucciones si el jugador está inactivo durante un tiempo limitado	Ponemos una nueva locución para indicar que el juego está ejecutándose.
Las locuciones colapsan en el menú si cambian rápido las opciones.	El controlador del audio no controla si hay otro audio reproduciéndose antes de reproducir el nuevo.	Controlamos por código si hay otro audio reproduciéndose, y si lo hay lo paramos para reproducir el nuevo
Conflicto con JAWS en los controles.	Los controles de movimiento del personaje con el teclado son las teclas 'W', 'A','S' y 'D'. Esto provoca que cada cierto tiempo JAWS vocalice las teclas pulsadas.	Cambiamos los inputs para que el personaje se mueva con las flechas del teclado en vez de con las anteriores teclas.

Tabla 1 Conflictos leves.

Conflictos medios

Conflicto	Razón	Solución
El usuario no sabe dónde está la puerta, ni si se ha abierto.	La puerta no emite sonido, solo lo emite cuando se abre, y al tener estética futurista hace un sonido acorde. No obstante, ese sonido no forma parte del "estándar" de sonidos de puerta, por lo que no es reconocible para el usuario.	Cambiamos el sonido de la puerta por otro más parecido al estándar, pese a la pérdida "estética".
El sonido de los pasos está "retrasado".	El sonido de los pasos se produce cuando el personaje se ha movido durante X tiempo, y sigue en ello. Por la forma de jugar del usuario (pulsaciones) este sonido nunca se produce, por lo que no recibe *feedback* si se mueve.	Cambiamos el script de audio de los pasos y quitamos ese *delay* principal. De esta manera en cuanto se pulse la tecla se escuchará un paso.

Tabla 2 Conflictos Medios.

Conflictos Graves

Conflicto	Razón	Solución
El laberinto es difícil, y no hay manera de orientarse.	La interfaz de teclado no tiene el *feedback* de vibración, por lo que no hay manera de saber si has dejado de tocar una pared o no.	Este punto no va a ser corregido, con un mando funciona correctamente, y en VR funcionaría incluso mejor, ya que el usuario podría usar ambos mandos para "tocar" las paredes.

Tabla 3 Conflictos graves.

6. Conclusiones y trabajo futuro

6.1. Conclusiones

Como ya se ha mencionado previamente, existen pocos juegos que hayan sido adaptados para personas con problemas de visión, pero existen métodos para conseguir adaptarlos a este público. Es cierto que implica un gasto adicional en el desarrollo de un videojuego, pero aumenta las opciones de ocio de las personas con algún tipo de discapacidad visual.

Las mecánicas básicas de un juego convencional no son las de un juego accesible por norma general, ya que los estándares de desarrollo convencionales no son accesibles para personas con discapacidad visual.

Un ejemplo de esta falta de accesibilidad es que una persona vidente, para atravesar un pasillo virtual, se mueve pulsando la tecla “w” o la flecha direccional “arriba” hasta que llega a la pared contraria. Sin embargo, una persona ciega avanza poco a poco, revisa el entorno que le rodea usando las flechas direccionales con cortas pulsaciones, que le permitan medir la distancia recorrida de manera estimada. Generalmente, la persona ciega trata de acercarse a una de las paredes de la sala y formarse una idea de esta en la cabeza gracias al perímetro que forma la sala. Aunque este es sólo un ejemplo, ocurre con la mayoría de los controles y de la jugabilidad.

Una persona vidente puede ver que objetos hay en una sala con tan solo mirar el entorno al entrar. Una persona con alguna deficiencia visual necesita estímulos alternativo que le indiquen lo que hay en la sala, como por ejemplo el sonido los Non-Playable Characters (NPC’s) que indique que están presentes en esa sala. Como ejemplo, pongamos a MegaTgarrett, una persona ciega, usuario de youtube, que ha completado el Zelda: Ocarina of Time (Garret, 2011) y actualmente está jugando al Zelda: Majora’s Mask. Este jugador se hace una idea del mapa golpeando las paredes, ya que, al rebotar la espada contra estas, emite un sonido único. De esta manera es capaz de saber cómo es la sala y poder ir avanzando.

A la hora de desarrollar un videojuego para personas con algún problema de visión hay que tener en cuenta una serie de cosas. No es posible, o al menos recomendable, hacer el desarrollo de un videojuego estándar y que posteriormente se trate de adaptar. Desde nuestra experiencia, es mucho más eficiente empezar a desarrollar el videojuego teniendo en cuenta las mecánicas para hacer que sea accesible.

Es importante que el desarrollo se haga pensando en la accesibilidad desde el primer momento, pues nos permite diseñar e implementar funcionalidades que, si nos limitáramos a adaptar desde la jugabilidad convencional, no seríamos capaces de desarrollar ya que no tendrían sentido para un jugador vidente. Además, hacer el ejercicio de empatizar con el público objetivo durante el diseño de cada funcionalidad puede evitar situaciones confusas para el jugador debido a que dicha funcionalidad no sea tan descriptiva como pensábamos cuando la diseñamos para un juego convencional y la adaptación no termine de ser optima.

Un ejemplo de esta situación es algo tan insignificante como el sonido que se elija para un objeto del juego. Es probable que el usuario no relacione el sonido que emita un objeto por sí mismo o al ocurrir un evento porque no haya visto jamás dicha relación, bien porque sea un sonido poco común o porque pertenezca al espectro de la ficción. De esta manera, si basamos el juego en una ambientación de ciencia ficción, al usuario invidente le costará entender lo que sucede. Habrá numerosas conexiones de sonidos con eventos que no entienda porque son completamente nuevos para él y solamente percibe el sonido, sin mayor contexto ni información.

Además, hemos podido observar que los sonidos que hagan representación de objeto o eventos, por muy descriptivos que sean, no son suficiente para dar el contexto mínimo al jugador para entender cómo debe jugarse y qué debe hacerse. Este contexto mínimo tiene que ser transmitido al jugador a través de una introducción que tiene que hacerse a través de TTS o de notas de voz pregrabadas. Una vez se hayan dado las instrucciones más fundamentales, sí se puede delegar en los iconos acústicos.

Dentro de esta introducción previa, se necesita explicar:

- Los controles del juego y su posición en el periférico que va a utilizar el usuario, ya que no tiene por qué conocerlo previamente y tendrá que pasar por una fase de explorar el mando y los botones que este contiene. Sería recomendarle localizar cada botón que se va a utilizar en las propias instrucciones, para que el jugador no se pierda buscando en el periférico.
- Qué cosas se va a encontrar a lo largo del juego (enemigos, peligros, eventos…), sobre todo si las relaciones de las que hemos hablado previamente no son inmediatas para el sonido y el evento. Es conveniente que la introducción a cada elemento cuyo sonido no sea evidente o no tenga una relación en la vida real venga provisto de una demostración del sonido. Por ejemplo, si se cuenta al jugador que hay enemigos, es conveniente mostrarle que sonidos pueden hacer los enemigos en cada uno de sus estados.
- En el caso de haber menús, se debe explicar cómo funcionan, qué contienen y cómo se navega por ellos dado que son elementos que tendrán que estar en la cabeza del jugador a lo largo del juego y no puede ir descubriéndolos él solo a través de iconos auditivos. Además, a menos que los menús sean lo suficientemente simples como para representar su contenido a través de iconos auditivos, necesitarán que se utilice del TTS para que el usuario pueda saber cuál es la opción actual.

Respecto a los controles, es importante simplificar, cuantos menos botones y más movimientos naturales necesite el jugador para controlar el juego, menos controles deberá tener constantemente mapeados en su mente, permitiendo al usuario centrar sus esfuerzos en construir mentalmente el mapa del entorno y en recordar los iconos auditivos, y no tanto en ubicar los controles en el periférico. Los juegos que nos mostraron en la ONCE eran juegos que se controlaban deslizando el dedo por la pantalla de un móvil en una dirección porque era más intuitivo controlar así los juegos que a través de una combinación artificiosa de botones.

6.2. Trabajo futuro

Debido a la situación actual del Covid-19 tuvimos que cambiar la idea inicial del proyecto. Nuestra idea era implementar el videojuego para realidad virtual, pero esta parte nos ha sido imposible ya que el material del que íbamos a disponer, así como el espacio para probarlo y usarlo eran de la universidad y, con el cierre de esta, nos quedamos sin poder acceder a estos recursos.

Uno de los objetivos a futuro sería modificar la implementación relativa al control del personaje para poder cambiar el controlador actual por uno de los controladores disponibles para realidad virtual. Esto, como ya hemos mencionado con anterioridad, podría facilitar la orientación del jugador, ya que el movimiento del personaje y sobre todo de la cámara es más acorde a la realidad, por lo que el jugador podrá comprender mejor esta mecánica y será más sencillo para él orientarse dentro del juego.

Además, por este problema con la disponibilidad del material, algunas mecánicas que mejoran la adaptabilidad se fueron dejando en la recámara para poder adaptarnos a la nueva realidad del proyecto.

6.2.1. Mecánicas

Sería interesante implementar en el futuro las siguientes mecánicas haciendo uso de la RV:

- Uso de los mandos de realidad virtual a modo de bastón blanco. El jugador siempre tiene un arma en la mano, y planteamos la posibilidad de usarla a modo de bastón para orientarse, de tal manera que el arma al chocar con una pared o con cualquier otro elemento emitiese un sonido e hiciese vibrar el mando.
- Uso del mando como guía. Otra de las mecánicas que propusimos fue la de “apuntar” con el mando y las gafas hacia un punto, y que el juego informase vía audio o voz de lo que existía en esa dirección, o incluso a qué distancia está del siguiente objeto. Tanto este punto como el anterior serían extremadamente útiles en la solución del laberinto.
- Sistema de migas de pan: Nos habría gustado darle alguna herramienta al jugador para dejar en el momento que decidiese lo que llamamos una “miga de pan”. En nuestro caso pensamos en unas campanas, que sonasen al pasar cerca, de tal manera que cuando las escuchase sonar el jugador supiese que ya ha pasado por ahí. Las migas de pan ayudarían a disfrutar más del juego, ya que el jugador tendría más herramientas para orientarse y solucionar con éxito la parte del laberinto.

6.2.2. Ampliación de contenidos

Además de las mecánicas previamente mencionadas, podría expandirse el juego añadiendo contenido en las siguientes direcciones:

- Más niveles: Añadir un sistema de pisos que vaya aumentando la dificultad tanto de los enemigos como de los laberintos a cada piso que se avance.
- Nuevas clases: Para variar la experiencia de juego, podrían añadirse dos clases de personaje más, un mago que lance hechizos varios, y un arquero que dispare con un arco, valiéndose de la posición de los mandos.
- Mejorar el ataque: Con el uso de la RV se podría dar profundidad al ataque con el hacha, valiéndose del movimiento de los mandos y no de la pulsación de un botón.
- Nuevas armas: Obtenibles al final de un nivel. Como podrían ser una lanza con más alcance para el guerrero, otros hechizos para el mago o una ballesta para el arquero. Se manejarían de manera diferente a las armas básicas y tendrían diferentes estadísticas.
- Puntuación: Que fuera sumando en función de la velocidad a la que se termina cada nivel.
- Historia: Es importante añadir un hilo conductor dentro del juego, que cuente al jugador como se juega a la par que le cuenta una historia.
- Cooperativo: Para fomentar la integración sería buena idea permitir que varios jugadores completen la mazmorra ayudándose los unos a los otros.

7. Conclusions and future work

7.1. Conclusions

As mentioned previously, there are few games that have been adapted for the visually impaired, but there are methods of making them suitable for this audience. It is true that this implies an additional expense in the development of a video game, but it increases the leisure options for people with some kind of visual disability.

The basic mechanics of a conventional game are not those of an accessible game as a rule since conventional development standards are not accessible to people with visual impairment.

An example of this lack of accessibility is that a sighted person, to cross a virtual corridor, moves by pressing the "w" key or the "up" directional arrow until he or she reaches the opposite wall. However, a person who is blind gradually moves forward, checking the environment around him using the directional arrows with short presses, which allow him to measure the distance traveled in an estimated way. Usually, the blind person tries to approach one of the walls of the room and get an idea of it in his head thanks to the perimeter that the room forms. Although this is only one example, it occurs with most controls and gameplay.

A sighted person can see what objects are in a room just by looking at the surroundings when entering. A person with a visual impairment needs alternative stimuli to indicate what is in the room, such as the sound of Non-Playable Characters (NPC's) to indicate that they are present in that room. As an example, let's take MegaTgarrett, a blind person, user of youtube, who has completed Zelda: Ocarina of Time (Garret, 2011) and is currently playing Zelda: Majora's Mask. This player gets an idea of the map by hitting the walls, since, when the sword bounces off them, it emits a unique sound. In this way he can know what the room looks like and be able to move forward.

When developing a video game for people with a vision problem, there are several things to consider. It is not possible, or at least recommended, to develop a standard video game and then try to adapt it. From our experience, it is much more efficient to start developing the video game considering the mechanics to make it accessible.

It is important that the development is done thinking about accessibility from the first moment, because it allows us to design and implement functionalities that, if we were limited to adapt from the conventional gameplay, we would not be able to develop since they would not make sense for a sighted player. In addition, making the exercise of empathizing with the target audience during the design of each feature can avoid confusing situations for the player because the feature is not as descriptive as we thought when we designed it for a conventional game and the adaptation is not optimal.

An example of this situation is something as insignificant as the sound you choose for an object in the game. It is likely that the user does not relate the sound that an object makes by itself or when an event occurs because he or she has never seen such a relationship, either because it is an unusual sound or because it belongs to the spectrum of fiction. Thus, if we base the game

on a science fiction setting, the blind user will have a hard time understanding what is going on. There will be numerous sound connections to events that he does not understand because they are completely new to him and he only perceives the sound, without any further context or information.

In addition, we have observed that sounds that make representations of objects or events, however descriptive they may be, are not sufficient to give the player the minimum context to understand how it should be played and what should be done. This minimum context must be transmitted to the player through an introduction that has to be done through TTS or pre-recorded voice notes. Once the most fundamental instructions have been given, it can be delegated to the acoustic icons.

Within this preliminary introduction, an explanation is needed:

- The controls of the game and their position in the peripheral that the user will use, since he does not have to know it previously and will have to go through a phase of exploring the controller and the buttons that it contains. It would be advisable to locate each button to be used in the instructions themselves, so that the player doesn't get lost searching in the peripheral.
- What things will be found throughout the game (enemies, dangers, events...), especially if the relationships we have talked about previously aren't immediate for the sound and the event. It is convenient that the introduction to each element whose sound is not evident or does not have a relationship in real life comes with a demonstration of the sound. For example, if the player is told that there are enemies, it is convenient to show him what sounds the enemies can make in each of their states.
- In the case of menus, you must explain how they work, what they contain and how to navigate through them, since these are elements that will have to be in the player's head throughout the game and he cannot discover them by himself through audio icons. Also, unless the menus are simple enough to represent their content through audio icons, they will need to be used from the TTS so that the user can know what the current option is.

Regarding the controls, it is important to simplify, the fewer buttons and more natural movements the player needs to control the game, the fewer controls he will need to have constantly mapped in his mind, allowing the user to focus his efforts on mentally mapping the environment and remembering the audio icons, and not so much on locating the controls on the peripheral. The games we were shown at ONCE were games that were controlled by sliding your finger across a mobile screen in one direction because it was more intuitive to control the games that way than through an artificial combination of buttons.

7.2. Future work

Due to the current situation of the Covid-19 we had to change the initial idea of the project. Our idea was to implement the video game for virtual reality, but this part has been impossible for us since the material we were going to have, as well as the space to test and use it were from the university and, with the closing of this one, we were left without being able to access these resources.

One of the future objectives would be to modify the implementation related to the control of the character to change the current controller for one of the controllers available for virtual reality. This, as we have mentioned before, could facilitate the orientation of the player, since the movement of the character and especially of the camera is more in line with reality, so the player will be able to better understand this mechanic and it will be easier for him to orient himself within the game.

Also, because of this problem with the availability of the material, some mechanics that improve the adaptability were left in the bedroom to be able to adapt to the new reality of the project.

7.2.1. Mechanics

It would be interesting to implement the following mechanics using VR in the future:

- Using the VR controls as a white cane. The player always has a weapon in his hand, and we proposed the possibility of using it as a stick for orientation, so that when the weapon hits a wall or any other element, it makes a sound and vibrates the controller.
- Use of the controller as a guide. Another mechanism we proposed was to "point" the controller and the glasses towards a point, and the game would inform via audio or voice what was in that direction, or even how far it is from the next object. Both this point and the previous one would be extremely useful in the solution of the maze.
- Breadcrumb system: We would have liked to give some tools to the player to leave at the moment he decides what we call a "breadcrumb". In our case we thought about some bells, which would ring when passing by, so that when he heard them ringing the player would know that he had already passed by. The breadcrumbs would help to enjoy the game more, since the player would have more tools to orientate himself and solve the maze part successfully.

7.2.2. Expansion of contents

In addition to the previously mentioned mechanics, the game could be expanded by adding content in the following directions:

- More levels: Adding a floor system that increases the difficulty of both enemies and mazes to each floor you advance.
- New classes: To vary the game experience, two new character classes could be added, a mage who casts various spells, and an archer who shoots with a bow, using the position of the controls.
- Improved Attack: Using VR, the attack can be deepened with the axe, using the movement of the controllers rather than the push of a button.
- New Weapons: Available at the end of a level. These could be a spear with more range for the warrior, other spells for the mage, or a crossbow for the archer. They would be handled differently than the basic weapons and would have different statistics.
- Score: Adding up according to the speed at which each level is completed.
- History: It is important to add a thread within the game, which tells the player how it is played at the same time as it tells a story.
- Cooperative: To encourage integration it would be a good idea to allow several players to complete the dungeon by helping each other.

8. Trabajo individual

8.1. Adrián Menéndez

Fase de investigación

Durante la fase inicial, pertenecí al grupo que se ocupaba de ir buscando información sobre la accesibilidad en los videojuegos, centrándome más en las técnicas, los estándares y los ejemplos de desarrollo de juego completamente accesibles.

Durante la búsqueda de información, encontré lo principales *papers* en los que nos apoyaríamos para el desarrollo de la aplicación y la creación de las funcionalidades de accesibilidad del proyecto. Además, esta investigación se recopiló en una pequeña presentación y un documento de texto que serviría como esqueleto para el punto de estado del arte de la memoria final (punto 3).

Fases de implementación

Durante la primera fase de implementación, estuve familiarizándome con el nuevo entorno gracias al prototipo ya desarrollado y me ocupé del desarrollo de la interacción con objetos del entorno y en habilitar el ataque cuerpo a cuerpo.

Para la tarea de la interacción, tuve que guardar las entidades de la escena con las que se podía interactuar en una lista, para poder manejarlas de manera continua con respecto a la posición del jugador. Gracias a una de las funcionalidades de Unity, podía obtener la posición en la escena de las entidades del jugador y de los objetos con los que se podía interactuar y así calcular la distancia entre ambos.

Además, implementé un método para que el jugador dejara caer el arma que tenía actualmente equipada cuando el objeto con el que interactuaba fuese un arma, creando una nueva entidad que equivalía al arma que tenía equipada en ese momento. De esta manera se podría recuperar el arma que tenía previamente equipada si se prefería.

Para el ataque cuerpo a cuerpo, tuve que hacer una implementación distinta a la de los disparos del arma de fuego, que fuera más acorde a la animación del ataque y esperara a que esta terminara. Además, limité el rango con el que alcanzaba a los enemigos.

Por último, lo más complicado de estas implementaciones fue que el modelado que cogimos para el hacha estaba hecho con el brazo del jugador (cosa que también le pasaba al arma de fuego). Por este motivo, fue necesario que remodelase ambas armas a través de “Blender” para quitar los componentes del personaje de ambos modelados y así poder mostrarlos en la escena, como objetos sueltos en el suelo.

Antes de continuar en las siguientes implementaciones, asistí junto a mis compañeros a una reunión con una persona que orientó su master al desarrollo del sonido envolvente para que nos informara un poco de qué podíamos hacer para dar una sensación de audio más realista y orientar mejor al jugador a través del audio.

En las implementaciones posteriores, me ocupé de hacer las funcionalidades de generar sonido al chocar el jugador con la pared o al pegar con el arma en una. Ambas implementaciones eran posibles gracias a las *tags* de Unity con las que se podía diferenciar de qué tipo era cada entidad de la escena.

Por un lado, el choque del jugador necesitó que utilizara un evento específico de las entidades que detecta cuando una entidad entra en el área de otra, de esta manera, si la entidad que recogía el evento era de tipo *Wall*, se reproducía un sonido específico. El sonido del arma fue más simple, detectando el objeto que había inmediatamente delante a la hora de atacar.

Memoria

En la redacción de la memoria, me he ocupado del punto del estado del arte (punto 3) junto a José María, del subpunto de la fase de investigación (punto 4.1), de la información de las implementaciones de las que yo me he ocupado y del control de versiones en el subpunto de fase de implementación (punto 4.2) y de las traducciones de los puntos de introducción (punto 2) y conclusiones (punto 7)

8.2. Jose Maria Gómez-Trabadela García

Fase de investigación

Durante la fase inicial, pertenecí al grupo que se ocupaba de ir buscando información sobre la accesibilidad en los videojuegos. La parte en la que me centré fue en la correcta definición de la realidad virtual, en sus distintos tipos y en los periféricos que se usan para la captura del movimiento.

Como esta definición está basada en términos de hardware por lo que vi necesario definirlo en otros términos más naturales e investigar el término de presencia y telepresencia ya que estos términos no se basan en el hardware, sino que ayudan a definir la experiencia. Esta experiencia es lo que nosotros buscamos investigar y simular para poder ponernos a crear un juego adaptado. (punto 3.2 y punto 3.3)

Antes de la fase de implementación participé en la reunión grupal del Centro de Tiflotecnología e Innovación (CTI) de la Organización Nacional de Ciegos Españoles (ONCE) en el que planteamos preguntas para recabar más información de cómo realizar el tipo de proyecto que teníamos entre manos.

Fases de implementación

Basándome en el prototipo que habían realizado mis compañeros en la primera fase de la implementación desarrollé la base del arma de cuerpo a cuerpo. Esta arma era un hacha de leñador cuyas texturas y modelado provienen de open source. Las partes que implementé fueron el control de las animaciones y el control de la propia hacha.

Además de esto implementé un inventario de armas. Este inventario es un vector que guarda todas las armas recogidas y mediante teclado permite seleccionar el arma que queremos usar.

Por último, en la primera fase añadí el control del mando de Xbox 360 mapeando los botones y el jostick de este a distintos inputs que tiene Unity. Al hacerlo a través de los inputs de Unity no hay conflictos entre el teclado y el mando pudiendo pasar de uno a otro. Además de esto a la hora de programar las respuestas a distintos inputs no hay que diferenciar entre teclado y mando por lo que el código es más limpio.

En la segunda fase de implementación me centré en la implementación del Boss o jefe final. Para el boss se usó unas texturas y un modelo open source. La implementación fueron el control de animaciones, el patrón de movimiento y los sonidos.

El patrón de movimiento y las animaciones van de la mano ya que cada parte de ese patrón requerirá que el boss realice una animación. Esta animación nos dará información visual de en qué punto de ese patón se encuentra.

El patrón de movimiento se divide en varias fases por las que va pasando de una a otra según unas variables que tiene el controlador. Cada fase tendrá una condición que hará que esa fase termine y pase a la siguiente.

El patón es el siguiente: El boss detecta al jugador en un área alrededor suyo, el boss corre hacia esa posición, realiza un golpe o una arremetida y vuelve a su posición inicial para esperar la próxima intrusión del jugador dentro del área. El patón de movimiento es sencillo para que con una o dos iteraciones con él sirvan para descubrirlo. Para una persona sin discapacidad es sencillo de ver, pero para una persona discapacitada es imposible. Por eso necesitamos transmitirle estos patrones a través de otro canal, el auditivo.

El sonido el boss nos indica qué está haciendo en cada momento. La fuente del sonido es la posición del boss y con la implementación del sonido en 3d podemos saber su posición únicamente con el sonido. Cada vez que el boss cambia de fase cambia el sonido que produce.

El patrón de sonido es el siguiente: Cuando no detecta al jugador hará gruñidos metálicos, cuando detecte al jugador oiremos los pasos del monstruo, durante el ataque dará un golpe según el ataque que realice y cuando se retire volveremos a oír los pasos del monstruo.

En la siguiente implementación me puse con la detección de las paredes a través del mando. Con el fin de que una persona pueda detectar las paredes, no solo a través del oído, si no que además a través del tacto. Por eso utilicé la vibración del mando y la detección de colisiones para ese fin.

Añadí un hitbox o una caja de colisión al personaje que controlas para que el personaje pudiera detectar colisiones. Puse en las cajas de colisiones de las paredes del laberinto una etiqueta llamada “wall” para que cuando el jugador, en su script de movimiento, detecte una colisión con esa etiqueta mande una señal al mando para que vibre. Para la implementación de esa vibración se usa una librería de código libre que nos permite controlar la vibración que de otro modo unity no te permite.

Memoria

En la redacción de la memoria, me he ocupado del punto del estado del arte (punto 3) junto a Adrián, del subpunto del Plan de Desarrollo de las Implementaciones (punto 4.2) proporcionando información general de las implementaciones y extendiendo esta información en el (punto 4.2.3) de las implementaciones de las que he participado personalmente.

8.3. Enrique Maestro Mañanes

Desarrollo UME

Durante la fase inicial estaba en el grupo que se dedicaba a realizar una UME (unidad mínima de experiencia) para una primera toma de contacto, tanto con el lenguaje usado para programar el videojuego como para ir probando la herramienta, en este caso unity.

La UME que decidimos hacer era un juego de estilo shooter, en el cual aparecieran enemigos y tuvieras que ir eliminándolos de la sala. Decidimos dividir el trabajo en dos partes, uno se encargaría de realizar los enemigos y su ataque y el otro se ocuparía del movimiento del personaje.
Yo realice la parte del personaje, tanto el movimiento como la cámara. Para empezar, desde la propia herramienta de Unity, creas un objeto vacío al cual le asignas un tag de jugador, esto es necesario para poder implementar los movimientos. Después de añadir los controles básicos para poder moverte (WASD, teclas para moverte en cada dirección) y el movimiento con el ratón para la cámara del jugador, implemente dos scripts, el primero permitía al jugador esprintar y agacharse y saltar, en el segundo añadí sonido de pisadas al jugador al andar.

Fases de implementación

Durante la primera fase de implementación, me encargue de realizar el diseño del mapa, tanto las salas donde estarían los enemigos, las salas del principio y final del mapa y los laberintos. También tuve que implementar los materiales que se usarían tanto en las paredes como en el suelo.

Lo interesante de esta parte fue la creación del laberinto, decidí hacer una implementación que fuera aleatoria cada vez que entras, de tal manera cada partida seria única y no sería repetitivo. El laberinto se crea a partir de un algoritmo en el que puedes introducir el tamaño de los lados del laberinto (X e Y, usados para crear laberintos rectangulares), posición tanto de la puerta de entrada como la salida del laberinto (para poder personalizar el laberinto dependiendo del mapa), y la posición donde quieres que se cree dentro del mapa (útil para poder cambiar la ubicación de manera sencilla e intuitiva).

Implemente una celda, la cual se compone de cuatro paredes y un suelo, esta celda la usa el algoritmo del laberinto para su creación. Las celdas permiten al alterar el laberinto de manera fácil sin necesidad de modificar el algoritmo, ya que puedes cambiar los modelos de las paredes y del suelo y su tamaño. Esto permite que se puedan hacer las paredes más altas o ampliar el tamaño de cada celda si es necesario.

Otra de las implementaciones que hice fueron las puertas del mapa, las cuales se abren cuando el usuario está cerca de ellas. Para que el jugador no pueda saltarse una sala sin haber eliminado previamente a los enemigos, hice uso de un contador, el cual cada vez que un enemigo muere se llama a una función que actualiza su valor. Cada puerta tiene un script con una entrada que indica el número de enemigos que tienen que ser eliminados para que se abra. Con estos dos valores puedes saber si la puerta se tiene que abrir o no.

Memoria

Por último, realice una primera versión del menú inicial, en el cual había dos botones, uno para empezar a jugar y otro para cerrar el juego. El primero, cargaba la escena del juego y creaba el personaje en la sala inicial.

En la redacción de la memoria, me he ocupado de los capítulos de introducción (punto 1) y conclusión y trabajo a futuro (punto 6).

8.4. Jorge Gómez Baraibar

Mi trabajo personal en el proyecto ha ido ligado, sobre todo, a la parte del desarrollo en Unity.

Desarrollo UME

Durante la primera fase del TFG, nos dividimos en dos grupos, y junto con Enrique nos encargamos de hacer una unidad mínima de experiencia (UME).

En esta parte yo me encargué de hacer un enemigo básico, que se moviese en dirección al jugador hasta colisionar con él. también desarrollé un sistema de creación de enemigos, que generaba un objeto en Unity del tipo enemigo cada X segundos, siendo X una variable pública editable desde el inspector.

Desarrollos posteriores

En los posteriores desarrollos e implementaciones me he encargado de varios puntos:

- Creación del enemigo y su comportamiento. Descrito con detalle en el punto 4
- Creación de un sistema de vida. Este sistema se utilizaría tanto en el jugador principal como en los enemigos, dotando de un final al juego, ya que el personaje principal podía morir. Para hacerlo accesible el script gestiona un audio de latidos de corazón, acelerándolo según la vida vaya bajando.
- Creación de un tutorial. He sido el encargado de crear una nueva escena, y del botón del menú principal que la haga accesible. Esta escena se compone de una sala cuadrada y vacía, donde el jugador aparece e inmediatamente después suena la locución del tutorial. El jugador es libre de moverse por la sala, y dispone de un botón que genera enemigos para practicar el combate.
- Locuciones: Me he encargado de la guionización y la grabación de todas las locuciones del juego, estas son:
    - Locución del tutorial
    - Locución de la introducción.
    - Entrada al juego
    - Botones del menú
    - Mensajes auxiliares.
- Creación del bucle del juego. Esto es hacer que el juego tenga un principio y un final, cuando el jugador muere o mata al jefe final, el juego tiene que producir un feedback al jugador, y volver al menú principal.
- Corrección de bugs y pulido del juego.

Experimentación

Fui el encargado de hablar con la persona que iba a probar nuestro juego (Jose Luis Fernández Coya, mencionado varias veces a lo largo del documento).

Con él realicé dos entrevistas, una previa a las pruebas y otra posterior. En la previa discutimos temas que debían estar en el juego obligatoriamente, descritas aquí.

Una vez probado el juego tuvimos otra entrevista donde me listó los problemas que había encontrado, y discutimos como solucionarlos.

Una vez acabado, me encargué de solucionar los mencionados en las tablas del punto 5

Memoria

En la memoria he sido el encargado de redactar los puntos 4 (a excepción de los redactados por Adrián) y el punto 5.

Bibliografía

Adams, E. (2010). *Fundamentals of Game Design*. Retrieved from http://ptgmedia.pearsoncmg.com/images/9780321929679/samplepages/0321929675.pdf

Adams, E. (2014). *Fundamentals of Game Desing*. Retrieved from https://books.google.es/books?id=Lm1jAgAAQBAJ&printsec=frontcover&dq=Adams,+Ernest%3B+Andrew+Rollings+(2006).+Fundamentals+of+Game+Design.+Prentice+Hall&redir_esc=y&hl=es#v=onepage&q&f=false

AGRIP. (n.d.). Retrieved from http://agrip.org.uk/

Alonso, Á. (2016). Hero Shooters: Overwatch, Battleborn, Paragon... Retrieved from https://www.hobbyconsolas.com/reportajes/hero-shooters-overwatch-battleborn-paragon-141388

Apperley, T. H. (2006). Genre and game studies: Toward a critical approach to video game genres. *Simulation & Gaming*, *37*(1), 6–23. https://doi.org/10.1177/1046878105282278

Archambault, D., Ossmann, R., Gaudy, T., & Miesenberger, K. (2007). Computer Games and Visually Impaired People. *Upgrade*, *VIII*(2), 43–53. Retrieved from http://www.upgrade-cepis.org/issues/2007/2/upgrade-vol-VIII-2.html

Arias, F. (n.d.). Diversión accesible. Retrieved from https://play.google.com/store/apps/details?id=com.fastudios.diversionaccesible&hl=es

Asociación D.O.C.E. (2015). CRITERIOS DE VALORACIÓN DEL GRADO DE DISCAPACIDAD VISUAL EN ESPAÑA. TEXTO COMPLETO Y GRÁFICOS. NORMAS PARA LA VALORACIÓN DE LA DEFICIENCIA VISUAL. Retrieved from https://asociaciondoce.com/2015/12/21/criterios-de-valoracion-del-grado-de-discapacidad-visual-en-espana-texto-completo-y-graficos-normas-para-la-valoracion-de-la-deficiencia-visual/comment-page-1/

Asociación Mácula Retina. (2019). ¿Qué es un escotoma? Retrieved from https://www.macula-retina.es/que-es-un-escotoma/

AudioGames Games List. (n.d.). Retrieved from https://audiogames.net/list-games/

Balan, O., Moldoveanu, A., & Moldoveanu, F. (2015). Navigational audio games: An effective approach toward improving spatial contextual learning for blind people. *International Journal on Disability and Human Development*, *14*(2), 109–118. https://doi.org/10.1515/ijdhd-2014-0018

Best of Torque. (n.d.). Retrieved from http://www.garagegames.com/best-of-torque/torque-3d

Bierre, K., Chetwynd, J., Ellis, B., Hinn, D. M., Ludi, S., & Westin, T. (2005). *Game Not Over : Accessibility Issues in Video Games*.

Biggs, B., Yusim, L., & Coppin, P. (2018). The Audio Game Laboratory: Building Maps from Games. *The International Conference on Auditory Display (ICAD 2018)*, 10–15. Retrieved from http://openresearch.ocadu.ca/id/eprint/2389/%0Ahttp://openresearch.ocadu.ca/id/eprint/2389/1/Biggs_Audio_2018.pdf

BOE. (2020). Retrieved from BOE website: https://www.boe.es/diario_boe/txt.php?id=BOE-A-2020-3692

Bouzo, O. (2009). “Doom Rails”, un shooter sobre raíles... literalmente. Retrieved from https://www.vidaextra.com/videos/doom-rails-un-shooter-sobre-railes-literalmente

Cabacas, T. (2012). 50 años de Spacewar!, el primer videojuego de ordenador. Retrieved from https://www.muycomputer.com/2012/02/13/50-anos-de-spacewar-el-primer-videojuego/

Ceguera y pérdida de la visión. (n.d.). Retrieved from https://medlineplus.gov/spanish/ency/article/003040.htm

Chen, J. (2007). Flow in games (and everything else). *Commun. ACM*, *50*, 31–34. https://doi.org/10.1145/1232743.1232769

Chion, M. (2012). The three listening modes. *The Sound Studies Reader*, 48–53.

Colley, S. (n.d.). Stories from the Maze War 30 Year Retrospective. Retrieved from http://www.digibarn.com/history/04-VCF7-MazeWar/stories/colley.html

Construct 3. (n.d.). Retrieved from https://editor.construct.net/

Dager System. (n.d.). Retrieved from https://dagersystem.com/

Definición de Shooter On Rails. (2020). Retrieved from http://www.gamerdic.es/termino/shooter-on-rails

Dingler, T., Lindsay, J., & Walker, B. N. (2008). Learnability of Sound Cues for Environmental Features: Auditory Icons, Earcons, Spearcons, and Speech. *14th International Conference on Auditory Display*, 1–6.

DOWiNO. (n.d.). Blind Legend. Retrieved from https://es.ulule.com/a-blind-legend/

Drag & Drop. (n.d.). Retrieved from https://neoattack.com/neowiki/drag-drop/

Drive. (n.d.). Retrieved from https://drive.google.com

Flowers, J. H., Turnage, K. D., & Buhman, D. C. (2005). Desktop Data Sonification: Comments on Flowers et al., ICAD 1996. *ACM Transactions on Applied Perception*, *2*(4), 473–476. https://doi.org/10.1145/1101530.1101545

Gaggioli, A., Bassi, M., & Fave, A. D. (2003). *Quality of Experience in Virtual Environments*. (May 2014).

Galvez Enrique. (2019). Que es el Sonido Envolvente (Surround Sound) y Como Conseguirlo. Retrieved from Que es el Sonido Envolvente (Surround Sound) y Como Conseguirlo

Game Maker Studio. (n.d.). Retrieved from https://www.yoyogames.com/

Game Making in Education. (n.d.). Retrieved from https://www.construct.net/en/make-games/education

Garage Games. (2012). *Torque3D*. Retrieved from https://github.com/GarageGames/Torque3D

García, A. (2018). Space Invaders: Los “marcianitos” cumplen 40 años. Retrieved from https://www.lavanguardia.com/tecnologia/20180725/451089770519/space-invaders-40-aniversario-marcianitos-arcade.html

Garret, T. (2011). Canal de Terry Garret: MegaTgarrett. Retrieved from https://www.youtube.com/user/MegaTgarrett

Gaudy, T., Natkin, S., & Archambault, D. (2006). Classification des jeux sonores selon leur type de jouabilit{é}. *Proceedings of Handicap 2006 Conference*, (June), 221–226.

Gaudy, T., Natkin, S., & Archambault, D. (2009). Pyvox 2: An audio game accessible to visually impaired people playable without visual nor verbal instructions. *Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, *5660 LNCS*(January 2009), 176–186. https://doi.org/10.1007/978-3-642-03270-7_12

Gaudy, T., Natkin, S., Prado, C. Le, Dilger, T., & Archambault, D. (2007). *TAMPOKME : A MULTI-USERS AUDIO GAME ACCESSIBLE TO VISUALLY AND MOTOR IMPAIRED PEOPLE*.

Gillig, R., Hymers, B., & Wang, F. (2009). *XInputDotNet*. Retrieved from https://github.com/speps/XInputDotNet

GitHub. (n.d.). Retrieved from https://github.com/

Guidelines Page. (2009). Retrieved from http://www.sidar.org/traducciones/wcag20/es/

Javi Andrés. (2020). El videojuego más accesible de la historia: The Last of Us: Parte II. Retrieved from https://www.once.es/blog/articulo/2020-06-17/videojuego-mas-accesible-de-historia-last-us-parte-ii

Johnson, D., & Wiles, J. (2003). Effective Affective User Interface Design in Games. *Ergonomics*, *46*, 1332–1345. https://doi.org/10.1080/00140130310001610865

Kionay. (2019). What is a Looter Shooter? Retrieved from https://medium.com/@kionay/what-is-a-looter-shooter-b168f1fdf9ac

Krygier, J. B. (1994). Sound and geographic visualization. In *Modern cartography series* (Vol. 2, pp. 149–166). Elsevier.

L-Works. (n.d.). Retrieved from https://www.l-works.net/seh.php

Lessard, N., Paré, M., Lepore, F., & Lassonde, M. (1998). Early-blind human subjects localize sound sources better than sighted subjects. *Nature*, *395*(6699), 278–280. https://doi.org/10.1038/26228

López, S. (2012). Análisis de Skullgirls. Retrieved from http://www.juegosdb.com/analisis-de-skullgirls-ps3-xbox-360/

Matas, F. G. (2019). Rainbow Six Siege alcanza los 50 millones de jugadores en todo el mundo. Retrieved from https://vandal.elespanol.com/noticia/1350726664/rainbow-six-siege-alcanza-los-50-millones-de-jugadores-en-todo-el-mundo/

Matatk. (2012). *agrip*. Retrieved from https://github.com/matatk/agrip/tree/initial

Merino, M. J. (2016). Daltonismo. Retrieved from https://www.clinicarementeria.es/patologias/daltonismo

Minigames make with Stencyl. (n.d.). Retrieved from https://www.newgrounds.com/collection/stencyl

Minsky, M. (1980). *Telepresence*.

Monkey Island 2: LeChuck's Revenge Complete Walkthrough. (n.d.). Retrieved from

https://steamcommunity.com/sharedfiles/filedetails/?id=285316139https://www.groupsteam.com/product/monkey-island-2-special-edition-lechucks-revenge/

Moore, B. (2014). No coding required: How new designers are using GameMaker to create indie smash hits. Retrieved from https://www.pcgamer.com/no-coding-required-how-new-designers-are-using-gamemaker-to-create-indie-smash-hits/

Natkin, S. (2006). *Video games and Interactive Media: A Glimpse at New Digital Entertainment*.

Olivetti, J. (2012). The Game Archaeologist: Maze War. Retrieved from https://www.engadget.com/2012-06-12-the-game-archaeologist-maze-war.html?guccounter=2

OMS. (2018). Ceguera y discapacidad visual. Retrieved from https://www.who.int/es/news-room/fact-sheets/detail/blindness-and-visual-impairment

Ortega, J. L. (2018). Ya puedes reservar el libro La guía de los matamarciano. Retrieved from https://www.hobbyconsolas.com/noticias/ya-puedes-reservar-libro-guia-matamarcianos-306893

Ossmann, R., & Miesenberger, K. (2006). Guidelines for the development of accessible computer games. *Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, *4061 LNCS*, 403–406. https://doi.org/10.1007/11788713_60

Pozuelo, G., & Álvarez, J. (n.d.). Zarodnik. Retrieved from http://es.blind-faith-games.e-ucm.es/zarodnik

Real Academia Española. (n.d.). Definicion de Videojuego. Retrieved from https://dle.rae.es/videojuego

Romagosa, I. (2017). Alteraciones de la visión cromática que impiden o limitan acceder a ciertas profesiones. Retrieved from https://www.admiravision.es/es/articulos/divulgacion/articulo/discromatopsia-o-daltonismo#.XrLfsJlS-Co

Sanchez, J. (2015). El resurgir de los Shooter Arcade, una realidad del todo virtual. Retrieved from https://livingplaystation.com/2015/05/06/el-resurgir-de-los-shooter-arcade-una-realidad-del-todo-virtual/

Scroller Menú. (n.d.). Retrieved from https://www.gametopia.es/learning/article/03/2019/60/diseno-ui-para-crear-videojuegos

Shooter game. (n.d.). Retrieved from https://en.wikipedia.org/wiki/Shooter_game

SimCarriere.com : jeu en ligne & online gratuit. Gérez une carrière d'avocat. (2009). Retrieved from http://cqcounter.com/site/simcarriere.be.html

Spacewar! (n.d.). Retrieved from http://1995-2015.undo.net/it/mostra/150826

Stencyl. (n.d.). Retrieved from http://www.stencyl.com/

Steuer, J. (1992). Defining Virtual Reality: Dimensions Determining Telepresence. *Journal of Communication*, *42*(4), 73–93. https://doi.org/10.1111/j.1460-2466.1992.tb00812.x

StickMUD. (n.d.). Retrieved from https://www.stickmud.com/index.php/Home

Straub, J. (2014). Skullgirls Encore Updated For Blind Accessibility. Retrieved from https://dagersystem.com/skullgirls-encore-updated-for-blind-accessibility/

Tactical shooter. (n.d.). Retrieved from https://en.wikipedia.org/wiki/Tactical_shooter

Tassi, P. (2019). There's Still Something Wrong With Legendary Farming In 'Borderlands 3.' Retrieved from https://www.forbes.com/sites/paultassi/2019/12/11/theres-still-something-wrong-with-legendary-farming-in-borderlands-3/#5a5d2a307e65

The Phantom. (n.d.). Retrieved from https://es.3dsystems.com/haptics-devices/3d-systems-phantom-premium

Tree Node. (n.d.). Retrieved from https://www.planetminecraft.com/project/logros-para-survival/

Unity 3d. (n.d.). Retrieved from https://unity3d.com/es/unity/beta/2019.3

Universidad de Alicante. (2008). Pautas de accesibilidad del contenido en la Web 2.0. Retrieved from http://accesibilidadweb.dlsi.ua.es/?menu=wcag-2.0

Unreal Engine. (n.d.). Retrieved from https://www.malavida.com/es/soft/unreal-development-kit/#gref

Velasco, J. (2011). Space Invaders, el matamarcianos que llegó del lejano oriente. Retrieved from https://hipertextual.com/2011/08/space-invaders-el-matamarcianos

Wawro, A. (2016). Hero Shooters: Charting the (re)birth of a genre. Retrieved from https://www.gamasutra.com/view/news/271933/Hero_Shooters_Charting_the_rebirth_of_a_genre.php

Willings, C. (2019). Vision Classifications. Retrieved from https://www.teachingvisuallyimpaired.com/vision-classifications.html

Witmer, B. G., & Singer, M. J. (1998). Measuring presence in virtual environments: A presence questionnaire. *Presence: Teleoperators and Virtual Environments*, *7*(3), 225–240. https://doi.org/10.1162/105474698565686

Zhao, H., Plaisant, C., Shneiderman, B., & Lazar, J. (2008). Data sonification for users with visual impairment: A case study with georeferenced data. *ACM Transactions on Computer-Human Interaction*, *15*(1). https://doi.org/10.1145/1352782.1352786