VIDEOJUEGO NARRATIVO PARA PERSONAS CIEGAS

A NARRATIVE VIDEOGAME FOR BLIND PEOPLE

Arturo Aguirre Calvo
Eduardo Andrés Morais
Alberto Casado Trapote
Fernando Cortés Sancho
Héctor Marcos Rabadán
Diego Martínez Simarro

Grado en Ingeniería Informática
Grado en Desarrollo de Videojuegos

Facultad de Informática
UNIVERSIDAD COMPLUTENSE DE MADRID

Trabajo de Fin de Grado

Directores:
María Guijarro Mata-García
Joaquín Recas Piorno

Agradecimientos

En primer lugar, los miembros de este TFG queremos dar las gracias a nuestros profesores directores María y Joaquín por proporcionarnos su orientación en un campo que nos era desconocido, por darnos su apoyo durante todo el desarrollo y en especial durante la situación tan complicada que atravesamos.

A la gente de la ONCE por compartir con nosotros conocimientos sobre accesibilidad, por permitirnos visitar su Centro de Tiflotecnología e Investigaciones y por probar el resultado de este trabajo en las difíciles circunstancias ocasionadas por el COVID-19. Su colaboración fue de inestimable ayuda.

A todos nuestros allegados, que nos dieron apoyo durante todo este proceso, con sus buenos y malos momentos, y a los que se ofrecieron a probar el resultado de nuestro trabajo. Sus comentarios fueron de gran utilidad.

A toda la comunidad de desarrolladores de videojuegos que se preocupan por la accesibilidad en sus productos e intentan mejorar el medio con su trabajo. Gracias a ellos cada vez más personas pueden disfrutarlos.

Índice

Índice de figuras

Lista de acrónimos

API	Interfaz de programación de aplicaciones
GDV	Grado en Desarrollo de Videojuegos
GII	Grado en Ingeniería Informática
HTML	HyperText Markup Language
JS	JavaScript
ONCE	Organización Nacional de Ciegos Españoles
PHP	Hypertext Preprocessor
SQL	Structured Query Language
STT	Speech To Text
TFG	Trabajo de Fin de Grado
TTS	Text To Speech

Palabras clave

Accesibilidad
Android
Discapacidad visual
Servidor web
Desarrollo web
Unity
Desarrollo de videojuegos

Resumen

En este trabajo se presenta un videojuego accesible para dispositivos **Android** desarrollado con el motor **Unity**. Este videojuego es accesible de forma nativa para personas ciegas o con visión reducida, imitando los gestos con los que se controla un revisor de pantalla. El juego toma elementos de aventuras gráficas y juegos basados en salas, con una narrativa ligera y con puzles en forma de minijuegos basados en el uso del oído y el tacto. Para esto, han sido necesarias investigaciones sobre la accesibilidad en general en el medio de los videojuegos y en específico en los teléfonos inteligentes, haciendo hincapié en la accesibilidad para discapacidades visuales y en las prácticas habituales en dispositivos **Android**.

Al juego lo acompaña un servidor en la nube en el cual se almacena el progreso del jugador y las estadísticas de juego, también se dispone de guardado local. Además, se cuenta con una página web como portal donde se puede descargar el juego y conocer a fondo cómo funciona, con conectividad con el servidor. Consecuentemente, la web también se ha realizado dando gran importancia a la accesibilidad, siguiendo las pautas marcadas en estándares. Esta web es mantenida en el servidor, cuyo montaje, configuración y mantenimiento se realizó sobre un ordenador propio.

Tanto el juego como la web fueron sujetos a múltiples pruebas a lo largo de la realización del trabajo, con el objetivo principal de evaluar su accesibilidad y facilidad de uso.

Keywords

Accessibility
Android
Visual disability
Web server
Web development
Unity
Game development

Abstract

This project consists in an accessible videogame developed for **Android** devices using the **Unity** engine. The videogame has been created from the ground up to be accessible to blind players and people with reduced vision alike. This is done by following the example of a screen reader in gestures and way of use. The game has elements of graphic adventures and room-based games, with a light narrative and puzzles in the shape of minigames based on the use of sound and touch. In order to develop the game, investigations about accessibility in general in the context of videogames and specifically for smartphones were necessary, with emphasis on accessibility measures for visual disabilities and on common practices on **Android** devices.

The game is complemented by a server on the cloud in which save data and game statistics are saved, the game also supports local saving. Furthermore, a webpage was developed as a portal where the game can be downloaded and where its mechanics are explained in detail. The webpage is also connected with the server. Consequently, the web was also developed focusing on accessibility, following guidelines set by standards. This web is maintained in the server, whose setup, configuration and maintenance were done on a member of the team own computer.

Both the game and web were subjected to thorough testing through the development cycle, being the evaluation of their accessibility and ease of use the main objective of said testing.

1. Introducción

Este Trabajo de Fin de Grado fue planteado como la realización de un videojuego accesible para personas con una pérdida de visión total o con restos de visión. El desarrollo de este videojuego sería acompañado por el de una infraestructura en internet, formada por una página web, también realizada bajo los parámetros de la accesibilidad, y un servidor propio, donde alojaríamos datos del videojuego, así como la propia web. Por esto, el tema de mayor importancia en este trabajo es la accesibilidad, siendo el principal objetivo que los resultados fruto de este desarrollo puedan ser usados por personas con discapacidad visual.

Desde un principio el desarrollo se planteó con los teléfonos móviles inteligentes como plataforma objetivo, al ser de uso habitual para toda la población y en particular para la gente que tiene esta discapacidad sensorial. Esto es en gran medida gracias a las opciones de accesibilidad de las que disponen estos dispositivos, que varían según su sistema operativo, aunque es uniforme la existencia de interés por parte de las grandes productoras de estos dispositivos en el desarrollo de tecnologías de este tipo.

Por detrás de la accesibilidad, los sectores principales en los que se engloba este trabajo son el desarrollo de videojuegos, el desarrollo para dispositivos móviles y el desarrollo web. Por lo tanto, fueron imperativas las investigaciones sobre la accesibilidad, sus estándares y prácticas habituales en estos contextos, haciendo especial énfasis en las realizadas con las discapacidades visuales en mente.

Para esta investigación, especialmente en sus primeras etapas, fue instrumental el apoyo proporcionado por la ONCE, que permitió a los estudiantes asistir a una visita a su Centro de Tiflotecnología e Innovación. En esta visita, explicarían la forma en que las personas ciegas utilizan sus dispositivos móviles en el día a día e introducirían las tecnologías que hacen posible este uso. Estos conocimientos sirvieron como base en la que fundamentar la investigación y posterior desarrollo de este trabajo.

El trasfondo de los estudiantes responsables de este trabajo es heterogéneo, al incluir estudiantes tanto del Grado de Ingeniería Informática (GII) como del Grado de Desarrollo de Videojuegos (GDV). Esta dualidad se quiso trasladar al trabajo, añadiendo el enfoque multidisciplinar entre el desarrollo del videojuego y el de la web. De esta forma se buscaba enriquecer el aprendizaje de los participantes, aplicando conocimientos de sendos grados, reforzando de esta forma los propios y familiarizándose con los ajenos.

Además, esta memoria se ha realizado teniendo en cuenta los estándares sobre accesibilidad en la medida de lo posible, utilizando conocimientos adquiridos durante el progreso del proyecto y las herramientas de revisión accesible incluidas en **Microsoft Word** (Microsoft, 1983).

1.1. Motivación

Vista, gusto, oído, olfato y tacto. Los tradicionales cinco sentidos. Y se dicen tradicionales porque hay muchos científicos que afirman distinguir algunos más, como la propiocepción (la percepción que se tiene sobre la situación o posición de las distintas partes del cuerpo) o la termorrecepción, mediante la cual el ser humano es capaz de percibir temperaturas. En todo caso, la pérdida de cualquiera de estos sentidos se traduce en un cambio en la manera en la que se interactúa con el mundo exterior. En particular, perder el sentido de la vista puede suponer un gran problema en la vida de las personas. Solo hace falta intentar realizar cualquier actividad cotidiana con los ojos cerrados para darse cuenta de que la dificultad es desmesurada. De acuerdo con la Organización Mundial de la Salud, se estima que hay 1300 millones de personas con alguna discapacidad visual, de los cuales 36 millones tienen una ceguera total. (OMS, 2018)

Es una realidad que, en la actualidad, todo va enfocado al público general y a menudo las personas con discapacidades no son tenidas en cuenta. Esto es mucho más evidente en el campo de la tecnología. El crecimiento de su uso en la vida cotidiana durante los últimos veinte años ha sido explosivo. Ello ha propiciado un cambio de paradigma en muchas industrias, pero en la del ocio en particular ha sido total. Internet se ha llenado de todo tipo de plataformas y herramientas con las que ocupar nuestro tiempo. Una de estas herramientas ha sido el videojuego, que en las últimas décadas ha llegado a un público mucho más amplio, gracias en gran medida a la aparición de teléfonos inteligentes. Tal ha sido su expansión que ha llegado a colocarse como la industria del ocio que más dinero genera del mundo. Lamentablemente, este rápido desarrollo no siempre ha venido acompañado de alternativas accesibles para usuarios ciegos, particularmente en el sector de los videojuegos, en el que apenas existen antecedentes, y a menudo se trata únicamente de proyectos de investigación.

Es difícil encontrar material de ocio suficiente que pueda ser disfrutado por todo tipo de público y es una pena, considerando que todo el mundo lo necesita para vivir plenamente.

Para poder desarrollar este tipo de productos se debe dar una vuelta al modelo actual de interacción con el usuario, y es por eso por lo que hay pocos videojuegos que cubran estas necesidades.

1.2. Objetivos

El proyecto tiene como objetivo principal el desarrollo de un producto accesible en forma de videojuego, acompañado por una página web. Para ello se ha realizado un estudio profundo de la accesibilidad, centrado principalmente en la manera en la que las personas con discapacidad visual interactúan con las distintas tecnologías. Tanto el videojuego como la página web deben ser completamente accesibles para poder ser disfrutados en su totalidad por personas ciegas.

1.3. Estructura de la memoria

A continuación, se explican brevemente los distintos capítulos de los que consta la memoria:

- **1. Introducción:** Aquí se explican brevemente los temas principales de este trabajo, la motivación detrás de este y los objetivos que se han perseguido.
- **2. Introduction:** Incluye los mismos contenidos de la introducción en inglés.
- **3. Estado del arte:** Se detallan antecedentes en el campo de la accesibilidad en videojuegos en general, y se realizan casos de estudio sobre videojuegos para móvil adaptados para personas ciegas.
- **4. Metodología:** Contiene una explicación sobre nuestra metodología de trabajo, las etapas en las que se dividió el desarrollo y un Diagrama de Gantt.
- **5. Tecnología utilizada:** Aquí se detallan las tecnologías y herramientas utilizadas en el desarrollo, para qué se utilizaron y por qué se optó por ellas.
- **6. Desarrollo técnico:** Contiene, encapsulados en apartados, una descripción detallada de los diferentes elementos que fue necesario diseñar y/o implementar para el trabajo. En cada apartado se habla del período completo de desarrollo, desde sus concepciones iniciales hasta el resultado final, incluyendo los cambios que sufrieran en el desarrollo y las decisiones que los causaron.
- **7. Pruebas:** En este apartado se describen las pruebas realizadas a lo largo del desarrollo sobre tanto el videojuego como la página web, recogiendo como se realizaron, la información obtenida en ellas y cómo se aplicó.
- **8. Conclusiones y trabajo futuro:** Conclusiones sobre el trabajo, donde se considera si los resultados cumplen con los objetivos planteados. También incluye varios caminos posibles para la realización de trabajo futuro sobre este proyecto.
- **9. Conclusions and future work:** Incluye los mismos contenidos del anterior en inglés.
- **10. Contribuciones:** Se indican las contribuciones realizadas por cada estudiante.


2. Introduction

This TFG was conceived as the implementation of a videogame accessible for blind players, this meaning people with a total vision loss or those who still have same vision traces. The development would be accompanied by an infrastructure on the internet, formed by a webpage, also accessible, and a server, that would serve to store save data from the game and host said web. Because of this, the main subject of this work is accessibility, being the main objective that the products of this development may be used with ease by people with this disability.

From the beginning, the development was set to be on smartphones as the target platform, since they are often used by these people. This is mainly because of the accessibility options given by these devices, which vary depending on their operating system. Furthermore, the existence of interest in the development of this kind of technologies is common in the main manufacturers of these devices.

Apart from accessibility, the main subjects of this work are game development, development for smartphones and web development. Thus, investigations on accessibility, its standards and common practices, were necessary, especially in terms of visual disabilities.

For this investigation, especially in its early stages, the support given by ONCE were of great help and importance, allowing the students to visit their installations. In this visit, the way in which blind people use their mobile devices on their daily life was explained, along with the technologies that make this possible. This knowledge would serve as the base for the investigations and later development of this project.

The background of the students responsible for this work is varied, including students both from the Computer Engineering Degree (GII, in Spanish) and the Game Development Degree (GDV, in Spanish). This variety was transferred to the project itself, adopting a multidisciplinary approach in the development of the game and web. This was done to enrich the learning of the participants, applying knowledge from both grades. This would help to strengthen the knowledge from each student own grade while gaining insight on that from the others.

Furthermore, this document has been done taking into account accessibility standards when possible, using the knowledge gained in the making of this project and the accessibility tools included in Microsoft Word.

2.1. Motivation

Sight, taste, hearing, smell and touch. The traditional five senses. Traditional, because many scientists state the existence of more, such as proprioception (each one's perception of the position of the different parts of their bodies) or thermoreception (by which the human being can discern temperatures). In any case, the loss of one of these senses translates to a drastic change in the way a person interacts with the world. Losing the sight in particular can be a great problem in the life of people. One needs only to try any day-to-day activity with their eyes closed to realize the tremendous difficulty. According to the World Health Organization, it is estimated that about 1,3 billion people has some level of visual disability, of which about 36 million are blind. (OMS, 2018).

It is a reality that nowadays everything is with a majority of people in mind, and people with disabilities are often not taken into account. This is even more evident in the field of technology. The use of such technology on our daily life has experienced an explosive growth in the last 20 years. This has propitiated a change in paradigm in many industries, especially in leisure. The internet has become chockfull of all kind of platforms and tools with which to occupy our free time. One such tool has been the videogame, that in the last decades has reached an even bigger audience, thanks in no small part to the appearance of smart mobile devices. Such has been the growth in this sector that it has taken the place as the most profitable leisure industry worldwide. Regrettably, this rapid growth hasn't always come accompanied by accessible alternatives to blind users, particularly in terms of videogames. In this sector there are not many antecedents, and they often are part of investigation projects.

This means it is often difficult to find media that can be enjoyed by all kinds of public, and it's a pity, given the positive impact in can have in our life. To develop this kind of products its necessary to reconsider the usual model in which the user interacts with it from the ground up. It is because of this reason that not many videogames do it.

2.2. Objectives

The projects main objective is the development of an accessible product in the shape of a videogame, accompanied by a webpage. To this end, a thorough study on Accesibility was conducted, focused on the way blind people interact with various technologies. Both the videogame and the webpage must be completely accessible to blind people so that they can be able to enjoy them.

2.3. Document Structure

Here, the chapters conforming this document are briefly explained:

- **1. Introducción:** Briefly explains the subject of this work, the motivation behind it and its objectives.
- **2. Introduction:** The contents from chapter 1 translated to English.
- **3. Estado del arte:** Details antecedents in the field of videogame accessibility, developing case studies on games for mobile devices accessible to blind people.
- **4. Metodología:** Contains an explanation on the work methodology, the stages of the development and a Gantt Diagram.
- **5. Tecnología utilizada:** Details the various technologies and tools used in the development, along with why they were used and on what.
- **6. Desarrollo técnico:** Contains, separated in sections, detailed descriptions of the elements required to design and implement this project. In each section the full development cycle is described, from conception to final result, including relevant changes and the reasons that caused them.
- **7. Pruebas:** Contains the tests realized throughout the development both on the game and the webpage. Explains how they were realized, the information obtained and how it was used.
- **8. Conclusiones y trabajo futuro:** Conclusions obtained from the work and assessments of the results obtained against the initial objectives. Also lays out several possible paths to take in future developments over this work.
- **9. Conclusions and future work:** The contents from the previous chapter translated to English.
- **10. Contribuciones:** The contributions done by each student are mentioned.

3. Estado del arte

El mercado de juegos para personas ciegas es muy reducido. Al mismo tiempo que se pensaba en una idea para el juego, se realizó una búsqueda de antecedes, videojuegos ya creados que servirían como referentes e inspiración. De ese modo se adquiriría conocimiento sobre prácticas comunes en este tipo de desarrollo, tanto positivas como negativas, que sirvieran en este para crear algo accesible e innovador. Por esto se dio prioridad a videojuegos desarrollados para dispositivos móviles, la plataforma objetivo del proyecto. Se hablará en detalle de los antecedentes más destacables en sus respectivos casos de estudio.

Antes de ello, cabe hablar de la accesibilidad en el medio de los videojuegos, su estado actual, evolución, y las diversas formas que toma, tema sobre el que se realizó otra labor de investigación.

3.1. Accesibilidad en videojuegos

La accesibilidad en el medio de los videojuegos es un tema amplio. Esto se debe a la gran variedad de medidas que los desarrolladores pueden tomar para hacer sus productos accesibles, estas medidas nacidas a su vez de los diversos tipos de discapacidades e impedimentos varios que pueden sufrir los jugadores. También cabe destacar la ausencia de estándares y guías de uso generales en este campo, siendo las que existen fruto del trabajo de particulares interesados en el tema o de uso interno entre las grandes desarrolladoras. Esta falta de estándares lleva a que cada desarrollador tome las medidas que considere oportunas, si es que decide incluir opciones de accesibilidad. Esto complica a los usuarios interesados conocer el grado de accesibilidad de los productos.

Para dar contexto, se hablará de los principales problemas enfrentados por personas con diferentes condiciones y medidas recomendadas para mejorar la accesibilidad del juego. Estos estarán divididos en cuatro tipos: Auditivos, visuales, motores y cognitivos.

A la hora de realizar esta investigación fueron instrumentales los recursos proporcionados en **Game Accesibility Guidelines** (Esfuerzo colaborativo (Ellis, Ford-Williams, Graham, Grammenos, Hamilton, Lee, Manion & Westin), 2019), página web creada por varios profesionales del sector involucrados en la accesibilidad, que recopila documentación detallando diversas medidas para mejorar la accesibilidad de distintos elementos de un videojuego.

Estas medidas las clasifican en categorías (básicas, medias y avanzadas) basándose en su potencial alcance (número de personas cuya experiencia mejoraría), impacto (cuánto mejoraría su experiencia como consecuencia de esa medida) y coste (el coste que implicaría su implementación).

También resultó de gran utilidad la serie **Designing for disability** (Brown, 2019), creados en colaboración con **Game Accesiblity Guidelines** y consultando a varios grupos de jugadores con diversas condiciones.

3.1.1. Auditivas

En este grupo se incluyen los elementos que afectan a personas sordas o con pérdidas de audición. Lo principal para hacer un juego accesible a este tipo de personas es que toda la información esencial no se transmita exclusivamente por audio, sino que también se refleje, por ejemplo, con un indicador visual. Este es uno de los apartados que más se suele tener en cuenta, una gran cantidad de videojuegos incluyen subtítulos y cada vez más incluyen indicadores visuales adicionales, especialmente los de disparos. Además, es sencillo realizar pruebas, ya que basta con ejecutar el juego con el audio silenciado.

- **Subtítulos:**
  Uno de los principales elementos afectados por el audio son los diálogos y formas habladas transmitidas por audio. La forma más común de hacer accesibles estos elementos son los subtítulos. Cabe remarcar que deberían subtitularse todos los diálogos posibles, especialmente si dan información importante, en muchos casos a pesar de incluir subtítulos en las escenas principales, estos se omiten en otras partes del videojuego. Sin embargo, no basta con incluirlos, para que sean accesibles deben cumplir ciertas condiciones. Los subtítulos no son exclusivos de este medio, aparecen de forma intrínsecas en otros medios como el cine y la televisión (Netflix, 2019) (BBC, 2019). Los procedimientos explicados en estas guías también pueden aplicarse los videojuegos:

    - Ofrecer fuentes de mayor tamaño, preferiblemente hacerlo ajustable.
    - Usar fuentes claras y fáciles de leer, el uso de fuentes temáticas con el juego puede dificultar la legibilidad. En cualquier caso, se puede dar la opción al jugador de elegir entre una y otra.
    - Diferenciar claramente el texto del fondo, que puede variar según el momento. Para esto se pueden usar rebordes negros, sombreados, o mejor aún, una caja negra que envuelva el texto.

- Usar frases de un tamaño apropiado, no demasiado largas, para facilitar la lectura. Cortarlas en puntos coherentes.
- Señalizar claramente quién está hablando. Para esto suelen usarse códigos de color o indicar el nombre antes de cada frase.

- **Señales auditivas posicionadas:**
  En muchos casos hay audios que transmiten información de gran importancia al jugador, un ejemplo claro serían los sonidos de disparos, explosiones y demás en juegos de disparos. Otro ejemplo serían diálogos de personajes en juegos de sigilo. En estos casos, la posición de la que proviene el sonido es tan importante como el sonido en sí mismo. Para transmitirla hay varias opciones, si la estamos transmitiendo con un texto, este texto se puede posicionar en la pantalla para indicar su ubicación. En otros casos más comunes como los disparos a menudo se implementan indicadores visuales específicos, por ejemplo, como marcas en los bordes de la pantalla o en una rueda centrada.

- **Puntos de bloqueo:**
  A la hora de plantear la accesibilidad, también hay que tener en cuenta que en ciertos juegos, por sus características específicas, puede suponer un mayor trabajo hacerlos accesibles, o directamente carecer de sentido. Esto por ejemplo sucedería en juegos basados en gran medida en audio, como los juegos rítmicos basados en música u otros que lo usen intensivamente para sus mecánicas. Este no es el caso, sin embargo, cuando en un juego hasta entonces perfectamente accesible para personas con discapacidades auditivas se incluye una prueba basada en audio que resulta obligatoria para avanzar, impidiendo a estas personas continuar. En estos casos una solución sería adaptar la prueba para que también proporcione esta información por medios visuales o dar la opción al usuario de saltarla.

3.1.2. Visuales

En este grupo se incluyen las diferentes formas de daltonismo, pérdidas en la capacidad visual y ceguera. Si bien es relativamente habitual encontrar opciones para daltonismo y cada vez más desarrolladores incluyen opciones para modificar el tamaño de textos e interfaces, en muy pocos casos se desarrollan juegos con las personas ciegas en mente. Esto requeriría un desarrollo específico para este público, muy reducido, puesto que adaptar un juego ya existente para hacerlo accesible a estas personas supondría realizar muchos cambios y por tanto un alto coste.

Es por esto por lo que la mayoría de los juegos adaptados para personas ciegas son fruto de proyectos de investigación o financiados por diversas organizaciones. Aun así, hay ciertos tipos de juego y géneros que son más susceptibles a ser adaptados que otros, como se explicará más adelante.

- **Daltonismo:**
  El daltonismo es una condición que afecta a la forma en la que percibimos los colores, causando que ciertos colores resulten imposibles de distinguir unos de otros por las personas afectadas. Además, existen varios tipos de daltonismo, que tienen efectos diferentes, afectando a la percepción de distintos colores. Para realizar pruebas sobre los efectos de estos trastornos sobre un juego si no se dispone de personas daltónicas como probadoras existen varios recursos gratuitos como **Color Oracle** (Jenny, 2018), y motores de creación de videojuegos de uso habitual como **Unreal Engine** ofrecen esta funcionalidad (Epic Games Inc., 2020).

  Una medida tomada a menudo consiste en dar la opción de aplicar un filtro al juego, que debería ayudar a hacer estos colores más diferenciables. Sin embargo, plantea ciertos problemas. Primero, afecta a toda la pantalla, no solo a los elementos importantes que no serían distinguibles a causa del daltonismo. Además, suele incluirse únicamente uno de estos filtros, dejando de lado a los demás tipos de daltonismo. De todos modos, esta solución puede ser correcta, si proporciona varios filtros diferentes y que han sido probados.

  Una aproximación mejor a este problema es plantearlo desde el diseño del juego, evitando que haya elementos de gran importancia que solo sean diferenciables por el color. Para este fin pueden añadírseles otros atributos distintivos como formas, texturas y sonidos. También puede resultar útil incluir la opción de añadir bordes u otros indicadores visuales que resalten estos elementos. En ciertos juegos en los que el código de color es de gran importancia, por ejemplo, para diferenciar el equipo propio del rival, una muy buena alternativa es dar la posibilidad al jugador de modificar estos colores.

- **Problemas de visión leves:**
  En casos de personas con capacidades visuales reducidas, pero sin llegar a ser de mayor gravedad, basta con mejorar la visibilidad de los elementos de la pantalla. Para realizar pruebas de este tipo algo muy básico pero que a menudo se ignora es probar el juego en una pantalla a cierta distancia o de tamaño reducido, en lugar de hacerlo con el monitor desde el que se trabaja.

Para mejorar la claridad de elementos de la interfaz y textos son útiles muchas de las medidas explicadas para los subtítulos: ofrecer tamaños mayores para los textos del juego, utilizar fuentes claras o dar opciones de tamaño para los elementos de la interfaz. Además, pueden ser útiles ciertas opciones del visualizado, como permitir aumentar el contraste con el fondo y añadir indicadores visuales que destaquen elementos importantes.

Como siempre, lo mejor es que el usuario tenga libertad de activar y desactivar estas opciones y modificar sus valores. Estas opciones de visualización pueden resultar de ayuda también a personas que sufran cinetosis (mareo por movimiento), reduciendo el mareo causado por ciertos efectos de visualizado y ajustes de cámara como el *depth of field* o el *motion blur*, estos mareos son particularmente habituales en los juegos desarrollados para realidad virtual, entorno que plantea sus propios desafíos tanto en accesibilidad como en las demás secciones del desarrollo.

- **Personas ciegas o con restos visuales:**
  Para personas con problemas de visión más graves o pérdidas totales de visión el enfoque anterior no sirve, ya que por mucho que mejoremos la claridad de los elementos seguirán sin poder percibirlos. Deben darse grandes cambios en el juego para conseguir que toda la información, o al menos la de mayor importancia transmitida visualmente, lo haga por otros medios, habitualmente el auditivo.

  Para realizar pruebas con este tipo de accesibilidad, al igual que en los demás, es óptimo hacerlas con personas con estas condiciones, en este caso es especialmente importante, ya que la forma en que interactúan con los dispositivos (teclados, apenas utilizan el ratón, teléfonos móviles, etc.) y los gestos utilizados son muy distintos de los habituales. Aun así, en caso de no poder hacerse, una alternativa habitual son las pruebas a ciegas, en las que los probadores no ven nada de la aplicación mientras las realizan, bien apagando las pantallas, bien con una venda.

  Lo imprescindible en un videojuego adaptado para personas ciegas es que toda la información necesaria para jugar se transmita por canales ajenos al visual. El principal y más usado es el auditivo, aunque otros medios como las vibraciones también pueden ser útiles. Además, es importante eliminar o mitigar la necesidad de *inputs* basados en una posición concreta (por ejemplo, pulsaciones con el ratón o toques en la pantalla en posiciones específicas). Por esto suelen favorecerse los *inputs* de teclado o mando frente a los de ratón en ordenadores.

Para esto, lo que se hace es añadir toda la información posible como audio, ya que esto no solo ayuda a las personas ciegas, sino que también mejora la accesibilidad de jugadores con problemas de visión más leves o de jugadores cualesquiera en situaciones que dificulten la visibilidad. El audio es una parte esencial de un videojuego, y es por esto por lo que se usa muy a menudo para transmitir información, al margen de la accesibilidad. Este proceso, sin embargo, resulta inabordable en la mayoría de los videojuegos, ya que este se trata de un medio con un fortísimo componente visual.

Aun así, se han dado casos de juegos que, por su propia naturaleza, se prestan más a esta adaptación. Es curioso el caso de ciertos juegos de lucha, que por su escenario y movimiento limitado y gran cantidad de *feedback* sonoro de base, gozaban de cierto seguimiento entre jugadores ciegos, incluso participando en competiciones oficiales (Gach, 2017). Debido a este inesperado seguimiento, el género de los juegos de lucha se ha consolidado como uno de los pioneros en este campo, escuchando las sugerencias de jugadores y tomando iniciativas. Entre ellas destacan el caso de **Injustice**: **Gods Among Us** (NetherRealm Studios, 2013), que tras recibir peticiones de jugadores ciegos incluyeron la opción de un modo de accesibilidad, que incluía más indicaciones sonoras. (Straub, 2014) o el de **Skullgirls** (Lab Zero Games, 2012), que en una actualización incluyó soporte a lectores de pantalla, entre otras opciones de accesibilidad (Lab Zero Games, 2014).

Otro género con mucho potencial en este tipo de accesibilidad son las aventuras gráficas y similares, basados en gran medida en texto. Estos juegos, una vez adaptado su sistema de control y proporcionado soporte para lectores de pantalla o implementada una funcionalidad similar, no tendrían grandes problemas a la hora de hacerse accesibles a personas ciegas.

De todos modos, los casos anteriores son de juegos lanzados al público general, que más tarde añadirían estas opciones de accesibilidad por petición de los jugadores. A la hora de crear desde cero un videojuego para personas ciegas, a menudo se toman otras consideraciones. Lo más habitual es desarrollarlos para dispositivos móviles. Tiempo atrás, la plataforma **iOS** (Apple Inc., 2007) era la más habitual, al proporcionar más soporte a la accesibilidad que la competencia. Sin embargo, tras ponerse al día **Android** (Google, 2008) con sus propios sistemas, actualmente existe un nicho de aplicaciones adaptadas en ambas plataformas.

Es habitual que estos juegos se basen principalmente en audio y hagan uso de los lectores de pantalla de sendas plataformas o, en su defecto, imiten su modo de uso y funcionamiento general. Sobre esto se profundizará en la sección Diseñando la accesibilidad de la aplicación. Lectores de pantalla.

3.1.3. Motoras

En este grupo se incluyen las discapacidades motoras, incluyendo aquellas que dificultan la interacción física con los dispositivos de entrada, por ejemplo, a la hora de realizar secuencias de *inputs* complicadas, pruebas que requieran reflejos rápidos, pulsaciones prolongadas en el tiempo, de múltiples botones o teclas u otras actividades que supongan un esfuerzo físico que puede resultar de gran dificultad para estas personas. Para realizar pruebas de este tipo no hay sustitutivos claros a las pruebas con personas que padezcan estas condiciones, aparte de tratar de ponerse en su lugar y tener en cuenta su caso en el propio diseño del juego y realización de las pruebas. Entre las medidas recomendables destacan:

- **Mapeado de controles:**
  Permitir al jugador modificar el mapeado de los controles es de gran utilidad en este tipo de casos, ya que les permite adaptar la interacción con estos a la forma que les resulte más conveniente. Esta opción es muy habitual en juegos lanzados para ordenadores, pero no tanto en consolas, aun tratándose en ocasiones del mismo juego. Algunas consolas proporcionan la posibilidad de cambiar este mapeado a nivel de consola, de modo que se aplica en sus propios menús y en todos los juegos. Esta opción, aunque es una inclusión positiva, no sustituye al mapeado en cada juego, ya que esto permite al usuario adaptarlo a cada juego, con sus diferentes requisitos.

  Cabe mencionar también los avances en dispositivos adaptados para personas con discapacidades de este tipo, en particular el caso de los mandos adaptativos de Xbox (Microsoft, 2019), que permiten además de un mapeado extensivo de los botones la modificación de las propias partes del mando para que el jugador lo adapte a sus necesidades. Estos mandos además son soportados por varias plataformas.

- **Reducir exigencias físicas**

  Los QTE (*quick time event*, o eventos de tiempo rápido) son un tipo de mecánica que requiere que el jugador realice unas acciones específicas, generalmente en un período de tiempo reducido desde que se le indica que debe hacerlo, también suele incluir otros requerimientos como realizar muchas pulsaciones rápidamente o realizar giros con *joysticks*. Este tipo de mecánica requiere del jugador reflejos rápidos, coordinación y esfuerzo físico. Es por esto por lo que resultan muy difíciles o hasta imposibles para personas con discapacidades motoras, llegando a poder causar dolor o lesiones. Este tipo de mecánicas han existido durante mucho tiempo, pero su uso se hizo muy habitual a finales de la década de los 2000 y principios de la siguiente, acabando por ser habituales reacciones negativas de jugadores al respecto. Actualmente su uso parece haberse reducido y muchos desarrolladores proporcionan alternativas, como reducir notablemente su dificultad, o directamente permitir omitir esas secciones.

  También cabe mencionar los juegos que utilizan controles por movimiento en algunas acciones específicas, ya que este tipo de control pueden resultar imposibles a personas con este tipo de condiciones, la solución pasaría por proporcionar controles convencionales como alternativa.

- **Proporcionar configuraciones**

  También es importante proporcionar la posibilidad de modificar diversos valores asociados al *input*. Cabe destacar la sensibilidad de la cámara, elemento prácticamente omnipresente en los videojuegos en 3D. Una opción menos obvia son los ajustes de la vibración del dispositivo de entrada, esta debería poder reducirse u omitirse, transmitiendo su información por otros medios, ya que puede causar dolores a una de estas personas o hacer que pierda el agarre del dispositivo.

  Otra configuración muy habitual son los ajustes de dificultad, que suelen ofrecer varios modos preestablecidos (Fácil, Medio, Difícil, etc.). Estos modos ayudan, ya que pueden reducir el desafío para una persona con una discapacidad de este tipo. De todos modos, al igual que en el resto de las opciones de accesibilidad, cuanta más flexibilidad se proporcione al jugador a la hora de ajustar su experiencia de juego, mejor. Frente a este enfoque de modos preestablecidos la alternativa es ofrecer configuraciones más concretas, que permitan modificar elementos específicos. De este modo, podrían configurar el elemento que cause problemas y mantener los demás intactos.

3.2. Caso de estudio: A Blind Legend

Uno de los juegos accesibles más conocidos se llama **A Blind Legend** (Dowino, 2014). Este es un juego de acción y aventuras para varias plataformas donde el sonido es lo que propicia todas las características del juego, desde el movimiento por los escenarios hasta los combates.
En el juego encarnas a un maestro espadachín con esposa e hija. La historia base trata del secuestro de su mujer por parte de un malvado personaje (el cual también es el responsable de su pérdida de visión), y las siguientes peripecias hasta su posterior rescate. El héroe emprende un largo viaje junto a su hija (quien le guía a menudo) con el objetivo de rescatar a su mujer y vengarse del villano, atravesando por el camino peligrosas ubicaciones y enfrentándose a numerosos enemigos.

El videojuego no posee apartado visual, debido, además de su naturaleza accesible y objetivo general del juego, al acompañamiento por parte de la historia, ya que el protagonista tampoco puede ver, provocando así al jugador un mayor sentimiento de empatía. La pantalla es constantemente negra con tonos grisáceos, y en ocasiones tiene pequeños efectos visuales (como simples movimientos de la espada en los combates).

Las mecánicas de juego fundamentales se basan en el movimiento por escenarios con la ayuda de la voz de la hija, quien indica hacia dónde se debería girar propiciando así un pseudo sistema de guiado 3D, y combates de acción por sonidos, en los que se debe atacar en el momento justo para dañar al oponente.

Cabe mencionar el modo en que funciona el *input*:
En lo relacionado al movimiento, el jugador puede avanzar en cuatro direcciones. Puede girar 90º realizando un deslizamiento en esa dirección. Para avanzar, el jugador debe realizar un deslizamiento vertical y mantener el dedo sin levantar en esa posición para seguir avanzando, tal como se muestra en la Figura 1 - Gesto de avance .

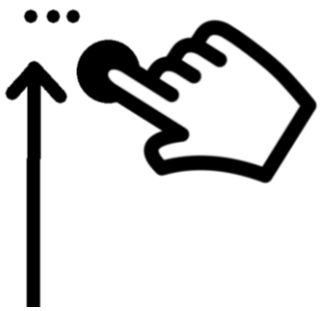


Figura 1 - Gesto de avance

En el combate, cada enemigo distinto tiene un audio propio el cual, por norma general, tiene ciertas características sonoras donde se le ofrece al jugador la sensación de vulnerabilidad. Esto se produce en el punto más álgido del audio y permite al jugador reaccionar en un pequeño lapso, tal como se ve en la Figura 2 - Gráfico de reacción A Blind Legend. En este tiempo, el jugador deberá realizar un deslizamiento vertical para dañar al enemigo. Si no lo hace a tiempo, perderá vida y su pulso aumentará.

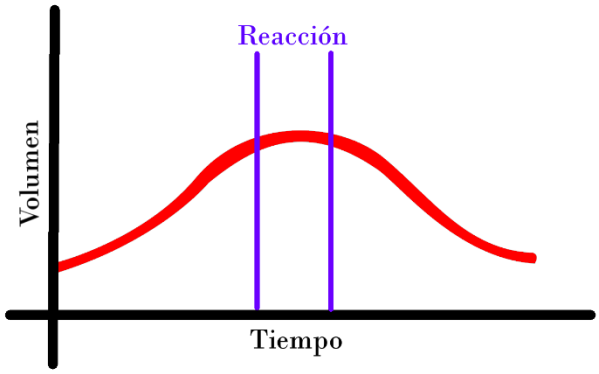


Figura 2 - Gráfico de reacción A Blind Legend

Además, el juego posee una voz explicativa en determinados momentos nuevos para el usuario, con la que se suelen enseñar las mecánicas que acompañarán al jugador durante el resto de la historia.

Aun siendo el videojuego más conocido para personas ciegas, y estando desarrollado por un estudio con fuertes inversiones y con un buen apartado técnico y sonoro, se pensó que varios aspectos siguen siendo mejorables, como el del movimiento y guiado, ya que ciertas situaciones provocan la "pérdida" del jugador y propician estrés. Además, a nivel mecánico se pensó que había potencial sin aprovechar del todo.

A Blind Legend está desarrollado para sistemas **Android** y **iOS**, siendo gratuito en sus dos versiones. El videojuego también se encuentra en la plataforma **Steam** (Valve Corporation, 2003), aunque de forma no gratuita. Fue desarrollado por el estudio francés **DOWINO** en colaboración con una emisora de radio (**France Culture**). También recibió asistencia y financiación de diversas entidades y asociaciones, además de una campaña de micro financiación que concluyó en éxito (DOWINO, 2014) en la plataforma **ULULE** (Boucherot, 2010).

3.3. Caso de estudio: FREEQ

Mientras se buscaban antecedentes de juegos adaptados para personas ciegas desarrollados en **Unity** (Unity Technologies, 2005), se dio con la historia de **FREEQ**, (Hughes, 2013) desarrollado por el estudio **Psychic Bunny** (Psychic Bunny, 2016) y lanzado para **iOS** en 2013.

Este juego estaba basado en el sonido (se denominaba una aventura de audio) con la mayoría del trabajo centrado en esta componente, aparte de elementos visuales mínimos y algunos menús. Tras superar con éxito una campaña de micro financiación en la plataforma **Kickstarter** (Vigil, 2013) este juego fue lanzado en abril de 2013. El desarrollo lo habían realizado en el motor de videojuegos **Unity** y lo habían lanzado sin tener en cuenta la accesibilidad para jugadores ciegos, colectivo de cuya existencia desconocían. El juego consiguió un éxito moderado teniendo en cuenta su escala y la poca fama del estudio, ganando adeptos entre los fanáticos del audio.

Fue tras el lanzamiento del juego cuando los desarrolladores comenzaron a recibir por varios canales comentarios de jugadores ciegos que tras haber leído la descripción del juego en sus dispositivos lo habían adquirido para llevarse la desagradable sorpresa de no poder jugarlo al no estar adaptado. Ante este descubrimiento decidieron investigar sobre cómo se podría implementar esa accesibilidad. En esta tarea les fue de mucha ayuda la información aportada por los jugadores que les habían contactado y que además les indicaron los canales en los que habitualmente se hablaba de la accesibilidad de juegos y en los que se encontraban otros desarrolladores. Fue entonces cuando aprendieron sobre **VoiceOver** (Apple Inc, 2005), el lector de pantalla de dispositivos **iOS**, el cuál hasta donde sabían les facilitaría mucho el trabajo de hacer la aplicación accesible. Esperaban esto porque en muchas plataformas de desarrollo para **iOS**, **VoiceOver** una vez activado se encarga por sí mismo de reconocer elementos como texto y elementos interactivos.

Sin embargo, **Unity** no era una de estas plataformas ya que, como descubrieron al lanzar la aplicación con **VoiceOver** activado y ver que se quedaba bloqueada, no da soporte a este lector de pantalla y, dado el caso, a ningún otro. En lugar de descartar **VoiceOver** decidieron aprender lo que podían sobre cómo funciona y su forma de uso para poder así adaptar esos gestos comunes a una implementación propia en **Unity** que imitara el funcionamiento del lector. Durante este desarrollo les fue de inestimable ayuda el apoyo de los jugadores ciegos con los que habían contactado antes, que hicieron de probadores y asesores durante todo este proceso.

Este proceso les resultó relativamente rápido y sencillo, ya que su juego se basaba en audio y apenas tenía componente visual. A medida que hacían progreso comenzaron a compartirlo en páginas web usadas habitualmente por jugadores ciegos y recibieron una respuesta muy positiva de personas que esperaban el juego y prometían adquirirlo cuando fuera lanzado. Los desarrolladores hacen hincapié en lo agradecidas y comprensivas que eran estas comunidades.

Finalmente, tras este segundo ciclo de desarrollo, relanzaron el juego en su nueva forma accesible, publicando previamente anuncios y tráileres enteramente en audio. El equipo relata que la acogida fue mucho mejor a la primera vez que lo lanzaron, logrando en 3 días el mismo número de descargas que en 6 meses.

Destacan sobre todo lo positivos y agradecidos que eran los jugadores ciegos, que además de proporcionar *feedback* constante y de utilidad compartían el juego con sus conocidos y por redes sociales. También denotan la respuesta positiva a los tráileres en audio, ya que no es habitual para estos jugadores que los desarrolladores apelen directamente a ellos.

Un detalle al que también dan importancia los desarrolladores es que, al haber usado para todas las voces e instrucciones del juego grabaciones de voz real en lugar de sintetizada, los jugadores experimentaban mayor inmersión en el juego y era a menudo una de sus partes favoritas del juego. Esto no les costó demasiado partiendo del juego base ya centrado en el audio, además, un miembro del equipo también era actor de voz y se encargó de grabar y editar los audios que fue necesario añadir.

4. Metodología

La organización y la eficiencia son clave a la hora de realizar cualquier proyecto. Con los mismos recursos, dos proyectos con metodologías y planificación distintas difieren en el tiempo y en el resultado. Este caso tiene supone desafío particular, debido a que el grupo era de seis personas, y además de dos competencias que, aunque afines, distintas. Esto supuso un gran reto, puesto que ambos equipos se tuvieron que amoldar a la forma de trabajar del otro. A su vez, cada grupo se ha enriquecido y ha compartido conocimientos que de no haber sido así no se hubieran adquirido.

La metodología que se ha seguido es *agile*, pues se decidió que era la mejor para mantener una buena comunicación entre los dos equipos y la que mejor se ajustaba al proyecto. En la Figura 3 se puede observar que se hacen iteraciones constantes a las distintas partes del proyecto. Las principales características de esta metodología son:

- Reuniones frecuentes para resolver problemas y ver avances.
- Trabajo cercano.
- Requisitos cambiantes.
- Producto funcional.
- Auto gestión de los equipos.
- Entregas frecuentes.

Figura 3 - Metodología agile

Se pueden distinguir tres fases bien diferenciadas en este proyecto y en cada una se tiene una organización y comunicación distintas:

4.1. Fase 1. Definición del proyecto.

Septiembre – Noviembre

En esta fase se tuvieron reuniones presenciales quincenales en las cuales, mediante tormenta de ideas, se fueron recopilando ideas para ir dándole forma al proyecto, puesto que el concepto de "Videojuego narrativo para personas con discapacidad visual" era muy amplio. La principal comunicación entre los estudiantes se hizo por la aplicación de mensajería instantánea **Whatsapp** (WhatsApp Inc., 2009) y las ideas que se fueron obteniendo, así como enlaces y algún borrador, se guardaban en una unidad compartida de **Google Drive** (Google, 2012).

Se obtuvo una primera idea de un juego de cartas, concretamente póquer accesible, pero se descartó porque se buscaba algo más innovador y decidió enfocarlo más a una aventura narrativa. En esta fase también se realizó una visita a la **ONCE**, la cual ayudó mucho en estas etapas iniciales.

4.2. Fase 2. Investigación, desarrollo e implementación hasta COVID-19

Diciembre – Marzo

Durante la segunda fase, con la idea de proyecto bien definida, se empezó a repartir tareas entre los estudiantes participantes, estas aparecen en la sección de Diagrama de Gantt y en Contribuciones. En esta fase, a grandes rasgos, el equipo del GDV se encargaría de implementar la base del juego en **Unity** con los movimientos accesibles, mientras que el equipo del GII se encargaría del servidor web, el sistema de síntesis de voz en base a texto (TTS) y el de síntesis de texto en base a voz (STT).

Durante el mes de diciembre, se desarrollaron prototipos y partes del código de la aplicación. Pero cuando llegó la época de exámenes en la facultad, las tareas del proyecto se pararon temporalmente. Esto ocurrió entre enero y mediados de febrero. A partir de ese punto, se retomó con fuerza y se volvió al funcionamiento normal. Las reuniones se mantuvieron en persona, pero más espaciadas en el tiempo. Eso sí, siempre en continuo contacto a través de mensajería instantánea. Llegados a este punto no había tantas dudas de concepto y había que ponerse manos a la obra.

La comunicación y organización pasó a unificarse, a excepción del código de la aplicación, a través de la aplicación **Microsoft Teams** (Microsoft, 2017), la cual proporciona almacenamiento en la nube, mensajería instantánea, trabajo colaborativo en tiempo real con **Microsoft Word** (Microsoft, 1983), planificación de las tareas y llamadas de voz y video.

El código de la aplicación se alojó en **Github** (Github Inc., 2007), ya que gracias a las características de **git** permite tener un control de versiones, así como las contribuciones de cada miembro en una línea temporal.

4.3. Fase 3. Finalización de la implementación durante COVID-19

Marzo – Junio

En la tercera y última etapa, el equipo fue sorprendido por el Estado de Alarma provocado por el COVID-19, el cual supuso que a partir de ese día todas las reuniones y trabajo debieron hacerse telemáticamente.

Las reuniones con los directores del TFG se hicieron cada dos semanas aproximadamente de forma telemática, en ellas se fueron enseñando las versiones estables y funcionales del juego, depurando fallos y consiguiendo *feedback.*

Estas reuniones aumentaron su frecuencia a semanal en las últimas etapas tanto entre los desarrolladores como con los directores, en las que se intensificó la comunicación para la realización de pruebas sobre los resultados finales y la elaboración de la memoria.

4.4. Diagrama de Gantt

Para mostrar de forma gráfica el transcurso de las tareas del proyecto, se ha realizado un diagrama de Gantt (Figura 4 - Diagrama de Gantt). En él se muestra el tiempo que se ha dedicado a las diferentes tareas del proyecto, aunque no son necesariamente proporcionales a la cantidad de trabajo de estas. Así, se quiere proporcionar una visión general de las localizaciones temporales de cada tarea del trabajo. El diagrama se ha dividido en cinco apartados: Aspectos generales, Videojuego, Base de datos, Web y Otros.

Como se mencionaba en los apartados anteriores de Metodología, durante los primeros meses el equipo se dedicó a los objetivos de diseño y prototipado mayoritariamente. Durante la época de exámenes, el diagrama muestra vacío en todas las tareas del trabajo. A partir de febrero, llega la parte fundamental del desarrollo, cumpliendo los objetivos de la mayoría de las tareas designadas. Al llegar a mediados de abril, la mayor parte de los objetivos fundamentales estaban terminados, por lo que se continuó con el perfeccionamiento de las tareas sin concluir, y el desarrollo de la parte de postproducción, formado por el *testing*, el pulido tanto del videojuego como de la web, y el desarrollo de la memoria.

Tal como se ve en el diagrama, se han añadido ciertos *milestones* o hitos del proyecto. Pese a que la aplicación se dio a probar constantemente y teníamos versiones relativamente

estables de forma continuada, estos hitos reflejan partes importantes en el desarrollo del trabajo. Así, se observan dos *milestones*:

- MS1: En este punto se tenía una aplicación con introducción y primera habitación, lo cual era jugable y mostraba las mecánicas base del juego, además de tener la base de datos conectada a este.
- MS2: Aquí, la aplicación ya estaba terminada en su totalidad. Se tenían todas las funcionalidades desarrolladas, a la espera del *feedback* proporcionado por el *testing*. Además, la base de datos estaba completa y la web, a falta de pequeños retoques.

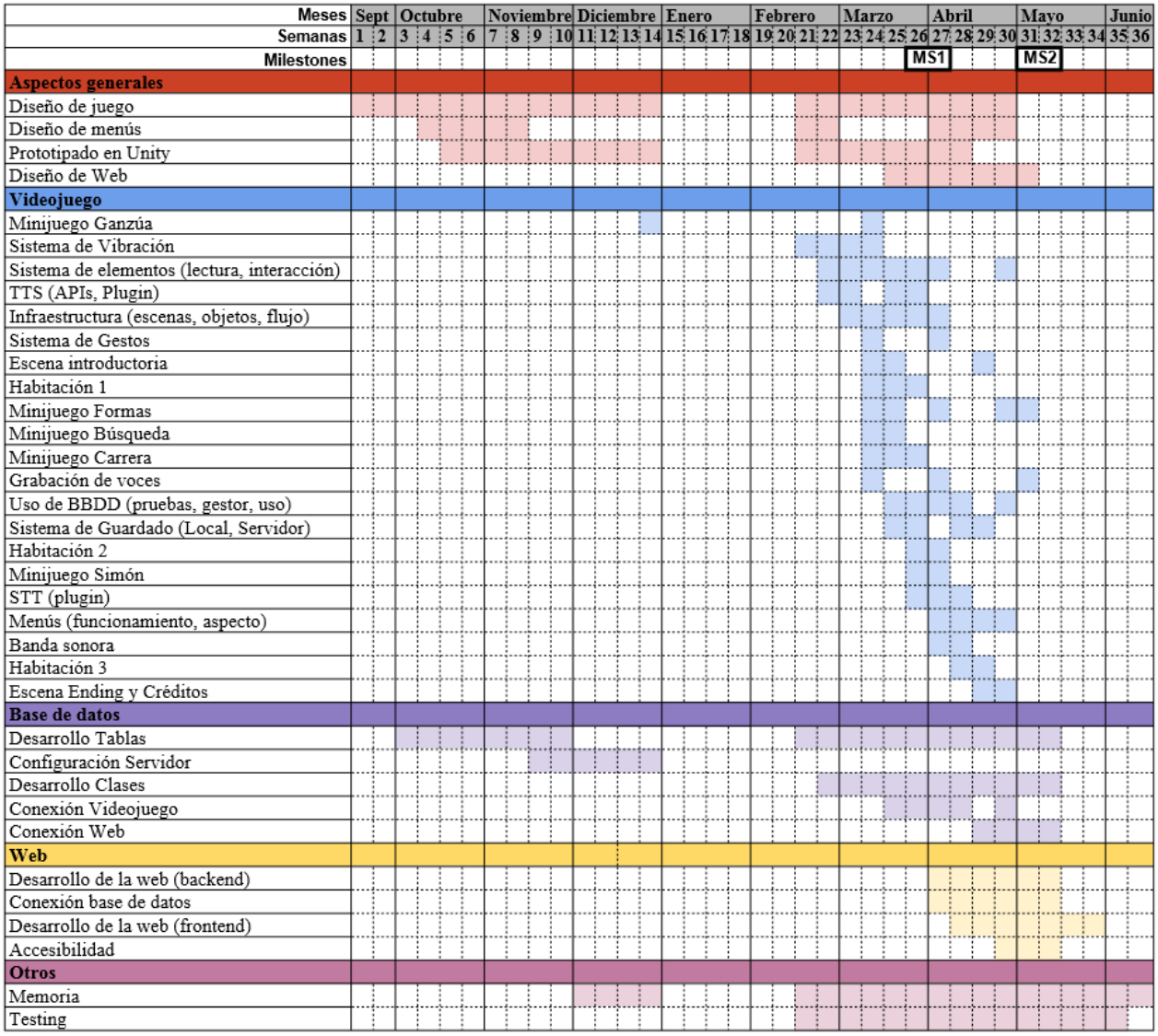


Figura 4 - Diagrama de Gantt

5. Tecnología utilizada

5.1. Unity (C#)

Para el desarrollo del videojuego se optó por el motor **Unity** (Unity Technologies, 2005), de uso gratuito bajo ciertas condiciones entre las que se encuentra el académico. Esta decisión fue tomada por varios motivos. En primer lugar, por la facilidad que proporciona a la hora de exportar el producto a distintas plataformas, ya que permite crear ejecutables para, entre otros, **Microsoft Windows** (Microsoft, 1985), **Android** (Google, 2008) y **iOS** (Apple Inc., 2007). La otra causa principal era agilizar el desarrollo de la base del videojuego, ya que los estudiantes del GDV estaban habituados a usar este software. Esto permitiría dedicar menos esfuerzos a la creación de elementos básicos del videojuego y hacer uso de las funcionalidades que proporciona el motor de base.

Así, sería posible centrarse en la implementación de la accesibilidad de la aplicación ya que, como se comprobaría más tarde, el soporte de **Unity** en ese campo es prácticamente nulo. También permitiría a los estudiantes del GII tomar contacto con este motor, de uso muy habitual en el desarrollo de videojuegos, tanto en el campo académico como en el profesional. El código desarrollado para **Unity** se ha organizado en componentes o *scripts*, escritos en el lenguaje **C#** (Microsoft, 2000).

5.2. Plataforma destino de la aplicación (Android)

Durante las primeras reuniones, los directores del TFG hicieron hincapié en que la aplicación desarrollada estuviera disponible en **iOS**. Ya que de un tiempo a esta parte muchas personas ciegas optaban por estos dispositivos gracias a sus opciones de accesibilidad.

Ninguno de los estudiantes participantes había trabajado nunca en el desarrollo de software para esta plataforma, lo que planteaba una notable problemática, unida al hecho de que no disponían de dispositivos de este tipo para hacer pruebas durante el desarrollo. En un principio se solventó el problema del desconocimiento de la plataforma optando por hacer el desarrollo en **Unity**, aunque se seguía sin poder hacer pruebas en **iOS** de forma fiable.

Más tarde se descubrió que las diferencias entre ambas plataformas eran mayores de las esperadas, lo cual se explica posteriormente en la sección de Desarrollo técnico, y que para solventarlas se necesitaban unos conocimientos y familiaridad con **iOS** de las que se carecía.

Finalmente, estos problemas sumados al *feedback* recibido en la visita a la **ONCE** llevaron a optar por un desarrollo exclusivo para **Android**. En esta reunión se hizo saber que esa preferencia generalizada por **iOS** ya no se daba, al haberse puesto los dispositivos **Android** a la par en la materia de accesibilidad con el desarrollo de tecnologías como el lector de pantalla **TalkBack** (Google, 2011), de uso gratuito. Debido a orientar el desarrollo a **Android**, más adelante se haría uso de ciertas tecnologías específicas de esta plataforma: los sistemas de vibración, síntesis de voz y síntesis de texto en base a voz.

A lo largo del desarrollo las pruebas de la aplicación se realizaron sobre los propios dispositivos móviles de los estudiantes participantes.

5.3. Android Studio (Java)

Puesto que el sistema operativo **Android** está en lenguaje **Java** (Sun Microsystems, 1995) y en **Unity** se utiliza **C#**, para hacer uso de ciertos servicios de **Android** fue necesario el desarrollo de *plugins*. Estos *plugins* se crearon en el programa **Android Studio** (Google, 2014), plataforma oficial de desarrollo para **Android**. Este programa ya había sido usado anteriormente los estudiantes de GDV, aunque no para crear *plugins*. Tras una investigación conjunta sobre cómo hacerlo, la realización quedó en manos de estudiantes del GII, que adquirieron así experiencia en el manejo de esta plataforma, de uso muy común en el desarrollo para **Android**.

5.4. Audio

Se ha utilizado el programa de edición de audio **Audacity** (Audacity Team, 2000) para el tratamiento general de sonido del juego. Es un programa gratuito de uso público con herramientas muy útiles y, por lo general, sencillas para el usuario.
Además, se ha utilizado el software **Reaper** (Cockos, 2006) para la generación de la banda sonora y la edición de determinados audios.

5.5. Servidor en la nube

Para el almacenamiento y procesado en la nube se ha utilizado el paquete **LAMP** (**Linux Apache MySQL PHP**) (Varios, 1995) porque tiene todas las características necesarias y de una forma muy cómoda. Se procede al análisis uno a uno:

Linux (Torvalds, 1991) (aunque es solo el núcleo normalmente se asocia a **GNU/Linux** (Stallman, 1983)) es el sistema operativo que se eligió para alojar el servidor porque se estaba familiarizado al haberse utilizado en ambos grados. Además, es de lo más estable que se puede encontrar y su cuota de servidores en el mundo lo respalda.

Apache (Apache Software Foundation, 1995) es un servidor web http de código abierto para prácticamente todas las plataformas. Es muy popular en la comunidad ya que además es modular y extensible, llegando a una cuota del 65,31% del total de dominios en España (Glasdom, 2017). Es desarrollado por una comunidad de usuarios supervisados por **Apache Software Foundation**.

MySQL (MySQL AB, Sun Microsystems, Oracle Corporation, 1995) es probablemente el sistema de gestión de bases de datos relacionales más utilizado del mundo que utiliza **SQL** (IBM, 1974) como lenguaje de consulta. Se ha elegido una base de datos relacional en vez de las no relacionales como pueden ser **MongoDB** (MongoDB Inc., 2009) porque los datos que se manejan se relacionan entre sí, su volumen crecería de forma paulatina y además se tiene una estructura bien definida de los mismos.

PHP (Lerdorf, 1995) (**PHP**: *Hypertext Preprocesor*, Preprocesador de hipertexto) es un lenguaje de programación del lado del servidor. Se ha elegido porque tiene una muy buena integración con las bases de datos a la hora de hacer la conexión y las consultas. Además, se implementa de forma muy sencilla en los documentos **HTML** para generar las páginas web.

También cabe mencionar que el protocolo para enviar las peticiones es **HTTP** (Hypertext Transfer Protocol, Protocolo de transferencia de hipertexto) (World Wide Web Consortium and Internet Engineering Task Force, 1991) a través de los métodos *GET* y *POST*.

5.6. Página web

Respecto a la página web las tecnologías que se han utilizado se pueden dividir en dos grandes grupos:

- **Front-end**. Se ha utilizado **HTML** (WHATWG, 1993) como lenguaje de marcado para el contenido de la página web, **CSS** (W3C, 1996) para su presentación y **JavaScript** (Netscape Communications, 1995) para su comportamiento. Además, cabe mencionar que se ha utilizado **AJAX** (Ullman, 2007) para una comunicación transparente para el usuario con el servidor para recuperar información de la base de datos en determinadas páginas de la web.

- **Back-end**. Se ha utilizado PHP para el procesamiento de las peticiones y la conexión con la base de datos y SQL para la implementación de esta última.

5.7. GitHub

El trabajo se realizó sobre un repositorio en **GitHub** (Github Inc., 2007), disponible en: https://github.com/artuyero/Videojuego-Narrativo

5.8. Microsoft Teams

Se utilizó **Microsoft Teams** (Microsoft, 2017) como herramienta organizativa en las tareas, para almacenar documentos relevantes y anexos y para elaborar la memoria en **Microsoft Word** (Microsoft, 1983).

5.9. Reuniones telemáticas

Las reuniones periódicas realizadas se desarrollaban utilizando aplicaciones que permitieran comunicación por texto, audio y video. Para las reuniones entre alumnos, más casuales, se usó **Discord** (Discord Inc, 2015), mientras que en las reuniones periódicas con los tutores se utilizó **Skype** (Skype Technologies, 2003). Las reuniones pasaron a ser telemáticas en su totalidad durante el periodo de aislamiento por el COVID-19.

6. Desarrollo técnico

6.1. Diseñando la accesibilidad de la aplicación. Lectores de pantalla

Actualmente lo más habitual al desarrollar aplicaciones con accesibilidad para personas ciegas es hacer uso de los lectores de pantalla. Estos sistemas se ejecutan a nivel de dispositivo y capturan el *input* del usuario, traduciéndolo en acciones que envía al sistema en cuestión. De esta forma, facilitan la navegación por los elementos en pantalla en base a unos gestos simples. Existen muchos programas de este tipo, tanto en ordenadores como en teléfonos móviles. Algunos ejemplos son **JAWS** (Freedom Scientific, 1995) para sobremesas, **VoiceOver** en dispositivos **iOS** y **TalkBack** en dispositivos **Android**.

En el campo de las aplicaciones móviles, estos gestos suelen ser deslizamientos rápidos y pulsaciones dobles en la pantalla. Con los deslizamientos laterales se avanza o retrocede en la lista de elementos mostrados (esta lista no suele ser cíclica, para que principio y final estén bien marcados), la doble pulsación suele equivaler a pulsar sobre el elemento seleccionado actualmente. Los distintos programas tienen también otros gestos más específicos, formados a veces por combinaciones de gestos, por ejemplo, en **TalkBack** deslizar hacia arriba y luego a la izquierda rápidamente lleva al primer elemento de la lista. Esta información se encuentra disponible habitualmente en páginas web de las empresas desarrolladoras (Google, 2020). También son de gran ayuda documentos orientativos creados por terceros basados en la documentación oficial (Interactive Accesibility Inc., 2019).

Cuando se optó por hacer el desarrollo en **Unity** los directores del TFG indicaron la importancia de estos programas, y que se debía investigar si este motor proporcionaba compatibilidad con alguno de estos.

Efectivamente, después de investigar, se descubrió que **Unity** no proporcionaba soporte para ninguna de estas herramientas de forma oficial, pese a que miembros de su equipo de desarrollo habían mostrado en ocasiones interés por implementar funciones en este campo (Hammilton, 2019) (Llu, 2019). Se observó que las únicas herramientas que daban soporte eran desarrolladas por terceros, que las publicaban en la **Asset Store** (Unity Technologies, 2020) de **Unity**, tienda en la que se permite publicar para su venta elementos creados por particulares.

La cosa no acababa ahí, sino que, al no dar soporte al lector de pantalla, ejecutar la aplicación con uno activo causaría comportamientos no esperados que dificultarían o imposibilitarían su uso normal. Esto se debe al funcionamiento del lector, que, al capturar el *input* para transmitirlo traducido a la aplicación, impide que esta lo reciba de forma directa, de modo que en aplicaciones no compatibles no se recibe ningún *input*. Por esto, se requiere que al lanzar la aplicación se indique que se debe apagar el lector de pantalla.

El juego había sido diseñado desde un principio para ser usado con un lector de pantalla o alguna herramienta similar. Básicamente se trataba de una variación en el género de las aventuras gráficas, con elementos sacados de juegos de puzles basados en avanzar de sala en sala y añadiendo minijuegos adaptados para personas ciegas. Las secciones de minijuegos tendrían controles propios, que se explicarían en cada uno, mientras que las secciones en habitaciones, que estaban pensadas para usarse con un lector, consistían en inspeccionar elementos en un escenario e interactuar con ellos

Mientras se pensaba que elección tomar frente a este problema, se buscaron antecedentes de desarrolladores que se habían visto en situaciones similares, se encontró el caso de **FREEQ**, desarrollado por **Psychic Bunny** (Caso de estudio: FREEQ). (Hughes, 2013)

Tras conocer este caso y los buenos resultados obtenidos con el enfoque de imitar un lector de pantalla, se llegó a la conclusión de que encajaría en nuestra aplicación. Finalmente, se optó por seguir su ejemplo e implementar dentro de Unity los sistemas y funcionalidades necesarias para imitar un lector de pantalla, de modo que en las secciones que lo necesitan un usuario acostumbrado a su uso pudiera navegarlos de forma intuitiva y fácil.

6.2. Sistema de gestos

Tras optarse por un desarrollo para **Android** en **Unity**, y sabiendo que no se proporcionaba soporte directo para la gestión de movimientos en **Android**, se decidió implementar un sistema propio que imitase el funcionamiento de un lector de pantalla convencional. Se tomó como referente **TalkBack**, el lector de pantalla incluido de base en dispositivos **Android**. Por lo tanto, la recogida de *input* de la aplicación intenta asemejarse lo máximo posible a la del **TalkBack**, ofreciendo un sistema de gestos y acciones similares cuando es viable.

Por un lado, se utiliza un sistema de selección de objetos similar al del **TalkBack**, con movimientos laterales para desplazarse entre opciones (Figura 5 - Deslizamiento derecha) y doble pulsación para la selección (Figura 6 - Doble Pulsación).

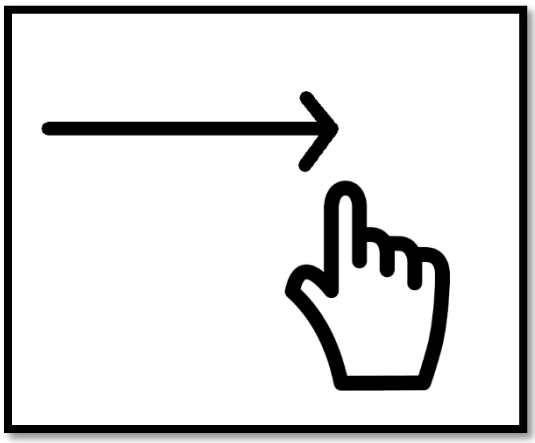


Figura 5 - Deslizamiento derecha

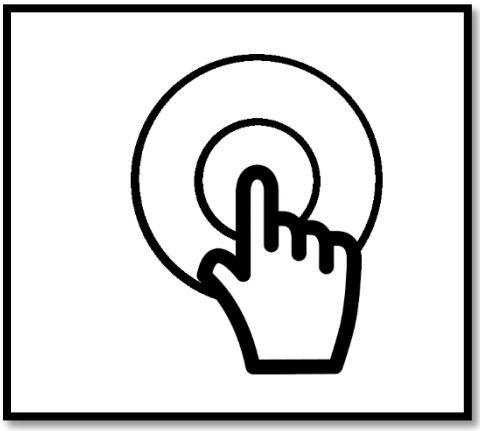


Figura 6 - Doble Pulsación

Por otro lado, además de los deslizamientos en cuatro direcciones y la doble pulsación, eran necesarios más gestos con los que se controlarían distintas partes del juego. Así, el sistema de gestos también es capaz de recoger pulsaciones largas y movimientos circulares.

En el caso del movimiento circular (Figura 7 - Movimiento Circular), la detección propia de este sistema es superficial; esto significa que realmente no otorga control sobre movimientos circulares, sino que proporciona varios datos para su posterior recogida y tratamiento en las escenas que lo utilizan (en este caso Ganzúa), como son la dirección actual del movimiento y la posición en pantalla.

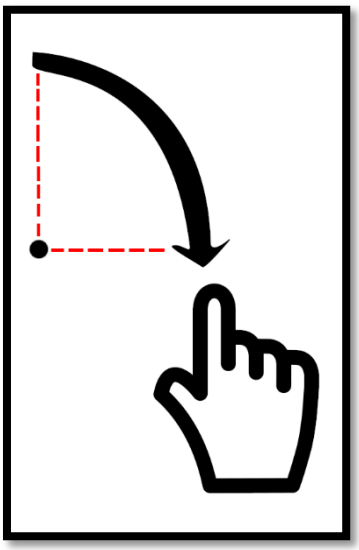


Figura 7 - Movimiento Circular

Sobre el gesto de Mantener Pulsado (Figura 8 - Mantener pulsado), en principio iba a tener un uso más amplio en la aplicación. Este se iba a utilizar como gesto de ayuda; es decir, si dejaras el dedo pulsado quieto durante 3 segundos, surgiría un mensaje de ayuda actual sobre la situación del juego. Esto se acabó descartando, por lo que el gesto ha acabado implementado, pero sin uso real en el juego.

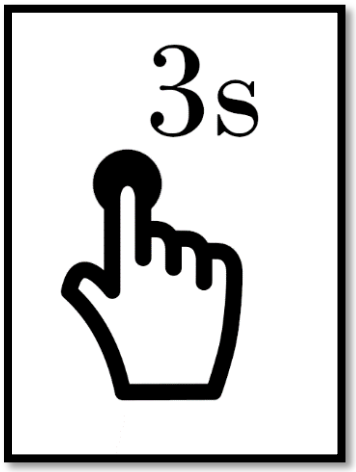


Figura 8 - Mantener pulsado

El sistema de recogida de *input* del usuario esta desarrollado en un script de **C#** implementado con funciones propias de **Unity**. El funcionamiento de esta clase se basa en el tratamiento del *input* y la devolución del movimiento actual realizado a quien lo pida desde otra clase.

De este modo, si alguien llama a esta clase recibiría en un enumerado lo que está sucediendo en escena, sea del tipo que sea. Además, también se puede obtener información extra, como la posición y la dirección del movimiento actual, en caso de tenerla.

La clase obtiene los gestos con la detección y tratamiento de las posiciones del dedo en pantalla. Así, los deslizamientos se detectan obteniendo el lugar inicial de pulsación y hallando la dirección en la que acaba el gesto.
En el caso de la doble pulsación, el *script* posee un contador que se reinicia décimas de segundo tras una pulsación normal, por lo que si se realiza otro antes del reinicio se cuenta como tal. El resto de los gestos derivan de estos tratamientos.

6.3. Imitando un lector de pantalla. Implementación.

Después de decidir que se seguiría el enfoque de imitar **TalkBack**, tocaba realizar su implementación. En un principio se realizaron pruebas con prototipos simples en **Unity**. En estas, se probaba a trasladarse entre los objetos en escena con los deslizamientos detectados por el sistema de gestos y se reproducía el audio correspondiente al objeto al que se desplazaba el foco al haber un cambio (todo esto siguiendo el ejemplo de **TalkBack**). Se obtuvieron buenos resultados con esta prueba, a falta de pulir algunos gestos y convenios más concretos como que el desplazamiento no fuera cíclico.

Se decidió entonces realizar una implementación siguiendo estos mismos principios pero que proporcionara más flexibilidad a la hora de usar y montar estas listas de elementos, además de incluir otras funcionalidades que estos prototipos no ofrecían, como la interacción con los elementos.

Para empezar, se optó por diferenciar claramente los elementos de las listas y las propias listas que los contienen en sendos componentes: *SRElement* y *SRList*.

SRElement sería el componente que se añadiría a cada elemento que debiera ser leído por el lector de pantalla. Por lo tanto, contenía etiquetas identificativas, en texto y audio, que debían reproducirse cuando pasara a estar en el foco del lector. Para la interacción con elementos se necesitaba un enfoque que permitiera llamar al método encargado de realizar su acción de forma genérica a nivel de elemento, aunque los diversos elementos hubieran de tener distintas implementaciones de este.

Para esto se decidió utilizar herencia y polimorfismo. Los comportamientos realizados al interactuar quedarían encapsulados en componentes que heredaran de la clase *Actable* e implementaran su método *Act*, añadiendo además todas las demás funciones que necesitaran. De esta forma a cada *SRElement* se le puede asignar un componente que hereda de *Actable* y cuando se interactúa con él, se llama la implementación concreta de *Act* de ese elemento. Los contenidos de cada *SRElement* y su jerarquía con *SRList* se observa en la Figura 9 - Estructura *SRList – SRElements*.

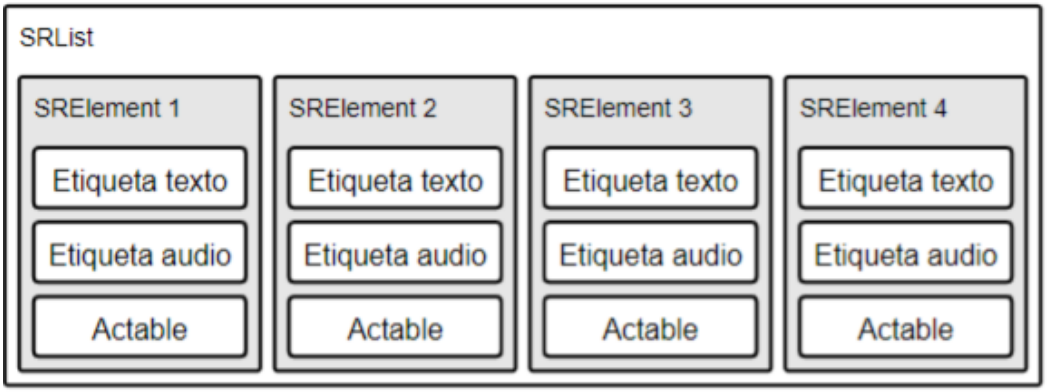


Figura 9 - Estructura *SRList – SRElements*

SRList estaría por encima de *SRElement*, conteniendo cada instancia una lista en la que se organizan los elementos que contiene. Este componente contiene por lo tanto dicha lista de *SRElements* y un índice que señala el foco actual. Este foco es el elemento seleccionado actualmente (por lo que su etiqueta es la última que se ha leído) y con el que se interactuará al hacer una doble pulsación. Aparte de esto, básicamente se encarga de implementar las funciones a través de las cuales se interactúa con esta lista (avanzar y retroceder en la lista, leer el foco, interactuar con el elemento en el foco, insertar y eliminar elementos, buscar elementos…) y respetando el orden de elementos para evitar fricciones y confusión en el uso. En la Figura 10 - Ejemplo *SRList* Inventario, se puede observar un ejemplo de una *SRList* con sus *SRElements*.

Figura 10 - Ejemplo *SRList* Inventario

Una peculiaridad es el comportamiento en los extremos. Para los usuarios videntes los más natural sería hacerlas toroidales, de modo que avanzar en el último elemento lleva al primero y retroceder en el primero lleva al último. Este no es el caso en la mayoría de los lectores de pantalla, sino que no se puede ir más allá de ambos extremos de las listas. La principal causa es que así es posible avanzar o retroceder sucesivamente muy rápido para ir a los elementos de los extremos y orientarse en la lista (aun así, varios lectores tienen también gestos específicos para estas funciones, útiles al manejar listas con gran cantidad de elementos). En un principio se optó por repetir la etiqueta del elemento extremo al darse uno de estos casos. Más tarde esta implementación se cambiaría por otra que reprodujera un sonido específico (similar a un golpe a una pared) al intentar avanzar desde los extremos, siguiendo el ejemplo de **TalkBack**.

Por encima de estas dos clases quedaría otra, *SRManager*, que sería la que gestiona la SRList activa en el momento y actuar sobre ella en base a los gestos recibidos del sistema de gestos. Esta clase se implementó siguiendo el patrón *Singleton*, para hacerla accesible al resto de elementos en escena, especialmente a los *SRElements*. Aunque en un principio se planteaba que tuviera esta funcionalidad y poco más, acabo cambiando y creciendo bastante en base a requisitos que fueron encontrándose a medida que se desarrollaba el juego. Por ejemplo, sería esta clase la que contuviera la fuente de audio (*AudioSource*) usada para reproducir las etiquetas de los elementos, evitando así que se superpongan entre ellos. El enfoque de lista y elemento activos se muestra en la Figura 11 - *SRManager* con elemento activo.

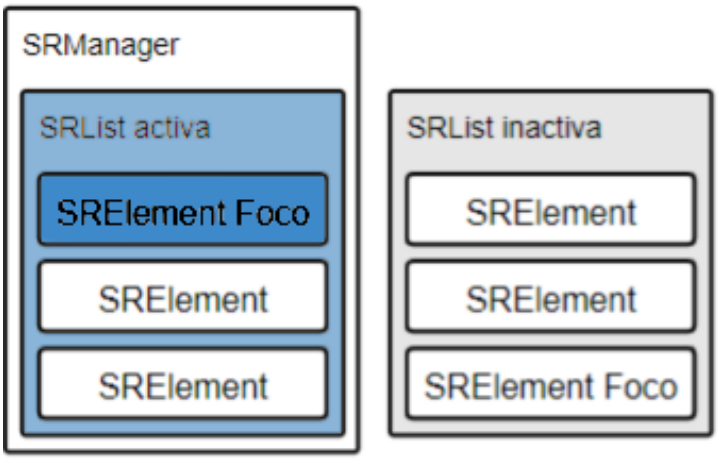


Figura 11 - *SRManager* con elemento activo

En un principio se adoptó un enfoque de listas conectadas, ilustrado en la Figura 12 - Enfoque cambio entre listas descartado, de modo que cada lista tenía una lista previa a la que se volvía al deslizar hacia abajo. De esta forma, había una lista intermedia entre escenario e inventario formada por estos dos botones, que enviaban a sendas listas. Para cambiar de uno a otro con este enfoque había que deslizar hacia abajo para volver a la lista de selección y luego seleccionar la lista deseada dentro de esta. Aunque este enfoque era funcional, se acabó descartando, por considerarlo innecesariamente enrevesado, se optó entonces por alternar entre inventario y escena directamente al deslizar hacia abajo, sin paso intermedio.

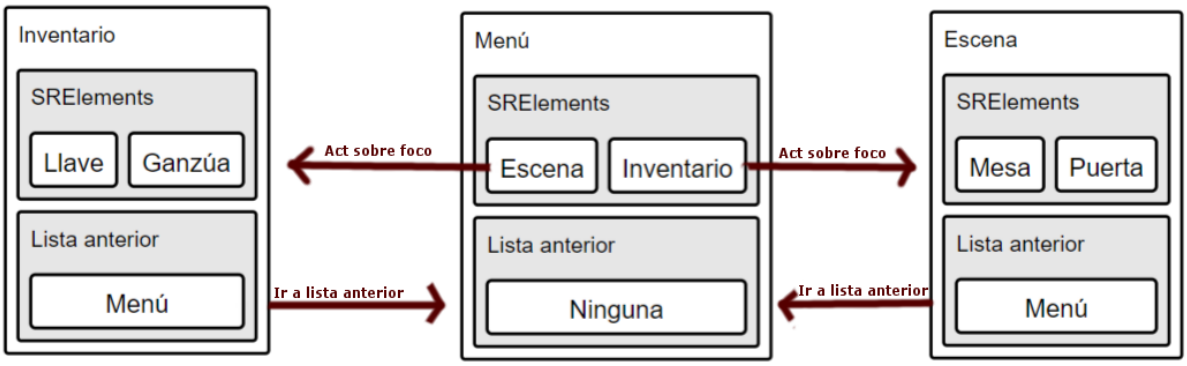


Figura 12 - Enfoque cambio entre listas descartado

A partir de este momento se decidió que *SRManager* tuviera sendas listas para inventario y escena, ya que esto también permitía hacerlas más accesibles al resto de elementos. Fue a partir de este punto que se empezó a incluir en esta clase funcionalidades más específicas del juego, como las llamadas a carga de niveles y el bloqueo del *input* en ciertas ocasiones entre otros. Un ejemplo de esto se puede encontrar en la Figura 13 - *SRManager* con enfoque específico para el juego.

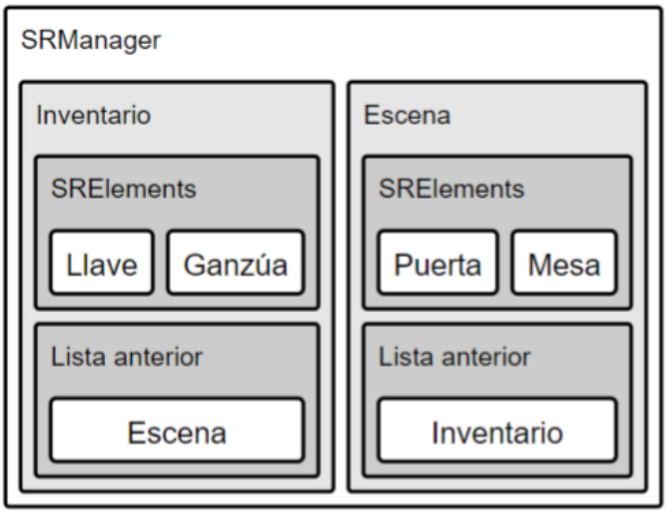


Figura 13 - *SRManager* con enfoque específico para el juego

Esto suponía ciertos problemas, y es que estas funcionalidades no debían darse en escenas que no fueran las de exploración, como menús y minijuegos. En un principio, para aislar estas funcionalidades se planteó el uso de herencia para crear distintas implementaciones de *SRManager* para estos distintos casos. Al usar el patrón *Singleton*, este enfoque resultaba poco prometedor. Finalmente se optó por un enfoque de tipos, de modo que el *SRManager* tuviera un enumerador que indicara en qué tipo de escena se encontraba. De esta forma la carga de objetos solo se daba si estaba marcado como habitación y los gestos que se recogían también podían variar entre distintos tipos. Así se consiguió conciliar el uso de esta componente en todas las escenas del juego en que se usara nuestra implementación del lector de pantalla.

A lo largo del desarrollo de este sistema de lectura de pantalla hubo varias refactorizaciones del código, especialmente cuando se incluyó el cargado de habitaciones y las funcionalidades online (estas funcionalidades están implementadas en el *GameManager*). A lo largo de estos cambios también se reforzó la conectividad entre los elementos, de modo que cada *SRElement* siempre sabía a qué lista pertenecía (se le asigna al insertarlo), y cada *Actable* supiera a que *SRElement* está asignado (asignado en la inicialización de la clase base).

En este proceso sufrió especialmente la clase *SRManager*, que pasó de una interfaz simple para el manejo de las listas en base a los gestos a contener gran parte de la gestión de las habitaciones. Aun con consciencia de las fallas que llevaron consigo estas refactorizaciones y expansiones, se consiguieron los objetivos planteados a la hora de imitar un lector de pantalla y hacerlo mediante la creación de un sistema fácil de usar desde editor y con capacidad de extensión.

6.4. Sistema de vibración

A la hora de crear contenido para personas ciegas la intención era aprovechar todo el *feedback* no visual que se pudiera proporcionar, para ello desde un principio el foco se puso en las dos que proporcionan la mayoría de los dispositivos móviles: sonido y vibración. Con el sonido no se esperaba demasiado conflicto, ya que suele darse por estándares bastante comunes y el motor utilizado (**Unity**) proporcionaba una implementación sólida y compatible tanto con **Android** como con **iOS**. Sin embargo, a la hora de utilizar la vibración se dio con más problemas. Para empezar, la funcionalidad que proporcionaba **Unity** resultaba insuficiente (solo permitía realizar una vibración de duración predefinida, sin permitir modificar duración, patrón, potencia, hacerla cíclica etc.). Ante esto se decidió que había que crear un sistema que proporcionara más funcionalidades y opciones a la hora de hacer vibrar al dispositivo.

La idea era tener una implementación común que según el tipo de dispositivo en el que estuviera llamara a la API Android o de iOS. A la hora de hacer esto se descubrió por qué esta funcionalidad a primera vista tan obvia no estaba disponible en Unity. Y es que las vibraciones en iOS y Android tenían muy poco en común. En esta investigación resultó de gran ayuda un proyecto similar alojado en GitHub. (Freslon, 2020)

Mientras que Android permite hacer vibraciones con la duración que se quiera y elaborar patrones de vibración con periodos de activación y pausa (que además pueden reproducirse de forma cíclica), en iOS solo se permite realizar vibraciones de entre unos pocos tipos predefinidos. Esto hizo evidente que conseguir vibraciones similares en ambas plataformas sería muy complicado o incluso imposible. De haberse tratado de vibraciones de menor complejidad o importancia, para notificar alertas o puntuar ciertos momentos, hubieran valido los tipos predefinidos. Pero al necesitar que ciertas partes del juego se basen enteramente en la vibración, estos se quedaban cortos.

Esto, junto a la nula experiencia de los miembros del equipo en el desarrollo para iOS y la carencia de dispositivos de este tipo para testear durante el desarrollo, fueron las razones porque las que se optó por un desarrollo en exclusivo para Android.

Una vez tomada esta decisión lo que restaba era hacer una implementación que permitiera llamar a la clase *Vibrator* (Android Inc, 2020) de Android (en Java) desde Unity (en C#). Para esto fueron muy útiles tutoriales y recursos *on-line* (Robert, 2019) (era la primera vez ningún miembro del equipo usaba funcionalidades de Android de esta forma). El problema que se encontró es que, a primera vista, no se podía modificar las potencias de las vibraciones. Esto era un requisito importante para la funcionalidad, ya que se habían creado diseños de minijuegos en los que uno debía guiarse por la potencia de la vibración en distintas partes de la pantalla. Por lo tanto, se tuvo que buscar alguna alternativa.

Una de las primeras ideas que se plantearon era falsear un control de la potencia intercalando periodos de vibración y pausa muy breves (milisegundos). Sin embargo, no se consiguió la sensación que se buscaba en parte porque pausar y encender tan rápido no funciona bien a nivel de hardware (el rotor toma más tiempo en encenderse y apagarse). Pero experimentando con esta implementación se conceptualizó un enfoque distinto, realizable con las opciones que proporcionaba Android. Consistía en imitar los patrones de un sónar, con períodos de actividad y pausa más largos y acentuados, de modo que las pausas entre vibraciones se acortan cuanto más te acercas al punto que tendría mayor intensidad, donde pasa a ser una vibración constante. Este es el enfoque por el que se optó al final para las funcionalidades que usaban potencia en un principio.

Más tarde se observó que en versiones modernas de **Android** (API 26 *Oreo*, o superior) se había añadido la clase *VibrationEffect* (Android Inc, 2020) con la que se podía modular la potencia. Se sopesó si valía la pena añadirla a costa de perder compatibilidad con dispositivos más antiguos, pero se eligió probarla primero antes de tomar la decisión. Se añadió esta funcionalidad a nuestro código y se probó a usar esta modulación de potencia en lugar del enfoque sónar. Los resultados llevaron a la conclusión de que los cambios en potencia que permitía apenas eran perceptibles, no lo bastante distintivos y claros como para basar los minijuegos en ellos. Por esto se optó por seguir con el enfoque sónar y no utilizar *VibrationEffect*.

6.5. Síntesis de voz a partir de texto. TextToSpeech (TTS)

Tanto para poder probar la lectura de elementos en pantalla y la reproducción de instrucciones al jugador antes de disponer de los audios grabados, como para la reproducción de la información variante (números, nombres de usuario, etc.) se concluyó que se necesitaba alguna forma de generar sonido hablado en base a texto (TTS). Lo primero que se hizo fue comprobar si **Unity** aportaba esta funcionalidad y, aunque había ciertos *plugins* y extensiones de pago en la **AssetStore de Unity**, el motor como tal no daba soporte a esta función.

La primera alternativa tomada fue utilizar una **API** externa a la cuál hacer llamadas a través de internet para solicitarle los audios. Para ello, se buscaron alternativas en **RapidApi** (RapidAPI, 2019), un *marketplace* de **APIs** bastante conocido por tener muchas opciones gratuitas. Allí se encontró la API de **Voicerss** (Voircerss Org, 2018), que tenía bastante popularidad y una tasa de acierto muy alta, que sin duda se necesitaba para que el sistema funcionara perfectamente.

En un principio, se hizo una prueba creando una página web que tenía un campo de texto que, al ser rellenado, se transformaba a voz. Para ello se utilizó **JavaScript**, ya que era uno de los lenguajes en el que más cómodos se encontraban los estudiantes para hacer una llamada a una API. Al ver que funcionaba se decidió pasar a **C#**, lenguaje utilizado en **Unity**. Se creó una aplicación de consola en **C#** que tenía la misma funcionalidad que la página web y se observó que el comportamiento era el esperado. Después se integró en un proyecto de **Unity**, que solo tenía una interfaz con un campo de texto, igual que con la web.

Se logró tener un buen funcionamiento en **Unity**, pero ciertos inconvenientes fueron encontrados. Esta API en concreto tenía un límite de usos si no se abonaba una tarifa, lo que impedía ser usada para un juego mantenido en el tiempo, ya que las llamadas dejarían de funcionar tras un número de peticiones.

También cabe mencionar que la calidad del audio era bastante baja. Al usar esto también se observó el problema de pedir y recibir los audios a través de internet, y es que además del consumo de datos móviles que se podía llegar a tolerar, implicaba un retraso notable en la reproducción del audio.

La decisión de usar una API externa se tomó cuando aún no se había descartado el desarrollo para iOS, porque esto permitía que fuera independiente de la plataforma. Pero al descartar el desarrollo en iOS y centrarse este en Android se observó que este proporcionaba la funcionalidad del TTS. Para usarla serían útiles los conocimientos adquiridos a la hora de utilizar el sistema de vibración de Android desde Unity.

Sin embargo, a la hora de implementarlo se descubrió que no era suficiente con hacerlo de la misma forma que la vibración, sino que se necesitaría una capa intermedia en forma de *plugin*. Mientras que la vibración se considera un servicio básico en el entorno Android y por lo tanto se podía obtenerla directamente con una petición desde código C#, el TTS no lo es, por lo que no se podía acceder a él de esta forma. Además, para usarlo es necesario implementar ciertos métodos sobrescribiendo los de la clase base. Visto esto, se observó que se necesitaba afrontar el problema desde otro enfoque. Después de investigar y encontrar casos de desarrolladores que se hubieran visto en situaciones similares, se concluyó que la mejor forma de hacer esto en esta situación era desarrollar un *plugin* en Java para usarlo en Unity.

Tras buscar información sobre cómo es habitual enfocar el desarrollo de un *plugin* de estas características, se concluyó que lo más conveniente sería hacerlo a través de Android Studio (el entorno de desarrollo oficial para Android, propiedad de Google y de uso libre para desarrolladores, disponible en su web) (Android Inc, 2020). Los estudiantes del GDV ya tenían cierta familiaridad con el entorno, pero ninguno había trabajado en la creación de *plugins* anteriormente. Se aprovechó la oportunidad para que los estudiantes del GII se familiarizaran con este programa.

Sobre el desarrollo en Android, se creó una clase en Java que llamaba a la clase *TextToSpeech* (Android Inc, 2020) de Android y se experimentó con sus posibilidades. Se le añadió el atributo idioma, para poder elegir entre español o inglés, además de un sistema de opciones que controlaba la velocidad de la voz del audio en tres modos distintos: rápido, normal y lento. Dado el poco uso que se hace del TTS en la aplicación final, varias de estas opciones quedaron en desuso más allá de la configuración por parte de los desarrolladores.

Una vez hecho el *plugin*, este se compiló para obtener una versión que se pudiera utilizar en **Unity**. Para utilizarlo desde este motor era necesario otro *script* que actúa como intermediario y gestiona las llamadas al *plugin*. Estas llamadas se hacen de manera similar a las de la vibración (actuando sobre *AndroidObjects* genéricos y haciendo las llamadas con *Call()*, que recibe el nombre del método como argumento).

Se comprobó que el TTS nativo de **Android** tenía mayor calidad que el que se había utilizado antes: la voz se oye más nítida y con un volumen normal, las peticiones de audio toman menos tiempo y además se reproducen de forma independiente, directamente desde el dispositivo **Android**, por lo que no es necesario procesar la reproducción de los audios en la aplicación.

Esto último también podía considerarse un inconveniente, ya que así no se tenía control sobre cuándo acababa un audio, algo necesario en ciertas partes de la aplicación en las que, por ejemplo, se bloquea el *input* hasta acabar una explicación. Una posible solución a este problema sería forzar las esperas en base a la duración que tomara la reproducción, que tendría que medirse a mano, haciendo tanto el desarrollo como el código tedioso e ininteligible.

Aunque finalmente se optara por utilizar audios pregrabados en prácticamente todo el juego, se piensa que de haber tomado la decisión de reproducir todo el texto con TTS, esta implementación hubiera permitido hacerlo sin mayores problemas a excepción del ya expuesto. También cabe mencionar que este es el mismo sistema de TTS que usan los lectores de pantalla estándar en la plataforma **Android** (**Google TalkBack**), al que los usuarios ya están acostumbrados y que funciona tanto *online* como *offline*. De forma *offline* funciona haciendo uso de los idiomas instalados en el teléfono.

6.6. Reconocimiento de voz. SpeechToText (SST)

Para facilitar el acceso al videojuego, se pensó en la posibilidad de incorporar un sistema de reconocimiento de voz, y así evitar el uso del teclado. Dado que ya se tenía experiencia con el plugin de TTS se investigó sobre la posibilidad de hacer un plugin de STT. Tras la investigación, se empezó el desarrollo en **Android Studio**, entorno de desarrollo con el que ya se había trabajado para el anterior *plugin*. Al igual que con el caso de la vibración, se encontró un proyecto en **Github** con características similares, que sirvió como referencia. (Srinivas, 2018)

En el desarrollo se hizo uso de un *Intent*, una descripción abstracta de una operación que se va a realizar. En concreto se utilizó la clase *RecognizerIntent* (Android Inc, 20), que contiene los parámetros necesarios para llevar a cabo el reconocimiento de voz. Entre ellos, por ejemplo, está el lenguaje que se quiere utilizar, o el número de resultados que se quiere obtener. Estos resultados se ordenan en una lista, siendo el primero de la lista el resultado en el que el reconocedor tiene mayor confianza y el último, en el que menos confianza tiene. Para nuestro proyecto, se pensó que solo se utilizaría el primer resultado, ya que es lo suficientemente fiable como para no tener que obtener más de uno.

Una vez puestos los parámetros necesarios en *RecognizerIntent* se debe crear una variable que funcionará como código para llamar a una actividad de **Android**. Una actividad es, descrito por la página de referencia oficial de **Android** (Android Inc, 2020), "una cosa única y enfocada que el usuario puede hacer". Estas actividades se utilizan para la interacción con el usuario. En el STT, a diferencia del TTS, se debía interactuar con el usuario para recoger su voz. Una vez iniciada la actividad, se debe recoger el resultado haciendo uso del código antes mencionado. Este código se utiliza para saber que lo que devuelve la actividad ha sido lanzado por ti. Entonces, simplemente se compara el código de la petición de esa actividad con el que se ha decidido y ya se obtendría el resultado.

Sobre el envío del resultado a **Unity**, STT tenía una complicación añadida respecto de TTS. En el TTS se tenía una función a la que se llamaba y devolvía el audio deseado. En este caso, al ser una actividad, el resultado no se podía devolver hasta que la actividad devolviera el resultado. Por ello, para enviar el resultado desde **Android** a **Unity**, se hace uso de *UnityPlayer*, que nos permite interactuar con **Unity** mediante *UnitySendMessage*, donde se especifica el nombre de la función a la que se quiere devolver el resultado, y el propio resultado. Por lo tanto, se debía tener en **Unity** una función que recogiera el resultado y lo tratara.

Además, dado que se trataba de una actividad, se debía incluir esta en el *AndroidManifest*, un archivo que contiene metainformación de la aplicación de **Android**. Se debía escribir en este archivo la declaración de la actividad en formato *xml*, usando la etiqueta *<activity>.*

Por último y de igual manera que con el TTS, el STT de **Android** también funciona offline, mediante los idiomas instalados en el teléfono.

6.7. Diseño del juego

En lo referente a la temática del juego, se informó que debía tratarse de un juego fuera del ámbito educativo o instructivo, y se propuso como público objetivo personas ciegas de entre los 15-25 años, que estuvieran ya acostumbradas y se desenvolviesen con soltura en el uso de las mecánicas accesibles. Con este público dirigido se optó por una temática y una narrativa con un enfoque algo más oscuro, con ciertos toques que podrían denominarse paranormales y manteniendo un ambiente serio durante el transcurso general del juego. Esta narrativa y ambientación se explican en el Apartado narrativo.

Una vez se decidió la idea del juego narrativo con minijuegos y con la inclusión de salas de "puzle", se comenzó a investigar ideas de minijuegos que se pudiesen acoplar a las restricciones que se tenían (vibración, sonido y gestos). Finalmente, se acabaron diseñando un total de 5 minijuegos, sin contar las salas tipo puzle. Todos ellos se explican con más profundidad en sus apartados correspondientes.

Teniendo en cuenta los 5 minijuegos que estaban diseñados, se tuvo que adaptar el diseño del juego para que se pudieran dar en un flujo general lógico. Podemos observar en la Figura 14 - Esquema del flujo de salas del juego la estructura del juego, la cual se va a explicar con más detalle a continuación.

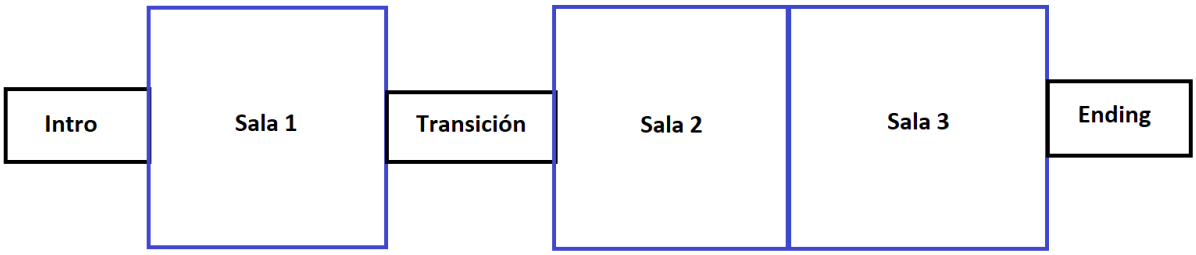


Figura 14 - Esquema del flujo de salas del juego

Siguiendo el esquema de la Figura 14 - Esquema del flujo de salas del juego, el jugador empezaría en una escena inicial de introducción. En esta introducción se le presenta al jugador una parte de la historia, y se utilizan dos mecánicas simples para introducirle en la narrativa del juego. El jugador debe avanzar manteniendo el dedo pulsado, lo cual propicia sonidos de pasos y lo acerca a la puerta. Al llegar a la puerta, deberá deslizar varias veces en cualquier dirección para arrancar unas "bandas policiales", tras lo que se abre la puerta. La estructura se puede ver en la Figura 15 - Esquema Introducción.
Aquí no se espera ningún tipo de bloqueo por parte del jugador, ya que las mecánicas son sencillas, intuitivas y se explican en el momento.

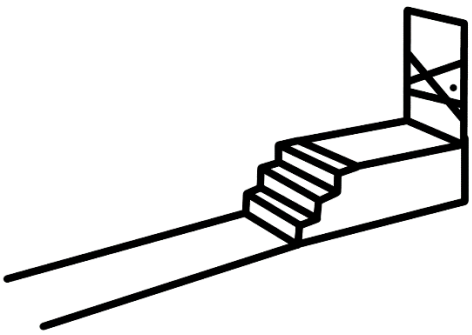


Figura 15 - Esquema Introducción

Tras terminar la introducción, el jugador pasaría a la primera sala tipo puzle, la Sala 1, de la que tiene que salir y avanzar. Aquí se encuentra el primer minijuego (Ganzúa), al que se accede tras permitirle al jugador manejarse por la sala. Esta sala, al ser la primera, trata de mostrar los controles que se van a utilizar durante el resto del juego y contiene explicaciones adicionales. El minijuego es bastante sencillo y no penaliza los fallos, por lo que no se esperan bloqueos. Un esquema de la habitación se muestra en la Figura 16 - Esquema sala 1.

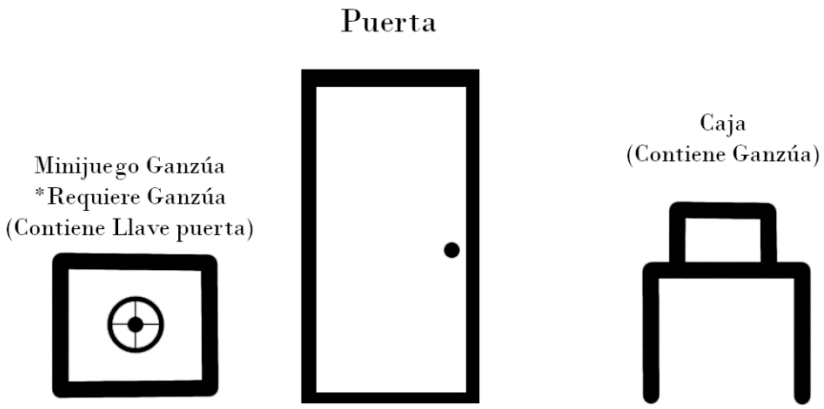


Figura 16 - Esquema sala 1

Tras completar el primer minijuego y avanzar de sala, se pasará directamente a la siguiente escena de transición, el minijuego Persecución. Este funciona como un conector entre la primera sala y la segunda, conteniendo más acción, ya que se trata de un juego de reflejos. Este minijuego se puede fallar, haciendo que se repita la escena hasta que se termine satisfactoriamente. El juego es laxo a la hora de los fallos, ya que el tiempo de reacción es muy amplio y requiere de varios *inputs* incorrectos para perder y reiniciar, por lo que tampoco se espera que resulte difícil superarlo.

Al terminar el juego Persecución, se pasa a la siguiente habitación, la Sala 2. En esta sala ya se espera conocimiento por parte del jugador en el manejo de la escena, y se compone de varios objetos con interacción y de dos de los minijuegos. El primer minijuego (Búsqueda) muestra por primera vez las mecánicas de vibración del juego. Es el más sencillo de los dos, así que se posiciona primero para aumentar la dificultad con el siguiente. Requiere de algo de concentración por parte del jugador para ser completado de forma normal. Igualmente, el juego se puede completar con prueba y error en caso de bloqueo del jugador, por lo que no debería dar muchas dificultades.

El siguiente minijuego es el más complicado del juego (Reconocimiento de formas por vibración), tratando como mecánica principal la vibración para el reconocimiento de objetos. Aquí se observa el mayor bloqueo por parte de los jugadores, a pesar de los varios cambios que se han hecho sobre él (explicados en Reconocimiento de formas por vibración y en Pruebas). Al no penalizar los errores, el juego también se puede completar con prueba y error en caso de bloqueo. El esquema de esta habitación se encuentra en la Figura 17 - Esquema sala 2.

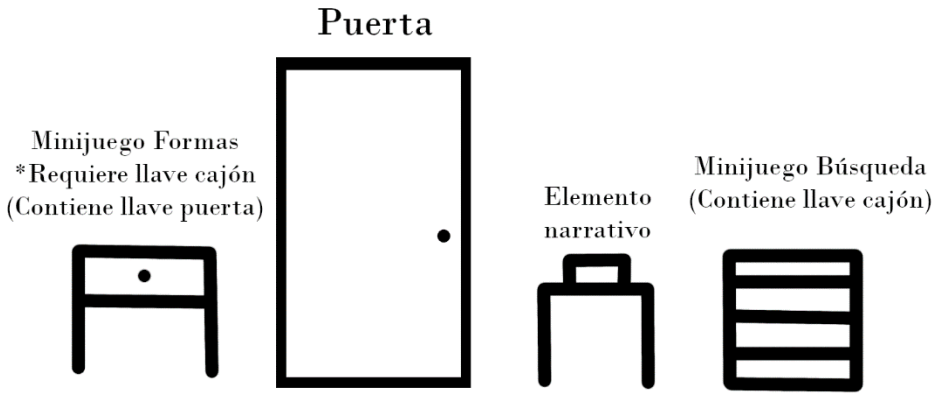


Figura 17 - Esquema sala 2

Tras terminar esta habitación, se avanzará a la siguiente y última (Sala 3), la cual no requiere de mucho manejo de objetos en sala, pero sí de inventario. Tras interactuar con los objetos que hay en sala, y tras mencionar al jugador que debería investigar el inventario, el jugador entrará en el último minijuego (Simón). Menos complicado que el minijuego anterior, es un juego memorístico simple, el cual se puede fallar, pero no se reinicia completamente utilizando un sistema de rondas. Este funciona como transición también a la escena Ending, la cual comienza al completarlo. La sala contiene un objeto con el que se debe interactuar varias veces, y posteriormente se debe usar el inventario para acceder a dicho minijuego (Figura 18 - Esquema sala 3).

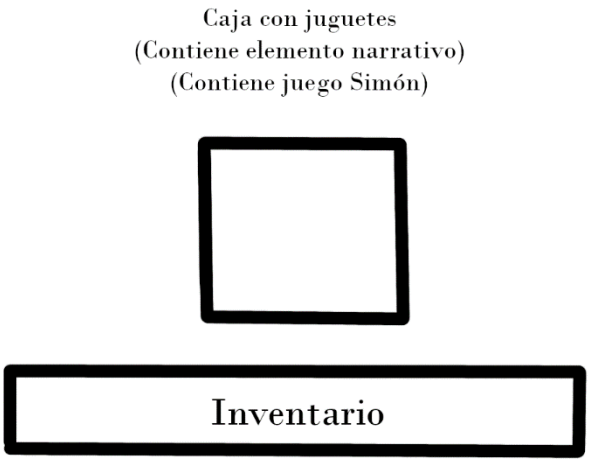


Figura 18 - Esquema sala 3

La escena Ending, por su parte, ofrece el fin a la historia del juego, en la que se descubren los últimos trazos de esta. Hace transición directa a los créditos, que son la última parte de la aplicación.

En resumen, el juego está diseñado para ofrecer al jugador una pequeña historia haciendo uso de mecánicas innovadoras, pudiendo ser completado satisfactoriamente por el jugador medio y por las personas ciegas, a las cuales va dirigido.

6.8. Apartado narrativo

Se escribió un guion base para el juego, el cual experimentó varios cambios. Este guion se encuentra en el anexo Guion base.
A partir de esa idea base, el juego fue tomando forma y cambiando para adecuarse a las ideas de mecánicas y estilo que íbamos desarrollando. Así, un pequeño resumen de este apartado sería el siguiente:

La historia comienza con un *flashback*, en el que se escucha ruido de fuego (un incendio) y unos gritos.
El protagonista se "despierta" delante de una casa y, desorientado, avanza hacia ella. Cuando sube las escaleras del porche de madera, se encuentra con unas bandas policiales en la puerta, las cuales arranca y avanza al interior de la casa.
La casa está muy oscura, por lo que investiga los objetos de la habitación palpando. En la primera habitación, con la ayuda de una ganzúa consigue abrir la siguiente puerta.
Nada más pasar la puerta, el protagonista oye la voz de su hija, y comienza a perseguirla por los pasillos, hasta que llega a otra habitación en la que deja de oírla. Consigue también abrir esta puerta, y entra en la habitación de su hija.

Esta habitación está llena de los objetos de su hija (varios chamuscados), y entre ellos, coge un juguete electrónico Simón y lo prueba. Los pitidos del Simón vuelven a desorientar al protagonista, y se empiezan a oír pitidos de máquinas de hospital. La voz de su hija va cogiendo más fuerza y, tras abrir los ojos, la escucha con claridad.

En un principio, la narrativa de la historia tenía la finalidad de provocar una situación en la que no fuera necesario el uso de la visión (esto se refleja en la oscuridad que hay en el interior de la casa), pero igualmente se siguió ofreciendo un apartado gráfico simplista ocasional para personas videntes, el cual no les aporta ventaja en el apartado mecánico.
La historia base coge inspiración de otros videojuegos de renombre, como **The Last of Us** (Naughty Dog, 2013), en el que el lazo paternofilial que se forma es lo que en gran parte sostiene la narrativa y "engancha" al jugador. En el caso de esta aplicación, con la corta duración del juego "demo", no se espera que el jugador cree fuertes lazos con los dos personajes, ya que poseen poco tiempo de diálogo y desarrollo; sin embargo, sí se quiere que no se pase por alto y se pueda ver el posible potencial narrativo si la aplicación llegara a forjarse como un juego a distribuir comercialmente.

El estilo narrativo utilizado es austero, con diálogos breves y pocas descripciones. Durante el juego, se pueden encontrar objetos narrativos con los que se puede interactuar desde el inventario, los cuales te ofrecen más información sobre la historia. De este modo, el jugador podría ir formando la historia con suposiciones, hasta que llegara al final de esta y la descubriera. Aun así, se obvian muchas cosas y se ocultan otras, dando libertad imaginativa sobre lo sucedido y propiciando que cada jugador pudiera verla desde un punto distinto o con diferentes matices.

6.9. Recursos gráficos

La aplicación, dada su naturaleza, contiene muy pocos elementos gráficos, siendo estos omitidos en los minijuegos con la finalidad de no aportar algún tipo de ventaja a los jugadores videntes. Los elementos que aparecen se limitan a los menús para aportar claridad e introducir el sistema de manejo de forma intuitiva a jugadores con visión y algunas escenas con animaciones simples, además del logo de la aplicación y los recursos utilizados en la página web.

Los recursos gráficos utilizados en los menús del juego y el logotipo del juego, además de varios recursos de la web, fueron creados en la herramienta de edición de imagen de uso libre **GIMP** (Natterer, 2020). Los recursos restantes usados fueron basados en imágenes vectorizadas de uso libre.

A nivel de juego, las animaciones que se presentan se utilizan a modo de adorno visual; es decir, no otorgan más que una representación gráfica de lo que está ocurriendo a nivel de historia. Los colores usados en la aplicación han sido el blanco, negro, naranja, rojo y morado, siendo estos distinguibles por su alto contraste en sus momentos utilizados, además de que mantienen coherencia con la página web. Igualmente, estos colores que se han desaturado y se ha añadido un efecto de aleatorización de saturación cromática. Esto se ha hecho añadiendo una máscara grisácea que actúa sobre todos los colores en escena, con la finalidad de no dañar la vista de personas con problemas de visibilidad (colores accesibles), además de evitar colores planos simples. Este efecto ha sufrido varios cambios, ya que tras realizar pruebas con distintos móviles **Android** se descubrió que el *shader* utilizado no era realmente soportado por algunas versiones específicas del sistema. Esto provocaba graves problemas de rendimiento, con lo que era prácticamente imposible jugar; por lo tanto, y aunque el *shader* sigue incluido en el proyecto, el efecto se redujo a la máscara de capa anteriormente descrita que no propiciaba problema alguno.
Durante el transcurso del juego podemos observar lo siguiente:

- **Escena introductoria e inicio mecánico:** Al empezar, la historia se presenta con una simple animación de "fuego" que sube y baja en pantalla, como una especie de *flashback* que representa el incendio que sucede en la casa. La Figura 19 – Incendio muestra una captura de esta parte.

Figura 19 – Incendio

Tras una transición de fundido en negro, se muestra en pantalla una escena relativamente clara, de color blanco, con una puerta lejos, en el fondo. Tras "caminar" unos segundos, el jugador se va acercando hacia la puerta, la cual posee unas bandas negras cruzadas. La animación de acercamiento a la puerta está sincronizada con el movimiento del jugador, haciendo también el efecto propio de

subir las escaleras del porche ficticio. Estas bandas actúan individualmente cuando se "rompen", haciendo una animación de despegado de la puerta y desvanecimiento. Posteriormente, tras quitar todas las bandas, la puerta se abre y le introduce al jugador en la oscuridad de la primera sala. La Figura 20 - Puerta de introducción muestra también una captura de esta parte.

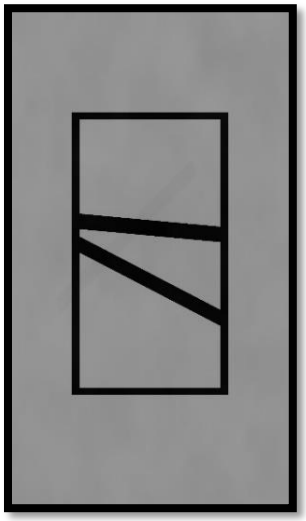


Figura 20 - Puerta de introducción

- **Escena final/ending**: Esta escena es muy simplista visualmente. Tras acabar el último minijuego, se hace una transición en negro al *ending*, el cual se compone de un fundido a blanco emulando la apertura de los ojos del jugador. Así, dos bandas negras se abren dando paso a la luz blanca del lugar en el que te encuentras. Esto también propicia claridad en la voz de la hija, la cual escuchas durante toda la transición, y da paso a la música de créditos. La Figura 21 - Apertura de ojos muestra una captura de esta parte.

Figura 21 - Apertura de ojos

Todo esto ha sido realizado con la herramienta de animación propia de **Unity**, utilizando *sprites* básicos para la creación de formas y entrelazándolo con el código necesario para su correcto funcionamiento.

6.10. Núcleo del juego y peculiaridades de Unity

Como se explica en Unity (C#), se optó por el motor de videojuegos **Unity** para el desarrollo del juego, en este apartado se habla de cuestiones relativas al uso de este motor y elementos varios asociados al funcionamiento base del videojuego.

Para controlar el flujo general del juego, **Unity** proporciona herramientas simples y suficientes para la gestión de escenas. Como es una arquitectura por componentes, cada objeto se actualiza individualmente en su propio *Update* y pueden poseer métodos públicos para ser llamados desde otros componentes. Las inicializaciones y labores similares se dan en los métodos *Start* y/o *Awake*.

A pesar de poder tener objetos individuales que pudieran colaborar entre ellos y llevar a cabo el transcurso del juego, es importante tener un *GameManager* que maneje dicha colaboración y gestione las escenas. Así, el *GameManager* en este caso se encargará de los siguientes aspectos:

- Cambio de escenas, reinicios y gestión de niveles.
- Guardado, borrado y cargado de datos.
- Control de los objetos de inventario y escena.
- Colaboración superficial con uso Web y base de datos.
- Gestión del tiempo de juego (para estadísticas).

Así, siendo este *GameManager* una clase estática, pública y accesible para todos los elementos de escena, actúa como una capa intermedia que permite la interacción entre los elementos del juego y evita la pérdida de información entre escenas. Varias funcionalidades incluidas en este componente podrían haberse extraído en otros para aligerar su carga, pero dada la naturaleza del desarrollo, en el que surgían nuevos requisitos a medida que se avanzaba, y a la ventaja que supone hacer todos estos recursos accesibles desde el *GameManager*, se optó por mantenerlas dentro de este.

Unity ayudó a agilizar el desarrollo considerablemente, al proporcionar muchas funcionalidades básicas. Por otra parte, no daba soporte a la mayoría de las tecnologías de accesibilidad que utilizamos, lo cual permitió investigar e implementarlas a los miembros del equipo.

6.10.1. Gestión de las escenas

Para explicar cómo se manejan las escenas de la aplicación a nivel de cómo se instancian e interactúan con los datos de guardado, se clasificarán en tres tipos:

- **Habitaciones:** Las habitaciones son aquellas escenas en las que se navega por los objetos en el escenario e inventario. Para reflejar el progreso del jugador respecto a los datos de guardado, los objetos en estas escenas se generan cada vez que se accede a ellas. Los objetos se generan en base a *Prefabs* o "modelos", de uso habitual en Unity para instanciar objetos con unas características concretas, estos modelos los identificamos por su nombre (*LlaveOxidada1*, por ejemplo). A esos objetos, una vez generados se les asigna el estado correspondiente según los datos guardados, por defecto *DEFAULT*.

  Los objetos del inventario siempre se generan en base a los datos de guardado, quedando vacío si no hubiera. Los de la escena hacen una distinción, si se trata de la primera vez que se entra a una sala, estos se crean en base a un fichero interno que contiene la disposición inicial de cada sala. De no ser la primera visita, se generan en base a los datos de guardado, igual que los del inventario. La detección de la primera vez que se entra a una sala permitiría evitar repetir tutoriales (se decidió repetirlos de todos modos para evitar atascos en caso de no escucharlos la primera vez), o aportar información relevante al progreso actual del jugador.

- **Minijuegos:** A nivel de escena, los minijuegos tienen dos métodos que encapsulan su inicio y final, quedando el resto del comportamiento entre estos dos. En estos dos métodos se puede usar el número de la última habitación visitada para modificar los patrones o el resultado de superarlos, como a qué habitación te devuelven o si modifican el inventario y escenario guardados. Este comportamiento no se usa en el juego final ya que cada minijuego solo se visita una vez.

- **Transiciones:** Las escenas de transición (la introducción y la escena final), actúan como su nombre indica, de transiciones. Esto quiere decir que se limitan a llevar al jugador a otra sala sin poder fallarse, de modo que no interactúan con los datos de guardado, ni para saber la última habitación visitada, ni para realizar guardados (este se hace al entrar a la sala destino). Por esto, su generación y comportamiento siempre son los mismos, ya que siempre se visitan en las mismas circunstancias y una única vez.

6.10.2. Desarrollo para Android

Al hacer uso la aplicación de tecnologías específicas de **Android** estas tuvieron que ser integradas en el motor **Unity**, además de ciertas necesidades y configuraciones específicas de la plataforma.

- **Plugins:** Como ya se ha explicado anteriormente en sus respectivos apartados, se realizaron *plugins* para hacer uso de ciertos servicios proporcionados por la plataforma **Android**, en concreto la síntesis de voz (TTS) y el reconocimiento de voz (STT). Estos *plugins*, realizados en **Android Studio**, fueron exportados como archivos *.jar* e incluidos en el proyecto. Para poder realizar llamadas desde los componentes en **C#** se realizaron clases intermedias que hicieran llamadas a los *plugins*. Estas clases se hicieron estáticas, para facilitar su acceso y evitar requerir una instancia en escena de ellas. El caso de la vibración fue más sencillo, al proporcionarse *Vibrator* como un servicio básico en **Android**, las llamadas se podían realizar directamente desde un componente **C#** similar a los de TTS y STT, pero sin necesidad de un *plugin*.

- **Manifiesto de Android:** Las aplicaciones **Android** deben tener un *AndroidManifest*, el cual contiene información relevante y necesaria de esta. En esta información se incluye por ejemplo el nombre de la aplicación, las versiones de **Android** con las que es compatible, los permisos que puede solicitar, entre otros. En **Unity**, este archivo es generado por el motor de forma automática en base a la configuración elegida desde su propia interfaz. Como ya se explicó en Síntesis de voz a partir de texto. TextToSpeech (TTS), para hacer uso del STT de **Android**, era necesario mantener un *AndroidManifest* con el plugin para identificar la actividad de la aplicación. De ese archivo se eliminaron todas las demás configuraciones, para que no entraran en conflicto con las seleccionadas en **Unity**. Cuando se hace la *build* de la aplicación, el motor combina ambos archivos para generar el que quedará en la *apk*.

- **Otras configuraciones:** Para el correcto funcionamiento de la aplicación se tuvieron que configurar ciertos comportamientos específicos de la plataforma. Ente ellos se encuentran bloquear la rotación automática de la pantalla, que se configura como parte del *AndroidManifest* en **Unity**. La rotación se bloqueó porque la aplicación está diseñada para un uso exclusivo en modo retrato, nunca en horizontal. Además, permitir invertir el modo retrato podría llevar a situaciones confusas, en especial a los usuarios ciegos, que acostumbran a acercarse el móvil a la oreja y colocarlo en ángulos poco habituales.

También se desactivó el bloqueo automático del dispositivo mientras se está en la aplicación, ya que, si el dispositivo se bloqueara, un usuario vidente podría pensar que la aplicación había fallado y se había quedado atascada, al no haber un cambio visual entre la pantalla en negro del juego y la del bloqueo.

6.11. Audio

Debido a la poca relevancia que se le da al apartado visual en una aplicación de esta índole, el apartado sonoro es fundamental para su uso general. El audio no sólo expresa la narrativa y la música, sino también casi todo el *feedback* que recibe el jugador.
En este apartado se va a distinguir entre las tres partes en las que se ha dividido el trabajo de audio: Producción, edición, y tratamiento del audio 3D.

6.11.1. Producción

Gran parte de los efectos de audio se han obtenido de la página **Freesound** (Freesound Team, 2005), sitio web que ofrece audio totalmente gratuito y sin derechos de autor para ser utilizados de forma tanto no comercial como comercial. El audio se descarga únicamente registrándose en la página, y posteriormente se cambia y edita para introducirlo en el juego.
Debido a esta carencia de generación directa de la mayoría de los efectos de audio, en este apartado se puede tener en cuenta lo relativo a la grabación de audio (Audios explicativos TTS y voces humanas) y la creación y composición de la banda sonora.

Sobre el grabado de audios explicativos, la plataforma **Windows** proporciona una funcionalidad propia de TextToSpeech, la cual es utilizada comúnmente en aplicaciones como **Discord** para lectura automática sonora. Esta herramienta se consideró de calidad apropiada para el juego, ya que la entonación del se asemeja a una persona más que cualquier otra que se hubiera probado. Particularmente se optó por la implementación **Microsoft Helena** (Microsoft, 2012), la cual es una versión del paquete de voz que se proporciona de forma base con el sistema operativo **Windows**.

Teniendo en cuenta que los audios a los que sustituyen en un principio iban a ser grabados con una voz real (antes de los problemas del COVID-19), y que la aplicación estaba destinada a **Android** (por lo que no se podría incorporar esta herramienta al juego), se decidió que la mejor opción era convertir los TTS del lector de **Windows** en archivos de audio (MP3 y WAV) e introducirlos en el juego con un sistema propio de reproducción.
Este sistema, mediante un canal separado de audio, reproduce las frases grabadas que le envíen de cualquier parte del juego con un volumen normalizado.

El sistema está totalmente preparado para simplemente sustituir los audios TTS por voces de verdad donde sea necesario, por lo que en un futuro se podrían cambiar y hacerlo como en principio se quería. Este proceso de grabación de audios explicativos pregrabados y su posterior edición (se explicará más adelante) ha sido costoso y ha consumido mucho tiempo, ya que se ha requerido el uso de casi un centenar de ellos.
Además del audio obtenido de Freesound y del TTS de Windows, el juego requiere de voces humanas para sus dos personajes, por lo que se decidió grabarlos utilizando personas de confianza que tuvieran los requisitos de la voz pensada para el juego. Debido a la gran problemática del COVID-19, los audios no se pudieron grabar presencialmente. De este modo, para los personajes:

- **Protagonista:** Inicialmente se grabó la voz de Alberto Casado Trapote, pero posteriormente se cambió por la de Héctor Marcos Rabadán y finalmente la de Diego Martínez Simarro. Las tres voces fueron grabadas, editadas e introducidas en el juego, pero se acabó decidiendo que la de Diego sería la definitiva por tener un timbre de voz más grave.
- **Hija de protagonista:** Esta voz se ha obtenido de una familiar de 13 años, que mediante audios mandados por Whatsapp (debido al Covid-19) y su posterior edición se ha conseguido introducirla en el juego. Se grabaron un total de 6 audios, los cuales han sido modificados y se han convertido en 9 distintos.

Estos audios, posteriormente, se han editado en Audacity (Audacity Team, 2000) como se explicará en el siguiente apartado.

En lo que atañe a la composición musical, el juego es simplista; no requiere especial complejidad en la banda sonora, ya que el estilo de la narrativa y diseño general no propicia su uso. Es por eso por lo que el juego posee en total dos temas musicales:

- **Música de ambiente en escena "Laberinto / Persecución":** Esta parte que, descrita en su apartado, trata de una persecución en la que se escuchan las voces de tu hija, requería de un audio de fondo que expresara la situación en la que se encuentra el jugador. Esta música de fondo, con el uso de instrumentos como arpas, campanas y otros habituales en bandas sonoras, intenta transmitir al jugador una experiencia parecida a un sueño, una carrera etérea en la búsqueda hacia su hija. El audio, de una duración de un minuto aproximadamente (lo que dura la escena de Persecución), trabaja con tonos alargados, ecos y reverberaciones para conseguir la sensación descrita. En la Figura 22 - Música de escena Persecución podemos observar los tres canales para los instrumentos utilizados y su estructura.

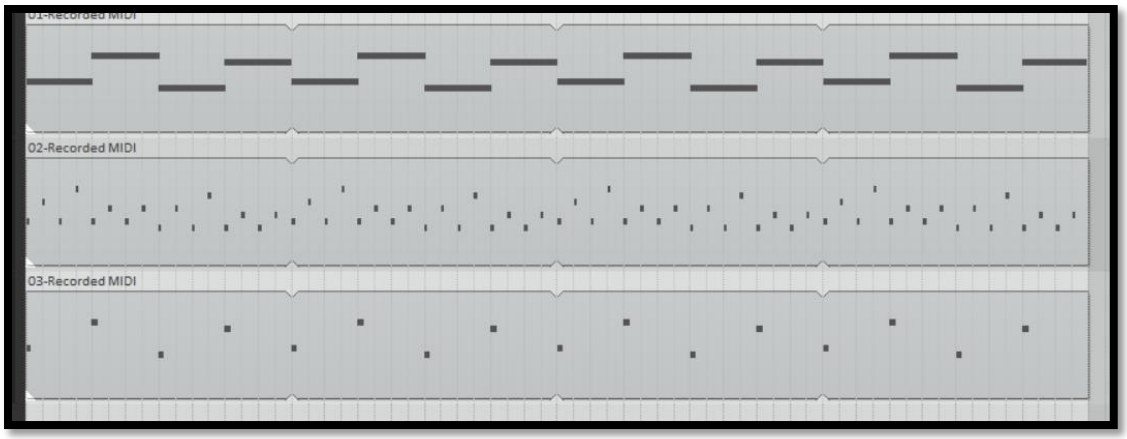


Figura 22 - Música de escena Persecución

- **Música final/de créditos:** En esta parte se compuso una canción con tempo lento relativamente acorde a la sensación de conclusión que se quería ofrecer al jugador. Esta canción, de una duración de unos 45 segundos en bucle, se compuso con herramientas de muestreo para bandas sonoras, sonidos atmosféricos, piano y triángulos. Este audio conecta directamente con el final del juego, en el cual suenan pitidos de máquinas de hospital, por lo que se quería hacer una transición de los sonidos electrónicos a los propios de los instrumentos. Así, la canción se "parte" en dos mitades, siendo la primera simplemente unos pequeños tonos de piano agudos (bastante parecidos a los pitidos mencionados) y cuya segunda parte es la que realmente entra en bucle y queda como cuerpo del tema. Se puede observar la estructura descrita en la Figura 23 - Música de fin de juego, estando marcada con un tono más blanquecino la zona principal de bucle.

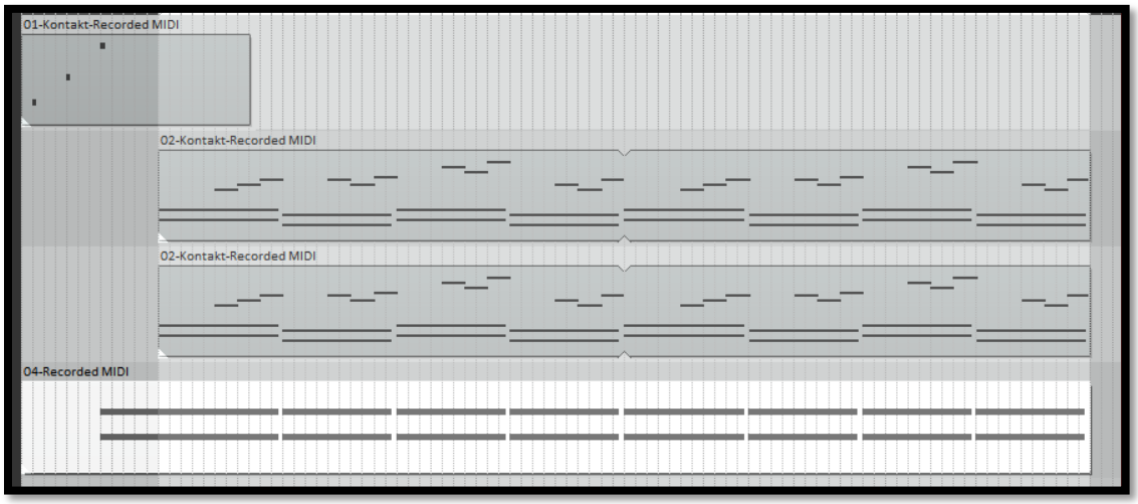


Figura 23 - Música de fin de juego

La producción de estas canciones se ha realizado en la herramienta de producción y edición de audio **Reaper** (Cockos, 2006), la cual permite, con el uso de *plugins*, la incorporación de instrumentos de diversa índole necesarios para la creación sonora y su edición. Se ha utilizado en concreto un *plugin* muy conocido llamado **Kontakt** (Native Instruments, 2012), al cual se han incorporado los paquetes de instrumentos **Kontakt Factory Selection** y **Play Series Selection**. Del primer paquete se obtuvo el instrumento piano, y del segundo la herramienta de muestreo y otros instrumentos como los triángulos, flauta y sintetizador electrónico.

Todo este proceso de creación y edición en **Reaper**, a nivel de composición, ha seguido en lo posible las pautas y conocimientos adquiridos en asignaturas de sonido cursadas en el GDV, mientras que a nivel artístico se ha buscado inspiración y conocimiento en bandas sonoras existentes.

6.11.2. Edición

Todos los audios mencionados en el apartado anterior han requerido de algún tipo de edición. Por lo general, todos los audios han sido editados con la herramienta **Audacity**, aunque varios en concreto fueron editados con **Reaper**. Para hablar de los métodos de edición, se ha de distinguir entre las mismas categorías que el apartado anterior, ya que los audios similares han seguido relativamente las mismas pautas. Así:

- **Audios explicativos TTS:** El apartado más numeroso, pero que requiere de menos edición. Tras obtener los audios, el proceso general ha tratado de recortar las frases y palabras desde **Audacity**, eliminando los espacios silábicos y entre palabras indeseados, para así acelerar la reproducción y hacerla menos pesada para personas ciegas, ya que por lo general utilizan lectores de pantalla muy rápidos. La velocidad general de las frases obtenidas no es tan rápida como en los lectores de verdad, pero es algo más veloz que la predeterminada del TTS de **Windows**, y de este modo sigue siendo viable para personas no conocedoras de estos sistemas. Además, todos estos audios han sido normalizados para reproducirse al mismo volumen, y posteriormente nivelados desde los reproductores de audio de **Unity** para evitar volúmenes indeseados respecto al resto de sonidos del juego.
  Aquí se encuentran alrededor de 90 audios distintos.

- **Efectos de sonido:** Este apartado engloba al audio general del juego. Aquí se encuentran sonidos como los pasos del jugador, golpes, apertura de objetos y efectos en general. Como ya se ha mencionado, la mayoría de estos audios se han obtenido de **Freesound**, pero han sido editados para acomodarse a las necesidades

de la situación en la que se incorporan. De este modo, es realmente difícil poder hablar de la edición que se ha realizado a cada audio, por lo que se van a mencionar de un modo general.

Para los efectos de pasos, se han obtenido muchas muestras, han sido recortadas con una duración similar, y se han editado de tal forma que encajen en los mismos tonos. Por ejemplo, los pasos realizados en madera se han obtenido mezclando sonidos de pasos normales distintos con golpes de madera y crujidos. Del mismo modo se han hecho pasos en hierba y cemento. Estos en ocasiones han requerido también de reducciones de ruido (por mala calidad del audio inicial) y cambios de tono.

También ha sido necesario el añadido de efectos de sonido no realistas, como es el caso de *flashbacks*. A nivel de edición, se ha tratado de añadirles a los efectos base tonos agudos con mucho eco y reverberaciones para darles sensación de profundidad. Para el resto de los efectos de audio no encapsulables, se han utilizado mezclas, cambios de tono*, fade-in* y *fade-out*, filtros *low-pass* y otros varios para sonidos específicos. Aquí se encuentran alrededor de 50 audios distintos.

- **Voces humanas:** Respecto a la edición de las dos voces que hay en el juego, la del protagonista no ha requerido de demasiada edición. Solamente se ha reducido el ruido de las grabaciones iniciales y se ha cambiado ligeramente el tono para hacerlo un poco más grave. Además, en un caso particular se ha aplicado un filtro *low-pass*.

  En el caso de la voz de la niña, se ha necesitado mayor edición. La voz se transmitió por audios de **Whatsapp**, y el micrófono con el que se grabaron no era muy bueno, por lo que se tuvo que reducir el ruido en gran medida, además de recortarse sílabas y añadir tonos generados por el propio **Audacity** para complementar las carencias de la voz. Además, las situaciones en las que era necesaria esta voz, sin contar una de ellas, tuvieron que ser tratadas para ser “del más allá”, es decir, con efectos de tal forma que pareciera que la voz no está presente de forma física. Así, se le han añadido ecos, reverberaciones, filtros *low-pass* y *high-pass,* cambios de tono, filtros *Paulstretch* para alargar ciertas sílabas (Audacity, 2000), y cambios de velocidad.
  Además, estas voces han tenido que ser unidas con otros efectos de sonido propios del juego para encajarlas en la dinámica de la escena. Por ejemplo, en ciertas escenas la voz de la niña ha sido encajada con efectos de *flashback* y sonidos varios, haciendo mezclas también entre ellos.

Hay que considerar el hecho de que ni las voces ni las formas de grabación son profesionales, lo que provoca que el resultado final no sea perfecto. Igualmente, el proceso de edición se habría realizado de una forma relativamente similar con otros audios de mayor calidad. Este apartado cuenta con un total de 16 audios.

6.11.3. Audio 3D

El sonido 3D o estereofónico, a diferencia del convencional (o monoaural), posee dos canales de audio en lugar de uno, los cuales son tratados y mandados a distintos lados en los dispositivos que se encuentren adaptados para su reproducción (como son, por ejemplo, ambos lados de unos auriculares). El objetivo principal de este recurso es proporcionar una mayor sensación de naturalidad del sonido al oyente. La modificación o edición de este tipo de audio permite, entre otras cosas, dar la sensación al usuario de que el sonido proviene de una dirección específica en el espacio.

En los momentos del juego donde se requiere este audio 3D (minijuegos Laberinto / Persecución y *Simon Says* Sonoro), se han utilizados audios obtenidos también de la plataforma **Freesound**, tratados posteriormente de la misma manera en el editor **Audacity** y asegurándose de que se encuentran grabados en estéreo. Sin embargo, a la hora de implementarlos en el juego, el motor de **Unity** facilita sustancialmente el trabajo ya que ofrece herramientas para tratar un audio como 2D o 3D. Con la opción de audio 3D seleccionada, el sonido se panea (se asigna a uno de los canales estéreo, por lo que suena con más intensidad en uno de los auriculares) automáticamente desde su posición en la escena de juego, tomando como posición del jugador (desde la que se escucha) la posición de la cámara. Además, **Unity** también ofrece ciertos elementos de modificación del audio 3D, como son la distancia a la que comienza a escucharse el sonido (con un volumen mínimo), la distancia a la que el volumen del sonido alcanza su pico máximo, posibles opciones de reverberación, y variaciones de tono y volumen. Muchas de estas opciones han sido modificadas desde el editor de juego para que los sonidos 3D se adecúen más al ambiente de juego y no choquen con el resto de música y sonidos.

6.12. Menús del juego

A pesar de la simplicidad de los menús del juego tanto en complejidad como en aspecto, su diseño fue un tema sobre el que se realizaron varias iteraciones y fue el centro de varios debates. Esto se debe a que debían ser coherentes e intuitivos tanto para usuarios ciegos como videntes, y facilitar su uso por parte de personas con restos de visión.

El principal compromiso con los usuarios ciegos era respetar las pautas marcadas en el resto de la aplicación en el modo de interacción, siguiendo el ejemplo de **TalkBack**.

Con los usuarios videntes, el desafío consistía en mantener esa coherencia a la vez que se le introducía al lenguaje de un lector de pantalla, ya que, aunque es de esperar que los usuarios ciegos conozcan su modo de uso, quienes no hubieran usado uno no tenían por qué conocerlo. Al ser estos menús la primera toma de contacto con la aplicación, forman una oportunidad para facilitar la transición a este modo de uso que debía ser aprovechada.

En las reuniones presenciales en la primera etapa del desarrollo se plantearon algunos prototipos en papel, que servirían como base para los diseños elaborados en etapas posteriores, aparecen recreados en la Figura 24 - Prototipo menú 1 y la Figura 25 - Prototipo menú 2 (deslizante).

Figura 24 - Prototipo menú 1

Figura 25 - Prototipo menú 2 (deslizante)

En la aplicación final acabaría habiendo dos tipos de menús, con objetivos distintos que llevaron a compromisos entre usabilidad y aspecto diferentes.

6.12.1. Menús deslizantes

El primer tipo de menú que se trabajó era el que se encontraría en el menú principal, nada más abrir el juego, para elegir entre distintos modos de juego, cerrar la aplicación, etc. Una aproximación cómoda y habitual en este tipo de menús sería una similar a la que vemos en la Figura 24 - Prototipo menú 1, en que se muestran todas las opciones a la vez, ordenadas verticalmente. Sin embargo, pronto se observaría que ese enfoque no serviría en esta aplicación.

Aunque para un usuario ciego no causaría problemas, siempre que se respetara el modelo de TalkBack, con usuarios videntes causaría fricciones en el uso. Con este tipo de distribución, estos usuarios esperarían seleccionar las opciones pulsando sobre ellas, pero esta no es la forma en la que se interactúa con ellos, ya que, de poder pulsar sobre las opciones, los usuarios ciegos podrían hacerlo involuntariamente al realizar deslizamientos, un comportamiento no esperado. Además, la forma de desplazarse entre elementos es con deslizamientos horizontales, poco intuitivos con una distribución vertical. Es por esto por lo que ese diseño de menú es poco recomendable en este tipo de situación.

El objetivo entonces era claro, conciliar los deslizamientos horizontales y la distribución de los elementos del menú. La respuesta se encontró sin muchos problemas, ya que es muy habitual en aplicaciones móviles: los menús deslizantes, en los que los elementos aparecen ordenados horizontalmente y se alterna entre ellos con deslizamientos hacia los lados, de forma muy similar al modelo de lector de pantalla que seguía la aplicación.

En base a esto se decidió probar este modelo, como el mostrado en la Figura 25 - Prototipo menú 2 (deslizante) este modelo resultó ser una buena elección ya que conciliaba el modo de uso y el aspecto visual de forma orgánica. Aun así, se encontraron algunos problemas a la hora de su implementación. El primero fue bastante sencillo de resolver, y consistía en hacer entender al usuario que se encontraba ante un menú de este tipo, bastó con hacer visibles los elementos adyacentes, dando a entender que se puede desplazar a ellos.

El principal problema surgía de la dirección de los deslizamientos. Estos debían avanzar al siguiente elemento al deslizar hacia la derecha, y retroceder al deslizar hacia la izquierda, para ser consistente con un lector de pantalla. Por el contrario, los usuarios videntes esperarían un desplazamiento hacia la izquierda al desplazar hacia la derecha y viceversa, ya que es lo habitual en este tipo de menús.

Este comportamiento se muestra en la Figura 26 - Comportamiento habitual en menús deslizantes. Como se vería tras elaborar sobre esta inconsistencia, el problema surgía de la colocación de los elementos.

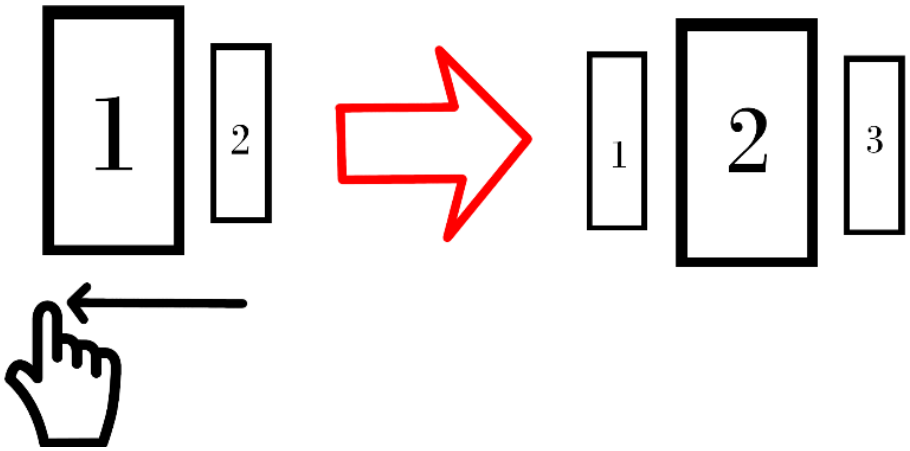


Figura 26 - Comportamiento habitual en menús deslizantes

Al situar el primer elemento de la lista a la izquierda y los siguientes a la derecha, el deslizamiento que un usuario vidente esperaría hacer para avanzar sería deslizar a la izquierda, el contrario del que se le debía comunicar. Tras examinar como solucionar este problema, la solución con la que se dio resultó ser bastante sencilla, pudiendo considerarse un "engaño". Lo que se haría sería colocar el primer elemento en el extremo derecho de la lista, el que un usuario vidente consideraría el último, esto lo muestra la Figura 27 - Menú deslizante inverso.

De este modo le resultaría natural deslizar hacia la derecha para pasar al siguiente elemento. Así, realizaría el movimiento correcto de forma intuitiva, aunque pensara estar retrocediendo en lugar de avanzar. También ayudó a comunicar este enfoque una animación de transición entre elementos.

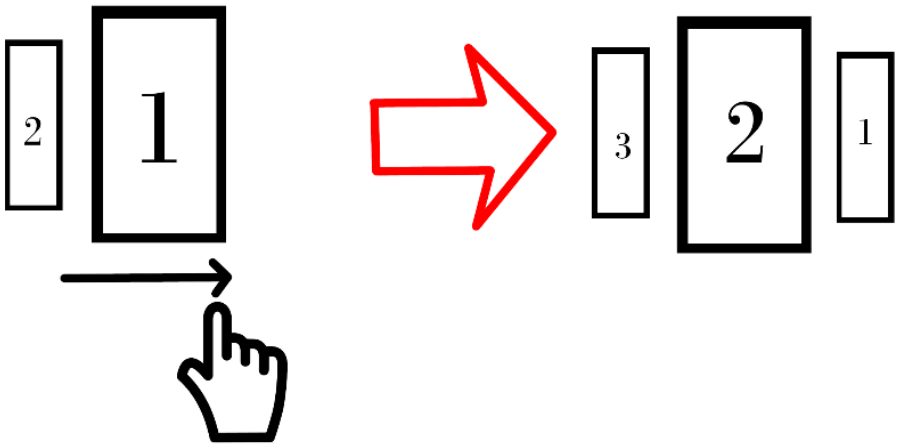


Figura 27 - Menú deslizante inverso

Este modelo dio muy buenos resultados, ya que, manteniendo el lenguaje esperado por usuarios ciegos, era intuitivo para usuarios videntes o con restos visuales. Para estos últimos se decidió hacer que los botones fueran muy grandes, con iconos distintivos, además de emplearse colores de alto contraste, obteniendo un resultado como el que aparece en la Figura 28 - Menú deslizante. Otro beneficio de este modelo era que introducía a los usuarios videntes a los gestos que se esperarían de ellos durante el resto de la aplicación, sirviendo como una primera toma de contacto que asienta esos gestos.

Figura 28 - Menú deslizante

6.12.2. Menús con *input* por voz

En un momento más avanzado del desarrollo, se observó un problema importante dentro del modo “con conexión”. Como muestra la Figura 29 - Menú login 1, cuando un usuario entraba en dicho modo, este necesitaba introducir datos como texto para poder acceder al juego, ya fuera iniciando sesión o creando una cuenta nueva. En ambas pantallas el problema estaba a la hora de introducir los datos en los recuadros de texto. Durante el desarrollo, para hacer pruebas, se interactuaba pulsando alguno de los recuadros para introducir datos, lo que hacía aparecer el teclado del positivo **Android** para introducirlos. Este método de introducir los datos para personas videntes es el más habitual.

En cambio, era obvio que para una persona ciega este sistema sería un gran impedimento o incluso llegaría a imposibilitar su uso el tener que introducir los datos de esta manera. Por ello se debían pensar en otros métodos de introducción de texto que resultaran más sencillos para las personas ciegas.

Fue esto lo que llevaría a la implementación mostrada en la Figura 29 - Menú login 1. Con este sistema, cuando se activa uno de estos elementos, aparecería automáticamente el recuadro emergente del dictado por voz de **Google**, como el que aparece en la Figura 30 – Menú login 2, en el cual tendría que decir lo que quisiera introducir y al terminar se introduciría de forma automática.

Figura 29 - Menú login 1

Figura 30 – Menú login 2

Una vez solucionado el problema de la introducción de texto, restaban el modo de navegar el menú y cómo mostrar los elementos. Este debía ser consistente con el resto del juego. Para esto se consideró usar el mismo enfoque de menús deslizantes explicado anteriormente, pero se decidió que no era deseable para este tipo de menú. Esto se debía primero, a las proporciones de los recuadros de texto, muy anchos y de poca altura, que dejarían muy poco espacio a los lados para mostrar elementos adyacentes y mucho espacio vacío arriba y abajo. Segundo, al tratarse de pantallas en las que se introducen datos, era importante que los usuarios con visión pudieran ver todos los elementos de la pantalla simultáneamente, para poder comprobarlos rápidamente sin tener que desplazarse por el menú. En base a esto, se volvió al diseño planteado en la Figura 25 - Prototipo menú 2 (deslizante), y se realizaron cambios para adaptarlo lo mejor posible a las necesidades de este menú.

Lo primero fue utilizar colores de alto contraste, igual que en los menús deslizantes, en especial al borde de color que se usó para destacar el foco actual. El principal problema de este modelo seguía siendo el choque entre los deslizamientos horizontales que deben realizarse y la distribución vertical, pero se decidió realizar este compromiso. Ya que, para llegar a este menú, el usuario ya debía haber navegado los anteriores y, de este modo, haber interiorizado los gestos que deben utilizarse. En este punto del desarrollo se pasó por alto un problema, y era hacer visibles los datos introducidos a personas ciegas después de introducirlos, este problema no saldría a la luz hasta que nos lo hiciera saber el *feedback* recibido de la ONCE, mucho más tarde, que aparece en la sección de Feedback ONCE.

6.13. Minijuegos

Como ya ha sido mencionado en apartados anteriores, la aplicación cuenta con 5 minijuegos durante el transcurso total de la historia. Todos ellos se centran en el uso de la vibración, el audio 3D y el sistema de gestos con el fin de que sean completamente accesibles para las personas ciegas. Como es lógico, ninguno de estos minijuegos posee apartado visual; en el caso de tenerlos, estos no deberían dar ventaja a una persona vidente a la hora de resolverlos. Los minijuegos, además, han ido siendo modificados y ajustados en función del *feedback* recibido durante las fases de prueba del proyecto.

A continuación, se expone de manera más específica información de cada minijuego, clasificada en los siguientes cuatro apartados:

- **Concepto**: Expone la idea general del minijuego, así como de dónde surgió el concepto y las posibles referencias.
- **¿Cómo se juega?**: Detalla la jugabilidad del minijuego y los controles del usuario.

- **Desarrollo**: Narra cómo fue el proceso de desarrollo y la estructura realizada para construir el minijuego.
- **Postproducción**: Explica cómo fue el *feedback* recibido durante las pruebas y cómo se modificó el minijuego en función de este.

Los minijuegos aparecen ordenados según su aparición en el juego, mientras que las figuras que ilustran los conceptos de los minijuegos contienen un número de acuerdo al orden en que plantearon.

6.13.1. Ganzúa

- **Concepto**

La idea de este minijuego surgió de las clásicas películas de ladrones, en las que alguien intenta abrir una caja fuerte encontrando la combinación del candado únicamente por sonido. El ladrón coloca la oreja en la puerta y escucha los pequeños clics que hace el candado al ser girado, y cuando el sonido es ligeramente distinto, es que ha encontrado la posición de la combinación.
En este pequeño puzle el objetivo del jugador es hallar el punto de desbloqueo del candado de una caja con una ganzúa anteriormente recogida, tal y como se puede ver en la Figura 31 - Concepto inicial del minijuego Ganzúa. Este tipo de puzle es relativamente habitual en el sector de los videojuegos, encontrado en varios juegos famosos, como en **The Elder Scrolls V: Skyrim** (Bethesda Game Studios, 2011).
El minijuego se basa en el apartado sonoro, hay que escuchar y encontrar el sonido distinto en el candado.

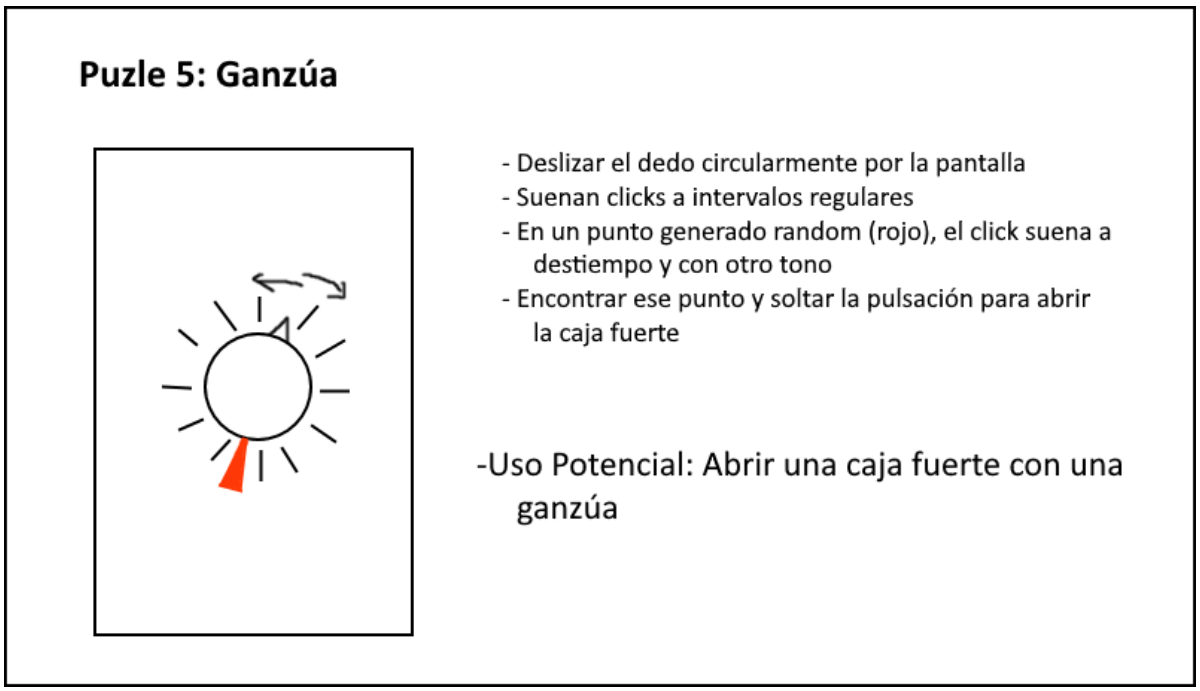


Figura 31 - Concepto inicial del minijuego Ganzúa

- **¿Cómo se juega?**

El jugador deberá deslizar el dedo de forma circular por la pantalla, como si estuviera rotando el objeto. Mientras se está rotando, sonarán unos pequeños clics, y en la posición de desbloqueo este clic sonará de manera distinta. Cuando el jugador encuentre ese clic distinto, bastará con levantar el dedo para desbloquear el candado.

- **Desarrollo**

Desarrollado con tratamiento de *input* propio de Unity y con la ayuda de la clase que gestiona los gestos, el minijuego se ha desarrollado "físicamente", es decir, se ha utilizado un objeto en escena como candado que rota siguiendo la posición del dedo en pantalla. La posición de desbloqueo del candado se elige al entrar en la escena, siendo esta un cono de unos 15 grados colocado aleatoriamente de 30 a 330 grados sobre la posición base. Esta colocación sirve para que la posición de desbloqueo no se encuentre en la posición original y que no puedas abrir el candado "sin querer" nada más comenzar el minijuego. Cada vez que el candado se rota más de X grados en la misma dirección, se libera un sonido de clic. Si se encuentra dentro del "cono de desbloqueo", este sonido no sólo es distinto, sino que suena asíncronamente respecto a los demás clics; es decir, si se mantiene una velocidad de giro constante, los clics suenan en intervalos constantes salvo el de desbloqueo, que se desfasa ligeramente.
Todo esto está "tapado" visualmente.
El minijuego es relativamente sencillo e intuitivo, aunque igualmente se le explica al jugador antes de empezar.

- **Postproducción**

El *feedback* recibido en este minijuego en concreto fue relativamente bueno desde la primera versión mostrada. La mayor complicación de la que se tuvo constancia fue en las primeras versiones, debido a que en estas todavía no había audios instructivos al comienzo de los minijuegos explicando los controles. Una vez estos tutoriales fueron introducidos en versiones posteriores, no hubo ningún otro problema encontrado en este minijuego.

6.13.2. Laberinto / Persecución

- **Concepto**

Desde que se empezó a plantear el diseño de los minijuegos se quiso utilizar las propiedades del sonido 3D para darle más opciones de jugabilidad. Esto se tradujo principalmente en utilizar el paneado del sonido (lo cual requiere de auriculares, aunque ya se contaba con ello) de tal modo que el jugador pudiera percibir el sonido de una manera más direccional e inmersiva, pudiendo reconocer con más o menos exactitud la procedencia espacial de los sonidos que escuche durante el transcurso del minijuego.

Con esto en mente se tuvo la idea de realizar un laberinto guiado por sonido 3D. En este laberinto el jugador se iría moviendo de manera automática hacia delante hasta llegar a una bifurcación del camino, momento en el cual se le ofrecería al jugador un sonido procedente del camino correcto, mandado completamente a uno de los dos lados del auricular. Gracias a esto el jugador podría reconocer fácilmente la dirección, como se puede apreciar en la Figura 32 - Concepto inicial del minijuego Laberinto Sonoro. De ese modo el jugador proseguiría avanzando por el camino, llegando eventualmente a la meta.

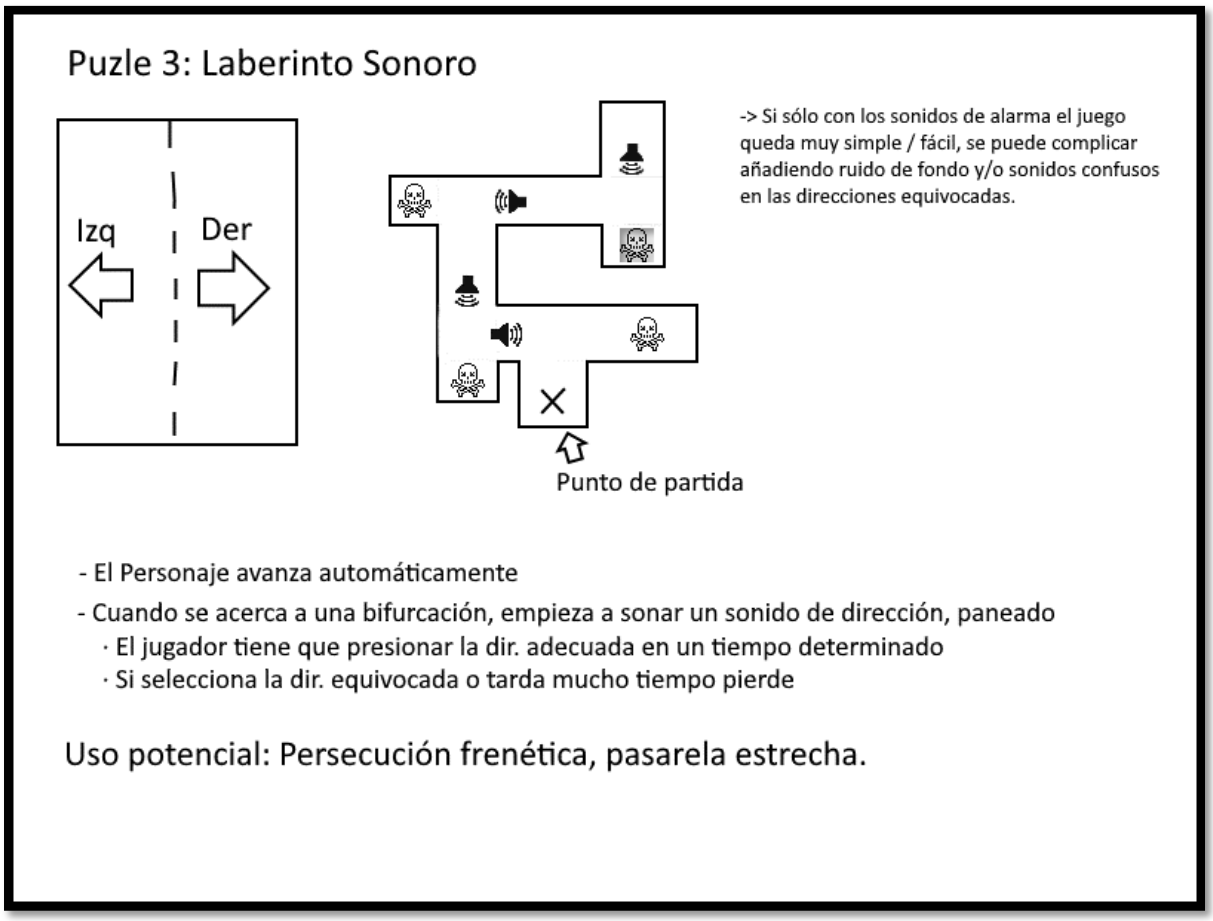


Figura 32 - Concepto inicial del minijuego Laberinto Sonoro

- **¿Cómo se juega?**

Ya que este minijuego está pensado para ser algo más frenético con respecto al resto del juego en general (es el único momento que cuenta con *quick time events*), los controles del minijuego son sencillos y directos. Esto permite reducir el esfuerzo del jugador al introducir el *input* y poder centrarse más en su inmersión. De esta manera, los únicos movimientos que puede realizar el jugador son deslizamientos hacia la izquierda o hacia la derecha en la pantalla.

Cuando el jugador llegue a una bifurcación del camino y se reproduzca el sonido correspondiente, el jugador deberá reconocer si el sonido proviene de su izquierda o de su derecha, y realizar el deslizamiento correcto antes de que el sonido acabe. En caso de que el jugador gire hacia el lado erróneo, o tarde demasiado en introducir el *input*, se le restará una vida u oportunidad. En el momento en el que el jugador pierde un total de 2 vidas, perderá, y deberá comenzar de nuevo el minijuego desde el principio.

- **Desarrollo**

El desarrollo de este minijuego pasó por algunas trabas con respecto al planteamiento inicial y la manera de representar la idea en **Unity**. De este modo, se realizaron hasta tres versiones distintas del minijuego, con diferencias notables que se explican a continuación.

Primera implementación. Se comenzó con una representación física del laberinto, con muros y bifurcaciones, en la que el jugador que avanzaba linealmente hacia adelante. Al acercarse a una bifurcación, entraba en colisión con una zona de reconocimiento que activaba el sonido correspondiente, y mientras se mantuviera dentro de esa zona, se recogían los *inputs* introducidos. Una vez se tuviese información del deslizamiento introducido, se paraba el audio en reproducción, se posicionaba al jugador en el centro del pasillo (para evitar acercamientos imprevistos a los muros laterales) y se giraba en el sentido del deslizamiento introducido. En el caso de que el jugador se equivocara de dirección, se reiniciaba el laberinto directamente, sin contar con varias oportunidades.

Segunda implementación. Posteriormente, se pensó en cambiar el sistema del minijuego para que el movimiento del jugador fuera libre, pudiendo girar de dirección en cualquier momento, ya que se creía que en la versión anterior las limitaciones de movimiento y la dificultad elevada (perder en el minijuego por fallar únicamente un *input*) eran un problema. En este sistema, el sonido que guiase a la solución al jugador estaría activado constantemente. Esto, combinado con el propio movimiento del jugador, daría una mayor sensación de inmersión. Además, el sistema estaba preparado para detectar si el jugador se estaba acercando demasiado a una pared, y mostraba un sonido de advertencia para que cambiara de dirección antes de que fuera demasiado tarde. En lugar de penalizar al jugador por elegir los caminos incorrectos, lo que se hizo fue cerrar esos caminos, convirtiéndolos en caminos sin salida. Se esperaba que el jugador se diera cuenta, gracias a la percepción del sonido del camino correcto y del sonido de advertencia de las paredes, de que debía de darse la vuelta para poder continuar. No obstante, en el sistema se mantenía la condición del sistema previo en el cual el jugador perdía de manera inmediata al chocar con una pared, reiniciando el minijuego de nuevo.

Tercera implementación. Más adelante se decidió que las versiones anteriores no encajaban con los objetivos. Ambas tenían una dificultad demasiado alta respecto a la deseada. Por esto, se hizo un rediseño del minijuego y se construyó lo que sería la versión final del minijuego, mucho más abstracta con respecto a las anteriores en cuanto a la forma de representar el "laberinto". En esta versión, se eliminaron por completo los muros del laberinto, y se hizo que el jugador avanzase linealmente hacia adelante durante todo el minijuego. Periódicamente, colisionaría con zonas de reconocimiento que activarían el sonido paneado (similar al funcionamiento del primer sistema), que reproducen el sonido bien a la izquierda o a la derecha del jugador. Mientras el jugador se encuentre en una de estas zonas, deberá realizar un desplazamiento hacia la dirección de la que proviene el sonido, antes de que este termine. Si acierta, se pasará a un pequeño evento en el que el jugador rotará, haciendo que el sonido pase a escucharse desde un lateral a estar centrado, con el fin de dar la sensación de que el personaje se gira para orientarse hacia el sonido. Si falla o no introduce un deslizamiento en el tiempo necesario, sonará un sonido de golpe indicando que se ha equivocado, y se restará una vida al total de 2 que posee.

También cabe mencionar que todos los datos que dirigen el juego son modificables de manera sencilla desde el editor de Unity. Esto incluye entre otras cosas, la velocidad a la que se mueve el jugador, el tiempo que duran las zonas de "bifurcación", el posicionamiento del sonido 3D, el número de vidas, la longitud del minijuego y el número de eventos que ocurren durante el mismo. Cambiar la configuración de estos parámetros permitiría crear niveles muy diferentes entre sí sin tener la necesidad de acceder al código, pudiendo ajustar la dificultad al nivel que se desee.

- **Postproducción**

Como ya se ha mencionado, este minijuego pasó por varias fases hasta llegar a su versión final. Sin embargo, este proceso de cambio de fases tuvo lugar íntegramente dentro del desarrollo del proyecto, con los cambios basados en la opinión de los desarrolladores. Por esto, todo el *feedback* recibido por parte de probadores está basado en la versión final del minijuego. Este *feedback* fue positivo casi en su totalidad, excepto casos en que los probadores jugaban al juego sin usar auriculares, a pesar de que al inicio del juego se indica su necesidad. Fuera de estos casos, no hubo más problemas con este minijuego, de modo que se mantuvo prácticamente intacto desde que se completó su desarrollo.

6.13.3. Búsqueda por vibración

- **Concepto**

La siguiente herramienta que se quiso explotar en el desarrollo de los minijuegos fue la vibración. Con esto en mente, se investigaron diseños de juegos que usaran esta función extensivamente. Se tomó como referencia un juego muy sencillo mostrado durante la visita a la ONCE. Este consistía en una pompa circular que se desplazaba por la pantalla y el usuario tenía que explotarla haciendo clic sobre ella. Para encontrarla, se usaba la vibración del teléfono móvil, que vibraba con mayor o menor intensidad en función de la distancia de la pompa al punto que el usuario estuviera presionando. Este juego, en su simpleza, sirvió como base para tratar de diseñar algo más complejo y que presentara un reto algo mayor.

Se pensó en las clásicas escenas donde se apoya la oreja en un suelo o pared para escuchar con mayor precisión algo que se encuentre al otro lado. También se añadió a la tormenta de ideas el funcionamiento del sónar de los barcos y submarinos.

Con estos elementos se acabó diseñando un minijuego en el que hay varios focos de vibración por la pantalla, siendo sólo uno la solución. Las vibraciones guían al jugador hasta el centro los focos, como se puede observar en la Figura 33 - Concepto inicial del minijuego Búsqueda por Vibración, hasta que el jugador acabe llegando al foco solución y se pase el minijuego.

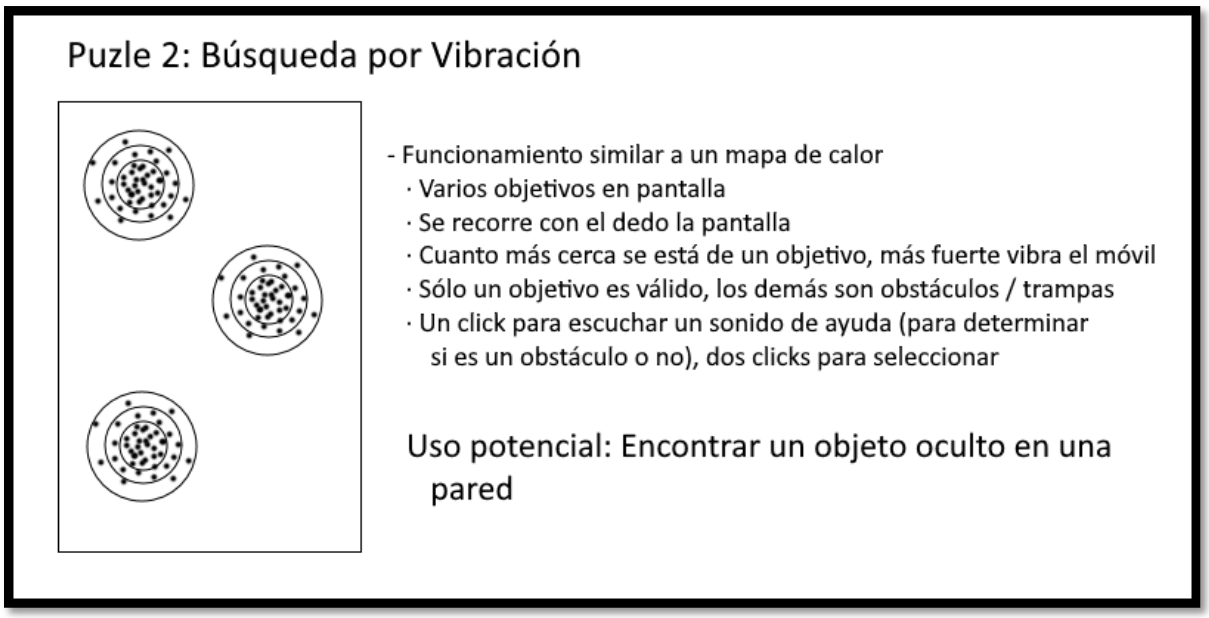


Figura 33 - Concepto inicial del minijuego Búsqueda por Vibración

- **¿Cómo se juega?**

Se utilizan dos gestos para desenvolverse en este minijuego: el propio movimiento del dedo sobre la superficie del móvil y doble clic sobre los focos de vibración.

El jugador debe ir recorriendo con el dedo la pantalla. Cuando se acerque a uno de los focos, el móvil comenzará a vibrar con una cierta frecuencia. La frecuencia irá aumentando por escalas según se acerque al centro del foco de vibración, momento en el cual la vibración pasará a ser continua. Para seleccionar el foco e intentar resolver el minijuego, el usuario deberá hacer doble clic sobre el mismo punto. En caso de fallar, salta un sonido indicando el fallo en el intento y se procede a eliminar el foco de vibración, facilitando el minijuego al quedar menos focos activos con cada fallo. Por el contrario, si se acierta, se consigue completar exitosamente el minijuego, desbloqueando el ítem oculto y prosiguiendo con el juego principal.

- **Desarrollo**

El desarrollo del minijuego es bastante simple, gracias a la buena implementación previa del sistema de gestos y el de vibraciones. En escena se cuenta con un mánager que gestiona una lista de posiciones, que serán los focos de vibración. Para cada uno de estos puntos, se tiene además tres indicares de rango para especificar a partir de qué distancia se empieza a vibrar con una cierta potencia. Los parámetros de las vibraciones (potencia y duración) y los rangos de acción de cada foco son editables desde el editor.

Cuando el usuario realiza un desplazamiento por la pantalla o un doble clic se procede a hallar el foco más cercano a la posición donde se encuentra el usuario, recorriendo la lista. Una vez se tiene el centro más cercano se mira el tipo de *input* del usuario. En caso de ser de tipo desplazamiento, se comprueba la distancia al foco, y se procede a activar la vibración correspondiente a dicha distancia. Con el doble clic se realiza el mismo proceso anterior, pero, en caso de estar dentro de la zona céntrica, se comprueba si es el foco solución o no y se activan los sonidos y eventos correspondientes.

- **Postproducción**

La postproducción en este minijuego resultó ser algo más notoria, incluyendo algún cambio relevante en el transcurso del juego. Inicialmente, cuando se detectaba un doble clic sobre el centro de un foco erróneo de vibración, se restaba una vida al jugador (que contaba con un total de 2) y el foco no desaparecía, complicando el minijuego notablemente.

En unas primeras versiones, había otra funcionalidad que rápidamente fue desechada debido al feedback recibido. Esta trataba del uso de un clic normal en lugar del doble para tratar de saber si el punto seleccionado era la solución, ofreciendo un audio distinto. Se observaba que la sensación general era bastante confusa, ya que la gente parecía no entender bien la diferencia entre los dos tipos de clics. Con estas opiniones se optó por facilitar el juego, en principio cambiando los audios para diferenciar más el clic del doble clic, posteriormente haciendo que los focos desapareciesen tras hacer doble clic sobre sus centros, luego eliminando el sistema de vidas, y finalmente optando por desechar la funcionalidad del clic normal. Tras todos estos cambios el *feedback* recibido fue mayoritariamente positivo, sin ningún otro problema a destacar.

6.13.4. Reconocimiento de formas por vibración

- **Concepto**

Este minijuego fue el primero en ser diseñado y representa un concepto bastante sencillo. El objetivo era diseñar un sistema de reconocimiento de objetos por medio del tacto, similar a cuando se intenta coger o agarrar algo en una habitación oscura o rebuscamos en un cajón, que no queda más remedio que ir palpando los objetos hasta reconocer el objeto buscado. Con este objetivo se implementó un minijuego en el que se presenta una forma en la pantalla solamente reconocible por medio de la vibración, que se debe investigar antes de continuar, como se observa en la Figura 34 - Concepto inicial del minijuego Reconocimiento de Formas. El minijuego engloba adivinar en serie tres formas distintas.

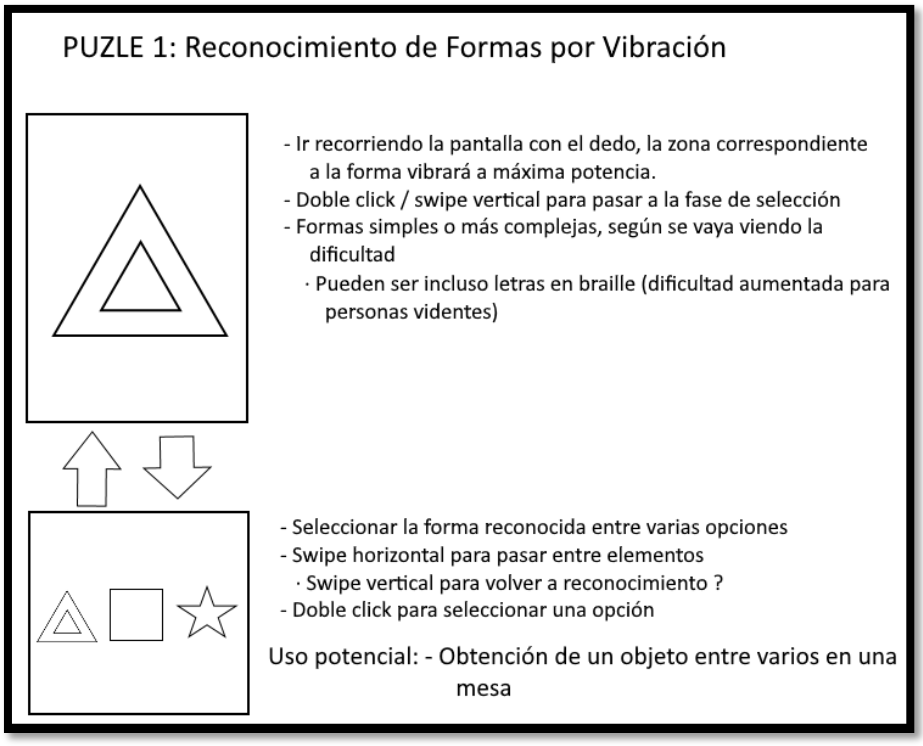


Figura 34 - Concepto inicial del minijuego Reconocimiento de Formas

- **¿Cómo se juega?**

Inicialmente se presenta al jugador la primera forma a reconocer. El jugador deberá ir moviendo el dedo por la pantalla del dispositivo móvil para encontrar las zonas en las que se activa la vibración y en las que no. Con estos trazados se espera que el jugador se vaya haciendo una idea de la forma a reconocer. Una vez se tenga una idea más o menos clara de qué forma podría ser, el jugador tendrá que hacer doble clic en cualquier lugar de la pantalla para pasar de la fase de reconocimiento a la fase de selección. En esta fase se desactivan las vibraciones del móvil y se le ofrecen al jugador varias formas como opciones, siendo solo una la correcta. También se incluye la opción de volver atrás y seguir reconociendo la forma. Mediante deslizamientos, el jugador irá moviéndose por las opciones, para finalmente elegir la opción que crea que es la correcta. En el caso de acertar, se avanza al siguiente nivel con una nueva forma a reconocer.

- **Desarrollo**

Para el funcionamiento del reconocimiento de las formas se han utilizado objetos "físicos" con dichas formas, ocultos a la vista debido al fundido negro general del juego. Los objetos detectan, por medio del motor de físicas y de la función reservada de Unity *OnMouseOver*, cuándo se da una pulsación sobre ellos mismos. Detectan también cuándo se entra o sale de su área, con una precisión elevada incluso en formas complejas (estrella, por ejemplo). Cuando se detecta una pulsación o entrada en la forma, se activa la función de vibración del dispositivo a máxima potencia, simulando una vibración continua. Por el contrario, si el jugador levanta el dedo o sale de la forma, se detienen las vibraciones. Los momentos en que una pulsación sale de la forma se detecta con otra función reservada que ofrece Unity, *OnMouseExit*.

En cualquier momento durante la fase de reconocimiento, si se detecta un doble clic, se cambia a la fase de selección. Durante este cambio se desactiva el objeto correspondiente a la forma con el fin de evitar que active vibraciones. Ya en la fase de selección, se le ofrece al jugador las posibles soluciones al problema actual de reconocimiento, además de una opción extra para volver a la fase anterior y seguir con el reconocimiento de la forma actual. Estas opciones, y las correctas en cada fase, están contenidas en una lista, de modo que son configurables desde el editor. La navegación por las opciones es consistente con el resto del juego, se espera que el jugador navegue entre ellas usando deslizamientos hacia los lados, indicando el foco con etiquetas de audio. Al hacer doble clic, se comprueba si la opción sobre la que se encuentra el jugador coincide con la solución (salvo en el caso de la opción de volver a la fase de reconocimiento). En caso de haber acertado, se le indica al jugador y se pasa a la siguiente fase de reconocimiento con una nueva forma o se concluye el minijuego si es la última.

- **Postproducción**

Este minijuego fue por el que más ciclos de postproducción pasó, debido a ser el de mayor dificultad. Esto también dificultó su equilibrado y ajustes. Con las primeras nociones de *feedback* recibido, se comprobó que el minijuego era casi imposible, y sólo se podía completar con prueba y error de las posibles soluciones, al no penalizar los errores.
Se achacó este problema a que los desarrolladores poseían lo que se llama la *Maldición del Conocimiento*, con la cual para ellos resolver el minijuego resultaba notablemente más sencillo que para un jugador que no hubiera participado en el desarrollo. Sabiendo esto se decidió retocar las formas, aumentándolas de tamaño para que se notasen más las diferencias entre ellas, además de cambiar las opciones posibles para que fuesen más dispares, facilitando al jugador escoger la opción correcta.

Tras estos cambios se volvió a mostrar el minijuego a nuevos probadores. Para sorpresa de los desarrolladores, en esta ocasión se diferenciaban dos grandes grupos de *feedback* recibido, uno positivo de probadores que no tuvieron mucha dificultad en pasarse el minijuego, y otro negativo de probadores a los que les seguía costando bastante reconocer las formas apropiadamente. Se observó que los integrantes del primer grupo, casi en su totalidad, tenían bastante más experiencia en el ámbito de juegos y tecnología que el segundo. Debido a que la intención del proyecto es que el juego sea para todo tipo de público, se decidió tener más en cuenta las opiniones del segundo grupo de probadores y dar otro ciclo de ajustes al minijuego con el fin de facilitarlo un poco más. Tras esto se propuso incluir, al comienzo de cada nivel del minijuego, un audio indicando las posibles opciones que se pudiesen escoger en el mismo. Esto con el fin de que el probador pudiera saber qué posibles formas debía intentar reconocer desde el principio, en lugar de intentar reconocer una forma genérica y luego pasar a las posibles opciones. También se volvieron a cambiar las opciones de cada nivel para que fuesen más dispares, con el fin de que el jugador distinguiera entre ellas de una manera más fácil durante el reconocimiento.

Con esta nueva característica implementada, se volvió a mostrar el minijuego a los probadores, y en esta ocasión finalmente todos ellos pudieron completar el minijuego sin quedarse atascados, de modo que se decidió que el desarrollo del minijuego pasaba a estar finalizado. Igualmente se sigue teniendo constancia de que es el más complicado de todos, aún tras todos los cambios realizados.

6.13.5. *Simon Says* Sonoro

- **Concepto**

Siguiendo con la investigación de posibles diseños de puzles o minijuegos basados en el sonido, se pensó en realizar una adaptación del clásico juego de mesa *Simon* (US Patente nº 4207087, 1977), que se basa en el juego tradicional *Simón Dice*. El juego consiste en un dispositivo con forma de disco, dividido en cuatro cuadrantes de diferentes colores, que se van iluminando de manera aleatoria y emitiendo a la vez un sonido específico para cada uno. Tras terminar una serie de estas activaciones, se espera que el jugador repita la misma secuencia en el orden correcto. Si el jugador acierta la secuencia, se comienza una nueva, añadiendo una activación adicional al final de la secuencia previa. De esta forma va aumentando la dificultad con cada secuencia acertada.

Con esta funcionalidad descrita la adaptación que se planteó pasaba por sustituir los elementos visuales del juego (botones de colores e iluminación de cada uno de ellos), por sonido 3D inmersivo en base al cual el jugador debería distinguir su procedencia espacial, siguiendo un esquema similar al que aparece en la Figura 35 - Concepto inicial del minijuego *Simon Says* Sonoro. De este modo, en el minijuego creado se tiene una versión un poco más simplificada del juego original, con tres posibles sonidos en lugar de cuatro, que se ubican a la derecha, a la izquierda y al frente del jugador. Cada uno de estos puntos reproduce un sonido distinto a los demás con el fin de facilitar la tarea de recordar la secuencia. En el diseño inicial, se consideraban cuatro zonas, pero la A y C acabarían combinadas, al no ser lo suficientemente distinguibles entre sí.

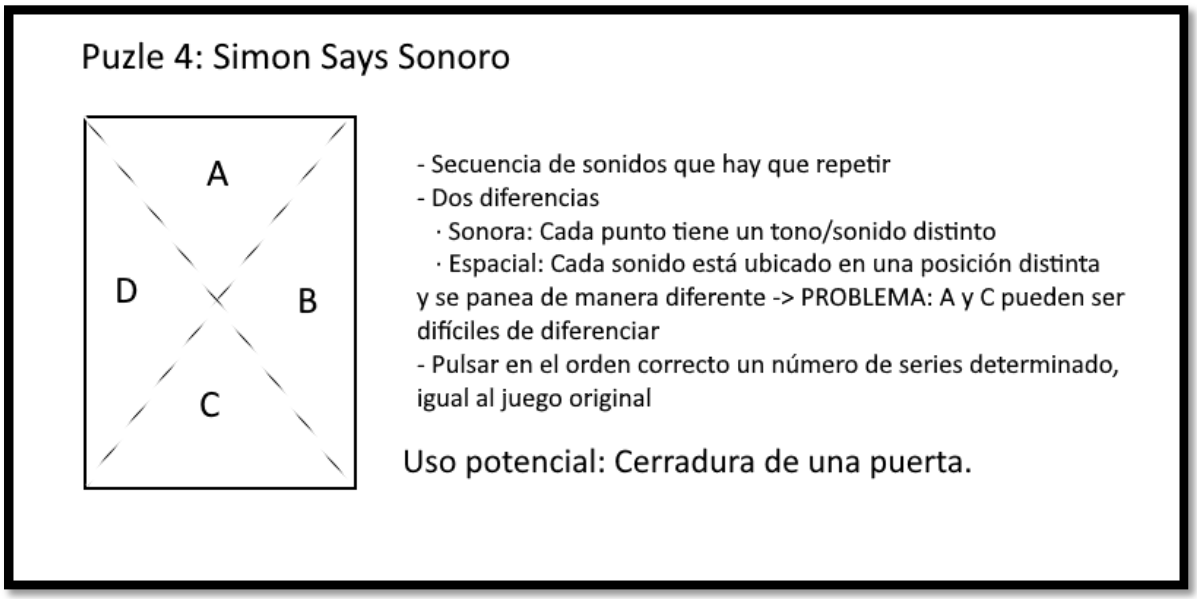


Figura 35 - Concepto inicial del minijuego *Simon Says* Sonoro

- **¿Cómo se juega?**

La forma de jugar es la misma que en el juego original, comenzando con una única activación de sonido procedente de una de las zonas y esperando posteriormente a que el jugador la repita correctamente. Con cada acierto aumenta el tamaño de la secuencia.
Los controles para seleccionar las posibles opciones son deslizamientos en la dirección de la que se cree que proviene el sonido.

- **Desarrollo**

Unity ofreció un gran alivio en la carga de trabajo a la hora de desarrollar este minijuego, gracias a la facilidad con la que permite crear un entorno de sonido 3D. De esta manera, en escena se colocó un número configurable de puntos desde los que se emitirían los distintos sonidos. Después, se creó un mánager que llevara a cabo la gestión de las mecánicas de juego. Este mánager posee un enumerado con las posibles posiciones del *Simon* (acorde a los puntos), y una lista con la secuencia de estos enumerados. Este enumerado existe principalmente para aligerar la tarea de asociación de los elementos de la lista con la posición que representan. Por otro lado, la lista es configurable desde editor, permitiendo diseñar la secuencia. Además, también cuenta con una opción para crear un recorrido aleatorio.

Cuando comienza la partida, se crean dos índices auxiliares, uno para indicar hasta qué posición de la lista se van a reproducir sonidos, y otro para llevar la cuenta del sonido actual que hay que reproducir. Durante el turno del jugador, se van ajustando los índices para comprobar si el jugador está realizando la secuencia actual correctamente, y aumentar el nivel en el caso de que logre llegar a su fin. Si el jugador falla en algún momento durante una secuencia, se detecta y notifica el fallo, y se vuelve a comenzar esa misma ronda, pasando el turno de nuevo a la máquina y reproduciendo la misma secuencia. En el momento que se detecta que se ha completado la lista completa, se trata como condición de victoria y se procede a activar los eventos consecuentes.

- **Postproducción**

Inicialmente, el minijuego sólo contaba con un sonido para las tres posibles direcciones del simón, y tras la primera ronda de *feedback* recibido, se tuvo constancia de que esto debía cambiarse. Se modificó la estructura del minijuego para soportar sonidos distintos para cada dirección y se crearon sonidos en base al original para cada dirección. Además, se probó con diferentes combinaciones de longitud y configuración de la secuencia a reproducir del Simón, para ajustar su dificultad. Tras los cambios, el *feedback* fue positivo, sin ningún problema a destacar.

6.14. Servidor en la nube

Hoy en día en cualquier aplicación o juego se necesita conexión a internet para disfrutarlos, o por lo menos para sacarle el máximo partido. Ya sea para compartir los resultados, para enviar mensajes, guardar la partida, etc. En este videojuego, la idea inicial era que fuese cien por cien online, es decir, que para poder jugar se necesitase conexión a internet e identificarse en el servidor con una cuenta previamente creada. Se sopesó lo que esta decisión conllevaría, y si iba ser más un perjuicio que una ventaja y al final se decidió por un sistema híbrido, es decir, con las dos opciones disponibles. Nada más entrar a la aplicación, en el menú principal se da la opción de jugar sin conexión o con conexión.

- *Jugar sin conexión*. Guarda y carga los datos con el progreso en el juego en local, en un archivo del dispositivo donde estás jugando.
- *Jugar con conexión*. Si eliges esta opción te llevará a identificarte en el servidor (iniciar sesión o crear una cuenta), donde se guardarán todos los datos, de manera que si cambias de dispositivo puedes retomar la partida en el mismo punto donde lo dejaste, sin perder nada.

Una de las primeras decisiones que se tuvieron que tomar era si se iba a alojar el servicio en una empresa dedicada a ello, teniendo así garantías de disponibilidad y seguridad, pero con el consiguiente sobrecoste económico; o si por el contrario se iba a alojar en algún ordenador de uno de los estudiantes, encargándose ellos de todo. Se sopesaron las dos posibilidades y al final se decidió que por lo menos durante la fase de desarrollo el servidor fuese alojado en un ordenador propio.

Elegida esta opción, se decidió dedicar un ordenador exclusivamente para este propósito para asegurar el máximo rendimiento posible y que no hubiera conflictos con otras aplicaciones o servicios similares. Para empezar con la configuración, se eligió la distribución LUBUNTU (Behling, 2008) de GNU/Linux basada en Ubuntu (Canonical Ltd. / Fundación Ubuntu, 2004) y a su vez basada en Debian (Proyecto Debian, 1993) como sistema operativo, ya que tenía todo lo necesitado e incluía el escritorio LXQt (The LXQt Team, 2013) en lugar de GNOME (Fundación GNOME, 1999)o KDE (Ettrich, 1996), ya que estos últimos consumen muchos más recursos. Al fin y al cabo, la interfaz solo se utilizaría para las primeras configuraciones y después directamente se gestionaría mediante SSH (Ylönen, 1995).

La configuración inicial se hizo como para cualquier usuario cotidiano. Se crearon el usuario *root* que no se utilizaría y un usuario limitado desde donde se gestionaría todo con los mínimos permisos necesarios en *sudoers*.

En este punto ya se tiene el ordenador funcionando, pero aún no da ningún servicio. Volviendo al origen del problema, lo que se quiere es servir determinados datos a través de internet y además que tenga persistencia en el tiempo. La mejor manera de abordarlo es con una base de datos dentro de las cuales existen dos grandes tipos:

- *Relacionales.* Son un tipo de base de datos donde los elementos tiene relaciones predefinidas entre ellos. Se organizan en tablas con filas y columnas en las cuales los tipos de datos también están predefinidos. El estándar para interactuar con bases de datos relacionales es el lenguaje SQL. Las principales ventajas de las bases de datos relacionales son:
    - Madurez. Es el sistema que más años lleva y existen una gran cantidad de soluciones para implementarlo. Esto significa también que es el sistema que más ha sido probado.
    - Reglas ACID. Atomicidad, consistencia, aislamiento y durabilidad.

- *No relacionales*. Son un tipo de base de datos donde los elementos no tienen relaciones previamente definidas. A priori no se sabe qué se va a almacenar ni si en el paso del tiempo van a cambiar los tipos de datos. El rey indiscutible de esta categoría es MongoDB. Las principales ventajas de las bases de datos no relacionales son:
    - Versatilidad. Ofrece gran versatilidad en cuanto a crecimientos y cambios sobre la forma en la que se almacena la información.
    - Crecimiento horizontal. Soportan estructuras distribuidas para un balanceado de carga añadiendo nuevos nodos.

Analizados los dos tipos de bases de datos, se decidió que el mejor sistema para la aplicación serían las bases de datos relacionales, ya que los tipos de datos que se manejarían serían siempre los mismos y estarían bien definidos.

Antes de continuar, se tuvo que tomar otra decisión importante: instalar un servidor de bases de datos que respondiera a consultas SQL directamente desde el puerto 3306, con lo cual había que prepararlas en el cliente; o crear un servidor Apache escuchando en el puerto 80, que tomase parámetros por peticiones *GET* y *POST* sencillos, y se procesase todo en el servidor con PHP y se mandasen las consultas ya preparadas al servicio de SQL en localhost. Al tener más conocimientos de PHP que de C# y además con la idea de hacer una página web posteriormente, se decidió la segunda opción.

A continuación, se descargó el paquete LAMPP de la página oficial comentado en el apartado Servidor en la nube y se instaló con el gestor de paquetes **dpkg** con la opción **-i**. Una vez instalado, para poner en funcionamiento los servicios había que hacer lo siguiente:

$ sudo /opt/lamp/lamp start

De esta forma se encienden y de forma análoga con el atributo *stop* se detienen. Con los servicios encendidos y escuchando en los puertos correspondientes, lo que se necesitaba era darle visualización al servidor en internet. Para ello, se decidió usar un dominio que se tenía de proyectos anteriores. Dicho dominio apunta al enrutador de la casa, por lo tanto, es necesario hacer una redirección de puertos.

Al conectar el ordenador por cable a la red interna de la casa, el protocolo **DHCP** se encarga de asignarle una dirección **IP**. Para evitar posibles conflictos se configuró para que siempre se le asignase la misma, en este caso fue la 192.168.1.57. A continuación, se elige un puerto del enrutador, por ejemplo, el 4398, fuera del rango de los *Well-known ports* y sin otro servicio escuchando en ese momento, para que todas las conexiones que se hagan ahí sean redirigidas al puerto 80 (**http**), predeterminado de **Apache** junto con el 443 (**https**), donde está escuchando nuestro servidor. Esto se puede hacer con *iptables* por la línea de comandos o con alguna interfaz, modificando la tabla **NAT**.

Una vez que el servidor es visible en internet, el siguiente paso fue diseñar la base de datos con **SQL**, la cual ha ido depurándose a lo largo de varias versiones hasta la actual, su esquema se puede observar en la Figura 36 - Esquema de la base de datos.

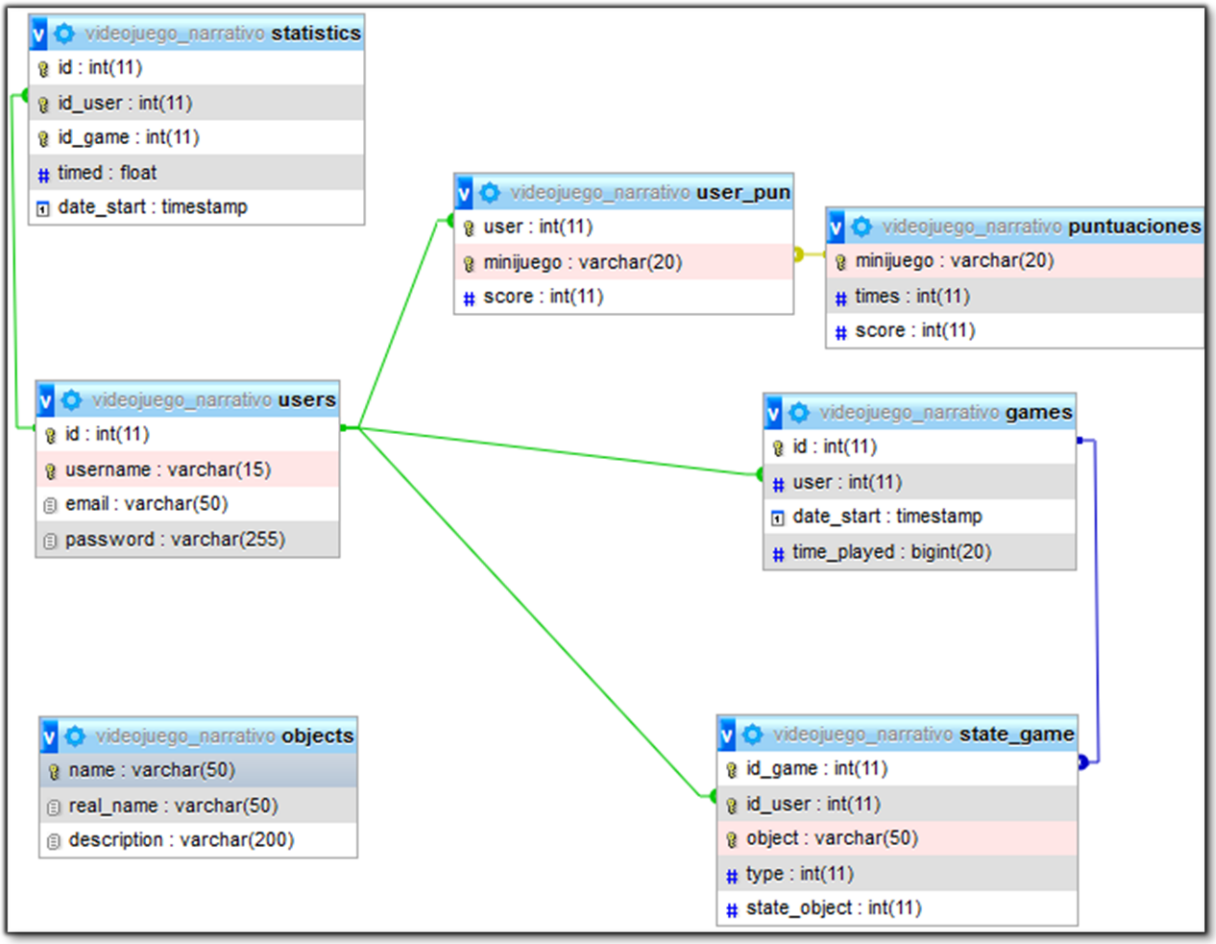


Figura 36 - Esquema de la base de datos

A continuación, se procede a explicar cada una de las tablas de la base de datos:

- **Users**. Tabla donde se almacena toda la información de los usuarios de la aplicación. Contiene los siguientes campos:
    - ID. Identificador único para cada usuario. Sólo se usa para control interno de la aplicación. Tipo de datos entero. Tiene el autoincremento activado.
    - Username. Nombre de usuario con el cual se identificará en la aplicación. También es único. Tipo de datos cadena de caracteres.
    - Email. Dirección de correo electrónico a través de la cual se podría establecer contacto con el usuario. Tipo de datos cadena de caracteres.
    - Password. Hash de la contraseña con la cual se identifica el usuario.

- **Games**. Tabla donde se almacena toda la información relacionada con la gestión de partidas de los usuarios. Contiene la única partida en curso posible del usuario. Contiene los siguientes campos:
    - ID. Identificador único de la partida del usuario. Tipo de datos entero. Tiene el autoincremento activado.
    - User. ID del usuario de la partida. Clave externa. Tipo de datos entero.
    - Date_start. Fecha en la que se inició la partida. Tipo de datos fecha.
    - Time_played. Tiempo jugado en esa partida. Tipo de datos gran entero.

- **State_game.** Tabla donde se almacena el estado de una partida. Con estado se refiere a los objetos en escena con los que el jugador interactúa. Contiene los siguientes campos:
    - Id_game**.** Identificador de la partida la cual estamos guardando la información. Clave externa. Tipo de datos entero.
    - Id_user**.** Identificador del usuario de la partida. Clave externa. Tipo de datos entero. Aunque pueda parecer redundante, se puso para que en el futuro soporte partidas de varios usuarios colaborando para conseguir el objetivo. En tal caso habría que diferenciar de qué usuario estamos guardando el estado de los objetos, ya que cada usuario podría tener distintos.
    - Object**.** Objeto del cual se va a guardar el estado. Tipo de datos cadena de caracteres. El nombre es único.
    - Type**.** Entero que indica de qué tipo es el objeto: escena o inventario.
    - State_object**.** Estado del objeto. Se creó un convenio de codificación entre el servidor y la aplicación. Tipo de datos entero.

- **Statistics.** Tabla donde se almacenan las partidas terminadas de los usuarios. Esta tabla se utilizará principalmente en la página web para mostrar información de tus partidas terminadas. Contiene los siguientes campos:
    - ID. Identificador de la estadística. Tiene el autoincremento activado. Tipo de datos entero.
    - Id_user. Identificador del usuario al que pertenece esta entrada. Clave externa. Tipo de datos entero.
    - Id_game. Identificador de la partida. No es clave externa porque en la tabla *Games* solamente está la partida actual y en *Statistics* se quiere guardar la historia de todas las partidas.
    - Timed. Tiempo que se ha tardado en terminar el juego.
    - Date_start. Fecha en la que se inició la partida.

- **Objects.** Tabla donde se almacenan los objetos del juego, tanto de escena como de inventario. Es una tabla aislada porque se pensó que era la mejor opción ya que los datos de esta tabla sólo se utilizarían en la página web para mostrar información de los objetos. Contiene los siguientes campos:
    - Name. Nombre con el que se identifica al objeto dentro de la aplicación. Tipo de datos cadena de caracteres.
    - Real_name. Nombre con el que se muestra el objeto al usuario. Tipo de datos cadena de caracteres.
    - Descripción. Pequeña descripción del objeto que se muestra en la web.

- **Puntuaciones.** Tabla donde se almacenan las puntuaciones que hacen los usuarios desde la página web sobre los minijuegos, del uno al cinco. Contiene los siguientes campos:
    - Minijuego. Nombre del minijuego al cual se está puntuando. Tipo de datos cadena de caracteres.
    - Times. Número de veces que se ha puntuado el minijuego. Sirve para poder sacar la nota media. Tipo de datos entero.
    - Score. Nota media del minijuego. Tipo de datos entero.

- **User_pun.** Tabla donde se almacena qué usuario ha puntuado qué juego para impedir que se vote varias veces y además para recordar la puntuación que le dio. Contiene los siguientes campos:
    - User. Usuario que hace la puntuación. Clave externa. Tipo de datos entero.
    - Minijuego. Minijuego el cual se está puntuando. Tipo de datos cadena de caracteres. Clave externa de la tabla *Puntuaciones.*
    - Score. Puntuación que se le ha dado al minijuego. Tipo de datos entero.

Seguidamente, se utilizó **phpmyadmin** para importar la base de datos y configurar los usuarios de esta, los cuales van a estar en el código de la aplicación PHP para poder autenticarse y hacer las consultas.

El siguiente paso es implementar los archivos PHP que van a recibir las peticiones *GET* y *POST* y las clases de los objetos. Podemos dividir los archivos PHP en tres grandes grupos: videojuego, página web y comunes.

- **Videojuego.** Las clases de PHP de este grupo estarían en la raíz del directorio de escucha de Apache, es decir, en htdocs. De esta forma, detrás de la *url* no hay que poner nada más que el archivo ya que es donde se busca por defecto. Estos archivos simplemente validan los datos proporcionados por POST con un pequeño procesado en algunos casos y se los mandaría a una de las clases comunes encargada de hacer los cambios en la base de datos.

  <url>:<puerto>/<archivo.php>

- **Comunes.** Son aquellos archivos PHP que trabajan a más bajo nivel encargándose de leer y escribir en la base de datos. Estos archivos son llamados tanto desde el videojuego como la página web. Al principio se implementó un archivo por cada tabla creada en la base de datos, pero al finalizarlos se observó que no había ninguna función útil y que lo único que podían tener eran los atributos. Por poner un ejemplo, en la tabla *Objects* que tiene los objetos de forma estática no tiene mucho sentido crear una clase para devolver los valores puesto que se puede hacer directamente. Por lo tanto, se decidió eliminar algunos de ellos ya que no se iban a utilizar y, aunque sí se iban a guardar datos en tabla de la base de datos, la clase PHP no iba a tener ninguna funcionalidad. Debido a esto solamente permanecieron las clases *Game.php*, *User.php* y *Minijuego.php*. Pero además de estos, también se reutilizaron otros archivos PHP por su similitud en funcionalidad desde el videojuego y la página web como por ejemplo *login.php*.

- **Página web.** Son aquellos archivos que se encargan de la funcionalidad de la página web como son la autenticación, el registro de un nuevo usuario, la posibilidad de ver datos de tu perfil, mantener los datos de la sesión de un usuario, etc. A estos archivos PHP los acompañan gran cantidad de archivos HTML con el contenido de la página web, CSS con el estilo, JavaScript con el comportamiento de los elementos, etc. Todos ellos se explican en sus apartados correspondientes de Tecnología utilizada y Página web.

En la Figura 37 - Esquema simplificado del servidor, se representa con un alto nivel de abstracción la organización del servidor. No están todas las clases de PHP representadas porque no añadirían información extra a la estructura.

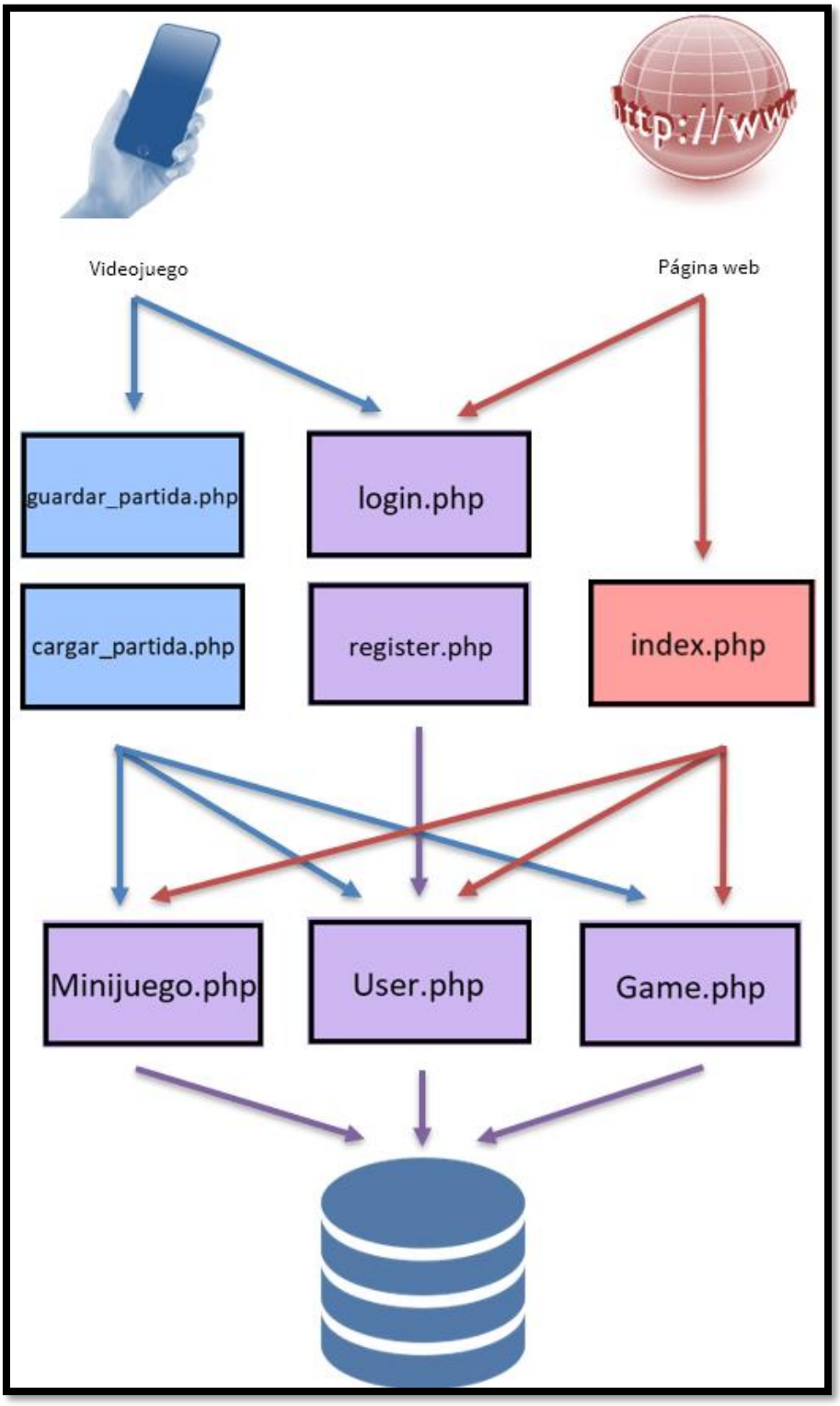


Figura 37 - Esquema simplificado del servidor

En este punto ya se tiene la parte del servidor terminada, había que hacer la parte del cliente que se conecte al servidor y envíe los datos. Al elegir **Unity**, el lenguaje de programación era **C#**, por lo que para realizar las peticiones se hizo una clase en este. Para lanzar las peticiones y recoger su resultado se utilizó la librería *HttpWebRequest*.

El modo de hacerlo fue sencillo, primero se indica la *url* del servidor donde se tiene que conectar, y después se añaden creados mediante clave-valor tanto el tipo como los parámetros de la petición. Entonces la petición es enviada, y se espera la vuelta de la respuesta, que queda capturada. Esta respuesta es entonces devuelta para poder ser procesada por el componente que hubiera hecho la petición.

En la clase implementada, para cada tipo de llamada al servidor se incluyó un correspondiente método asociado a la llamada de ese archivo. Son estos métodos los que se llaman desde fuera de esta clase. Esta clase se hizo estática para hacerla accesible al resto de elementos del juego sin requerir una instancia en escena.

Estos métodos tienen como parámetros de entrada la información necesaria para realizar la llamada al servidor (prácticamente todas requieren el nombre de usuario, pero según el caso se añade más información: estado del juego serializado al guardar, tiempo de juego al guardar esta estadística, etc.). Como parámetro de salida todos ellos tienen una cadena de texto, en la cual se devuelve la respuesta del servidor. Estos métodos sirven como intermediarios, ya que después de crear la cadena de texto que constituye la petición, esta se envía a un método común, que es el que se encarga de lanzarla con *HttpWebRequest*, esperar y devolver la respuesta, la cual se procesa en algunos casos antes de devolverla.

Para unificar las respuestas del servidor se tomó la cadena “0” como indicador de que se había realizado la operación sin fallos y “1” para indicar fallo. En caso de contemplarse tipos de fallos específicos, estos se codificaban con los números consecutivos. Las llamadas que devuelven información del servidor, en concreto cargar partida, devuelven esta información después del indicador de éxito, que se omite a la hora de procesar esa información recibida.

6.15. Persistencia y carga de los datos de guardado

Se consideró importante a la hora de implementar la aplicación asegurar que cualquier avance relevante del jugador fuera guardado de forma segura. Al trabajar en una plataforma móvil, es muy habitual el cambiar de aplicación activa, minimizar la actual sin cerrarla, cerrarla sin pasar por menús de salida y otros comportamientos que no suelen ocurrir en otras plataformas, especialmente en consolas, que suelen tener mucho control sobre el flujo de las aplicaciones.

Concretamente al trabajar en **Android** ya se sabía desde el principio que no bastaría con guardar al detectar un cierre de aplicación, por lo comentado anteriormente. Por ello se decidió tomar el enfoque de guardar los datos cada vez que hubiera un cambio relevante.

Para esto se consideraron como cambios relevantes los momentos en que el jugador obtiene un objeto o supera algún obstáculo o minijuego. En estas ocasiones se realiza el guardado de su progreso, que se codifica como el número de habitación en la que se encuentra y la lista de objetos tanto en escenario como en su inventario, con sus correspondientes estados, todo ello serializado para poder ser leído posteriormente. Este guardado se realiza tanto en el modo sin conexión como en el de con conexión.

El juego da la opción de jugarlo enteramente *offline*, por ello, tuvieron que realizarse ciertas gestiones para el funcionamiento de tanto este modo como el *online*. Se considera que mientras que no se haya hecho *login* (bien en ese momento o bien anteriormente y este se encuentra guardado en el archivo local) se está jugando de forma *offline*.

- **Modo *offline:*** Los datos siempre son cargados de un archivo local en el dispositivo y se omiten todas las llamadas al servidor. No se requiere un usuario para jugar y no se guarda ningún dato de este tipo.

- **Modo *online*:** Tiene preferencia la carga de datos de la web, asociados al usuario. Aun así, también se realiza el guardado local, para permitir continuar la partida en modo *offline.* En el archivo local también se guarda la información de *login*, para evitar la necesidad de introducir estos datos reiteradamente. Además, en este modo los datos de guardado incluyen pequeñas estadísticas.

La ventaja del modo *online* es la portabilidad, ya que permite mantener el progreso asociado a un usuario al migrar de dispositivo una vez se inicia sesión. Aunque el juego es de duración corta y lo más habitual es completarlo en una o dos sesiones de juego a lo sumo, se consideró importante realizar estas funcionalidades, ya que son algo básico a nivel de usabilidad en todo tipo de aplicaciones y especialmente teniendo en cuenta la accesibilidad. Esta consistencia de los datos de guardado evita fricciones en el uso de la aplicación.

6.16. Página web

Las páginas web son el escaparate de cualquier tienda o aplicación en la web. Por ello, debe ser clara, concisa y agradable. Además, es un buen método para dar a conocer el producto en cualquier lugar del mundo.

6.16.1. Primera implementación

En un principio, la idea era hacer la página web partiendo de un diseño ya implementado y posteriormente adaptarlo al contenido de nuestro videojuego, ya que al ser un proyecto de accesibilidad se pensó que la dificultad no debía ser el hacer una página web, si no hacerla accesible. Se buscó en varias páginas que proporcionaban diseños ya implementados y tras barajar varias opciones, se optó por html5up.net (HTML5UP, 2020) en la cuál había bastantes diseños de páginas web y muy variados. Mediante una encuesta a los miembros del equipo se escogieron los más apropiados y se pusieron en común. Seguidamente se realizó una votación para el diseño final y se escogió uno llamado “Hyperspace”.

Una vez elegido el diseño se fijaron las secciones que iba a tener la página. Al principio estas eran el inicio, los minijuegos, los objetos y la parte de iniciar sesión a o crear una cuenta. Estas secciones sufrirían algunos cambios más adelante. Una vez fijadas las secciones se fue completando el contenido de cada sección. Primero este avance se limitó a ir poniendo en cada sección la información que debería ir con un breve resumen.

Sin embargo, cuando se empezó a implementar la accesibilidad, se llegó a la conclusión de que la mejor opción era hacer la página web desde cero para tener en cuenta la accesibilidad desde un principio, cosa que la web prediseñada no contemplaba.

6.16.2. Web desde cero

La tecnología empleada ha sido HTML5, CSS, Javascript y JQuery para *front-end*, encargado de la presentación e interacción de la web, y PHP para *back-end*, dedicado a la parte de funcionalidad y lógica. También se ha utilizado AJAX, una herramienta de *front-end* usada para la comunicación con *back-end*. Además, y como se verá más adelante, se ha utilizado Bootstrap, uno de los *frameworks* de *front-end* más utilizado. Por ejemplo, la página web de la NASA lo usa. (Bootstrap, 2020)

Aunque era desde cero, se utilizó la estructura pensada con la plantilla, como las secciones en las que se iba a dividir la página, alguna información que ya se había añadido en la página anterior y algunas de las fotos que se habían escogido para decorarla.

Lo primero fue hacer la cabecera y el pie, ya que iban a ser iguales para todas las páginas a las que se accediera. En la cabecera se pusieron las secciones que se habían definido anteriormente, pero hubo alguna modificación como la desaparición de la pestaña objetos. Esto fue debido a que en un principio se pensó mostrar todos los objetos del juego a las personas que visitasen la página, pero se vio más lógico que a cada usuario del juego se le mostrasen solamente los objetos que tenía en posesión una vez iniciase sesión.

Después se hizo el pie, en el que habría información sobre el grupo de trabajo, el nombre de los miembros del equipo, las diferentes formas de contactos (un correo creado para el TFG) y la ubicación, que en este caso sería la dirección de la Facultad de Informática de la Universidad Complutense de Madrid.

Una vez terminadas estas dos partes, se pasó a hacer el inicio de la página web. Este se dividió en dos partes. Por una parte, estaba la información relativa al videojuego. Esta información engloba las dos características más importantes del videojuego, la accesibilidad y la funcionalidad online gracias al servidor en la nube. La otra parte del inicio estaba formada por los minijuegos incluidos en el propio juego. Para todos los elementos de ambas partes, se incluyó una foto que lo represente, además de una pequeña descripción y la opción de acceder a otra página en la que se ofrece información ampliada.

Una vez terminado el inicio se crearon las páginas que incluyen más información sobre la accesibilidad, la nube y los minijuegos. El desarrollo de estas páginas fue sencillo ya que se parecen mucho unas a otras.

Después se pasó al diseño relacionado con la parte de los usuarios. En esta etapa se hicieron los diseños de la página de *login* y de registrarse.

Llegados a este punto se debatió la manera en la que se mostraría a los usuarios sus estadísticas. Surgieron dos ideas principales, una era mostrarlas en el inicio al principio del todo y la otra era en la cabecera añadir una opción adicional para acceder a tu perfil, que redirigiría a otra página en la que se mostrarían las estadísticas. Esta decisión fue más fácil que otras anteriores, ya que la elección de la segunda opción fue unánime. De esta manera se añadió código a la cabecera para que detectase si se había iniciado sesión y así mostrar la nueva opción para acceder al perfil. En la nueva página se introdujeron dos tablas con información del usuario. La primera con las estadísticas de cada partida con su identificador, la última vez que jugó y el tiempo que tardó en terminarla. La segunda tabla tendría los objetos conseguidos en la partida actual del usuario, con el nombre y descripción de cada uno.

Terminada esa parte ya solo quedaba ir mejorando ciertas partes del diseño. Se introdujeron nuevas fotos, animaciones para decorar la página y se hicieron algunos pequeños cambios de diseño que no afectaron a la funcionalidad.

Por último y con el fin de dar una mayor interacción al usuario en la página web se añadieron los botones de eliminar partidas ya acabadas y eliminar la partida actual junto con los objetos de esta. También se ideó un sistema de votación con estrellas para puntuar los minijuegos. Estas estrellas van del uno al cinco y solo se puede puntuar si el usuario esta registrado.

Las imágenes e iconos se han sacado de una conocida página web llamada Flaticon (Flaticon, 2020), aunque después muchas imágenes se han personalizado e incluso convertido en animaciones.

Se pueden encontrar imágenes de la web en el Anexo - Imágenes de la página web.

6.16.3. Web accesible

Desde el primer momento se acordó que la web debía ser accesible, puesto que iba a ser accedida por el mismo público que el videojuego. Desde entonces la parte de accesibilidad ha sido la más importante ya que era la que se desconocía en un principio.

Lo principal a entender cuando se habla de web accesible es cómo navegan los usuarios por una web. En el caso de las personas ciegas, lo hacen usando un lector de pantalla que va dictando los elementos de la web. Esto hace que la importancia de cómo estén definidos estos elementos y cómo se relacionan entre ellos, sea máxima.

Para la implementación se han seguido los pasos de las WCAG (Web Content Accessibility Guidelines), que proporciona pautas de accesibilidad para el contenido web. Fue elaborada por la W3C (World Wide Web Consortium) mediante la WAI (Web Accessibility Initiative. El 5 de junio de 2018 se publicó la última y más nueva recomendación, las WCAG 2.1.
Estas pautas se organizan en torno a cuatro principios que se deben cumplir para que una página web sea accesible:

- Perceptible: el contenido web debe ser accesible y disponible para los sentidos: vista, oído y/o tacto. Debe ser adaptable y se deben poder distinguir sus elementos por separado. Esto, como se verá más adelante, se traduce, por ejemplo, en dar alternativas para el contenido media.
- Operable: la navegación, control y los formularios de la web deben poder ser operables. Por ejemplo, toda la funcionalidad, además de poder ser controlada con un ratón, debe poder ser controlada también mediante el teclado.
- Comprensible: el texto y los componentes de la página web se deben poder entender por sí mismos.

- Robusta: para que una web sea robusta significa que debe serlo lo suficiente para que pueda interpretarse por una gran variedad de usuarios, incluyendo por el ejemplo el uso de las tecnologías de apoyo, como los lectores de pantalla. (W3C, 2018)

Durante el desarrollo de la web se irán mencionando apartados de las WCAG por lo que se procede a mencionar a que corresponde cada apartado. Todas las pautas del apartado 1 de las ayudan a que la página sea más perceptible. Los apartados descendientes del 2 ayudan a que sea más operable. Y los apartados descendientes del 3 ayudan a que sea más entendible. Por último, los apartados descendientes del 4 ayudan a que la página sea robusta.

Además de estos cuatro principios, las WCAG reconocen tres niveles de conformidad: A, AA y AAA, siendo esta última la más completa en cuanto a accesibilidad. Además, también se proporcionan prácticas con ejemplos que ayudan a mejorar aspectos de accesibilidad de las páginas web. Estas buenas prácticas se han utilizado como referencia a lo largo del desarrollo de la página web. (WAI-ARIA, 2019)

Para tener adquirir una base sobre accesibilidad, se encontró un curso gratuito en Udemy (Udemy, 2015), impartido por profesores de la Universidad de Alicante. Junto a esto, se encontró la web del curso con sus contenidos, por lo que ambos recursos se utilizaron al inicio. Al principio del curso se menciona a Jim Thatcher, un ingeniero de ciencias de la computación, que ha escrito libros de accesibilidad web y tiene un gran reconocimiento internacional en el área de la accesibilidad. J.Thatcher, especifica que la accesibilidad web se podía dividir en varias categorías: contenido, navegación e interacción. (Thatcher, 2006)

Se puede ver una relación clara entre los principios descritos por las WCAG y las categorías definidas por Jim Thatcher, siendo estos últimos más específicos. Además, se podría añadir otra categoría llamada diseño, que se encarga de que la presentación de la página sea también accesible. Debido a que el curso online también se estructuraba de esta manera, se decidió dividir la implementación de la web en esas secciones.

Contenido accesible

El principal problema al que se enfrentan los lectores de pantalla cuando dictan una web son las imágenes. Esto se debe a que no siempre se ofrece una descripción de la imagen. Para ello, y tal y como se indica en el apartado 1.1.1 de las WCAG, se debía utilizar el atributo '*alt*' (*alternative*, en inglés) dentro de una etiqueta de imagen, con una descripción.

Dentro de las WCAG también se referencia a las imágenes decorativas que no añaden ninguna información extra. Los iconos son un ejemplo claro de imágenes de este tipo. Estos iconos están colocados junto con su subtítulo particular, por lo que además de no añadir información, son redundantes. Para estos casos se debe poner la alternativa vacía, para evitar que sea leída por el lector de pantalla.

El segundo elemento más problemático son los enlaces. Suele ocurrir que en muchas páginas web se tiene el texto 'Leer más' en un enlace. Esto puede dar problemas de contexto ya que el lector de pantalla leerá literalmente el contenido del texto del enlace y puede que este texto no sea suficientemente explicativo. El enlace no debe depender de su contexto, sino que debe quedar claro solo con el texto que contiene, como bien indica el apartado 2.4.4. Durante el desarrollo de la web se encontró este problema, ya que se quería tener un enlace de 'Leer más' en la página principal en las tarjetas de minijuegos, para que cada una llevara a la página del minijuego correspondiente. Para solucionar esto se pueden utilizar atributos en la etiqueta del enlace que proporcionen una descripción adicional. Una opción el usar el atributo '*title*' pero algunos lectores de pantalla tienen dificultad para leerlo, por lo que se descartó esta opción. La opción que se tomó y la recomendada por el apartado 2.4.9 es el uso del atributo '*aria-label*' junto con la descripción del propósito del enlace. Esta solución forma parte del nivel AAA de conformidad. En nuestro caso se añadió dicho atributo con "leer más sobre" junto al minijuego en cuestión.

Desde las WCAG se alerta también de la confusión que puede generar usar *links* como botones, ya que no son links al uso y al no tener un destino, el usuario que utilice la web con el lector de pantalla lo leerá como un enlace vacío. En un principio se tenían varios enlaces vacíos que se utilizaban como botones, pero al ver la recomendación sobre esto se cambiaron todos por botones.

En una página web es importante que el usuario tenga una visión global de ella para saber dónde encontrar las cosas y cómo navegar. Esto no solo se puede hacer con el diseño, sino que también hay una forma de hacerlo para los lectores de pantalla. Como se señala en el apartado 2.4.10 de las WCAG, los encabezados se utilizan para la organización del contenido, es decir, el elemento <h>, que tiene opciones del uno al seis, siendo el primero el más importante. El lector de pantalla hará uso de estos encabezados para proporcionar una estructura lógica de los contenidos. Por ello, se deben usar de manera adecuada y siguiendo unas pautas. Se deben usar estos elementos únicamente para encabezados y no para cosas que no lo sean. Los encabezados deben estar bien ordenados, es decir, primero debe ir el uno, luego el dos y así sucesivamente. No se debe dar la opción de saltar de uno a otro. Siguiendo esto, se consigue el nivel AAA de conformidad establecido por las WCAG.

Además, se facilita a los lectores de pantalla la posibilidad de crear una correcta jerarquía de la página. Siguiendo esas pautas y para comprobar la buena implementación de estas se optó por utilizar una extensión de **Mozilla Firefox** (Mozilla Foundation, 2002) muy conocida entre los desarrolladores web llamada **Web Developer** (Pederick, 2020). Esta extensión, entre otra mucha funcionalidad, permite generar el esquema que se ve en la Figura 38 - Jerarquía de los encabezados. Se fue probando en cada página hasta que se determinó que todas las jerarquías de todas las páginas estaban correctas.

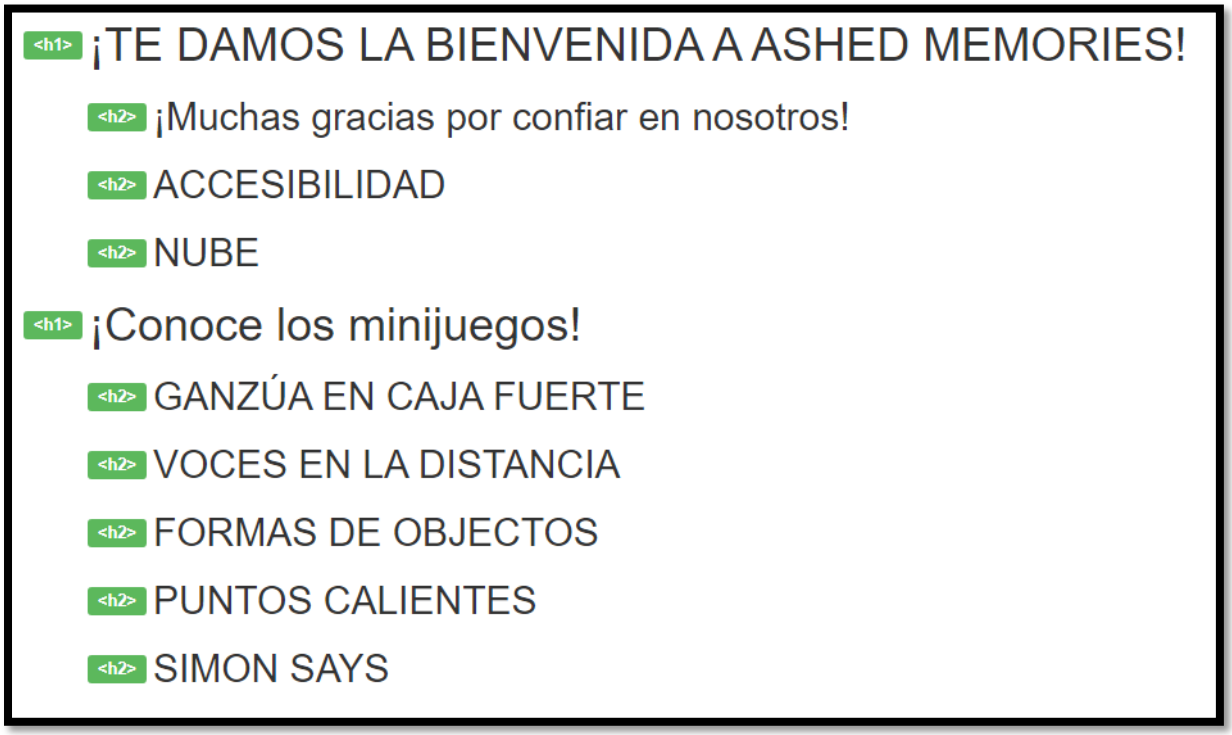


Figura 38 - Jerarquía de los encabezados

Además de los encabezados también es importante la manera en la que dividimos el contenido. Cada parte del contenido tiene un propósito y así se debe hacer saber al lector de pantalla mediante el uso de secciones, como dice el apartado 1.3.6. Esta estructuración del contenido se debe realizar haciendo uso del atributo '*role*', indicando el rol que tiene ese apartado, o bien usando *landmarks* propios de **HTML5**, como <*nav*> o <*footer*>. Dado que todos los navegadores actuales soportan **HTML5**, se optó por la segunda opción, siendo además la recomendada por la **WAI**. En nuestras páginas se utilizaron los siguientes elementos:

- <*nav*>, para indicar la sección que se encarga de la navegación por la web.
- <*header*>, que contiene el logo, y el nombre del videojuego.
- <*main*>, que denota el contenido principal.
- <*footer*>, el pie de página.

También se han usado <*form*> para los formularios y <*section*> para dividir las secciones dentro del contenido principal. Esta estructuración lógica de la web tiene nivel AAA de conformidad.

Otro aspecto que se debía tener en cuenta era el lenguaje. Mediante la declaración de este, se ayuda al lector de pantalla a saber qué tipo de voz y qué acento debe usar al dictar el contenido de la página. Como bien explica el apartado 3.1.1 de las WCAG, se debía usar el atributo '*lang*' dentro del elemento <*html*> al inicio del nuestro código HTML junto con el lenguaje elegido. Además de indicar el lenguaje, también se puede indicar la región del lenguaje. Por ejemplo, 'es-419' corresponde al español de américa latina. En nuestro caso, y por razones obvias se decidió utilizar el mismo idioma que el del videojuego: el castellano, sin especificar ningún tipo de región. Esto se hizo en todas las páginas que fueran accesibles por el usuario. Este apartado también explica que cualquier cambio en el lenguaje del contenido de la página debe estar especificado con su correspondiente atributo '*lang*' en el elemento en el que se cambie de idioma. Dado que en la página principal de nuestra web se utilizaban palabras como Unity o Android, que no son propias del lenguaje español y se suelen leer en inglés, se tuvo la duda de si se debía especificar el cambio de idioma. En un principio se consideró que así sería, sin embargo, el apartado 3.1.2 señala que en el caso de nombres propios, expresiones técnicas o palabras que no tienen un determinado lenguaje no se debe cambiar el idioma. Por lo tanto, no se hizo.

Diseño accesible

A la hora de hacer el diseño de la página web se debió tener en cuenta que la accesibilidad no solo comprende a personas con discapacidad visual total, sino que también hay personas que tienen problemas visuales parciales o limitados. Por lo tanto, para hacer que el diseño de la web fuera lo más accesible posible se tuvieron varios aspectos en cuenta.

Lo primero fue el color. El color es muy importante, no solo por el aspecto visual que otorga a la página, sino también porque una mala elección, con un contraste bajo con el texto, por ejemplo, puede dificultar la lectura a algunas personas. Para la elección del color de fondo y el de las fuentes se usó una página web llamada colorsafe.co (Colorsafe, 2020). Esta herramienta indica el coeficiente de contraste que tiene un texto, relacionado con el color del fondo, siguiendo los estándares determinados por las WCAG. Además, también indica el nivel de conformidad del contraste. Desde el primer momento se tuvo claro que se buscaría un diseño de página que tuviera el nivel AAA. Así pues, siguiendo los colores del logo, se eligió el morado como color principal y el rojo como secundario.

Se valoraron varios tonos de morado y varios colores de texto. Por ejemplo, se dudó entre el color negro o blanco del texto. Pero en la página antes mencionada se indicaba que el blanco tenía un mayor contraste con el tono de morado elegido. Por lo tanto, se acabó eligiendo el tono de blanco que aparece en la Figura 39 - Elección de color en Colorsafe.co. Todos los colores tanto del texto como del fondo se eligieron de acuerdo con el nivel AAA de conformidad.

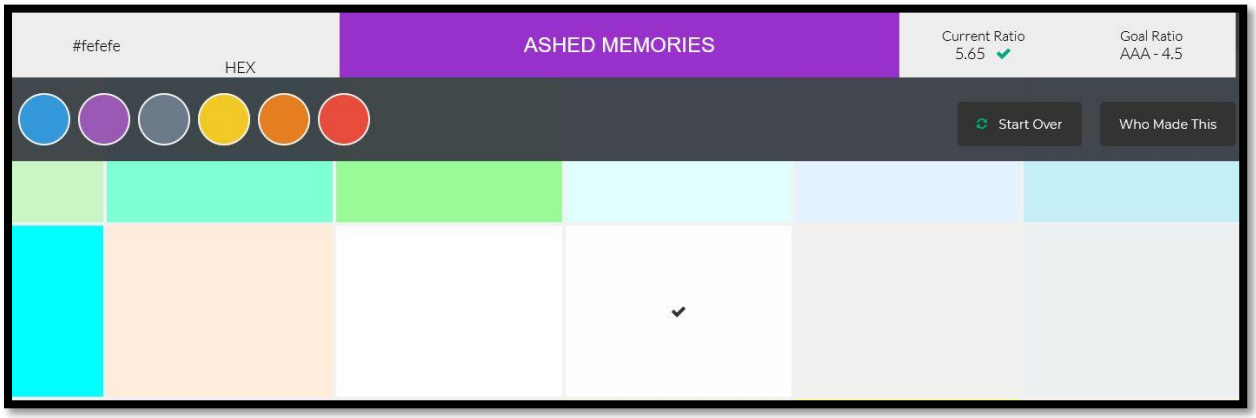


Figura 39 - Elección de color en Colorsafe.co

Además de lo anterior, se incorporó un botón de alto contraste que cambiaba los colores de la página a unos con un contraste mucho mayor. Estos colores son el negro, que se utiliza como fondo, y el amarillo, para el color de texto o colores secundarios. Se puede ver el resultado de poner la página web en contraste alto en la Figura 40 - Página web en modo alto contraste. Este botón se implementó siguiendo las recomendaciones de accesibilidad a la vez que se le otorgaba un aspecto estético agradable. (Pickering, 2017)

Figura 40 - Página web en modo alto contraste.

En un principio esto se realizaba cambiando el estilo desde **JavaScript** al pulsar el botón de cambio de contraste. Cuando el botón se pulsaba, se llamaba a función de **JS** que mediante la función '*.css*' de **JQuery** se añadía la propiedad de **CSS** que se deseara. Sin embargo, durante la investigación se vio que muchas personas con discapacidad visual importan sus propios estilos en las páginas web. Así, estos usuarios no se tienen que preocupar por los diseños de las páginas ya que usan diseños y colores que ellos ya saben que les funciona.

Tal y como estaba hecho entonces el sistema de cambio de contraste, los usuarios no podían importar sus estilos, ya que los colores se aplicaban mediante **JS**, es decir, en línea. Esto hacía que cualquier estilo de color importado por un usuario fuera sobrescrito por el *script*. Se tomó por lo tanto la decisión de añadir o quitar clases de **CSS**. Cuando se pulsa el botón se llama a una función de **JS** que añade una clase con un color distinto a los elementos, haciendo así que si un usuario importa sus estilos pudiera ver la página a su gusto.

Una herramienta muy conocida y que no solo es utilizada por usuarios con discapacidad visual se llama **Stylish** (Stylish, 2020), que cuenta con más de un millón de descargas en **Google Chrome**. Se trata de una extensión para navegadores mediante la cual puedes cambiar el tema de cualquier sitio web. En esta extensión cada usuario puede subir sus propios estilos y la comunidad puede descargárselos y aplicarlos a cada sitio que desee. Se procedió a instalar algunos diseños subidos por la comunidad y comprobar que funcionaban en nuestra web. Por ejemplo, al aplicar un estilo llamado *Global Dark Style*, que cambia los colores por unos tonos oscuros, la página webs se vería como en la Figura 41 - Página web con estilo importado .

Figura 41 - Página web con estilo importado

Con el botón de alto contraste se tuvo también el problema al inicio de que cuando se cambiaba de página, aunque hubieras cambiado la página anterior a alto contraste, la nueva se cargaba en modo normal. Esto hizo que nos viéramos obligados a utilizar *cookies* que cargaran la página en función de si el usuario la quería en alto contraste o modo normal. Esto se explicará con más detalle más adelante en el apartado Cookies.

Junto con las *cookies* se pensó en la posibilidad de añadir un cuadro de dialogo que indicara que el sitio web las usa. Se investigó y se encontró un ejemplo práctico (WAI-ARIA, 2018), además de la sección sobre cuadros de diálogo en las prácticas recomendadas por la WAI-ARIA. En ellas, se destaca la importancia de usar los atributos '*role* = *dialog*' y 'aria-modal'. Este último oculta al lector de pantalla lo que hay detrás del dialogo, haciendo así que el lector se centre únicamente en el cuadro de dialogo. Otro aspecto importante es que se debía devolver el foco al lugar donde estaba antes de aparecer el cuadro modal. En este caso, dado que el cuadro aparece al entrar en la página se debía retornar el foco al inicio de la web. Para el desarrollo se utilizó la clase que proporciona Bootstrap sobre diálogos y se añadieron las modificaciones mencionadas, entre otras de menos importancia.

Se pensó en añadir algún elemento artístico que acompañara el apartado de los minijuegos con el fin de dar un aspecto más atractivo a la web. Siguiendo con el tema y logo del videojuego se procedió a añadir un ojo que sigue al ratón por la pantalla y se cierra si se pasa el ratón por encima. Este ojo fue elaborado en CSS y JS. Dado que este ojo era puramente estético y no tenía texto se añadió el atributo '*role* = *presentation*' para indicar al lector de pantalla que se trataba de un elemento decorativo.

Por último, se debe comentar que, en una de las primeras versiones de la web, se añadieron botones que aumentaban y disminuían los tamaños de las fuentes del texto. Estos elementos fueron más tarde desechados al investigar y encontrar una opción similar en los ajustes de Android. Además, este aumento de texto se puede realizar también en cualquier navegador del ordenador, por lo que los botones propios no resultarían útiles.

Navegación accesible

La forma en la que las personas que utilizan los lectores de pantalla navegan con la página es mediante el uso de la tecla *tab*. Con esta tecla se va pasando el foco entre los elementos de la página, como botones o links. Para personas que no tienen una discapacidad visual completa, WAI indica que se debe señalar visualmente dónde está el foco, como se indica en la sección 2.4.7. Además, se debe dejar la funcionalidad, es decir, que cuando se pulse la tecla *intro* se debe realizar la misma operación que cuando se hace *click* sobre el elemento.

Teniendo esto de referencia se añadieron subrayados a todos los enlaces cuando se hacía *focus* sobre ellos y a los botones se les añadieron distintos diseños que indicaban que el objeto estaba en el foco. Al botón de cambio de contraste, por ejemplo, se le agrandó el circulo interno.

Además de esto hay una opción recomendada que ayuda a comprender la estructura de la web y a identificar la página. Se trata de "las migas de pan" y son muy comunes ahora mismo en las páginas web. Son una sección de la página que se utiliza para navegar y te indica como están encadenadas las páginas, como se puede apreciar en la Figura 42 - Migas de pan en la página web. Es muy útil para no perder rastro de cómo hemos llegado a la página actual. Además, forma parte del nivel AAA del criterio 2.4.8 de las **WCAG** sobre localización. Para la implementación de las migas de pan se ha utilizado la clase proporcionada por **Bootstrap** y se ha modificado para cumplir con las exigencias de accesibilidad, ayudándose de un ejemplo de práctica proporcionado por la **WAI**. (WAI , 2018). Se añadieron los atributos *aria-label* y *aria-current=" page"*. Este último, para indicar la página web actual. Además, dado que la clase de **Bootstrap** era de un elemento de lista ordenada y no de un elemento <*nav*> se añadió el rol de navegación, como especifican las **WCAG**.

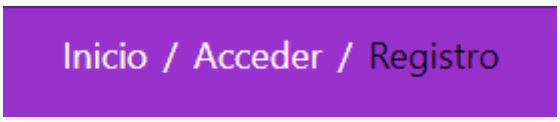


Figura 42 - Migas de pan en la página web

Interacción accesible

Los principales elementos con los que se interactúa en la página web son los formularios. Un formulario web es una sección de la página que permite al usuario introducir datos, como casillas de selección o un campo de texto. Para su desarrollo, se han seguido las prácticas recomendadas por la **WAI**. Por ejemplo, se debe destacar el atributo '*for*' del elemento *label* que debe ser igual al id del elemento *input* al que acompaña. Esto hace que el lector de pantalla entienda qué etiqueta corresponde a cada *input*. En algunos casos se han añadido además el atributo *aria-label*, para dar una descripción más amplia del *input*.

Además de los formularios, los otros elementos con los que se puede interactuar en la web son los botones. Los que más problemas dieron a la hora de ser accesibles fueron los botones de alto contraste y el sistema de puntuación de estrellas.

El botón de contraste es de tipo *checkbox*, estos elementos se activan mediante el botón de espacio. Se revisaron las prácticas de la **WAI-ARIA** en las que se indicaba que así debe ser.

No obstante, se pensó en la posibilidad de permitir activar estos botones con el *enter* también, ya que este es el botón más utilizado para *clickar* links o botones y así se hizo.
En cuanto a las estrellas de puntuación, se buscó la mejor manera de representarlas para que fueran accesibles y se encontró este artículo de **CSS-Tricks** (CSS-Tricks, 2019) en el cual hablan de varios métodos. Se tomó entonces la decisión de representarlas mediante caracteres **Unicode**, un sistema de codificación de caracteres. Y de representar los botones de votación como estrellas generadas mediante **SVG**.

Además, la **WAI** proporcionaba un ejemplo en **SVG** que se utilizó como referencia (WAI, 2019). Para que el lector de pantalla pudiera leer la puntuación se añadió el atributo *aria-label* junto con la variable numérica de la puntuación. Esto hacía que el lector leyera: "Puntuación 2 de 5 estrellas".

6.16.4. Web para móvil y tablet

Algo que se tuvo claro desde un primer momento es que la página iba a ser visitada generalmente en el mismo entorno que el videojuego, es decir, en dispositivos móviles. Este hecho motivó la elección de usar **Bootstrap**, ya que tiene plantillas de diseño que son muy *responsive*. El término *responsive* hace referencia a lo bien que reacciona una página web a la reducción de tamaño de la pantalla. Durante el desarrollo de la web se tuvo en consideración esto durante todo el momento, haciendo que se tuviera que realizar un diseño para pantallas grandes de ordenador y otro para pantallas pequeñas. El menú de navegación diseñado para ordenador era notablemente pequeño en móvil, por lo que se procedió a realizar algunos cambios. Se ensanchó la barra horizontal y se escondieron los botones de cambio de contraste, sustituyéndolos por un simple botón con la misma función. Además, se metieron los enlaces de navegación en un menú 'acordeón', que se puede ver en la Figura 43 - Vista del menú 'acordeón' en móvil. Este menú despliega los enlaces de navegación al pulsar un botón.

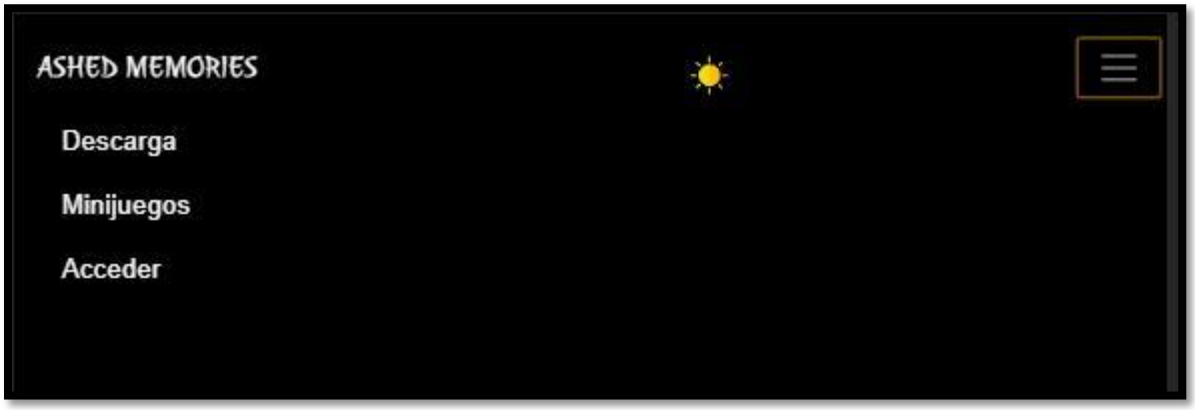


Figura 43 - Vista del menú 'acordeón' en móvil.

El encabezado con el logo y el título del videojuego cambió también. En vez de representarse con un orden horizontal, en móvil y *tablet* el orden es vertical. En un principio las tarjetas de los minijuegos se representaban de la misma manera en móvil y en ordenador. Lo que ocurría era que había que ampliar la pantalla para poder leer mejor. Se decidió entonces tener un diseño que evitara tener que ampliar. Este diseño fue el de mostrar cada tarjeta en orden vertical.

6.16.5. Lector de pantalla (NVDA) y TalkBack

Una vez terminado todo el diseño de la página web siguiendo las normas de accesibilidad se decidió que lo más correcto sería hacer pruebas con un lector de pantalla para ver si todo se había hecho de manera correcta. Se comenzó por las pruebas en ordenador y después se pasó a móvil.

Se debe mencionar que no todos los lectores de pantalla funcionan de la misma manera con todos los navegadores, ya que los diferentes navegadores utilizan distintas *APIs* de accesibilidad para leer el contenido web, como explica este artículo (Garaventa, 2013). Lo que recomienda este artículo es que utilizar las combinaciones de lectores de pantalla y navegadores más comunes. Se realizó una investigación y se dio con otro artículo que proporciona el porcentaje de usuarios que utiliza cada combinación (A11Y, 2018). En este se puede ver que la combinación más usada es **Internet Explorer** (Microsoft, 1995) con **JAWS**, pero al ser este de pago, se eligió utilizar la segunda opción: **NVDA** (NVDA, 2020), con **Mozilla Firefox**.

Una vez descargado **NVDA**, comenzaron las pruebas. Al principio resultaba extraño la utilización del lector ya que era la primera vez que se utilizaba tecnología de este tipo. Hubo un tiempo de familiarización con este software para aprender lo básico y una vez se tuvo la confianza suficiente se procedió a navegar por la web. En términos generales el resultado fue bueno, aunque hubo algunos errores que se tuvieron que corregir. Por ejemplo, la representación de la puntuación mediante estrellas no era dictada por el lector, aunque tenía la etiqueta '*aria-label*'. Esto se solucionó añadiendo '*role=img*' que se utiliza para indicar que se trata de un gráfico, un conjunto de imágenes, o caracteres especiales.

Para móvil se utilizó **Google TalkBack** y al igual que con el ordenador se necesitó un tiempo para acostumbrarse a utilizarlo. Se navegó a fondo por la web y no se encontró ningún fallo destacable.

6.16.6. Cookies

En la parte de accesibilidad un elemento que tuvo una gran importancia fueron las *cookies*. Su necesidad surgió con la funcionalidad de alto contraste. Al principio cuando se pulsaba el botón de "alto contraste" el modo cambiaba, pero en cambio cuando se navegaba a otra página distinta de la actual automáticamente volvía el modo normal. Esto es debido a que los cambios que se producen son en local y cuando navegas a otro sitio lo que hace es solicitar al servidor la página que por defecto esta con los otros colores. Para arreglar dicho problema al principio se pensó añadiéndole una variable *$_SESSION* al usuario. Así cuando se pulsase el botón para cambiar de un modo a otro, se cambiaría el valor de la variable y dependiendo del valor de esa variable la página se mostraría un modo u otro. Su implementación fue sencilla ya que solamente hubo que añadir una variable más cuando un usuario hacía *login*. Una vez implementada esta opción se observó que tenía un gran fallo. Dicho fallo era que esta solución solo funcionaba en el caso de que el usuario hubiese hecho *login* ya que era cuando se creaba una nueva sesión y con ello se podían crear las variables *$_SESSION*. Llegado a este punto se tuvo que pensar una solución para este problema.

Esta fue la creación de *cookies,* las cuales son variables globales las cuales almacenan información relativa al usuario. El fundamento de todo esto era crear una *cookie* cuando el usuario, registrado o no, pulsase el botón de cambiar de modo. Además, cada vez que navegase a otro sitio dentro de la página web se comprobaría el valor de la *cookie* para determinar si la página se carga con el modo normal o con el modo alto contraste.

Una vez pensada la solución se pasó a la ejecución. Primero se buscó por internet la manera de manejar las *cookies* ya que los conocimientos que tenía el grupo sobre este tema eran bastante reducidos. Las *cookies* son variables muy parecidas a las variables *$_SESSION* pero con la diferencia que las segundas van ligadas a una sesión y las primeras son más globales. De esta manera cuando el usuario, registrado o no, pulsaba por primera vez el botón de cambiar de modo, se creaba una *cookie* llamada "*accesibility*" la cual adquiría el valor de "*true*". Esto significaba que quería el modo especifico de accesibilidad que era el de alto contraste. De esta manera si el usuario navegaba a cualquier otra página dentro de la web, antes de cargarla se miraría el valor de dicha *cookie* para determinar con cuál de los dos modos se cargaba. De igual manera, si el usuario volvía a pulsar el botón de cambiar de modo, la *cookie* pasaría a tomar el valor "*false*".

Después de haber terminado esto, el grupo se percató de un aspecto importante de las *cookies*, su política. Para ello se hizo un *pop-up* que apareciera al entrar por primera vez en la página web, dando la opción de leer más sobre la política de cookies y la de aceptarla.

Si se pulsaba "política de *cookies*" se redireccionaba a una página que contenía información sobre cómo son utilizadas las *cookies* en el sitio web. En cambio, si se pulsaba en el botón de aceptar, simplemente desaparecía el *pop-up*. Con este *pop-up* también surgió un problema el cual fue que cada vez que se cargaba la página de inicio volvía a salir, aunque anteriormente se hubiera pulsado el botón de "Aceptar". De nuevo la solución fue la utilización de otra *cookie*. De esta manera cuando un usuario entrase en la página web se comprobaba la existencia de la *cookie* llamada "*cookie*". Si no existía era que aún no se habían aceptado la política de privacidad. En cuanto el usuario pulsaba el botón de "Aceptar" se creaba dicha *cookie* automáticamente y así ya no volvía a aparecer el *pop-up* a no ser que le propio usuario borrase las *cookies* de su navegador.

6.16.7. Conexión web – videojuego

Al principio se pensó que la web iba a tener solo finalidad informativa, es decir para ver información acerca del videojuego como las principales características de este o sobre los minijuegos que hay dentro de él. Pero según se iba desarrollando la página web se propuso la idea de que los usuarios del videojuego pudiesen acceder a la página web con su usuario y contraseña y así poder ver algo de información como sus estadísticas. De esta manera se incluyó la conexión web – videojuego.

Por eso mismo, una de las características más importantes de la página web es que está conectada directamente con la misma base de datos a la que está conectado el videojuego. Una de las principales ventajas de esto es que, una vez creada la cuenta desde el videojuego o la web, no hace falta que lo vuelvas a hacer en el otro sitio. Gracias a esto un usuario puede consultar sus estadísticas del videojuego entrando en la página web con el mismo usuario y contraseña que tiene para el videojuego y viceversa. Cuando un usuario inicia una nueva partida automáticamente se añade a la tabla de partidas. Además, según va progresando en la partida y va encontrando nuevos objetos, se le van añadiendo a la tabla de objetos de la partida actual.

Los datos relacionados acerca del videojuego de la base de datos se pueden modificar tanto desde el videojuego como desde la página web. La página web te da la opción de borrar los datos de tus partidas que no quieras guardar. Además, desde la página web tienes la opción de descargarte el juego en tu dispositivo directamente.

En el *login* y en el registro, tanto el videojuego como la página web guardan la contraseña de los usuarios tras aplicarle una función hash, de esta forma se evita guardarla en claro, mejorando así la seguridad del usuario.

7. Pruebas

La fase de pruebas de una aplicación o juego puede ser tan importante como el propio desarrollo, ya que en esta fase pueden encontrarse errores o mejoras que antes no se habían contemplado. Además, es importante que las pruebas no las hagan las mismas personas que han desarrollado el producto, ya que están "contaminados" por conocerlo en profundidad. Es necesario que lo prueben personas que no hayan tenido ningún contacto con la aplicación.

Por otro lado, al tratarse de una aplicación para personas ciegas o con problemas de visión, se tenía previsto un período de pruebas con personal de la ONCE, quienes se ofrecieron a hacer este favor al equipo, que sería de gran ayuda, al darnos *feedback* directamente de este tipo de personas. Sin embargo, esto no pudo realizarse plenamente debido a la situación derivada de la pandemia producida por el COVID-19 y el período de cuarentena.

Se pueden dividir las pruebas en los dos grandes módulos que tiene este trabajo:

7.1. Videojuego

El videojuego fue lo primero de lo que se consiguió tener una unidad mínima de experiencia. En esta primera aplicación se podía iniciar, elegir el modo de juego (con conexión o sin conexión) y se podía completar la primera sala. El juego fue proporcionado a personas cercanas (familiares y amigos, sin discapacidades visuales) para que lo probasen y las conclusiones que se sacaron de su *feedback* fueron:

- Los movimientos accesibles nativos utilizados en el menú y la primera sala, pese a que fueran intuitivos para personas ciegas, ya que están acostumbrados a ellos, para personas videntes (a las cuales también está dirigido el juego) no era tan intuitivo. Por lo tanto, había que añadir una explicación extra al inicio del juego para que quedase bien claro cuáles eran los controles.
  Tras añadir dicha explicación a la primera sala y volver a hacer pruebas, el movimiento por los objetos fue bastante bien acogido. A la inmensa mayoría de las personas no les costó acostumbrarse previamente conocidos los controles. Claro que, estando la explicación de los controles en la primera sala, el menú en su primera fase seguía sin ser intuitivo. Esto se arregló bastante más tarde con los menús deslizantes.

- El minijuego de la ganzúa fue bien valorado en general. Se recibieron buenas críticas en cuanto a la idea del minijuego y la dificultad. Aun así, también se recibieron ciertos comentarios sobre la frase explicativa, que fue ligeramente cambiada.

Una vez apuntadas estas críticas y mayormente solucionadas, se continuó con el desarrollo.

La siguiente versión que se dio a probar de la aplicación añadía la escena introductoria y el minijuego de transición Carrera. El objetivo principal de estas pruebas era comprobar si la introducción de parte de la historia era del agrado del jugador, además de probar las mecánicas añadidas y los textos explicativos nuevos. Así, se recibió el siguiente *feedback*:

- La parte introductoria tenía algunos sonidos demasiado estridentes, molestando al jugador. Arreglado ese problema, esta escena fue bien acogida, siendo sencilla e intuitiva. La historia se dejaba entrever, y se recogieron comentarios sobre posibles continuaciones que fueron útiles a la hora del desenlace aún por desarrollar.

- La escena Carrera tuvo ciertos comentarios útiles, siendo uno de ellos un punto de reflexión interesante. Este comentario sugería una pequeña ambigüedad en el uso de los deslizamientos laterales para girar; es decir, la persona en cuestión vio raro que el girar hacia la derecha se realizara por un deslizamiento en esa dirección, puesto que en ciertos juegos y aplicaciones ese movimiento propicia una rotación o fuerza inversa al deslizamiento, tal como se muestra en la Figura 44 - Deslizamiento inverso. Este comentario acabó por obviarse, ya que no recibimos la misma crítica del resto de probadores, aunque se anotó.

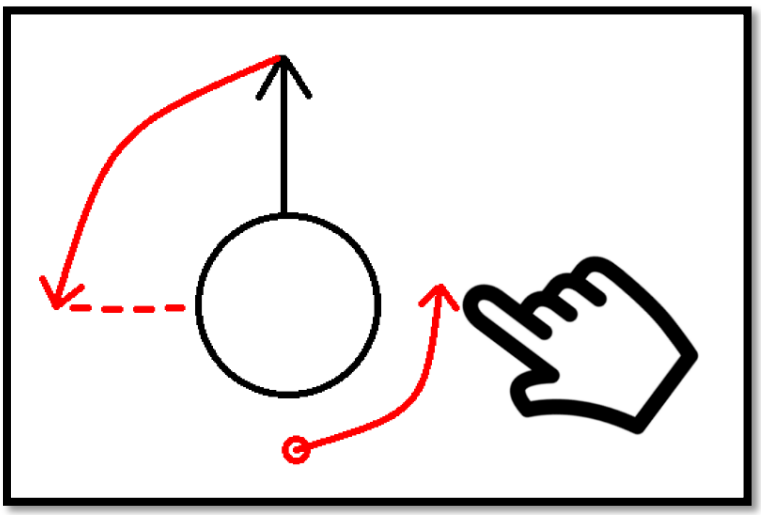


Figura 44 - Deslizamiento inverso

La próxima versión que se dio a probar fue una vez terminada la aplicación. Se había añadido la Habitación 2 y 3, el resto de los minijuegos y el final. El resto de las escenas se habían modificado y se habían añadido funcionalidades. Aquí recibimos más *feedback*:

- Los comentarios más habituales que se recibieron fueron sobre el minijuego de Formas. Este no daba información suficiente, era muy complicado y se bloqueaban constantemente en el juego. El minijuego sufrió varios cambios, entre los que se encuentran reorganización del flujo del juego, añadido de nuevas explicaciones y reducción en la dificultad de las formas. Este proceso de cambio fue tedioso, ya que constantemente se recibían nuevas dificultades por parte de los usuarios. Finalmente, el minijuego, pese a ser el más complicado de la aplicación aún, se encuentra en un estado válido, ya que es bastante más sencillo e intuitivo, sobre todo teniendo en cuenta que a nivel de diseño se planteó como el más difícil del juego.

- También se recibió *feedback* del minijuego Búsqueda, según el cual a veces se perdían ligeramente, por lo que se añadió algún sonido extra que facilitara su comprensión.

- Sobre la tercera y última habitación, se recibieron comentarios sobre el uso del *input* para abrir el inventario. Realmente aún no se explicaba al jugador, un error grave, ya que es una mecánica necesaria para completar el juego. Así, se añadieron mensajes explicativos en la primera sala que ayudaron enormemente.

- Por último, se recibieron ciertos comentarios sobre el fin del juego. Según estos comentarios, el juego daba sensación de acabar de una forma bastante abrupta. Esto se solucionó añadiendo tiempo a la escena, dando más distancia entre los elementos auditivos.

Se han obviado en este apartado los cambios realizados al juego que, o bien no se han producido gracias al *feedback* externo, o bien carecen de la dimensión necesaria para siquiera mencionarlos aquí.
Todos estas pruebas y cambios sobre el videojuego se han ido realizando de manera continua, siempre intentando mejorar la aplicación respecto al *feedback* recibido para hacerla lo más accesible posible y, en definitiva, resulte entretenida y agradable.

7.1.1. Formularios videojuego

Una vez obtenida la versión final de la aplicación, y tras distribuirla a los probadores, se creó un formulario Google para que, además de obtener el *feedback* de forma verbal como íbamos haciendo, se pudiera recoger de forma más organizada y sacar conclusiones de este.
El formulario, tomando tres preguntas base sobre la demografía del jugador, trata de obtener información sobre dificultades encontradas, tanto generales a nivel de controles como específicas a nivel de juego e historia.

Los datos obtenidos siguen la norma de lo que se pensaba por parte de los desarrolladores, ya que las mayores dificultades se encontraron sobre todo en el minijuego Formas. Obviando esta parte, la tónica general de la encuesta es positiva, ya que sólo un pequeño porcentaje de los probadores encontró problemas con los controles del juego, además de entender la historia y completarlo en su totalidad. Aun así, teniendo en cuenta que el juego también se dio a probar a público no objetivo (personas poco habituadas a los videojuegos), obtuvimos ciertos comentarios sobre dificultades, tanto a nivel de controles, como de historia y de minijuegos.

Cabe destacar también que no se encontraron fallos de funcionamiento del juego o comportamientos inesperados, y se obtuvieron algunos comentarios adicionales positivos.

7.1.2. Feedback ONCE

Debido a la problemática del COVID-19, las pruebas que en un principio se establecían con la fundación ONCE sufrió cambios considerables. El más importante fue la carencia de retroalimentación durante el desarrollo del proyecto, lo que llevó al equipo a diseñar el apartado accesible de la aplicación sin el factor más importante, el *feedback* de usuarios objetivo. Aun así, con las dificultades que probablemente estuviera atravesando la fundación, sacaron tiempo para ofrecernos su ayuda y probar nuestra aplicación una vez ya completado el desarrollo.

Este *testing* sucedió en unas fechas bastante cercanas a la entrega del proyecto y la memoria (comienzos de junio), por lo que, pese a que el *feedback* recibido es de gran valor e importancia, no ha sido posible la realización de cambios en base a la información recibida. Aun así, estos comentarios han sido analizados detenidamente por los desarrolladores, y se agradecen enormemente.

A continuación, se comentará el *feedback* recibido y lo que este implica en nuestra aplicación:

El primer comentario es un problema que sorprendió a los desarrolladores, ya que fue la primera vez que se recibió algo similar. Este menciona la necesidad de utilizar dos dedos en lugar de uno para activar cualquiera de los gestos del juego. Al parecer esto está relacionado específicamente con la versión de **Android** del dispositivo, provocando que únicamente suceda en algunas. Por parte de los desarrolladores, este error inesperado no tenía forma realista de ser previsto, ya que la base del *input* utilizado procede de la propia gestión de **Unity** y no surgió anteriormente una situación similar. Los probadores de la ONCE mencionan que lo han probado satisfactoriamente con otros dispositivos en otras versiones de **Android**.

El siguiente comentario es, probablemente, el que más afecta al apartado accesible de los recibidos. Se comenta que, debido a la necesidad de apagar el **TalkBack** para poder utilizar correctamente la aplicación, no es posible (por lo menos no sencillo) la creación de una cuenta. Esto sucede debido a la incapacidad de leer el texto de los campos de usuario y contraseña una vez escritos mediante la síntesis de voz. Esto se podría solucionar añadiendo lectura por parte del sistema de TTS propio después de la escritura del usuario, aunque este enfoque también habría que someterlo a pruebas para determinar su efectividad.

Se comenta también que, a nivel general, las instrucciones que se le ofrecen al jugador son correctas, pero se presentan dificultades en las partes más problemáticas, en específico el minijuego de reconocimiento de formas por vibración. Esto no es nada nuevo, ya que el *feedback* general obtenido anteriormente señalaba directamente a esta parte como la más complicada.

También se recibieron otros comentarios, como la imposibilidad de salir de la aplicación con controles específicos una vez dentro de una partida, aunque sí se puede desde el menú principal. Esa funcionalidad, que se pasó por alto en el desarrollo, se podría añadir en la forma de un elemento que aparezca siempre en el inventario y permita volver al menú principal.

Se menciona que el término utilizado por la ONCE para los gestos que implican deslizamientos es “flik”. Esto ya era conocido por los miembros del equipo, pero se decidió el uso del término “deslizamiento” en su lugar debido a que gran parte del público objetivo probablemente lo desconociera, ya que se utiliza normalmente en un ámbito accesible menos usual para el usuario medio.

Otro comentario importante también es el uso del término “Persona invidente”. Según el *feedback* recibido, es más adecuado utilizar “Persona ciega”; esto provocó el reemplazo de este término en la memoria del proyecto.

Por último, se menciona que los comentarios mandados son cuestiones de mejora, y que la idea del juego y la sensación general es muy buena, felicitando por el resultado obtenido.

Este apartado se quiere concluir agradeciendo enormemente la labor de la ONCE y la ayuda proporcionada, siendo esta extremadamente útil, no sólo para mejorar la aplicación de cara a una versión futura, sino también por el conocimiento sobre las discapacidades visuales que se ha proporcionado y que ha hecho mella en los desarrolladores.

7.2. Página web

La página web se fue terminando después del videojuego, pues este último era la prioridad. En términos de funcionalidad se hicieron pruebas realizando todas las actividades que la página permite. Por ejemplo, se creó una cuenta y se probó el videojuego con esta, para comprobar que aparecían las estadísticas y objetos en el perfil de la web.

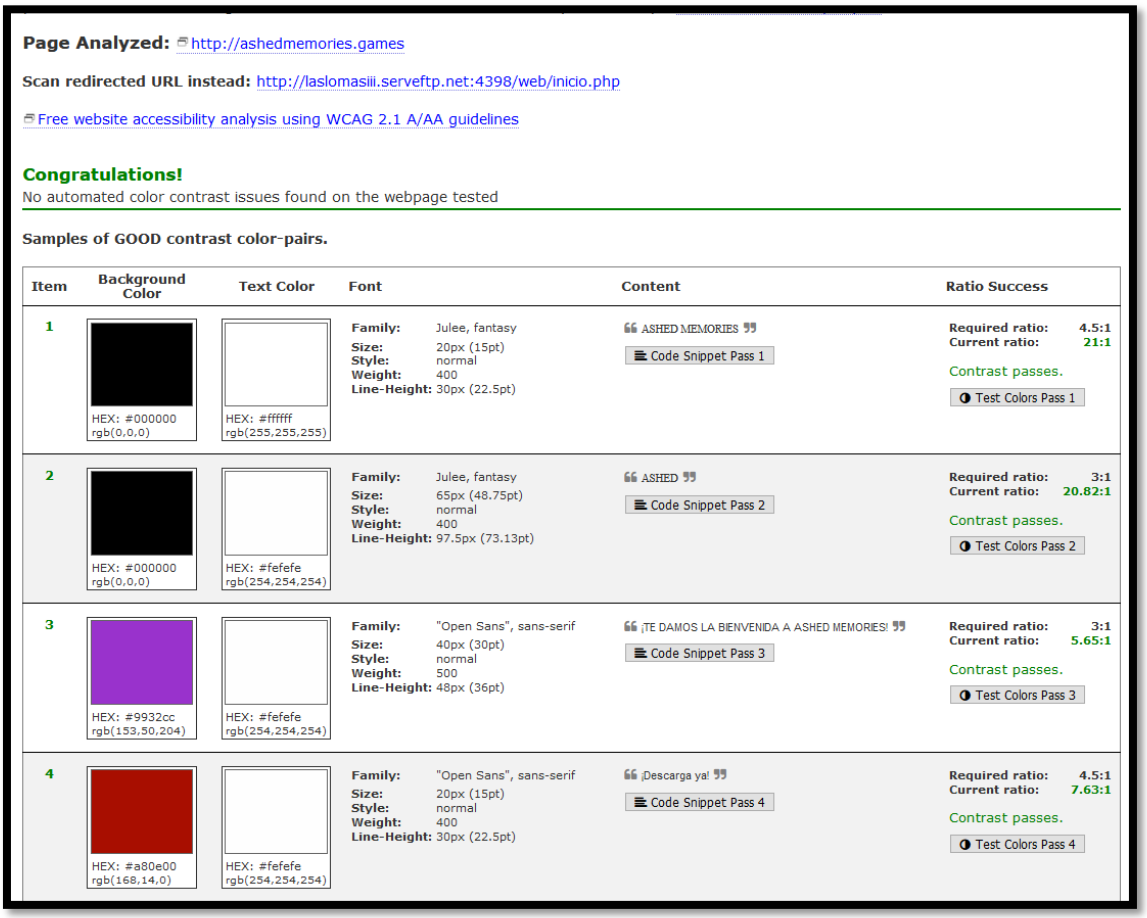
En cuanto a aspectos visuales se pasó el enlace a distintas personas del entorno que aún no hubieran visto nada de ella y nos dieron sus primeras opiniones sobre ella:

- La estética de la página fue bien recibida en general.
- El botón de cambiar a alto contraste les pareció buena idea y que funcionaba muy bien.
- Algunos textos les parecían demasiado pequeños como para que fuese cómodo leerlos.
- Algunos enlaces estaban camuflados y costaba encontrarlos.

Para solucionar los problemas que se indicaron, se procedió a cambiar el CSS de la página web para que los enlaces fueran fácilmente visible e identificables como tales. También se aumentaron los textos de algunos lugares para mejorar su legibilidad.

Además, se tenía que comprobar la accesibilidad de la web. Como se ha visto en apartados anteriores, se testeó la web de maneral manual utilizando TalkBack, pero al no ser usuarios frecuentes de este software y con el fin de tener una mayor seguridad sobre qué tan accesible era la página web, se investigaron otras maneras de comprobarlo. La W3C proporciona una lista de herramientas que validan el código HTML y CSS en términos de accesibilidad (W3C, 2016). Estas herramientas determinan si la web cumple con las guías de accesibilidad proporcionadas por la WCAG.

Una de estas herramientas es un analizador de contraste que verifica si la página cumple con los criterios de conformidad llamado color.a11y (A11Y, 2020). Se utilizó este validador para corroborar que todos los colores elegidos cumplían con el criterio de conformidad AAA.Y, como se puede ver en la Figura 45 - Contraste analizado por color.a11y.com, no se encontró ningún error de contraste.



Page Analyzed: http://ashedmemories.games

Scan redirected URL instead: http://laslomasiii.serveftp.net:4398/web/inicio.php

Free website accessibility analysis using WCAG 2.1 A/AA guidelines

Congratulations!

No automated color contrast issues found on the webpage tested

Samples of GOOD contrast color-pairs.

Item	Background Color	Text Color	Font	Content	Ratio Success
1	HEX: #000000 rgb(0,0,0)	HEX: #ffffff rgb(255,255,255)	**Family:** Julee, fantasy **Size:** 20px (15pt) **Style:** normal **Weight:** 400 **Line-Height:** 30px (22.5pt)	ASHED MEMORIES Code Snippet Pass 1	**Required ratio:** 4.5:1 **Current ratio:** 21:1 Contrast passes. Test Colors Pass 1
2	HEX: #000000 rgb(0,0,0)	HEX: #fefefe rgb(254,254,254)	**Family:** Julee, fantasy **Size:** 65px (48.75pt) **Style:** normal **Weight:** 400 **Line-Height:** 97.5px (73.13pt)	ASHED Code Snippet Pass 2	**Required ratio:** 3:1 **Current ratio:** 20.82:1 Contrast passes. Test Colors Pass 2
3	HEX: #9932cc rgb(153,50,204)	HEX: #fefefe rgb(254,254,254)	**Family:** "Open Sans", sans-serif **Size:** 40px (30pt) **Style:** normal **Weight:** 500 **Line-Height:** 48px (36pt)	¡TE DAMOS LA BIENVENIDA A ASHED MEMORIES! Code Snippet Pass 3	**Required ratio:** 3:1 **Current ratio:** 5.65:1 Contrast passes. Test Colors Pass 3
4	HEX: #a80e00 rgb(168,14,0)	HEX: #fefefe rgb(254,254,254)	**Family:** "Open Sans", sans-serif **Size:** 20px (15pt) **Style:** normal **Weight:** 400 **Line-Height:** 30px (22.5pt)	¡Descarga ya! Code Snippet Pass 4	**Required ratio:** 4.5:1 **Current ratio:** 7.63:1 Contrast passes. Test Colors Pass 4

Figura 45 - Contraste analizado por color.a11y.com

Estos validadores ayudaron a mejorar el código con respecto a la accesibilidad. El inconveniente fue su lentitud, ya que cualquier cambio que se hacía en la web, cuando estaba finalizada, suponía subirlo a **Github**, sustituir el código en el servidor y volver a utilizar las herramientas de validación.

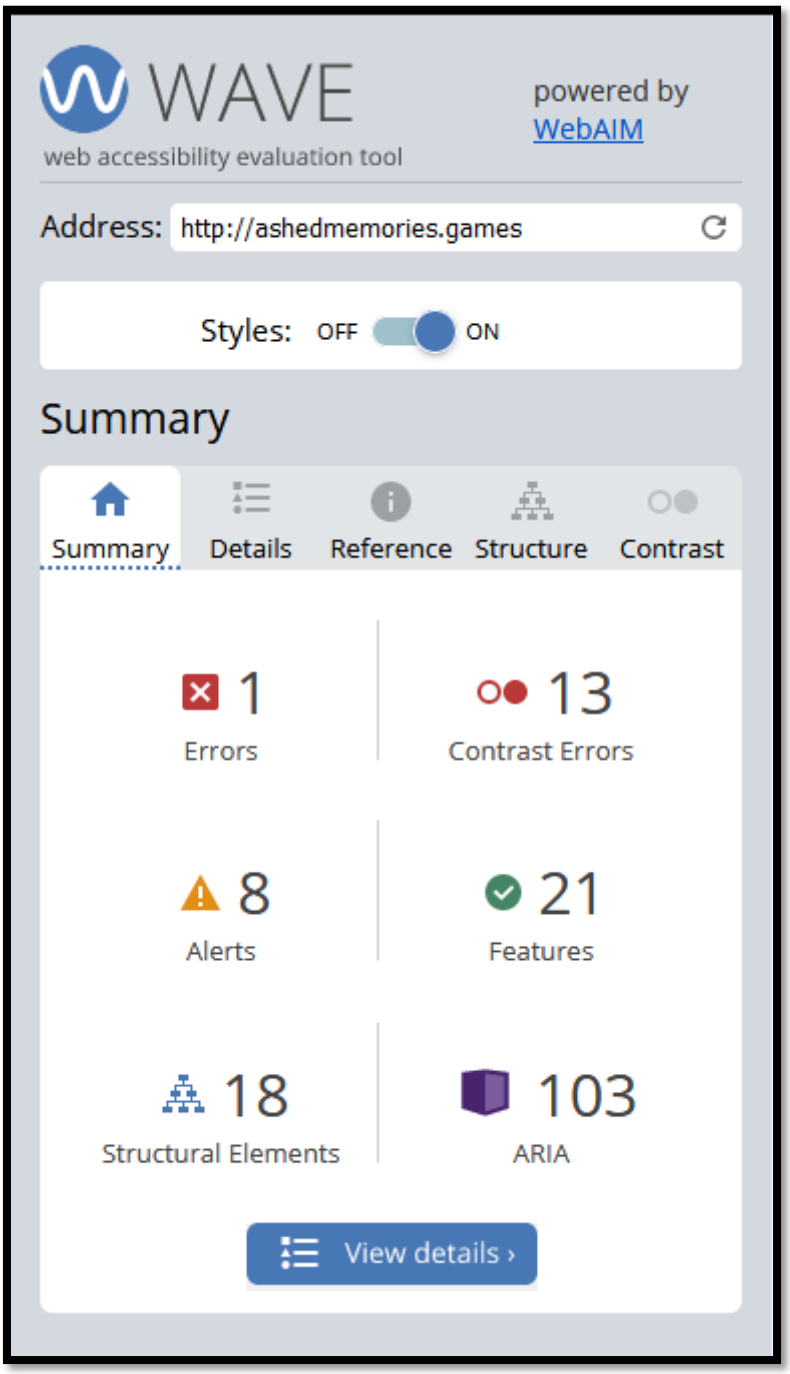


Figura 46 - Resultado de evaluación de accesibilidad web

Los errores se fueron solucionando o se fueron sustituyendo por las prácticas recomendadas. Por ejemplo, el error en la Figura 46 - Resultado de evaluación de accesibilidad web se debe al uso de una etiqueta vacía, sin embargo, la propia herramienta indica que si una etiqueta debe ser vacía, se puede utilizar el atributo '*title*' describiendo su función. Dado que esta etiqueta, por motivos de diseño, se quería vacía, se implementó esta recomendación.

También se encontró este artículo (HTML.COM, 2019) que citaba 23 herramientas gratuitas de este tipo, de las cuales se han usado la mayoría.
Unos de estas herramientas (I AM RESPONSIVE, 2013), además de validar el código, prueba la capacidad de respuesta de nuestra web a los diferentes formatos de pantalla y muestra cómo se vería en cada uno de ellos, tal y como se ve en la Figura 47 - Representación de la web en diferentes pantallas. Esto era una cosa importante, ya que se pretendía que la web fuese bonita y estética en la misma plataforma en que se juega el videojuego, **Android**.

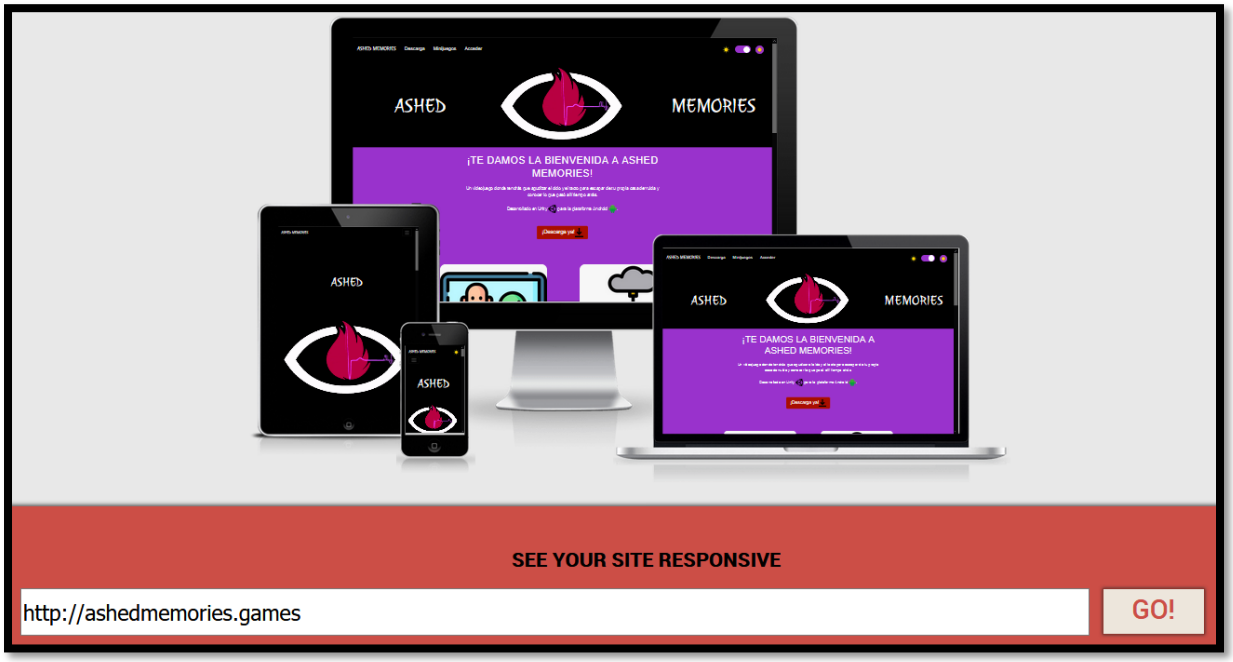


Figura 47 - Representación de la web en diferentes pantallas

7.2.1. Formularios web

Al igual que con el videojuego, una vez terminada la web, se creó un formulario Google para distribuirlo entre los probadores con el objetivo de obtener *feedback* y poder sacar conclusiones de los resultados obtenidos.
El formulario empieza con tres preguntas sobre la demografía del jugador y continúa con preguntas orientadas a obtener información sobre accesibilidad, la usabilidad y la utilidad de la página web.

Por lo general los resultados del formulario han sido positivos. El modo de alto contraste ha sido valorado muy positivamente. Por otra parte, cabe destacar que a la mayoría de las personas encuestadas la información proporcionada sobre los minijuegos le ha ayudado a entender mejor cómo se juega. Por último, la parte de las estadísticas, al ser un juego muy corto algunas de las personas encuestadas lo ven como algo prescindible.

8. Conclusiones y trabajo futuro

En la realización de este trabajo el objetivo principal siempre fue la accesibilidad para personas ciegas, tanto en el desarrollo del videojuego como en el de la página. Para esto fue necesario adquirir conocimientos en esta área, tanto de las prácticas habituales que se siguen para conseguirla como del modo en que estas personas interactúan con la tecnología, especialmente con sus teléfonos móviles. Estos conocimientos se pudieron obtener gracias a la guía de los profesores directores del trabajo, la colaboración de la ONCE y la labor de investigación realizada por los estudiantes.

Mediante esta información se llegó a la conclusión de que el elemento central en la accesibilidad para personas ciegas, especialmente en teléfonos inteligentes, son los lectores de pantalla. Estos sistemas, a los que las grandes empresas del sector cada vez dedican más recursos, son la norma en lo que a este tipo de accesibilidad se refiere. Es por ello por lo que se tomaron estos sistemas como referencias y como elemento base sobre el que se construiría lo demás.

En cuanto al videojuego, se tomó la decisión de imitar estos lectores de pantalla con la implementación de un sistema propio, que ofrecería una forma de interacción similar a la de estos, facilitando su uso por personas ya habituadas a estas tecnologías. La implementación propia fue necesaria a causa de otra de las conclusiones alcanzadas durante este trabajo, y es que el sector de la accesibilidad para personas ciegas aún no es habitual, especialmente en el sector de los videojuegos. Esto se pudo observar, por ejemplo, en que el motor de videojuegos utilizado en el desarrollo (Unity) a pesar de ser uno de los más conocidos y usados, y siendo habitual tanto en el campo académico como el profesional, no ofrecía soporte a la mayoría de las tecnologías utilizadas para conseguir esta accesibilidad.

Respecto a la web, el enfoque fue distinto. Se basó en hacer que la web fuera compatible con los lectores de pantalla, diseñándola de forma que su modo de uso encajara con el habitual utilizando esta tecnología.

Otro factor importante es el hecho de que dentro de la accesibilidad para personas con discapacidades visuales también se incluyen las personas que conservan pequeños restos de visión. Estas personas, aunque no del mismo modo que las que tienen problemas de visión más leves, pueden seguir apoyándose en esos restos de visión. Por ello, también se tomaron medidas que les facilitaran este uso. En concreto, el uso de colores de alto contraste, fuentes de texto claro y tamaño grande, tanto en la web como en el videojuego.

Este trabajo fue acompañado de varios periodos de pruebas, situados a lo largo del desarrollo. Gracias a estas pruebas se obtuvo información de gran utilidad para realizar cambios durante su desarrollo, ayudando a detectar elementos confusos, no suficientemente explicados y comportamientos no deseados. También cabe mencionar que estas pruebas se vieron en parte truncadas por la epidemia del COVID-19 y el consiguiente período de cuarentena, ya que evitaron realizar un período de pruebas finales en colaboración con la ONCE de la forma prevista inicialmente.

Teniendo en cuenta los resultados obtenidos, se pueden considerar una serie de modificaciones y añadidos de cara al futuro, siendo los más relevantes los siguientes:

- **Cambios en base a *feedback*:** Basándose en la información recibida durante las pruebas, en particular en las encuestas realizadas al final del desarrollo, se podrían detectar puntos conflictivos y posibles mejoras, tanto en el videojuego como en la web. Serían prioritarios los cambios basados en el *feedback* dado por la ONCE (véase Feedback ONCE). En el juego principalmente tendrían lugar en los minijuegos, centrándose en su equilibrado y en la corrección de fallos que hubieran quedado sin detectar durante el desarrollo. El balanceo sería sencillo gracias a la posibilidad de modificar muchos de sus parámetros directamente desde el editor. En la web el objetivo sería similar, resolver fricciones en su uso y fallos.
- **Dar soporte a más idiomas:** Dado el tamaño reducido del público al que está dirigido el juego, sería interesante ampliarlo haciendo llegar el juego a más personas. El principal idioma objetivo sería el inglés, por su gran cantidad de hablantes y su base establecida de jugadores ciegos. Una traducción del videojuego supondría un esfuerzo principalmente a nivel de producción, ya que, aunque su inclusión en el proyecto no supondría grandes problemas, obtenerlo implicaría duplicar todo el proceso de grabación. Respecto a la web, requeriría una traducción total y el cambio automático al idioma del dispositivo del usuario. Ambas traducciones necesitarían además la traducción de los recursos gráficos que contengan texto. Cada idioma adicional, requeriría repetir todo este proceso.
- **Soporte de Google Play:** Actualmente, la conexión online del juego se produce mediante un servidor dedicado propio. Sería posible la inclusión de los servicios de Google Play al publicar el juego, reorganizando la aplicación para que funcionase mediante este. Esto permitiría la inclusión del sistema de logros propio de Google Play, además de eliminar la carga que supone el servidor que se posee. A nivel de funcionalidad, obviando el sistema de logros, el juego se mantendría similar, ya que con el servidor actual se proporcionan las características necesarias y suficientes para emular su uso.

- **Llevar el juego a la plataforma IOS:** Del mismo modo que con el idioma, llevar el juego a esta plataforma aumentaría considerablemente sus jugadores potenciales. Para esto, además de los conocimientos y medios para el desarrollo y realización de pruebas en esta plataforma, serían necesarios notables cambios en tecnologías usadas por el juego. Como actualmente se hace uso de tecnología específica de la plataforma Android (Vibración, TTS, STT), sería necesario hacer uso de sus equivalentes en la plataforma iOS, o en caso de no ser posible, modificar el juego para ofrecer alternativas a su uso.
- **Ampliación del juego:** El juego actual se comporta, a nivel práctico, como una demo. En el caso de querer ampliarlo, además de añadir contenido a la historia, se podrían añadir minijuegos que aporten mecánicas nuevas y accesibles, que ofrezcan nuevas opciones de juego a los usuarios. Un ejemplo de esto podría ser la inclusión de un minijuego con características online, que requiera la cooperación o competición de más de un solo jugador para su superación.

9. Conclusions and future work

The main objective of this project was always accessibility for blind players, both in the game development and in the webs. To this end, it was necessary to acquire knowledge in this area, of the common practices to achieve and of the way in which this people interact with technology, especially mobile devices. This knowledge could be obtained thanks to que guiding hand of the works directing teachers, the support given by ONCE and the students investigation labor.

With this information a conclusion was reached, that the main element of accessibility for blind people, especially in smartphones, are screen readers These systems, in which the main companies of the sector spend more and more resources, are the norm in this type of accessibility. For this reason, these systems were taken as reference and base element over which to build the rest.

Regarding the videogame, the decision was taken to mimic this screen readers with an implementation of our own, one which would offer similar ways of interaction, making its use easier for people already accustomed to these technologies. This implementation was necessary because of another of the conclusions reached through this work: that accessibility to blind people is not commonplace, particularly when talking about videogames. This can be observed, for example, in the engine used in the development (Unity). Even though it is one of the most known and used, both academically and professionally, it offers no support to most of the technologies used to achieve this kind of accessibility.

Regarding the web, the approach was different. It was based on making the web compatible with screen readers, designing it so that its way of use fitted with the ones that are common when using this technology.

Another important factor is the fact that accessibility to people with visual disabilities also includes people with small traces of vision left. This people, even though not in the same way as those with slighter vision problems, can still make use of this traces of vision. Because of this, measures were also taken to facilitate this way of use. Specifically, high contrast colors and clear text fonts with big sizes were used, both in the web and videogame.

This project was also accompanied by several testing periods, taking place along the development. Thanks to these tests, greatly useful feedback was acquired, which allowed to make changes during development by detecting confusing elements, those that required more explanations and undesired behaviors. It is also worth noting that these tests were affected by the COVID-19 epidemic and the quarantine period, since they had to be done telematically, and an initially planned final period of thorough testing in collaboration with ONCE was not possible.

Teniendo en cuenta los resultados obtenidos, se pueden considerar una serie de modificaciones y añadidos de cara al futuro, siendo los más relevantes los siguientes:

- **Changes based on feedback:** With the feedback obtained during testing as a base, particularly from tests after the development had ended, troublesome spots and possible improvements could be detected. Changes based on feedback from ONCE (seen in Feedback ONCE) would take priority. In the game most of the changes would be done to the minigames, centered on balancing them and fixing previously unnoticed errors. The balancing would be easier thanks to many of the minigames parameters being modifiable from the editor. In the web the goal would be similar, fixing frictions in its use and mistakes.
- **Supporting more languages:** Given the small size of the games target audience, it would be interesting to expand it by making it reach more people. The main target language would be English, because of its large number of speakers and its stablished base of blind players. Translating the game would require mostly production effort. Even though including the new audios on the game would not cause much trouble, obtaining those audios would require, besides translating, repeating all the recording process. In terms of the web it would also require a full translation, plus detecting the device set language. Both translations would require translating graphic elements that contain text. Each additional language would require repeating all this process.
- **Google Play support:** Currently, the game's online connectivity is done through a dedicated server of our own. It would be interesting to include Google Play services when publishing the game, making changes so that it works with these. This would allow using Google Play's achievement system and reduce the strain on the server. In terms of functionality, besides achievements, the game would stay similar since the server already provides features similar to those of these services.
-


- **Porting the game to iOS:** This would greatly increase the number of potential players. To do this, besides the knowledge and means to develop and test on this platform, notable changes on the technologies used in the game would be necessary. Since currently technologies specific to **Android** are used (Vibration, TTS, STT), it would be necessary to use their equivalents on **iOS**. If this was not possible, the game would have to be modified, offering alternatives to these technologies.
- **Expanding the game:** The current game acts as a demo. If it were to be expanded, besides adding content to its story, new minigames with accessible mechanics could be added, offering new ways of playing to the users. An example could be a minigame with online characteristics, that required cooperation or competition between multiple players.

10. Contribuciones

A continuación, cada miembro del equipo indica sus contribuciones principales al trabajo en su respectivo apartado.

Además de esas tareas, más específicas a los miembros del equipo, varias tareas se han realizado en común. Esto ha ocurrido especialmente en las primeras etapas del proyecto, gracias en gran parte al enfoque presencial que aún había. dichas tareas serían las siguientes:

- Investigación inicial sobre el modo de uso de los dispositivos móviles por parte de personas ciegas. Esto incluye los contextos en los que los utilizan, las formas de hacerlo que suelen ser preferentes y los medios que hacen ese uso posible. Fueron de especial interés los lectores de pantalla. Esta búsqueda de información tomo como base la proporcionada por la ONCE en la visita a sus instalaciones.
- Diseño del concepto del juego y sus mecánicas generales. Este pasó por varias iteraciones, que incluían la adaptación de un juego de cartas, recreación de aventuras gráficas clásicas y la imitación de un *escape room,* entre otros. Para esto se valoró cuanto se adaptaban estos conceptos al enfoque accesible, además de las mecánicas que permitirían introducir y problemas que conllevarían.
- Diseño de la interacción entre el videojuego y el servidor. En un principio se planteaban funcionalidades para varios jugadores en tiempo real que acabarían descartadas por los problemas que acarreaban a nivel de diseño, accesibilidad y desarrollo. La comunicación entre videojuego y servidor se rige por estándares, que fue necesario crear.
- Diseño de los menús del videojuego con el objetivo de hacerlos accesibles y que gracias a su componente visual actúen como forma de introducir a usuarios a los gestos de TalkBack.
- Búsqueda de usuarios para la realización de pruebas sobre videojuego y web. Realización de dichas pruebas, valoración de los resultados para la introducción de cambios en base a la información obtenida e implementación de estos cambios.
- Labores organizativas a través de los canales de comunicación usados, para ofrecer apoyo a los demás miembros del equipo y evitar cuellos de botella.

10.1. Arturo Aguirre Calvo

Investigación en accesibilidad en páginas web, así como el uso de los teléfonos móviles inteligentes por parte de las personas ciegas a través de los gestos explicados anteriormente. Haciendo hincapié en lectores de pantalla y programación para su correcto funcionamiento, contraste de colores, tamaño de letra y comportamiento, entre otros.

Diseño de la base de datos del servidor en la nube, inicialmente en diagrama Entidad-Relación, así como sus posteriores iteraciones según las necesidades del proyecto y su implementación en lenguaje SQL.

Configuración del servidor en la nube desde cero. Esto incluye la instalación del SO tras una breve investigación según las necesidades del proyecto y la instalación del servidor web HTTP, nuevamente tras investigar las soluciones que existen, centradas siempre en el software libre. Dotación de visibilidad del servidor en internet con dominio propio y NAT, y al servicio phpmyadmin para todo el equipo. Además, alojamiento de este en ordenador propio.

Implementación de la lógica del servidor en la nube referente al videojuego, así como la estructura de las clases en lenguaje PHP y conexión con la base de datos. En colaboración con Eduardo Andrés Morais.

Investigación en C# e implementación de la clase en cliente que reúne todas las llamadas al servidor y se comunica con este. Además, investigación y utilización del protocolo HTTP para peticiones *GET* y *POST* con las cuales se manda la información al servidor.

Aprendizaje básico del motor Unity utilizado en el desarrollo del videojuego e implementación de una primera pantalla de *login* y registro simple para pruebas de conectividad cliente-servidor. En colaboración con Alberto Casado Trapote.

Diseño e implementación de la parte de *front-end* de la página web aplicando los conocimientos sobre accesibilidad adquiridos durante el todo el proyecto.

Diseño e implementación de la parte de *back-end* de la página web utilizando PHP para la lógica y procesado de los datos, así como la implementación de las clases que realizan las consultas en SQL a la base de datos.

Investigación e implementación de *cookies* para el funcionamiento del método de alto contraste con colores accesibles de la página web y la política de cookies.

Realización del formulario de la página web para recoger las opiniones de los usuarios probadores de una forma más ordenada y automática, para su posterior análisis y conclusiones.

Todas las tareas relacionadas con la página web se han hecho en colaboración, en mayor o menor medida según la tarea concreta, con Fernando Cortés Sancho y Eduardo Andrés Morais.

Actividad formativa en tecnologías usadas por personas ciegas proporcionada por la Universidad Complutense de Madrid en la Semana de la Informática.

Realización de un curso en **Udemy** sobre accesibilidad en páginas web, ofrecido por profesores de la Universidad de Alicante.

10.2. Eduardo Andrés Morais

Investigación tanto en accesibilidad en videojuegos como en la manera que las personas ciegas hacen uso de los teléfonos móviles inteligentes utilizando los gestos que se han explicados en los apartados anteriores. En particular se hizo hincapié en los distintos niveles de pérdida de visión, así como la incidencia que tiene dicha pérdida en el uso de dispositivos electrónicos, en especial el de dispositivos móviles.

Diseño e implementación de la base de datos con el método de Entidad-Relación. Además de la realización de los cambios necesarios en esta debido a las necesidades del proyecto. La implementación de la base de datos es en lenguaje SQL.

Implementación y diseño de la lógica de las clases PHP referentes a la conexión con el videojuego. También realización de la conexión del videojuego con la base de datos. En colaboración con Arturo Aguirre Calvo.

Investigación en C# e implementación de la clase en cliente que reúne todas las llamadas al servidor y se comunica con este. Además, investigación de protocolo HTTP para peticiones GET y POST.

Todas las tareas relacionadas con la página web se han hecho en colaboración, en mayor o menor medida según la tarea concreta, con Fernando Cortés Sancho y Arturo Aguirre Calvo.

Investigación en accesibilidad web adquiriendo conocimientos sobre lectores de pantalla y la forma en las que se utilizan, distintos tamaños de letra, contraste de colores, entre otros. Además, investigación sobre las normas de accesibilidad para la correcta implementación de la página web con el objetivo de poder utilizar el lector de pantalla correctamente.

Diseño e implementación de la parte *back-end* de la página web utilizando PHP para la lógica y el procesado de los datos además de la implementación de las clases que realizan las consultas en SQL a la base de datos. También la realización de la implementación de algunos detalles de la parte de *front-end* de la página web.

Investigación exhaustiva sobre el funcionamiento y la utilidad de las *cookies,* así como de toda la legalidad que llevan detrás. Además, implementación de las *cookies* para el funcionamiento de la funcionalidad de “alto contraste” así como para la “política de *cookies*”.

Realización del formulario de la página web con el fin de recoger las opiniones de los usuarios probadores de una forma automática y más ordenada. Además, realización del análisis sobre los resultados obtenidos en el formulario.

Actividad formativa en tecnologías usadas por personas ciegas proporcionada por la Universidad Complutense de Madrid en la Semana de la Informática.

Realización de un curso online en **Udemy** sobre accesibilidad en páginas web, ofrecido por profesores de la Universidad de Alicante.

10.3. Alberto Casado Trapote

Investigación sobre accesibilidad para discapacidades visuales, haciendo hincapié en el uso de **Google TalkBack** (en colaboración con Diego Martínez Simarro). Realización de Entrevista a persona con problemas de visión e investigación sobre videojuegos accesibles, concretando con Caso de estudio: A Blind Legend.

Prototipado inicial en **Unity**. Realización de pruebas de interacción con elementos accesibles imitando a los lectores de pantalla y sistemas de guiado 3D por sonido.

Búsqueda de audio, generación y edición completa en el videojuego, con ayuda ocasional. Aquí se incluyen tanto la generación de efectos, como las voces TTS explicativas, como ambas ediciones y su posterior introducción en el juego. (No se incluye aquí lo relacionado con el sistema 3D de audio). La creación y edición de los audios se ha hecho en **Audacity**, y ocasionalmente en **Reaper**.

Composición de la banda sonora. Creación de dos temas musicales ajustados a la temática del juego de duración aproximada al minuto cada uno, generados en **Reaper** con *plugins* para desarrollo musical e instrumentos.

Arquitectura, flujo de juego y tareas de estructura, destacando la gestión de los estados de los elementos interactivos en coordinación con el lector de pantalla. Estas tareas se han realizado en colaboración con Diego Martínez Simarro.

Programación del sistema de gestos utilizado en el proyecto, imitando a los ofrecidos por el **TalkBack** y añadiendo además otros adaptados a las necesidades de los minijuegos, realizándose de forma genérica para su uso sencillo en la aplicación.

Diseño y desarrollo de la escena introductoria y escena final, con su consiguiente generación del apartado gráfico dentro del videojuego, sus animaciones y su unión sonora. El apartado gráfico se ha realizado teniendo en cuenta pautas de accesibilidad, uso de colores no intrusivos y formas diferenciables. Las animaciones se han generado con el motor de animación propio de **Unity**, dándole importancia a la coordinación con el apartado sonoro a nivel de programación.

Diseño, prototipado y programación del minijuego Ganzúa. Desarrollo de este primer minijuego en fases tempranas, involucrando gestos accesibles y usos de voces explicativas TTS.

Desarrollo y escritura del guion base del juego, junto con el diseño de la narrativa en el juego. Varias aproximaciones sobre la historia inicial, contando con diferentes versiones que propiciaban distintos diseños de juegos. Esta tarea se ha realizado teniendo en cuenta al resto del equipo, cuyas ideas e intervenciones encaminaban la historia y perfilaban el diseño narrativo del juego.

Encargado de aplicación, configuración y generación de versiones de esta. Aproximaciones sobre la versión de Android a utilizar, preferencias de aplicación y empaquetado, además de la creación y preparación inicial del proyecto de **Unity** en **GitHub**.

Labor de enseñanza sobre **Unity** e instrucción en su uso al grupo de Ingeniería Informática. Ayuda con la integración en la aplicación de las pantallas de registro e inicio de sesión realizadas por Arturo Aguirre Calvo, con su consiguiente aprendizaje sobre llamadas al servidor y base de datos desde **Unity**.

Creación de formularios para probadores del y análisis de los datos recibidos. Este apartado se ha realizado en colaboración con Héctor Marcos Rabadán.

10.4. Fernando Cortés Sancho

Investigación y aprendizaje sobre el motor de videojuegos multiplataforma **Unity** con el fin de comprender su estructura, funcionamiento y sus posibilidades. Se incluye también una investigación sobre el lenguaje de programación utilizado, **C#** y la manera en la que se integra con el motor.

Investigación sobre *Text to Speech* y sus posibilidades de implementación. Esto incluye investigación sobre *APIs* y el desarrollo de un prototipo con llamadas a una *API* de *TTS* en **C#**, así como su integración con **Unity**.

Investigación sobre *plugins* y sus posibilidades de integración con **Unity**. Esto incluye investigación sobre el entorno de desarrollo **Android Studio** y el estudio de las clases que ofrece Android en el lenguaje de programación **Java**. Se incluye también el desarrollo de los *plugins* de *Text to Speech* y de *Speech to Text*, además de su integración con el motor **Unity**. Para la integración se contó con la ayuda de Diego Martínez Simarro.

Estudio e investigación sobre accesibilidad web. Esto incluye el estudio de las pautas de accesibilidad en el contenido web (**WCAG**) publicados por la **Web Accessibility Initiative** (rama pertenece a la comunidad **World Wide Web Consortium** encargada de la accesibilidad en la web). Se incluye también la realización de un curso online ofrecido por profesores de la Universidad de Alicante en la plataforma **Udemy** y una investigación a cerca de los lectores de pantalla en ordenadores.

Diseño de la página web y desarrollo de la parte de *front-end*, aplicando los conocimientos obtenidos sobre accesibilidad que se han explicado a lo largo de la memoria y buscando que fuera accesible para usuarios que utilizan lectores de pantalla. Esto incluye también el diseño de la web adaptable a los diferentes tipos de pantalla, especialmente en la pantalla del teléfono móvil.

Implementación la parte de *back-end* de la página web, que se encarga de la lógica de la web y su funcionamiento, así como la conexión con la base de datos.

Investigación sobre el uso de *Cookies* para su posterior implementación en la página web, con el fin de conseguir una mejor experiencia para el usuario.

Las tareas de la página web se han realizado de manera conjunta con los alumnos Arturo Aguirre Calvo y Eduardo Andrés Morais.

Investigación sobre lectores de pantalla en dispositivos móviles con el sistema operativo **Android**, en concreto **Google TalkBack**, con el fin de comprender la manera en la que los usuarios interactúan con el software y poder hacer pruebas en la navegación de la página web en móvil.

Realización de pruebas de accesibilidad en la página web, utilizando las herramientas y validadores ofrecidos por la **W3C**.

Diseño de recursos gráficos para su uso en la página web mediante **GIMP**. Esto incluye la modificación de imágenes y la creación de animaciones, como el logo del videojuego con la onda en movimiento, o los que se utilizan en las secciones de los minijuegos, para una mejor explicación de estos.

Labor auxiliar en el diseño de la base de datos en las iteraciones que se hicieron según surgían nuevas necesidades en el proyecto.

10.5. Héctor Marcos Rabadán

Labores de investigación sobre accesibilidad en videojuegos para personas ciegas, centrada en los recursos, mecánicas y controles que se utilizan de manera general en juegos de este tipo. En particular se han estudiado las mecánicas basadas en sonido y vibración, con el fin de crear una jugabilidad innovadora en el ámbito de los juegos accesibles.

Creación de conceptos de minijuegos accesibles para personas ciegas, basados en los resultados obtenidos de las investigaciones previas. Estos conceptos fueron la base de casi todos los minijuegos que se encuentran en el proyecto.

Desarrollo completo del minijuego Laberinto Sonoro, basado en la percepción del audio 3D y los *quick time events*. Esto incluye los apartados de prototipado inicial, diseño del juego y programación de todos los elementos de juego necesarios, así como su implementación en una escena de Unity. También se incluye la búsqueda y edición de alguno de los efectos de sonido encontrados en el minijuego, así como la edición e implementación de audio 3D por medio de las herramientas ofrecidas por el propio motor.

Desarrollo completo del minijuego Búsqueda por Vibración, apoyándose en el sistema de vibración creado. Esto incluye los apartados de prototipado inicial, diseño del juego y programación de todos los elementos de juego necesarios, así como su implementación en una escena de Unity. Se añaden además las labores de búsqueda, edición e implementación de los efectos de sonido requeridos para el minijuego.

Desarrollo completo del minijuego Reconocimiento de Formas por Vibración, haciendo uso tanto del sistema de vibración como del sistema de gestos. Esto incluye los apartados de prototipado inicial, diseño del juego y programación de todos los elementos de juego necesarios, así como su implementación en una escena de Unity. También se tiene en cuenta la creación de las formas o modelos requeridos para el funcionamiento del juego, su adaptación continua en función del *feedback* recibido durante las pruebas, y la integración con el sistema de lectura de pantalla, en colaboración con Diego Martínez Simarro.

Implementación de la funcionalidad del menú inicial, en particular el comportamiento explicado en el apartado sobre menús deslizantes.

Colaboración en la gestión y unión de los minijuegos con el flujo general del juego.

Desarrollo completo del minijuego *Simon Says* Sonoro, basado de nuevo en el audio 3D y en el juego electrónico de mismo nombre. Esto incluye los apartados de prototipado inicial, diseño del juego y programación de todos los elementos de juego necesarios, así como su implementación en una escena de **Unity**. Se incluye además la búsqueda e implementación de los efectos de sonido necesarios, y su conversión o adaptación para tratarlos como sonido estereofónico.

Creación de un formulario para probadores centrado en las características del videojuego, con el fin de analizar los resultados y poder realizar los cambios pertinentes en función de estos. Se realizó en colaboración con Alberto Casado Trapote.

Labores de equilibrado de los minijuegos en función del *feedback* recibido durante las múltiples fases de prueba del proyecto. Este equilibrado varía desde la variación de ciertos parámetros de juego desde el editor de **Unity** hasta la modificación o eliminación completa de mecánicas confusas o problemáticas.

10.6. Diego Martínez Simarro

Investigación generalista sobre accesibilidad en el medio de los videojuegos, en el ámbito de los videojuegos para consola, ordenador y dispositivos móviles, con el fin de encontrar prácticas habituales y referentes que las practiquen. Se puso especial atención en la accesibilidad referente a discapacidades visuales y en dispositivos móviles.

Investigación sobre opciones de accesibilidad en **Unity**, las iniciativas propuestas por los desarrolladores oficiales del motor y las herramientas creadas por particulares. En particular sobre lectores de pantalla, su posible inclusión y opciones alternativas a ellos en este motor. Fue de especial interés la experiencia de los desarrolladores de **FREEQ** (Caso de estudio: FREEQ).

Labor de investigación sobre el funcionamiento y modo de uso del lector de pantalla **TalkBack** en colaboración con Alberto Casado Trapote. Hecha de forma previa a la implementación de los respectivos sistemas de lectura y de gestos.

Diseño e implementación del sistema de lectura de pantalla, con el que se permite recorrer elementos organizados en listas mediante los gestos establecidos. Estos gestos, la forma en que se leen y el resto de los comportamientos se realizaron siguiendo el ejemplo de **TalkBack**.

Implementación de los elementos interactivos de la aplicación que funcionan en coordinación con el sistema de lectura de pantalla de implementación propia, previo diseño de su estructura. Esto se realizó en concordancia con el lector propio, buscando la flexibilidad, de modo que todos los elementos interactivos del juego usan este sistema.

Arquitectura general de la aplicación. Esto incluye el flujo de juego, con la gestión de los datos de escenas, su instanciación en base a datos de guardado o por defecto y la coherencia de estos datos entre las distintas escenas. En estas tareas relativas a la infraestructura del juego también cabe mencionar su coordinación con el sistema de lectura de pantalla y la generación de los elementos que puede leer. Estas tareas se realizaron en amplia colaboración con Alberto Casado Trapote.

Sistema de guardado de datos del juego en fichero local. La serialización y deserialización de estos datos de guardado y estadísticas, además de la coordinación con el servidor desde la aplicación para el correcto funcionamiento del guardado web.

Investigación sobre las opciones de vibración proporcionadas por **Unity** y alternativas ofrecidas por terceros. Estudio de los sistemas nativos de las plataformas **Android** y **iOS** y sus diferencias. En base a la información obtenida, implementación del sistema de vibración en **Unity** orientado a los diseños de minijuegos planteados, haciendo uso de las funcionalidades de **Android**. Quedando así contenido en una clase que da acceso a estos servicios al resto de la aplicación.

Investigación sobre los servicios de *TextToSpeech* ofrecidos de forma oficial en la plataforma **Android** y las diferentes formas de crear un *plugin* **Android** para su inclusión en **Unity**.

Integración en el proyecto de **Unity** de los *plugins* de TTS y SST mediante clases para ambos que gestionan sus llamadas y dan acceso a estos servicios al resto de la aplicación.

Implementación de la clase común que se encarga de las llamadas al servidor, en base a las anteriores implementaciones. Gestión de dichas llamadas desde la aplicación y de la coordinación entre modo en línea y sin conexión.

Creación de los recursos gráficos de los menús y del logotipo del juego en **GIMP**, siguiendo el esquema de color general planteado con colores de alto contraste y buscando claridad con iconos distintivos de gran tamaño.

Labor auxiliar en la grabación de audios del juego y grabación de la voz del protagonista.

11. Referencias

A11Y. (12 de 05 de 2018). *Screen Readers & Browsers! Which is the Best Combination for Accessibility Testing?* Obtenido de https://www.digitala11y.com/screen-readers-browsers-which-is-the-best-combination-for-accessibility-testing/

A11Y. (2020). *Color Contrast Accessibility Validator*. Obtenido de https://color.a11y.com/

Android Inc. (05 de 05 de 20). *Android Recognicer Intent*. Obtenido de https://developer.android.com/reference/android/speech/RecognizerIntent

Android Inc. (2020). *Activity*. Obtenido de https://developer.android.com/reference/android/app/Activity

Android Inc. (16 de 3 de 2020). *Android Developers*. Obtenido de https://developer.android.com/reference/android/os/VibrationEffect

Android Inc. (16 de 3 de 2020). *Android Developers*. Obtenido de https://developer.android.com/reference/android/os/Vibrator

Android Inc. (2020). *Android Developers*. Obtenido de https://developer.android.com/studio

Android Inc. (05 de 05 de 2020). *Android TextToSpeech*. Obtenido de https://developer.android.com/reference/android/speech/tts/TextToSpeech

Apache Software Foundation. (1995). Apache. Obtenido de apache: https://www.apache.org/

Apple Inc. (29 de 4 de 2005). VoiceOver. EEUU.

Apple Inc. (29 de 6 de 2007). iOS. EEUU.

Asociación Doce. (2017). *Escala de Wecker*. Obtenido de https://asociaciondoce.com/escala-de-wecker/

Audacity. (2000). *Paulstretch*. Obtenido de https://manual.audacityteam.org/man/paulstretch.html

Audacity Team. (Mayo de 2000). Audacity. Estados Unidos.

Baer, R. H., & Morrison, H. J. (1977). *US Patente nº 4207087.*

BBC. (1 de 4 de 2019). *BBC Subtitle Guidelines*. Obtenido de http://bbc.github.io/subtitle-guidelines/

Behling, M. (2008). LUBUNTU.

Bethesda Game Studios. (8 de Diciembre de 2011). The Elder Scrolls V: Skyrim. Estados Unidos.

Bootstrap. (2020). *Get Bootstrap*. Obtenido de https://getbootstrap.com/

Boucherot, A. (Enero de 2010). *Ulule*. Obtenido de https://es.ulule.com/

Brown, M. (22 de 11 de 2019). *YouTube*. Obtenido de https://www.youtube.com/playlist?list=PLc38fcMFcV_vvWOhMDriBlVocTZ8mKQzR

Canonical Ltd. / Fundación Ubuntu. (2004). Ubuntu.

Cockos. (23 de Agosto de 2006). Reaper. Estados Unidos.
Colorsafe. (2020). *Colorsafe.co*. Obtenido de http://colorsafe.co/
CSS-Tricks. (26 de 11 de 2019). *Five Methods for Five-Star Rating*. Obtenido de https://css-tricks.com/five-methods-for-five-star-ratings/
Discord Inc. (13 de 5 de 2015). Discord. EEUU.
DOWINO. (3 de 7 de 2014). *Campaña A Blind Legend en ULULE*. Obtenido de https://es.ulule.com/a-blind-legend/
Dowino. (3 de 7 de 2014). *Ulule*. Obtenido de https://es.ulule.com/a-blind-legend/
Duplessis, T. (2010). *lichess.org*. Obtenido de https://lichess.org
Epic Games Inc. (2020). *Unreal Engine Documentation*. Obtenido de https://docs.unrealengine.com/en-US/BlueprintAPI/Widget/Accessibility/SetColorVisionDeficiencyType/index.html
Esfuerzo colaborativo (Ellis, Ford-Williams, Graham, Grammenos, Hamilton, Lee, Manion & Westin). (20 de 10 de 2019). *Game Accesibility Guidelines*. Obtenido de http://gameaccessibilityguidelines.com/
Ettrich, M. (1996). KDE.
Flaticon. (2020). *Flaticon*. Obtenido de https://www.flaticon.com/
Freedom Scientific. (1 de 1995). JAWS. EEUU.
Freesound Team. (5 de Abril de 2005). *Freesound*. Obtenido de https://freesound.org/
Freslon, B. (3 de 1 de 2020). *GitHub*. Obtenido de https://github.com/BenoitFreslon/Vibration
Fundación GNOME. (1999). The GNOME project/Proyecto GNU.
Gach, E. (4 de 2 de 2017). *Kotaku*. Obtenido de https://compete.kotaku.com/blind-player-racks-up-a-win-at-his-first-street-fighter-1793936241
Garaventa, B. (2013). *How Browsers Interact with Screen Readers and Where ARIA Fits in the Mix*. Obtenido de https://www.levelaccess.com/how-browsers-interact-with-screen-readers-and-where-aria-fits-in-the-mix/
Github Inc. (16 de 10 de 2007). *GitHub*. Obtenido de https://github.com/
Glasdom. (2017). *Glasdom. Estadísitcas del internet español*. Obtenido de https://web.archive.org/web/20170214002305/https://www.glasdom.es/estadisticas-del-internet-espanol
Google. (23 de 8 de 2008). Android. EEUU.
Google. (2011). Google Talkback. EEUU.
Google. (27 de 4 de 2012). *Google Drive*. Obtenido de https://www.google.com/drive
Google. (8 de 12 de 2014). Android Studio.
Google. (2020). *Android Accesibility Help*. Obtenido de https://support.google.com/accessibility/android/answer/6151827?hl=en

Hammilton, I. (1 de 7 de 2019). *applevis*. Obtenido de https://www.applevis.com/forum/unity-screenreaders-progress

HTML.COM. (30 de 06 de 2019). *23 Free Tools To Make Sure Users With Disabilities Can Enjoy Your Content*. Obtenido de https://html.com/blog/25-website-accessibility-checker-tools/

HTML5UP. (2020). *HTML5UP*. Obtenido de https://html5up.net/

Hughes, D. (20 de 11 de 2013). *gamasutra*. Obtenido de https://www.gamasutra.com/blogs/DianaHughes/20131120/205346/

I AM RESPONSIVE. (14 de 09 de 2013). *RESPONSIVE DESIGN*. Obtenido de http://ami.responsivedesign.is/

IBM. (1974).

Interactive Accesibility Inc. (2019). *iOS & Android Screen Reader Gesture Reference Cheatsheet*. Obtenido de https://www.interactiveaccessibility.com/blog/mobile-screen-reader-gestures#.XtqimkUzaUk

Jenny, B. (5 de 5 de 2018). *Color Oracle*. Obtenido de https://colororacle.org/

Lab Zero Games. (10 de 4 de 2012). Skullgirls.

Lab Zero Games. (26 de 4 de 2014). *Notas de parche Skullgirls (Encore 1.01 Patch).* Obtenido de http://skullgirls.com/2014/04/skullgirls-encore-1-01-patch-notes/

Lerdorf, R. (1995). PHP. Groenlandia. Obtenido de php: https://www.php.net/

Llu, M. (13 de 6 de 2019). *Unity forums*. Obtenido de https://forum.unity.com/threads/accessibility-and-inclusion.694477/

Microsoft. (25 de 10 de 1983). Microsoft Word. EEUU.

Microsoft. (20 de 11 de 1985). Microsoft Windows. EEUU.

Microsoft. (16 de 8 de 1995). Internet Explorer. EEUU.

Microsoft. (2000). C#. Washington, EEUU.

Microsoft. (2012). Microsoft Helena. Estados Unidos.

Microsoft. (14 de 3 de 2017). Microsoft Teams. EEUU.

Microsoft. (2019). *Xbox adaptative controller.* Obtenido de https://www.xbox.com/en-US/accessories/controllers/xbox-adaptive-controller

MongoDB Inc. (2009). MongoDB.

Mozilla Foundation. (23 de 9 de 2002). Mozilla Firefox. EEUU.

MySQL AB, Sun Microsystems, Oracle Corporation. (23 de mayo de 1995). MySQL. Obtenido de mysql: https://www.mysql.com/

Native Instruments. (2012). Kontakt. Alemania.

Natterer, M. (24 de 2 de 2020). *Gimp*. Obtenido de https://www.gimp.org/

Naughty Dog. (14 de Junio de 2013). The Last of Us. Estados Unidos.

Netflix. (20 de 8 de 2019). *Netflix Studios Partner Help*. Obtenido de https://partnerhelp.netflixstudios.com/hc/en-us/articles/215758617-Timed-Text-Style-Guide-General-Requirements
NetherRealm Studios. (3 de 4 de 2013). Injustice: Gods Among Us. Estados Unidos.
Netscape Communications. (1995). JavaScript. EEUU.
NVDA. (2020). *NVDA*. Obtenido de https://nvda.es/
OMS. (11 de 10 de 2018). *Blindness and visual impairment*. Obtenido de https://www.who.int/es/news-room/fact-sheets/detail/blindness-and-visual-impairment
Pederick, C. (13 de 01 de 2020). *Web Developer Tool Firefox*. Obtenido de https://addons.mozilla.org/es/firefox/addon/web-developer/
Pickering, H. (31 de 03 de 2017). *Inclusive Components toggle button*. Obtenido de https://inclusive-components.design/toggle-button/
Proyecto Debian. (1993). Debian.
Psychic Bunny. (2016). *psychicbunny*. Obtenido de http://www.psychicbunny.com/
RapidAPI. (2019). *RapidAPI*. Obtenido de https://rapidapi.com/voicerss/api/text-to-speech-1/endpoints
Robert, R. (10 de 12 de 2019). *GitHub*. Obtenido de https://gist.github.com/ruzrobert/d98220a3b7f71ccc90403e041967c46b
Skype Technologies. (29 de 8 de 2003). Skype. Luxemburgo.
Srinivas. (3 de 05 de 2018). *Github Speech Recognition*. Obtenido de https://github.com/gsssrao/UnityAndroidSpeechRecognition/tree/master/SpeechPlugin
Stallman, R. (1983). GNU.
Straub, J. (29 de 1 de 2014). *DAGERSystem*. Obtenido de https://dagersystem.com/netherrealm-studios-patches-accessibility-mode-into-injustice-gods-among-us/
Stylish. (03 de 03 de 2020). *Stylish extension*. Obtenido de https://chrome.google.com/webstore/detail/stylish-custom-themes-for/fjnbnpbmkenffdnngjfgmeleoegfcffe?hl=es
Sun Microsystems. (23 de 5 de 1995). Java. EEUU.
Thatcher, J. (24 de 07 de 2006). *WEB ACCESSIBILITY: WEB STANDARDS AND REGULATORY COMPLIANCE*. Obtenido de https://jimthatcher.com/book2/chapter06.html
The LXQt Team. (3 de 7 de 2013). LXQt.
Torvalds, L. (17 de septiemrbe de 1991). Linux. Obtenido de linuxfoundation: https://www.linuxfoundation.org/
Udemy. (02 de 2015). *Curso Accesibilidad*. Obtenido de https://www.udemy.com/course/aprende-accesibilidad-web-paso-a-paso/

Ullman, C. (2007). *Beginning Ajax.*
Unity Technologies. (2005). Unity. Copenhague, Dinamarca.
Unity Technologies. (2020). *https://assetstore.unity.com/*. Obtenido de https://assetstore.unity.com/
Valve Corporation. (12 de Septiembre de 2003). *Steam*. Obtenido de https://store.steampowered.com/?l=spanish
Varios. (1995). ApacheFriends. Obtenido de ApacheFriends: https://www.apachefriends.org/es/index.html
Vigil, J. (12 de 9 de 2013). *kickstarter*. Obtenido de https://www.kickstarter.com/projects/thesingularity/freeq-an-interactive-radio-drama-indie-video-game?lang=es
Voircerss Org. (2018). *Voice RSS*. Obtenido de http://www.voicerss.org/api/default.aspx
W3C. (17 de 12 de 1996). CSS.
W3C. (03 de 2016). *Web Accessibility Evaluation Tools List*. Obtenido de https://www.w3.org/WAI/ER/tools/
W3C. (5 de 6 de 2018). *WCAG 2.1*. Obtenido de https://www.w3.org/TR/WCAG21/
WAI . (2018). *WCAG 2.1 Technique 68*. Obtenido de https://www.w3.org/WAI/WCAG21/Techniques/general/G65.html
WAI. (27 de 07 de 2019). *Custom controls*. Obtenido de https://www.w3.org/WAI/tutorials/forms/custom-controls/
WAI-ARIA. (2018). *Modal Dialog Example*. Obtenido de https://www.w3.org/TR/wai-aria-practices/examples/dialog-modal/dialog.html
WAI-ARIA. (14 de 08 de 2019). *WAI-ARIA Authoring Practices*. Obtenido de https://www.w3.org/TR/wai-aria-practices-1.1/
WhatsApp Inc. (1 de 2009). WhatsApp.
WHATWG. (1993). HTML.
World Wide Web Consortium and Internet Engineering Task Force. (1991). HTTP. Obtenido de wikipedia http.
Ylönen, T. (1995). SSH.

12. Anexos

12.1. Guion base

Leyenda:
() -> Comentario provisional, de situación.
** -> Comentario de acción.
- -> Dialogo.

(Sonido de insectos, flores, típico)
Enfrente de una casa en ruinas
- Aquí fue. Aquí fue donde todo ocurrió.
(Flashback de casa en llamas, fuego, y devuelve al presente)
Camina hacia la puerta, pasando por escaleras de madera
- No me había atrevido a volver a este sitio hasta este momento. Creo que estoy preparado.
Rompe cuerdas policiales de la puerta
(Entra en minijuego de "ganzúa")
Se abre la puerta, entra y la cierra. Enciende linterna en el móvil
Avanza hacia unas escaleras, se rompe un peldaño y cae hacia abajo
(Se despierta tras un duro golpe con el móvil roto totalmente a oscuras)
(Escucha ecos de su hija desde el fondo de la habitación)
Empieza a perseguir la voz de su hija
(Entra en minijuego “carrera”, en el que la voz de su hija con ecos y movidas le guían.
El prota mientras grita cosas como “No corras cariño” o “Puedo oírte hija, ven hacia mí”)
* Acaba la carrera con golpe de puerta cerrada (“La niña ha pasado y ha cerrado”) *
- Ábreme la puerta… ¿Por qué huyes de mi…?
(Silencio moderado)
- Esta habitación… ¡Es mi dormitorio!
(Se entra en modo “escape room”, con intención de pasar la puerta hacia la habitación de su hija)
(El prota hace comentarios a veces del estilo “Espérame hija hallaré la forma de entrar”)
(Aquí se insertan minijuegos estilo hallar forma con vibración, encontrar llave en la pared con vibración, etc.…)

Abre la puerta, es la habitación de su hija
- ¿Dónde estás…?
(Se insertan minijuegos tipo Simón dice con un Simón de verdad de la hija, esto provoca algún flashback informativo; “Este era el juguete favorito de X” -> te lo pasas y sale flashback pequeño, cosas así)

Se acaba descubriendo que el padre provocó el incendio en la casa sin querer, matando a su hija mientras estaba en su cuarto

OPCIONALES:
--
El padre está en el hospital debido al primer golpe en la cabeza, nada de lo que ha visto ha pasado, eran alucinaciones. Está con la mujer y la mujer le da esperanzas tipo “no fue culpa tuya”.
--

12.2. Entrevista a persona con problemas de visión

Esta entrevista se realizó durante las fases tempranas de diseño del juego, con el fin de hallar información sobre accesibilidad y conceptos generales sobre esta disciplina.

10 DE OCTUBRE DEL 2019 A LAS 15:30, Facultad de Informática, UCM.
Duración de la entrevista: 30 minutos.

12.2.1. Asistentes

Entrevistador: Alberto Casado Trapote
Entrevistado: David Pacios Izquierdo

12.2.2. Notas previas

David Pacios Izquierdo tiene graves problemas de visión, y también es afectado por una enfermedad rara de la piel que, a grandes rasgos, provoca estiramientos indeseados con sangrado.

12.2.3. Entrevista

¿Qué rango visual/porcentaje de visión posees?

En cierta escala legal (Asociación Doce, 2017), posee un valor de 0'05. Esta visión es muy baja. Posee una discapacidad mayor al 65%.

¿Cuál es tu rutina diaria?

Al levantarse, debe comprobar su cuerpo entero por si tiene heridas en la piel. También debe preocuparse por el dolor en los ojos, y temblores por el cuerpo. Su mujer le ayuda en todo lo que puede. Por la mañana ve mejor que por la tarde. Tiene que llevar palo para ciegos, pero de costumbre no lo lleva. En ocasiones la gente se ha burlado de él y ha llegado incluso a agredirle por desconocimiento de su enfermedad. Esto es propiciado por el desconocimiento general sobre las discapacidades (la gente pone en duda que alguien sea ciego o vea mal si no lleva gafas de sol u otros aparatos).
A medida que avanza el día, tiene que forzar los ojos, lo que provoca dolor y peor visión. Su capacidad auditiva ha mejorado con el avance de la enfermedad para cubrir ciertas necesidades. El desplazamiento por lo general es complicado, por sitios desconocidos necesita ayuda.

¿Qué tecnología accesible utilizas? ¿Cómo?

Utiliza un programa de lectura en voz alta para el móvil (Android) y para el ordenador (Windows), los cuales cada vez usa menos. Este lector de pantalla dice en alto todos los mensajes que recibe. El mal uso del lenguaje a la hora de escribir provoca que el lector falle,

y pierda mucho tiempo deletreando palabras en lugar de decirlas de forma normal. Por ejemplo, al leer una palabra escrita “AVRNIDA” debe deletrearla “A V R N I D A” haciendo que pierda mucho tiempo y a veces no entienda lo que dicen. Esto también pasa con el lenguaje inclusivo (NOSOTRXS - N 0 S O T R X S).
El ‘@’ en los sistemas que utiliza es la forma de hacer que el lector grite, por lo que el utilizarlo de forma distinta también provoca “fallos” en el sistema.
En el móvil no puede tener casi nada, aparte de lo esencial. El lector a veces se hace muy largo con ciertos mensajes.
También poseía una especie de “lupa lectora de libros”, la cual colocaba en cualquier libro y lo leía en alto. La lupa estaba bien, pero sólo funcionaba con libros nuevos que no tuviesen ninguna pintada ni subrayado, por lo que dejó de usarlos.

¿Qué aficiones tienes? ¿Qué es lo que más te gusta hacer?

Puede ir al cine, pero siempre que sea en español y en ciertas posiciones. Se imagina lo que pasa en la película, y a veces puede ver ciertos fragmentos. Algunas personas no quieren ir con él al cine por las posiciones en las que se tiene que sentar, igual que en conciertos.
El teatro es algo que sí que le gusta y suele ir con frecuencia. Antes de ir, tiene que conocer más o menos la obra, e igualmente sentarse en una buena posición para ver.
Él toca música, tiene un buen oído. En su tiempo libre genera canciones a ordenador, toca instrumentos, etc.
En lo referente a videojuegos, puede jugar muy poco. Juega casi a ciegas, y se cansa muy rápido. Lo echa de menos.
Para leer, lo mejor en su opinión es Kindle, ya que tiene muchas opciones de accesibilidad y edición. Aun así, ya casi no lo usa. El uso constante propicia que tenga que comprar de forma frecuente y cuesta mucho dinero.
Le encanta pasear, pero es algo que tampoco puede hacer de forma normal ni de costumbre. Normalmente necesita ayuda, se puede perder. Cuando viaja en metro, puede perderse también ya que a veces ciertas líneas no avisan de la parada actual o están estropeados los altavoces.
Le encanta viajar también, pero sólo puede hacerlo en determinadas situaciones y condiciones. Siempre debe tener un hospital cerca por si pasara cualquier complicación.

¿Cómo te orientas espacialmente?

Su percepción espacial es mala. Se choca a menudo si no conoce el lugar o no tiene ayuda/compañía. Ha llegado a tener serias heridas por choques con columnas o mobiliario.
Se ayuda mucho de su oído (muy desarrollado).

¿Cuáles son tus temas preferidos?

El tema que más le gusta es la fantasía. Todo lo relacionado con dragones, caballeros, elfos, etc. También le gusta la ciencia ficción (tema de espacio). Lo que menos le gusta son las novelas realistas. Todo lo que ha pasado en su vida hace que cada vez le guste menos escuchar historias realistas, quiere evadirse a otros mundos.

¿Qué videojuegos te gustaría jugar y no puedes?

El **Counter Strike** le gusta mucho, pero ya no puede jugarlo. Los juegos de disparos en general ya no los puede jugar. También le gusta el **Fire Emblem**, pero tampoco puede. La cantidad de colores que tiene el juego le hacen daño y no consigue verlos bien. Se pasó el primer juego de South Park, y le gustó mucho, pero ya no puede jugar a los siguientes.
También le gustaría jugar al nuevo juego de Doraemon, pero tiene unos gráficos "raros" que también le dañan los ojos.
Le gusta mucho ver anime, pero es complicado ya que algunos no están traducidos oralmente y no puede verlos. Se planteó aprender varios idiomas para poderlos ver en su lenguaje original (Ya sabe varios idiomas).

¿Qué mejoras propondrías para los sistemas actuales de accesibilidad?

Los sistemas actuales, pese a ser pocos, no están nada mal. El problema no son las aplicaciones ni las máquinas, son las personas. La gente actúa de forma indebida con los sistemas comunes, lo que hace que personas con discapacidades lo tengan más difícil a la hora de utilizarlos. Esto provoca también que se puedan relacionar de formas más complicadas que las personas sin discapacidades.

¿Hay algo más que deba saber o alguna pregunta que no haya hecho?

Pregunta: En caso de tener visibilidad parcial, ¿qué colores dañan más los ojos?
Respuesta: Hay algunos colores que son más inclusivos que otros. Determinados tipos de negros, verdes y naranjas son buenos para personas con visibilidad reducida. La mayoría de las páginas web no están adaptadas, por lo que hacen daño. Un buen ejemplo de página inclusiva es lichess.org (Duplessis, 2010). Esta, en su modo accesible, tiene un negro grisáceo de fondo no dañino, y sus botones y textos son blancos no brillantes y naranjas pálidos, que a la vez son muy visibles y poco dañinos.

CONCLUSIONES GENERALES

El mundo no están lo suficientemente informado sobre las personas con discapacidades. La tecnología actual ayuda mucho, pero el uso erróneo de esta hace muy difícil su labor y las personas ciegas tienen muchos problemas a la hora de vivir en sociedad.

12.3. Informe sobre influencia del COVID-19 en el TFG

Este documento se realizó a finales de abril para proporcionar un informe al vicedecano sobre el modo en que la pandemia del COVID-19 y la situación de cuarentena habían afectado a este TFG.

Informe influencia COVID-19
Nombre del TFG:
Desarrollo de un juego narrativo e interactivo para personas ciegas.

Cambios causados por la situación
Afortunadamente el proyecto no ha sufrido cambios drásticos, ya que el enfoque telemático no ha conllevado impedimentos directos al desempeño del trabajo. Todas las reuniones pasaron a hacerse a través de Internet, tanto las realizadas entre los estudiantes para sacar adelante el trabajo, como las reuniones periódicas con los directores para comprobar el progreso, resolver dudas y dar orientación.

A lo que más ha afectado ha sido a la colaboración propuesta con la ONCE, por la cual nos proporcionarían soporte como probadores de la accesibilidad de lo desarrollado, acceso a una línea braille para comprobar la accesibilidad de la web, y por lo general resolver dudas y dar retroalimentación en cuestiones de accesibilidad. La situación ha imposibilitado esta colaboración, al eliminar posibles visitas presenciales, el proceso de pruebas ya mencionado y el contacto para resolución de dudas. También nos ha imposibilitado la realización de partes del trabajo que en principio iban a ser presenciales, como la grabación de voces.

Esto se traduce en que el desarrollo del TFG se ha dado sin realizar pruebas con personas ciegas, con las consecuencias que acarrea a nivel de desarrollo y de resultado final.

12.4. Imágenes de la página web

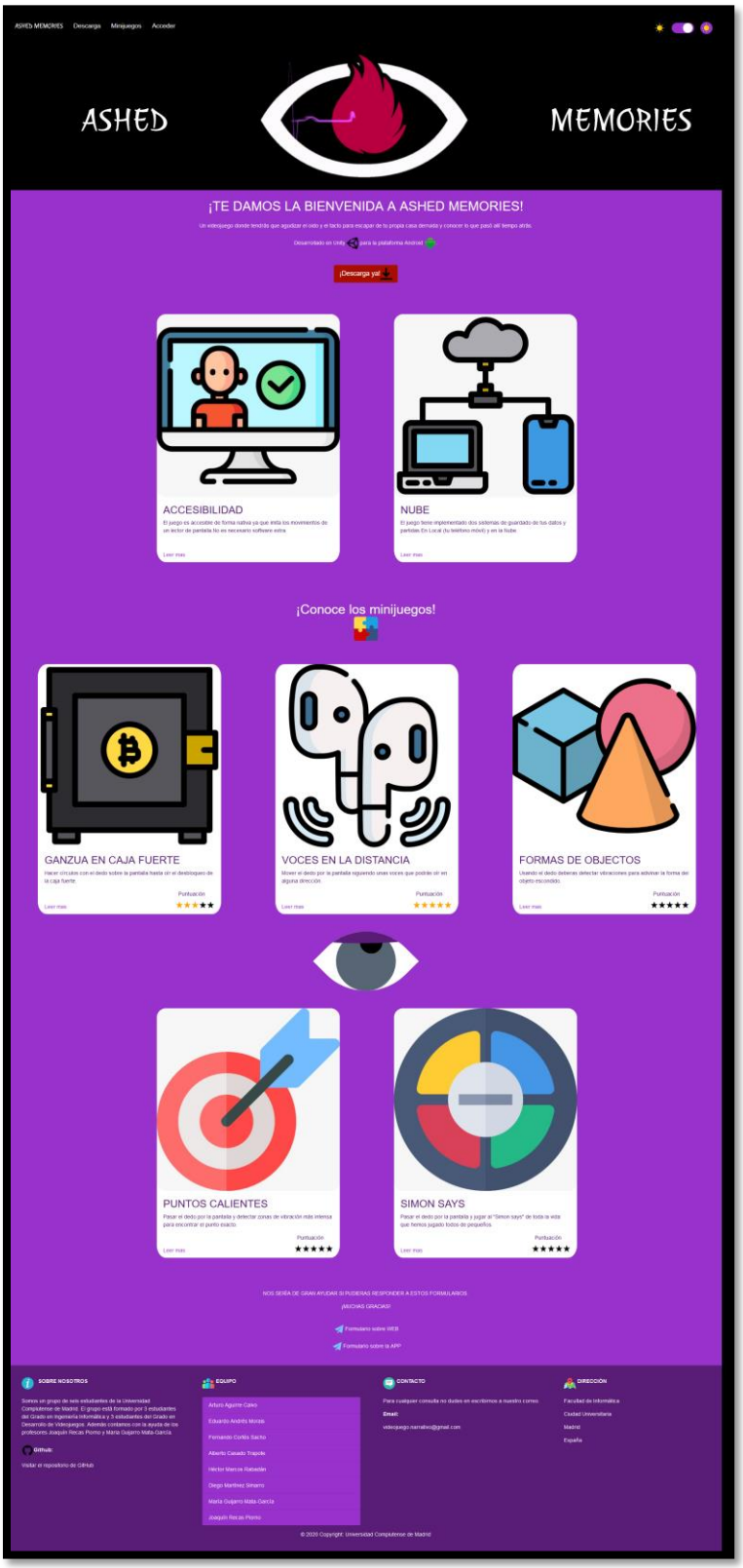


Figura 48 - Inicio. Página web.

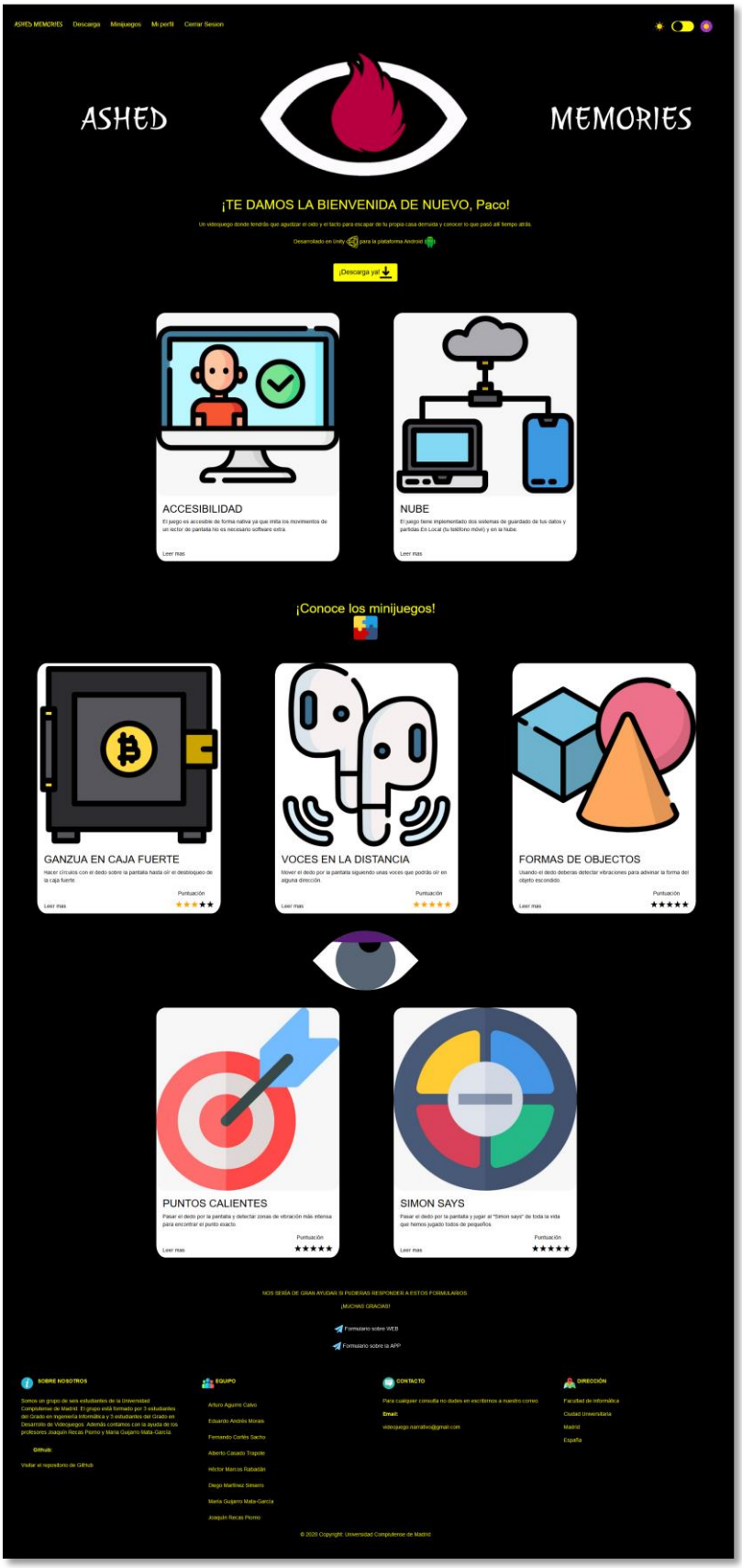


Figura 49 - Inicio con alto contraste. Página web

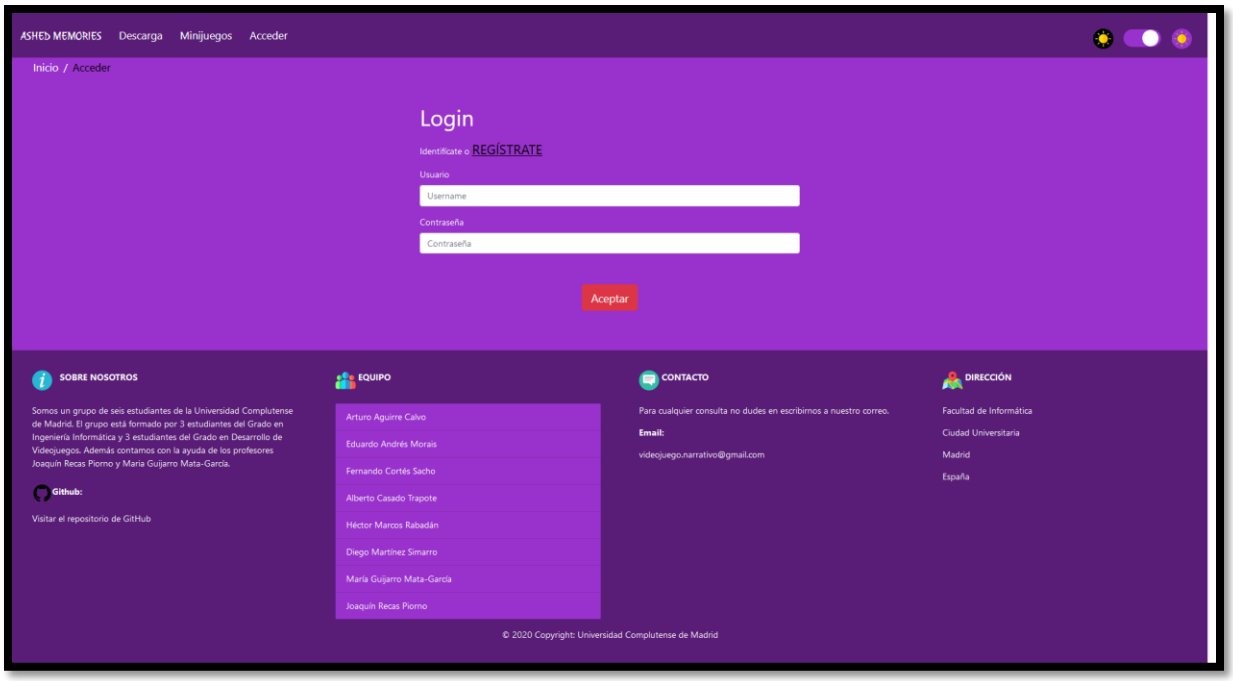


Figura 50 - Login. Página web.

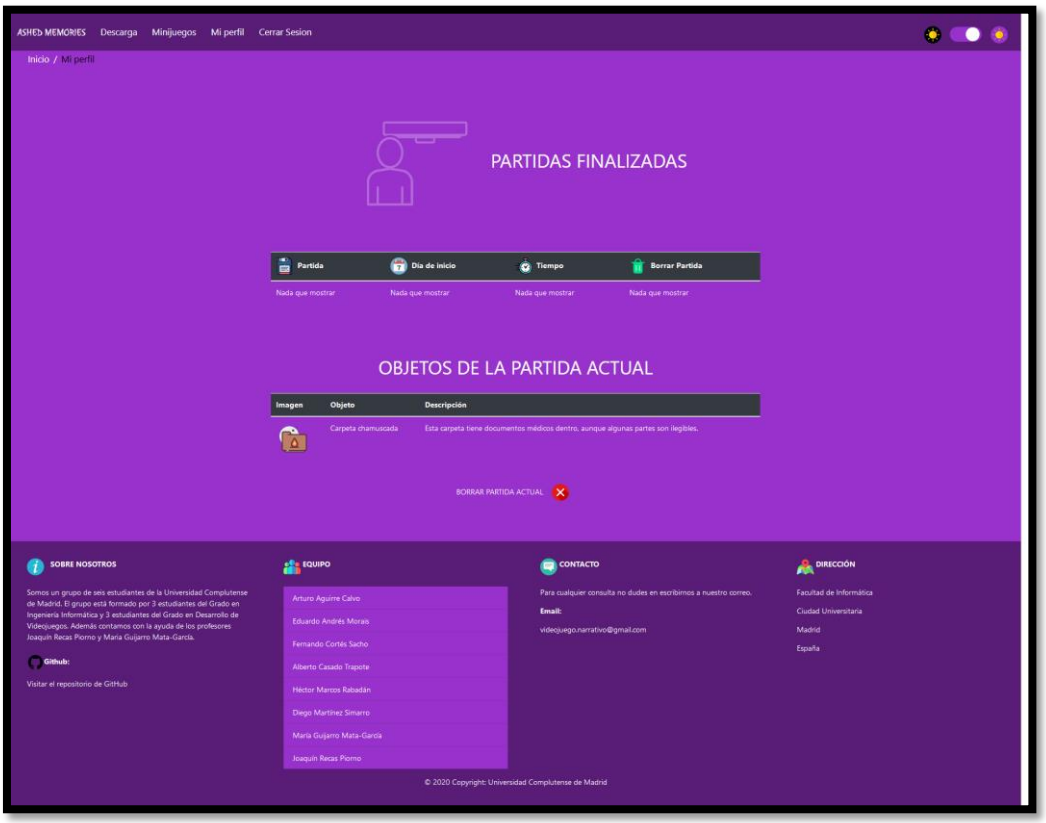


Figura 51 - Mi perfil. Página web.