Juegos accesibles para ciegos en plataformas móviles

Proyecto de sistemas Informáticos
2011/2012

Facultad de Informática

Universidad Complutense de Madrid

Gloria Esther Pozuelo Fernández
Fco. Javier Álvarez Obeso

Directores:
Baltasar Fernández Manjón
Javier Torrente Vigil

Se autoriza a la Universidad Complutense de Madrid a difundir y utilizar con fines académicos, no comerciales y mencionando expresamente a sus autores, tanto la memoria como el código, la documentación y/o el prototipo desarrollado

Gloria Esther Pozuelo Fernández

Fco. Javier Álvarez Obeso

Índice

Figuras

Tablas

Resumen

La creciente integración de dispositivos móviles inteligentes o *smartphones* en nuestras vidas ha tenido un alto impacto en actividades relacionadas con la informática, como navegar por internet, leer el correo electrónico, acceder a la información o jugar a videojuegos. Sin embargo, el auge de estas tecnologías puede suponer un motivo de división digital para las personas con discapacidad por las nuevas formas de interacción que introducen (por ejemplo, la interacción táctil).

Anteriormente a la llegada de la nueva generación de dispositivos móviles o *smartphones*, las personas con discapacidad visual se han apoyado en lectores de pantalla y en teclados como sistema de entrada a la hora de interactuar con las TIC (Tecnologías de la Información), lo que al menos en parte solucionaba el problema. Las nuevas tecnologías traen consigo un cambio en la manera de interactuar y por lo tanto es necesario adaptar el diseño de los videojuegos a estas tecnologías. Así pues, se tiene que informar al usuario sobre lo que ocurre en el juego mediante alternativas que no estén basadas en la vista, como la vibración, o eventos sonoros.

Con este proyecto se ha intentado promover esa accesibilidad hacia las personas con discapacidad visual, con el objetivo de romper las barreras que muchas veces las propias tecnologías imponen, desarrollando funcionalidades que permitan a todos disfrutar de ellas por igual.

El proyecto que se expone en el presente documento ha tenido como principal resultado *BFG Toolkit*, un conjunto de herramientas que facilitan el desarrollo de juegos accesibles a otros programadores. Este conjunto de herramientas dispone de clases que proporcionan una mayor comodidad a la hora de trabajar con los lectores de pantalla y sonido 3D permitiendo que la implementación sea más sencilla.

Para lograr este objetivo se han desarrollado un conjunto de juegos con los que poder abordar el problema de manera incremental, de forma que cada uno de ellos sirva como captura de requisitos y para incorporar nueva funcionalidad al sistema. Además, cada uno integra distintos modos de dificultad que posibilitan que el juego sea un reto para el usuario y un sistema de comentarios y estadísticas que permiten a desarrolladores tener un sistema de retroalimentación para poder mejorar la experiencia de juego para el usuario y una mayor cantidad de información y más detallada para los creadores.

Por último, con el objetivo de concienciar sobre la problemática de los discapacitados visuales en plataformas móviles, se ha integrado en todos los juegos un modo “pantalla en negro” que permite a los usuarios sin discapacidad visual experimentar las sensaciones de una persona ciega al interactuar con los controles y el sonido del juego.

Abstract

The crescent integration of intelligent devices (smartphones) in our lives has had a high impact in all the activities related with computer, like surfing the internet, read e-mails, access to information or play videogames. However, the rising use of this technology can mean a motive of digital division to the people with disability by using new forms of interaction (example: tactile interaction).

Previously to the arrive of the new generation of this mobile devices, the people with disability had used screen readers and keyboards as input systems when they wanted to interact with the IT (Information Technology), which could solve part of the problem. New technologies bring a change in the way of interact, and therefore is necessary to adapt the design of the videogames to this technologies. In this manner, the player must be informed about what is occurring in the game using alternatives not based on the sight.

This project has try to help along that accessibility to people with visual disabilities, with the main objective of breaking the walls that the technologies impose in many occasions, developing functions which allow to enjoy everyone in the same way.

The project exposed in this document has had as main result *BFG Toolkit*, which is a set of tools for developers to make easy create new accessible videogames. This set of tools has new classes that give more cosiness at working with screen readers and 3D sound, making the implementation easier.

In order achieve this objective we have developed a set of games to address the problem gradually, so that each serves as a capture requirements and to incorporate new functionality to the system. Moreover, each of them has different difficulty modes that allows the game to be a challenge to the player and a comment & statistics system which allows to the developers to have feedback in order to improve the game experience for the player and best information.

Finally, with the main objective of have in mind the trouble of the visual disability at the mobile devices, a "black screen mode" was integrated allowing users without any disability experiment the same feelings of a blind person when interact with the controls and the game sound.

Palabras clave

- Plataformas móviles
- Accesibilidad
- Discapacidad visual
- Discapacidad
- Videojuegos

Keywords

- Smartphones
- Android
- Tablets
- Video games
- Accessibility
- Disability

1. Introducción

El siguiente apartado introduce los motivos por los que se ha realizado un proyecto de estas características. A continuación, se discuten las posibles causas de la escasa accesibilidad en juegos y los objetivos del presente proyecto. También se incluye una breve sección con la terminología que es necesario conocer para poder comprender este documento y para finalizar, cómo se estructuran las siguientes secciones del mismo.

1.1. Motivación

Los videojuegos son una de las principales industrias del entretenimiento digital en términos absolutos de recaudación, llegando a superar los resultados de sector cinematográfico por ejemplo, la venta de videojuegos supuso unos ingresos de $15.900 millones en 2010 frente a $10.474 millones del cine según (ESA, 2012; Nash Information Services, 2012). En este sentido, los videojuegos se han convertido en parte de nuestra cultura, de acuerdo con *NPD Group*, empresa que lleva más de 25 años realizando estudios sobre datos de ventas de videojuegos, actualmente se pueden encontrar más de 100 millones de consolas solamente en los hogares de los Estados Unidos (NPD Group, 2007) y aproximadamente el 63% de la población de Estados Unidos juega a videojuegos, de los cuales un 51% lo hace al menos una vez por semana. Una posible explicación a su popularidad podría ser que los juegos proporcionan algo que otras formas de entretenimiento como los libros, música o películas no permiten: **una elevada interactividad**.

Pero más allá del puro entretenimiento, los videojuegos son un medio que puede servir como recurso educativo. Algunos docentes están descubriendo el potencial educativo de los juegos, cuya eficacia a la hora de mejorar el rendimiento de los alumnos ha sido contrastada en educación superior en campos tan diversos como la medicina (Kanthan & Senger, 2011; Rosser et al., 2007), la ingeniería (Ebner & Holzinger, 2007; Mayo, 2007), o la administración y dirección de empresas (Blunt, 2007) así como en educación primaria y secundaria (Annetta, Minogue, Holmes, & Cheng, 2009; Tuzun, Yilmazsoylu, Karakus, Inal, & Kizilkaya, 2009).

Son en definitiva, un recurso que puede ser muy eficaz, pues requiere que el jugador tome un papel mucho más activo, consiguiendo captar mejor su atención, además de aumentar la motivación del mismo al plantearle los objetivos a superar (Sedeno, 2010).

En su proceso de crecimiento, la industria de los videojuegos ha alcanzando nuevas plataformas para llegar a un público más amplio, como son los *Smartphones* o teléfonos móviles de última generación, que se han convertido en una de las principales tecnologías de juegos, comparables a algunas de las plataformas específicas de este ámbito, según datos de la propia asociación de software de entretenimiento[1].

[1] http://www.theesa.com

Entre 2009 y 2010, se ha reducido el gasto en ocio electrónico sobre los videojuegos tradicionales (consolas, juego, descargas) y se ha incrementando en actividades relacionadas con *smartphones* que no incluyen llamadas telefónicas (aplicaciones, juegos) según datos en (Nielsen Entertainment Research Group, 2011). El crecimiento de la industria de los juegos en estas plataformas está previsto que alcance los $54 billones para 2015 y se estima que los desarrolladores han obtenido entorno a $87 millones en 2011. Dentro de las plataformas móviles el 70%-80% de las descargas son juegos e incluso se puede destacar como referencia el juego *Angry Birds*, que ha sido descargado más de 150 millones de veces según datos en (Geekaphone, 2011) .

Sin embargo, este crecimiento de la industria de los videojuegos en las plataformas móviles no ha tenido en cuenta en muchos casos a las personas con discapacidad, que encuentran numerosas barreras que les impiden disfrutar de los videojuegos (Warschauer, 2010).

La dificultad del acceso a los videojuegos por parte de las personas con discapacidad es un problema grave cuando se usan dentro del ámbito del ocio, pero lo es aún más cuando se proponen los videojuegos como herramientas educativas, pues la educación es un derecho fundamental de todas las personas.

El interés en utilizar videojuegos dentro de la comunidad de usuarios con discapacidad es suficientemente amplio como para justificar la necesidad de mejorar su accesibilidad. Una prueba de esto son las diversas comunidades de jugadores con discapacidad que podemos encontrar *online*, que de manera activa discuten sobre soluciones de accesibilidad que les permitan jugar a los últimos videojuegos[2]. Además la base de personas a las que puede llegar este mercado es amplia. En (US CensusBureau, 2002) se muestra que para los Estados Unidos, que es en definitiva el mercado más grande de videojuegos en el mundo y es para el que se pueden encontrar datos más concretos en términos de accesibilidad, el número estimado de personas que pueden jugar videojuegos pero se ven afectadas por una discapacidad es de 32.213.000 (11% de la población de los Estados Unidos). Dentro de este grupo se distingue 6.267.000 (2%) de individuos que no pueden jugar a ningún juego y 25.946.000 (9%) que pueden jugar, pero con una experiencia inferior a la del resto de la población.

Cómo mejorar la accesibilidad de los juegos es un aspecto abordado desde hace algunos años especialmente desde el plano académico. En (Bierre, 2006) se plantean 3 razones por las que los juegos deberían ser accesibles y que reflejan parte del problema que se trata en este documento:

La primera es **moral**. Una persona que tiene una discapacidad, debería tener acceso a los mismos servicios y recursos que el resto de la población en todos los ámbitos de la vida. Estas medidas mejorarían la calidad de vida de los discapacitados.

[2] http://www.ablegamers.com/
http://audiogames.net/

La segunda es **económica**. No procurando accesibilidad a los juegos la industria está perdiendo una cantidad potencial de clientes para sus juegos, teniendo en cuenta las estadísticas mencionadas anteriormente.

La tercera es **legal**. En algunos países obligan por ley a que todos tengan acceso por igual a los servicios proporcionados por la administración al ciudadano. En particular, en los Estados Unidos, la sección 508 (US Government, 1998), exige que los servicios que proporciona la administración al ciudadano deban cumplir con unos requisitos mínimos de accesibilidad. La *American with Disabilities Act* (ADA) promueve accesos iguales para todos en muchas áreas de la sociedad.

1.2. Causas de la baja accesibilidad en juegos

El bajo nivel de accesibilidad en los videojuegos puede atribuirse a varias razones, entre ellas a la dificultad que supone desarrollar juegos que puedan ser utilizados por personas con y sin discapacidad, tanto desde el punto de vista de diseño como de costes de implementación.

Para la mayoría de la industria de los videojuegos, la accesibilidad en juegos tiene una prioridad baja. Una razón es **la falta de concienciación entre los desarrolladores** acerca de los problemas de accesibilidad que plantean algunas estrategias de diseño, dado que al no tener discapacidades no son conscientes de ellos, además de no conocer técnicas que puedan hacer un título más accesible a las personas con discapacidad.

Otra razón es que **los desarrolladores tienen una cantidad de tiempo y recursos limitada**. Los análisis de la relación entre costes y beneficios obtenidos, a menudo concluyen con que los problemas de accesibilidad no merecen la pena ser tratados y que se realicen inversiones por parte de la industria de los videojuegos para su resolución, debido a dos supuestos que asume esta misma y que se proponen en ("Making Video Games Accessible: Business Justifications and Design Considerations," 2005):

- El coste de implementar opciones de accesibilidad no devolverá la cantidad invertida en su incorporación.

- No hay una audiencia lo suficientemente grande para hacer los desarrollos accesibles viables.

Estas suposiciones son bastante discutibles. *Microsoft Corporation commissioned Forrester Research, Inc.* realizó un estudio para medir el potencial de mercado de la tecnología accesible en los Estados Unidos y comprender cómo se utiliza la tecnología accesible hoy en día. El estudio determinó que el 57% de los usuarios con ordenador probablemente se beneficien del uso de la tecnología accesible. De hecho hay quien aboga porque la accesibilidad es un problema de todos. El estudio también determinó que el uso de las características de accesibilidad no estaba limitado a personas con discapacidad. Entre los usuarios que utilizan utilidades y opciones de accesibilidad:

- 32% no tienen discapacidad
- 68% tienen una discapacidad leve o severa

Las características de accesibilidad son a menudo utilizadas por gente sin discapacidad sólo para mejorar su experiencia de juego. Por ejemplo, un usuario con un brazo roto.

Considerado el potencial incremento del uso de la tecnología accesible es importante instruir a diseñadores, desarrolladores y *testers*. Por ello, parte del problema de la baja accesibilidad en juegos puede resolverse si se incluyen herramientas que faciliten la accesibilidad en los programas que utilizan los profesionales. Con esto se consiguen dos cosas: por un lado se aumenta la visibilidad de la accesibilidad y por otro facilita el trabajo, reduciendo costes.

Además, parece necesario explorar formas de mejorar el diseño de los juegos para que sean más accesibles, así como soluciones que reduzcan sus costes, especialmente en aquellas plataformas a las que el jugador con discapacidad visual está menos habituado, como son los *smartphones*.

1.3. Objetivos generales

Por todo lo tratado hasta ahora, el presente proyecto tiene como objetivos las siguientes líneas de actuación:

- **Identificación de barreras de accesibilidad en plataformas móviles**. Para ello se tratará el problema de la accesibilidad en juegos dentro de una plataforma poco natural para los discapacitados, como es *Android*, explorando soluciones de diseño que mejoren la accesibilidad de los juegos en *smartphones* (o *tablets*).

- **Propuesta de soluciones de diseño** que mejoren la accesibilidad de los juegos.

- Desarrollo de un **conjunto de herramientas** que faciliten la programación de juegos accesibles que incluirán las soluciones de diseño propuestas (ver sección 4).

De esta manera se lograrían dos cosas: por un lado aumentar la visibilidad de la accesibilidad entre los desarrolladores con un *framework* que facilite la incorporación de opciones de accesibilidad a los juegos y por otro reducir costes a la hora de implementarlas.

1.4. Limitación de alcance

La accesibilidad en software y juegos es un asunto muy complejo. Se está tratando de conseguir que un sector de la población muy heterogéneo cada uno con sus problemas, sus discapacidades y con diferentes grados dentro de cada una de ellas trate de utilizar y disfrutar de un recurso tan popular como son los videojuegos.

Definición del público objetivo: usuarios ciegos

Cada tipo de discapacidad plantea una serie de problemas en la interacción con los juegos completamente diferentes. A la hora de resolver una tarea en un juego, una persona con discapacidad visual puede tener dificultades para conocer la tarea que tiene que realizar aunque que sea mentalmente y físicamente capaz de comprenderla y llevarla a cabo, ya que puede que toda la información transmitida para conocer el estado del juego se base estímulos visuales. Mientras que una persona con problemas cognitivos puede tener problemas para comprender la tarea en sí misma. Por ello, por limitaciones de alcance, se ha optado por el perfil más conocido, el de la discapacidad visual, más concretamente en las personas con una pérdida completa de la visión.

Plataforma objetivo: Android

Por otro lado está la plataforma. Los *smartphones* y *tablets* entre sus múltiples usos está el de ser una plataforma para juegos. En algunos casos podría decirse que son la principal plataforma para juegos casuales (Juul, 2009; Kim & Ricaurte, 2011).Sin embargo, la accesibilidad en juegos móviles es un asunto poco explorado, por lo que cualquier desarrollo supone una contribución significativa.

Un desarrollo multiplataforma sería muy complejo para que el trabajo sea abordable dentro del tiempo que se dispone. De entre las diferentes plataformas móviles se ha optado por aquellos dispositivos con sistema operativo *Android,* por dos razones:

- **Ser una plataforma abierta** que muestra una tendencia al alza en el mercado en estos momentos (Gartner Inc., 2012)

- **Apenas existen juegos accesibles**, por lo que cualquier desarrollo supone una contribución significativa, en comparación con otras plataformas como *iOS*

1.5. Terminología

Para comprender el contenido y las acciones a las que se hace referencia en algunas de las secciones presentes a lo largo de este documento, es necesario introducir los siguientes conceptos relacionados con la interacción en dispositivos móviles. Para más información sobre los términos utilizados en este documento, lea el glosario que se adjunta como Anexo I.

Tap, Click: evento táctil que consiste en la realización de una pulsación sobre la pantalla del dispositivo móvil.

Scroll: es el evento de desplazamiento producido en pantalla mediante uno de los dedos normalmente en sentido vertical.

Android: es un sistema operativo basado en *Linux*, por lo que dispone de un núcleo de sistema operativo libre, gratuito y multiplataforma, diseñado para ser utilizado en plataformas móviles como *smartphones, tablets, netbooks* y otros dispositivos.

iOS : sistema operativo móvil de *Apple* para dispositivos *Iphone, Ipod* y *IPad*, que rivaliza con *Android* por el dominio del mercado de los *smartphones*.

ICS (*Ice Cream Sandwich)*: nombre por el que es conocida la versión 4.0.x del sistema operativo *Android*, liberada en octubre de 2011. Es la primera versión que integra opciones de accesibilidad significativas.

1.6. Sobre este documento

El presente documento consta en primer lugar de una descripción del proyecto de forma global, las causas de la escasa accesibilidad en juegos, los objetivos e introducción a los conceptos básicos.

Posteriormente, se introducen las necesidades de interacción de los discapacitados en los dispositivos móviles, centrándose especialmente en las personas que tienen discapacidad visual, con el fin de contextualizar mejor los problemas en los que se centra el proyecto y para situar al lector, en un punto a partir del cual poder aportar soluciones interesantes. Además, se introduce el estado de la accesibilidad en juegos, así como también se detallan diferentes características de accesibilidad sobre los principales sistemas operativos en el campo de las plataformas móviles como son *Android* y *iOS*.

Después, se explica en detalle la aportación realizada. Se analizan los diferentes juegos desarrollados y su contribución al conjunto de herramientas para el desarrollo de juegos accesibles así como una descripción a alto y bajo nivel del mismo.

A continuación, se aborda el grueso de la parte técnica donde se describe el proceso de análisis, diseño e implementación del sistema que satisface los objetivos establecidos. A ésta le sigue una sección sobre las diferentes tecnologías empleadas durante el desarrollo, comparativas entre ellas y decisiones tomadas sobre las mismas.
Por último, se proporciona una sección de evaluación de las aplicaciones tanto por parte del público como de expertos en accesibilidad, unas breves conclusiones, así como una reflexión del trabajo futuro, las principales aportaciones, las referencias y la bibliografía.

2. Estudio del dominio

Tras introducir la motivación y las posibles causas de la baja accesibilidad en juegos y los objetivos del proyecto, en esta sección se realiza un estudio de aquellas áreas que son relevantes para el problema abordado. En primer lugar se analiza el problema de la accesibilidad en juegos para discapacitados visuales. A continuación, se discuten algunas estrategias de diseño para juegos accesibles. Finalmente, se realiza un estudio sobre cuál es el estado de los juegos actuales en temas de accesibilidad tanto en sistemas de sobremesa, como *PC* o *Wii*, como en plataformas móviles como *Android*.

2.1. Necesidades de interacción de las personas ciegas

Centrándonos en la discapacidad visual, para hacer frente al problema tratado en este proyecto, se va a utilizar el modelo de interacción para juegos que se plantea en (Yuan, Folmer, & Harris, 2010), para analizar las barreras a las que se enfrenta un ciego al jugar a un juego. Este modelo plantea tres pasos:

1. **Estímulo recibido**: necesario para que el usuario pueda jugar al juego de tres maneras distintas: visual, auditiva y háptica.
    a. Primario:
        En la mayor parte de los casos el estímulo recibido es visual. Aunque los dos restantes, el háptico y el auditivo, están presentes no suelen proporcionar información suficiente para jugar.
    b. Secundario:
        Es el complementario del estímulo visual. Normalmente de tipo auditivo y háptico.
2. **Determinar la respuesta**: a partir de los estímulos recibidos el usuario debe decidir cómo actuar dentro del juego. Esta respuesta típicamente depende del género y del juego.
3. **Ejecutar la respuesta** a través de algún mecanismo de entrada: tras decidir la respuesta, debe ejecutar la acción y que ésta se lleve a cabo dentro del juego. Típicamente se realiza a través de teclado, ratón o algún tipo de *joystick*.

Figura 1. Diagrama Interacción Jugador - Juego

Tras la sucesión de estos tres pasos el estado del juego cambia y éste envía un nuevo estímulo.

A continuación se ejemplifica este ciclo con una serie de juegos de distintos géneros y la manera concreta de interactuar para cada uno de ellos.

FPS (Quake)	**Puzle (Tetris)**	**Carreras (NFS)**	**Jugador**
Se visualiza un enemigo en pantalla. Se escuchan armas de fuego y explosiones.	Se ve un bloque cayendo. Puede escucharse música.	Se visualiza el camino a seguir y el coche de un oponente. Se escucha el sonido de los motores. El mando proporciona *feedback* háptico.	1-Recibe el estímulo
El jugador decide disparar su arma.	El jugador decide cambiar la orientación y posición del bloque y posteriormente dejarlo caer.	El jugador decide adelantar al oponente, girando a la derecha y acelerando.	2-Determina Respuesta
El jugador pulsa el botón correspondiente para disparar.	El jugador pulsa la tecla de rotación y la de cambio de orientación. Finalmente pulsa la tecla necesaria para dejar caer el bloque.	El jugador utiliza los controles asociados a la acción correspondiente.	3 - Ejecuta la respuesta

El enemigo muere. El estado del juego cambia y aparece un nuevo enemigo.	Una línea desaparece y un nuevo bloque cae.	El jugador adelanta al oponente y observa un camino libre a recorrer.	Repite los pasos 1,2 y 3

Tabla 1. Interacción con juegos de diferentes géneros

Normalmente el problema de la accesibilidad en los juegos para los ciegos, se centra fundamentalmente en el primer paso, el usuario no puede recibir el estímulo primario. Sin esta información es imposible que conozca lo que pasa en el mundo del juego y no puede llevar a cabo los dos pasos siguientes, aunque sea cognitiva y físicamente capaz de hacerlo.

Este problema se puede solucionar mediante el uso de lectores de pantalla o estímulos sonoros si la mecánica del juego lo permite, como es el caso de juegos tipo puzle como *Sudoku* o *Buscaminas.* Pero estas soluciones no son aplicables a todos los juegos, ya que muchos de ellos requieren de una interacción más dinámica y resulta muy complicado lograr una interacción fluida con el usuario discapacitado únicamente con estos recursos. Por ello, se pueden añadir opciones de accesibilidad que permitan regular la velocidad del juego, puntos de referencia representados por sonidos (debido a la dificultades que pueda tener un usuario ciego para percibir posiciones absolutas en el mundo del juego) u otras alternativas que dependen del género al que pertenezca el juego.

En el caso de las plataformas móviles se plantea un problema adicional. En los últimos años, las pantallas táctiles son mucho más habituales en su uso doméstico en *smartphones*, *PDAs* y ordenadores. En particular, los *smartphones* ofrecen una serie de características y aplicaciones a un precio razonable que resultan muy útiles para personas ciegas. Sin embargo, las primeras pantallas táctiles de este tipo de dispositivos, son muy difíciles de usar para los usuarios ciegos, pues no integran ningún tipo de mecanismo basado en estímulos hápticos, como el teclado o líneas braille (Torcolini, 2010). En consecuencia, el usuario también tiene problemas para responder ante un determinado evento ocurrido en el juego, que corresponde al tercer paso del modelo de la Figura 1.

2.2. Diseño de juegos accesibles

En el campo del desarrollo de videojuegos, ha habido varios intentos por definir un conjunto de directrices para el diseño de juegos accesibles. La *International Game Developers Association* (IGDA) organización relevante en el mundo del desarrollo de videojuegos que reúne a más de 10,000 desarrolladores (IGDA, 2012) a través de su grupo *Special Interest Group* (SIG) en accesibilidad en juegos (IGDA-SIG, 2012) publicó (IGDA, 2004). Se trata de un documento en el que se proponen 19 directrices de accesibilidad derivadas de una muestra de 20 juegos accesibles que ya existían en el mercado. La *Norwegian Medialt organization* publicó un total de 34 directrices (Tollefsen, 2006) en su web, basadas en 19 de IGDA y algunas propias. A continuación se muestran algunas de estas directrices obtenidas de (IGDA, 2004) (la tabla completa se muestra en el Anexo II):

Directriz	Descripción
GPS	En un juego accesible para ciegos, un sistema de posicionamiento global puede ser utilizado para obtener las posiciones concretas de los objetos en la zona además de la posición del avatar.
Modo alto contraste	Capacidad para modificar el contraste y otras características como la iluminación para ayudar a personas con baja visión a ver escenas más claramente.
Mejorar soporte *hardware* para dispositivos especiales	Ratón, *joystick* o *game-pad* son soportados habitualmente en los juegos, sin embargo, algunos usuarios con problemas de movilidad usan otro tipo de dispositivos especiales. Expandir el soporte de la aplicación para este tipo de dispositivos permitiría a muchos de ellos jugarlos.
Control más preciso de la dificultad	Permitir la modificación de la dificultad del juego más allá de lo habitual en juegos. Por ejemplo, para juegos en tiempo real, añadir un mecanismo que controle la velocidad o un botón que permita habilitar un sistema basado en turnos.
Gestión de síntesis de voz propia	La capacidad de proveer la lectura de los textos que son mostrados en el juego beneficiaría a personas ciegas y con problemas de baja visión. Hay ya una gran cantidad de *software* que proporciona esta funcionalidad y que puede integrarse en juegos.
Mejores tutoriales	Esta característica sería útil para casi todos los jugadores. A mucha gente le gusta comenzar a jugar sin leer un manual. Guiarles a través del juego y proporcionarles *feedback* extra sería útil ya que ellos podrían comprender muchos de los aspectos de los juegos fácilmente. Especialmente en el caso de personas con problemas de aprendizaje, con problemas de concentración para la lectura de un enorme manual.
Subtítulos	Cualquier tipo de diálogo que se reproduce en el juego debería mostrarse en forma de texto en la pantalla. Esto ayudaría a jugadores que son sordos o presentan dificultades en la audición. Esta opción puede ser configurable como una opción del juego.

Tabla 2. Algunas directrices para el diseño de juegos accesibles[3]

El principal problema de estas directrices es que estaban basadas en un conjunto de géneros de juegos muy limitado. Además, tuvieron una escasa aceptación en la industria. Por otro lado en 2004, difícilmente pudieron cubrir los aspectos de accesibilidad relacionados con las plataformas móviles.

[3] Tabla completa en Anexo II

Existe una cierta demanda de los colectivos de discapacitados, por ejemplo, comunidades de jugadores como AbleGamers[4] y AudioGames[5] ya que cuando un jugador es atacado en un juego por la espalda y no puede oír los pasos del atacante, es probable que no lo encuentre muy entretenido, puesto que no se le ha dado al usuario ninguna oportunidad de huir. En la tabla siguiente obtenida de (IGDA, 2004), se presentan los problemas más comunes que pueden tener los jugadores discapacitados en los juegos actuales:

Problema	**Razón**
Incapacidad para seguir la trama de la historia del juego.	• No dispone de texto subtitulado y la historia se cuenta en escenas de transición (Audición) • Historia muy compleja y difícil de seguir. (Problema cognitivo)
Incapaz de completar un puzle o tarea.	• Pistas vitales se dan en escenas de transición sin subtitular (Audición) • Todas las pistas se dan como texto (Visual) • Requiere tiempos muy precisos con los controles (Movilidad) • Requiere de la habilidad para colocar un cursor con precisión (Movilidad)
Incapaz de seguir la mecánica para jugar al juego.	• Ausencia de un tutorial • Pobre documentación • Documentación escrita de forma demasiado compleja para la audiencia
No dispone de la habilidad para usar hardware adaptativo.	• El juego sólo soporta un conjunto limitado de dispositivos
El avatar del jugador resulta herido o muere repetidamente en el juego.	• No hay identificación de pistas basadas en audio (Audición) • No hay indicadores de situación de peligro • No dispone de la habilidad para responder rápidamente con los controles (Movilidad) • No es posible cambiar la velocidad del juego. (Movilidad)

Tabla 3. Problemas comunes de los jugadores discapacitados

[4] http://www.ablegamers.com/

[5] http://audiogames.net/

2.3. Juegos accesibles para usuarios ciegos

A lo largo de esta sección se han analizado juegos específicamente diseñados para ciegos, otros cuya finalidad es concienciar al usuario de la situación de estas personas en el campo de los videojuegos y otros que combinan ambas opciones.

En general, los juegos desarrollados para plataformas específicas de juegos, como *Wii* o *Xbox*, o incluso PC, no dejan de ser casos aislados o bien resultados de proyectos de investigación, pero prácticamente no se pueden encontrar entre los juegos más populares o *mainstream,* alguno que considere incorporar opciones de accesibilidad. Es posible que esta situación se deba en parte a una falta de concienciación por parte de la industria, que considera que los costes adicionales de hacer un juego más accesible no compensen en términos de ventas dicha inversión.

Uno de los géneros más populares dentro de la comunidad de jugadores ciegos, son los **juegos basados en texto**, como las clásicas aventuras conversacionales que se llevan realizando desde principios de los 80, ya que pueden ser fácilmente transcritos y no requieren de una interacción fluida entre el jugador y el juego.

Por otro lado están los **"audiojuegos"**, juegos con un cierto carácter experimental, basados íntegramente en sonido. Algunos de estos juegos incluso disponen de hardware que incorpora GPS y que permite poder jugar con unos auriculares y acceso a una red Wi-Fi, según se está caminando por la calle. También existen herramientas para poder desarrollar este tipo de juegos como *Audio Game Maker*[6], que permite a discapacitados visuales hacer sus propios "audiojuegos". Este género dispone de su propia comunidad de usuarios[7].

Figura 2. The Explorer and the Mystery of the Diamond Scarab - Wii

[6] http://www.audiogamemaker.com/

[7] http://audiogames.net/

Por otro lado, también hay algunos ejemplos en plataformas específicas de juegos, como Nintendo *Wii*, donde se ha publicado ***The Explorer and the Mystery of the Diamond Scarab***[8]. Éste es uno de los primeros juegos que permite jugar por igual a niños con discapacidad visual y aquellos que no la padecen. El juego posee locuciones y señales auditivas necesarias para poder jugar sin ver y forma parte de un intento por entrar en un mercado todavía poco explorado, en el que cualquier juego que se publique tendrá toda la atención por parte de su público objetivo.

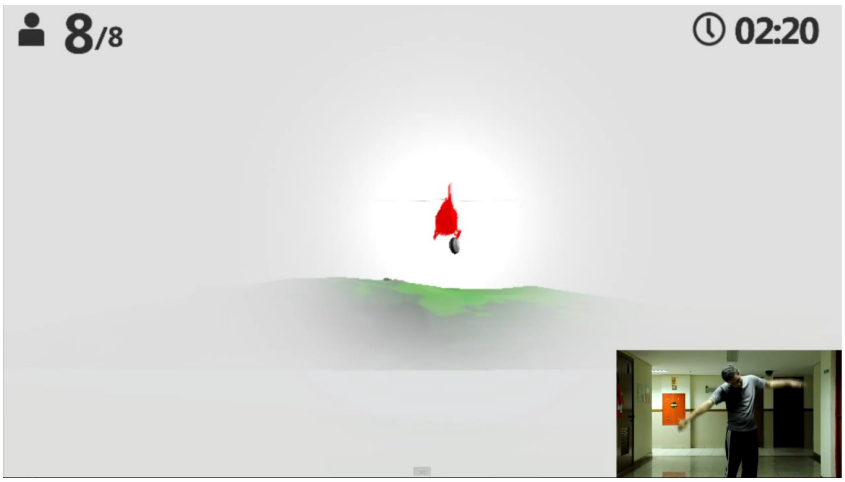


Figura 3. Herocopter - PC

También se pueden encontrar juegos como ***Herocopter***[9], desarrollado por el grupo BSB IT, en el que se controla un helicóptero de rescate utilizando *Kinect*. Este juego está desarrollado para las plataformas *XBOX 360* y *PC*, en las cuales se puede controlar un helicóptero con movimientos de nuestro cuerpo gracias a *Kinect*. *Herocopter* se apoya en distintos estímulos sonoros para indicar la distancia a la que se encuentra el usuario de cada elemento del juego y un narrador que indica la altura a la que se encuentra el helicóptero del objeto cuyo sonido se reproduce en ese momento. Además, en el juego existe una niebla para personas sin discapacidad visual que simula las sensaciones de una persona ciega al jugar.

Por otro lado**, existen iniciativas interesantes como *Creatureddiscomforts.org***, realizada por la fundación *Leonard Cheshire Disability*[10]. Es una campaña de concienciación sobre las barreras a las que se enfrentan las personas con discapacidad y con qué actitud las afrontan.

Se dispone de cuatro juegos, protagonizados cada uno de ellos por un animal distinto. Cada protagonista está basado en una persona real con discapacidad, que además dobla al personaje y del que posteriormente, se muestra un vídeo contando su propia experiencia.

[8] http://www.gambas-games.nl/

[9] http://www.youtube.com/watch?v=1Y66A0DwYcA

[10] http://www.creatureddiscomforts.org/

Figura 4. Flyzzz game - PC

Uno de ellos es ***Flyzzz game*** (con Callum the chameleon como protagonista - un camaleón ciego). Es un juego *flash* que consiste en atrapar moscas con la lengua mediante *clicks.* Para localizar a las moscas se deben distinguir los distintos sonidos emitidos por el aleteo producido al volar. Dispone de un modo en el que se emite únicamente el audio y un modo que incluye también interfaz visual, que es algo más complejo. El control del juego resulta ser muy sencillo, puesto que sólo se deben hacer *clicks* con el ratón, cuando se crea que la mosca esta dentro del rango de alcance de la lengua del camaleón.

2.4. Juegos accesibles para usuarios ciegos publicados en plataformas móviles

Aunque en menor número e impacto, también hay juegos accesibles desarrollados específicamente para dispositivos móviles, entre los que se puede encontrar ***Accessible Minesweeper*** en *iOS*, basado en *VoiceOver*, que proporciona un cambio en la forma de interacción cuando se inicia la aplicación, para facilitar su uso a personas con discapacidad visual.

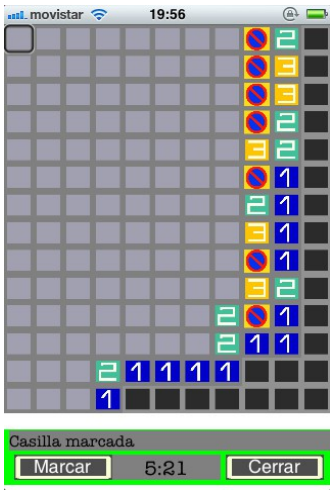


Figura 5. Accessible Minesweeper - iOS

También **se pueden encontrar juegos basados íntegramente en texto**, que aunque no estuvieran pensados inicialmente para ser accesibles, son bastante utilizados por la comunidad de jugadores ciegos como ***Fantasy Hero***[11], o las aventuras textuales ***Frotz***[12] y ***Zork***[13].

Fantasy Hero *es* un *RPG* en el que los jugadores disponen de un personaje al que tienen que entrenar enfrentándose a otro jugadores dispone de tienda *online* en la que se realizan compras de objetos para mejorar al personaje dentro del juego. La interacción resulta ser muy simple para las personas con discapacidad visual, puesto que hace uso de *VoiceOver*, se basa en el clásico cambio de interacción por pulsaciones sobre las opciones de los menús: para la lectura de la acción que realiza el botón se realiza un *tap* y para su selección doble *tap*.

Figura 6. Fantasy Hero - iOS

Zork en su versión para PC, fue uno de los primeros videojuegos de ficción interactiva. La historia se desarrolla en un laberinto subterráneo, teniendo como único objetivo, regresar sano y salvo repleto de tesoros mediante la exploración de cuevas.

El único mecanismo de entrada es el teclado, ya que para poder avanzar en la aventura, se deben dar órdenes mediante texto. Es por esto que la interacción para personas con discapacidad visual en este tipo de juegos, es más natural pues únicamente necesitan hacer uso del sistema lector de pantallas para conocer lo que ocurre en el videojuego.

[11] http://www.zhurosoft.com/

[12] http://code.google.com/p/iphonefrotz/wiki/FrotzMain

[13] http://itunes.apple.com/us/app/frotz/id287653015?mt=8

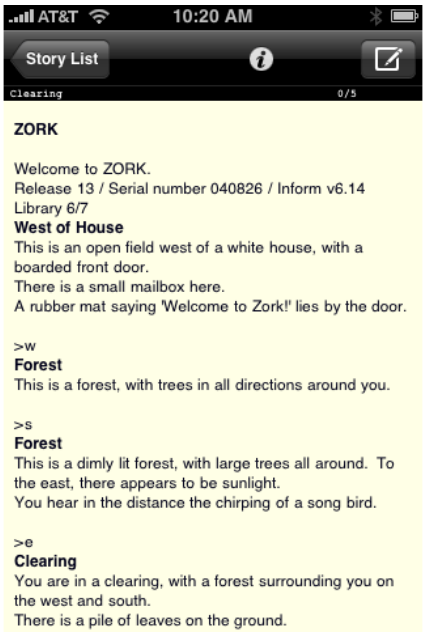


Figura 7. Aventura Interactiva Zork - iOS

También se pueden encontrar juegos como ***Papa Sangre***[14], un videojuego sin vídeo. Es un *thriller* en primera persona basado en su totalidad en audio. El protagonista está encerrado en el palacio de *Papa Sangre* y debe escapar. Aunque existe un problema, el lugar está lleno de monstruos que hay que eludir, pero está demasiado oscuro para poder verlos, por lo que la única forma de identificarlos es mediante un determinado sonido que tiene asociado cada uno de ellos.

Figura 8. Brújula y áreas para desplazar al personaje – Papa Sangre

Este juego logra crear un entorno de sonido 3D, que el jugador puede explorar en una dirección fijada mediante una brújula. Además, el jugador establece un ritmo en su avance por la escena mediante la frecuencia con la que realiza *taps* alternativamente sobre ciertas zonas en la pantalla que se pueden ver en la Figura 8. Esta frecuencia juega un papel importante a la hora de dotar de cierta emoción al juego, puesto que

[14] http://www.papasangre.com/

si fuera demasiado alta los monstruos podrían detectar al protagonista y automáticamente el juego finalizaría.

También se han llevado a cabo **iniciativas como *The Sound of Football***[15], que trató de facilitar la práctica del fútbol a personas ciegas. En primer lugar se identificaron los elementos cuya posición debía conocer un futbolista a la hora de poder jugar como son la pelota, las porterías o los rivales. A cada uno de ellos se le asoció un sonido y mediante una aplicación sobre *iOS*, en base a la distancia y posición a la que se encontraba el jugador de cada objeto, se reproducía un sonido 3D con un determinado tono e intensidad. Cada objeto representaba la fuente de ese sonido y se trataba de modular su atenuación para conocer la distancia y su tono para conocer la orientación.

Con esta idea se organizó un partido de fútbol entre futbolistas con y sin discapacidad visual. A cada uno de ellos se le proporcionó un equipo similar al de la imagen, para que de esta forma todos estuvieran en igualdad de condiciones.

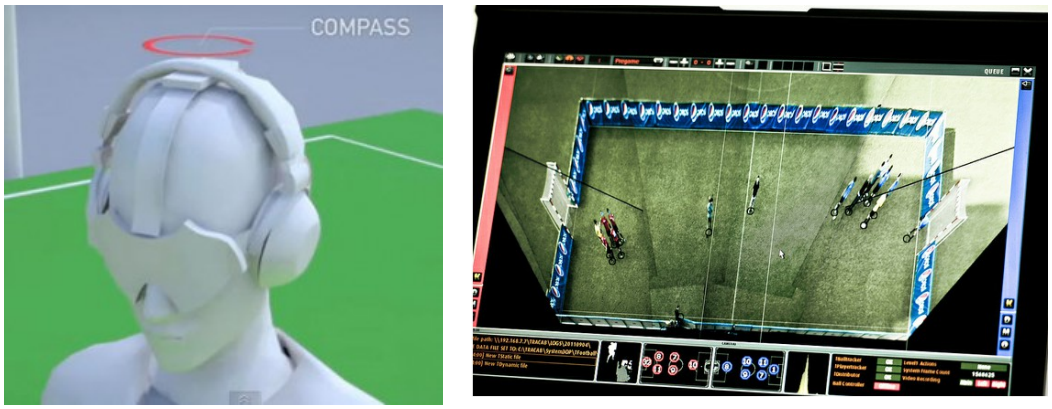


Figura 9. Sistema Receptor basado en un iPhone (izquierda) y sistema de seguimiento (derecha)

Se montó un sistema de seguimiento con un conjunto de cámaras y sensores que permitía conocer la posición de cada objeto. Esta información era transmitida en tiempo real al dispositivo *iPhone* que tenía cada jugador y se utilizaba el mecanismo de sonidos para guiar a los jugadores en el juego.

Por otro lado, en *Android*, dentro de su mercado de aplicaciones *Google Play* aunque el catálogo de títulos que puedan ser utilizados con facilidad con discapacidad visual es más reducido que en *iOS*, se pueden encontrar títulos como los expuestos a continuación.

Se ha adaptado un juego para PC, ***Attractor HD***[16], para *iPad* y *Android*. El objetivo es sencillo, llevar unas pequeñas bolas blancas a unos agujeros negros, pero no se pueden mover directamente las bolas, se tienen que activar unos "atractores" y "repulsores" que harán que las bolas se muevan.

[15] http://thesoundoffootball.com/

[16] http://play.google.com/store/apps/details?id=air.com.thegamekitchen.attractorhd

Figura 10. Attractor HD – Android/iOS

El juego está pensado para que se juegue al ritmo que se desee, no existe un límite de tiempo para completar cada nivel con lo que se puede terminar sin ningún tipo de presión. Además, si aún así se tiene algún problema, se puede cambiar la velocidad del juego en las opciones y ajustarlo a nuestras capacidades.

Desde el punto de vista de la discapacidad visual la característica más importante es el modo de juego en alto contraste aunque el impacto de esta opción para el público ciego es bajo.

Figura 11. Stemp Jumper+ - iOs

Otro título conocido es ***Stemp Jumper+***[17] se trata de un juego de tipo puzle con más de 100 niveles que consiste en guiar a una planta a través de un laberinto de casillas, cada una de las cuales tiene un sonido, esquivando obstáculos. Para ello se utilizan una serie de herramientas, desde un tirachinas hasta maíz. También se puede encontrar una versión en *iOs* de esta aplicación.

[17] http://www.ananseproductions.com/

Como se ha podido ver el en el sistema operativo utilizado en los dispositivos móviles de *Apple, iOS*, la efectividad del servicio *VoiceOver* ha tenido mucha aceptación entre los usuarios ciegos, lo que ha propiciado un cierto interés en desarrollar aplicaciones accesibles para ciegos en esta plataforma. No obstante, el número de juegos accesibles disponibles en el mercado sigue siendo bajo, tanto los juegos específicamente diseñados para ciegos como juegos accesibles en general, escaseando especialmente en la plataforma Android, ya que todavía constituye un entorno poco natural para personas ciegas.

2.5. Características de accesibilidad para ciegos en plataformas móviles

En esta sección se analiza el estado actual de la plataforma *Android* en términos de opciones de accesibilidad, antes y después de la última versión del sistema liberada *ICS*. Además, se realiza una comparativa con su principal competencia en el mercado de los *smartphones* como es *iOS*.

2.5.1. Comparativa de accesibilidad visual entre Android antes de ICS y iOS

Hasta la aparición de *Android* 4.0 (también conocido como *Ice Cream Sandwich*), el nivel de accesibilidad del sistema operativo *iOS* superaba al de *Android*. Esto ha convertido a *iPhone* en la plataforma de *smartphones* de referencia para usuarios ciegos.

Para hacer uso de todas las características de accesibilidad en ***iOS***, es necesario activar un servicio del sistema operativo denominado ***VoiceOver***. Se trata una solución integral de accesibilidad que va más allá de lo que es un lector de pantallas. Además de realizar las funciones de este último, cambia la forma de interactuar con el dispositivo:

- **Interacción con la interfaz**: cambia totalmente los gestos a realizar. Un solo *tap* (*click* o pulsación) sólo cambia el foco de sitio, mientras un doble *tap* se utiliza para activar un botón o ejecutar una aplicación. Por último, para hacer el *scroll*, se realiza el mismo gesto que sin activar opciones de accesibilidad - arrastrar - pero con tres dedos sobre la pantalla.
- **Lector de pantallas**: el dispositivo reproduce dónde se encuentra el foco de selección y qué características tiene el elemento del interfaz seleccionado.
- **Configuración**: existe una gran variedad de opciones para configurar el dispositivo si el servicio *VoiceOver* está activo: idioma, velocidad de la voz, tono, volumen, entonación, así como lectura carácter a carácter según se va escribiendo o que avise cuando se produce un cambio de foco.

En ***Android***, no existía un mecanismo tan sofisticado para facilitar el uso de los dispositivos móviles a personas ciegas. Se encontraron algunos proyectos que trataban el tema de la accesibilidad en dispositivos móviles, destacando entre ellos *Eyes Free Project*, hasta *Android* 4.0, con sus utilidades, ***TalkBack*,**

***SoundBack* y *KickBack*.** La primera es un lector de pantallas simple, construido sobre el sintetizador de voz configurado en el dispositivo móvil *Android,* que informa al usuario de su ubicación dentro de la interfaz gráfica y de los eventos que se producen de su interacción con la pantalla (reproduce el contenido de las etiquetas asociadas a las vistas de un interfaz - *contentDescription*). *SoundBack*, se basa en sonidos para informar de los eventos y *KickBack* es análogo al anterior pero basado en estímulos hápticos.

En los últimos meses se han publicado nuevas soluciones software como ***Mobile Accessibility***[18], que tratan de solucionar el problema de acceso a pantallas táctiles para personas ciegas o con baja visión. Para ello, proporciona un conjunto de 10 aplicaciones accesibles que permiten utilizar la funcionalidad básica de un *smartphone* (Llamadas, SMS, Alarma, Calendario, Email, Navegador, Geoposicionamiento, Configuración) que han sido específicamente diseñadas para personas ciegas. Dispone de una interfaz basada en gestos similares a los descritos anteriormente, eventos de arrastre para seleccionar los elementos y leerlos mediante síntesis de voz y doble *tap* para seleccionarlos. Además, en lugar de la síntesis de voz dispone de una salida para la conexión de un visor *braille* que muestre la información en lugar de reproducirla de forma sonora.

También realiza la función de lector de pantallas, del mismo modo que *Talkback,* por la interfaz estándar del sistema aunque para ello es necesario que el dispositivo *Android* disponga de *trackball*, *trackpad* o teclado portable, éste último poco práctico si se habla de teléfonos móviles.

No obstante, no deja de ser una aplicación bastante costosa (75€) y que no proporciona una gran mejora en accesibilidad a toda la plataforma, como sí se espera que lo hagan futuras características accesibles a partir de *Android* 4.0.

Parece que el uso de dispositivos que emplean el sistema *Android* hasta el momento para usuarios ciegos, es poco recomendable dado que disponen de menos opciones de accesibilidad que su competencia y las que hay, o bien son bastante mejorables, o conllevan gastos extra que sólo permiten utilizar la funcionalidad de un teléfono móvil común.

2.5.2. Android a partir de ICS

En la última versión del sistema liberada por *Google Android* 4.0, conocida como ICS (*Ice Cream Sandwich*), han mejorado considerablemente las opciones de accesibilidad, facilitando la forma de usar la interfaz sólo con los dedos y oídos. De esta manera la plataforma está haciendo un esfuerzo por ponerse a la altura de su competidor *iOS*. Cabe destacar, que en la actualidad esta versión de *Android* se está introduciendo paulatinamente en dispositivos móviles de ciertos fabricantes. En la tabla siguiente se muestra una comparación en términos de opciones de accesibilidad para discapacitados visuales entre ambos sistemas.

[18] http://www.codefactory.es/en/

Característica	ICS	iOS
Funcionalidad Zoom. Amplifica la pantalla.	Sí	Sí
Modo inversión de color. Invierte los colores de la pantalla (sólo para baja visión).	No	Sí
Auto-corrección hablada. Lectura de correcciones en el texto y sugerencias mientras el usuario escribe.	No	Sí
Voice Control. Permite al usuario realizar llamadas y controlar cierta funcionalidad básica del dispositivo mediante voz.	Sí	Sí
Tipo de información en la lectura de elementos de interfaz.	• Nombre • Tipo	• Nombre • Tipo • Posición en la pantalla • Comportamiento en activación
Lectura de interfaces de usuario con un *tap.*	Sí	Sí
Selección de elementos de interfaz de usuario con doble tap.	Sí	Sí
Interacción basada en el número de dedos posicionas sobre la pantalla (*scroll* solamente si se realizan arrastres con 2 dedos).	No	Sí

Tabla 4. Comparativa ICS - iOS

Las nuevas opciones de accesibilidad ofrecen características como, la lectura de las opciones que se puede activar con el arrastre de nuestro dedo sobre la pantalla y el desplazamiento de ésta mediante el arrastre producido con dos dedos. También el sistema avisará de las acciones que se van realizando, además de tener la posibilidad de hacer un doble *tap* para seleccionar un elemento de la interfaz[19]. No obstante, todavía es pronto para realizar un análisis sobre el impacto real que puede tener *ICS* en la comunidad ciega, dada su reciente introducción en el mercado.

Por último, cabe destacar que todas estas medidas son de naturaleza similar a los mecanismos de interacción con las interfaces que se han implementado en cada una de las aplicaciones.

[19] http://developer.android.com/about/versions/android-4.0-highlights.html

3. Juegos desarrollados

A lo largo de este proyecto se han desarrollado cuatro juegos: *Buscaminas Accesible, Golf, Zarodnik* y por último, *The Shadow of the Past*. Se comenzó con el desarrollo de un buscaminas, dado que se carecía de experiencia de desarrollo en plataformas móviles. Además, ya existía un buscaminas accesible desarrollado por un ciego en la plataforma *iOS*, por lo que suponía una buena oportunidad para tomar contacto con el desarrollo de aplicaciones accesibles para ciegos. Posteriormente, se continuó con el desarrollo de *Golf*, que planteaba la problemática adicional de que el lector de pantallas dejaba de ser una alternativa a la hora de proporcionar información al usuario sobre lo que ocurra en el juego. A continuación se llevó a cabo *Zarodnik*, cuya característica principal es el uso de sonido 3D para poder identificar todos los elementos que se presentan en la pantalla. Se finalizó con *The Shadow of the Past*, un juego basado íntegramente en texto, que narra una historia interactiva que va proporcionando al usuario distintas opciones conversacionales.

El orden en el que se desarrollaron los juegos es significativo, dado que cada uno de ellos añade nuevas características que conllevan un incremento progresivo de la dificultad del desarrollo. Para su desarrollo se ha hecho uso de la herramienta *BFG Toolkit*, obtenida como resultado de la implementación de estos juegos. En concreto, *Buscaminas Accesible* permitió la creación de las clases encargadas de gestionar el sintetizador de voz y utilidades de gestión de sonidos, que supondrían el comienzo de esta herramienta. *Golf* sirvió como base para identificar los elementos comunes de este tipo de juegos y realizar una abstracción de una serie de módulos. Posteriormente, *Zarodnik* añadió toda la funcionalidad relacionada con la gestión de sonido 3D. Por último, *The Shadow of the Past* permitió el desarrollo de un módulo para cargar pequeñas aventuras basadas en diálogos.

En esta sección se exponen las características comunes de cada uno de los juegos desarrollados con *BFG Toolkit* (ver sección 4), analizados anteriormente, así como su descripción en detalle siguiendo una estructura claramente definida. En primer lugar, se realiza una breve introducción a cada juego en la que se resumen los motivos por los que se decidió desarrollar el mismo. A continuación se describe el diseño del juego, es decir, en qué consiste, cuáles son los objetivos, etc. Posteriormente se explica el mecanismo de interacción que los hace accesibles a usuarios ciegos. Finalmente, si procede, se incluye una sección sobre las sucesivas versiones de la aplicación en cuestión y unas conclusiones sobre lo que se aprendió de su desarrollo de cara a mejorar *BFG Toolkit*.

3.1. Características comunes

Dos modos de interacción

Estos juegos están destinados especialmente a gente con discapacidad visual, pero ello no implica que las personas sin discapacidad visual no puedan jugarlos. Por este motivo, cada uno de estos juegos dispone de dos modos de interacción con la pantalla. Por un lado, se tiene un modo estándar, en el que para acceder a cada elemento del menú basta con realizar una pulsación y por otro lado, un modo para personas con discapacidad visual, en el que para realizar las acciones anteriores se ha de realizar una pulsación larga. De este modo, con una pulsación el sintetizador indica la opción que se ha seleccionado y con una pulsación larga se accede al contenido de la misma. Además, dentro de las opciones de configuración se dispone de una característica que permite la repetición del último mensaje reproducido por el sintetizador.

Tutoriales y opciones de configuración

Otro aspecto a destacar presente en todos los juegos es que disponen de tutoriales explicativos que introducen las diferentes formas de interacción y las distintas opciones de configuración, que permiten al usuario personalizar la interacción a su gusto. Los tutoriales permiten suavizar la curva de aprendizaje.

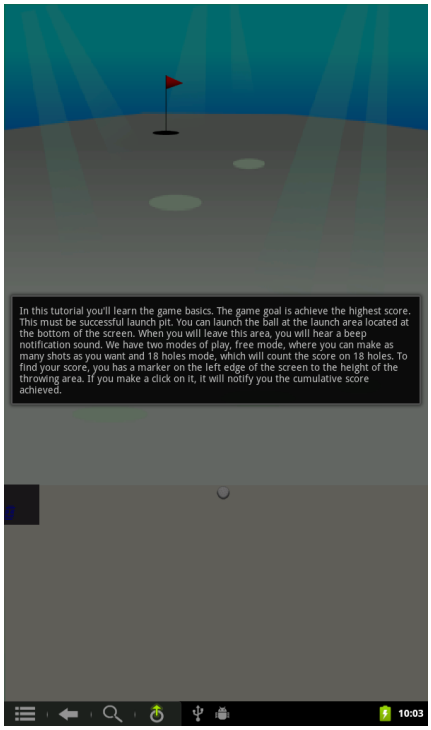


Figura 12. Tutoriales Zarodnik

Modo “Pantalla en negro”

Una vez se tenía una versión estable de todos los juegos, se decidió añadir un modo “pantalla en negro”, con el cual poder transmitir a las personas sin discapacidad visual cómo es la experiencia de juego cuando el usuario no dispone de estímulos visuales que le sirvan de guía. Cuando se activa este modo, toda la pantalla se visualiza en color negro, teniendo que guiarse mediante el sintetizador de voz y eventos sonoros y hápticos. Con esto se pretende también mejorar la concienciación sobre la necesidad de mejorar las opciones de la accesibilidad en los juegos y aprovechar las características de accesibilidad para hacer juegos más interesantes para personas sin discapacidad visual.

Modo “onomatopeyas”

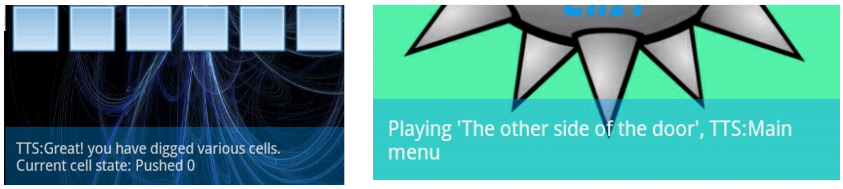


Figura 13. Sistema de Transcripción BFG Toolkit

También se ha implementado un sistema de transcripción de audio y voz de forma que todo sonido que se reproduce en el juego se visualiza en forma de subtítulos en la pantalla (ver Figura 13). Este modo llamado modo “onomatopeyas” se ha desarrollado por varias razones:

- Las aplicaciones son más accesibles a personas con problemas de audición.
- Permite jugar a personas sin discapacidad visual sin sonido. Útil de cara a presentar las aplicaciones, hacer demostraciones o jugar en entornos donde no se pueda utilizar sonido (para personas sin discapacidad visual únicamente).

Localización

Como característica adicional, se ha de destacar los esfuerzos de internacionalización en dos idiomas, inglés y español, para tratar de alcanzar una máxima difusión en cada uno los juegos.

3.2. Buscaminas

3.2.1. Introducción

Al comienzo del proyecto, no se tenía una gran experiencia sobre el desarrollo en plataformas móviles y mucho menos en la creación de una aplicación que se pudiera considerar accesible. Por ello se comenzó a buscar información y algunas referencias en el mercado que pudieran tomarse como punto de inicio para el desarrollo.

En ese momento se encontró un juego accesible. Un buscaminas que se había hecho para dispositivos *iPhone* y *iPad,* y que además lo había desarrollado un discapacitado visual. Por lo que a partir de ese juego, podían sacarse ideas acerca de las necesidades especiales que un discapacitado visual necesita a la hora de jugar a un videojuego.

Figura 14. Buscaminas Accesible - iOS

Otro de los motivos por los que se decidió comenzar con un buscaminas fue el hecho de que en este tipo de juegos la exploración del "mundo" del juego (o universo) no depende de la localización espacial. Un discapacitado visual conoce el concepto de tabla, por lo que es más fácil situar al usuario en el tablero mediante el uso de coordenadas. Esto se conoce como posicionamiento espacial relativo, que consiste en localizar los distintos elementos mediante coordenadas relativas, como si de una tabla se tratara. Mientras que el posicionamiento espacial absoluto, es dependiente del movimiento del elemento que se quiere referenciar, por lo que las coordenadas de su localización (coordenadas x e y de los *píxeles*) no proporcionan al usuario información útil acerca del estado del elemento. Además, la distribución es más similar a la de una página web, este es otro aspecto que facilita la localización del usuario en el juego, ya que este tipo de personas con discapacidad visual están habituados a navegar con lectores de pantalla.

Así pues, tomando como referencia la forma de interacción que utilizaba el juego desarrollado por el discapacitado visual, se trató de adaptar y mejorar en la medida de lo posible para la plataforma *Android*.Diseño del juego
En el *Buscaminas Accesible*, se dispone de una matriz de casillas. Cada casilla puede contener los números del 1 al 8, o bien, casilla vacía. El objetivo es descubrir todas las casillas del tablero evitando destapar alguna mina. Para ello, a lo largo de la partida, se irán destapando casillas que aportarán información acerca de las minas a través de su contenido. Los dígitos mostrados indican el número de minas que hay alrededor de la casilla destapada. En caso de que esté vacía indica que no hay ninguna mina alrededor.

Como herramienta adicional para ayudar a detectar las minas, se podrá hacer uso del modo exploración. Una vez activado éste, permitirá poner una bandera en una casilla deseada de forma segura, independientemente del número de pulsaciones que se haga en la pantalla.

Se dispone de tres dificultades: fácil, medio y difícil. En el modo de dificultad fácil se mostrará un tablero con dimensiones 6x6 y 5 minas a evitar. En el modo de dificultad medio, se encontrará un tablero de 8x8, con 10 minas y por último en cuanto al modo de dificultad difícil, se dispone de un tablero de 10x10, con 15 minas. De esta forma, se va dificultando más el poder destapar todo el tablero.

3.2.2. Diseño de la interacción

A continuación se detallarán los controles del juego. Con una pulsación sobre la pantalla se seleccionará la casilla deseada y se obtendrá información acerca de su estado. Con dos pulsaciones se destapará la casilla e informará sobre el número de minas que hay alrededor y con tres pulsaciones se pondrá una bandera sobre la casilla seleccionada, con la intención de señalar la casilla donde se crea que hay una mina. También incorpora una serie de opciones de accesibilidad que se describen en la Tabla 5.

Además, el usuario tiene la posibilidad de interactuar con un teclado. Independientemente de que éste no venga incorporado por defecto en los dispositivos móviles, si se dispone de uno compatible, se puede realizar el cambio de foco entre casillas mediante las flechas de dirección, seleccionar la casilla con una pulsación sobre el *trackball* del dispositivo si dispone de éste etc., así como asignar las distintas opciones de configuración que se ofrecen en el juego, a las diferentes teclas disponibles en el teclado.

3.2.3. Comparativa entre Buscaminas Accesible iOS y Buscaminas Accesible Android

Dada la inexperiencia en el desarrollo de videojuegos accesibles, como se comentó previamente, se tomó como base la aplicación existente para iOS desarrollada por un discapacitado visual. Así pues, a continuación se muestra a modo de explicación, una tabla comparativa entre la aplicación *Buscaminas Accesible* disponible

en *iOS* y la aplicación se ha desarrollado en este proyecto disponible para *Android*, en vista a obtener una visión general sobre las características de las que dispone.

Características	**Buscaminas Accesible Android**	**Buscaminas Accesible iOS**
Un evento de pulsación simple selecciona.	Sí	Sí
Un evento de pulsación larga activa la opción.	Sí	Sí
Un evento de arrastre con 3 dedos desplaza la vista.	No, puesto que la aplicación dispone de una vista global del tablero.	Sí
Información de las casillas que se seleccionan mientras se realiza *scroll*	Sí	Sí
El lector de pantallas informa sobre...	• La situación del foco de selección • Coordenadas y estado de la casilla • Repetir el último mensaje reproducido por el sintetizador de voz • Activación o desactivación de los distintos modos disponibles • Lectura del estado de las casillas que se encuentran alrededor de la seleccionada • Lectura de las instrucciones	•La situación del foco de selección • Qué características tiene esa vista o casilla • Coordenadas de la casilla
Modo pantalla en negro permite hacer uso de la aplicación sin ninguna referencia visual. Está destinado a personas sin discapacidad visual, con el objetivo de concienciar al usuario sobre cómo se maneja una persona con discapacidad visual sobre este tipo de juegos con pantalla táctil.	Sí	No
Modo onomatopeyas, es un sistema de transcripción de sonido y voz. Este modo está destinado para personas sin discapacidad visual y	Sí	No

que no tengan instalado ningún motor de síntesis en el dispositivo móvil.		
Modo zoom, permite centrar la pantalla en la casilla que se ha seleccionado para facilitar su pulsación. Además en este modo cambia la interacción a una pulsación para destapar y dos pulsaciones para explorar, para mayor comodidad.	Sí	No
Modo exploración, para colocar banderas independientemente del número de pulsaciones realizadas.	Sí	No
Distintos modos de interacción con el fin de adaptarse tanto a personas con discapacidad visual como para aquellas que no tengan discapacidad visual.	Sí	Sí

Tabla 5. Buscaminas Accesible iOS y Buscaminas Accesible Android

3.2.4. Proceso de desarrollo

A lo largo del desarrollo de esta aplicación, antes de obtener la versión final de ésta, se exploraron distintos modos de interacción hasta obtener aquel que resultara más cómodo tanto a personas ciegas como a personas sin discapacidad, pero dando prioridad a las primeras. Se desarrollaron las siguientes versiones: 1.0a, 1.0a1, 1.0a2, 1.0b y 1.0c que resultó ser la versión final. A continuación se describe la evolución entre cada una de las versiones obtenidas.

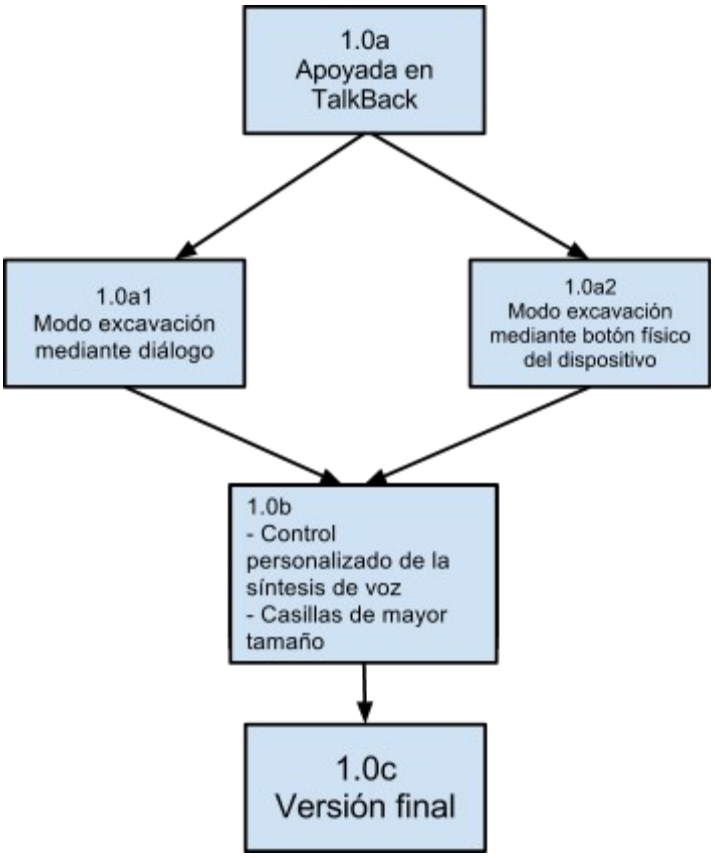


Figura 15. Diagrama de evolución de Buscaminas Accesible

En un principio el desarrollo se centró en lograr la compatibilidad con *Talkback*, que como ya se ha mencionado en la sección 2.5.1 es simplemente un lector de pantallas construido sobre el sintetizador de voz configurado en el sistema *Android,* por lo que bastaba con asociar una descripción a cada elemento de la interfaz de usuario.

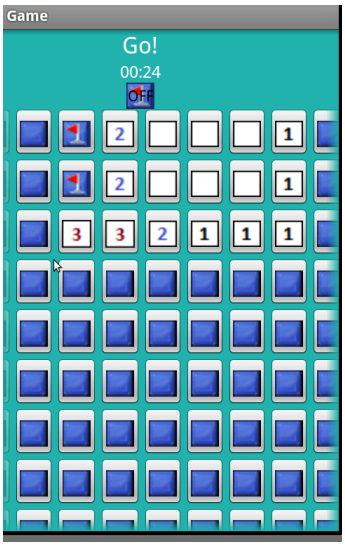


Figura 16. Primera versión de Buscaminas (1.0a)

Así pues, la primera versión fue un sencillo buscaminas con un cronómetro y un botón en la parte superior para alternar entre el modo exploración (marcar una mina con una bandera) y el modo excavación (destapar una casilla) como se puede ver en la Figura 16. *Talkback* iba leyendo las características de cada celda (las coordenadas, si estaba pulsada o no y el número de minas alrededor) con cada cambio de foco.

Se decidió seguir desarrollando la primera versión 1.0a para mejorar la forma en la que el usuario podía interactuar con el juego. Una mejora fue dar la posibilidad de activar o no la lectura de las coordenadas de cada celda por parte de *TalkBack*. Además, se implementaron un par de soluciones diferentes para el cambio entre los modos exploración y excavación que dieron lugar a las versiones 1.0a1 y 1.0a2.

En la versión 1.0a1, al hacer una pulsación sobre la celda se produce un cambio de foco, activando de esta forma *TalkBack*. Con una pulsación larga se muestra un diálogo que ofrece la posibilidad de explorar o destapar esa celda tal y como se muestra en la Figura 17.

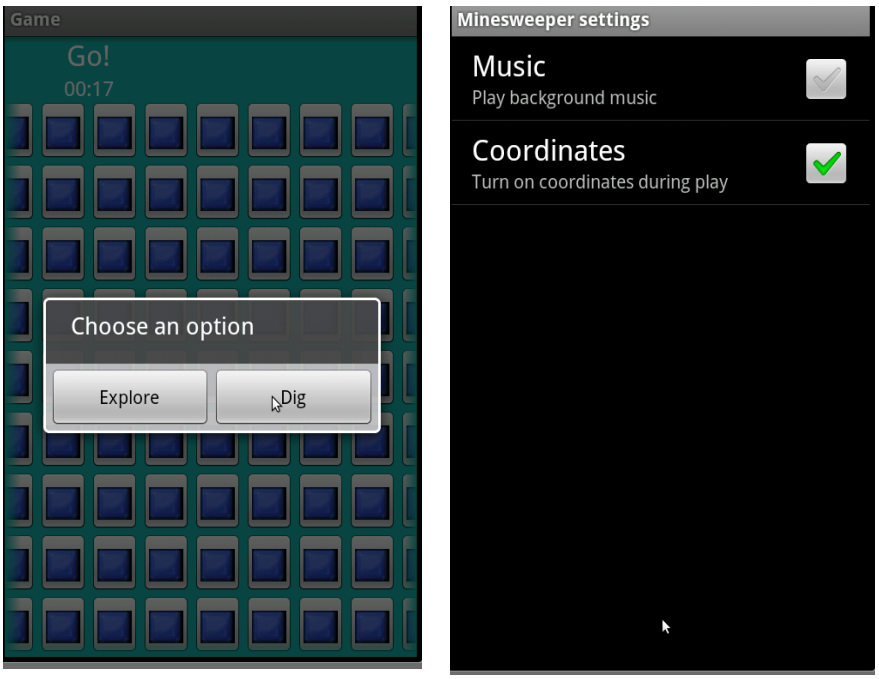


Figura 17. Primera versión mejorada - Buscaminas (1.0a1)

En la versión 1.0a2, para activar/desactivar el modo exploración era necesario pulsar el botón menú del dispositivo en lugar de la pulsación larga de la versión 1.0a1. Con una pulsación simple se destapa o se pone una bandera según el modo que se haya activado.

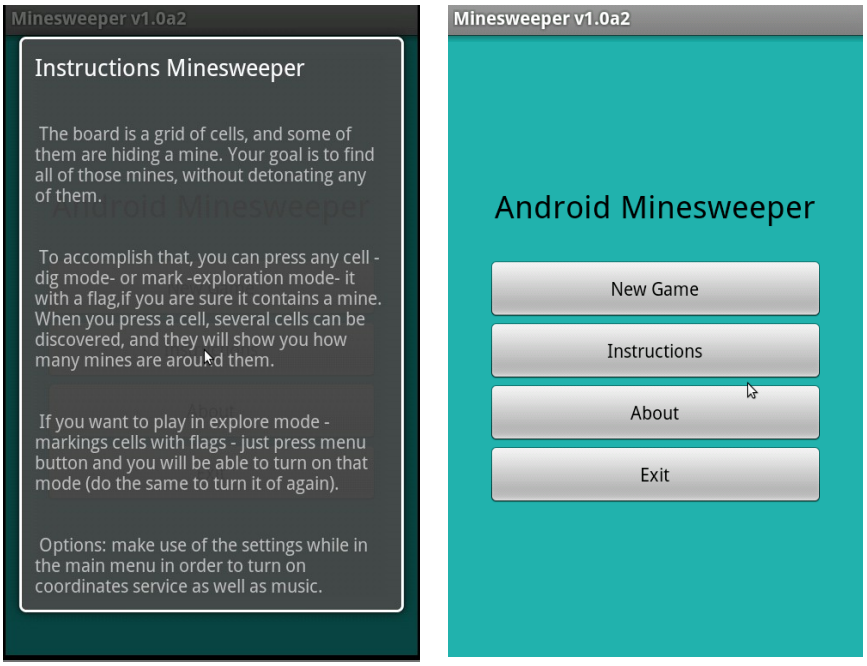


Figura 18. Primera versión mejorada - Buscaminas (1.0a2)

Los resultados obtenidos de estas versiones iniciales no fueron satisfactorios, pues dificultaba el control de los eventos por parte del desarrollador y la experiencia en el juego resultaba más incómoda que en el caso de *VoiceOver*. Como se ha descrito en la sección 2.5.1, el servicio *VoiceOver* de *iOS* no sólo es un lector de pantallas, sino que cambia completamente la interacción con el dispositivo. De modo que según el número de dedos que se utilicen y el gesto que se realice con ellos, el sistema reproduce una descripción del elemento de la interfaz o realiza una acción. Con *Talkback* el cambio de foco de una vista a otra era lo que provocaba la activación del lector de pantallas**,** lo que resultaba insuficiente para satisfacer todas las necesidades de interacción de los usuarios. Era conveniente poder activar el sintetizador de forma controlada por el programador.

Se trató de controlar la síntesis de voz apoyándose en el motor *text to speech* nativo de *Android*. La aplicación ya no depende de *TalkBack*, de manera que la reproducción de voz dentro del juego se activa sobre eventos táctiles o pulsaciones.

Para controlar la pulsación de una celda se emplea la duración del evento (pulsación larga). Así mismo, se trató de facilitar la forma en la que el usuario realizase estos eventos utilizando *zoom* o recursos gráficos de mayor tamaño (Figura 19) para facilitar la precisión de las pulsaciones. El principal problema de esta versión 1.0b, además de la evidente pérdida de la visión global del campo de minas, es la dificultad que puede suponer para un ciego la acción de arrastre (*scroll) sobre* la pantalla.

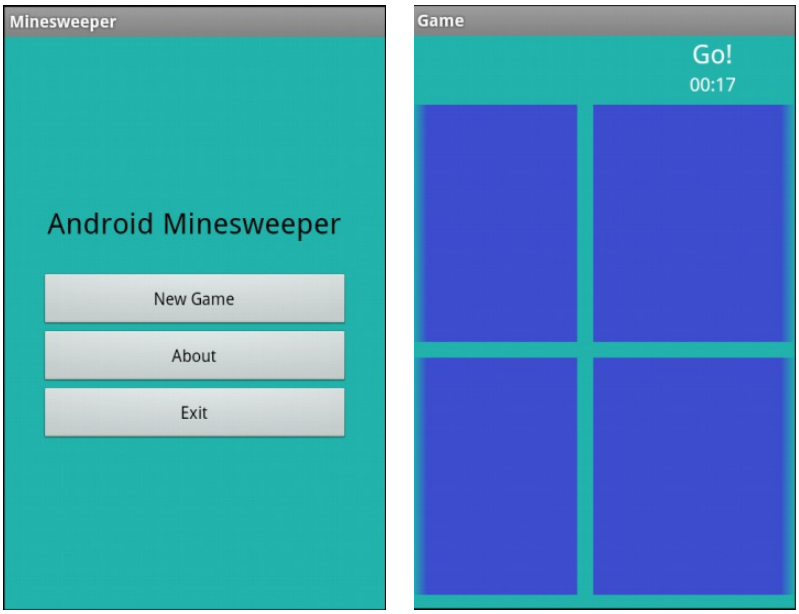


Figura 19. Segunda versión - Buscaminas (1.0b)

En la última versión 1.0c, se desarrolla una nueva vista sobre la que se actúa directamente en un *canvas*. La principal diferencia entre trabajar sobre un *canvas* y utilizar los elementos gráficos de la interfaz *UI* de *Android*, es que con el *canvas* se puede gestionar cada evento de manera personalizada, aspecto que no se puede lograr con los elementos gráficos disponibles de *Android*. Esto dota de un mayor control sobre la interacción y el retorno de la información al programador, además de ser un estilo de programación frecuente en el desarrollo de juegos.

Durante la partida es posible alternar entre una vista global del juego y una vista centrada en una casilla, para facilitar la acción de destapar tal y como se puede observar en la Figura 20 y Figura 21.

Además, cambia la forma de interacción a una más similar al juego de *iOS*, del que se parte como referencia. En la vista general, para cambiar el foco sobre una casilla y leer su estado se realiza una pulsación, para revelar su contenido doble pulsación y la exploración se realiza con una triple pulsación. En cambio, en el modo alternativo basta con hacer una pulsación sobre la zona central para destapar, o en las áreas asociadas a las flechas para desplazarse.

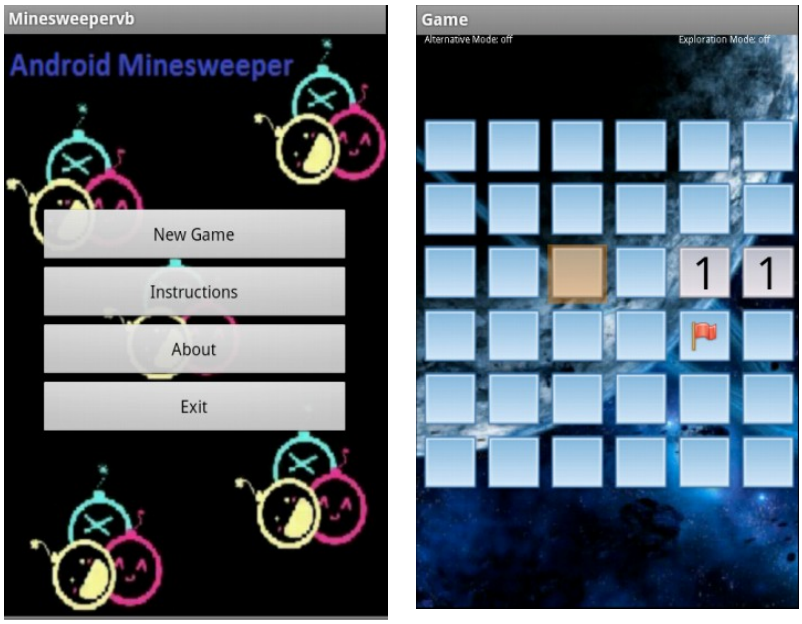


Figura 20. Menú Principal - Modo vista general - Cuarta versión Buscaminas (1.0c)

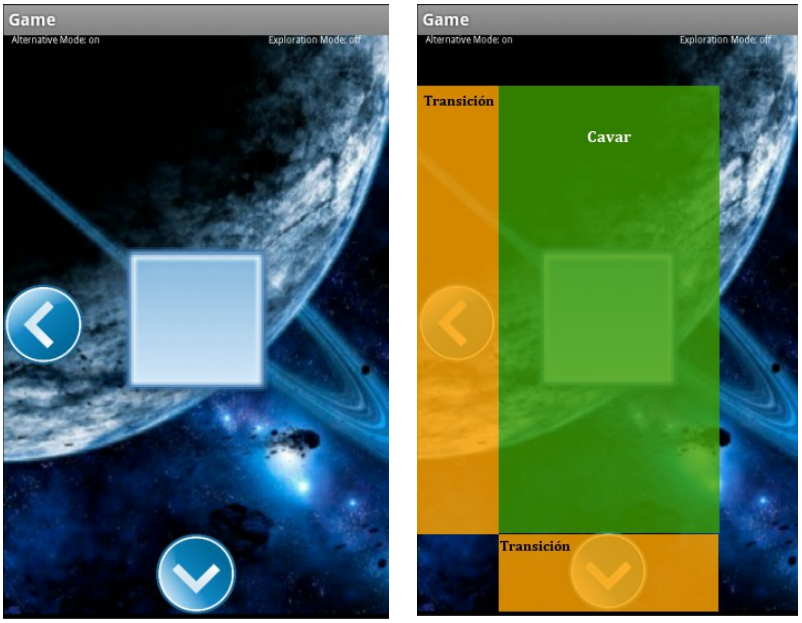


Figura 21. Modo vista alternativa - Cuarta versión Buscaminas (1.0c)

Al igual que en la versión 1.0b, se ofrece la posibilidad de activar la síntesis de voz según se realice *scroll* sobre cada una de las celdas para informar al usuario sobre su estado.

3.2.5. Conclusiones

Uno de los motivos principales por los que se desarrolló una evolución en versiones de esta aplicación, fue la incompatibilidad de los sistemas de accesibilidad de Android 2.3.x con los mecanismos de interacción para juegos. Es decir, *Talkback* activaba la lectura cuando se producía un cambio de foco, por lo que esta

herramienta de accesibilidad no satisfacía todas nuestras necesidades. Se necesitaba controlar la activación del lector de pantallas de una forma personalizada, por lo que fue necesario integrar soluciones propias. Por ello, a partir de este desarrollo se incorpora esta funcionalidad de gestión de motor de síntesis de voz a *BFG Toolkit*, que sería utilizada en todos los desarrollos posteriores. Esta nueva funcionalidad es un sistema *screen reader* adecuado para juegos.

En cuanto al diseño del modo de interacción, hay que tener en cuenta que la interacción en eventos críticos no se realice mediante un evento simple de pulsación sobre la pantalla. Es decir, a la hora de tratar los eventos se tiene que estar seguro de que el usuario realmente quería generar ese evento sobre esa posición en la pantalla y no estaba tanteando o pulsando accidentalmente en un sitio no deseado. Además, también se encontró el problema del número de teclas de los dispositivos móviles, es decir, la interacción mediante teclas es un mecanismo muy cómodo para las personas con discapacidad visual, pero no es viable en los dispositivos móviles actuales que tienden a tener cada vez menos teclas y por lo tanto, a volcar toda su interacción mediante las pantallas táctiles, las cuales resultan más difíciles de manejar por este tipo de personas.

3.3. Golf

3.3.1. Introducción

Una vez finalizado el buscaminas, se propusieron **varias alternativas** para el desarrollo del siguiente juego. La primera de ellas fue un ***Pinball*** clásico con dos elementos, las paletas y la pelota, siendo esta última la única fuente de sonido. El motivo de su propuesta fue que se incorporaba un elemento a localizar con respecto a las palas, la pelota, por lo que dificulta un poco más el desarrollo frente al buscaminas, ya que en éste último bastaba con dar información acerca del estado en el que se encontraba el tablero.

Otra alternativa fue la realización de un juego genérico tipo ***point & shoot***, como *Paper Toss*[20], cuya dinámica consiste en alcanzar un objetivo mediante el lanzamiento de un objeto a través de un evento de arrastre sobre la pantalla, del que se tiene en cuenta el ángulo con respecto a la normal hacia el objetivo. Además, se puede considerar la fuerza de rozamiento provocada por el viento para incrementar la dificultad. Del mismo modo que en un *Pinball*, se dispone de un elemento a localizar su posición con respecto al objetivo, que en el caso de *Paper Toss*, es la papelera.

[20] https://play.google.com/store/apps/details?id=com.bfs.papertoss&hl=es

Figura 22. Juego Paper Toss para Android

Finalmente, se propuso un juego tipo ***Puzzle Bobble***, en el que a cada color se le asociaría un sonido diferente y cuya acción se limitaría a hacer coincidir la burbuja en el lanzador con aquella a la que se esté apuntando. Se descartó por la simplicidad que supondría la dinámica del juego y por tanto, quedaría limitada a comparar melodías.

Figura 23. Juego Puzzle Bobble

Se seleccionó un juego con una mecánica de juego tipo *point & shoot*, que consiste en lanzar algún objeto hacia un objetivo determinado, por la facilidad que supone la identificación de la fuente de sonido para el jugador y porque ofrece más entretenimiento que las otras alternativas propuestas.

3.3.2. Diseño del juego

En la pantalla de juego se dispone de los elementos principales del mismo, como son la pelota y el hoyo, así como de una zona para realizar el lanzamiento de la pelota situada en la parte inferior de la pantalla. El objetivo consiste en lanzar la pelota para que alcance el objetivo, que en este caso es una bandera.

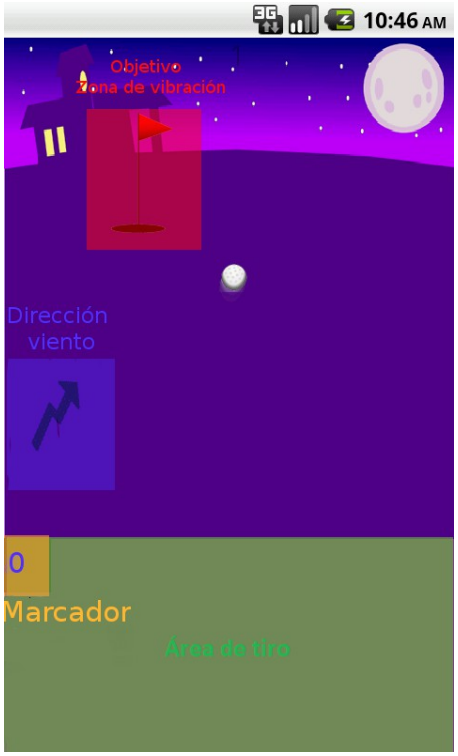


Figura 24. Descripción zonas en pantalla Golf

Según el éxito del lanzamiento, se reproducirá un sonido que transmitirá una sensación positiva o negativa, según sea el caso. También se llevará un registro de las máximas puntuaciones de partidas. Cabe destacar en este caso, la importancia de los tutoriales que se introdujeron en la sección 3.1 de cara a instruir al usuario ciego sobre los sonidos que están asociados a cada uno de los lances y elementos del juego.

Otro aspecto a destacar es el elemento viento, pues estas aplicaciones, además de dirigirse a un público específico como es el de las personas con discapacidad visual, incluyen características que añaden cierta dificultad para aquellas personas sin discapacidad visual. Este elemento hará que el lanzamiento de la pelota se desvíe en una dirección, que irá cambiando de manera aleatoria y que se indicará con una flecha que se mostrará en pantalla. Esta opción no está disponible en el modo de interacción para ciegos (ver sección 3.1).

3.3.2.1. Modos de juego

En *Golf* se dispone de dos modos de juego. Uno es el **modo 18 hoyos**, en el que se deberá obtener la mayor puntuación posible en 18 lanzamientos. Por cada acierto que se obtenga aumentará el marcador de puntuación en 25 puntos. En cambio, por cada lanzamiento que se falle el marcador disminuirá en 5 puntos. Para finalizar este modo se deberá acertar los 18 hoyos, ya que en caso de fallar alguno de éstos, se volverá a presentar la posibilidad de realizar el lanzamiento de manera satisfactoria, pero habiendo disminuido la puntuación en el marcador.

El otro modo de juego es el llamado **modo libre**, en el que el marcador mostrará información acerca del mayor número de hoyos seguidos que se han acertado hasta el momento y el número de hoyos seguidos acertados actualmente. En este modo de juego no hay limitación en el número de hoyos jugados, por lo que la duración de éste es infinita.

3.3.3. Diseño de la interacción

Los controles de que se disponen son muy amplios. Por defecto, para lanzar la pelota se deberá realizar un evento de *scroll* o arrastre para determinar el ángulo deseado en el lanzamiento y soltar la pulsación rápidamente, como si de un tirachinas se tratase.

Además, en este juego se introducen los perfiles de usuario, que proporcionan ayuda en cuanto a la configuración se refiere, ya que se dispone de un amplio abanico de opciones de accesibilidad que pueden combinarse para proporcionar una buena experiencia de juego al usuario con discapacidad visual.

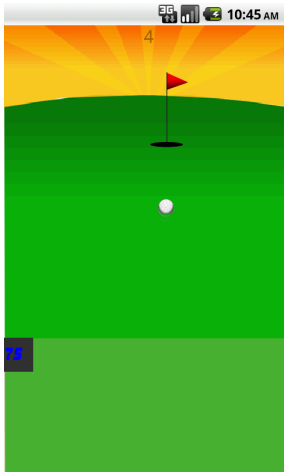


Figura 25. Juego Golf Accesible

3.3.3.1. Opciones de configuración

A continuación se muestran las distintas opciones que se ofrecen en la configuración de *Golf* y su explicación:

- **Efectos de sonido** para ayudar a localizar la bandera, mediante pulsaciones táctiles por la pantalla.
- **Un modo alternativo de lanzamiento**, que consiste en realizar un evento *scroll* o arrastre, al igual que en el modo por defecto, pero con la diferencia de que para definir el ángulo se puede emplear el tiempo necesario y liberar la pulsación cuando se desee.
- **Efecto de vibración** para ayudar en la determinación del ángulo que hay que tomar en el lanzamiento para alcanzar el objetivo. Se basa en estímulos hápticos cuando el usuario se encuentra en un cierto rango de acierto.

- **Sonido estéreo** que ayuda a encontrar el objetivo. Su función es la misma que la opción de vibración, pero mediante estímulos sonoros.
- **Disminución progresiva del volumen cuando se realiza un lanzamiento**. Esta opción emite un efecto de sonido, que resulta ser más intenso, dependiendo de si el resultado del lanzamiento está más cerca o más lejos de la bandera. Además, si se utilizan auriculares con esta opción de configuración, se puede localizar el objetivo. Por ejemplo, se lanza la pelota y la desviación de la trayectoria se ha producido hacia la derecha del objetivo, entonces el efecto de sonido se escuchará únicamente por el auricular derecho. De esta forma se sabrá en el siguiente intento, que se deberá desviar el lanzamiento hacia la izquierda.

A continuación se muestra un gráfico con la distribución de las opciones de configuración en los distintos perfiles propuestos.

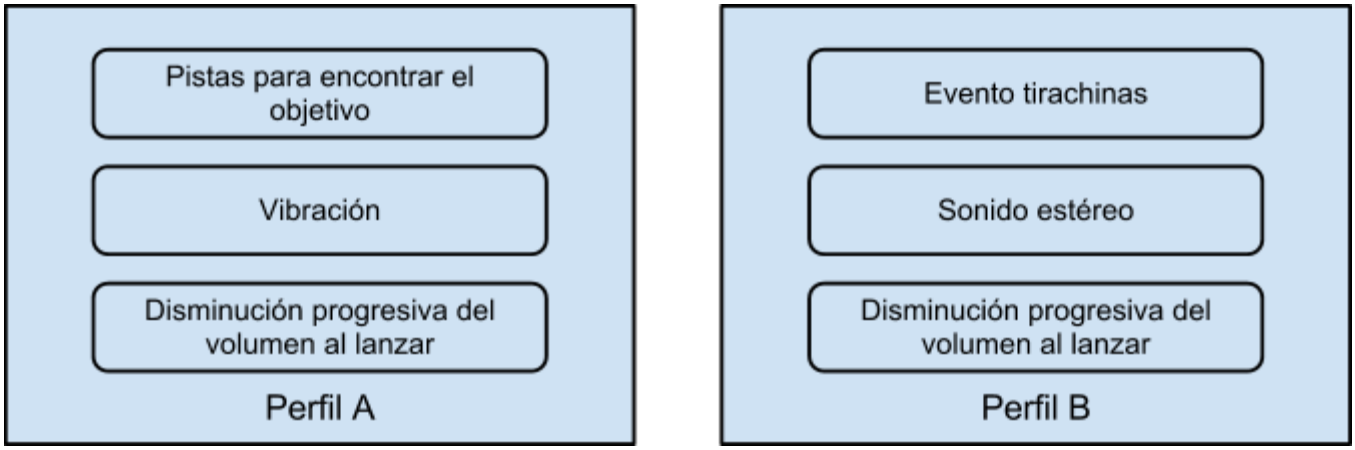


Figura 26. Diagrama de configuraciones por perfiles

La distribución de las opciones de configuración por perfiles tiene como objetivo facilitar al usuario la configuración del juego. Como podemos ver en el diagrama, en los dos perfiles se ofrece la opción de la disminución progresiva del volumen y una opción que ayuda al usuario a realizar el evento de lanzamiento mediante eventos hápticos, en el perfil A o eventos sonoros en el perfil B. La principal diferencia entre ambos perfiles, es que en el perfil A se ofrece la opción de las pistas para encontrar el objetivo mediante pulsaciones, mientras que en el perfil B cambia el modo de lanzamiento a uno más sencillo, ya que se puede mantener la pulsación mientras se toma el ángulo deseado, para posteriormente soltar la pulsación y con ello completar el disparo.

En conclusión, el **perfil A** dispone de **más ayudas** para localizar el objetivo, mientras que el **perfil B**, ofrece un modo de **lanzamiento más cómodo** y menos ayudas para la localización del hoyo. El usuario no está obligado a mantener las opciones de configuración que se ofrecen en los perfiles, ya que tiene la libertad de configurar las opciones como desee. Los perfiles simplemente son una forma de configuración rápida del juego.

3.3.4. Conclusiones

Tras este desarrollo, se concluye con la necesidad de hacer que las aplicaciones sean lo más **configurables** posible, puesto que se puede tener diferentes perfiles de jugador cuyas preferencias en términos de opciones de accesibilidad sean distintas o las decisiones de diseño no sean del todo acertadas en la práctica, por lo que es importante que el usuario pueda tener varias alternativas. Ésta es además una recomendación típica en el campo de la accesibilidad, donde se aboga por una mayor flexibilidad en la configuración de interfaces.

Otra razón es la importancia de unos **recursos de sonido** agradables de cara a diseñar un juego que se va a basar en estímulos sonoros para proporcionar información al usuario. Esto supone una dificultad, ya que no es fácil encontrar sonidos cuando no hay una metáfora de referencia clara vinculada al elemento del juego o a la idea que se quiere transmitir o bien la hay, pero el sonido asociado es desagradable.

Finalmente, este desarrollo contribuyó a ***BFG toolkit*** con una mejora del subsistema de entrada basado en eventos, rediseño del sistema de sonido y depuración del *toolkit* en general, pues es el primero de todos los juegos en utilizar íntegramente la funcionalidad de esta herramienta.

3.4. Zarodnik

3.4.1. Introducción

Se tenía un juego de tablero y otro con el que se esperaba por parte del usuario una actitud más activa para percibir todos los eventos que se produjeran en pantalla. Se decidió hacer un juego con un poco más de acción, es decir, un juego en el que el usuario deba tomar decisiones de forma más dinámica, aumentando la complejidad de localización mediante el aumento progresivo del número de elementos e introducción de movimiento.

Para ello, una opción era adaptar el juego clásico ***Snake*** para personas con discapacidad visual. En este juego se controla a una serpiente cuyo objetivo es conseguir la mayor cantidad de comida posible, moviéndola por un área delimitada, evitando chocar con los límites del mismo, los obstáculos que pudiera haber dentro o incluso con su propio cuerpo, ya que por cada alimento que se ingiere el cuerpo crece dificultando el movimiento por dicho área.

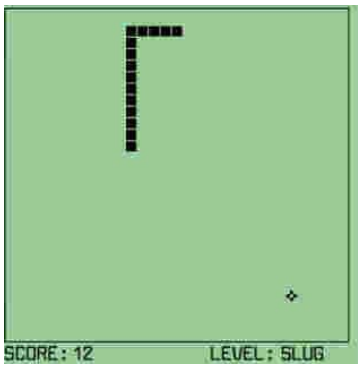


Figura 27. Juego Snake

Durante el análisis de este juego surgió otra idea que podría aportar más jugabilidad, basándose en el juego ***Spore Origins*** disponible para el sistema operativo *iOS* de *Apple*. La idea consistía en conseguir comer el mayor número de presas posible, sin ser devorado por los depredadores que irían apareciendo a lo largo de la partida. Este juego aportaba algo realmente nuevo con respecto a los juegos anteriores, el hecho de tener que identificar distintas fuentes de sonido con el foco móvil, puesto que el protagonista se va desplazando por la pantalla para identificar los distintos elementos. Esta idea es la que posteriormente daría lugar a *Zarodnik*.

3.4.2. Diseño del juego

En este juego se controla a un monstruo llamado *Zarodnik*, cuyo objetivo es salvar a los *Obeaune*, una raza de anfibios extremadamente débiles, que se encontraban amenazados por otra raza más fuerte llamada *Sprouf*. Esta aventura se desenvuelve en el océano de *Izlude*, uno de los más grandes del mundo y que está dominado por los *Sprouf*.

El flujo de movimiento se hace por pantallas. Es decir, inicialmente el jugador comienza en una pantalla en la que habrá un *Obeaune* al que deberá salvar, rodeado por varios *Sprouf*. El objetivo es almacenar al *Obeaune* en el interior de *Zarodnik*, para posteriormente pasar a la siguiente pantalla mediante la aproximación del protagonista *Zarodnik*, a uno de los extremos de la misma

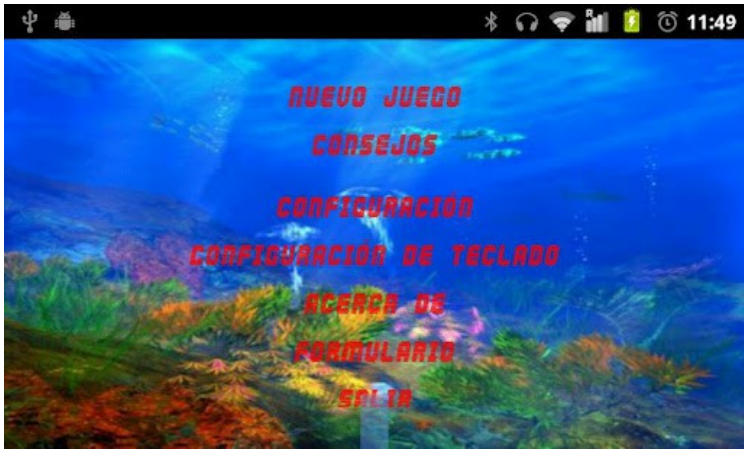


Figura 28. Menú principal Zarodnik

Como ya se ha mencionado, se trata de conseguir el mayor número posible de *Obeaunes*, sin ser devorado por los *Sprouf*. Existen, como en cualquier juego de este tipo, niveles de dificultad, objetos que al ser devorados modifican algún factor del juego que puede ser perjudicial o beneficioso para el protagonista. A través de estos objetos, se logra introducir al usuario las opciones de accesibilidad dentro del propio mundo del juego. En cambio, en juegos como Golf (ver sección 3.3.3.1) formaban parte de los menús y el usuario tenía que configurarlas si las quería utilizar. Entre ellos se encuentran:

- Emisor de radio *Lotam*; ayuda al usuario a determinar la dirección que debe tomar para localizar al *Obeaune* que se encuentra en esa pantalla.
- El legendario *Chainfish*: da pistas para poder evadir a los *Sprouf* durante un cierto tiempo.
- La cápsula del profesor Konstatin: permitirá rescatar un mayor número de *Obeaune* reduciendo el tamaño de *Zarodnik*.

A continuación se muestra la representación gráfica de los distintos elementos que se podrán encontrar en el juego:

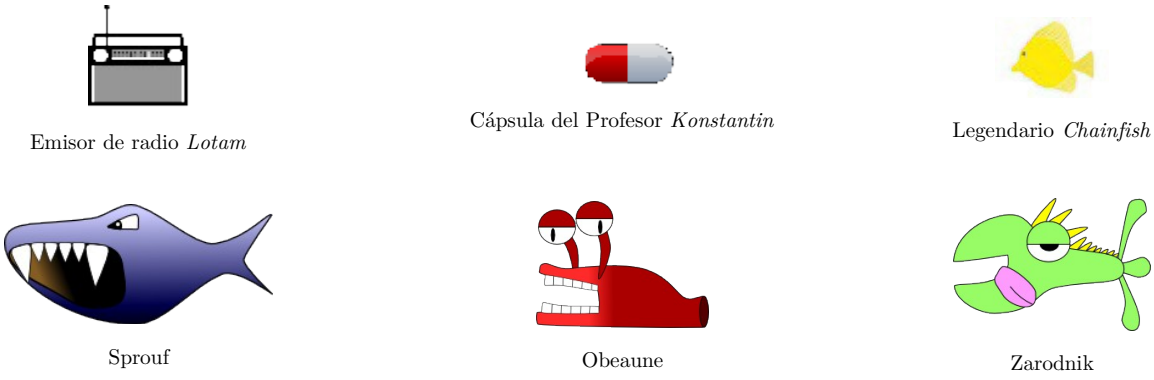


Figura 29. Elementos Zarodnik

3.4.3. Diseño de la interacción

Cada partida comienza con el personaje principal centrado en la zona inferior de la pantalla. El jugador tiene que desplazar al protagonista, guiándolo con el dedo a través de la pantalla mediante eventos de pulsación y *scroll*. Conforme se recorre la pantalla, el jugador irá escuchando sonidos representativos de los elementos de la habitación. Si éstos indican peligro (*Sprouf*), el jugador tratará de alejarse de la fuente de sonido. Si no lograse escapar del depredador la partida finalizará y se hará un recuento de puntuación.

En cambio, si se escucha un sonido que representa un elemento con efectos positivos sobre el protagonista, el jugador deberá dirigirse hasta este elemento (por ejemplo, un *Obeaune*) incrementándose la puntuación y reproduciendo un sonido de confirmación en caso de éxito. Si el elemento es un Obeaune, *Zarodnik* (el protagonista) aumenta de tamaño, de forma que se incrementa la dificultad del desplazamiento por las siguientes pantallas, puesto que resulta más complicado evadir a los *Sprouf*.

Además, cuando el jugador detiene la acción de *scroll* no sólo se detiene *Zarodnik*, sino todos los elementos en pantalla, permitiendo de este modo que el jugador analice detenidamente las fuentes que está percibiendo en ese instante determinado. Esta decisión se tomó con el objetivo de proporcionar mayor comodidad en la interacción para el discapacitado visual, puesto que si el usuario ciego deja de realizar *scroll* para pensar en qué dirección moverse y no se detienen los elementos del juego que están en pantalla, la dificultad podría resultar excesiva. Esta opción también se puede interpretar como una forma de pausar el juego.

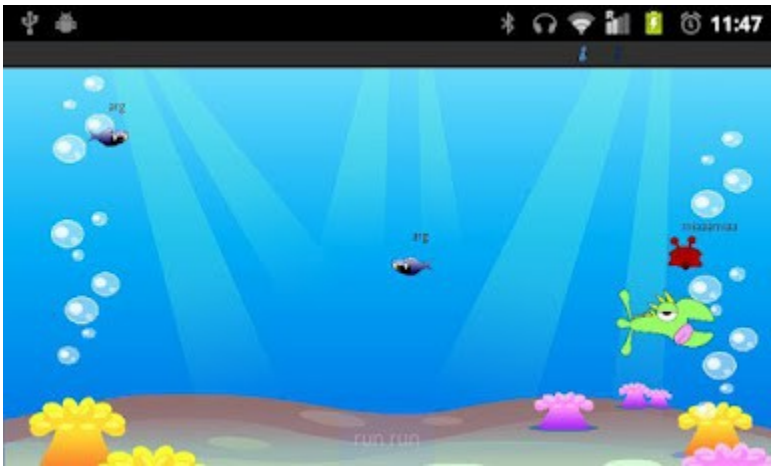


Figura 30. Juego Zarodnik

En *Zarodnik* se le da una gran importancia a la identificación de los elementos del juego distribuidos sobre la pantalla mediante sonido 3D. El mecanismo se basa en asociar diferentes fuentes de sonido a los elementos del juego mencionados en la sección anterior. Se utilizan propiedades del sonido como la atenuación (distancia a la fuente) y el tono (posición relativa en el eje de ordenadas) para indicar la posición relativa del oyente (jugador) con cada una de las fuentes como se ve en la Figura 31. A modo de ejemplo, si el jugador se está aproximando a un *Sprouf* por la izquierda y se encuentra a una determinada distancia, se escuchará su sonido representativo (del *Sprouf*) por el auricular derecho.

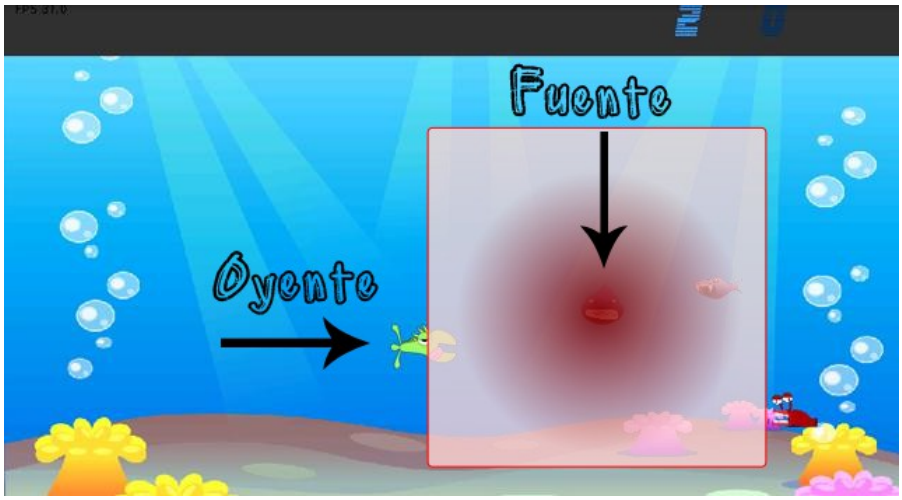


Figura 31. Demostración de sonido 3D

Por último, cabe destacar la especial importancia que en este juego adquieren los tutoriales que se abordaron en la sección 3.1, ya que son fundamentales de cara a instruir al usuario sobre los sonidos que están asociados a cada uno de los elementos del juego.

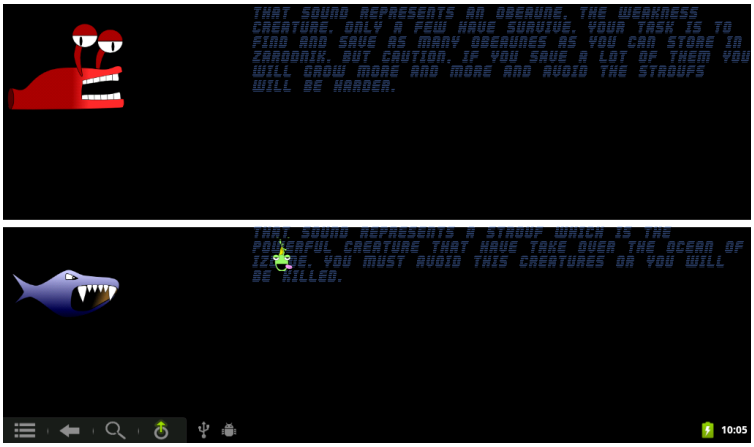


Figura 32. Tutorial Zarodnik

3.4.4. Conclusiones

Con este desarrollo surgió la necesidad de incorporar información adicional a los estímulos sonoros convencionales (voz y sonidos convencionales). Sobre todo en ciertos géneros en los que se requiere de una interacción más fluida, como es el caso de *Zarodnik*. No sólo se proporciona información mediante el tipo de sonido, sino que también se pueden utilizar propiedades del mismo, como la atenuación, el tono o la amplitud, con el fin de aportar información necesaria al usuario discapacitado, para que pueda responder a los hechos que ocurren en el mundo del juego. Por ello, a partir de esta aplicación se incorpora el sistema de sonido 3D a la herramienta para desarrollo de juegos accesibles que se describe en la sección 4.1.1.3.

3.5. The Shadow of the Past

3.5.1. Introducción

El siguiente propósito fue desarrollar un juego con un fuerte componente narrativo, lo que aportaría al proyecto un género nuevo y con un modelo de interacción diferente. El motivo del desarrollo de este tipo de juego, es la fusión de los audiolibros con los videojuegos de manera que fuese posible llevar a cabo la dinámica y narrativa de un juego de aventuras a un discapacitado visual. También es importante destacar el alto potencial educativo que tiene la narrativa en este tipo de juegos (Dickey, 2006)(Amory, 2001).

Los videojuegos que han influenciado en *The Shadow of the Past* son todo juegos de tipo *point and click*. A continuación se detallan algunos de estos juegos.

Broken Sword: La leyenda de los templarios trata de un abogado y una periodista que se ven involucrados en la investigación de un asesinato ocurrido en un café. La principal influencia de este juego, es la discriminación de opciones conversacionales, es decir, dispone de **múltiples finales** pudiendo incluso llegando a situaciones en las que el jugador acaba su partida de forma anticipada debido a una mala decisión en la selección de opciones en ciertos puntos del juego.

Figura 33. Broken Sword: La leyenda de los templarios

New York Crimes es una aventura gráfica que trata sobre crímenes. Narra la historia en la que se ve envuelto *John Yesterday*, un hombre que ha sido contratado por el filántropo *Henry White* para investigar los asesinatos de mendigos que se suceden en la ciudad de Nueva York. La temática de **novela policíaca** es el **vínculo principal con *The Shadow of the Past***.

Figura 34. New York Crimes.

3.5.2. Diseño del juego

En cuanto a la mecánica de juego, se puede decir que se trata de una narración de una novela policíaca a través de la interacción con personajes mediante el dispositivo móvil. La estrategia que hace interesante este juego es el hecho de introducir al jugador dentro de una historia inmersiva, dándole múltiples posibilidades de decisión en los diálogos de conversación.

Figura 35. Menú Principal The Shadow of the Past

La historia se desarrolla en *Sewer City*. El jugador controla a John Tidiac, un ex detective al cual un antiguo compañero de trabajo pide ayuda para resolver un caso, ya que a pesar de su despido por parte del jefe de comisaría, sigue gozando de una gran reputación con un largo historial en la resolución de casos. Se trata de un homicidio que el jugador deberá resolver inspeccionando, interrogando y ante todo eligiendo la opción correcta en cada caso.

Durante la investigación a lo largo del juego, se mostrarán distintas opciones de diálogo. El jugador deberá ser cuidadoso porque algunas de las opciones que se plantean podrán llevarle a una situación en la que el protagonista *John Tidiac* muera y por lo tanto falle en la resolución del caso.

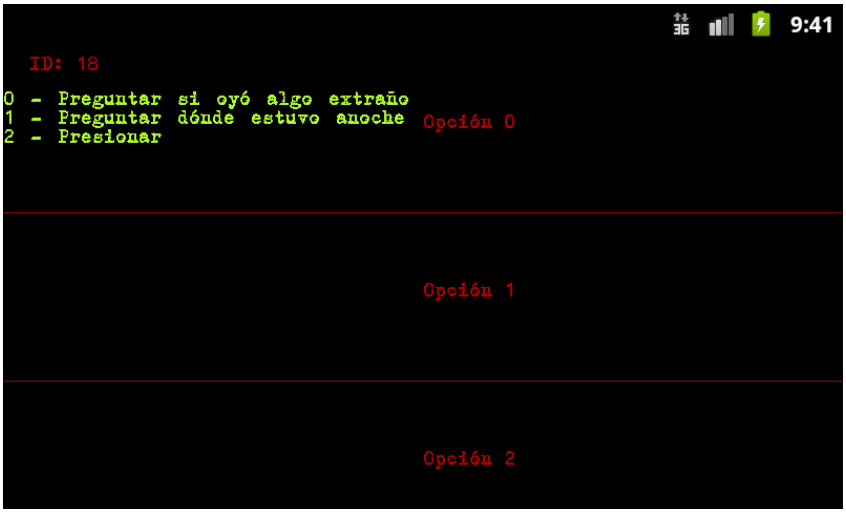


Figura 36. Opciones de diálogo The Shadow of the Past

3.5.3. Diseño de la interacción

En *The Shadow of the Past* se puede encontrar dos tipos de pantallas. En una de ellas simplemente se describen los hechos ocurridos, narrados por el lector de pantallas. Para avanzar en el juego el jugador simplemente deberá hacer una doble pulsación. El otro tipo de pantalla es en la que se muestran las distintas acciones a realizar, en la cual si el usuario arrastra el dedo sobre la pantalla, el lector las irá describiendo, para finalmente hacer una doble pulsación sobre la deseada.

3.5.4. Conclusiones

Al diseñar juegos accesibles para discapacitados visuales basados en aventuras gráficas, se ha aprendido a valorar la importancia de la trama en los juegos para este tipo de usuarios.

Una de las mejoras que se podría desarrollar, es la adición de eventos basados en la comunicación háptica, de manera que el usuario reciba una respuesta vibratoria ante determinados eventos. Además, existe la posibilidad de añadir distintos tipos de escenas, como pudieran ser minijuegos, ya que como se menciona en la sección 4.2.6, se dispone de un campo para especificar el tipo de la escena. También se podría hacer uso de acelerómetros en el diseño de minijuegos o para dotar de un mayor dinamismo a la toma de decisiones en los diálogos mediante la ejecución de diferentes movimientos con el dispositivo para seleccionar diferentes opciones.

Finalmente, se incluyó a la herramienta *BFG Toolkit* el módulo de carga de aventuras generado gracias al desarrollo de este juego.

3.6. Comparativa de interacción

A continuación, se muestra a modo de resumen, una tabla donde se puede observar la forma en la que un jugador interacciona con cada uno de los juegos presentados, siguiendo el modelo descrito en la sección 2.1.

Buscaminas	**Golf**	**Zarodnik**	**The Shadow of the Past**	**Jugador**
Escucha el estado de la casilla que el usuario ha seleccionado mediante un evento táctil.	Escucha por el auricular izquierdo o derecho un sonido tras el lanzamiento de la pelota. También se reproduce un sonido que indica un fallo.	Escucha un sonido procedente de un enemigo.	Escucha un fragmento de la historia. Se le plantean varias opciones para llevar a cabo.	1 - Recibe el estímulo
Decide destapar la casilla.	Decide volver a intentarlo.	Decide alejarse de él.	Decide seleccionar la primera de las opciones.	2 - Determina Respuesta
Realiza doble *tap* sobre la casilla	Describe el gesto de lanzamiento tipo "tirachinas" sobre la pantalla.	Arrastra el dedo sobre la pantalla en la dirección que desea moverse.	Arrastra el dedo por la pantalla y una vez localiza el botón de la opción deseada, realiza un doble *tap* sobre la misma para seleccionarla.	3 - Ejecuta la respuesta
Hay una mina y el juego acaba.	Acierta y avanza hasta el siguiente hoyo.	Finalmente colisiona con el enemigo y el juego finaliza.	El juego avanza hasta la siguiente escena de la historia.	Repite pasos 1,2,3

Tabla 6. Ejemplo de interacción de un ciego con cada uno de los juegos desarrollados

4. BFG Toolkit. Conjunto de herramientas para facilitar el desarrollo de juegos accesibles

Tras la exposición de los diferentes juegos desarrollados, en esta sección se introduce y analiza el conjunto de herramientas que se ha desarrollado para facilitar la elaboración de juegos accesibles. Este *toolkit* se presenta en dos niveles de abstracción, una parte más técnica o de bajo nivel y otra en la que se exponen las funcionalidades reutilizables que ofrece. Para concluir esta sección se incluye una breve guía de uso (para el programador).

4.1. Descripción a alto nivel

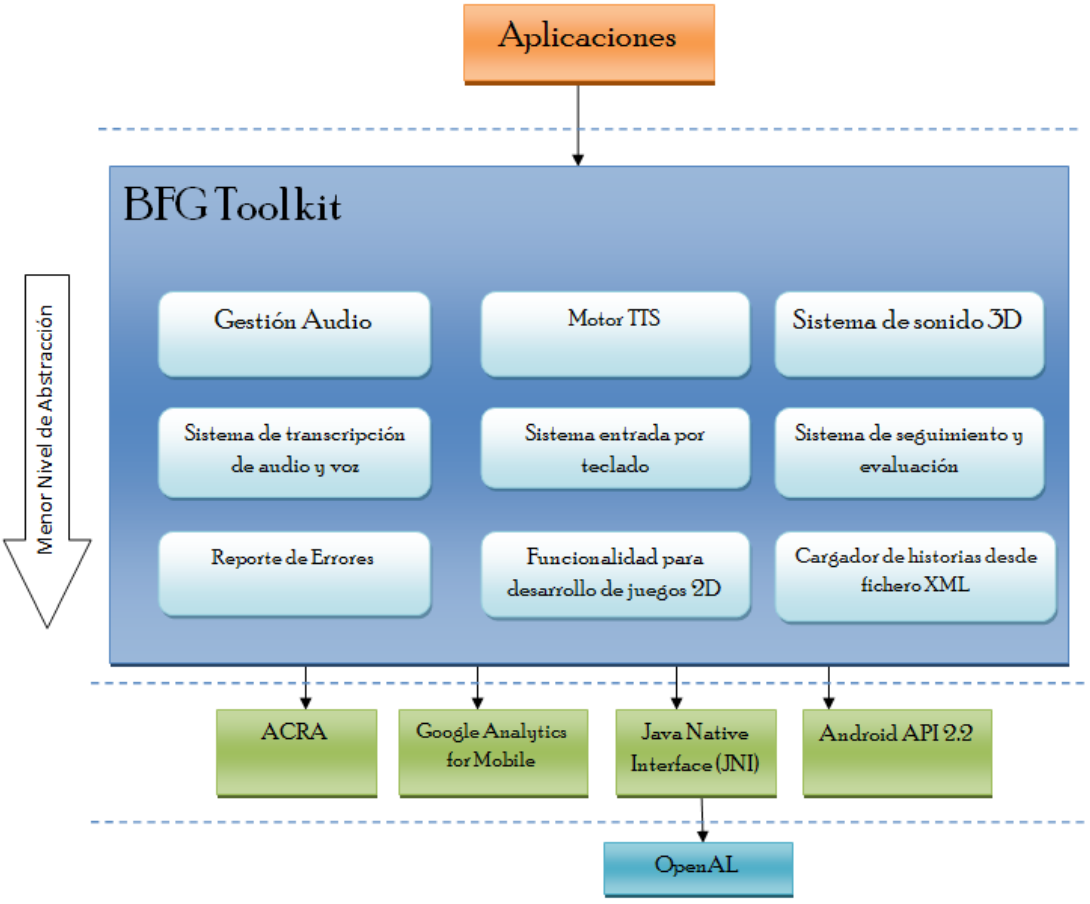


Figura 37. Diagrama funcionalidad básica de BFG Toolkit

Como se puede observar en la Figura 37, *BFG toolkit* está compuesto por una serie de módulos que proporcionan distintas funcionalidades, como son la gestión de audio, el motor de síntesis de voz, el sonido 3D, la transcripción de audio y voz, el sistema de entrada por teclado, el sistema de seguimiento y evaluación, el reporte de errores, la funcionalidad necesaria para el desarrollo de juegos 2D y el cargador de historias desde fichero XML. Algunas de estas funcionalidades trabajan conjuntamente con librerías externas:

- ACRA: módulo de reporte de errores.
- Google Analytics for Mobile: sistema de seguimiento y evaluación.
- OpenAl: hace uso de *Java Native Interface* (ver sección 6.1) para la integración del sonido 3D.

A continuación se exponen detalladamente algunas de estas características.

4.1.1. Sistemas basados en estímulos sonoros

4.1.1.1. Gestión Audio

BFG Toolkit proporciona funcionalidad necesaria para la reproducción de música en múltiples formatos, (ver Anexo IV) en cualquier momento de la ejecución de las aplicaciones. Aunque no es una opción de accesibilidad en sí misma, cabe destacar la importancia que tiene para los discapacitados visuales el disponer de **recursos de sonido de calidad** en los juegos.

Por otro lado, también permite la reproducción de efectos de **sonido significativos**, que sean representativos y transmitan una sensación o idea, como por ejemplo, unos aplausos pueden ser utilizados para informar sobre diferentes situaciones que pueden ocurrir en un juego: errores, aciertos o comienzos de partida, como se puede ver en la sección 3.3. Además, es recomendable la incorporación de tutoriales que instruyan al usuario sobre cada sonido y su representación.

4.1.1.2. Sintetizador de voz

Proporciona la funcionalidad básica para la gestión del sintetizador de voz habilitado en el sistema y cuyo principal uso es la transmisión de información al usuario ciego sobre lo que está ocurriendo en un juego.

En videojuegos pertenecientes a géneros en los que la mecánica del juego lo permita, se puede utilizar para **dar indicaciones al usuario** sobre el estado del mismo o sobre lo que tiene que hacer. También puede utilizarse para guiar al usuario especialmente en juegos de tablero o puzle (como Sudoku o Buscaminas) donde está presente la idea de tabla y en los que, la presión por el tiempo, la interacción con otros jugadores o la propia inteligencia artificial del juego no es fundamental.

Del mismo modo que se mencionaba en la sección 4.1.1.1, para los usuarios con discapacidad visual es importante que el motor de síntesis de voz tenga un **buen grado de naturalidad e inteligibilidad**. Por

ello, en las aplicaciones desarrolladas con *BFG Toolkit* en su primera ejecución, se realiza una recomendación acerca de la necesidad de instalación de un motor de síntesis de cierta calidad, si éste no se detecta.

4.1.1.3. Sonido 3D

A la hora de hacer juegos accesibles a discapacitados visuales que requieren de posicionamiento absoluto, el uso de un sintetizador de voz únicamente es insuficiente, pues no permite transmitir la localización de elementos en pantalla con suficiente rapidez y dinamismo. Esto hace que sea **necesario complementar la síntesis de voz con otro mecanismo**.

El sonido 3D se puede utilizar para la **localización e identificación de elementos en escenas 3D y 2D**. Para ello, un posible uso es el de asociar a cada elemento un sonido lo más representativo posible e **incorporar tutoriales** que instruyan al usuario sobre cada sonido y su representación. La diferencia con respecto a los efectos de sonido mencionados en la sección 4.1.1.1, es que el sonido 3D permite dar información adicional sobre la localización de los elementos mediante la utilización de propiedades del sonido como la atenuación (distancia y orientación de las fuentes con respecto al oyente) y la frecuencia (orientación de las fuentes con respecto al oyente). Como se puede ver en la Figura 38 en esta escena existen 14 fuentes de sonido cuyas propiedades irán modificándose según cambia la posición relativa entre cada uno de los elementos del juego a los que están asociados y el oyente (avatar del jugador).

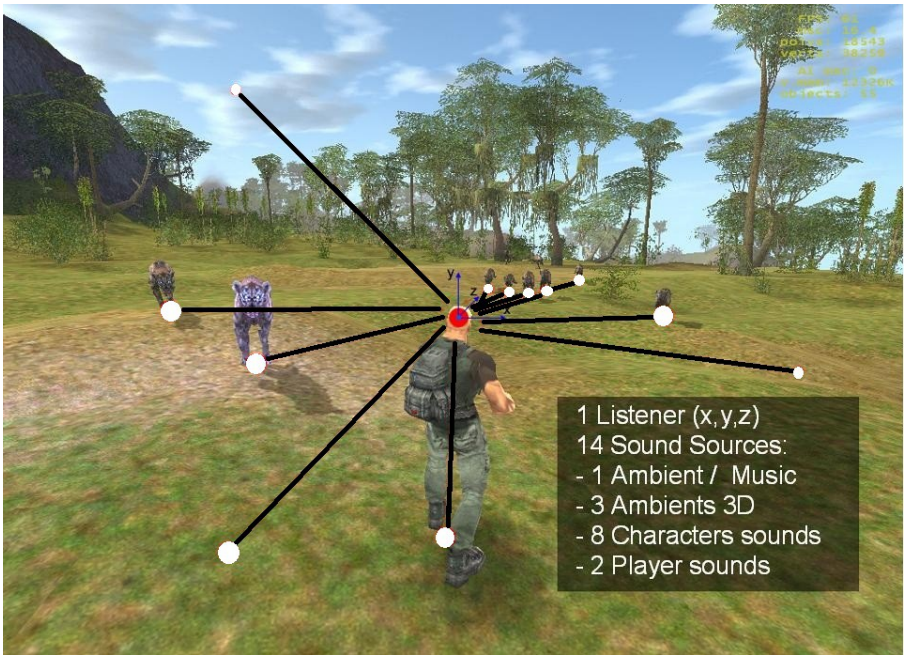


Figura 38. Escena sonido 3D[21]

[21] Propiedad de Action Forms ltd.

Otro posible uso del sonido 3D como característica de accesibilidad para ciegos, es la **utilización de puntos de referencia para orientar al jugador** ciego dentro de una escena 3D, de forma que si el usuario pierde la orientación en el juego, estos sonidos le ayuden a conocer su situación.

Para realizar su implementación se empleó una de las API de audio más conocidas, *OpenAL*[22].

4.1.2. Sistema de transcripción automática de voces y sonidos

BFG Toolkit proporciona herramientas para poder subtitular la voz del motor de síntesis, así como los sonidos que se reproduzcan con la funcionalidad del sistema.

Esto tiene especial interés por dos motivos. El primero es que permite hacer **aplicaciones más accesibles a personas con problemas de audición**. Aunque esté fuera del alcance del proyecto surge de forma natural la solución a esta problemática. En segundo lugar, permite **utilizar las aplicaciones sin sonido para personas sin discapacidad visual**.

4.1.3. Sistema de entrada por teclado

Este sistema permite asignar teclas a las diferentes configuraciones y acciones del juego, de forma que se puedan ejecutar distintas características de las aplicaciones usando los botones disponibles en un determinado dispositivo móvil.

Es importante de cara a hacer un juego más accesible para ciegos, si el tipo de juego lo permite, **incorporar un mecanismo de interacción alternativo basado en teclado**. De forma que, aunque pudiera ser menos práctico, si el usuario dispone de un móvil con teclado físico o un teclado conectado vía *micro-USB*, pueda utilizar la aplicación de una forma diferente al uso de los controles por pantalla táctil.

4.1.4. Feedback y evaluación

4.1.4.1. Sistema de seguimiento

BFG Toolkit proporciona un mecanismo de seguimiento basado en la librería de *Google Analytics for Mobile*, para conocer el uso que está realizando el usuario de la aplicación implementada con el sistema.

Permite el **seguimiento de actividades *Android***, de forma que se puede medir el tráfico de una determinada actividad de la aplicación por parte de los usuarios, como si de una página web se tratara, en términos de número de visitas o frecuencia de movimientos entre actividades. Es decir, si los usuarios visitan

[22] http://connect.creativelabs.com/openal/default.aspx

la actividad principal de la aplicación, se puede visualizar cuál es la siguiente actividad que suelen visitar con más frecuencia: juego, tutorial, configuración, etc.

Es posible definir **eventos personalizados** para conocer la interacción del usuario con cada actividad. De esta forma, se puede ver cómo está tratando de usar el usuario los juegos, si se ha comprendido cómo es el mecanismo de interacción o si utiliza los mecanismos de interacción planteados de forma adecuada o la curva de aprendizaje de los juegos es demasiado grande como para que los usuarios ciegos puedan utilizarlos.

También es posible definir **variables personalizadas** para almacenar cualquier información adicional que se desee sobre el dispositivo, o sobre la localización en la que se encuentra el usuario en un momento determinado.

Además, podemos conocer toda la información que proporciona *Google Analytics* para páginas web adaptada para aplicaciones *Android,* desde el país de origen de los visitantes, el sistema operativo del dispositivo móvil, el modelo del dispositivo o estadísticas de visitantes nuevos o que retornan.

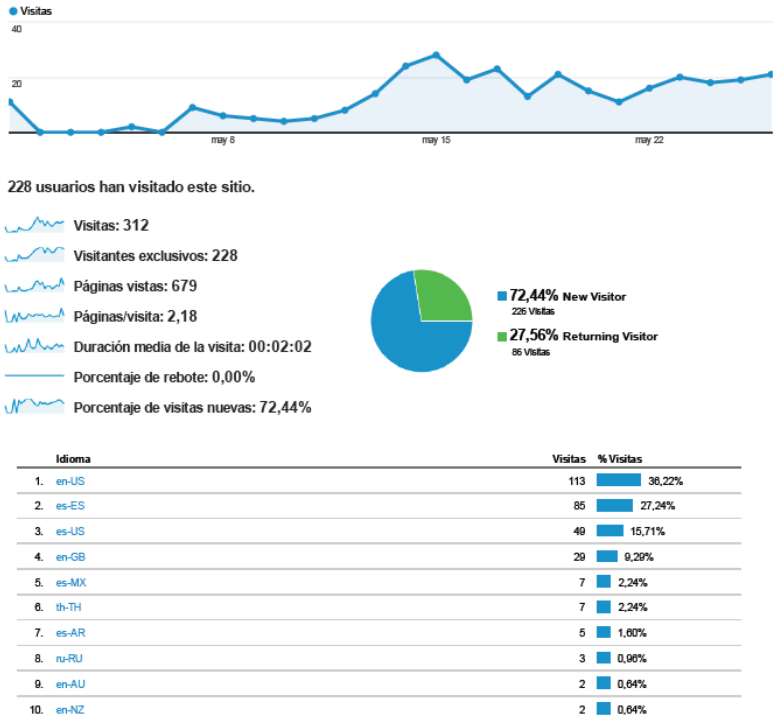


Figura 39. Página web de Google Analytics para Golf

Todas estas alternativas son interesantes de cara a conocer si las opciones de accesibilidad planteadas en el diseño de las aplicaciones son acertadas, o simplemente si se desea tener una fuente de información propia sobre el interés que han despertado las aplicaciones desarrolladas con *BFG Toolkit*.

4.1.4.2. Reporte de errores y evaluación

Aunque se puedan realizar pruebas sobre un conjunto representativo de dispositivos *Android*, dada la enorme variedad existente en el mercado, múltiples versiones del sistema operativo y resoluciones de pantalla diferentes, la realización de pruebas en muchos casos puede resultar **bastante costosa**. Por ello, es fundamental disponer en cualquier aplicación, de un **sistema que notifique al desarrollador de los errores** que pudieran surgir en la misma.

Se pueden enviar informes de error en forma de:

- Hojas de cálculo de *Google Docs*.
- Entradas en mensajes de correo electrónico.
- Consultas a servicios HTTP.

Los informes, una vez configurada la herramienta, se envían de forma automática en el caso de excepciones no controladas o también pueden ser enviados por demanda del programador.

Además de información sobre errores, es posible enviar cualquier otro dato que se desee, como por ejemplo, formularios de evaluación de las aplicaciones.

	A	B	C	D
1	**Día**	**Juego (Versión)**	**Dispositivo - SO**	**Respuestas**
2	25/05/2012 17:05:23	es.eucm.blindfaithgames.golfgame (3)	Motorola XT615 2.3.3	q1- NA q2- No tengo discapacidad visual q3- NA q4- NA q5- 1 q6- 1 q7- 1 q8- 3 q9- 2 q10- 2 q11- 1 q12-
3	23/05/2012 21:27:02	es.eucm.blindfaithgames.golfgame (3)	samsung GT-I9001	q1- NA q2- I have no visual disability q3- 2 q4- 2 q5- 1 q6- 2 q7- 2 q8- 2 q9- 2 q10- 2 q11- 2 q12-
4	07/05/2012 22:17:53	es.eucm.blindfaithgames.minesweeper (1)	LG Optimus 2X	q1- NA q2- No tengo discapacidad visual q3- 5 q4- 3 q5- 4 q6- 4 q7- 4 q8- 4 q9- 2 q10- 4 q11- 2 q12-
5	29/05/2012 07:33:54	es.eucm.blindfaithgames.golfgame(3)	samsung GT-S5830L	q1- NA q2- No tengo discapacidad visual

Figura 40. Hojas de cálculo con los formularios de evaluación de las aplicaciones.

Su implementación se ha realizado tomando como base la librería *ACRA*.

4.1.5. Funcionalidad para el desarrollo de juegos 2D

Además de características dirigidas a mejorar la accesibilidad, el *framework BFG Toolkit* incluye un sistema para el desarrollo de juegos 2D, que mediante una abstracción basada en entidades (ver sección 4.2), permite la **implementación de los elementos de una amplia gama de géneros de juegos**. Incorpora funcionalidad para gestión de colisiones, funcionalidad para liberar los recursos (sonido, imágenes etc.) de cada elemento del juego, guardar el estado de los juegos, control de la velocidad del juego, definición de la máquina de estados que modela las diferentes transiciones entre escenas de un juego, así como el dibujado y el control de movimiento de los elementos del juego.

Este sistema, así como una breve guía de uso (para el programador), se describen a más bajo nivel en la sección 4.2.

4.1.6. Cargador de historias desde fichero XML

BFG Toolkit permite la carga de historias desde fichero XML mediante el formato especificado en la sección 4.2.6 de este documento.

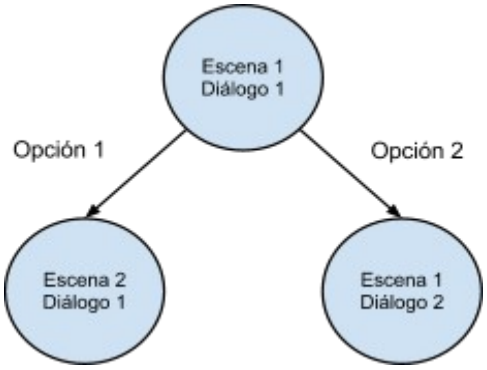


Figura 41. Diagrama diálogos en aventura

Estas historias se pueden representar mediante un árbol en el que cada rama es la secuencia de respuestas elegida por el usuario en cada diálogo. El juego se divide en escenas y cada una de ellas está constituida por un conjunto de personajes u objetos con los que el usuario puede interactuar. Dependiendo de las opciones escogidas se producirá la transición a una escena u otra de la historia.

Para interactuar en las diferentes pantallas, simplemente hay que realizar un evento de *scroll* para que se reproduzcan las distintas opciones por las que se va desplazando sobre la pantalla y un evento doble *tap* para seleccionar cada una de las mismas, tal y como se muestra en el juego desarrollado con esta funcionalidad en la sección 3.5.

4.2. Descripción a bajo nivel

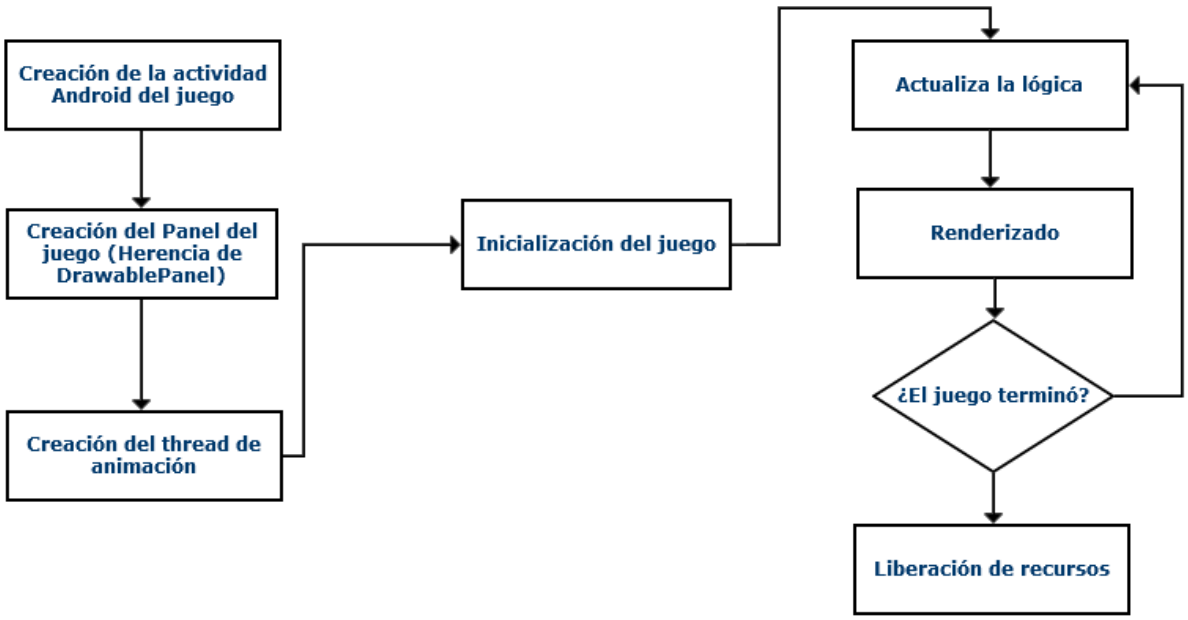


Figura 42. Diagrama funcionamiento básico bucle de juego

El sistema *BFG Toolkit* dispone de funcionalidad para que los desarrolladores puedan llevar a cabo sencillos juegos 2D. Para poder realizar esta tarea el *framework* sigue un flujo de ejecución clásico en juegos como se puede ver en la Figura 42. Estos pasos son a grandes rasgos:

1. Inicialización.
2. Comprobación de entrada. Actualización de la lógica.
3. Renderizado o dibujo de los elementos del juego.
4. Comprobación de terminación del juego. En caso negativo, volver a 2.

Existen algunos pasos adicionales en este caso debido a necesidades de la plataforma puesto que requiere de la implementación de ciertas clases para la definición de las vistas de la aplicación. Además, ya que toda aplicación en *Android* se basa en la interacción de una serie de clases que extienden la funcionalidad de una **actividad**, esta es una componente que proporciona una pantalla con la que el usuario puede interactuar para realizar una acción como hacer una foto, enviar un *e-mail* o en este caso interactuar con el menú principal o el juego en sí mismo, es necesario crear al menos una de ellas para cada aplicación. Cada una de estas actividades es apilada sobre las anteriores según se realizan las llamadas por la máquina virtual del sistema *Dalvik*. En el caso de los juegos desarrollados, se ha usado siempre una clase principal *MainActivity*, que presenta la vista del menú principal y gestiona el lanzamiento de las restante actividades:

- **GameActivity**: gestiona la vista y todos los elementos del juego que se describen en las siguientes secciones (*Game*, *GameStates*, Entidades, etc.).

- **InstructionsActivity**: muestra una descripción sobre cómo utilizar la aplicación.

- **AboutActivity**: muestra información sobre los desarrolladores de la aplicación.

- **SettingsActivity**: muestra un menú de configuración general de la aplicación. La posibilidad de habilitar ciertas funciones como el TTS, música o ciertas acciones específicas de cada juego, se encuentran en esta clase.

- **KeyConfActivity**: muestra el menú para la configuración de los controles de la aplicación.

- **FormActivity:** muestra el formulario de evaluación de la aplicación.

Dentro del conjunto de clases desarrolladas para facilitar la elaboración de cada uno de los juegos, se pueden distinguir 8 módulos que gestionan diferentes funcionalidades, que se detallan a continuación.

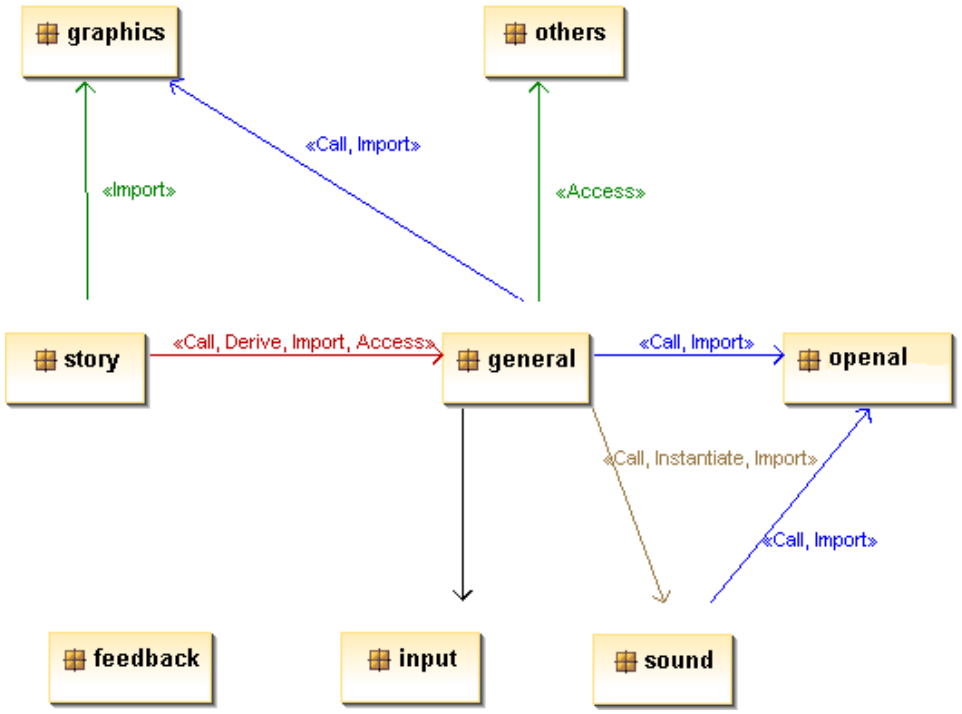


Figura 43. Dependencias entre paquetes

4.2.1. Módulo general

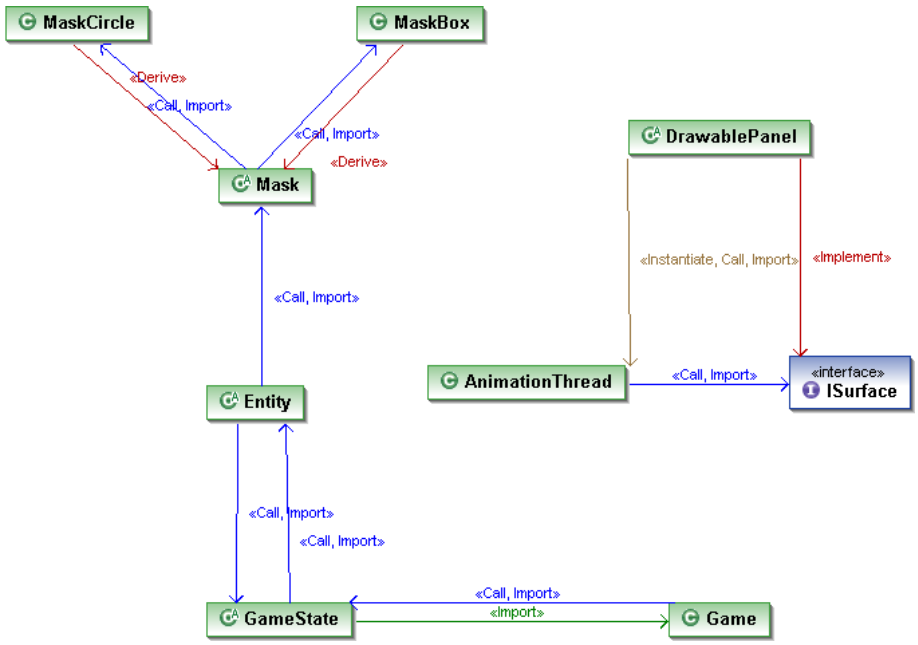


Figura 44. Diagrama de clases - módulo general

Game

Máquina de estados que gestiona cada acción de un juego. Requiere de la especificación de un conjunto de estados y el orden en que se desea su ejecución. A partir de ese momento se encarga de llevar a cabo los cambios de estado (*GameStates*), realizar el renderizado de los gráficos según el *gamestate* en el que se encuentre el juego, actualizar la entrada (tanto los eventos de teclado como los eventos sobre pantalla táctil) y controlar los frames por segundo (FPS) para que la rapidez de la ejecución de las aplicaciones no dependa de las características del dispositivo.

Su tarea principal es la ejecución de cada paso del juego, acción que será repetida hasta que el juego finalice. En cada paso se realizan las siguientes acciones:

- Comprueba la entrada.
- Actualiza el *gameState* y todas las entidades.
- Dibuja el *gameState* y todas las entidades.
- Control de FPS.

GameStates

Cada *gameState* mantiene todas sus entidades asociadas y se asegura de que interactúan entre sí correctamente. También tienen asociada una imagen de fondo.

Realiza de forma permanente las siguientes tareas:

1. En cada paso del juego, actualiza el estado de todas sus entidades y comprueba si existen colisiones entre ellas.
2. Posteriormente renderiza todas sus entidades.

Entidades

Una entidad es la máxima abstracción dentro de un juego. Tiene asociados una posición, una imagen estática o un *sprite sheet* para animaciones, una lista de fuentes de sonido 3D y conjunto de máscaras.

Las entidades son elementos independientes que actualizan su estado y se dibujan por solicitud del *gameState* que la instancia. Una entidad puede colisionar con otra o con elementos de un mapa. Sus acciones están controladas mediante un sistema de eventos. (*onStep*, *onTimer*, *onCollision*, etc.)

Máscaras

Para tratar las colisiones dentro de cada juego se ha utilizado un sistema de máscaras. Una máscara es un elemento abstracto que representa la física de una entidad. Una máscara puede tener forma geométrica de círculo o de paralelogramo. Cada entidad del juego tiene asociado un conjunto de máscaras para la representación de diferentes geometrías con combinaciones de las anteriores.

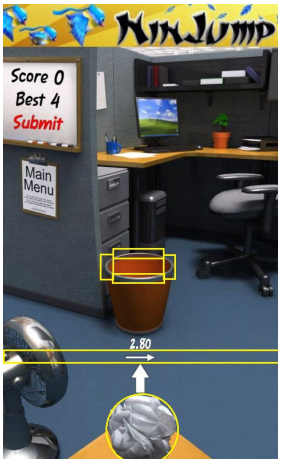


Figura 45. Máscaras juego Paper Toss - Android

Control de FPS

Esta funcionalidad asegura que el número de pasos del juego por segundo sea el indicado por el usuario.
Para conseguirlo, tras cada paso del juego (FPS) se pausa todo el sistema una cierta cantidad de tiempo hasta que el paso actual se consume. Si ese intervalo ya se ha gastado, no se produce ninguna pausa.

4.2.2. Sonido

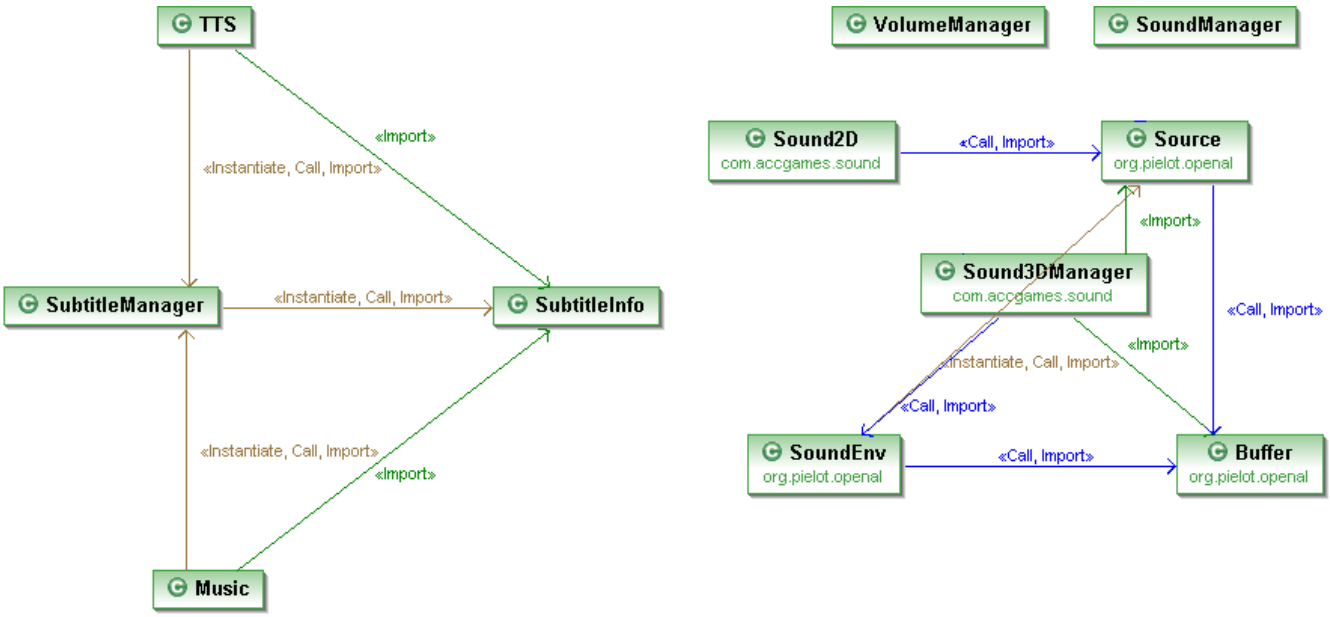


Figura 46. Diagrama de clases - Módulo de sonido

Sonido 3D

No existe ninguna librería *Android* que permita utilizar fuentes de sonido 3D de manera rápida. Es necesario utilizar código nativo escrito en C y mapear cada método a funciones java mediante la Java Native interface (ver sección 6.1) para, posteriormente, realizar las llamadas oportunas desde la aplicación. Para ello se han implementado una serie de clases puente entre C y Java utilizando JNI, apoyándose en una implementación en C de *OpenAL.*

Fijada la posición desde la que se está escuchando el sonido (posición del oyente) mediante estas clases se puede asociar a cada entidad un conjunto de fuentes de sonido 3D que permitirían determinar la dirección y gracias a la atenuación, la distancia a la que se pueden encontrar desde el oyente. Algunas de estas componentes han sido reutilizadas a partir de la librería *OpenAL4Android*, como son las clases que modelan las fuentes de sonido como *Source*, los *Buffer* asociados a éstas o la clase que gestiona la posición del oyente, la configuración y la colocación de las fuentes mediante llamadas a JNI, como es *SoundEnv*.

Efectos de sonido y música

En cada uno de los juegos, para poder realizar desplazamientos por los menús es necesario describir la interfaz mediante voz. Para ello se gestiona manualmente el motor de síntesis de voz que proporciona el sistema por defecto, mediante la clase TTS.

Dispone de funciones para la lectura, tanto de texto, como de objetos que extiendan la funcionalidad de la clase *View*. La música de fondo y los sonidos de los juegos están asociados a una clase de la librería que se basa en la encapsulación de un conjunto de instancias de la clase *MediaPlayer* que permite reproducir recursos sonoros en múltiples formatos (ver Anexo IV). Cabe destacar que esta funcionalidad es independiente del sistema de sonido 3D.

4.2.3. Entrada

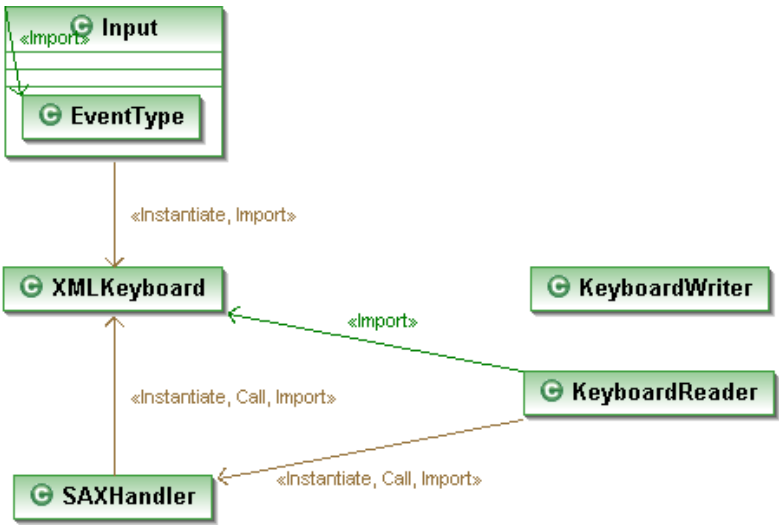


Figura 47. Diagrama de clases - módulo de entrada

Entrada estándar

El sistema de entrada es el encargado de mapear cada evento producido con su correspondiente vista en el juego. Las entidades consultarán este mapeado para actuar en consecuencia con el evento producido.

Estos eventos pueden ser tanto pulsaciones sobre la pantalla como teclas o combinaciones de gestos y teclas.

Configuración de teclado

Para hacer más personalizables los juegos, se introduce un sistema que permite asociar las acciones de cada juego a las teclas que el usuario considere oportunas. Para ello se realizan lecturas sobre un fichero XML que especifica la configuración de teclas y les asocia las acciones correspondientes.

Incorpora clases para realizar la escritura sobre este mismo fichero y para poder asociar las teclas que el dispositivo *Android* tenga disponibles a la funcionalidad que se desee de la aplicación.

4.2.4. Graphics

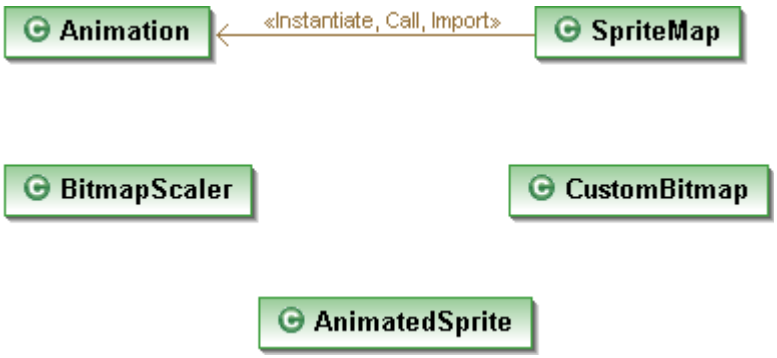


Figura 48. Diagrama de clases módulo de gráficos

Contienen toda la funcionalidad relacionada con imágenes estáticas o animaciones. También dispone de clases para escalado y manipulación de objetos de la clase *Bitmap*.

Toda entidad tiene que tener asociada una imagen estática o una animación para ser visualizada durante el juego. De cara a especificar una animación es necesario proporcionar un *sprite sheet* con cada uno de los *frames* (o pasos) de la animación y especificar en la instanciación del objeto de la clase *SpriteMap* asociado a la entidad, su animación mediante conjuntos de enteros que representarán la posición de cada uno de los *frames* dentro del *sprite sheet,* como se puede ver en la Figura 49.

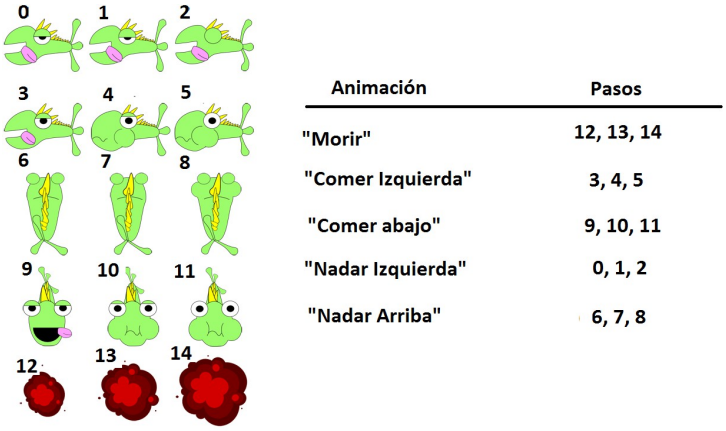


Figura 49. Animación

Tanto las animaciones como las imágenes estáticas pueden especificarse en múltiples formatos que se pueden ver en el Anexo IV.

4.2.5. Feedback y evaluación

Figura 50. Diagrama de clases - módulo de feedback

En este módulo se incluye un sistema de *Log* que se puede utilizar en cada uno de los juegos y que durante cada ejecución, permite anotan los "eventos" que han tenido lugar, junto con una serie de características de dichos juegos:

- *log*: cada vez que se ejecuta el prototipo, se construye el *log*. Puede tener, o no, dos tipos de hijos: *entry* y *event*. Como único atributo, la fecha en que se ejecuta el juego.
- *entry*: se crea cada vez que se empieza una partida durante la ejecución de la aplicación. Puede contener *event* y además tiene cuatro atributos que se refieren a la configuración del juego en esa partida en particular:
- *timeStamp*: impresión del tiempo en que se empieza la partida.
- *Configuration*: dificultad a la que se juega la partida (no es obligatorio).
- *event:* se trata de un evento en sí, cada vez que el usuario interactúa con la aplicación. Puede contener - o no - otro elemento, *comment*. Son cuatro los atributos que definen este elemento:
    - type: tipo de evento que se ha realizado. Un ejemplo puede ser pulsar un botón, hacer *tap* en algún lugar de la pantalla, etc.
    - timeStamp: momento en el que se ha realizado el evento.
    - comment: el elemento más primordial, la posibilidad de escribir algún pequeño comentario en un evento.

Este sistema de *Log*, utiliza la librería *Android* de *Google Analytics*, cuya funcionalidad principal se ha encapsulado a través de la clase *AnalyticsManager*. Para poder utilizar objetos de esta clase, basta con especificar un identificador de seguimiento proporcionado por el propio sitio web de *Google Analytics* y realizar las llamadas oportunas para el registro de eventos, páginas, variables (ver sección 6.2.3) de la API de *BFG Toolkit*.

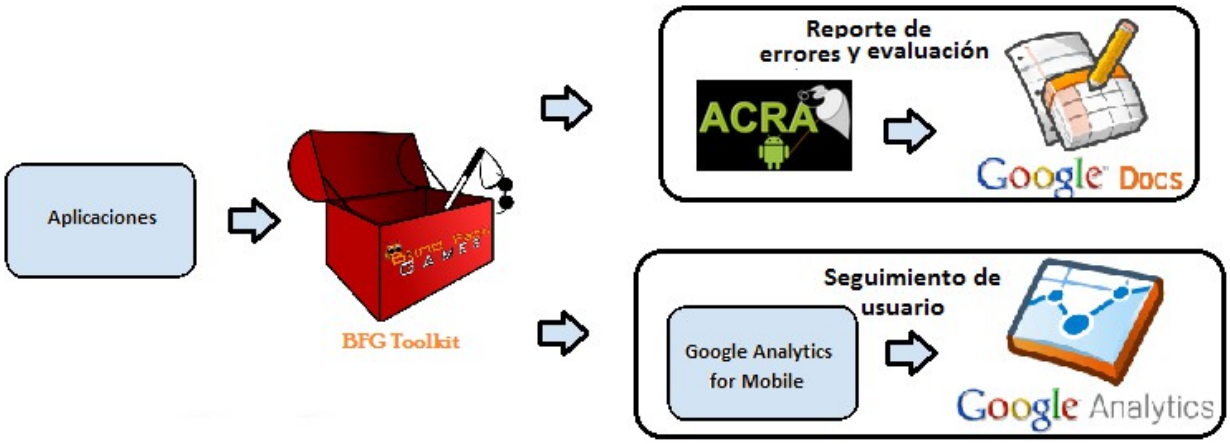


Figura 51. Diagrama conexión aplicaciones - Web

Este módulo también incorpora la librería *ACRA* (ver sección 6.4) para la notificación de excepciones en las aplicaciones que se han desarrollado con *BFG Toolkit*. La información se envía a una hoja de cálculo de *Google Docs* cuya identificación se debe especificar en *RuntimeConfig*. También se ha utilizado para el envío de los formularios de evaluación de las aplicaciones. Para poder utilizar esta funcionalidad se debe:

- Crear un formulario en *Google Docs*.
- Incluir la librería en el proyecto.
- Crear una clase que extienda de *Application*.
- Añadir el siguiente código en la clase, reemplazando el *formKey* por el obtenido del formulario creado en *Google Docs*.

```
import org.acra.*;
    import org.acra.annotation.*;

    @ReportsCrashes(formKey = "dGVacG0ydVHnaNHjRjVTUTEtb3FPWGc6MQ")
    public class MyApplication extends Application {
    }
```

- Sobreescribir el método *onCreate*.

```
@Override
    public void onCreate() {
        // The following line triggers the initialization of ACRA
        ACRA.init(this);
        super.onCreate();
    }
```

- Añadir el siguiente código en *AndroidManifest.xml* sustituyendo *MyApplication* por el nombre de la clase creada previamente.

```
<application android:icon="@drawable/icon" android:label="@string/app_name"
                android:name="MyApplication">
```

- Añadir los permisos necesarios en *AndroidManifest.xml*.

```
<uses-permission android:name="android.permission.INTERNET"></uses-permission>
```

4.2.6. Story

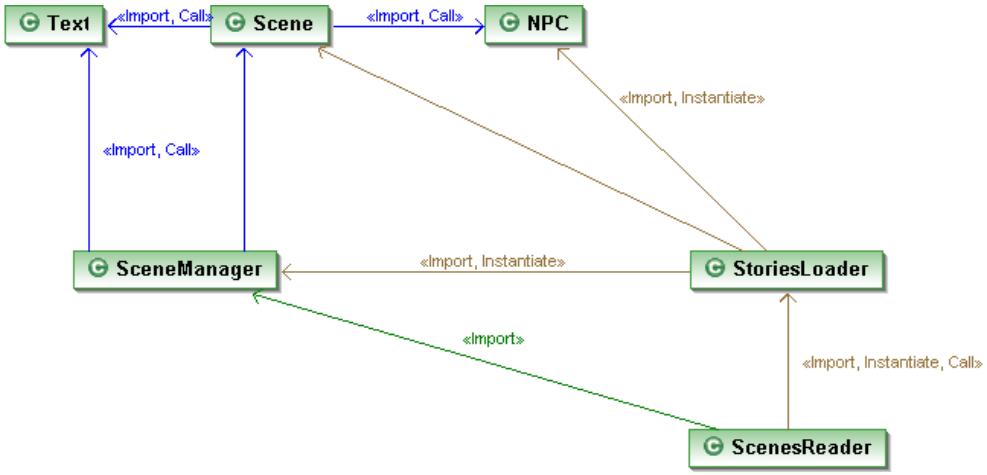


Figura 52. Diagrama de clases - módulo Story

Este módulo contiene una serie de clases que permiten cargar pequeñas aventuras basadas en diálogos, desde un fichero XML. Además, dispone de la funcionalidad necesaria para almacenar los datos obtenidos desde este fichero, mediante clases que se encargan de analizar la información y presentarla por pantalla en forma de texto.

El formato empleado es el siguiente:

```
<sceneManager>
	<scene type="normal" id = "41">
		<introMessage>
			Llegamos al seedy club tras el descubrimiento de la nota.
		</introMessage>

		<description>Seedy Club</description>

		<nextScenes>
			<idNextScenes id = "43"/>
		</nextScenes>

		<transitionCondition>
			<idTransitionCondition id = "36"/>
		</transitionCondition>

		<endCondition>
			<idEndCondition id = "46"/>
		</endCondition>
	</scene>
</sceneManager>
```

En esta estructura se dispone de un nodo raíz, *SceneManager*, que contiene todas las escenas del juego. Cada una de las escenas está definida por un identificador, usado para establecer las transiciones y las condiciones de inicio y fin de la misma, así como de un tipo que identifica la clase de escena (diálogo o minijuego). Aunque este último caso no pudo llegar a implementarse.

Para crear una escena se deben especificar las etiquetas *introMessage, description, nextScenes, transitionCondition y endCondition*, que se explican a continuación. En *introMessage*, se indica el texto que aparece como descripción de la escena, mientras que el texto contenido en *description* será aquel que se muestre como resumen de la opción. Es decir, en el ejemplo anterior, en un momento determinado se mostrarán distintas opciones que describirán lugares para visitar durante la aventura, por lo que en este caso, se mostrará la opción llamada *Seedy Club* junto con otras. En cambio, si en esta misma pantalla de selección se elige *Seedy Club*, se mostrará el texto contenido en *introMessage* una vez seleccionada ésta.

En cuanto a las etiquetas *nextScenes, transitionCondition y endCondition*: con la etiqueta *nextScenes*, se indica el conjunto de escenas que van a continuación de la especificada. Para cada escena que siga a la actual, se debe insertar una etiqueta *idNextScenes* y como atributo el número de su identificador. Con *transitionCondition*, se establecen las escenas que finalizaron previamente para llegar a esta. Para cada escena finalizada se debe insertar una etiqueta *idTransitionCondition*, acompañada de su identificador. Finalmente, con *endCondition*, se indican las escenas que deberán finalizar para que ésta llegue a su fin. Al igual que *transitionCondition y nextScenes*, se debe insertar una etiqueta *idEndCondition*, acompañada del identificador de la escena.

4.2.7. Others

Figura 53. Diagrama de clases - módulo Others

Incluye una serie de clases sin mucho peso con operaciones matemáticas básicas, generación de números aleatorios y una clase en la que se puede indicar ciertos valores de configuración para el motor en tiempo de ejecución, como los FPS.

4.3. Guía de uso para el programador

En esta sección se describirá cómo se ha de utilizar *BFG Toolkit* para implementar algunos de los juegos, con el objetivo de proporcionar una guía que facilite el aprendizaje a otros desarrolladores.

4.3.1. Introducción

Los cuatro pasos a seguir son los siguientes:

1. Entidades

Para poder comenzar a desarrollar un juego es necesario identificar cada uno de los elementos que lo constituyen y hacer que estos extiendan la clase entidad. Un objeto de la clase entidad puede tener como atributo su posición, una fuente de sonido 3D y su desplazamiento con respecto a la posición indicada por parámetro (ver API para más información).

Los métodos *onDraw* y *onUpdate* son los más importantes. En ellos se realizarán las acciones necesarias para que se pinten dibujos adicionales a los de la imagen estática o animada especificada por parámetro y se actualice la lógica de la entidad respectivamente.

Por otro lado, se dispone de otros métodos adicionales como *onCollision*, que permite definir qué acción realizar en caso de colisionar con otra entidad dentro de su ámbito de actuación.

También se da la posibilidad de sobrecargar el método *onTimer*, que permite definir una acción a realizar por la entidad tras un número determinado de pasos del juego. Para ello se activaría el temporizador con *Entity.setTimer (int timer, int steps);* y después se realizaría la distinción de casos en el método mencionado.

Finalmente se permite sobrecargar métodos para el guardado y cargado de información de entidades, para salvar el estado de la aplicación en caso de que el sistema decida realizar su liberación por falta de recursos para la ejecución de otras aplicaciones.

2. GameStates

A continuación, se debe identificar los estados del juego o *GameStates*, es decir los diferentes estados por los que puede pasar nuestra aplicación. Un conjunto de *GameStates* válido podría ser el siguiente: pantalla de inicio, introducción del juego, el juego en sí y fin de partida. Si fuera un juego por niveles cada nivel podría modelarse como un *GameState*. En definitiva, cada *GameState* delimita el ámbito de actuación de las entidades que se han creado en el paso anterior.

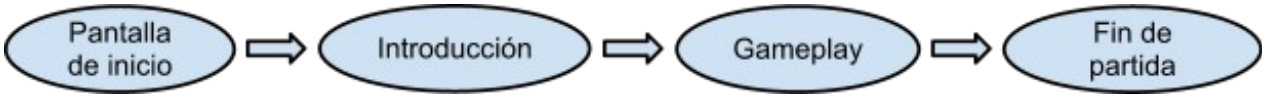


Figura 54. Diagrama con una estructura habitual de GameStates

3. Panel de la actividad

Es necesario definir la vista que se asociará a la actividad y sobre la que se realiza todo el renderizado de cada estado.

4. Crear el juego

Para finalizar, se instancia el juego especificando los estados de los que se compone y su orden relativo, dentro del método *onCreate* de una actividad cualquiera en la aplicación *Android*.

Se instancia la vista especificada previamente, se especifica al motor de síntesis de voz el contexto en el que actúa y una vez hecho esto se ha de instanciar la vista y el juego en sí.

Cada entero en orden, es la posición que ocupa dentro del vector de estados el estado que se desea que se ejecute y la posición de ese entero fija el orden de ejecución:

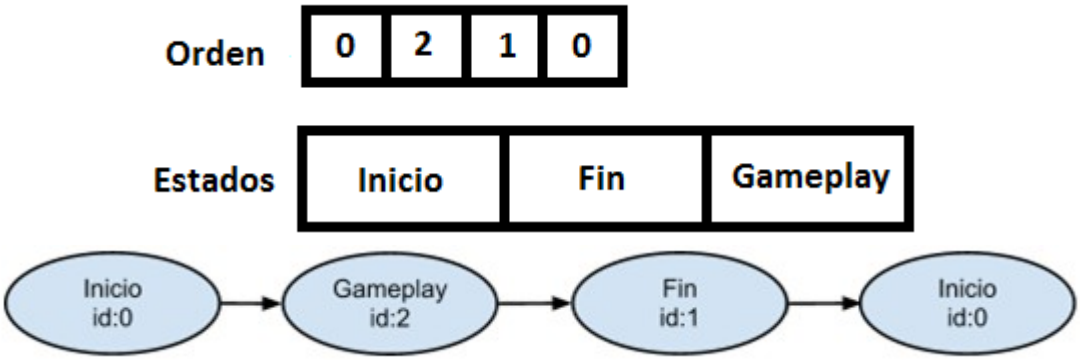


Figura 55. Estados y especificación de su orden de ejecución

Ejecuta Inicio, tras finalizar éste realiza *Gameplay*, a continuación Fin y en último lugar Inicio nuevamente.

4.3.2. Ejemplos. Zarodnik, Golf y The Shadow Of the Past

1. Golf

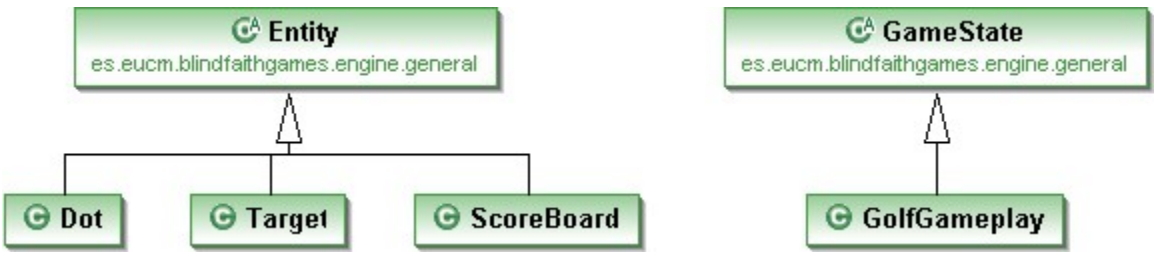


Figura 56. Diagrama de clases Golf

Como se puede observar en la Figura 56, en *Golf* se dispone de tres entidades que son el hoyo (*target*), la pelota (*dot*) y el marcador de puntuación (*ScoreBoard*). Todas ellas son instanciadas en *GolfGameplay*, que es un estado del juego que será **controlado** por la clase *Game* de *BFG Toolkit*.

2. Zarodnik

En el caso de *Zarodnik*, plantea un diagrama algo más complejo debido al creciente número de elementos en el juego.

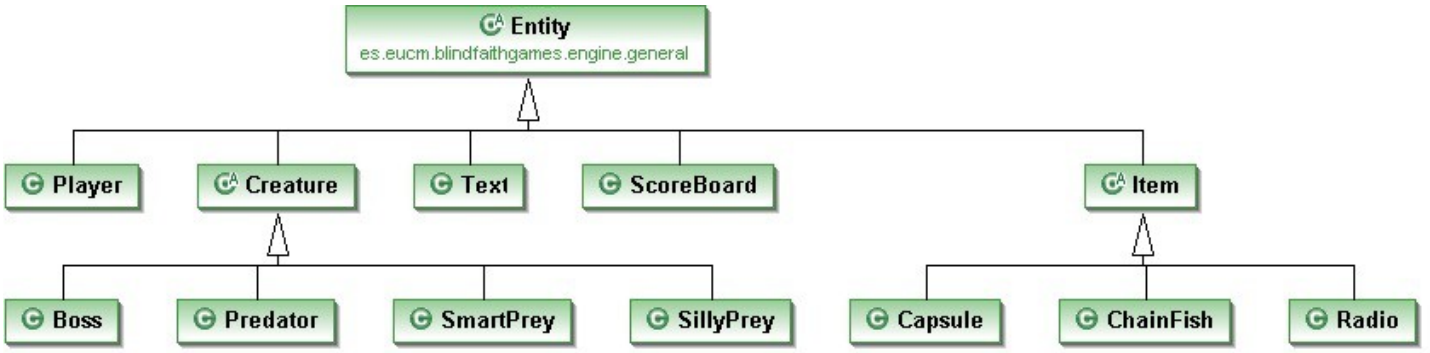


Figura 57. Diagrama de clases Zarodnik

Como se mencionaba anteriormente, se identifican las entidades del juego y para cada una de ellas se implementa una clase que extienda de entidad.

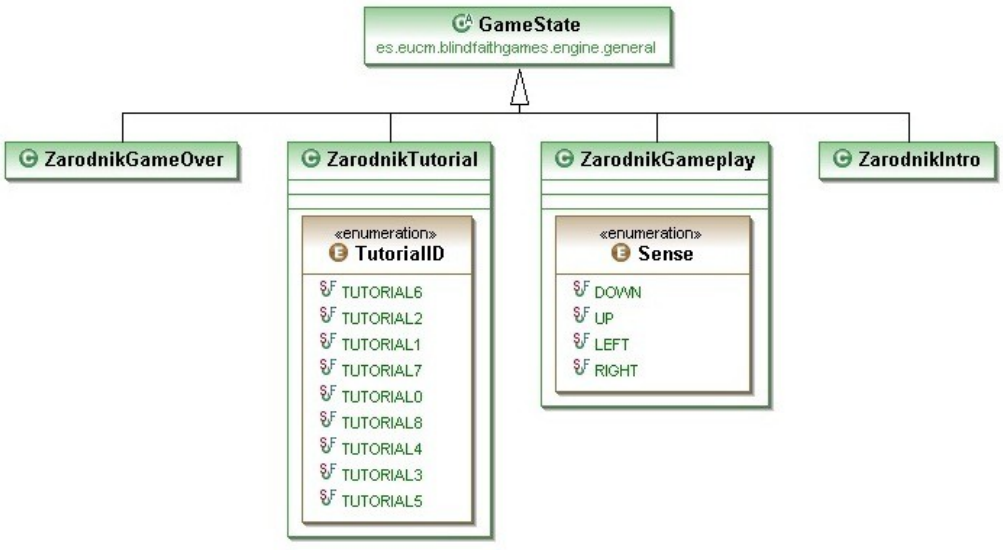


Figura 58. Diagrama de clases Zarodnik

Además, se definen los estados que va tener el juego. En cada estado se instancian las entidades que le correspondan al mismo. Por ejemplo, en *Gameplay* se instanciarán todas las entidades vinculadas con el juego en sí, como son el jugador, las presas y las criaturas.

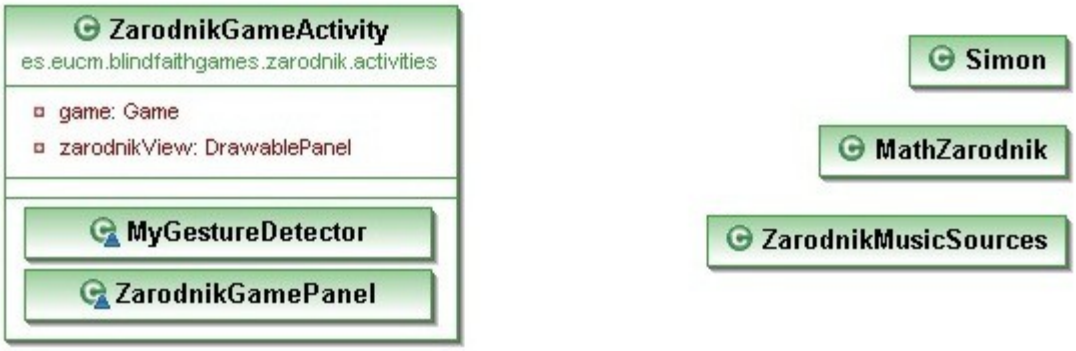


Figura 59. Diagrama de clases Zarodnik

A continuación, se define la vista sobre la que trabajará el juego y se asocia a la actividad *Android*.

Finalmente, sólo se tiene que crear el juego especificando el orden relativo, es decir, un diagrama de transición, entre los *GameStates*.

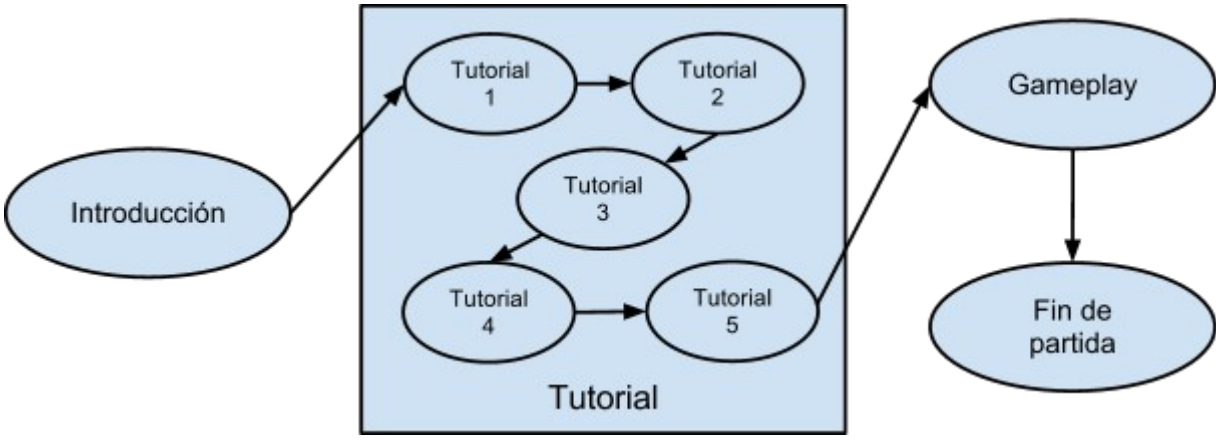


Figura 60. Diagrama de GameStates de Zarodnik

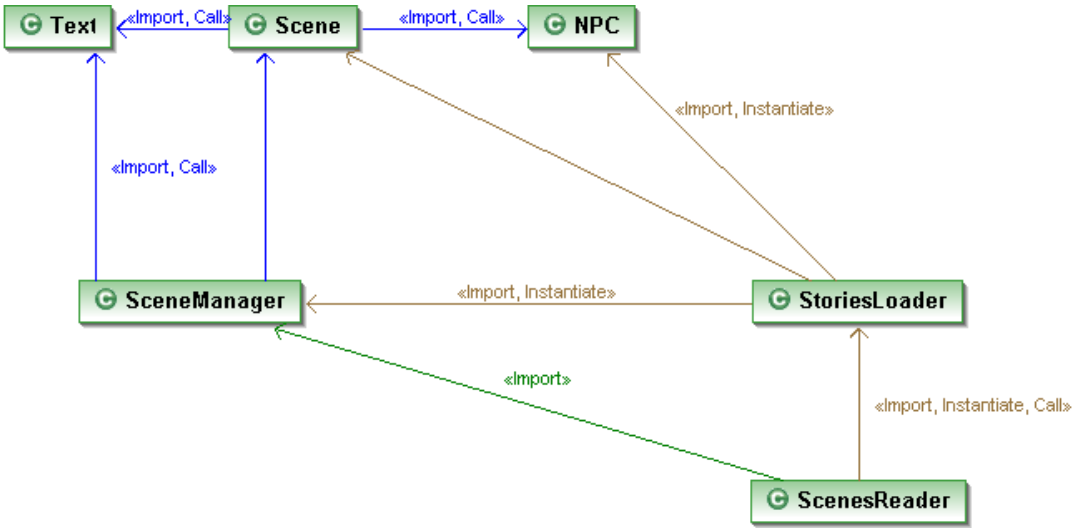


Figura 61. Diagrama de clases módulo *Story*

La principal novedad respecto a los juegos anteriores, es el uso del sistema de carga desde fichero XML del que dispone *BFG Toolkit*.

Para poder utilizar esta funcionalidad, es necesario obtener una instancia de la clase *SceneManager*. Para ello, hay que cargar la historia mediante la clase *SceneReader*. A continuación, únicamente se tiene que actualizar *SceneManager*, llamando a los métodos correspondientes en *onUpdate* y *onDraw* del *GameState* en el que se halla instanciado.

5. Proceso de desarrollo

En esta sección se describe el modelo de desarrollo empleado durante el proyecto, algunas de las herramientas utilizadas, como el IDE y una comparativa entre los distintos sistemas de versiones que se contemplaron como alternativas al comienzo del presente proyecto. También se introducen los principales riesgos planificados y analizados. Finalmente se concluye con la cronología del proyecto y unas métricas del mismo.

5.1. Metodología

5.1.1. Modelo de desarrollo de software

Para organizar el proyecto se decidió seguir una **metodología de desarrollo ágil**, basada en prototipado rápido, de perfil similar a ***SCRUM***.

SCRUM es una metodología para la gestión de proyectos que consiste en definir una serie de funcionalidades que se piden a una aplicación. Estas funcionalidades se dividen en tareas más sencillas que forman lo que se conoce con el término *sprint* o iteración. Una iteración consiste en marcar ciertos objetivos cada cierto tiempo, de forma que cada vez que se cumplan los plazos (habitualmente un mes), los objetivos marcados tengan que estar finalizados. Aunque a lo largo del proyecto se ha realizado alguna de las funcionalidades independientemente de la planificación, la mayoría de ellas se han llevado a cabo siguiendo un cierto orden.

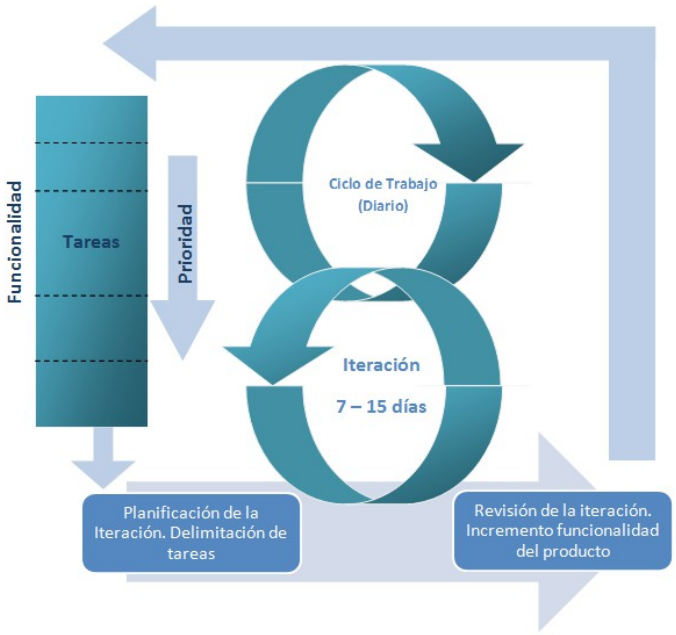


Figura 62. Diagrama proceso de desarrollo utilizado

El proceso de gestión y desarrollo del software definido en *SCRUM* ha sido adaptado de manera natural a las necesidades y posibilidades de los miembros del grupo. En este caso la planificación se define de la siguiente manera:

- **Trabajo diario y reuniones semanales** formadas por los integrantes del grupo de proyecto y los tutores del mismo, para establecer nuevos objetivos de cada *sprint* y revisar los hitos y entregas del anterior.
- **Desarrollo incremental del código y entrega de prototipos funcionales**.
- Presentaciones del producto a los directores del proyecto.
- **Los *daily meetings*** propios de *SCRUM* **no podían llevarse a cabo** de manera tradicional, ya que no se trabaja en el mismo lugar físico, ni los directores ni los desarrolladores del proyecto.
- **Dedicación a tiempo parcial al proyecto**: no se podían realizar estimaciones de tiempo sobre cada *funcionalidad / ticket* de los planificados en el *backlog*.

Esta metodología permite mantener flexibilidad en el desarrollo del producto gracias a los cambios realizados en las reuniones semanales y la continua comunicación entre los miembros del grupo.

De forma complementaria se han usado herramientas colaborativas que han ayudado a los integrantes del grupo a mantener una visión global del estado del proyecto en todo momento. Entre las herramientas se encuentran:

- Comunicación y planificación: *Trello*, *Google Docs* y correo electrónico
- Desarrollo de código: *Google Code*, *Git*
- Documentación: *Google Code Wiki*, *Google Docs*, *Microsoft Word*, *Mendeley*
- Almacenamiento de recursos: *Dropbox*

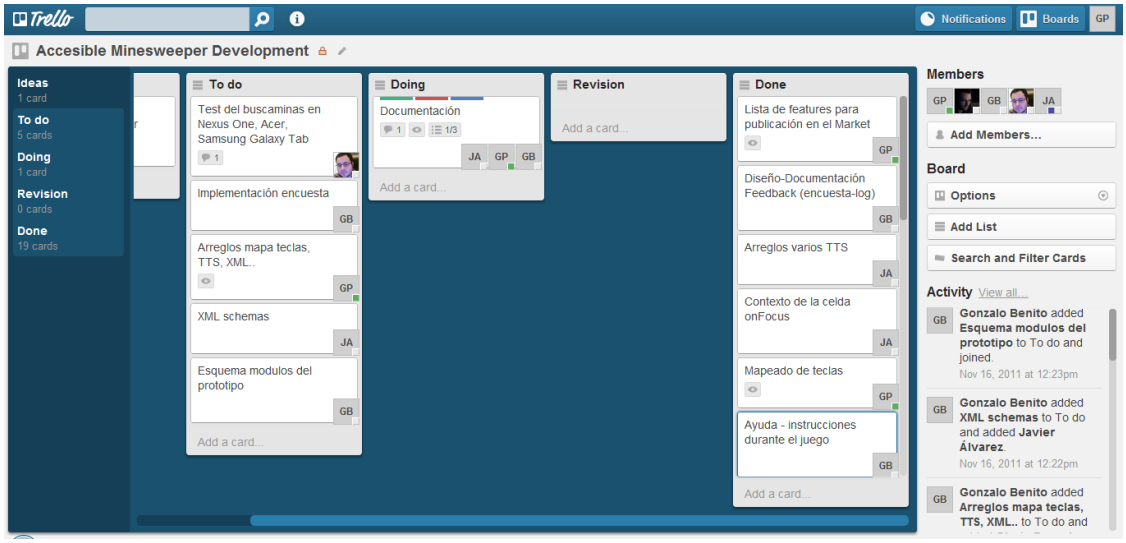


Figura 63. Herramienta Trello

5.1.2. Entorno de desarrollo

Para llevar a cabo el desarrollo del proyecto, se ha utilizado como Entorno de Desarrollo Integrado (IDE) *Eclipse Indigo* 3.7.2. *Eclipse* es un entorno de desarrollo compatible con múltiples lenguajes de programación basado en Java que permite la integración de distintos *plugin*.

En concreto se ha hecho uso del *plugin* ADT (*Android Development Tools*) para *Eclipse*, necesario para el desarrollo de aplicaciones *Android*. Este *plugin* amplía las capacidades de *Eclipse* para poder configurar rápidamente nuevos proyectos *Android*, depurar las aplicaciones utilizando la herramienta del *SDK de Android*, exportar las aplicaciones firmadas con el fin de distribuirlas, etc.

5.1.3. Control de versiones

5.1.3.1. Introducción

Dada la envergadura del proyecto era necesario hacer uso de un repositorio de código, así como de un sistema de control de versiones que permitiera ir avanzando de una manera sencilla y ordenada.

En un primer lugar se decidió utilizar un sistema de alojamiento de proyectos que ofrece *Google* denominado *Google Code*, que está destinado para desarrolladores que llevan a cabo proyectos *Open Source*. Su selección se llevó a cabo por los siguientes aspectos:

- Es **gratuito**. No es necesaria confirmación ni requiere ninguna espera para abrir el proyecto.
- **Sistema fácil de configurar**, con soporte para organizar la documentación del proyecto.
- **Compatible con varios sistemas de control de versiones** - *Subversion*, *Mercurial*, *Git* -.
- **Otros muchos proyectos de *Android*,** y más en concreto de accesibilidad para *Android* - por ejemplo *eyes-free* - **estaban alojados en *Google code***.
- Varios componentes del grupo tenían **conocimientos previos del servicio**.

Se decidió hacer **uso de *Subversion***, conocido como SVN, ya que se disponía de experiencia previa con el mismo. Se han empleado una serie de herramientas para trabajar con este sistema de alojamiento: *TortoiseSVN* (*Windows*), *RabbitSVN* (*Linux*) y el *plugin* de *eclipse* para *Subversion* (*Subclipse*).

5.1.3.2. Problemas y cambio de sistema de versiones

La naturaleza altamente experimental del proyecto requería de un flujo de trabajo dinámico que permitiera experimentar con distintas soluciones de accesibilidad de manera simultánea y constantemente susceptible al cambio. Subversion no satisfizo estos requisitos. Los principales problemas que se experimentaron fueron: ramas en las que no se podían hacer *commits*, desaparición del código alojado en la rama principal - *trunk* - e incompatibilidades con *Subclipse*.

Las dos alternativas más prometedoras del momento eran: *Git* y *Mercurial.*

La diferencia más notable entre los sistemas de versiones centralizados - como *Subversion* - y los segundos es que en los centralizados la operación *commit* se realiza directamente en el servidor, mientras que en los distribuidos se realiza localmente (se puede hacer varios *commits* antes de mandar los cambios al servidor).

5.1.3.3. Comparación de características y decisión

Tanto *Git* como *Mercurial* son DCVS (Sistemas de Control de Versiones Distribuidos), los cuales permiten trabajar de forma local, es decir, cada usuario tiene su propio repositorio y los distintos repositorios pueden intercambiar y mezclar revisiones entre ellos. Además, los dos están pensados para ser utilizados a través de consola, disponen de *plugin* para *Eclipse* y son compatibles con aplicaciones como *Tortoise* (*Windows*) o *Rabbit* (*Linux*). Ahorran espacio de almacenamiento - un mismo repositorio en *SVN* ocupa más espacio que en *Mercurial*, y sobre todo que en *Git* -. Por último no permiten hacer *Checkout* parcial como SVN - por ejemplo sólo de *HEAD* o de la revisión N, sino que clona todo el repositorio.

A continuación se exponen las principales diferencias:

GIT	**Mercurial**
Plugin para *Eclipse* que está desarrollado por un Team Provider de *Eclipse* y es propio de *Eclipse.org.*	El *plugin* para *Eclipse* está desarrollado por *Selenic*, al igual que el propio *Mercurial.*
Cada commit se identifica con un UUID - identificador único universal.	Identificadores incrementales, al estilo *Subversion.*
Pensado para trabajar con ramas, muy flexible en ese sentido (por ejemplo: permite hacer *merge* de varios padres, ramas locales independientes entre sí, etc.).	Hacer una rama es más sencillo que en SVN, aunque no tanto como en Git, además de no tener tantas posibilidades.
Documentación más escasa, los comandos de consola son más y diferentes a SVN. (aunque la interfaz de línea de comandos se ha unificado considerablemente en las últimas versiones, haciéndolo más accesible)	Más fácil de aprender (documentación más completa, comandos parecidos a SVN y menor cantidad).

Tabla 7. Comparativa GIT y Mercurial

Lo cierto es que se trata de dos sistemas muy parecidos, dado que las últimas versiones de *Git* se han vuelto mucho más fáciles de utilizar. Aunque en la actualidad *Mercurial* sigue teniendo una mayor implantación, *Git* es el sistema que mayor crecimiento ha tenido en los últimos tiempos, gracias en parte al impulso que le dio *Google* al incluirlo entre los sistemas de control de versiones compatibles con su plataforma *Google Code.*

Por todo ello finalmente se ha elegido la opción de *Git* y adoptar un sistema de trabajo en el repositorio como sigue:

- Rama principal (*master*): versiones estables con nuevas funcionalidades completadas. Uso de *Tags* para tener siempre accesibles las versiones que se deseen.
- Ramas de desarrollo (*develop1*, *develop2*, *develop3*): lugar donde se van acumulando las versiones en pleno desarrollo, no terminadas o a falta de depuración y pruebas.

5.2. Análisis de riesgos

Aunque la metodología de trabajo era fundamentalmente ágil, se optó por hacer un análisis de riesgos inicial como salvaguarda, frente a la falta de experiencia en desarrollo de aplicaciones accesibles de los miembros del proyecto.

En primer lugar, se analizaron los **riesgos derivados de la realización de proyectos** en general:

1. **Poca implicación de alguno de los miembros del grupo.**

Probabilidad: baja.
Impacto: muy alto.
Descripción: existe la posibilidad de que alguno de los miembros del grupo sufra algún contratiempo que le impida trabajar en el proyecto o directamente decida abandonarlo.
Medidas: No se puede anticipar esta situación de ninguna manera, tan sólo se puede redistribuir el trabajo en caso de que se produzca.
Evolución: el proyecto se inició con 3 miembros, uno de ellos lo abandonó entorno al segundo hito. Luego se puede considerar que el riesgo se produjo y estuvo presente a lo largo de todo el proyecto.

En segundo lugar, se analizaron los **riesgos derivados del uso de la tecnología** *Android*:

1. **La posible ausencia de fundamentos básicos para la implementación de juegos accesibles.**

Probabilidad: baja.
Impacto: muy alto.
Descripción: ninguno de los miembros del grupo ha desarrollado *software* accesible en general, ni ha implementado juegos accesibles.
Medidas: mayor documentación por parte de los miembros del proyecto.
Evolución: puesto que la complejidad de los juegos no es elevada, este riesgo no ha tenido ninguna evolución significativa desde los primeros *sprints*.

2. **Imposibilidad de llegar al público objetivo.**

Probabilidad: alta.

Impacto: bajo.

Descripción: las personas con discapacidad visual podrían no estar interesados en una plataforma con tan escaso apoyo a su sector hasta el momento como es *Android.*

Medidas: lo único que se podía hacer es tratar de tener más repercusión en la comunidad, un mayor esfuerzo en la divulgación del trabajo.

Evolución: se puede considerar que este riesgo se ha producido desde el comienzo del proyecto, ya que la plataforma *Android*, como se trata en buena parte de las secciones de este documento, hasta el momento no es la más accesible para usuarios ciegos.

3. **Problemas de estabilidad con nuevas versiones del sistema *Android.***

Probabilidad: media.

Impacto: medio.

Descripción: es un sistema en constante evolución, por lo que tanto la API como su funcionalidad están sujetos a cambiar.

Medidas: instalación en varios dispositivos con distintas versiones.

Evolución: probablemente tuvo su momento de alto impacto coincidiendo con los *sprints* previos a la publicación de las aplicaciones en *Google Play,* puesto que se liberó la versión de *Android* ICS, para la cual surgieron problemas de compatibilidad con *BFG Toolkit* que se subsanaron con actualizaciones de las aplicaciones. A fecha de redacción de este documento está solventado.

4. **Insuficiente variedad de dispositivos para realizar depuración y obtener versiones estables.**

Probabilidad: alta.

Impacto: medio.

Descripción: inestabilidad de las aplicaciones ante el elevado número de dispositivos y combinaciones con diferentes versiones del sistema operativo *Android*.

Medidas: lanzamiento de versiones beta que informen sobre los errores más significativos.

Evolución: probablemente tuvo su momento de mayor impacto coincidiendo los *sprints* previos a la publicación de las aplicaciones en *Google Play*. Tras superar los errores más graves durante la publicación se puede considerar resuelto.

5. **Framework no es lo suficientemente estable para desarrollar los juegos.**

Probabilidad: baja.

Impacto: alto.

Descripción: el uso de un motor propio en lugar de uno profesional, puede constituir un problema en el desarrollo de las aplicaciones, si éste no es lo suficientemente robusto.

Medidas: inversión de mayor tiempo en su depuración antes de desarrollar más aplicaciones.

Evolución: se superó una vez se dispuso de la funcionalidad básica alcanzado el segundo hito.

6. **Ausencia de estrategias de diseño y el estudio sobre el desarrollo de juegos accesibles en móviles.**

Probabilidad: alta.

Impacto: alto.

Descripción: ninguno de los miembros tiene experiencia en el diseño de características de accesibilidad en *software* o juegos.

Medidas: esfuerzo por documentarse por parte de los miembros y mayor frecuencia en las consultas a los directores de proyecto.

Evolución: la evaluación continua por los directores de proyecto, con experiencia en materia de accesibilidad y usabilidad, sirvió de ayuda para cada uno de los desarrollos. Por ello, podemos considerar que este riesgo no se ha producido desde los primeros *sprints*.

7. **La API de *Android* no proporciona la funcionalidad necesaria para realizar el desarrollo de *BFG Toolkit*.**

Probabilidad: media.

Impacto: bajo.

Descripción: cada aplicación, dependiendo de la versión del sistema *Android*, dispone de una cantidad de memoria limitada para los recursos que ésta utiliza, tanto de sonido como imágenes. Las dificultades en la gestión de la memoria por parte de *BFG Toolkit*, en su uso de las clases de la API, podrían hacer muy difícil el desarrollo de las aplicaciones en el proyecto.

Medidas: siempre fue posible recurrir a librerías externas o elaborar nuestra propia funcionalidad.

Evolución: el riesgo estuvo presente hasta los últimos *sprints* del proyecto, puesto que *BFG Toolkit* estuvo en continua evolución a lo largo de todo el proyecto.

5.3. Cronología del proyecto

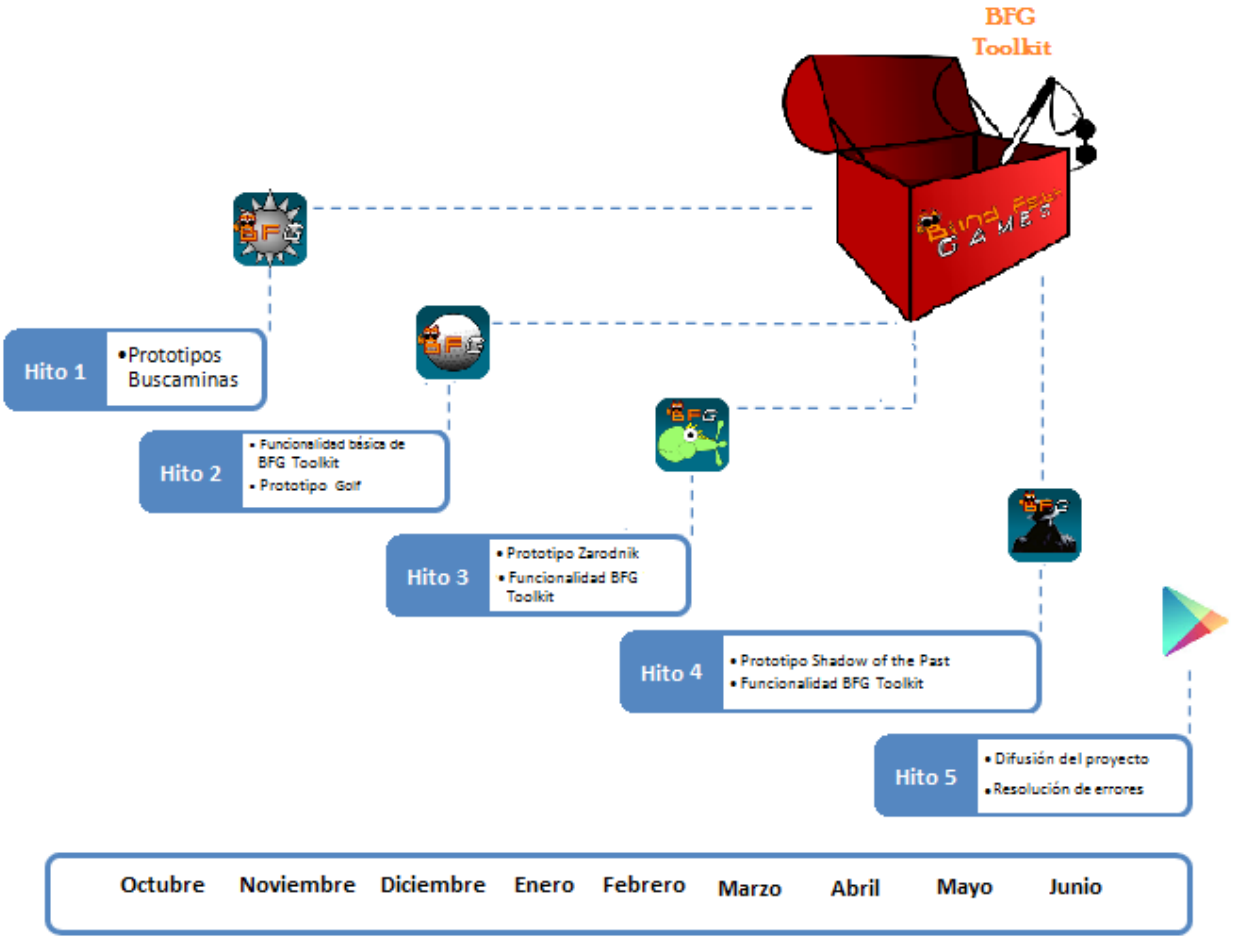


Figura 64. Representación gráfica de los hitos del proyecto

5.3.1. Primer hito

En esta etapa se tienen como principales hitos el diseño junto con el correspondiente desarrollo de varias versiones de la aplicación *Buscaminas Accesible*.

Sprint I 20-10-11

En este primer *sprint* se tuvo que llevar a cabo una primera toma de contacto con toda la tecnología entorno a la cual iba a girar todo el proyecto: *Android*. Por ello, se comenzó con el desarrollo de una aplicación sencilla como es un buscaminas.

Sprint II 27-10-11

Se mejoró la versión de buscaminas ya existente, ofreciendo un mejor aspecto visual de los menús y se realizó un estudio sobre los diferentes sintetizadores de voz, así como su viabilidad a la hora de hacer que el usuario instalara alguno de ellos. También se cambió el sistema de versiones para el repositorio. Se pasó de SVN a GIT.

Sprint III 03-11-11

Se diseñaron e implementaron varias versiones del buscaminas con diferentes opciones de accesibilidad más configurables, partiendo de la versión del primer *sprint* (ver sección 3.2). También se probó la aplicación sobre diferentes dispositivos y se llevó a cabo la documentación de las pruebas y diseños.

5.3.2. Segundo hito

Como principal hito en esta etapa se tiene la funcionalidad básica de *BFG Toolkit* así como un segundo juego, el *Golf*.

Sprint IV 10-11-11

Se comenzó a realizar el segundo juego. Para ello se plantearon diferentes alternativas a la hora de implementarlo. Podía utilizarse un motor de juegos exclusivo *Android* como *AndEngine*, pero ante la falta de documentación y soporte, además del riesgo que podía suponer que alguna de las características de accesibilidad no pudiera ser integrada, se descartó. Finalmente se optó por llevar a cabo un pequeño motor o framework, *BFG Toolkit*, que facilitó llevar a cabo los futuros desarrollos y que serviría como otro elemento más a incorporar al proyecto.

En este *sprint* se realizó el diseño del conjunto de herramientas antes mencionado, *BFG Tookit*. También se llevó a cabo la incorporación de características adicionales para el buscaminas, como un sistema de configuración de teclas, de forma que el usuario pudiera asociar a una funcionalidad, como por ejemplo la lectura de casilla, una determinada tecla de su dispositivo móvil.

Sprint V 17-11-11

Sprint en el que se realizó la evaluación y cierre de la primera versión funcional de *BFG Toolkit*. Así mismo, se diseña el funcionamiento de *Golf*, tanto desde un punto de vista de la mecánica de juego, como de las opciones de accesibilidad que éste dispondría.

Sprint VI 22-11-11

Se diseña e implementa el sistema de entrada para *BFG Toolkit*. Se comienza a investigar la forma de utilizar sonido 3D en *Android*.

Sprint VII 01-12-11

Durante este tiempo se experimentaron distintas alternativas a la hora de realizar el seguimiento del uso que hacían los usuarios de los juegos, así como de los formularios que les da la oportunidad de expresar su opinión acerca de las aplicaciones, con el objetivo de mejorarlas. Se implementó la alternativa *App Engine* en *BFG Toolkit* que posteriormente sería reemplazado por *Google Analytics for Android* y *ACRA*, debido a problemas de identificación de usuarios (ver sección 6.2).

Sprint VIII 14-12-11

Implementación de la aplicación *Golf*, una vez alcanzada una versión estable de *BFG Toolkit*. Además, se incorpora al *Buscaminas Accesible* alguna funcionalidad adicional.

También se estudia la posibilidad de utilizar una web para procesar los datos del sistema de seguimiento y los formularios recibidos, mostrándolos en forma de gráficos utilizando *GWT (Google Web Toolkit)* y *Google Charts*.

Se continuó con la implementación del sistema *App Engine* en *BFG Toolkit* para la recepción de *feedback* por parte del usuario.

Sprint IX 20-12-11

Depuración y mejora de la aplicación *Golf* tanto de forma visual, como en funcionalidad y sonidos. Búsqueda de recursos de sonido, así como edición o creación de algunos de ellos.

Evaluación del sistema *App Engine*. Se decide que es necesaria una modificación en el mecanismo de envío, puesto que se requería el uso de correo personal por parte del usuario para el envío de información.

Sprint X 02-01-12

Diseño y desarrollo de los recursos gráficos de *Golf*. Se finaliza la implementación del sistema *App Engine* para el envío de información desde las aplicaciones de forma anónima. También se llevó a cabo la implementación del cuestionario de evaluación en todas y cada una de las aplicaciones.

Sprint XI 12-01-12

Estudio de la API de *Google Docs* para su aplicación a la hora de almacenar información desde los juegos y para su posterior representación en forma de gráficos.

Además, se desarrolló una web utilizando *GWT* como visor para los datos de usuario en *App Engine*.

5.3.3. Tercer hito

El principal hito es el uso de fuentes de sonido 3D por parte del motor, así como el diseño y desarrollo de la tercera aplicación, *Zarodnik*.

Sprint XII 19-01-12

Comienzan a diseñarse los elementos de accesibilidad del tercer juego, inicialmente un *Snake*, que acabaría evolucionando en *Zarodnik*. Realización de una demostración técnica para evaluación del funcionamiento del sonido 3D en Android. Empieza a diseñarse la incorporación a *BFG Toolkit* del módulo de sonido 3D.

Sprint XIII 02-02-12

Identificación de los recursos de cada aplicación en términos de licencia, tanto aquellos propios como aquellos de terceros. Estudio de la forma de conectar en tiempo real un dispositivo *Android* a una pantalla de cara a realizar presentaciones. Además fue necesario revisar la media de transmisión de datos de las aplicaciones.

Sprint XIV 09-02-12

Revisión del contenido de los formularios. Además, se estudió la posibilidad de incorporar una librería para la notificación de errores en las aplicaciones.

Sprint XV 20-02-12

Se añaden tutoriales para reducir la curva de aprendizaje de cada una de las aplicaciones. Al mismo tiempo comienza a plantearse el desarrollo de un cuarto juego.

Sprint XVI 27-02-12

Para finalizar este hito, se incorporó un modo “pantalla en negro” a todas las aplicaciones, con el objetivo de transmitir a gente sin problemas de visión las sensaciones de una persona con discapacidad visual a la hora de jugar a juegos con estas características.

También se estudiaron diferentes formas de utilizar *Google Docs* para el almacenamiento de *feedback* que proporcionaran futuros usuarios.

Además se incorporó cierta funcionalidad adicional en el resto de las aplicaciones.

5.3.4. Cuarto hito

Diseño de la cuarta aplicación *The Shadow of the Past* y refinamiento de las aplicaciones anteriores para su publicación en *Google Play*.

Sprint XVII 05-03-12

Se comenzó a buscar referencias de cara a realizar un cuarto juego basado en historia o aventura textual. Debía permitir al usuario dialogar con los elementos del mismo, lo que determinaría qué rama de la historia seguir. Podría activar ciertos minijuegos, cambiar el desenlace de la trama, tal y como se explica en la sección 3.5 de este documento.

Se consideraron dos alternativas, adaptar una historia ya conocida de un libro, película o videojuego, o bien tratar de inventar una sencilla historia. Así pues, se comenzó a tratar de adaptar el capítulo I del conocido juego *The Monkey Island*.

Sprint XVIII 12-03-12

Se llevó a cabo un pequeño prototipo a partir de la adaptación de *The Monkey Island* con grandes dificultades debido al tamaño de la historia de este tipo de títulos. Éste permitiría añadir a *BFG Toolkit* funcionalidad para poder cargar historias muy simples desde ficheros XML.

Por otro lado, durante este *sprint*, también se portaron los juegos a *BlackBerry OS* 2.0 para *Playbook* con el objetivo de probar otras plataformas.

Sprint XIX 26-03-12

En esta etapa se comenzó a diseñar el cuarto juego, como se había considerado en anteriores *sprints*, una aventura textual. Se llevó a cabo un pequeño prototipo a partir de una historia no original y que podría ser la base para la versión final realizada en Mayo.

Al mismo tiempo, se pasaron el resto de aplicaciones a la última fase de producción con el objetivo de alcanzar la calidad suficiente para su publicación en *Google Play* a principios de Mayo.

Sprint XX 26-03-12

A lo largo de este *sprint* y las siguientes, se añadió funcionalidad común a todas las aplicaciones, como la transcripción de voz y sonido. Esta nueva característica estaba dirigida a mejorar la presentación del proyecto y también se finalizó la implementación del modo "pantalla en negro".

También se llevaron a cabo todas las tareas de localización de los juegos, es decir, su traducción al inglés de cara a su publicación en *Google Play*.

Sprint XXI 05-04-12

Se elaboraron los contenidos iniciales de las páginas web de los juegos, además de incorporar una nueva forma de interacción más cómoda para personas sin discapacidad visual en los menús de todos los juegos.

Sprint XXII 12-04-12/26-04-12

Este *sprint* se dedicó a realizar mejoras en la funcionalidad y en los recursos de cada una de las aplicaciones, a sugerencia de diferentes *testers*, de cara a la publicación final de las mismas en *Google Play*.

5.3.5. Quinto hito

Durante esta etapa se finalizó el desarrollo de los juegos y se realizaron las últimas modificaciones. También se implementó la versión final de *The Shadow of the Past*.

Sprint XXIII 10-05-12

Este *sprint* se dedicó prácticamente en su totalidad a la corrección de informes de errores recibidos desde las aplicaciones en *Google Play*.

Además, una vez diseñada la línea básica a seguir en la historia del cuarto juego, se decidió comenzar a escribir los diálogos del mismo.

Sprint XXIV 17-05-12

Además de continuar con la resolución de los problemas notificados que se recibieron en el informe de error tras la publicación de los juegos, se estudió nuevamente la forma de volcar en tiempo real, con una frecuencia de refresco aceptable, la salida por pantalla de un dispositivo *Android*. Así mismo, se grabaron vídeos promocionales de los juegos para la web y se realizó la traducción de ésta última.

Sprint XXV 24-05-12

Se reescribieron parte de los diálogos de *The Shadow of the Past* de cara a una posible publicación. También se realizó una refactorización de algunos módulos de la herramienta *BFG Toolkit*.

Sprint XXVI 31-05-12/14-06/12

Este último *sprint* se dedicó a la difusión del proyecto de evaluación tanto en comunidades y organizaciones expertas en accesibilidad, como en foros de jugadores ciegos.

5.4. Métricas del proyecto

Para dar una idea de las dimensiones del proyecto estas son algunas de las métricas en términos de líneas de código y número de clases

	LDC			Clases
	Comentarios Java	Código Java	Código XML	Java
BFG Toolkit	2711	6285	50	41
Buscaminas Accesible	1072	4763	1441	21
Golf Accesible	1414	4273	1521	18
Zarodnik	1200	5372	1063	28
Shadow of the Past	650	2018	1715	11
Total	8119	28501		119

Tabla 8. Métricas[23]

[23] Obtenidas mediante jHawk: http://www.virtualmachinery.com/jhawkprod.htm

6. Tecnologías utilizadas

En esta sección se exponen todas las tecnologías que se han empleado a lo largo del proyecto, el porqué de su uso y en algunos casos las alternativas que se probaron pero finalmente se descartaron.

6.1. OpenAL y JNI

Para poder orientar al jugador dentro de los juegos, dónde se encuentra el propio protagonista y notificar la posición y la distancia a la que está de estos elementos, se necesitaba alguna herramienta que facilite la gestión de este API de sonido.

6.1.1. Descripción

OpenAL es una API de audio multiplataforma desarrollada para el renderizado eficiente de audio posicional y multicanal en tres dimensiones. Es una de las principales especificaciones de referencia en audio. Está ideada para su uso en videojuegos y el estilo y convenciones del código es similar al de OpenGL[24] (API generalizada para gráficos).

OpenAL utiliza un mecanismo basado en extensiones. Cada cual puede incluir sus propias extensiones en la distribución de OpenAL, algo frecuente para agregar funcionalidades de hardware propietario. Las extensiones pueden ser promocionadas a un status ARB (Architecture Review Board), indicando una extensión estándar que será mantenida con compatibilidad hacia atrás. Las extensiones ARB pueden ser agregadas al núcleo de la API tras un cierto tiempo.

El funcionamiento global de OpenAL se puede dividir en objetos fuente, oyentes y *buffers* de audio. Un objeto fuente contiene un puntero a un *buffer*, además de una serie de atributos como la velocidad, posición, dirección o intensidad del emisor de sonido. Un oyente contiene información sobre la velocidad, posición y orientación del sistema de referencia, además de la ganancia general aplicada a todo sonido. Sólo puede haber un oyente.

En cuanto a JNI es un framework de programación que permite que un programa escrito en Java ejecutado en la Máquina Virtual Java (JVM), pueda interactuar con programas escritos en otros lenguajes como C, C++ y ensamblador.

Dado que en *Android* no existía ninguna herramienta que proporcionase sonido 3D para juegos de forma nativa se decidió comenzar a buscar librerías que ya estuvieran implementadas por otros desarrolladores para

[24] http://www.opengl.org/

facilitar el uso de OpenAL. Así pues, se encontró una librería denominada *OpenAL4Android*[25], la cual proporcionaba la funcionalidad necesaria para trabajar a un mayor nivel de abstracción sobre ésta.

De esta forma, mediante el mapeado de funciones C de OpenAL en funciones Java que realiza la librería *OpenAL4Android*, se logran implementar una serie de clases Java que pueden ser utilizadas por aplicaciones *Android* y que forman parte de *BFG Toolkit*.

6.2. App Engine y Google Analytics

6.2.1. Introducción

Una de las características que hubo que implementar durante el desarrollo del proyecto, era el almacenamiento de los datos que los usuarios proporcionaban, con el propósito de ir mejorando las distintas aplicaciones. Para ello se implementaron unos formularios y un sistema de *log* auxiliares a la librería ya descrita, que posteriormente se guardarían en una base de datos.

6.2.2. Google App Engine y Google Web Toolkit

Google App Engine (GAP en adelante) es un sistema que permite alojar aplicaciones web realizadas en una amplia variedad de lenguajes de programación con un coste razonable. En nuestro caso, se utilizó un sencillo cliente HTTP para realizar la comunicación con los servidores GAP. La parte de servidor, aplicación web y la interfaz que en sí se encarga de recibir los datos y llevar a cabo la comunicación, es *Google Web Toolkit* (GWT), otra herramienta desarrollada por *Google* que permite realizar páginas web mediante el lenguaje de programación Java. La principal ventaja de esta tecnología es que permite desarrollar una aplicación web de elevada complejidad en un sólo lenguaje cuyo desarrollo de otra forma necesitaría del uso de una amplia variedad de tecnologías (JavaScript, HTML, CSS, etc.)

Inicialmente en la parte del servidor se utilizó una base de datos de objetos Java con persistencia (JSON) que proporcionaba GAP, puesto que buscando información se pudo comprobar que eran de los sistemas que ofrecían una mejor solución para la comunicación entre aplicaciones *Android* y la web. De esta forma, consultando la base de datos de objetos, se pueden mostrar los resultados obtenidos de la evaluación de nuestras aplicaciones en una página web.

[25] http://pielot.org/2011/11/10/openal4android-2/

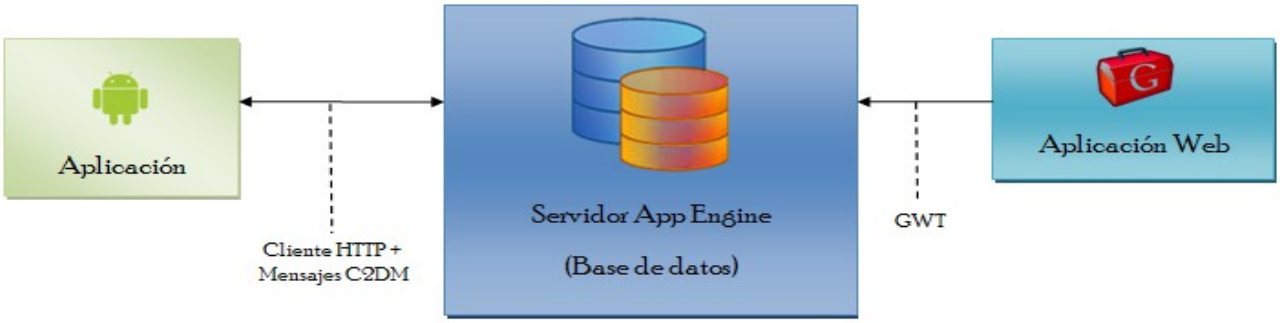


Figura 65. Diagrama Aplicación - Conexión App Engine - Web

Tras haber desarrollado todo el sistema para el envío de información, desde una de nuestras aplicaciones a la base de datos de *App Engine* con posibilidad de visualización mediante web, se descubrió otra posibilidad: *Google Analytics*. Frente a esta alternativa, nuestra infraestructura inicial resultaba ser menos manejable a la hora de aplicarla en diferentes aplicaciones.

6.2.3. Google Analytics for Mobile

Google Analytics es una herramienta que proporciona información sobre el tráfico de páginas web y actualmente también de aplicaciones *Android*. Dispone de un sistema de análisis que te permite ver en tiempo real el número de usuarios conectados a la aplicación y el número de eventos que generan en un tiempo determinado, pudiendo visualizar los datos mediante gráficos. Se decidió analizar esta posibilidad y elegir uno de los sistemas, puesto que es más fácil mantener actualizado un único sistema.

6.2.4. Comparativa Google App Engine y Google Analytics Mobile

Google App Engine	**Google Analytics Mobile**
Necesario desarrollo de web y sistema *log*.	No es necesario desarrollar la web para mostrar los datos, puesto que ya es proporcionada por Google.
Libertad para estandarizar y filtrar la información	Formato de datos y presentación fijos.
Dificultad de migrar a otras plataformas/tecnologías.	Proporciona mayor escalabilidad y protección.
Para mostrar visualización en tiempo real, hay que implementarlo.	Ofrece la posibilidad de visualización en tiempo real de usuarios conectados.

Tabla 9. Comparativa Google App Engine y Google Analytics

Finalmente, se optó por *Google Analytics*, por la comodidad que ofrece para registrar eventos, la facilidad para exportar la información recopilada y por posibilitar visualizar la información mediante gráficos que hacen que el análisis de datos sea más vistoso. Además, el uso de *Google Analytics* evita tener que actualizar la web de *Google App Engine* cada vez que se desarrolle un juego nuevo, por lo que también supone un ahorro en tiempo y esfuerzo.

6.3. XML parser

6.3.1. Introducción

Con el objetivo de permitir al usuario configurar el teclado como desee y más aún sabiendo de antemano, que no todos los dispositivos *Android* disponen del mismo número de botones físicos, se decidió modelar un sistema para guardar dicha configuración de usuario. La persistencia de datos se lleva a cabo con un documento XML que se aloja en la memoria del dispositivo, haciendo uso de módulos de *escritura-lectura* de documentos XML.

La estructura del archivo XML, queda de la siguiente manera.

```
<keyboard num = "2">
      <rowmap key = "94" action="blind_mode"/>
      <rowmap key = "80" action="zoom"/>
</keyboard>
```

Donde el atributo *num*, indica el número de acciones que se han definido para el juego y cada *rowmap* contiene como atributos la tecla asignada y la acción que realiza. El número asignado a cada tecla (key), se ha hecho acorde con las constantes definidas en la clase *KeyEvent* de la API de *Android*[26].

Las tecnologías que se han utilizado para cargar y guardar el fichero XML son SAX y DOM, respectivamente.

6.3.2. SAX

SAX son las siglas de *Simple API for XML*, originalmente, una API únicamente para el lenguaje de programación Java, que después se convirtió en la API estándar *de facto* para usar XML en Java. Existen versiones de SAX no sólo para Java, sino también para otros lenguajes de programación (como Python).

[26] http://developer.android.com/reference/android/view/KeyEvent.html

Presenta las siguientes características:

- Está orientado a eventos.
- Detecta cuándo empieza y termina un elemento o el documento, o un conjunto de caracteres, etc. (genera eventos)
- Gestiona los espacios de nombres.
- Comprueba que el documento está bien formado.
- Las aplicaciones necesitan implementar manejadores de los eventos notificados.
- SAX lee secuencialmente de principio a fin, sin cargar todo el documento en memoria.

En cuanto a las ventajas y desventajas:

- Eficiencia en cuanto al tiempo y la memoria empleados en el análisis.
- No dispone de la estructura en árbol.
- Es más difícil de manipular.
- Realiza una lectura secuencial del documento por lo que una vez leído no se puede volver atrás, algo que DOM sí permite.

6.3.3. DOM

El *Document Object Model* o DOM ('Modelo de Objetos del Documento' o 'Modelo en Objetos para la Representación de Documentos') es esencialmente una interfaz de programación de aplicaciones (API) que proporciona un conjunto estándar de objetos para representar documentos HTML y XML, un modelo estándar sobre cómo pueden combinarse dichos objetos, y una interfaz estándar para acceder a ellos y manipularlos. A través del DOM, los programas pueden acceder y modificar el contenido, estructura y estilo de los documentos HTML y XML, que es para lo que se diseñó principalmente.

6.4. Reporte de errores

6.4.1. Introducción

Una vez finalizado el desarrollo de las aplicaciones *Buscaminas Accesible*, *Golf* y *Zarodnik*, se tomó la decisión de publicarlas en *Google Play*, la tienda de software en línea desarrollada por *Google* para los dispositivos *Android*. Por lo que, una herramienta interesante que deberían implementar estas aplicaciones antes de publicarlas en la tienda en línea, es la recopilación de información de los posibles fallos que pudieran surgir, ya que el mercado de dispositivos *Android* es muy extenso y con una amplia gama de tecnologías y sistemas operativos diferentes entre sí. Así pues, se investigaron sistemas que permitieran aprovechar la oportunidad de publicar las aplicaciones a *Google Play* y con ello, ir mejorando las aplicaciones para ampliar su compatibilidad con más dispositivos *Android*.

6.4.2. ACRA

Es una herramienta con licencia *Apache License* 2.0, que permite a las aplicaciones *Android* enviar informes de error de diferentes formas:

- **Hoja de cálculo de *Google Docs***: este es el comportamiento por defecto, los informes de error se envían a una hoja de cálculo configurada para su uso.
- **Correo electrónico**.
- **Servidor HTTP personal**.

También te permite implementar tu propio informe de error, pudiendo de esta manera enviarlos mediante cualquier otro medio que se desee.

Se dispone de tres formas diferentes para interactuar con el usuario cuando se produce un error:

- **Silenciosa** (por defecto): las acciones de *ACRA* no son visibles. El informe de error se envía y posteriormente el sistema de errores de *Android* muestra su diálogo de forzar cierre si fuese necesario.
- ***Toast***: cuando se produce un error, *ACRA* muestra un *Toast* de *Android*, un mensaje que aparece en la pantalla, y simultáneamente envía el informe.
- **Notificación**: se muestra un *Toast* opcional cuando la aplicación lanza un error, pero el informe no se envía inmediatamente. Aparecerá una advertencia en la barra de notificación del dispositivo, indicando que se debe enviar un informe. Al seleccionar esta advertencia se mostrará un diálogo pidiendo autorización para enviar un informe de error con un comentario opcional por parte del usuario.

6.5. Motores de síntesis de voz

6.5.1. Introducción

La síntesis de voz es la producción artificial del habla humana. Un sistema *text-to-speech* (TTS) convierte texto escrito en un lenguaje determinado, en voz.

La voz puede crearse utilizando pequeños fragmentos de voz humana almacenados en una base de datos (suele ser una voz clara), o generada artificialmente por la computadora (generalmente robotizada o sin claridad). Para lograr la claridad más completa, lo adecuado es almacenar pronunciaciones de palabras completas.

Los sintetizadores de voz son muy útiles para las personas con discapacidad; por ejemplo, se utiliza un sintetizador de voz en programas de asistencia como los lectores de pantalla.

La calidad de los sintetizadores de voz se mide en relación a la semejanza con la voz humana y por su capacidad de entenderse.

Un sistema TTS o motor, está compuesto de dos partes, *front-end*, que se encarga de normalizar el texto convirtiendo símbolos en la pronunciación equivalente de una palabra, además de asignar transcripciones fonéticas a cada una de ellas. Por otro lado está el *back-end*[27], que es en sí el sintetizador cuya función es convertir las sílabas que componen cada palabra y frase en sonidos. Con esto queda finalizado todo el proceso de síntesis.

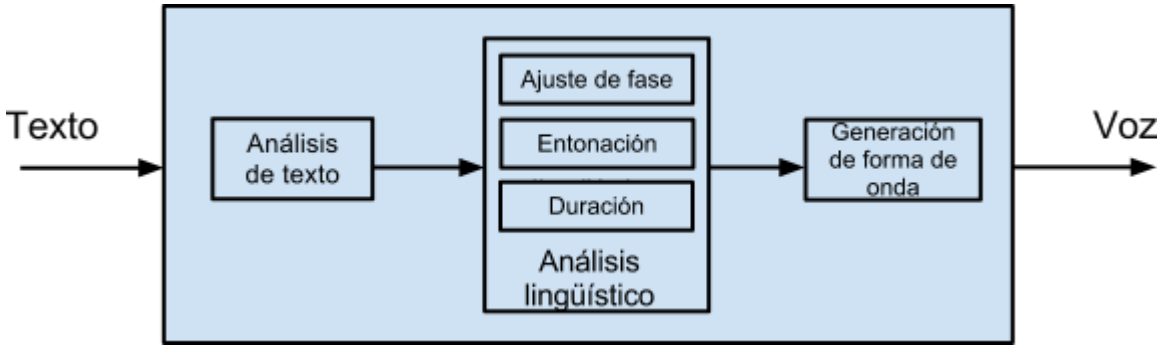


Figura 66. Sistema TTS

6.5.2. Síntesis de voz en Android

Desde la versión 1.6, *Android* incluye soporte para síntesis de voz. En primer lugar se debe seleccionar el idioma, ya que una misma palabra no se pronuncia igual en un idioma u otro. Por lo tanto, la voz y el diccionario son recursos específicos del idioma seleccionado y es necesario cargarlo antes de que el sintetizador comience a hablar.

A pesar de que todos los dispositivos *Android* soportan *Text-To-Speech*, algunos tienen almacenamiento limitado y es posible que no se disponga de algún archivo determinado de alguna lengua concreta. La API de TTS da la posibilidad de descargar e instalar los recursos que faltan. Por lo que una buena manera de empezar antes de lanzar nuestra actividad *Android*, es comprobar que se disponen de todos los recursos necesarios para el sintetizador.

```
Intent checkIntent = new Intent();
checkIntent.setAction(TextToSpeech.Engine.ACTION_CHECK_TTS_DATA);
startActivityForResult(checkIntent, MY_DATA_CHECK_CODE);
```

Se obtiene un resultado positivo mediante el código CHECK_VOICE_DATA_PAS, después de la creación del objeto TTS. Éste informa de que nuestro dispositivo ya está listo para hablar. En caso contrario se debe

[27] Términos que se relacionan con el principio y final de un proceso.

informar al usuario de la instalación de los recursos, mediante el *Android Intent* dado por ACTION_INSTALL_TTS_DATA, el cual llevará al usuario a *Google Play* permitiéndole iniciar la descarga.

```
private TextToSpeech mTts;
protected void onActivityResult(
       int requestCode, int resultCode, Intent data) {
   if (requestCode == MY_DATA_CHECK_CODE) {
       if (resultCode == TextToSpeech.Engine.CHECK_VOICE_DATA_PASS) {
           // success, create the TTS instance
           mTts = new TextToSpeech(this, this);
       } else {
           // missing data, install it
           Intent installIntent = new Intent();
           installIntent.setAction(
               TextToSpeech.Engine.ACTION_INSTALL_TTS_DATA);
           startActivity(installIntent);
       }
   }
}
```

Los motores de síntesis de voz están disponibles para descargar en *Google Play*. Los hay gratuitos y de pago. Por lo general, los de pago tienen mayor calidad de voz que los gratuitos. Para cambiar de motor de síntesis, se descarga el motor elegido y posteriormente los datos de voz, ya que se distribuyen por separado. Las opciones de configuración disponibles son: selección de voz dentro de ese motor específico, velocidad de voz e idioma, si se dispone de los datos de voz correspondientes en ese idioma. Estas opciones se pueden encontrar en los ajustes del dispositivo móvil, en la categoría "Entrada y salida de voz".

Además, las opciones de configuración del sintetizador de voz mencionadas anteriormente, se pueden modificar programáticamente mediante los métodos proporcionados por la API de TTS. Por ejemplo, para cambiar el idioma del sintetizador de voz se lleva a cabo lo siguiente:

```
mTts.setLanguage(Locale.US);
```

Para cargar el idioma Inglés Americano. Para comprobar si éste está disponible, se puede hacer uso del método isLanguageAvailable(). Si se quiere modificar la velocidad de voz programáticamente se ha de usar el método setSpeechRate(rate). También es posible establecer el motor de síntesis que se desee. Todas las opciones posibles las se pueden encontrar en la API de *Android*.

6.5.3. Comparativa motores de síntesis Android

Tabla 10. Motores de síntesis de voz[28]

Motores	Gratuito	Idiomas	Calidad	Voces	Fuente
picoTTS	Sí	Alemán, Inglés (UK), Español, Francés, Italiano	Mala	Voz robotizada	Viene instalado por defecto en la versión 2.2 de *Android*
eSpeakTTS	Sí	40 idiomas	Mala	Voz robotizada	https://play.google.com/store/apps/details?id=com.googlecode.eyesfree.espeak&hl=es
IVONA	No. Versión beta gratuita hasta que finalice el periodo	11 idiomas	Muy buena	Voz femenina y masculina dependiendo del idioma seleccionado	http://mobile.ivona.com
Classic Text To Speech Engine (SVOX)	No. Precio 2.99€ cada archivo de voz	Más de 25 idiomas	Buena	Más de 40 voces femeninas y masculinas	https://play.google.com/store/apps/details?id=com.svox.classic&hl=es
Loquendo TTS	No. Precio 3.99€ cada archivo de voz	18 idiomas	Buena. Además dispone de ciertos comandos que proporcionan al sintetizador "emociones"	Voz femenina y masculina	http://www.loquendo.com/es/demo-center/demo-tts/

[28] Estos datos se recopilaron en Noviembre de 2011.

6.5.4. Conclusiones

De los sintetizadores analizados en la sección anterior, se recomienda el uso del sintetizador IVONA, ya que está disponible de forma gratuita durante su periodo de prueba. Además, dispone de una buena calidad de voz, natural e inteligible, aproximándose prácticamente a la voz de una persona humana.

No sería adecuado obligar al usuario a instalar un motor de síntesis concreto para poder hacer uso de esta aplicación, pues cada uno es libre de tener instalado el que más le guste, aún así, en las aplicaciones que se han desarrollado se recomienda tener instalado este sintetizador, para usuarios que desconozcan sobre sintetizadores para *Android*.

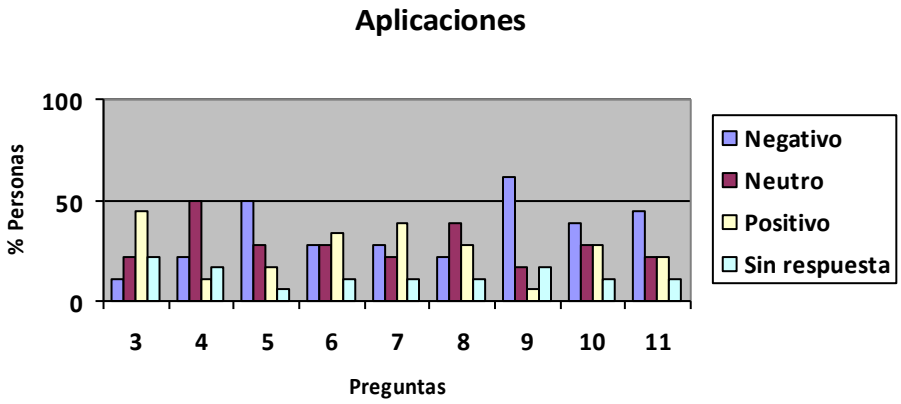
7. Evaluación

Dada la dificultad de encontrar personas pertenecientes al público objetivo al que va dirigido el proyecto, puesto que la plataforma *Android* no parece ser la más indicada para los discapacitados visuales y que no se disponía de presupuesto para la contratación de personal dedicado a la evaluación de las aplicaciones, se planteó una evaluación *online* que todavía está **en proceso**. La metodología se llevó a cabo mediante evaluaciones continuas con los directores del proyecto, que disponían de experiencia previa en el campo de la accesibilidad, ocupándose de especificar las principales barreras que se pueden encontrar en este campo. Además, se han realizado evaluaciones globales por parte de usuarios finales. A continuación se detallan los resultados obtenidos.

7.1. Evaluación de usuarios finales

7.1.1. Cuantitativa

En cada juego existe la posibilidad de realizar el envío de un cuestionario accesible (las preguntas del formulario se pueden consultar en el Anexo III) para la evaluación de la aplicación. Este cuestionario hace uso del formato *Likert,* es decir, el formato de las respuestas consta de 5 niveles (del 1 al 5). Aunque no ha sido posible la obtención de pruebas sobre un gran número de usuarios ciegos, las estadísticas son bastante positivas dado que se tratan de las primeras aplicaciones accesibles realizadas con la herramienta y además, la plataforma *Android* dispone de una amplia oferta de juegos, por lo que es un mercado muy competitivo. A continuación se muestra una gráfica con los resultados obtenidos de los formularios de evaluación que contienen los juegos publicados en *Google Play,* donde se ha considerado como "positivo" la suma de las frecuencias de 4 y 5, como "neutro" el valor 3 y como "negativo" 1 y 2. Se ha realizado esta agrupación para facilitar la comprensión de las gráficas.



Como se puede observar en la gráfica, una mayoría de los participantes son jugadores habituales de PC o sobremesa (pregunta 3) que suelen dedicar una cantidad de tiempo medio-alta a jugar juegos en plataformas móviles (pregunta 4).

Las preguntas peor valoradas por los usuarios, se corresponden con la valoración de la dificultad de aprendizaje de los juegos y de la historia de los mismos (preguntas 5 y 9). Se podría haber **reducido la curva de aprendizaje (pregunta 5)** integrando los tutoriales al comenzar la partida, al igual que se hizo con el último juego publicado, *Zarodnik*, de tal forma que se introduzca al usuario en el mundo poco a poco y no teniendo que realizar un tutorial, que en ocasiones puede ser pesado.

En cuanto a la **valoración de la historia (pregunta 9)**, sólo es aplicable a *Zarodnik*, ya que los otros juegos publicados no disponen de historia. Es significativo el resultado obtenido en esta pregunta, puesto que más de un 50% de los usuarios ha valorado negativamente este apartado, lo cual indica una demanda para mejorar la calidad de la historia de este juego. En este aspecto se trata con más profundidad en *The Shadow of the Past* (ver sección 3.5) juego que no se pudo llegar a publicar.

En cuanto a las preguntas 10 y 11 **referentes a las opciones de accesibilidad y a la retroalimentación en los juegos**, reflejan otro aspecto interesante y es que como la mayoría de los participantes en estos resultados no tienen discapacidad alguna, han encontrado la interacción con los juegos diferente al resto de aplicaciones que suelen utilizar, lo que repercutió negativamente en su experiencia. Estos resultados no parecen relevantes, ya que en este proyecto se dio más importancia a la interacción con usuarios ciegos.

Por último destacar, la parte referente al **grado de entretenimiento del juego** (preguntas 7 y 6), se puede ver que en este campo las aplicaciones en su conjunto han recibido un valor positivo y neutro por la mayoría de los participantes.

7.1.2. Cualitativa

También se llevó a cabo la divulgación del proyecto en distintas comunidades formadas por personas con discapacidad visual con el objetivo de recibir *feedback* y poder de esta manera, mejorar las aplicaciones desarrolladas.

Las comunidades que han realizado evaluaciones de las aplicaciones son muy diferentes: desde foros y sitios de referencia para jugadores ciegos, como *audiogames.net* o *ablegamers.net*, hasta *blogs* de desarrolladores y expertos en accesibilidad, como *videojuegosaccesibles.es* o empresas como *Technosite*, perteneciente al grupo empresarial de la Fundación ONCE.

En el caso de las **comunidades de jugadores**, se ha recibido información de usuarios que han transmitido dificultades para configurar alguna de las aplicaciones o la distribución de alguno de los elementos de las

interfaces de usuario en la pantalla de sus dispositivos. No obstante, han sido capaces de utilizarlas sin excesivos problemas.

Por otro lado, el ***blog videojuegosaccesibles.es***, escrito por el desarrollador experto en accesibilidad Javier Mairena (Fundador de la empresa de desarrollo de videojuegos *The Game Kitchen*[29]), ha valorado positivamente el trabajo realizado en cuanto al desarrollo de este tipo de juegos en la plataforma *Android* y la publicación de *BFG Toolkit* de forma gratuita. Sin embargo, criticó que los juegos desarrollados no estuvieran pensados para jugadores con visión baja, es decir, no se dispone de textos contrastados con el fondo u otras características de accesibilidad, para facilitar la lectura a personas con una pérdida de visión que no sea total. Cosa que por otro lado, está fuera del alcance del proyecto (ver sección 1.4.).

7.2. Estadísticas de uso

Como resultado de meses de trabajo, se publicaron a día 1 de Mayo los tres primeros juegos en el mercado de aplicaciones oficial de *Google*, *Google Play*. El objetivo era, una vez se tuviera una versión estable de cada aplicación, recibir opiniones de los usuarios sobre aspectos a mejorar y publicar nuevas versiones posteriormente.

A continuación se muestra un gráfico con las distintas versiones *Android* que han hecho uso de las aplicaciones:

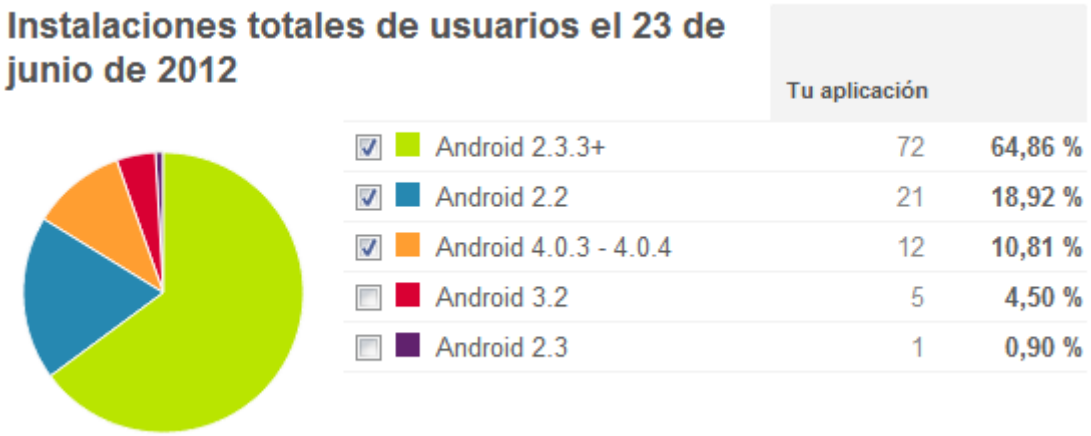


Figura 67. Versiones Android en Buscaminas Accesible

[29] http://www.thegamekitchen.com/

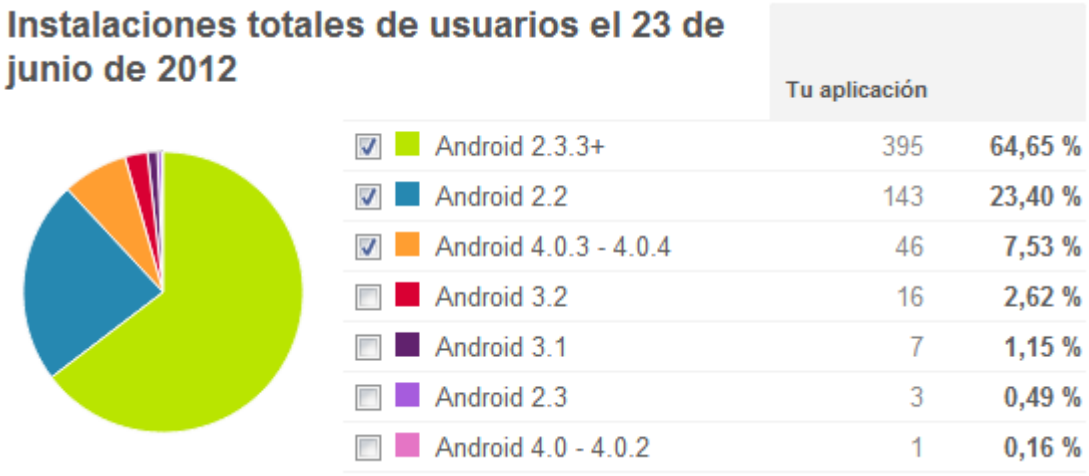


Figura 68. Versiones Android en Golf Accesible

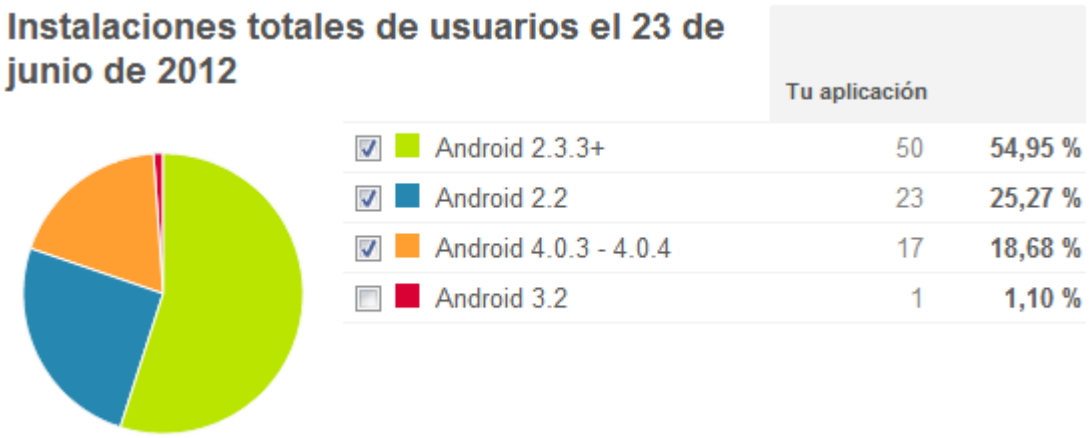


Figura 69. Versiones Android en Zarodnik

Como se puede ver en los gráficos anteriores, el rango de versiones va desde la versión 2.2 *Froyo* hasta la versión 4.0.X *Ice Cream Sandwich*, siendo la 2.3.3 *Gingerbread* la más utilizada. Además, existe un pequeño porcentaje de dispositivos con la versión 4.0 *Ice Cream Sandwich*.

El problema de la fragmentación en *Android* ("Fragmentation: Android's Real Problem or Google's Penetration Tactic? - Technically Personal!," 2012) es la consecuencia de tener un sistema operativo gestionando una gran variedad de dispositivos cuyo soporte y actualización dependen del fabricante. Muchos de estos fabricantes dejan de actualizar la versión del sistema operativo de sus dispositivos, lo que se traduce en una gran cantidad de usuarios que conservan versiones antiguas de *Android* simplemente porque el fabricante les dejó de dar soporte. Para el desarrollador es un problema, ya que se hace **necesaria la compatibilidad con una gran variedad de versiones** del sistema *Android* si no se desea perder cuota de mercado. A pesar de esta dificultad el *toolkit* logra ser estable para todas las versiones mostradas en las gráficas anteriores.

En cuanto a la variedad de dispositivos móviles que han hecho uso de las aplicaciones:

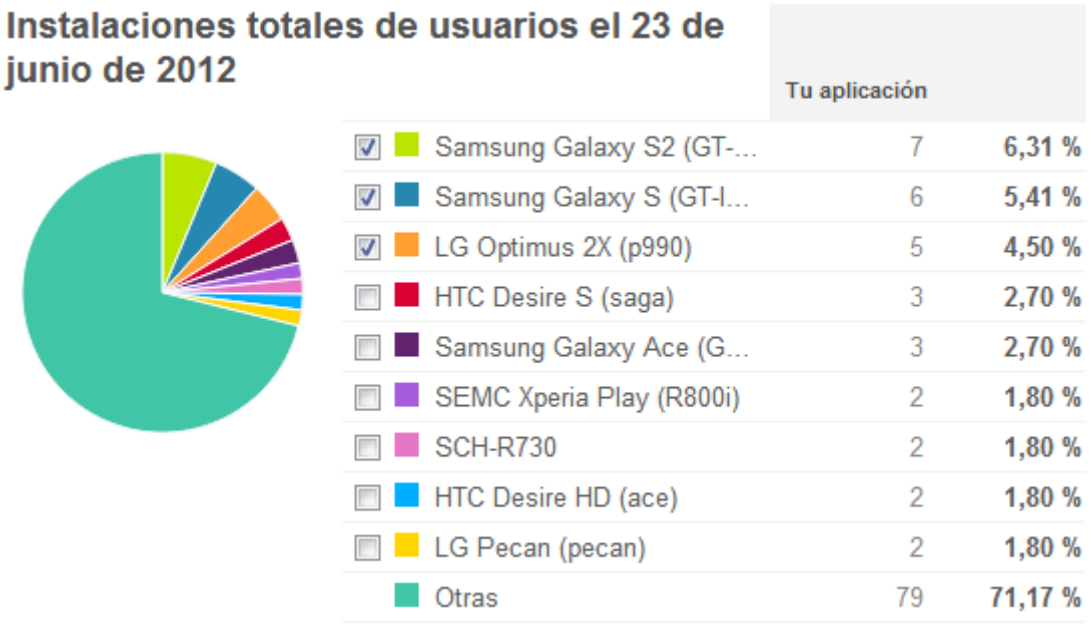


Figura 70. Dispositivos móviles en Buscaminas Accesible

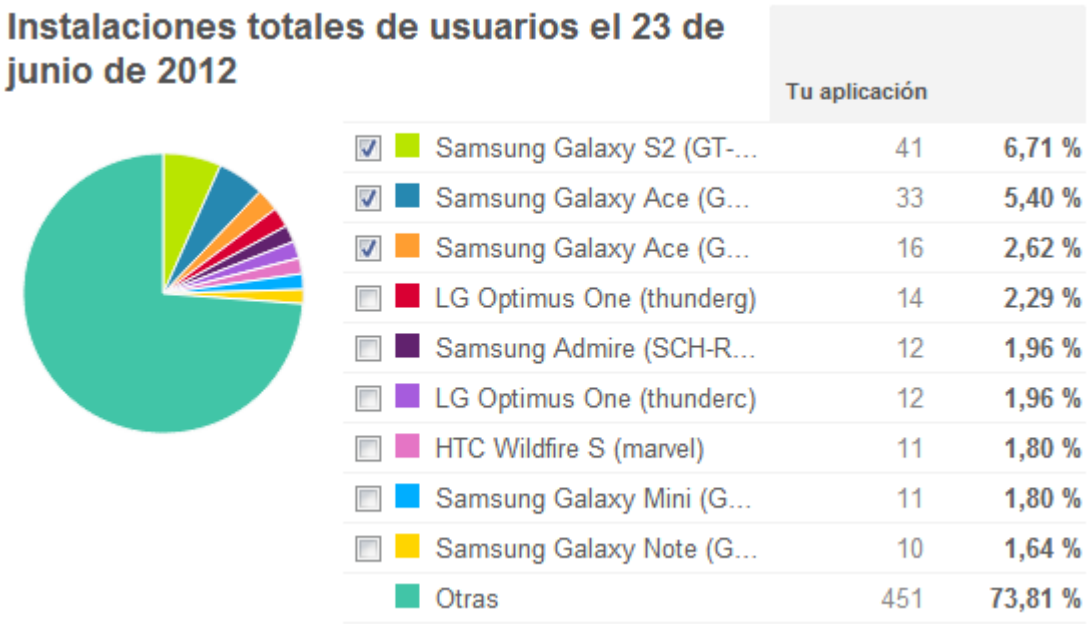


Figura 71. Dispositivos móviles en Golf Accesible

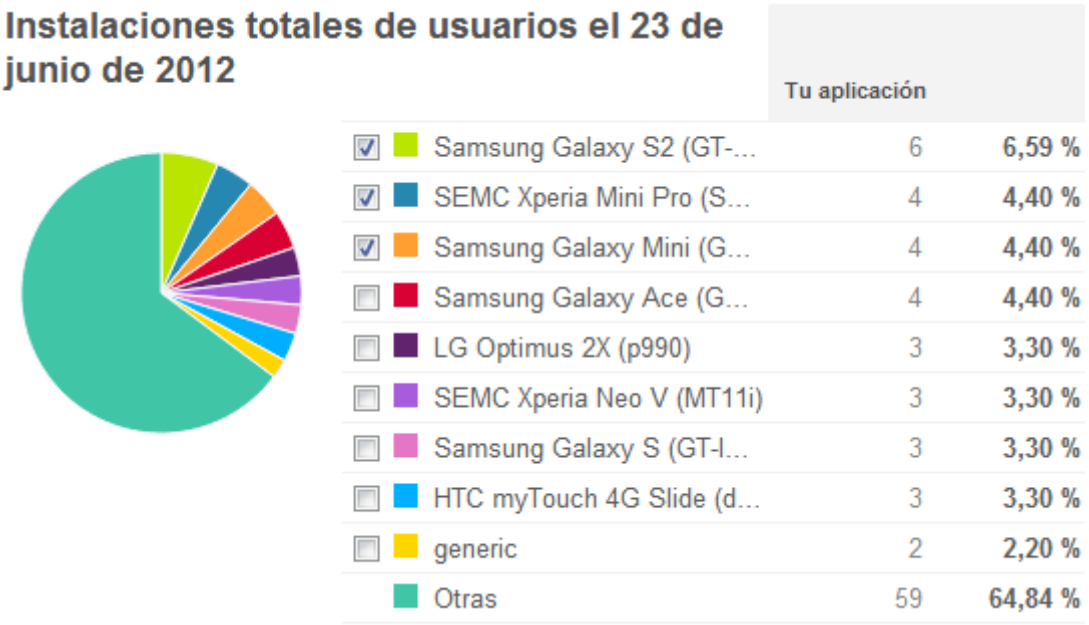


Figura 72. Dispositivos móviles en Zarodnik

Se puede observar hay una **gran variedad de dispositivos móviles** que han utilizado las aplicaciones desarrolladas, lo que ha permitido llevar a cabo una evaluación de la estabilidad de las aplicaciones más detallada, pues mediante los informes de error se recibían los fallos que sucedían en los dispositivos.

Por otro lado, **los resultados obtenidos en términos del número de instalaciones totales fueron bastante positivos**, en un mercado tremendamente competitivo como es *Google Play* y dado que la presencia del público objetivo es reducida en la plataforma *Android*.

Figura 73. Instalaciones totales Buscaminas Accesible

Figura 74. Instalaciones totales Golf Accesible

Figura 75. Instalaciones totales Zarodnik

Como se puede ver en las gráficas a fecha **24/06/2012** se ha alcanzado cerca de **120 instalaciones** en *Buscaminas Accesible*, **700** en *Golf Accesible* y **100** en *Zarodnik* en dispositivos de **18** países diferentes. El número de instalaciones activas siempre es creciente lo cual es un síntoma de que las aplicaciones han resultado de interés para cierta parte del público.

7.3. Evaluación técnica

Durante el desarrollo se presentaron algunas dificultades, como errores en la persistencia de datos de las aplicaciones que propiciaba en ocasiones la pérdida de información en caso de bloqueo del dispositivo, problemas de programación no segura o problemas de compatibilidad con ICS como el mencionado a continuación. Muchos de ellos se solucionaron gracias a la herramienta de informe de errores de las aplicaciones tras la publicación en *Google Play*.

Uno de los errores que se daba con más frecuencia provenía desde dispositivos con versión del sistema operativo *Android* actualizado, *Ice Cream Sandwich*, sistema que se había lanzado poco después de la subida de las aplicaciones. Todas aquellas personas que hubieran actualizado sus dispositivos eran incapaces de utilizar nuestras aplicaciones debido a este error crítico.

Pero además de éste, se recibieron hasta 20 errores que no se producían en los dispositivos utilizados para la evaluación de los juegos, algunos de los cuáles son errores de programación y otros son fallos conocidos de las librerías utilizadas, como *Google Analytics for Mobile* o la API de *Android* en su versión 4.0.

Gracias a la información que se ha obtenido a partir de los informes de error, se puede decir, que el *framework* es lo suficientemente estable y robusto como para ayudar a otros desarrolladores en la implementación de juegos accesibles para *Android*. No obstante, también existen aspectos de implementación que podrían haberse realizado de forma que la codificación resultante fuera más robusta, por ejemplo, un error producido a causa de los recursos de sonido, que puede estar asociado a la compatibilidad nativa de ciertos formatos de audio con el sistema *Android*. Además, existen otros informes de errores como los asociados a la librería *Google Analytics for Mobile* (ver sección 6.2.3), errores en la persistencia de datos de los juegos u otros que a fecha de redacción de este documento están siendo evaluados.

7.4. Conclusiones

En general se ha tenido una buena acogida, ya que ha habido una gran variedad de dispositivos móviles y versiones *Android* que han hecho uso de las distintas aplicaciones. Además, gracias a **las evaluaciones recibidas** por parte de comunidades, *blogs* y otras personas expertas en el campo de la accesibilidad en juegos, **contribuyeron a la estabilidad** de estas aplicaciones que queda demostrada con los datos de uso que aparecen en esta sección. Además, permiten concluir con que el conjunto de herramientas *BFG Toolkit* es **lo suficientemente estable como para ayudar a otros desarrolladores** a implementar opciones de accesibilidad en juegos dentro de la plataforma *Android*, pudiendo suponer un punto de partida para trabajos futuros en los que mejorar los aspectos de accesibilidad.

8. Conclusiones

8.1. Resultados y objetivos

Como objetivos iniciales del proyecto se propusieron los siguientes:

- **Identificación de barreras de accesibilidad en plataformas móviles**.
- **Propuesta de soluciones de diseño** que mejoren la accesibilidad de los juegos.
- Desarrollo de un **conjunto de herramientas** que facilite la programación de juegos accesibles que incluirán las soluciones de diseño propuestas (ver sección 4).

La **identificación de las barreras de accesibilidad** planteó una problemática difícil de superar para desarrolladores sin experiencia alguna en accesibilidad y usabilidad. En las primeras etapas del proyecto se cometieron errores elementales de diseño en algunos de los prototipos. Pero se superaron, ya que difícilmente se hubieran podido plantear las estrategias de diseño y en consecuencia los juegos, sin determinar previamente el problema que se pretendía solucionar.

Además, se plantearon **una serie de soluciones de diseño** para cada juego que permitieron resolver una gran parte de los problemas de accesibilidad presentes en los géneros a los que pertenecen. A partir de ellas se realizó el desarrollo de una serie de juegos que pertenecen a categorías completamente distintas y que representan de forma significativa mecanismos de interacción diferentes. Todo ellos completamente funcionales, incluso tres de ellos en fase de producción de cara al público.

Cada una de esas soluciones de diseño permitió mediante una serie de desarrollos ágiles, ir incrementando la funcionalidad de ***BFG Toolkit.*** Al finalizar el proyecto se ha obtenido como resultado un *framework* lo suficientemente robusto y estable como para desarrollar con él cuatro aplicaciones, tres de ellas en pleno funcionamiento publicadas en *Google Play*.

Por lo tanto, se podría considerar que se han cumplido todos los objetivos de manera satisfactoria. No sólo ha permitido desarrollar juegos accesibles para ciegos y su publicación con cierta repercusión (ver sección 7) dentro de *Google Play*, sino que además, se ha contribuido con una herramienta que reduce los costes de desarrollo de futuros juegos accesibles de forma significativa.

8.2. Trabajo futuro

En el campo de la accesibilidad en juegos, existe una inmensidad de problemas que no se han tratado en el presente proyecto y para los se podrían haber desarrollado herramientas dentro de *BFG Toolkit*. Por cuestiones de alcance, se limitó a tratar el problema de la accesibilidad para ciegos, pero se podría incorporar funcionalidad que tratara de solucionar problemas de interacción para personas con problemas de audición o baja visión entre otros. Se han implementado aquellas alternativas que parecían más significativas para el tipo de juegos que se han desarrollado a lo largo del proyecto. Sin embargo, es posible, como trabajo futuro, analizar algunas líneas de trabajo adicionales como las que se mencionan a continuación.

Desde un punto de vista **técnico** podemos considerar las siguientes líneas de trabajo:

- En lo referente a la funcionalidad para facilitar el desarrollo de juegos con opciones de accesibilidad para ciegos dentro de *BFG Toolkit*.

    - **Funcionalidad para facilitar el manejo de vistas *Android***. No se dispuso de tiempo suficiente para incorporar funcionalidad que facilitara a los desarrolladores la gestión más en profundidad de las vistas *Android*.
    - **Comunicación con visor Braille**. Añadir funcionalidad que permita la utilización de mecanismos alternativos a los basados en sonido, para la notificación de lo que ocurre en el mundo del juego.
    - **Algoritmo de optimización de espacio en pantalla**. Ideas para mejorar la gestión del espacio de la pantalla a la hora de seleccionar los diálogos de los juegos basados en texto.

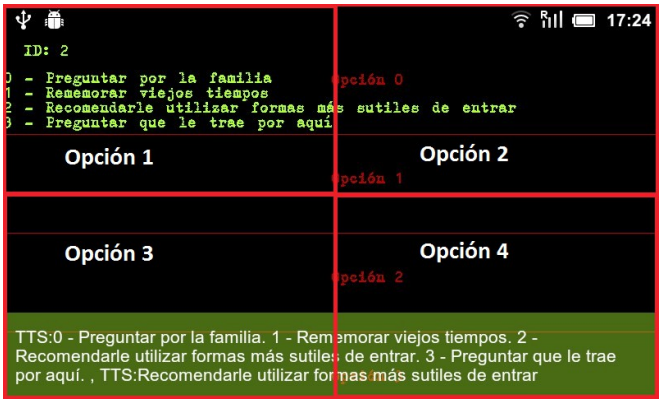


Figura 76. Optimización de espacio para 4 opciones

    - **Mejoras en el sistema de sonido 3D**. Actualmente existen muchas dificultades para trabajar con algunos recursos de sonido en términos de volumen y tono, que pueden obligar al desarrollador a su descarte. Se hace necesario incorporar funcionalidad que regule de forma automática la amplitud y la frecuencia de las fuentes de sonido, en función del recurso proporcionado. Además, la interfaz de las clases es mejorable.

- **Sistemas más sofisticados de navegación por menús basados en sonido.** Actualmente es posible la recuperación del último mensaje reproducido por el sintetizador de voz. Idealmente podría permitirse una navegación completa por toda la información que se haya reproducido por el sintetizador de voz.
- **Mecanismo de interacción por voz.** Alternativa interesante para la resolución del problema de interacción de los usuarios ciegos con las pantallas en los dispositivos móviles.

- Relativas a la herramienta que proporciona funcionalidad como motor de juegos:

  - **Optimización del framework.** Es posible mejorar el rendimiento mediante la codificación de parte del *toolkit* en C++. También se debería tratar de apoyar directamente sobre librerías de bajo nivel C intermedias con *OpenGL*, o bien directamente sobre *OpenGL*, siempre que sea posible especialmente en los aspectos relacionados con la gestión de imágenes y sonido.
  - **Eliminar dependencias con las clases de gráficos de la API de *Android*.** Cada aplicación *Android* tiene asignada por parte del sistema una cantidad de memoria limitada para sus recursos (sonido, imágenes, etc.). Si esta cantidad se supera, el sistema lanza una excepción que finaliza la aplicación. Al ser código Java se espera que el recolector de basura actúe periódicamente liberando los recursos no referenciados, pero es necesario que se liberen manualmente por el programador, de forma que cada vez que se instancia un objeto de las clases imagen de la API de *Android*, es necesario liberarlo después. En caso contrario, existirán problemas de estabilidad especialmente en dispositivos de gama baja que disponen de recursos más limitados.

    Existe por tanto, un problema de gestión de memoria con ciertas clases de la API de *Android* que puede solucionarse mediante la creación de un conjunto de clases que aporten la misma funcionalidad, pero que utilicen punteros inteligentes con la información de las mismas.
  - **Desplazamiento vectorial.** Para facilitar desplazamientos en 2D alternativos a los implementados.
  - **Sistema de carga de *scripts* para la especificación de comportamientos.** Utilizar algún lenguaje de *script* como LUA o *JavaScript* y un intérprete de los mismos, como base para la especificación de comportamientos para las entidades.

- Además, desde un punto de vista **experimental o de investigación** de cara a la mejora de las aplicaciones se podrían seguir las siguientes líneas:

  - **Explorar el uso de sensores tipo acelerómetro**. Estos sensores no están presentes en otras plataformas y pueden ser muy útiles tanto para ciegos como para personas sin discapacidad visual. Se llegó a plantear incluso un diseño de juego en este sentido, pero no se dispuso de tiempo para su implementación.
  - **Integrar las propuestas de los usuarios.** Algunas de ellas mencionadas en la sección 7.

- **Identificar y tratar los problemas de accesibilidad de usuarios con baja visión.** Mejorar el contraste y la legibilidad de los textos. Incluso incorporar mecanismo para configurar el color de distintos elementos del juego.
- **Realizar una evaluación más exhaustiva de las aplicaciones.** Hubiera sido interesante recibir más *feedback* por parte de un mayor número de personas y expertos en accesibilidad.

8.3. Principales aportaciones

A continuación, se exponen a modo de resumen las aportaciones del proyecto:

- **Un *toolkit* que facilita la gestión de características de accesibilidad para ciegos** en juegos para plataformas móviles. Es una aportación significativa, pues actualmente no existe ninguna solución de desarrollo que cubra las necesidades de aplicaciones tan particulares como son los videojuegos y que hagan especial hincapié en facilitar la introducción de características de accesibilidad con un coste mínimo.

- **Cuatro juegos accesibles** para personas ciegas, bilingües y completamente funcionales, de los cuales tres han sido ya puestos en producción a través del sistema de distribución de aplicaciones Google Play, obteniendo unos niveles de aceptación satisfactorios durante los primeros meses.

- **Un análisis del estado de la cuestión en materia de accesibilidad y juegos móviles** que puede ser útil en la elaboración de otros trabajos de este tipo.

- **Un conjunto de materiales de difusión del proyecto**, en inglés y castellano, que incluye páginas web, vídeos y gráficos divulgativos.

- **Difusión de los resultados del proyecto en comunidades de interés** como videojuegosaccesibles.es, audiogames.net, ablegamers.org, *Technosite* compañía de la fundación ONCE, etc., de los que se ha obtenido *feedback* valioso para mejorar tanto el conjunto de herramientas como las aplicaciones.

- Puesta en marcha de **un sistema de evaluación** con el fin de analizar la usabilidad final de los juegos para los usuarios finales (ciegos).

8.4. Conclusiones

Finalizado el proyecto, se puede hacer un balance de los objetivos cumplidos en base a los propuestos al inicio del mismo, así como las nuevas ideas que han ido surgiendo en las distintas fases del desarrollo, para incorporar soluciones a nuevas problemáticas.

Hay que tener en cuenta que este proyecto y en concreto la herramienta *BFG Toolkit*, puede ofrecer mucho más, es decir, existen numerosos caminos para conseguir una herramienta mucho más robusta y capaz de ofrecer un mejor rendimiento.

Por otro lado, *Android* es un sistema operativo móvil libre, con todas sus consecuencias. Está implementado en muchos dispositivos distintos y aunque en teoría el SDK ofrece compatibilidad para todos ellos, en la práctica se descubre que existen ciertos errores que son dependientes del dispositivo. Este tipo de problemas no aparecieron hasta la publicación de las aplicaciones en *Google Play* y se dispuso de los reportes de errores de nuestro sistema de seguimiento.

Por tanto, con los problemas obtenidos, se puede decir que desarrollar una aplicación estable para todos los dispositivos existentes en el mercado con plataforma *Android*, no es tarea fácil. Dispositivos con diferentes tamaños, distintas versiones del sistema y demás diferencias entre éstos, hacen que el desarrollo de aplicaciones para *Android* sea una tarea con un proceso continuado, ofreciendo a los usuarios nuevas actualizaciones para dar solución a los problemas que pueda haber en un futuro.

También estaba el problema presente dentro de las plataformas móviles *Android* en sus versiones previas a 4.0, la conocida como *Ice Cream Sandwich*. No disponía de opciones de accesibilidad que realmente permitieran a los desarrolladores tratar de hacer sus aplicaciones o juegos accesibles más allá de alguna aplicación que se discute en la sección 2.5 de este documento.

En ocasiones, se ha invertido más tiempo en funcionalidades secundarias no relacionadas con el campo principal de este proyecto, como permitir el sistema de seguimiento del uso que los usuarios hacen de las aplicaciones. Tiempo que podría haberse invertido en añadir y perfeccionar funcionalidades del *toolkit*.

Sin embargo, creemos que el balance del proyecto ha sido muy positivo y que se pueden extraer lecciones muy valiosas de lo aprendido en este proyecto. Esperamos que nuestra experiencia pueda servir de base para futuros desarrollos en este campo, evitando que se cometan los mismos errores y permitiendo aprender de nuestro trabajo.

Queremos agradecer la oportunidad que nos ha brindado este proyecto de adquirir nuevas habilidades tanto técnicas como personales, ya que sin duda nos serán de gran utilidad en el futuro.

9. Bibliografía y referencias

Amory, A. (2001). Building an Educational Adventure Game: Theory, Design and Lessons. *Journal Of Interactive Learning Research*, *12*(2/3), 249-263. Association for the Advancement of Computing in Education (AACE). Retrieved June 22, 2012, from http://www.questia.com/PM.qst?a=o&se=gglsc&d=5002415705

Annetta, L. A., Minogue, J., Holmes, S. Y., & Cheng, M. (2009). Investigating the impact of video games on high school students' engagement and learning about genetics. *Computers & Education*, *53*, 74-85.

Bierre, K. (2006). Improving Game Accessibility. *online GamaSutra*. Retrieved June 22, 2012, from http://www.gamasutra.com/view/feature/2342/improving_game_accessibility.php

Blunt, R. (2007). Does Game-Based Learning Work? Results from Three Recent Studies. Orlando, Florida, USA: NTSA. Retrieved June 22, 2012, from http://download.microsoft.com/download/a/4/f/a4f37ec6-809d-4098-9238-e2690a6fd8c4/GameBasedLearningStudies.doc

Dickey, M. D. (2006). Game Design Narrative for Learning: Appropriating Adventure Game Design Narrative Devices and Techniques for the Design of Interactive Learning Environments. *Educational Technology Research & Development*, *54*(3), 245-263. Springer. doi:10.1007/s11423-006-8806-y

ESA. (2012). Computer and Video Game Industry Tops $22 Billion in 2008. Retrieved June 22, 2012, from http://www.theesa.com/newsroom/release_detail.asp?releaseID=44

Ebner, M., & Holzinger, A. (2007). Successful implementation of user-centered game based learning in higher education: An example from civil engineering. *Computers & Education*, *49*(3), 873-890.

Ed. Burnette. "Hello Android *ATTiCA"*, Second Edition, 2009.

Fragmentation: Android's Real Problem or Google's Penetration Tactic? - Technically Personal! (2012). Retrieved June 23, 2012, from http://techpp.com/2012/05/22/android-fragmentation/

Gartner Inc. (2012). Worldwide Smartphone Sales in Fourth Quarter of 2011. Retrieved June 22, 2012, from http://www.gartner.com/it/page.jsp?id=1924314

Geekaphone. (2011). Mobile gaming by the numbers. Retrieved June 22, 2012, from http://geekaphone.com/blog/mobile-games-by-the-numbers/

IGDA. (2004). *Accessibility in Games: Motivations and Approaches* (p. 37). Retrieved June 22, 2012, from igda.org/accessibility

IGDA. (2012). International Game Developers Association (IGDA). Retrieved June 22, 2012, from http://www.igda.org

IGDA-SIG. (2012). International Game Developers Association (IGDA) Game Accessibility Special Interest Group (SIG). Retrieved June 22, 2012, from http://www.igda.org/accessibility

Juul, J. (2009). *A Casual Revolution: Reinventing Video Games and Their Players. Library* (p. 252). MIT Press. Retrieved June 22, 2012, from http://www.jesperjuul.net/casualrevolution/

Kanthan, R., & Senger, J.-L. (2011). The impact of specially designed digital games-based learning in undergraduate pathology and medical education. *Archives of pathology & laboratory medicine*, *135*(1), 135-42. doi:10.1043/2009-0698-OAR1.1

Kim, J., & Ricaurte, J. (2011). TapBeats: accessible and mobile casual gaming. *The proceedings of the 13th international ACM*, 285-286. Retrieved June 22, 2012, from http://dl.acm.org/citation.cfm?id=2049609

Making Video Games Accessible: Business Justifications and Design Considerations. (n.d.). Retrieved June 21, 2012, from http://msdn.microsoft.com/en-us/library/windows/desktop/ee415219(v=vs.85).aspx

Mayo, M. (2007). Games for science and engineering education. *Communications of the ACM*, *50*(7), 30-35. ACM Press. Retrieved June 22, 2012, from citeulike-article-id:1450072

NPD Group. (2007). Playing Video Games Viewed as Family/Group Activity and Stress Reducer. Retrieved June 22, 2012, from http://www.npd.com/press/releases/press_071212.html

Nash Information Services. (2012). US Movie Market Summary for 2010. Retrieved June 22, 2012, from http://www.the-numbers.com/market/2010.php

Nielsen Entertainment Research Group. (2011). U.S. Video Game Buyers Shifting Entertainment Budgets. Retrieved June 22, 2012, from http://blog.nielsen.com/nielsenwire/consumer/u-s-video-game-buyers-shifting-entertainment-budgets/

Roman Pichler, "Agile Product Management with Scrum," *Addison-Wesley Professional*, First Edition, 2010.

Rosser, J. C., Lynch, P. J., Cuddihy, L., Gentile, D. a, Klonsky, J., & Merrell, R. (2007). The impact of video games on training surgeons in the 21st century. *Archives of surgery (Chicago, Ill. ▪ : 1960)*, *142*(2), 181-6; discusssion 186. doi:10.1001/archsurg.142.2.181

Sedeno, A. (2010). Videogames as cultural devices: development of spatial skills and application in learning. *COMUNICAR*, *17*(34), 183-189. Retrieved June 22, 2012, from http://apps.isiknowledge.com/full_record.do?product=WOS&search_mode=GeneralSearch&qid=14&SID=N14OIeD5ne2NO2Ailh5&page=5&doc=226

T. Westin, K. Bierre, D. Gramenos, and M. Hinn. "Advances in Game Accessibility from 2005 to 2010," *Universal Access in HCI, Part II, HCII 2011*, vol. LNCS 6766, pp. 400-409, 2011.

Tollefsen, M. L. (2006). Guidelines for Developing Accessible Games. Based on Guidelines Defined by Medialt and IGDA.

Torcolini, N. (2010). Summer 2010 Improving Accessibility for the Blind on the Android Platform. *stanfordedu*, 1-6. Retrieved June 22, 2012, from http://www.stanford.edu/~elecator/MSWord_Final_Report.doc

Tuzun, H., Yilmazsoylu, M., Karakus, T., Inal, Y., & Kizilkaya, G. (2009). The effects of computer games on primary school students' achievement and motivation in geography learning. *Computers & Education*, *52*(1), 68-77. doi:10.1016/j.compedu.2008.06.008

US CensusBureau. (2002). Statistics on Disability. Retrieved June 22, 2012, from http://www.census.gov/hhes/www/disability/disability.html

US Government. (1998). *1998 amendment to section 508 of the rehabilitation act. In: SEC. 508. Electronic and Information Technology.*

Warschauer, M. (2010). Digital Divide. (R. Andrews & C. A. Haythornthwaite, Eds.)*Encyclopedia of Library and Information Sciences Third Edition*, *10*(755239602), 1551-1556. Taylor & Francis. Retrieved June 22, 2012, from http://tlc.nlm.nih.gov/resources/publications/sourcebook/adoptiondiffusion.html

Yuan, B., Folmer, E., & Harris, F. C. (2010). Game accessibility: a survey. (I. D. De Wolff, Ed.)*Universal Access in the Information Society*, *10*(1), 81-100. Springer Berlin / Heidelberg. Retrieved June 22, 2012, from http://www.springerlink.com/index/10.1007/s10209-010-0189-5

Anexo I

Terminología

Android: es un sistema operativo basado en *Linux*, por lo que dispone de un núcleo de sistema operativo libre, gratuito y multiplataforma, diseñado para ser utilizado en plataformas móviles como *smartphones, tablets, netbooks* y otros dispositivos. Fue desarrollado por *Android inc.* firma comprada por *Google* en 2005, momento en el que comenzó a desarrollar el popular sistema. En 2007 se creó la *Open Handset Alliance*, que agrupaba a un gran número de fabricantes de teléfonos móviles y *chipsets* al tiempo que *Google* proporcionaba la primera versión de Android junto con el SDK, para que los programadores empezaran a desarrollar aplicaciones para este sistema. Éste permite programar aplicaciones Java para la máquina virtual del sistema llamada *Dalvik*. En definitiva, se utiliza el lenguaje de programación Java para compilar un *bytecode* distinto de la conocida máquina virtual de Java de *Sun - Oracle*, aunque existen métodos de conversión entre los *bytecode* de ambas máquinas virtuales.

Versión	**Nombre**
Android 1.5	Cupcake
Android 1.6	Donut
Android 2.1	Eclair
Android 2.2	Froyo
Android 2.3 - Android 2.3.2	Gingerbread
Android 2.3.3 - Android 2.3.7	
Android 3.1	Honeycomb
Android 3.2	
Android 4.0 - Android 4.0.2	Ice Cream Sandwich
Android 4.0.3 - Android 4.0.4	

Tabla 11. Versiones de Android OS

Canvas: Es similar a un lienzo sobre el que el programador tiene el control para realizar todo tipo de acciones de dibujo.

Discapacidad Visual: es la cualidad de la persona con una privación parcial de la vista que no puede ser corregida adecuadamente con gafas convencionales, lentes de contacto, medicamentos o cirugía.

FPS(1): viene del inglés *First Person Shooter*. Es un género de juego de disparos que se centra en la perspectiva en primera persona desde el protagonista.

FPS(2): viene del inglés *frames per second*, es la medida de la frecuencia a la cual un reproductor o cualquier sistema que reproduce secuencias de imágenes genera distintos fotogramas o *frames*

Google Play: tienda de software en línea desarrollada por *Google* para los dispositivos *Android*. En ella se encuentran todas las aplicaciones Android firmadas por sus respectivos desarrolladores.

ICS (*Ice Cream Sandwich)*: nombre por el que es conocida la versión 4.0.x del sistema operativo *Android*. Es la primera versión que integra opciones de accesibilidad significativas.

iOS : sistema operativo móvil de *Apple* para dispositivos *Iphone, Ipod* y *IPad*, que rivaliza con *Android* por el dominio del mercado de los *smartphones*, proporcionando funcionalidades características similares a Android. Surgió prácticamente en el mismo momento que Android, a principios de 2008 y permite programar aplicaciones en Objective-C y C++.

iPad : dispositivo electrónico tipo *tablet* desarrollado por Apple.

Juego Flash: es un videojuego que se juega mediante un navegador web.

Joystick: dispositivo que se conecta con un ordenador o videoconsola para controlar de forma manual un software, especialmente juegos o programas de simulación.

Kinect: es un controlador de juego desarrollado para la videoconsola *Xbox 360* y para PC. Permite a los usuarios controlar e interactuar con la consola sin necesidad de tener contacto físico con un controlador de videojuegos tradicional, mediante una interfaz natural de usuario que reconoce gestos, comandos de voz y objetos e imágenes.

Log: término referido al registro de datos por parte de los usuarios.

Point and click: apuntar y hacer *click* es el método usado en el género de videojuegos conocido como aventura gráfica, que consiste en pulsar un botón del ratón sobre los objetos del juego para realizar las acciones del mismo.

RPG: viene del inglés *Role-Playing Game*. Es un juego en el que, tal como indica su nombre, uno o más jugadores desempeñan un determinado rol, papel o personalidad.

Scroll: es el evento de desplazamiento producido en pantalla mediante uno de los dedos normalmente en sentido vertical.

Smartphone: es un teléfono móvil construido sobre una plataforma de informática móvil, unida a la capacidad de computación avanzada y conectividad de un teléfono móvil. El término «inteligente» hace referencia a la capacidad de usarse como un pequeño ordenador personal, llegando incluso a reemplazar al mismo en muchas ocasiones.

Sprite: imagen 2D o animación.

Sprite sheet: conjunto de *sprites* combinados en una imagen más grande que componen las distintas partes de una o varias animaciones.

Tap, click: evento táctil producido sobre la pantalla del dispositivo móvil, que consiste en la realización de una pulsación sobre ésta.

Toast: es elemento del interfaz gráfico de usuario de *Android* que se usa habitualmente para indicar mensajes de notificación.

VoiceOver: se trata una solución integral de accesibilidad que utiliza *Apple* en sus dispositivos móviles, que va más allá de lo que es un lector de pantallas, ya que incorpora cambios en la interacción mejorando la accesibilidad en sus dispositivos.

Wii: videoconsola producida por Nintendo.

Xbox 360: videoconsola producida por Microsoft.

Anexo II

Directrices IGDA para desarrollo de juegos accesibles

Directriz	Descripción
Brújula por sonido	Para usuarios con baja visión, un sonido 3D representa el norte y mediante síntesis de voz en respuesta a pulsación de teclas las direcciones restantes.
No modos con gráficos 3D	Para jugadores ciegos, una opción que desactive el renderizado 3D en el menú inicial es importante para ellos porque probablemente no dispongan del hardware necesario para jugar al juego, ya que no lo necesitan. También una buena opción para los gráficos 3D es no usar aceleración por hardware, para evitar problemas con drivers gráficos anticuados que hagan inestable el juego. Esto permite que los usuarios ciegos no tengan que preocuparse por actualizar los drivers de sus tarjetas gráficas.
Orientación directa	Para ciegos, el uso de teclado numérico por un jugador ciego debería permitir orientar la dirección de su avatar en 8 direcciones.
GPS	En un juego accesible para ciegos, un sistema de posicionamiento global puede ser utilizado para obtener las posiciones concretas de los objetos en la zona además de la posición del avatar.
Sonar	Un sonido 3D puede al usuario ciego darle una ligera percepción de la distancia a los objetos en la dirección en la que se dirige el usuario. Pulsando una tecla el jugador puede comprobar el tipo de objeto. Los enemigos pueden ser automáticamente identificados por una voz.
Configuración de sonidos alternativos	Proporcionando sonidos alternativos puede ayudar a personas con dificultades de audición. Por ejemplo, proporcionando archivos de sonido que usen "vibración bass" del subwoofer para dar *feedback* a los usuarios sordos.
Modo alto contraste	Capacidad para modificar el contraste y otras características como la iluminación para ayudar a personas con baja visión a ver escenas más claramente.
Un interfaz simplificado opcional con controles alternativos	Para un juego con interfaz de usuario compleja, es necesario proporcionar un interfaz simplificado que sólo muestre los controles más utilizados comúnmente. La interfaz completa estaría todavía disponible, pero normalmente escondida al usuario. Esto ayudaría a aquellos con problemas de movilidad, en especial si el interfaz simplificado es más pequeño y requiere menos movimientos para navegar.
Mejorar soporte *hardware* para dispositivos especiales	Ratón, *joystick* o *game-pad* son soportados habitualmente en los juegos. Sin embargo, algunos usuarios con problemas de movilidad usan otro tipo de dispositivos especiales. Expandir el soporte de la aplicación para este tipo de dispositivos permitiría a muchos de ellos jugarlos.
Capacidad para configurar colores de los elementos del juego	La capacidad para controlar el color de los diferentes elementos del juego puede ayudar a personas con baja visión a identificar enemigos, compañeros y otras unidades importantes dentro de un juego.
Esquemas de colores para problemas de visión de	Proporcionando un esquema de colores alternativo puede permitir a aquellos con problemas de visión de colores a seleccionar los recursos gráficos que mejor les

colores	permitan un mejor visionado del juego.
Control más preciso de la dificultad	Permitir la modificación de la dificultad del juego más allá de lo habitual en juegos. Por ejemplo, para juegos en tiempo real, añadir un mecanismo que controle la velocidad o un botón que permita habilitar un sistema basado en turnos.
Gestión de síntesis de voz propia	La capacidad de proveer lectura de los textos que son mostrados en el juego beneficiaría a personas ciegas y con problemas de baja visión. Hay ya una gran cantidad de *software* que proporciona esta funcionalidad y que puede integrarse en juegos.
Controles personalizados	Mientras muchos juegos ya lo permiten, la posibilidad de configurar los controles sería útil para personas con movilidad reducida.
Mejores tutoriales	Esta característica sería útil para casi todos los jugadores. A mucha gente le gusta comenzar a jugar sin leer un manual. Guiarles a través del juego y proporcionarles *feedback* extra sería útil ya que ellos podrían comprender muchos de los aspectos de los juegos fácilmente. Especialmente en el caso de personas con problemas de aprendizaje, con problemas de concentración para la lectura de un enorme manual.
Control por teclado de todas las acciones con *feedback* visual y hablado	Permitir que todos los comandos sean introducidos mediante teclado. Con cada interacción se proporciona *feedback* visual o auditivo para indicar que se ha producido con éxito la acción. Esta característica ayudaría a jugadores con problemas de movilidad, visión o audición.
Presentación de textos estandarizada	*Microsoft* proporciona un lector de pantallas con su sistema operativo *Windows*. Para juegos cuya plataforma sea este sistema, tener texto que sea compatible con estos lectores de pantalla permitiría a personas con baja visión utilizar la herramienta proporcionada para la lectura del texto. El texto puede ser también usado por otro software como diccionarios especiales para jugadores con dislexia, que les ayudan a entender el texto.
Subtítulos	Cualquier tipo de diálogo que se reproduce en el juego debería mostrarse en forma de texto en la pantalla. Esto ayudaría a jugadores que son sordos o presentan dificultades en la audición. Esta opción puede ser configurable como una opción del juego.
Fuentes personalizables	Para cualquier texto mostrado en la pantalla debe existir la opción de editar los atributos de su fuente. Esto facilitaría la lectura a personas con baja visión.

Tabla 12. Directrices IGDA para desarrollo de juegos accesibles

Anexo III

Formulario de evaluación

1. ¿Qué juego o juegos va a evaluar?
   - [ ] Buscaminas
   - [ ] Golf
   - [ ] Zarodnik

2. Marque la opción que mejor describe su nivel visual actual
   - ○ Soy ciego
   - ○ Mi visión está limitada
   - ○ No tengo discapacidad visual

3. ¿Cuánto tiempo suele dedicar a jugar con ordenador o consola?. Donde 1 significa "Nunca" y 5 "Muy a menudo"
   - ○ 1
   - ○ 2
   - ○ 3
   - ○ 4
   - ○ 5

4. ¿Cuánto tiempo suele dedicar a jugar con dispositivos móviles (móvil o tableta)?. Donde 1 significa "Nunca" y 5 "Muy a menudo"
   - ○ 1
   - ○ 2
   - ○ 3
   - ○ 4
   - ○ 5

5. ¿Cuánto le ha costado aprender a jugar?. Donde 1 significa "Muy difícil" y 5 "Muy fácil".
   - ○ 1
   - ○ 2

- 3
- 4
- 5

6. ¿Cómo de divertida le ha parecido la experiencia de juego?. Donde 1 significa "Muy aburrido" y 5 "Muy divertido".
   - 1
   - 2
   - 3
   - 4
   - 5

7. ¿Cómo de entretenida le ha resultado el juego?. Donde 1 significa "poco entretenido" y 5 "muy entretenido".
   - 1
   - 2
   - 3
   - 4
   - 5

8. ¿Le parece cómoda la interacción con el dispositivo?. Donde 1 significa "incómodo" y 5 "muy cómodo".
   - 1
   - 2
   - 3
   - 4
   - 5

9. Evalúe la historia (si es aplicable). Donde 1 significa "No me ha gustado mucho" y 5 "Me ha gustado mucho".
   - 1
   - 2
   - 3
   - 4

- 5

10. Evalúe la accesibilidad y los ajustes de control. Donde 1 significa "No me ha gustado mucho" y 5 "Me ha gustado mucho".
    - 1
    - 2
    - 3
    - 4
    - 5

11. Evalúe la retroalimentación proporcionada en el juego. Donde 1 significa "No me ha gustado mucho" y 5 "Me ha gustado mucho".
    - 1
    - 2
    - 3
    - 4
    - 5

12. Finalmente, proporcione algunos comentarios o sugerencias para mejorar el juego.

Anexo IV

Formatos soportados por BF7G Toolkit

Este apartado describe los códec de audio y los formatos de compresión soportados por el conjunto de herramientas desarrollado durante este proyecto.

Tipo	Formato / Códec	Tipo de archivo soportado(s) / Formatos
Audio	AAC LC/LTP	• 3GPP (.3gp) • MPEG-4 (.mp4, .m4a) • ADTS raw AAC (.aac) • MPEG-TS (.ts)
	HE-AACv1	
	HE-AACv2	
	AMR-NB	3GPP (.3gp)
	AMR-WB	3GPP (.3gp)
	FLAC	FLAC (.flac) only
	MP3	MP3 (.mp3)
	MIDI	• Tipo 0 y 1 (.mid, .xmf, .mxmf) • RTTTL/RTX (.rtttl, .rtx) • OTA (.ota) • iMelody (.imy)
	Vorbis	• Ogg (.ogg) • Matroska (.mkv, Android 4.0+)
	PCM/WAVE	WAVE (.wav)
Imagen	JPEG	JPEG (.jpg)

	GIF	GIF (.gif)
	PNG	PNG (.png)
	BMP	BMP (.bmp)
	WEBP	WebP (.webp)

Tabla 13. Formatos soportados por BFG Toolkit