UNIVERSIDAD REY JUAN CARLOS

ESCUELA SUPERIOR DE INGENIERÍA INFORMÁTICA

INGENIERÍA TÉCNICA EN INFORMÁTICA DE GESTIÓN

Curso Académico 2009/2010

Proyecto de Fin de Carrera

Juego adaptativo para la ayuda a los desplazamientos en transporte público en superficies *"multitouch"* utilizando Fling

Autor: David Roldán Álvarez

Tutores: Estefanía Martín Barroso

Resumen

Muchas acciones de la vida cotidiana, como trabajar, estudiar, desplazarse en metro, comprar, etc.; pueden resultar muy fáciles para unas personas y ser demasiado complejas para otras. Las personas con necesidades especiales, en concreto, aquellas que tienen discapacidad cognitiva, tienen diversos problemas, entre los que podemos destacar su dificultad de comunicación, la falta de habilidades sociales, problemas en la toma de decisiones frente a situaciones previsibles o imprevisibles, etc.

En la actualidad desplazarse mediante transporte público es una de las tareas más complicadas que puede realizar una persona con discapacidad cognitiva. Sin embargo, la realización de esta tarea le hará gozar con total seguridad, de una mayor independencia y capacidad para desplazarse. Vamos a tratar de entrenar a estas personas tanto en el conocimiento de la red de Metro de la Comunidad de Madrid como en la toma de decisiones a la hora de viajar por el metro sin necesidad de ninguna ayuda extra. De esta manera, podrán superar una de las barreras con las que se pueden encontrar en la vida diaria. La primera forma de echarles una mano, es hacer que se familiaricen con la red de metro de forma interactiva, ya que así perderán el miedo. También, es necesario plantear situaciones esperadas e inesperadas entrenándoles en la toma de decisiones.

Con este proyecto, tratamos de crear un juego adaptativo (la aplicación realiza una serie de preguntas según el nivel de conocimientos del usuario) para ayudar a desplazarse en transporte público. Este juego se diseñará para que se ejecute en una mesa *multitouch* y pretenderemos que sea lo más interactivo posible. Esta aplicación se implementará mediante Fling, un entorno de desarrollo que permite crear aplicaciones para dispositivos *multitouch* mediante la programación en ActionScript.

Es un proyecto que requiere un alto nivel de abstracción, ya que lo estamos diseñando para gente con necesidades especiales, y por tanto tenemos que ponernos en su lugar a la hora de diseñar la aplicación. Un buen diseño se convierte en una característica muy importante en nuestro caso y una interfaz sencilla hace que el usuario aprenda rápidamente a manejar la aplicación.

ÍNDICE

ÍNDICE DE IMÁGENES

1.- Introducción

Esta sección muestra la motivación de este proyecto fin de carrera, las tecnologías involucradas tanto hardware como software, la metodología de trabajo y por último la estructuración de esta memoria

1.1 Motivación

Las personas con discapacidad intelectual muchas veces tienen dificultades para vivir independientemente porque no son capaces de realizar algunas tareas diarias como cocinar, comprar, ir al médico o coger el transporte público, entre otras. Mediante la creación de ciertos entornos para extender su independencia o para ayudarles a adquirirla, conseguimos que estas personas disfruten de una calidad de vida mejor.

Los ordenadores e Internet tienen un tremendo potencial para incrementar la independencia de las personas con discapacidad. Aquellos que tienen dificultades para abandonar sus casas pueden acceder a Internet para realizar sus compras, quedar con sus amigos, etc. En un pasado, las personas ciegas debían esperar mucho tiempo a que la información les llegase en formato Braille. Ahora pueden acceder a ella a través de Internet mediante audio y pueden interactuar utilizando teclados Braille. Las personas con problemas psicomotrices que tenían problemas a la hora de utilizar el teclado o el ratón de un ordenador, ahora tienen los más avanzados dispositivos de reconocimiento de voz para realizar tareas y poder interactuar con estos dispositivos tecnológicos.

En lo que a nuestro tema se refiere, el transporte público, podemos decir que se trata de uno de los sistemas más complejos que podemos encontrar en la sociedad moderna. Para aquellas personas que no puedan conducir debido a sus dificultades, este medio es la puerta para poder participar en actividades, socializarse y a fin de cuentas, ser independiente. Por ello, sería muy útil tener aplicaciones informáticas que permitan a estas personas familiarizarse con la red de metro de la Comunidad de Madrid, para que así, a la hora de la verdad sepan orientarse y tomar decisiones frente a posibles imprevistos.

Hemos de tener en cuenta que las personas con discapacidades cognitivas tienen más problemas a la hora de realizar determinadas actividades que las personas que no sufren ninguna discapacidad. Algunos procesos cognitivos afectados son los procesos de memorizar, prestar atención, comprender conceptos de forma visual y verbal, etc.

Además, para algunas personas con discapacidad cognitiva, desplazarse hasta su lugar de trabajo y volver, puede ser un gran problema que supondrá una barrera de integración. Por tanto, el no poder usar el transporte público de una forma segura, puede dañar su calidad de vida. Este hecho no sólo afectará a sus estudios o trabajo, si no que también afectará al uso de su tiempo libre.

Este proyecto coincide también con el auge de una tecnología que nos va a permitir una mayor interacción con los ordenadores y otros muchos más accesorios electrónicos. *Multitouch* es el nombre con el que se conoce a una técnica de interacción persona-ordenador y al *hardware* que la implementa. La tecnología *multitouch* consiste en una superficie táctil que reconoce simultáneamente varios puntos de contacto, así como el *software* asociado a ésta que permite interpretar dichas acciones.

1.2 Tecnología

En este apartado describimos brevemente los aspectos tecnológicos más importantes para el desarrollo de nuestro proyecto. Hay una descripción más detallada en el apartado 2 de esta memoria.

1.2.1 *Hardware*

Las superficies *multitouch* pueden ser de origen capacitivo, es decir, al presionar con los dedos se produce una corriente eléctrica continua que se conduce a través de un sensor .La característica más importante es que permiten obtener la posición de varios puntos de contacto sobre ella de manera simultánea. A menudo también puede calcular la presión o el ángulo de cada uno de los puntos de contacto de forma independiente, lo que permite hacer gestos e interactuar con varios dedos o manos de manera simultánea y proveer así de una interacción más rica y atractiva a través de gestos mucho más intuitivos.

En nuestro caso, esta tecnología nos interesa, ya que, en primer lugar queremos que varios usuarios puedan interactuar a la vez con la aplicación desarrollada. A su vez, las acciones que realicemos (seleccionar, acercar, alejar, mover, rotar) se realizan de una forma más directa con las propias manos que si las realizásemos con el teclado o con el ratón de un ordenador personal.

Sabiendo que la tecnología *multitouch* hace que los usuarios estemos más implicados con la aplicación que se está ejecutando, ¿por qué hoy en día está tecnología es tan desconocida y no la encontramos fácilmente? Aunque distintos dispositivos móviles, *iphones*

y demás pequeños aparatos van incorporando superficies *multitouch*, fabricar estas superficies a una escala mayor es hoy por hoy bastante caro. Por ejemplo, la mesa *multitouch* que ofrece Microsoft sobrepasa los 10000 dólares y otras pantallas como las ofrecidas por PQLabs (de 30'' por ejemplo) cuestan más de 2000 euros.

1.2.2 *Software*

No todo consiste en *hardware* y superficies capaces de reaccionar a la presión ejercida por los dedos. Detrás de todo esto, hay un *software* que permite interpretar los gestos. Además, para generar el *software* necesitamos una herramienta de acuerdo con la aplicación concreta que queremos implementar.

Los *frameworks* de *multitouch* consisten básicamente en dos módulos. El primer módulo se encarga de la interpretación de eventos, mientras que el segundo se encarga de emitir los gestos. Para el primer módulo, las señales de la entrada del interfaz son recibidas y traducidas en eventos atómicos. La mayoría de las herramientas reciben mensajes mediante el protocolo TUIO. Los eventos atómicos son interpretados y pasados a lo largo de la aplicación mediante rutinas específicas de la API, que varían de un *framework* a otro. Dependiendo de las posibilidades de la API la aplicación puede ser más o menos orientada a superficies *multitouch*.

Es importante mencionar, en qué consiste la tecnología Flash, que es la base en el desarrollo de nuestro proyecto. Flash es la tecnología más común utilizada para la creación de aplicaciones para Webs. Permite llevar a cabo animaciones de poco peso y, por lo tanto, son rápidamente cargadas por nuestro navegador.

Además, Flash ofrece en su entorno la posibilidad de interaccionar con el usuario. Para ello, Flash invoca un lenguaje de programación llamado ActionScript. Este lenguaje es orientado a objetos y tiene claras influencias de JavaScript. De este modo, se pone a nuestra disposición una tecnología pensada para aportar vistosidad a nuestra aplicación al mismo tiempo que nos permite interaccionar con el usuario.

1.3 Método de trabajo

El método de trabajo que se ha seguido para el desarrollo de este proyecto ha sido el siguiente:

a. Recopilación de información acerca de las necesidades de las personas con discapacidad cognitiva incluyendo la búsqueda de las necesidades especiales y la forma de facilitarles las tareas...
b. Estudio sobre las características de las mesas *multitouch.*
c. Descripción del problema, análisis de requisitos, diseño e implementación de la aplicación.
d. Estudio de las funcionalidades de las que provee la librería Fling.
e. Recopilación de información del producto Adobe Flash CS4, que nos va a permitir implementar la aplicación.
f. Crear los ficheros *.xml* con los datos de las estaciones y los arcos entre ellas. Esto nos sirve para que la aplicación lea la información almacenada de forma persistente en estos ficheros y cree el grafo correspondiente a la red de Metro de la Comunidad de Madrid.
g. Realización de las pruebas para comprobar el correcto funcionamiento de la aplicación informática en ordenadores personales y posteriores pruebas en una mesa *multitouch.*
h. Análisis de las pruebas.
i. Incrementos en la funcionalidad de la aplicación. (Crear nuevas preguntas que ayuden al usuario a entender mejor la línea de metro).

1.4 Estructura de la memoria

En el apartado 1, "**Introducción"**, se describe en qué se ha motivado este proyecto, presentando los objetivos y el método de trabajo seguido en su desarrollo. Además, se exponen brevemente las tecnologías utilizadas, tanto *hardware* como *software.*

En el apartado 2, "**Estado del arte"**, comienza explicando en qué consiste la programación orientada a objetos, incluyendo los conceptos fundamentales y las características que nos ofrece este paradigma. A continuación se explica brevemente en que consiste la tecnología multitouch explicando los conceptos de software propietario y software libre. Además explicamos que nos ofrece la librería Fling y por qué se utiliza en este proyecto. Por último, nos centramos en las herramientas utilizadas para el diseño de la aplicación: El lenguaje de programación ActionScript 3.0, el entorno Adobe Flash CS4 y el entorno de ejecución multiplataforma Adobe AIR.

En el apartado 3, "**Descripción informática"**, se describe como se ha realizado el diseño de la aplicación. En esta sección incluiremos diagramas que explican la interacción entre las clases, descripciones de los datos que maneja la aplicación, descripción de las preguntas que la aplicación hace al usuario, requisitos de la interfaz y un ejemplo visual de la ejecución de la aplicación.

En el apartado 4, "**Conclusiones**", describimos cuales han sido los logros alcanzados y las dificultades que nos han ido surgiendo a lo largo del desarrollo del proyecto. Por último hablamos sobre los posibles trabajos futuros que pudieran realizarse a partir de nuestra investigación.

A continuación se presenta la bibliografía y por último, aparecen una serie de anexos acerca de la instalación de las herramientas utilizadas, sobre la generación de un fichero .air y la instalación de este fichero en otros ordenadores. Este último paso es importante si se quiere probar la aplicación sin disponer del código fuente ni de Adobe Flash CS4.

2.- Estado del arte

El objetivo de esta sección es realizar una recopilación y un análisis previo de todos los elementos tecnológicos que intervienen en nuestra aplicación, desde concepto de programación orientada a objetos hasta la herramienta Adobe Flash CS4.

2.1 Programación Orientada a Objetos

La programación Orientada a Objetos o POO, es un paradigma de programación que usa objetos y sus interacciones para diseñar aplicaciones y programas de ordenador. Está basado en distintos conceptos incluyendo la herencia, abstracción y polimorfismo. Su uso empezó a ser importante en la década de 1990. En la actualidad, hay varios lenguajes que soportan programación orientada a objetos. En este caso, se ha utilizado ActionScript 3.0.

2.1.1. Conceptos fundamentales

- Clase: Describe el comportamiento y los atributos de los objetos a los que representa.
- Objeto: Son entidades *software* que poseen un conjunto de atributos los cuales definen el estado del mismo y una serie de métodos que especifican su comportamiento. Son las instancias de una clase determinada.
- Método: Código asociado a un objeto que se ejecutará tras la recepción de un mensaje. Un método especifica lo que un objeto puede hacer, es decir, su comportamiento.
- Evento: Es un suceso en el sistema.
- Mensaje: Es la solicitud de un servicio a un objeto.
- Atributo: Son los datos asociados a un objeto que definen su estado.

2.1.2. Características de la POO

- Abstracción: Proceso mental mediante el cual se extraen las características esenciales de un concepto, ignorando los detalles superfluos.
- Encapsulación: Proceso mediante el cual se ocultan los detalles de las características de una abstracción.

- Modularización: Proceso de descomposición de un sistema en un conjunto de elementos independientes y con alto índice de significado propio.
- Jerarquización: Proceso de estructuración por el que se organizan un conjunto de elementos en distintos niveles atendiendo a determinados criterios.

2.2. Tecnologías *multitouch*

Este es el nombre con el que se conoce a una técnica de interacción persona-ordenador y al *hardware* que la implementa. Como se ha indicado en la sección anterior, la tecnología *multitouch* consiste en una superficie táctil que reconoce simultáneamente múltiples puntos de contacto, así como el *software* asociado a ésta el cual permite interpretar dichas interacciones simultáneas. Podemos clasificar este tipo de superficies *multitouch* en dos categorías principales dependiendo de si el *software* utilizado por las mismas es libre o propietario.

2.2.1 *Software* propietario

Este tipo de mesas, funcionan a partir del software que el fabricante va desarrollando. No se puede modificar el *software* existente para las mesas *multitouch* que poseen este tipo de *software*. Existen varios tipos de mesas *multitouch* en el mercado actual como por ejemplo Microsoft Surface, mTouch...

2.2.2 *Software* libre

Software que permite a los usuarios utilizarlo, estudiarlo, actualizarlo y redistribuirlo, según lo que la licencia indique. Las mesas *multitouch* que permiten utilizar software libre o a medida, son las que nos interesan en la realización de nuestro proyecto.

2.2.3 Tipos de mesas *multitouch* actuales

- *Microsoft Surface*[1]: Una mesa *multitouch* de carácter propietario. Incluye un SDK también propietario para el desarrollo de aplicaciones compatibles. Esta mesa proporciona interacción directa, experiencia multiusuario y reconocimiento de objetos.

[1] *Microsoft Surface:* http://www.microsoft.com/surface/en/us/default.aspx

- *C-base Multitouch Console*[1]: Una mesa *multitouch* basada en la librería de código abierto libavg. Un proyector en la mesa muestra una imagen en la superficie de la misma, y una cámara infrarrojos es utilizada para captar los movimientos de los dedos que tocan la superficie.
- *Reactable*[2]: Una superficie *multitouch* especializada en la identificación de objetos fiduciales asociados a elementos musicales. Esta mesa es un nuevo instrumento musical electrónico que permite crear y representar música. Ofrece una interacción muy intuitiva, es fácilmente customizable y tiene una gran cantidad de posibilidades.
- NUI Group[3]: Representa una de las comunidades más activas en temas de interacción *multitouch.* Colaboran en la creación de técnicas de código libre para percepción sensorial orientada a aplicaciones educativas y artísticas. Entre el trabajo producido destacan los proyectos Community Core Vision, TouchLib y OpenTouch.
- *DiamondTouch*[4]: Una mesa multiusuario y *multitouch,* manipulable mediante las manos y enfocado a la colaboración de grupo. Se puede interactuar simultáneamente al estar preparada para detectar, mediante sensores, la pertenencia de cada dedo sobre la superficie.
- Mesas a medida: Hoy en día es posible construir una mesa a medida que nos permitirá desarrollar aplicaciones a específicas. El precio de las mismas oscila alrededor de los 4000 euros. Este tipo de mesa formará parte del entorno de pruebas de la aplicación desarrollada en este proyecto. La mesa que utilizamos ha sido construida por el laboratorio AmiLab de la Universidad Autónoma de Madrid.

2.2.3 *Frameworks* actuales

- *reacTIVision*[5]: Es un *framework* de código libre y multiplataforma especializado en identificar códigos fiduciales para su uso como parte de los datos de

[1] *C-base Multitouch Console:* http://metalab.at/wiki/The_Multitouch_Console_/_c-base
[2] *Reactable:* http://www.reactable.com/products/reactable_experience/reactable/
[3] NUI Group: http://nuigroup.com/
[4] *DiamondTouch:* http://www.merl.com/projects/DiamondTouch/
[5] *reacTIVision:* http://reactivision.sourceforge.net/

entrada de las aplicaciones *multitouch*. Su foco son las aplicaciones de manejo mediante objetos tangibles.

- *Libavg*[1]: Es una plataforma multimedia con soporte para seguimiento mediante el uso de cámaras. Es la única librería conocida que soporta nativamente no sólo la interacción a través de los dedos, si no también de las manos. Usa el protocolo TUIO para la comunicación entre el *driver* de seguimiento de dedos y manos, y las aplicaciones de interfaz de usuario creadas usando este *framework*.

- *Multi-Touch lib T-Labs*[2]: Es una librería de licencia pública desarrollada por *Deutsche Telekom Laboratories*. Incluye una serie de rutinas de manipulación de objetos virtuales que permiten realizar transformaciones de posición, rotación y escala de forma sencilla. Es capaz de emitir eventos mediante el protocolo TUIO para aplicaciones compatibles.

- *TouchLib*[3]: Se trata de uno de los primeros proyectos de NUI Group. Proporciona un proceso básico de *blobs* (puntos de sombra/iluminación que generan los dedos al hacer contacto con las superficies multi-contacto, y que son captados por la cámara del sistema) para sistemas basados en tecnología FTIR y DI.

- VVVV[4]: Es un *toolkit* para desarrollo rápido de prototipos que hace uso de la programación visual para asociar blobs captados con varios métodos de visualización. Es especialmente adecuado para grandes entornos multimedia con interfaces físicas, gráficos en tiempo real y múltiples usuarios.

- FLING[5]: Es un *framework* creado en el laboratorio AmILab de la Universidad Autónoma de Madrid para el desarrollo de aplicaciones multi-contacto usando Adobe Flash, Adobe Air o Adobe Flex. Proporciona eventos singulares de puntos de luz sobre la mesa, así como eventos de gestos generados mediante una capa de interpretación. Su estructura modular está enfocada a la inclusión de múltiples métodos de interacción para que las aplicaciones resultantes puedan ser manipuladas independientemente del contexto. Implementa el reconocimiento de gestos con múltiples dedos, un tapete virtual para la co-existencia de varias aplicaciones usadas por varios usuarios simultáneamente, y reconoce objetos tangibles mediante el uso de

[1] Libavg: http://www.libavg.de/
[2] *Multi-Touch lib T-Labs: http://code.google.com/p/multitouch/*
[3] *TouchLib: http://nuigroup.com/touchlib/*
[4] VVVV: http://vvvv.org/tiki-index.php
[5] FLING: http://amilab.ii.uam.es/doku.php?id=fling

etiquetas fiduciales. Este es el *Framework* que utilizaremos en nuestro proyecto y que describimos de manera más detallada en el siguiente apartado.

2.3 Fling

En este contexto surge la librería Fling. El objetivo de este *framework* es ofrecer al desarrollador de una capa de abstracción que se ocupa de la interacción con el usuario, de manera que los esfuerzos estén centrados en trabajar con las intenciones del usuario. Al añadir movimientos independientes a la aplicación, se aumenta la complejidad, ya que se aumentan las posibilidades de que reaccione de forma indebida a las acciones del usuario, ya que no solo debemos controlar que la aplicación maneja los datos de forma correcta si no que también el usuario puede hacer click en un botón antes de que corresponda. Por ejemplo, en nuestro caso el usuario podría pulsar el botón para comprobar una respuesta antes de marcarla y esto tenemos que controlarlo ya sea no dejando que pulse ese botón hasta que marque una respuesta o que al pulsar el botón le salga un mensaje para que marque una respuesta antes.

Fling (*Flash Library for Interpreting Natural Gestures*) es una librería para la interpretación de gestos naturales realizados con los dedos de las manos sobre superficies horizontales *multitouch*. Esta librería ha sido creada por el laboratorio de Inteligencia Ambiental (AmiLab) de la Universidad Autónoma de Madrid. Esta librería permite desarrollar una aplicación que pueda ser ejecutable tanto en un entorno PC utilizando el teclado y el ratón como en un entorno *multitouch* utilizando las propias manos. Permite el reconocimiento de una gran cantidad de gestos como hacer click sobre un objeto, rotar un objeto, acercar o alejar un objeto, moverlo, etc.

La librería Fling puede ser utilizada para desarrollar aplicaciones Flash o AIR utilizando el lenguaje de programación ActionScript como su principal mecanismo de control. Flash nos ofrece avanzadas y potentes capacidades gráficas a la vez que tiene un modelo jerárquico para los objetos visuales que encaja en el sistema de propagación de eventos de Fling. Fling reacciona a la interacción estándar con el ratón y teclado desde ordenadores personales, y en las superficies *multitouch*, escucha los mensajes recibidos con el protocolo TUIO. Los eventos de entrada pueden ser desde simples "*clicks*" con el ratón hasta complejos gestos con varios dedos (por ejemplo, seleccionar dos puntos extremos del objeto y moverlo y agrandarlo a la vez). Los gestos representan las intenciones del usuario y estos pueden unirse para formar diferentes acciones.

Fling ha sido creado para facilitar el reconocimiento de gestos naturales realizados por los usuarios de superficies horizontales multicontacto. Por este motivo, ha sido necesario distinguir entre eventos, gestos y acciones. Los eventos dependen del mecanismo de interacción, los gestos son las intenciones del usuario y las acciones a realizar en respuesta a los gestos del usuario dependen de cada aplicación.

Gracias a toda la funcionalidad que ofrece Fling, sobre todo a la interfaz para el reconocimiento de gestos naturales del usuario (seleccionar un objeto, redimensionarlo, girarlo, etc.), y a que es *software* libre que se puede actualizar en el caso que se desee añadir más funcionalidad, se ha decidido utilizarlo para el desarrollo de este Proyecto Fin de Carrera.

2.4. Adobe Air

Es un entorno de ejecución multiplataforma para la construcción de aplicaciones RIA (*Rich Internet Aplication*) utilizando por ejemplo, Adobe Flash. Adobe Air intenta ser un entorno de ejecución versátil, ya que permite que el código Flash, HTML o JavaScript existente sea reutilizado para construir programas tradicionales de escritorio. Una aplicación de Adobe Air requiere que sea empaquetada, firmada digitalmente, e instalada en el sistema de archivos local de los usuarios. Esto último ofrece almacenamiento local ilimitado y acceso al sistema de archivos, mientras que otras aplicaciones desplegadas en un navegador Web están limitadas por las restricciones del mismo, y donde los datos, por lo general, se eliminan periódicamente. Sin embargo, en la mayoría de los casos, las aplicaciones Web almacenan los datos de los usuarios en sus propios servidores, pero la capacidad para consumir y trabajar con datos en el sistema de archivo local de un usuario permite una mayor flexibilidad cuando una aplicación está trabajando sin conexión.

2.5. ActionScript

Es un lenguaje de programación orientado a objetos. Es utilizado para desarrollar aplicaciones en el entorno de programación de Adobe Flash CS4. Este lenguaje permite añadir dinamismo a las aplicaciones desarrolladas. Fue lanzado con la versión 4 de Flash, y desde entonces hasta ahora, ha ido ampliándose poco a poco. ActionScript está basado en especificaciones de estándar de industria ECMA-262, un estándar para Javascript, de ahí que ActionScript se parezca tanto a Javascript. Es el lenguaje de programación que ha sido utilizado para el desarrollo de la aplicación.

2.5. Adobe Flash CS4

Es una plataforma multimedia que nos sirve para añadir animación, video e interacción a la aplicación. Además, nos proporciona los elementos necesarios para crear la interfaz directamente y que no tengamos que estar llamando a los elementos de la interfaz mediante código. De esta forma, se simplifica el trabajo y nos dedicamos plenamente a la interacción persona-ordenador. Ésta es la penúltima versión de la herramienta Adobe Flash (la última es Adobe Flash CS5 que salió en 2010).

3.- Descripción informática

Tras el gran crecimiento que ha ido experimentando el metro de Madrid en las últimas décadas, y la gran facilidad de desplazamiento que ofrece se hace casi imprescindible saber desplazarse a través de el. Ver el plano de la red e interpretarlo para seguir un camino determinado parece una tarea fácil de realizar. Sin embargo, si creyéramos esto, estaríamos olvidando a las personas que poseen algún tipo de discapacidad cognitiva, como las personas con síndrome de Down.

Las personas con discapacidad cognitiva tienen serias dificultades a la hora de retener conceptos, y de realizar algunas tareas cotidianas, entre ellas los desplazamientos en transporte público y la toma de deciones ante imprevistos. El metro es una red complicada en la que se cruzan distintas líneas y se puede llegar a un sitio de maneras diferentes. En este sentido, sería muy útil disponer de un método entretenido y a la vez didáctico, que entrenase a este tipo de personas en sus desplazamientos en la red de Metro. De esta manera, podrían habituarse a esta enrevesada red, y ser capaces de desplazarse sin ningún problema con total independencia.

El objetivo de este proyecto es ayudar a las personas con necesidades especiales, en concreto a personas con discapacidad cognitiva, a conseguir la confianza suficiente para desplazarse utilizando el transporte público (en concreto Metro de Madrid). De esta manera, serían autosuficientes y ayudaría tanto a la toma de decisiones como a la eliminación de barreras para su integración.

Para cumplir este objetivo se ha desarrollado una aplicación que permite realizar un proceso de aprendizaje interactivo sobre la red de metro y favorece el proceso de toma de decisiones. Esta aplicación consiste en un juego en el que el usuario tenga que responder una serie de preguntas sobre el metro (pudiendo mirar el plano en cualquier momento). El objetivo es aumentar el conocimiento de los usuarios para que retengan los conceptos y las ideas que necesitan saber para moverse en el transporte público. Además, se pretende que múltiples usuarios puedan jugar a la vez sobre una superficie *multitouch* utilizando para ello sus propias manos dándole mayor dinamismo a la aplicación y favoreciendo el desarrollo de habilidades sociales tales como aprender a comunicarse, relacionarse con los demás, trabajar entre iguales, etc, ya que al interactuar sobre una superficie *multitouch*, varios jugadores pueden interactuar a la vez. El hecho que el usuario participe activamente con la aplicación hace que aprenda mientras se encuentra interactuando con el juego.

La tecnología *multitouch* es un campo que cada año sigue avanzando y cada vez se está convirtiendo en un producto más a tener en cuenta en la relación persona-ordenador. Los usuarios cada vez quieren sentirse implicados en las actividades que realizan. Está demostrado que participar en el propio proceso de aprendizaje es más instructivo que simplemente observar al profesor en una clase magistral. Las superficies multitouch favorecen las relaciones entre compañeros y entre alumnos y profesor. Además la manipulación directa con las manos hace que la interacción sea más natural y cercana a la realidad.

Como objetivos específicos, se pretende conseguir que el usuario sea capaz de aprender y comprender ciertos conceptos relacionados con el uso del transporte público. Los conceptos que se trabajan en esta aplicación son la búsqueda de una estación específica, el lugar donde hacer transbordo, saber en que línea está la estación que busca, relacionar una estación con las de sus alrededores y saber escoger el camino más corto. Con el objetivo de afianzar conocimientos, es muy importante la repetición del mismo tipo de pregunta en el caso de usuarios con discapacidad.

Como se ha comentado en las secciones anteriores de esta memoria, para crear esta aplicación hemos usado el lenguaje de programación ActionScript, usado generalmente para Flash, aunque en nuestro caso hemos aprovechado su clara orientación a objetos para diseñar una aplicación en la que cada elemento y objeto controla su comportamiento y no depende de otros objetos para desarrollar su cometido. De esta manera podemos reutilizar estos objetos en otras aplicaciones (reutilización de código).

De esta manera conseguimos un diseño orientado objetos que simplifica la modificación y extensión del *software*, haciendo que la mayor parte del mismo sea reutilizable. Además, es más fácil de entender ya que existe relación directa entre la estructura del software y la del problema.

Un aspecto muy ventajoso de la programación orientada a objetos, es la herencia, mediante la cual conseguimos que un atributo herede todos los atributos y operaciones de una clase. De esta forma, a partir de la clase FlingObj (que hereda de MovieClip), que es una de las clases principales que nos proporciona Fling y que básicamente representa un objeto que aparece en la interfaz (como por ejemplo, un botón), podemos crear una serie de objetos que actúan ante las mismas señales pero reaccionan a ellas de manera distinta. Así conseguimos ahorrar una cantidad ingente de código y hacer una aplicación más comprensible.

Para programar con el lenguaje ActionScript hemos utilizado la plataforma Adobe Flash CS4, que aparte de ofrecernos un gran soporte, nos permite crear la interfaz y los

elementos que aparecen en ella de una forma muy directa sin necesidad de hacerlo a través de código.

La aplicación desarrollada consistirá en una serie de preguntas a la que el usuario tendrá que responder seleccionando la respuesta que estime que es la correcta. Serán preguntas generadas al azar, con cuatro respuestas cada una (una correcta y tres incorrectas). En todo momento, el usuario tendrá la posibilidad de pulsar sobre un botón para consultar el plano de metro de Madrid. De esta manera, podrá buscar la respuesta correcta, cerrar el plano, y responder a la pregunta formulada en el juego.

Se partirá de un nivel inicial y la aplicación irá actualizando el nivel de dificultad dependiendo de las habilidades y conocimientos de los usuarios. El criterio de actualización de la dificultad será el siguiente: Cada vez que el usuario acierte 3 preguntas consecutivas se subirá la dificultad de las preguntas realizadas. Si por el contrario, el usuario falla tres preguntas seguidas, se bajará el nivel de forma transparente.

Al estar la aplicación diseñada para su uso en mesas *multitouch*, el usuario va a tener que "jugar" con la aplicación utilizando sus manos, ya que no va a disponer a priori ni de teclado ni de ratón. Esto hace que el usuario esté más involucrado en el juego.

A su vez, en el juego han de participar dos jugadores, cada uno con sus correspondientes preguntas y respuestas, de manera que se establezca una especie de competición en la que ambos usuarios salen beneficiados. Además, realizar algo en compañía es mucho mejor que hacerlo solo.

Por último, se ha tratado de crear una aplicación didáctica pero a la vez entretenida, cuyo objetivo es conocer más a fondo el metro de Madrid y así conseguir realizar recorridos en el menor tiempo posible, coger confianza en el uso del transporte público y en definitiva, ser más autónomos.

3.1.-Diagramas

El diagrama de clases de la figura 1 muestra una representación resumida de las jerarquías entre las clases y sus dependencias.

La clase principal es la clase concurso, que contiene BotonDifFacil, BotonDifNormal y BotonDifDificil para establecer la dificultad de las preguntas que se van a realizar en la aplicación. A su vez, contiene la clase preguntas que es la encargada de gestionar las preguntas que se realizan, las respuestas que se proponen, etc. Además, contiene la clase close (Al pulsar sobre el botón de la instancia se cierra la aplicación), los botones de respuesta (4 en

total), el botón para comprobar si la respuesta es correcta, un botón para consultar el plano del metro, el botón para comenzar una nueva partida. Con la clase Respuestas, generamos las respuestas correspondientes según el enunciado que se haya generado.

Por último queda hablar de la clase PlanoMetro, que contiene la instancia de la clase CerrarPlanoMetro (utilizada para cerrar el plano del metro una vez abierto) y un atributo loader que servirá para cargar la imagen del plano del metro.

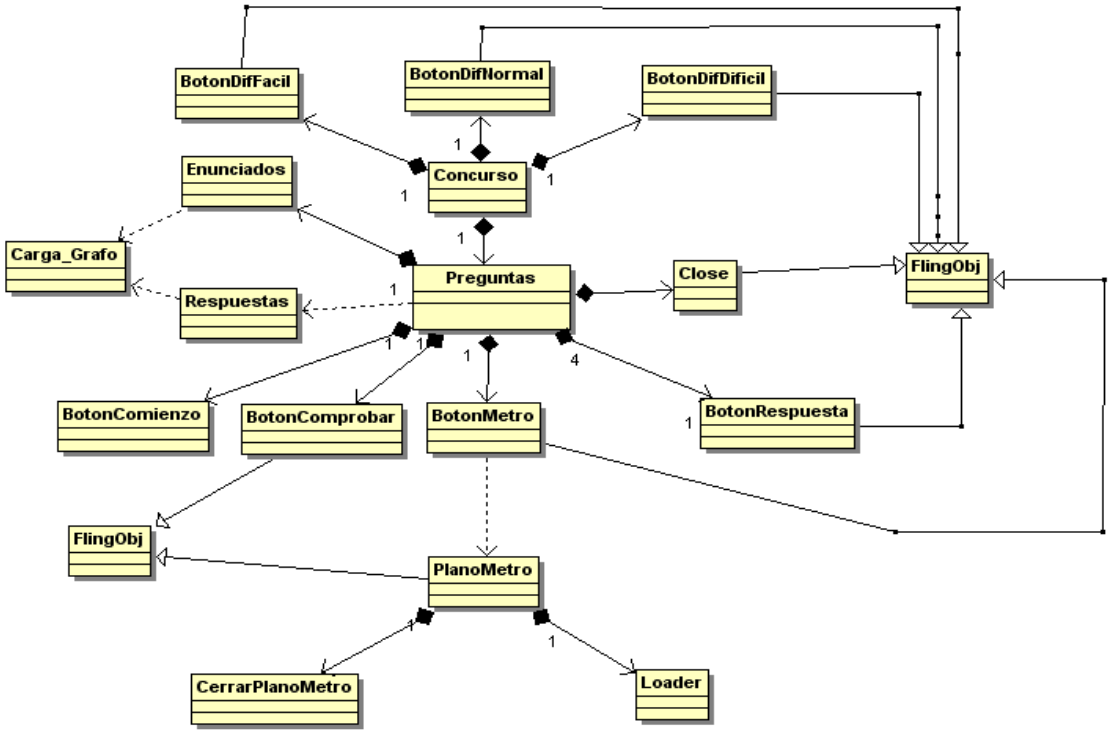

Figura 1. Diagrama de clases

Queda decir que todos los botones derivan de la clase FlingObj, proporcionada por la librería Fling, que es la que nos permite que esta aplicación se pueda ejecutar en una mesa *multitouch.*

Respecto a la secuencia de acciones seguidas por un usuario cuando se encuentra interactuando con la aplicación, en la figura 2 se puede ver un diagrama de secuencia relacionado con estas interacciones.

Como se puede observar el primer paso del usuario es hacer click en el botón COMIENZO. Tras esta acción, le aparecerán por pantalla al usuario los tres botones con las correspondientes dificultades y en ese caso, el usuario seleccionará la dificultad fácil. Tras pulsar en el botón para seleccionar la dificultad se generan las preguntas. En un primer lugar

se genera el enunciado, después la respuesta correcta y por último las tres respuestas incorrectas. Tras generarlo todo se escriben en los diferentes botones para que el usuario lo vea.

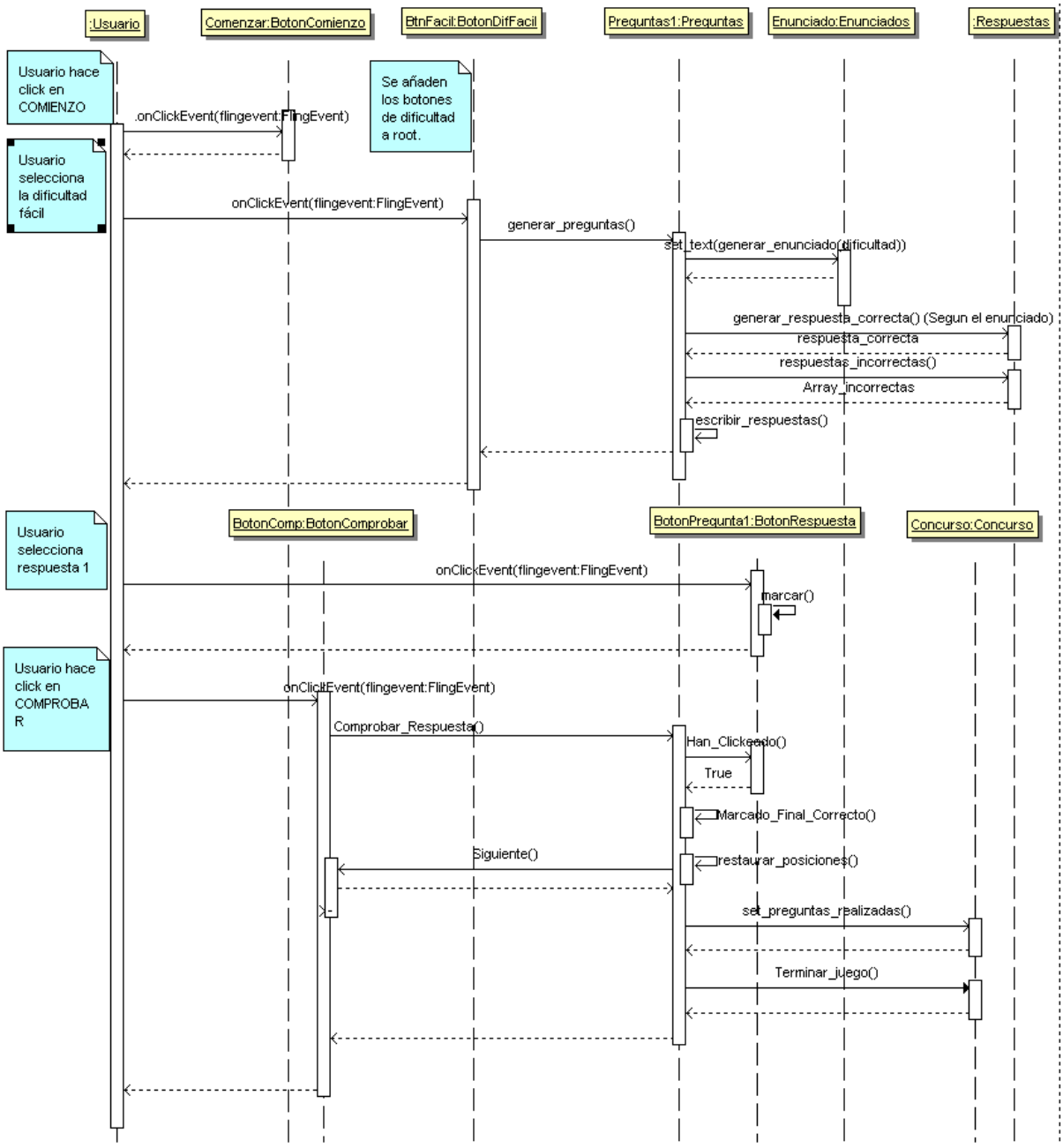

Figura 2. Diagrama de secuencia

Ahora es el turno del usuario, que puede marcar una de las cuatro respuestas posibles. Una vez marcada la respuesta el usuario pulsaría el botón COMPROBAR, que sirve para comprobar si una respuesta es la correcta. La función que comprueba si una respuesta es correcta comprueba si hay alguna respuesta marcada, reinicia las posiciones de las respuestas, comprueba si la respuesta marca (en caso de que la haya) es igual a la respuesta correcta,

comprueba si se han realizado todas las preguntas y cambia el botón COMPROBAR a SIGUIENTE, para poder generar otra pregunta.

3.2.-Tipos de datos

Para el correcto funcionamiento de la aplicación, en primer lugar, se necesita almacenar los datos de todas las estaciones de la red de metro con sus respectiva información se almacenan en un grafo. Una estación es un nodo del grafo que tiene asociada la información sobre las líneas a las que pertenece, el nombre de la estación y un identificador numérico (ver figura 3).

```
<metro>
      <estacion>
               <nombre>Parque_Lisboa</nombre>
               <id>1</id>
               <linea>12</linea>
      </estacion>
      <estacion>
               <nombre>Puerta_del_Sur</nombre>
               <id>2</id>
               <linea>12,10</linea>
      </estacion>
            ...
</metro>
```

Figura 3. Datos de las estaciones de metro

Es importante almacenar las uniones (arcos) que hay entre los diferentes nodos (estaciones). Para ello hemos creado un fichero .xml que contiene la información necesaria para crear estos arcos. Utilizamos el dato de la etiqueta "<id>" de cada estación para crear los arcos (ver figura 4).

```
<arcos>
      <union>
            <id1>1</id1>
            <id2>2</id2>
      </union>
      <union>
            <id1>2</id1>
            <id2>3</id2>
      </union>
                  ...
</arcos>
```

Figura 4. Ejemplo de datos para crear los arcos entre las estaciones

Además, es necesario que almacenemos tanto la respuesta correcta como las tres respuestas incorrectas que se generan cada vez que se realiza una pregunta. La respuesta dada por el usuario se almacena con el objetivo de que junto con el enunciado y las respuestas generadas se forma un fichero a modo de log. En este fichero tendremos toda la información de las preguntas y las respuestas que nos servirá para detectar algún posible error.

Debido a que la aplicación actualiza dinámicamente el nivel de dificultad de las preguntas mostradas según las habilidades y conocimientos del usuario, es necesario que guardemos en una variable el numero de respuestas correctas seguidas (con 3 se aumenta la dificultad) y el numero de respuestas incorrectas seguidas (con 3 disminuye la dificultad). Dicho esto, es necesario que la aplicación guarde también el valor de la dificultad actual.

En el proceso de generación de las preguntas, se crean líneas y estaciones que se guardan con el objetivo de utilizarlas para producir las respuestas.

3.3.-Tipos de pregunta

En este apartado se detallan los tipos de preguntas que se realizan al usuario y que posteriormente tendrá que responder. Hay que tener en cuenta, que el usuario tendrá acceso en todo momento a un plano del metro y que podrá visualizarlo pulsando sobre el botón Ver Plano Metro. De esta forma le damos un apoyo visual. Se han elegido estas preguntas ya que son con las que más información se aprende ya que el juego consiste en adquirir destreza a la hora de interpretar el plano del metro y no solamente memorizarlo.

Los tipos de preguntas con los que cuenta la aplicación son los siguientes:

1) *¿A que líneas pertenece X?*: Con esta pregunta hacemos que el usuario relacione una estación con sus respectivas líneas. De esta forma cuando el usuario sepa el nombre de una estación, al saber en que línea se encuentra, tendrá más facilidad a la hora de saber donde buscar.
2) *¿Cuál de estas estaciones está al lado de X?:* Es muy común que al mencionarnos una estación nos suene porque sabemos que está cerca de otra. Lo que intentamos con esta pregunta es que aunque el usuario no sepa donde está una estación, pueda deducirlo a través de otras que sabe que están cerca de ella
3) *¿Cuál de estas estaciones pertenece a la línea Y?:* Con esta pregunta queremos lograr el mismo objetivo que con la pregunta 1, salvo que esta vez utilizaremos un enfoque diferente. Con esta cuestión obligamos al usuario a que recorra una línea buscando la

estación que aparece en las respuestas que pertenece a esta línea, de tal forma que inconscientemente vaya aprendiendo otras estaciones de la misma línea.

4) *¿A cuántas paradas se encuentra X de Z?:* Tal vez esta sea la pregunta más difícil de todas. En la línea de metro puede haber muchos trayectos posibles desde una estación a otra y no siempre es sencillo encontrarlo. El camino más corto será aquel que pase por el menor número de estaciones.
5) *¿Si estas en la línea Y, en cual de estas estaciones hay que efectuar el transbordo para llegar a la línea W?:* Con esta pregunta intentamos que el usuario se aprenda a que líneas puede acceder si se encuentra en una línea determinada y a ser posible, en que estación debe hacer el trasbordo. Obligamos al usuario a situarse en una línea, y a partir de ahí, la recorre en busca de la estación que le permite hacer el trasbordo.
6) *Si estás en la estación X, ¿en cuál de estas estaciones debes efectuar el transbordo para llegar a la línea Y?:* Con esta pregunta intentamos que el usuario aprenda lo mismo que con la anterior, pero desde un punto de vista diferente. Esta cuestión es más difícil que la anterior, ya que encontrar una estación es más complicado que buscar una línea. Además, una vez que se localiza la estación se debe decidir que camino elegir para buscar la estación que nos va a llevar a la línea Y. Con esta pregunta el usuario investiga más a fondo en el plano del metro.

En nuestra aplicación, si la dificultad es fácil, solo se efectuaran las preguntas desde la uno hasta la tres (ambas inclusive) y, además, sólo se utilizarán los datos de las estaciones de las líneas 1 a la 4. En la dificultad media, ya se efectúan todas las preguntas y los datos se cogerán de las líneas 1 a la 8. En la dificultad difícil, se efectúan todas las preguntas utilizando los datos de la linea 1 a la 12.

3.4.-Requisitos de la interfaz

Para diseñar una buena interfaz es necesario tener en cuenta los siguientes aspectos. El tiempo que el usuario necesite para aprender a utilizar la interfaz de esta aplicación no debe ser demasiado elevado, debe ser fácil retener la forma de interactuar, debe reducir el número de errores que pueda cometer el usuario a causa de la interfaz, debe proporcionar rapidez a la hora de realizar las tareas y debe sencilla de utilizar para el usuario.

Un aspecto importante a tratar, es el diseño: colores, posiciones... Es importante que el texto sea legible. Si tratamos con usuarios con necesidades especiales, puede haber problemas

con los colores, por ejemplo. A su vez, la distribución de los botones se ha hecho de forma que todos se vean bien, no hay botones escondidos, y en todo momento el usuario verá todas las acciones que puede realizar, aunque a veces estas no podrán realizarse ya que depende del estado del programa (por ejemplo, no puede comenzarse una partida si ya hay una comenzada.

En la figura 5, vemos un ejemplo de lo que el usuario ve cuando está interactuando con la aplicación. En el apartado 3.6, se describen más detalladamente los elementos de la interfaz utilizada en nuestra aplicación.

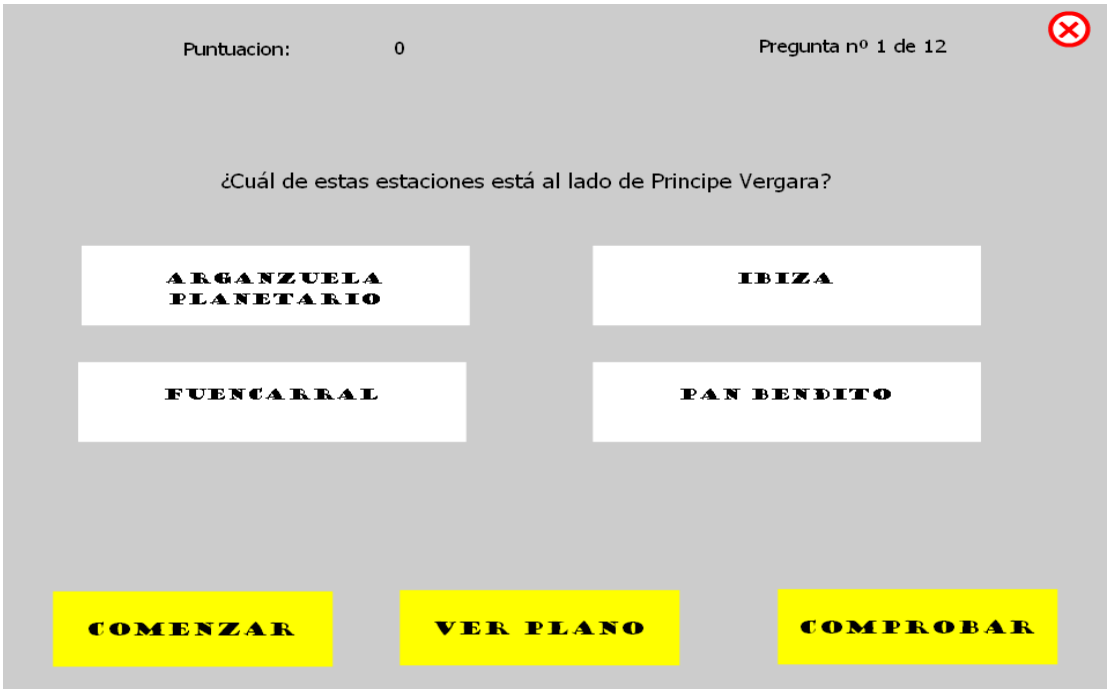

Figura 5. Interfaz

Para evitar errores típicos como equivocarse a la hora de marcar la respuesta correcta, el usuario tiene que pulsar el botón "comprobar una vez haya marcado la respuesta que cree que es la válida. De esta forma aunque se marque una respuesta, puede cambiarla tantas veces como quiera mientras no pulse el botón que comprueba si la respuesta es correcta.

Cada vez que el usuario responda una pregunta es necesario que haya realimentación. Esto lo conseguimos con un cuadro de texto que aparece debajo de los botones de las respuestas. En este cuadro se dirá al usuario si ha acertado la pregunta, si la ha fallado o si ha realizado una acción indebida.

Además, al estar la aplicación diseñada para personas con discapacidades cognitivas, es necesario que todos los elementos con los que interactúa el usuario sean visibles.

3.5.-Implementación

En este apartado se detallan las principales características y funcionalidad de cada una de las clases que forman parte de la aplicación desarrollada.

Clase Graph

Esta clase implementa la funcionalidad de un grafo. En la aplicación almacena toda la información sobre las estaciones, incluyendo datos como líneas a las que pertenecen, nombre de la estación, identificador y a que otras estaciones están unidas. Cada estación es un nodo en el grafo y gracias a esta clase conseguimos modelar la red del metro de Madrid.

Al tratarse de un grafo, disponemos de las operaciones elementales para operar con este: Crear un grafo, añadir un nodo, borrar un nodo, añadir un arco, borrar un arco, etc.

Hay un método de gran importancia para la implementación de la aplicación y es *breadthFirst* que implementa el recorrido en anchura de un grafo partiendo de un nodo que se le pasa por parámetro y una condición que también se pasa por parámetro. Con este método podemos calcular los caminos más cortos de un nodo a otro. En nuestro caso, es de vital importancia para solucionar los tipos de preguntas 4,5 y 6 en las que se pide el recorrido más corto de una estación a otra.

Los nodos del grafo son instancias de la clase GraphNode se almacenan en un array. También tenemos una serie de datos que almacenan el número actual de nodos en el grafo y el tamaño máximo de nodos que puede haber en el grafo.

Clase GraphNode

Con esta clase se representan los nodos del grafo, que sirven para simular las estaciones con todos los datos que las corresponden: nombre de la estación, un array cuyos elementos son las líneas a las que pertenece la estación y un identificador que corresponderá con la posición del array del objeto Graph en la que se encuentra el nodo.

Este objeto tiene la funcionalidad básica del nodo de un grafo. El constructor que sirve para crear un nuevo nodo, añadir un nuevo arco, borrar un arco y las funciones get y set para saber los valores de los atributos del nodo.

Clase GraphArc

Esta clase corresponde a los arcos que unen los nodos del grafo. Nos sirve para representar las uniones entre las diferentes estaciones.

Establecer bien las conexiones entre los diferentes nodos de nuestro grafo es un tema muy importante en el desarrollo de la aplicación ya que, un arco que no se halla creado puede dar lugar a que no se detecte un camino cuando este en realidad existe.

Por ello, se ha dedicado una cantidad importante de tiempo a comprobar la información almacenada en los ficheros *.xml* que contienen tanto los datos de las estaciones como los datos de los arcos existentes. Ha sido un proceso duro debido a la cantidad de información almacenada, pero es necesario para asegurar la calidad de nuestro *software* y el correcto funcionamiento del mismo.

Clase Concurso

Clase que controla el desarrollo general de la aplicación. Los datos más importantes son la puntuación del jugador y el número de preguntas realizadas. Como configuración inicial se ha decidido establecer que una partida contiene 12 preguntas.

Esta clase posee atributos que son los botones que permiten elegir la dificultad al principio de la partida (su funcionamiento se detalla más adelante).

En esta clase se encuentran las funciones que permitirán comenzar el juego y terminarlo. Además tenemos una función Aviso_Comienza_Partida() que avisa al usuario de que debe pulsar el botón Comienzo para comenzar el juego antes de realizar otra acción.

Terminar_Juego(), que es otra de las funciones importantes, se encarga de comprobar que el numero de preguntas realizadas ha llegado a 12. Si se han realizado 12 preguntas entonces la aplicación muestra la puntuación total del usuario y le da las opciones de comenzar una nueva partida o cerrar la aplicación.

Por último, esta clase posee una serie de funciones get y set para obtener los valores de los atributos en el caso de que sea necesario además de una serie de valores que reinician los atributos que guardan los datos del número de preguntas realizadas y la puntuación.

Clase Preguntas

Esta clase se encarga del control general de las preguntas que se formulan y las respuestas que el usuario marca. Se juntan las funcionalidades de otras clases para poder realizar la interacción con el usuario. El objeto que instancia esta clase engloba a todos los demás.

Es necesario crear un objeto que englobe a los demás para que fling pueda propagar correctamente los diferentes eventos que active el usuario con sus dedos. De esta forma, por

ejemplo, si queremos redimensionar la aplicación entera, al ejecutar la acción correspondiente sobre el objeto que instancia esta clase también se redimensionaran los objetos contenidos en ella.

También controlamos el número de respuestas correctas seguidas (si llega a 3 aumentamos en 1 la dificultad) y el número de respuestas incorrectas seguidas (si llega 3 reducimos en uno la dificultad). Este proceso es transparente para el usuario del juego.

El proceso que sigue esta clase para generar una pregunta es el siguiente:

1. Primero asignamos las posiciones en las que van a ir colocadas las respuestas (mediante las variables r1, r2, r3 y r4). Si por ejemplo r1 = 1 quiere decir que la respuesta 1 (la correcta) va a ir colocada en el BotonPregunta1. Si el valor fuera 3, pues iría colocada en BotonPregunta3.
2. Generamos el enunciado. Se genera un número aleatorio de 0 a número de preguntas -1. Si obtenemos 1 sacamos del array de la clase enunciados ese enunciado. (Explicamos este proceso con más detalle en la clase Enunciados).
3. Generamos la respuesta correcta y las tres respuestas incorrectas utilizando la clase Respuestas. (Explicamos con más detalle adelante).
4. Escribimos las respuestas en los botones según los valores de r1, r2, r3 y r4.

Una vez realizado este proceso esperamos a que el usuario haga click en una respuesta y la compruebe o haga cualquier otro tipo de acción como mirar el plano del metro.

En la clase Preguntas un atributo almacenará la cadena que forma la respuesta correcta, y por tanto, al realizar la comprobación de la respuesta se mirará si la respuesta elegida coincide con este atributo. Si es así, se le sumarán 5 puntos al usuario y se marcará la respuesta con un color verde. En caso contrario, se marcará la respuesta elegida en rojo y la respuesta correcta en verde.

Tras todo este proceso el usuario podrá hacer click en el botón con texto SIGUIENTE para generar una nueva pregunta y continuar el juego.

Clase Enunciados

Esta clase se va a encargar de generar el enunciado de la pregunta que se va a formular así como de guardar los parámetros usados (random_estacion, random_linea, etc) para su posterior uso a la hora de generar las respuestas. Tenemos una serie de enunciados predefinidos, que se guardan en un array. Para elegir un enunciado generamos un número aleatorio y sacamos el enunciado que se encuentre en la posición que dicta el número

aleatorio.Esta clase se instancia por el objeto Enunciado, que simplemente es un campo de texto en el que se muestran las preguntas.

Para generar un enunciado seguimos el siguiente proceso:

1. Generamos un número aleatorio (mediante el cual sacamos el enunciado básico).
2. Según el enunciado que salga, podremos necesitar generar una nueva estación o línea para completar el enunciado.
3. Una vez generadas las variables que necesitemos formamos el enunciado completo que será el que se muestre.

Esta clase guardará en sus atributos los valores generados para cada enunciado (estaciones, líneas, etc), ya que pueden ser muy importantes a la hora de generar la respuesta. Estos datos se guardan para no repetirlos en las respuestas y para partir de ellos a la hora de generar los caminos y las respuestas.

Clase Respuestas

Con esta clase generaremos la respuesta correcta y las tres respuestas incorrectas (de relleno) según los parámetros que hayan salido al generar el enunciado. Esta clase sólo agrupa funcionalidad, y ningún objeto instancia esta clase. Según el enunciado que se haya generado en la clase anterior hay una función para generar la respuesta correcta y las tres respuestas incorrectas. Además esta clase posee una serie de funciones auxiliares para generar estas respuestas.

En las funciones que calculan el camino más corto hacemos uso de la función *BreadthFirst* que se ha definido en la clase grafo. Esta función tiene un parámetro de entrada, tipo, que sirve para determinar lo que queremos que nos devuelva esta función, ya sea el camino entero o simplemente la estación que nos sirve para hacer el trasbordo. Es muy importante que después de realizar esta función utilicemos clearMarks() definido en Graph, que sirve para borrar las marcas que establecen que un nodo ha sido visitado.

Clase LogManager

Con esta clase gestionamos un fichero “log”. En este fichero almacenamos los datos de las preguntas y las respuestas que generan la aplicación y la respuesta dada por el usuario. Además, se almacena la fecha exacta en la que se produce la escritura en el fichero. De esta forma, podremos conseguir detectar futuros errores que se puedan descubrir en la aplicación.

Esta clase nos proporciona los métodos para crear el fichero, para escribir datos en él y para cerrarlo tras su uso.

Clase BotonDifFacil , BotonDifNormal, BotonDifDificil

Estas clases representan los botones mediante los cuales el usuario puede elegir la dificultad con la que quiere empezar el juego. Estos botones aparecerán cuando el usuario pulse el botón comenzar y permitirán al usuario definir la dificultad inicial con la que comenzar el juego. Aunque se defina una dificultad inicial, esta puede ir cambiando a lo largo de la partida según las respuestas correctas seguidas o las respuestas incorrectas seguidas.

Una vez se haga click en uno de estos botones, desaparecerán y el usuario podrá comenzar la partida. Estos botones volverán a aparecer si el usuario al terminar una partida decide comenzar una nueva.

Clase BotonComprobar

Esta clase se instancia con el objeto BotonComp. Este botón nos permite comprobar una respuesta una vez la hemos marcado, generar una nueva pregunta una vez se ha comprobado una respuesta y cerrar la aplicación al final de la partida.

Básicamente consta de dos atributos booleanos (siguiente y cerrar) que determinan el estado del botón y el comportamiento cuando el usuario haga click en este botón dependerá del estado de las variables.

Si el usuario hace click y siguiente = true entonces la aplicación generará una nueva pregunta. Si siguiente = false, querrá decir que al pulsar el botón la aplicación comprobará si la pregunta marcada por el usuario es la correcta. Y por último, si cerrar = true quiere decir que ya se ha terminado la partida y el usuario al hacer click en este botón cerrará la aplicación.

Clase BotonComienzo

El botón que instancia esta clase nos permite iniciar el juego. Tras pulsarle tendremos que elegir una dificultad.

Cuando el usuario hace click sobre esté botón comienza el juego en sí. Tras pulsarle se deshabilitará el botón. Volverá a poderse pulsar tras terminar la partida, en cuyo caso el usuario al hacer click sobre él hará que un nuevo juego comience.

Es importante controlar cuando se puede utilizar o no el botón, ya que un mal uso puede dañar la secuencia de las acciones o reiniciar una partida sin querer cuando esta ya se ha comenzado. Esto causaría frustración sobre todo si el usuario llevaba una buena puntuación en la partida antes de cometer el “error”.

Clase BotonMetro

Al hacer click sobre el botón que instancia esta clase se muestra al usuario el plano del metro. Este botón estará habilitado durante toda la partida con el propósito de que el usuario pueda acceder al plano del metro en el momento que considere necesario.

Clase Planometro

Con esta clase cargamos la imagen del plano del metro y también el objeto CerrarPlanoMetro que nos permite cerrar el plano una vez se ha abierto. Este mapa puede agrandarse, reducirse y arrastrarse para que el usuario pueda manejarlo de forma que pueda visualizarlo mejor. El plano de metro va a estar formado por la imagen del plano y una X (clase CerrarPlanoMetro). Al hacer click sobre esta X se cerrará el plano.

Clase CerrarPlanoMetro

Clase que carga la imagen que al pulsarla nos servirá para cerrar el plano del metro. Es la X de la que se ha hablado en la clase PlanoMetro. Esta X estará en la esquina superior izquierda del plano y se moverá, aumentara de tamaño según la acción que se ejecute sobre el plano en sí. Esto se consigue gracias a la propagación de eventos de fling en la que un evento que sucede en una capa, sucede en todas sus capas inferiores.

Clase CargarGrafo

Esta clase tiene la funcionalidad que nos permite crear un grafo con las estaciones que se encuentran en el fichero *estaciones.xml* con sus respectivos arcos (*arcos.xml*). Este grafo contiene toda la información que necesitamos para generar las preguntas y las respuestas.

Para generar el grafo seguimos los siguientes pasos:

1) Se cargan los datos del fichero *estaciones.xml*.
2) Se recorre la variable que ha obtenido los datos de *estaciones.xml*, añadiendo las estaciones (nodos) al grafo.
3) Se cargan los datos de *arcos.xml*.
4) Se añaden los arcos al grafo, completándolo.

Es muy importante que el grafo generado contenga todos los datos. Deben aparecer las estaciones con todas las líneas a las que pertenecen, todos los arcos que unen las estaciones y los datos deben ser los correctos. Un error en estos datos puede provocar que se generen respuestas erróneas y por lo tanto despistar al usuario y confundirle.

Clase BotonRespuesta

Clase que implementa la funcionalidad de los botones que contienen las respuestas. Al hacer click en una respuesta se marcará con un color naranja para que el usuario sepa cual es la respuesta que ha marcado hasta el momento. Cuando se presione el botón para comprobar la respuesta se comprobará la respuesta que previamente se ha marcado. Si no hay ninguna respuesta marcada se dirá al usuario que debe marcar una respuesta para poder comprobarla. Una vez hemos marcado una respuesta podemos desmarcarla haciendo click otra vez sobre ella y también podemos hacer click en otra respuesta marcándola y desmarcando la aplicación automáticamente la primera respuesta.

Los botones de las respuestas tendrán un texto en el cual se muestra la respuesta asignada a ese botón. Las funciones para marcar y desmarcar un botón se definen dentro de esta clase.

Clase Close

Al pulsar sobre el botón que instancia esta clase, el usuario podrá cerrar la aplicación directamente. Este botón se representa mediante un círculo y dentro una X. Se encuentra en la esquina derecha de la aplicación. Este botón está siempre operativo ya que el usuario puede decidir cerrar la aplicación por cualquier motivo sin necesidad de terminar una partida.

3.6. Ejemplo de ejecución

En este apartado detallaremos un ejemplo de ejecución de la aplicación para ver la interfaz, sus elementos y el funcionamiento de los mismos. A su vez, veremos como la aplicación reacciona ante los diferentes eventos y la realimentación que el usuario recibe según va interactuando con la aplicación y dando respuestas a las preguntas mostradas. En cada paso se detallará qué es lo que ha hecho el usuario y qué ha sucedido.

La pantalla que se le muestra al usuario nada más iniciar la aplicación se muestra en la figura 6. Como se puede ver en el cuadrado de texto de texto de la pantalla debe pulsar el botón de COMENZAR para iniciar el juego.

Una vez que el usuario pulsa el botón comenzar aparecen los botones de la figura 7 que permiten al usuario elegir la dificultad con la que quiere empezar el juego. Cuando haya varias personas con discapacidad cognitiva utilizando esta aplicación sobre una mesa multitouch, la elección del nivel de dificultad inicial la podría realizar el profesor o responsable que estuviera con ellos y decidir cuál es la más apropiada.

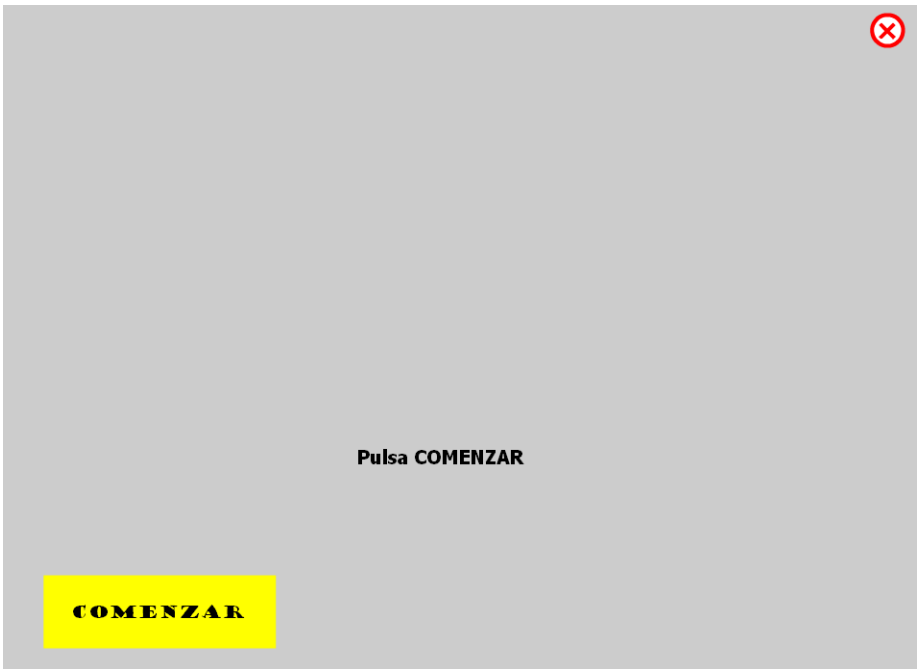

Figura 6. Inicio de la Aplicación

A medida que el usuario vaya cogiendo práctica con la aplicación y con la red de metro podrá elegir dificultades mayores. La dificultad fácil hace preguntas sobre estaciones y líneas que van desde la línea 1 hasta la 4, normal desde la 1 hasta la 8 y difícil desde la 1 a la 12. De esta manera, los usuarios poco a poco van conociendo las distintas líneas de metro.

Figura 7. Elegir dificultad

La figura 8 presenta un ejemplo de una posible pregunta del juego. Tal y como se puede observar en esta figura existen 7 áreas diferenciadas. Dentro de la parte superior de la aplicación se sitúa la puntuación actual obtenida en una partida (área 1), el número de preguntas que la aplicación ha realizado hasta ahora (área 2) y el botón para cerrar la aplicación (área 3). Debajo se muestra el enunciado de la pregunta actual (área 4). A continuación, se presentan las posibles respuestas que puede seleccionar el usuario dentro del área 5. El área 6, engloba un cuadro de texto que proporcionará al usuario la realimentación correspondiente. Por último, en el área inferior de la pantalla (área 6), se muestran los botones para comenzar una nueva partida, consultar el plano del metro para contestar a las preguntas y comprobar si la respuesta dada por el usuario es correcta.

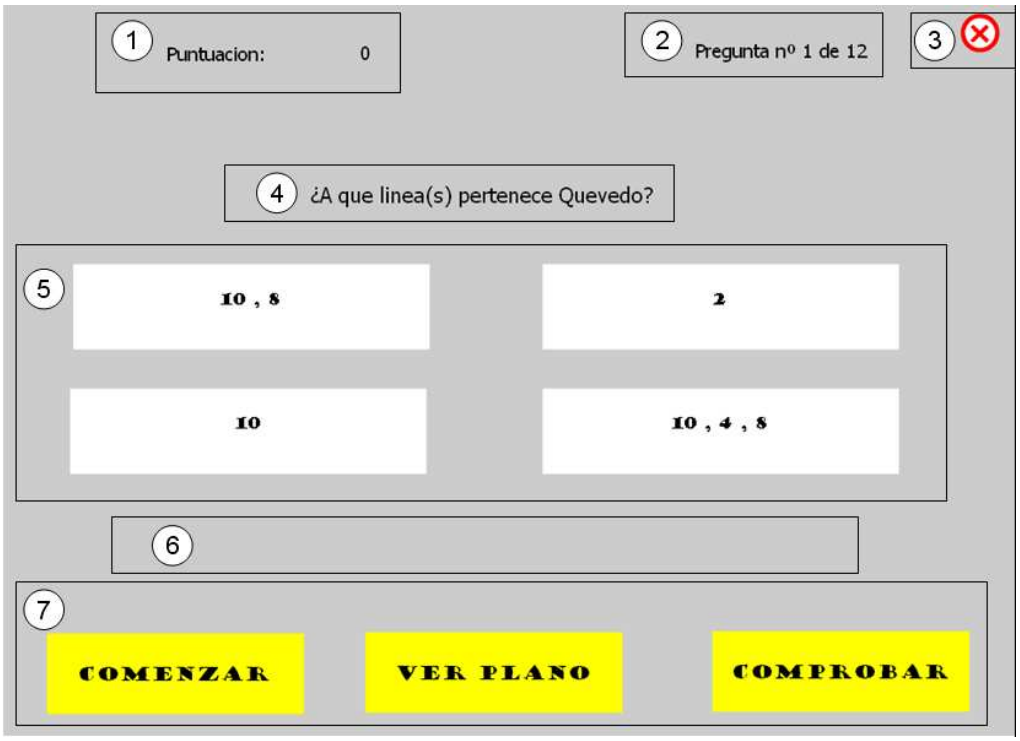

Figura 8. Ejemplo de generación de una posible pregunta del juego

Si el usuario pulsa sobre el botón comprobar y no ha seleccionado ninguna respuesta, se le mostrará un mensaje advirtiendo de ello (ver figura 9).

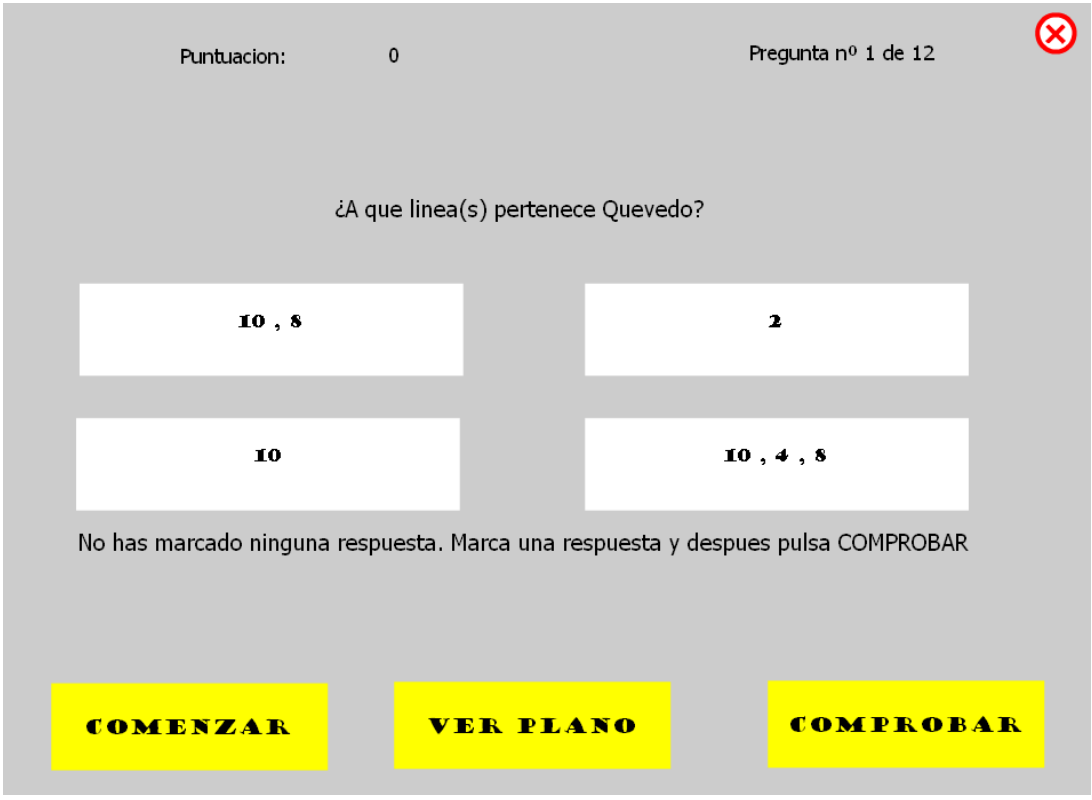

Figura 9. Información mostrada al usuario cuando no se ha marcado nada

Siempre que quiera, el usuario puede consultar el plano del metro para poder responder a la pregunta del juego. Para ello deberá pulsar el botón "Ver plano". Se abrirá el plano de la red de metro mostrado en la figura 10 y podrá buscar la información que necesita para responder a la pregunta. Vemos que aparece el plano con una en la esquina superior izquierda que sirve para cerrarlo. Una vez localizada la estación el usuario pulsará sobre la, el plano se cerrará y podrá continuar la partida.

Figura 10. Plano del metro de Madrid

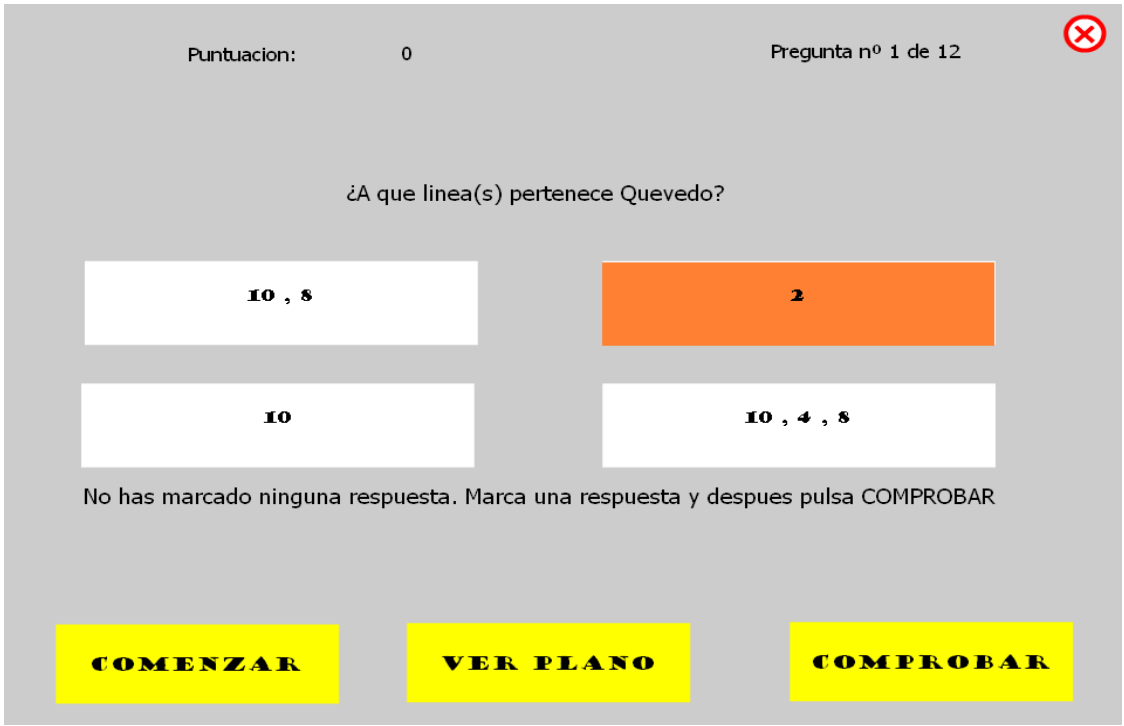

Figura 11. Marcar respuesta

Cuando el usuario conoce la respuesta a la pregunta propuesta por la aplicación desarrollada, la selecciona entre las posibles opciones. A seleccionar la respuesta, se marca en color naranja (ver figura 11).

A continuación, el usuario deberá pulsar el botón "comprobar". Al pulsar este botón se verificará si la respuesta es correcta. En caso afirmativo, se marcará en un color verde y se mostrará un mensaje sobre el acierto debajo de las posibles respuestas (ver figura 12).A su vez el texto del botón "comprobar" cambiará a "siguiente", para continuar el juego y generar una nueva pregunta aleatoria (ver figura 13). Además, la puntuación subirá 5 puntos.

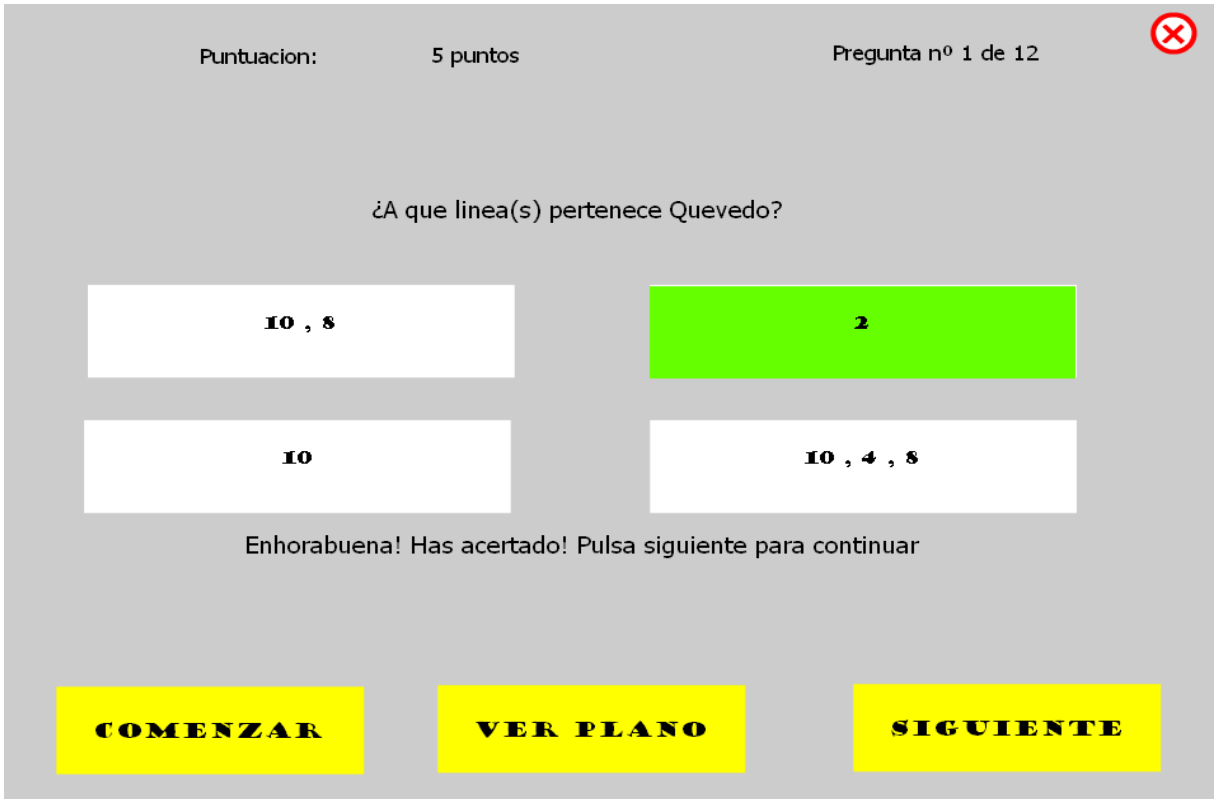

Figura 12. Respuesta correcta

Imaginemos que el usuario ha decidido marcar la respuesta "Ciudad de los Ángeles" por que piensa que es la correcta (ver figura 14). Al igual que en el caso anterior, la opción seleccionada se marca en color naranja y el usuario pulsa el botón "comprobar" para verificar la respuesta.

Figura 13. Nueva pregunta aleatoria generada por el juego

Figura 14. Usuario marca otra respuesta

Esta vez el usuario se ha equivocado y vemos como la aplicación marca automáticamente la respuesta correcta con un color verde y marca la respuesta dada por el usuario con un color rojo (ver figura 15). En el *feedback* dado al usuario aparte de mostrar la respuesta correcta se ofrece un mensaje de ánimo para que lo intentemos con la siguiente pregunta.

También podemos ver que la puntuación no se actualiza ya que el usuario no ha acertado y el texto del botón "comprobar" se cambia por "siguiente" para que podamos contestar a la siguiente pregunta.

Figura 15. Caso donde el usuario selecciona una respuesta incorrecta

El usuario sigue respondiendo preguntas hasta que, al responder la pregunta número 12, la aplicación da un aviso de que se ha llegado al final de la partida (ver figura 16) mostrando la puntuación final obtenida en la partida. En este instante podremos hacer varias acciones:

1.- Pulsar el botón "nueva partida" para empezar una nueva partida. En este caso nos volvería a aparecer una pantalla similar a la de la figura 4, que es donde elegimos de nuevo la dificultad de la partida. Una vez elegida el usuario jugaría una nueva partida.

2.- Cerrar la aplicación pulsando sobre el botón "cerrar" o sobre el icono

El usuario elegiría una de las dos opciones. En la primera, empezaría una nueva partida y en la segunda se cerraría la aplicación y acabaría el juego.

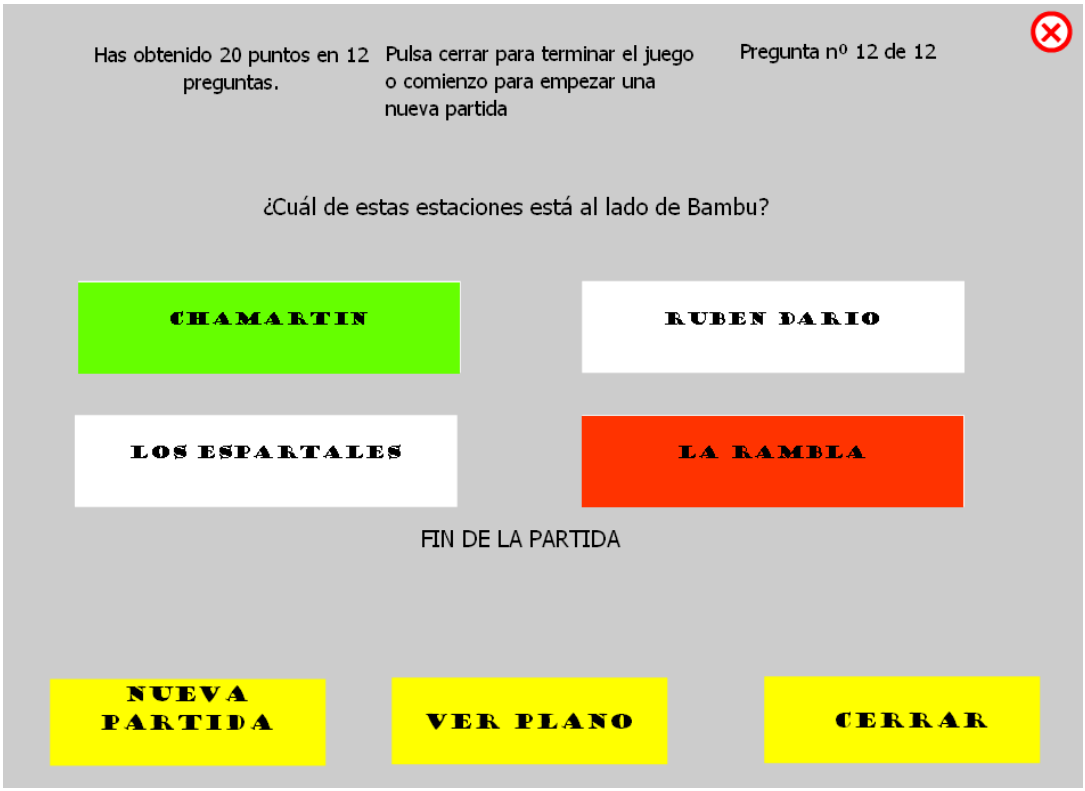

Figura 16. Fin de la partida

4.- Conclusiones

4.1. Logros Alcanzados

La red del metro es complicada y si a las personas que tenemos una plena capacidad mental a veces nos cuesta decidir un camino o buscar una estación, las personas con discapacidad cognitiva tienen muchas más dificultades. Por otro lado, es conocido que estas personas requieren muchos más ejercicios prácticos y reiterativos a lo largo de su proceso de aprendizaje. Por este motivo, la aplicación desarrollada hace especial hincapié en la familiarización con los nombres de las estaciones, las líneas, los recorridos, etc., de la red de Metro de Madrid para ofrecer una mayor autonomía en los desplazamientos a este tipo de usuarios.

Estamos viviendo como las nuevas tecnologías *multitouch* van apareciendo en nuestras vidas (móviles, Ipad, Ebooks) y por tanto se hace necesario el desarrollo de aplicaciones que puedan ser ejecutadas en estas superficies. En el plano educativo, las mesas *multitouch* ofrecen a los profesores un lugar donde sus alumnos pueden realizar actividades educativas al igual que en una mesa normal, ofreciéndoles la posibilidad de reutilizar materiales, almacenar datos de la interacción de los estudiantes y adaptar las tareas a las características y habilidades de los estudiantes. En este ámbito se enmarca la aplicación desarrollada que podrá servir para mejorar el conocimiento de los usuarios sobre la red de metro y entrenar habilidades como la toma de decisiones, el trabajo en grupo o la comunicación.

Los objetivos fijados al principio de este proyecto se han cumplido con la aplicación desarrollada. Además, esta aplicación ha sido probada tanto en ordenadores personales como en superficies multitouch por usuarios sin discapacidad cognitiva. Se espera que durante el próximo curso académico se pueda probar la aplicación desarrollada con usuarios de síndrome de Down dentro del marco del proyecto HADA[1].

4.2. Dificultades

Adobe Flash CS4 es un producto de pago, el cual necesita certificación para ser usado. Aunque disponemos de otra gran cantidad de productos para programar orientado a Adobe Air, la herramienta que hemos utilizado es de las pocas que ofrece la capacidad para crear interfaces sencillas.

[1] http://hada.ii.uam.es

Además, en el plano personal, Adobe Flash CS4 es un entorno que no había utilizado antes, por lo tanto primero tenía que conocer las funcionalidades que proporcionaba, y habituarme a él. Hubo que realizar un periodo en el que desarrollaba programas simples con la intención de ir aprendiendo como funcionaba tanto el entorno como ActionScript 3. Una vez adquirida la destreza necesaria, ya pude comenzar a desarrollar la aplicación.

Otro problema para desarrollar este tipo de aplicaciones es que el mercado de las mesas y pantallas *multitouch* ha aparecido recientemente y por lo tanto tendrían unos precios muy elevados. Nosotros hemos utilizado para probar la aplicación una mesa proporcionada por la Universidad Autónoma de Madrid. Aunque ahora, que estamos en tiempos de crisis, es imposible proporcionar una pantalla táctil a los alumnos, en un futuro y a medida que esta tecnología se vaya abaratando, se irán incorporando a nuestras vidas rutinarias y se irán creando nuevas aplicaciones.

Como se ha comentado en el apartado anterior, todavía no hemos podido probar la aplicación con usuarios reales, es decir, con personas que tuviesen discapacidades cognitivas. En el futuro se prevén hacer pruebas con estos usuarios para determinar si la aplicación se adapta a sus características y si la interfaz es fácil e intuitiva.

Por último, aunque Fling es una librería que nos ofrece una gran funcionalidad, es difícil entenderla sin un manual de usuario. Aún así, se disponen de una serie de ejemplos que nos permiten entender como funciona Fling y nos ayuda a aprender como usar Adobe Flash CS4 y ActionScript.

4.3. Trabajos futuros

Nuestro proyecto podría actualizarse con más tipos de preguntas que se le vayan ocurriendo al diseñador o profesor y que sean de utilidad para el objetivo principal de esta aplicación, familiarizar al usuario con la red de metro de Madrid. Además se podría recorrer toda la red de metro tomando el tiempo aproximado que se tarda en ir de una estación a otra para añadir estos datos al grafo. De esta forma el recorrido más corto, se haría teniendo en cuenta el tiempo real que se tarda en lugar del número de estaciones recorridas. Es una tarea tediosa pero que otorgaría gran realismo al juego desarrollado.

Otra posible mejora sería detectar el nivel de conocimiento del usuario más apropiado al inicio del juego sin necesidad de que él la seleccionase. Esta detección se podría realizar presentando al usuario distintas preguntas de diferente nivel de dificultad. En función de sus respuestas, se calcularía su nivel inicial.

5.- Bibliografía

Libros

- H . Stephen Kaye, Ph.D: Computer and Internet Use Among People with Disabilities. March 2000.

Artículos

- Dietz, P. and Leigh, D. DiamondTouch: a multi-user touch technology. In Proc. UIST ’01, pages 219–226, 2001.
- Han, J. Low-cost multi-touch sensing through frustrated total internal reflection. In Proc. UIST ’05, pages 115–118, 2005.
- Kaltenbrunner, M., Bovermann, T., Bencina, R., Costanza, E.: "TUIO - A Protocol for Table Based Tangible User Interfaces". Proceedings of the 6th International Workshop on Gesture in Human-Computer Interaction and Simulation (GW 2005), 2005.
- Shen, C., Vernier, F., Forlines, C., and Ringel, M. DiamondSpin: an extensible toolkit for around-the-table interaction. In Proc. CHI ’04, pages 167–174, 2004. Juana Mª Ortega Tudela: Bondades y limitaciones del material multimedia para personas con síndrome de Down.
- Constantine Stephanidis: Adaptative techniques for universal access.
- Anne Marie Piper, Eileen O’Brien, Meredith Ringel Morris, Terry Winograd: SIDES: A cooperative tabletop computer game for social skills development.
- Carmien, S., Dawe, M., Fischer, G., Gorman, A., Kintsch, A., and Sullivan, J.F., JR: Socio-Technical Environments Supporting People with Cognitive Disabilities Using Public Transportation.
- Llinás, P., Montoro, G., Haya, P., Herranz, M. G., Alamán, X.: Interacción gestual sobre superficies horizontales multi-contacto para un control natural.

Webs

- Sparsh-UI: http://code.google.com/p/sparsh-ui/.
- TouchKit: http://labs.nortd.com/touchkit/.

- Touchpy: http://code.google.com/p/touchpy/.
- Página Web del laboratorio Amilab de la Universidad Autónoma de Madrid : http://amilab.ii.uam.es/
- TUIO - A Protocol for Tangible User Interfaces: http://tuio.lfsaw.de/
- Actionscript: http://www.adobe.com/livedocs/flash/9.0/ActionScriptLangRefV3/
- Página de Adobe: http://www.adobe.com/

Anexos

ANEXO I Instalación Adobe Flash CS4

Para instalar el programa Adobe Flash CS4 ejecutamos el archivo .exe iniciándose la instalación y apareciendo la siguiente ventana:

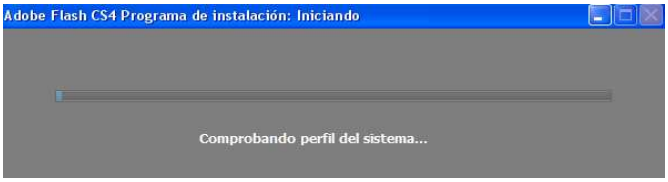

Figura 17. Inicio de la instalación de Adobe Flash CS4

Tenemos que esperar a que termine de cargar. Entonces, si tenemos abierta algún navegador Web nos aparece el siguiente aviso:

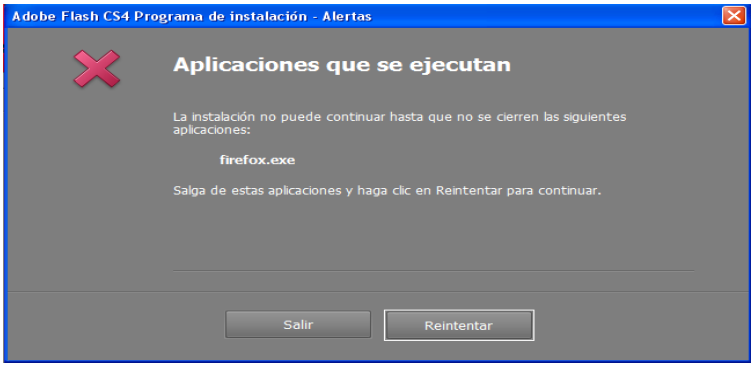

Figura 18. Aviso sobre navegadores Web abiertos

Cerramos los programas que nos solicita el programa de instalación para continuar con la misma. Tras cerrarlos, presionamos el botón reintentar y aparece el siguiente paso:

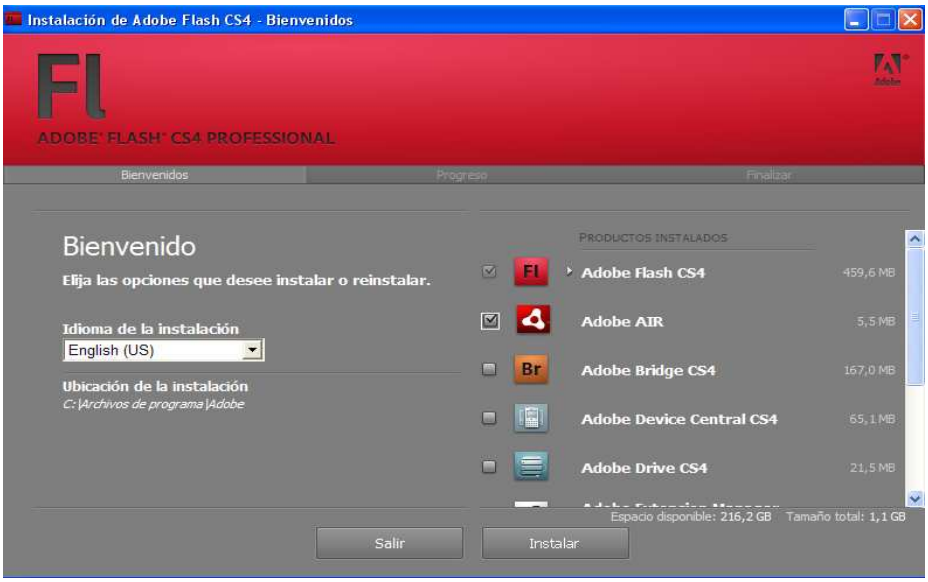

Figura 19. Seleccionar los componentes a instalar de Adobe Flash CS4

Tras elegir el idioma de la instalación y los productos que queremos instalar, en nuestro caso son Adobe Flash CS4 y Adobe Air pulsamos en el botón instalar y dará comienzo la instalación de los productos. Ahora sólo tenemos que esperar a que se complete el proceso para poder disfrutar de Adobe Flash CS4 y de Adobe Air (ver figura 20).

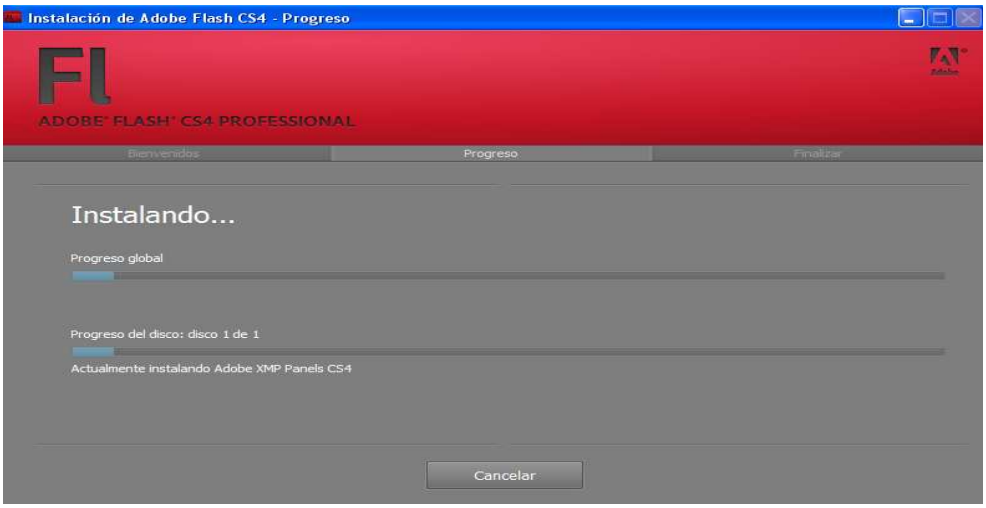

Figura 20. Progreso de la instalación

Si en algún momento nos arrepentimos podemos pulsar el botón de cancelar para interrumpir la instalación. Si la instalación se ha ejecutado correctamente tiene que aparecer la siguiente pantalla:

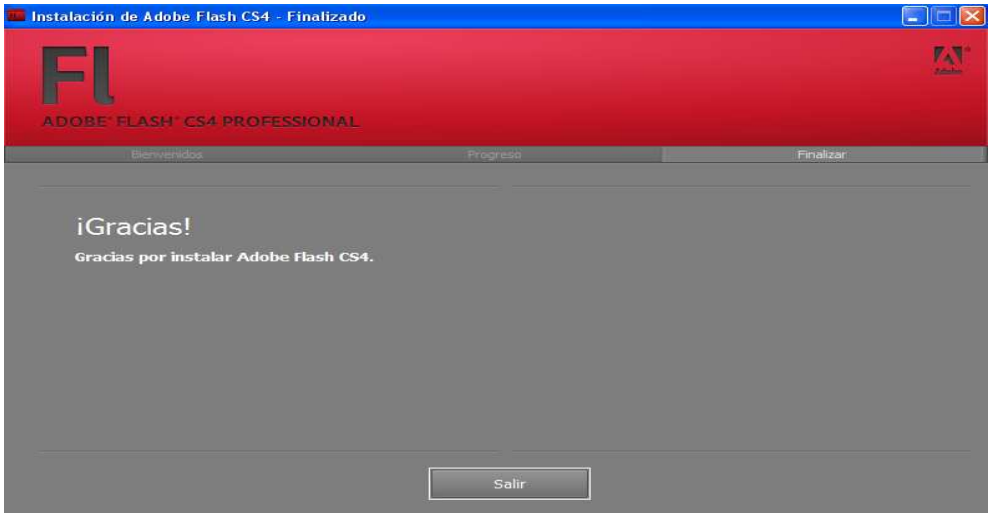

Figura 21. Finalización de la instalación

Por último, pulsamos el botón de salir.

ANEXO II Creación de un fichero .air

En este apartado vamos a ver el proceso de generar un archivo .Air para poder exportar nuestra aplicación a otros ordenadores, sin necesidad de llevar todo el código ni de tener Adobe Flash CS4 instalado.

Al abrir Adobe Flash CS4 la primera pantalla que nos sale es la análoga a la de la figura 22.

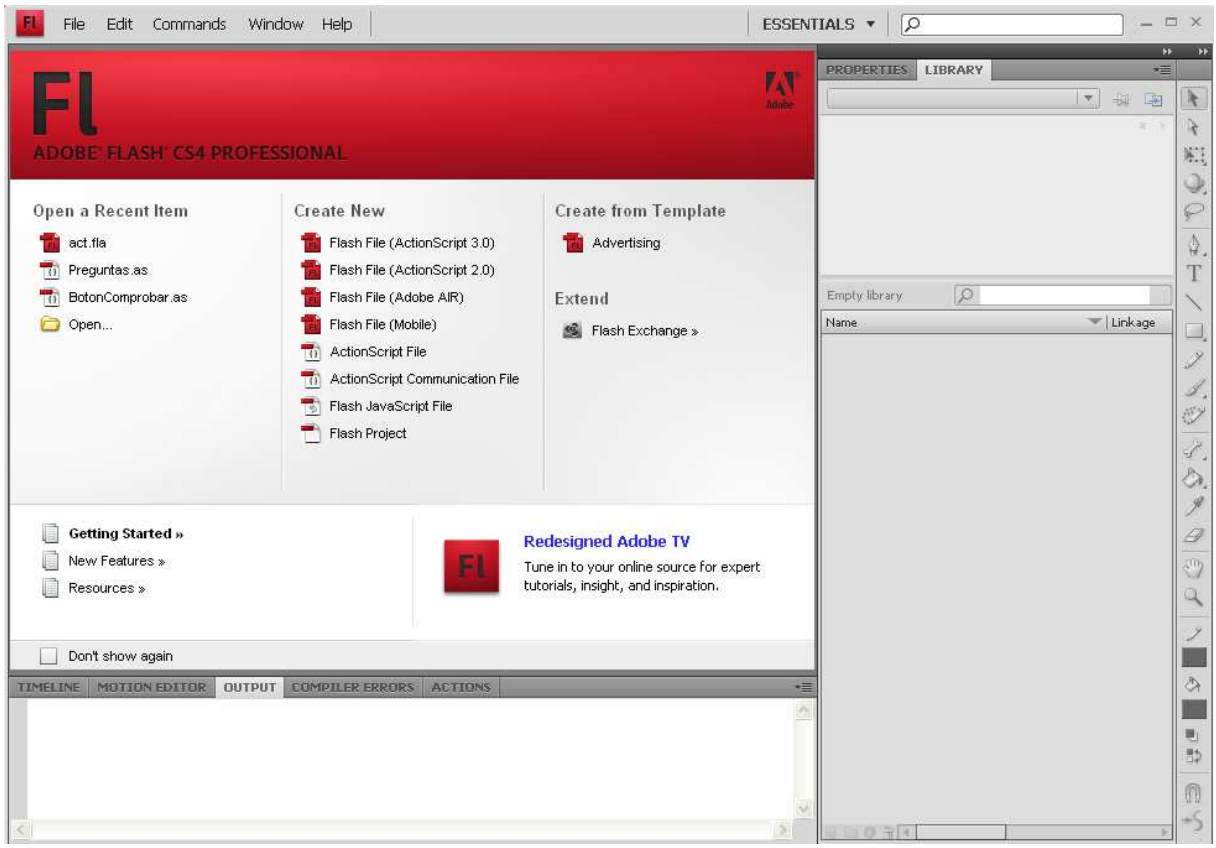

Figura 22. Inicio de Adobe Flash CS4

La base de las aplicaciones creadas mediante Adobe Flash son los ficheros .fla. Este archivo contiene el material fuente para nuestra aplicación Flash. Mediante este archivo podemos diseñar la interfaz inicial de la aplicación para posteriormente convertirlo en un archivo .swf, que podremos ejecutar en el ordenador.

Una vez que hemos creado nuestro fichero .fla para nuestra aplicación y tenemos toda la funcionalidad de la aplicación pulsamos en File/Publish Settings... y nos aparecerán una serie de opciones que se muestran en la figura 23. En esta pantalla pulsamos la pestaña Flash, que es donde cambiaremos los parámetros. Es muy importante cambiar estos parámetros

como se describe a continuación, ya que el "*Player*" (reproductor) por defecto que se encuentra en Adobe Flash CS4 es *Adobe Flash Player* y nosotros necesitamos Adobe Air, que es la que nos proporciona la funcionalidad necesaria para hacer táctil nuestra aplicación. Cabe mencionar que, además, gracias a Adobe Air seremos capaces de juntar en un mismo fichero todos los ficheros necesarios para el buen funcionamiento de nuestra aplicación (datos de las estaciones e imágenes), para después instalarlo todo a la vez en otro ordenador.

En "*Player*": seleccionamos la versión de Adobe Air que tengamos, en nuestro caso 1.1 y en la opción de "*Script*": seleccionamos ActionScript 3 (suele venir por defecto).

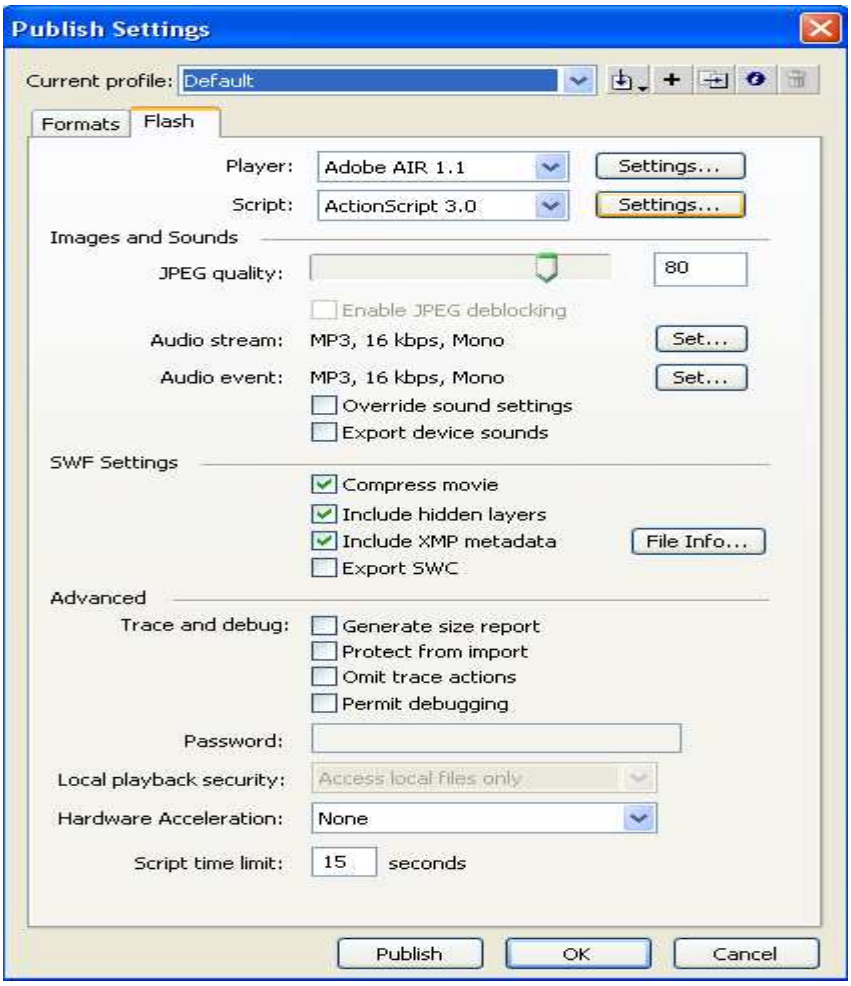

Figura 23. Opciones de publicación

Una vez hemos puesto todas las opciones correctamente cerramos esta pantalla y nos dirigimos al menú *File* y opción *Air Settings*... que es donde vamos a añadir todos los ficheros que son necesarios para nuestra aplicación (ver figura 24). Es importante acordarse de este paso ya que nuestro programa utiliza ficheros externos para generar el grafo e

imágenes para consultar el plano del metro. Si estos ficheros no se añadiesen, la aplicación no funcionaría una vez instalada.

Para añadir estos ficheros nos dirigimos al apartado "*included files*" y mediante el botón "+", buscamos el fichero que queremos incluir y lo añadimos. Cuando se instale la aplicación, en la carpeta aparecerán estos ficheros que ahora hemos añadido, y por lo tanto la aplicación podrá disponer de ellos.

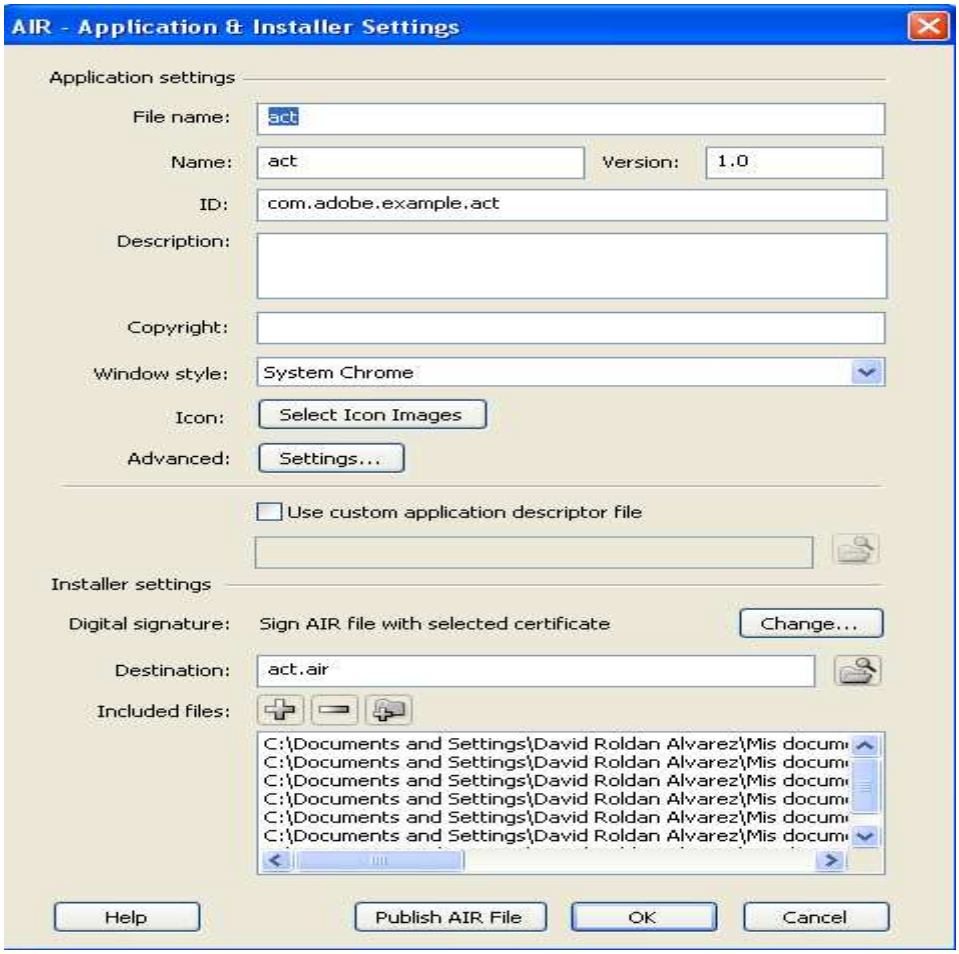

Figura 24. Configuración del fichero .air

Una vez que hemos configurado todo correctamente pulsamos en el botón Publish Air File y nos aparecerá la pantalla de la figura 25. En esta ventana tenemos que elegir un certificado (también podemos crear uno nuevo) para darle los datos del creador de la aplicación. Este paso es obligatorio. Nosotros hemos elegido un certificado que ya teníamos creado.

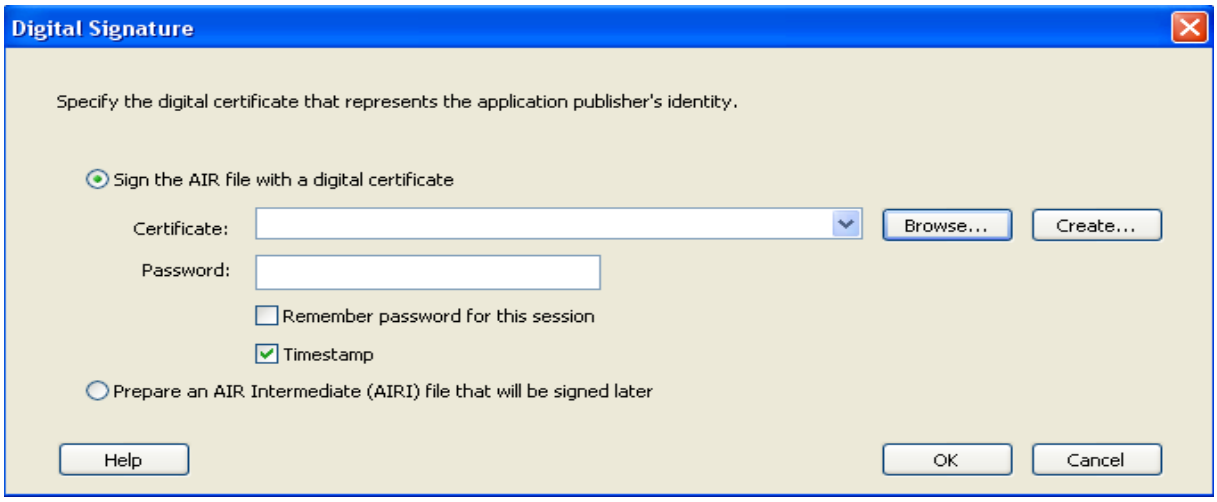

Figura 25. Certificado para la creación del fichero

Una vez añadidos los datos pulsamos OK y Adobe Flash CS4 nos dirá que el fichero .Air se ha creado correctamente.

ANEXO III Instalación de un fichero .air

Es importante que el ordenador donde se vaya a instalar la aplicación tenga Adobe Air. Este programa se puede descargar de la Web oficial de Adobe sin coste alguno. Una vez tengamos Adobe Air instalado hacemos pulsamos sobre el fichero .air y nos aparecerá una pantalla para la que confirmamos la instalación del fichero .air (figura 23).

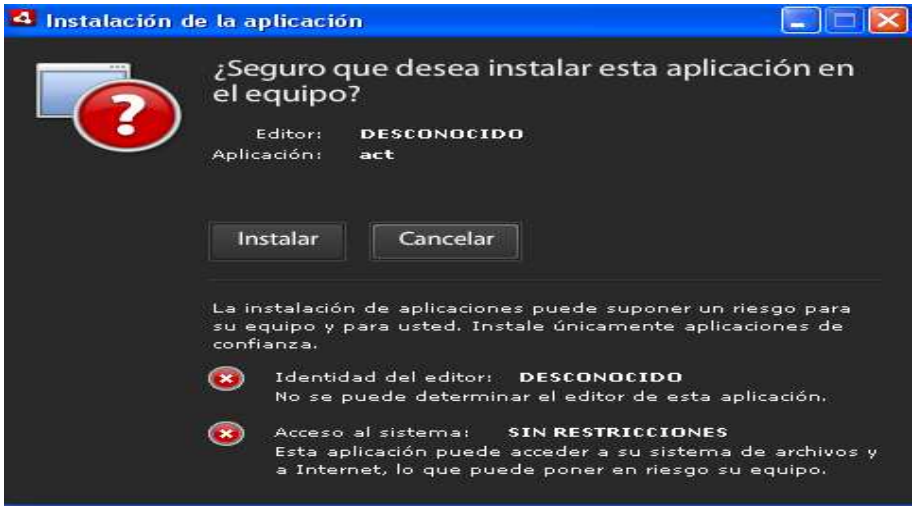

Figura 26. Instalación de un fichero .air

Presionamos en el botón Instalar y se iniciará la instalación de la aplicación. Nos aparecerá una ventana (ver figura 24) que nos da una serie de opciones y nos permitirá elegir el directorio donde queremos instalar la aplicación. Una vez elegido pulsamos el botón continuar y la aplicación se instalará por completo.

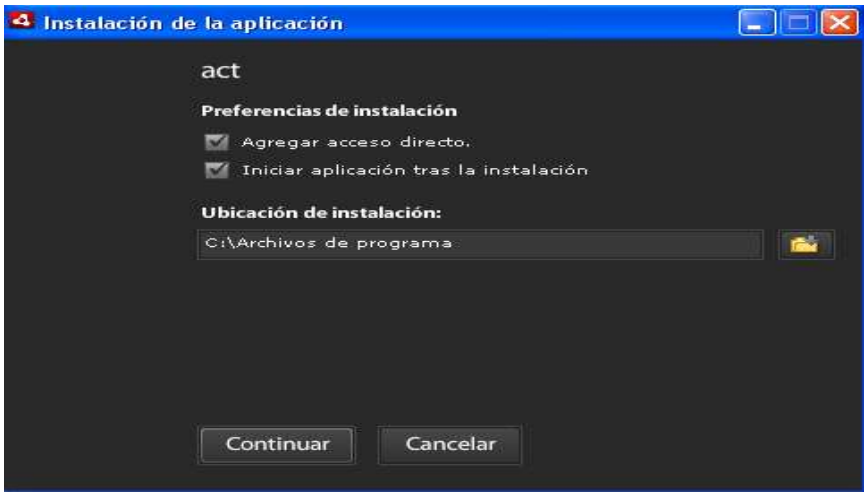

Figura 27. Opciones de instalación de un fichero .air