e-Mazing - Juego de autoaprendizaje en terminales móviles

Mario Viktorov Mechoulam Nikolaeva, Juan Hernández Serrano, Josep Pegueroles Vallés.
Grupo de servicios telemáticos, Departamento de Ingeniería Telemática
Escuela Técnica Superior de Ingeniería de Telecomunicación de Barcelona
Universitat Politècnica de Catalunya

ABSTRACT

¿Te gustan los juegos? En este artículo se describirá el resultado de un Proyecto de Final de Carrera de la EPSC: un videojuego educativo para teléfonos móviles. Los juegos y más concretamente los videojuegos, pueden ser herramientas muy potentes de aprendizaje, debido a la alta capacidad de atención y concentración que muestra el usuario durante su curso. Siendo uno de los propósitos llegar a un público amplio y heterogéneo, se ha escogido el teléfono móvil como plataforma de la aplicación, por su gran penetración en nuestra vida cotidiana, ya que presenta la posibilidad de usarlo en cualquier momento y lugar. Estos son los principios de e-Mazing.

Primero, se desarrollará el concepto SRS y la investigación del científico Sebastian Leitner a partir de la cual se creó, destacando en calidad sus aportaciones positivas para con este proyecto. En el mismo punto, se expondrán las razones y retos que ha planteado la elección de la plataforma de la aplicación. Por último, se explicará el funcionamiento global del software: la generación de contenidos y mapas procedimentales, la interfaz del SRS y el desarrollo prolongado del juego.

Siendo Java la tecnología más extendida en los dispositivos móviles, se decide utilizar el lenguaje de programación orientado a este tipo de plataformas: J2ME. Por otro lado, se ha querido proveer al usuario de un agradable entorno gráfico de manejo intuitivo, por lo que se ha recurrido a una de las últimas herramientas de Sun, la librería gráfica multiplataforma LWUIT. Basando nuestro trabajo en lo que se conoce hoy en día como un sistema de repetición espaciada (*SRS - Spaced Repetition System/Software*), buscamos presentar al usuario un conjunto de herramientas que le permitan crear y gestionar su propio contenido educativo; en contrapartida, el juego proporcionará el método para el posterior estudio y evaluación del contenido siguiendo las pautas que se explican en el cuerpo de este documento.

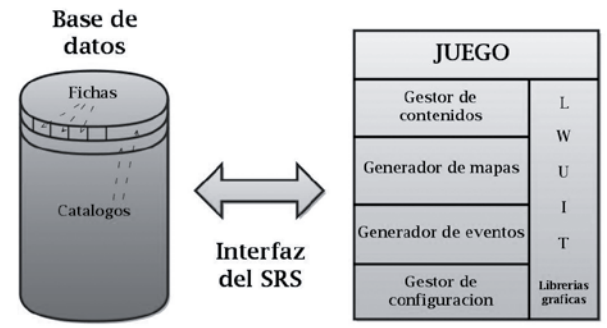

Figura 1 - Estructura de la aplicación

INTRODUCCIÓN

En 1970, Sebastian Leitner presenta su libro sobre la psicología del aprendizaje y la memoria, *How to learn to learn* **[1]**. En él, se explica un sistema para optimizar el aprendizaje y el tiempo que se dedica a él, minimizando el número de repeticiones que se necesitan para aprender un concepto, un **desafío** o como lo llama él mismo: “un elemento”. La teoría sostiene que para llegar a memorizar un desafío no basta con innumerables repeticiones durante unas semanas. Por poner un ejemplo: es mucho más productivo estudiarlo poco a poco, dejando transcurrir entre los repasos unos intervalos temporales que están relacionados con la estructura de la memoria. Al estudiarlo por primera vez, la respuesta de ese desafío permanece en la **memoria inmediata** de la persona unos minutos o tal vez unas horas; contrariamente a la opinión general, cualquier repaso durante ese intervalo no ayudará a reforzar la memorización del concepto. Es necesario que algunos días después de este

contacto inicial, la persona repase de nuevo el concepto y trate de invocar la respuesta del desafío de su memoria. En caso de lograrlo se habrá conseguido almacenar el concepto en la **memoria a corto plazo**. A partir de este punto, se van incrementando los intervalos de tiempo entre repasos y el sujeto vuelve a esforzar su mente continuamente hasta lograr retener la respuesta del desafío en su **memoria a largo plazo**. La parte más importante dentro de este trabajo es forzar la mente al límite a la hora de recordar los conceptos después de un periodo de contención, se tarde lo que se tarde, ya que es allí donde realmente sucede el proceso de aprendizaje. En caso de un fallo, se retrocede una categoría en función de intervalos temporales, con el fin de retomar el proceso desde un punto cómodo. De esta manera, el usuario no perderá tiempo repitiendo una y otra vez conceptos que ya conoce y aprovechará al máximo su tiempo de estudio repasando lo que de verdad necesita aprender. La correcta gestión de los intervalos de aparición de los desafíos permite optimizar el **tiempo dedicado al estudio** y crear el sistema que se conoce hoy por hoy como SRS.

Sin embargo, todo método de estudio puede acabar siendo aburrido o pesado. El mundo de la psicología a menudo ha demostrado que la mente humana tiende a recordar un mayor número de conceptos en un momento de felicidad o diversión y a olvidar los momentos “malos”. Observaciones personales han ayudado a detectar que un buen ejemplo de lo anterior son los videojuegos; las personas que hayan jugado seguramente recuerdan una gran cantidad de información como la historia, los nombres de personajes, localizaciones, objetos y por qué no, hasta combinaciones de botones. Todos estos conceptos son retenidos sin el más mínimo esfuerzo mientras nos divertimos invirtiendo nuestro tiempo en ocio. Nuestro objetivo es por tanto implementar un SRS en un videojuego, pero dónde? La elección de la plataforma viene condicionada por la posibilidad de su utilización en cualquier momento o lugar, y por un grupo de usuarios lo más heterogéneo posible: todo apunta hacia los teléfonos móviles.

La elección de los teléfonos móviles como plataforma implica que el lenguaje de programación utilizado será Java, más concretamente J2ME. Java es un lenguaje de alto nivel que fue creado a mediados de los 90 por Sun Microsystems. A pesar de que no fue diseñado para ello, sus usos e implementaciones crecieron rápidamente en diversas áreas, para lo cual fueron creadas distintas ediciones que se amoldaran a las necesidades concretas de las plataformas que las ejecutaban **[2]**. Una de estas distribuciones es la edición micro de Java (*Java 2 Micro Edition* – J2ME, ahora conocida simplemente como JME), ideada para dispositivos con capacidades computacionales y gráficas mermadas; entre los cuales se encuentran los dispositivos móviles. JME tiene funcionalidades reducidas respecto a sus hermanos mayores, posee una interfaz de programación de aplicaciones (*application program interface* – API) adaptada y una máquina virtual de kilobyte (*kilobyte virtual machine* – KVM) que se encarga de interpretar el código **[3]**. El perfil de información móvil del dispositivo (*Mobile Information Device Profile* – MIDP) son las librerías y conjunto de herramientas que proporcionan soporte a nuestro código. En 2009 existen dos versiones, la segunda de las cuales incorpora numerosas mejoras utilizadas en este proyecto como la pantalla a color, capacidad de reproducción de sonidos, multitud de mejoras para distintos tipos de comunicación o la API de multimedia para la reproducción de video (MMAPI). Toda aquella aplicación que se ha realizado con este perfil recibe el nombre de MIDlet y distintas aplicaciones pueden coexistir entre ellas; adicionalmente su estado siempre será uno de los tres: activo (ejecutándose), pausado o destruido. Por último, los MIDlets se empaquetan en una *suite* que es formada por dos archivos con extensión .JAD y .JAR; el jar contiene el código compilado a ejecutar, mientras el archivo descriptor Java (*Java Archive Descriptor* - JAD) contiene información sobre la aplicación, rutas, tamaño y decenas de etiquetas opcionales más **[4]**.

No obstante, como cualquier otra plataforma, presenta unos inconvenientes a superar. Por un lado, el espacio de memoria debe ser lo suficientemente

grande como para poder almacenar toda la información educativa e instalar la aplicación sin problemas; por otro, el poder computacional de esta es severamente exigente. Finalmente, la escasa memoria RAM y su deficiente gestión por parte de la mayoría de fabricantes de terminales móviles ha supuesto el mayor de los retos. La solución pasaba por una reducción general de la calidad de imagen y la constante optimización y reutilización de componentes. Un aspecto lateral, pero no menos importante ha sido las estrictas medidas de seguridad con respecto al acceso de los datos de usuario por parte de aplicaciones, implementadas a partir de la versión MIDP 2.0, que han obligado a firmar las aplicaciones con su consecuente desembolso económico.

Sopesando toda esta información, se decidió crear un MIDlet en lenguaje J2ME, implementando una variación del sistema de Leitner dentro de un videojuego con el fin de potenciar el factor de aprendizaje de todo el software. A grades rasgos, para su utilización el usuario deberá introducir algún contenido a estudiar y posteriormente jugar a un videojuego laberíntico, mientras una interfaz se encarga de presentarle periódicamente las preguntas preparadas. A continuación se explica todo esto con más detalle.

MÉTODOS

El trabajo está dividido en dos grandes sectores: el gestor de contenidos y el juego de aprendizaje. Se puede ver un diagrama de sus secciones en la Figura 1.

El gestor de contenidos está formado por un conjunto de pantallas y herramientas que permiten al usuario crear y almacenar todo el contenido que desee. Este es el primer punto en el que se introducen los conceptos **ficha** (cada uno de las preguntas, desafíos o elementos a estudiar) y **catálogo** (un contenedor para almacenar fichas de similar temática). Las fichas siguen la estructura "desafío – información/descripción/pistas – respuesta/s válida/s" (Figuras 2 y 3), el usuario es el encargado de redactar una pregunta y las posibles respuestas aceptadas; la información o las pistas son opcionales. Es posible almacenar tantas fichas como se desee (como permita la memoria del terminal móvil) dentro de un catálogo, así como editarlas, borrarlas, añadir otras nuevas y guardar todos estos datos en el disco duro. Para no abrumar al usuario desde el principio con la necesidad de introducir con el teclado todo el contenido a estudiar, la aplicación dispone de una opción para importar y transformar catálogos completos de otros SRS para ordenador. Aun así, uno de los puntos fuertes del MIDlet radica en la posibilidad de crear nuestros propios catálogos y fichas, escogiendo la temática según las necesidades de cada usuario. Por último, se ha dado un paso más en la estructura de las fichas: hasta el momento estas siempre se componían de texto, ya fuera una palabra a adivinar en otro idioma o unas frases detallando un problema matemático; en esta aplicación se permite añadir una imagen, un archivo de audio o un video como complemento al texto. De esta manera se amplían exponencialmente las posibilidades educativas: se podrá estudiar arte mediante imágenes, aprender música clásica escuchándola o inventar divertidos concursos sobre adivinar películas viendo pequeños segmentos de estas – y esto es tan solo un ejemplo.

Figura 2 – Pantalla de Card [pestaña Basic]

Figura 3 – Pantalla de Card [pestaña Answers]

El juego de aprendizaje es donde realmente sucede la acción. El usuario necesitará un catálogo antes de poder lanzarse a jugar. La mecánica del juego es sencilla: el jugador viajará por mapas laberínticos (Figura 4), buscando la llave y la salida que le llevará al siguiente nivel, mientras se sucederán batallas aleatorias; en

estas batallas, uno o varios desafíos del catálogo escogido aparecerán ante el jugador, estando obligado este a contestarlos debidamente. A medida que se conteste correctamente a los desafíos (Figura 5), los mismos aparecerán con menos frecuencia. Si, por el contrario, el usuario es incapaz de responder bien, perderá una parte de su barra de energía y el algoritmo hará que estos desafíos salgan más a menudo, análogamente al método de Leitner.

La responsabilidad de la elección de los desafíos recae sobre la interfaz SRS; mediante un algoritmo matemático de puntuación interno se realiza una selección de las fichas del catálogo que necesitan ser repasadas. Este sistema de puntos es muy importante de cara a implantar el SRS de Sebastian Leitner. Surge de la necesidad del software de gestionar y seleccionar los conceptos que, según él mismo, se necesitan aprender. La aplicación realiza una selección de entre todas las fichas que se disponen, dando preferencia primero a fichas completamente nuevas (las que se presentan para estudiar por primera vez) y luego a las fichas que tengan menor puntuación (ejecutando un algoritmo de ordenación *Quick-Sort*) **[5]** o lo que es lo mismo, mostrando con más frecuencia desafíos que el jugador tiende a fallar o que no logra memorizar del todo.

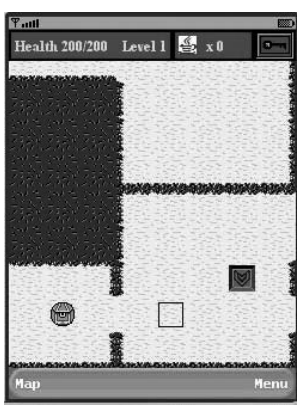

Figura 4 – Laberinto del juego

Figura 5 – Pantalla de batalla

Una de las partes más relevantes y extensas del proyecto es la creación y población de los mapas. La decisión más importante ha sido decantarse por el contenido procedimental (*procedural content*), una idea olvidada muy a menudo por las grandes compañías que desarrollan juegos, pero casi un requisito obligatorio para el programador *stand-alone*. El contenido procedimental, es básicamente contenido creado por la máquina, a diferencia del contenido generado por los artistas o diseñadores gráficos – unos magníficos ejemplos que impresionaron precisamente por su extensión o variedad debido al contenido procedimental fueron los videojuegos *Elite* o el reciente *Spore* **[6]**. A grandes rasgos, introducimos un número de 7 cifras (semilla) en el punto de entrada de un conjunto de algoritmos y operaciones previamente diseñadas. Estos métodos se encargan de tratar las cifras a lo largo de distintas etapas, produciendo un mapa completo con todos los elementos necesarios a su salida. Cada una de las múltiples combinaciones de números generará un *output* aleatorio, distinto de los demás, pero fiel a sí mismo – un valor dado siempre creará el mismo mapa. Se pueden detallar 3 ventajas principales a partir de esta elección:

1. Ahorro de espacio – un terminal móvil es un dispositivo muy limitado en términos de memoria y tamaño de la aplicación (archivo .jar). Si el contenido es generado de manera aleatoria, no será necesario almacenar todos los datos relacionados con los mapas.

2. Evitar la monotonía – si cada mapa es único y los elementos en él son emplazados de manera completamente diferente, se evita que el usuario se aburra o tenga una ventaja al reconocer mapas o patrones repetidos. Sin duda, un punto a favor frente a otros productos y una garantía para la competitividad entre usuarios.

3. Número de niveles – el programa puede amoldar el número de niveles al tamaño de los catálogos estudiados por el usuario; mientras siga habiendo fichas sin estudiar, el generador continuará creando mapas aleatorios. Además, no será necesario un grupo de personas que desarrollen los niveles, ni decidir un número seguro de niveles prefabricados para tamaños grandes de catálogo.

El algoritmo para producir un mapa aleatorio está constituido por varias secciones independientes con sus propios métodos cada una. Como en cualquier caso de este tipo, existían 2 planteamientos a seguir para conseguir el resultado deseado: *top-down* y *bottom-up*. El primer modelo implica empezar con una habitación del tamaño de nuestro mapa e ir generando dentro de ella el resto del contenido; por el contrario, el segundo modo comienza con una habitación pequeña inicial a la que se van anexionando otras formas geométricas. En e-Mazing se utiliza la aproximación *top-down*.

En un principio, se genera un cuadrado pasándole las dimensiones necesarias en tiles (un tile es una sección que representa detalles, texturas o gráficos de un fragmento del mapa). A partir de esta figura geométrica, se quiere obtener un entramado de habitaciones y pasillos a recorrer y, para ello, el primer paso es trocear el cuadrado horizontal y verticalmente. Los puntos exactos y el orden en el que se trocea son completamente dependientes de los números de la semilla. Además, las habitaciones resultantes vuelven a recorrer el método para ser segmentadas hasta que alcanzan unas dimensiones mínimas controladas por el tamaño del mapa y de los *tiles*.

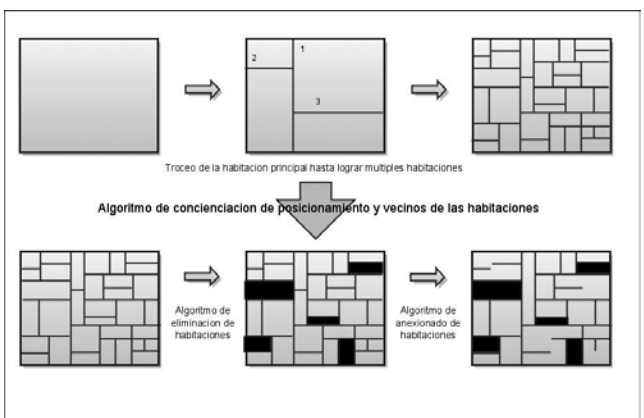

Figura 6 – Proceso de generación de mapas laberínticos aleatorios

Tras la obtención de las habitaciones cuadradas y rectangulares se propone embellecer y añadir más variedad al mapa. Primeramente, la ejecución de un algoritmo matemático y varios cálculos de posicionamiento vuelven conscientes a todas estas habitaciones de sus vecinos lindantes, así como de a qué costado se encuentran estos y cuáles son la longitud y coordenadas del muro que comparten. Con estos datos en poder, se procede a eliminar un número de habitaciones del mapa, la posición y la cantidad vienen determinadas por la semilla. Conocer a los vecinos es importante en este punto para evitar que se queden habitaciones "descolgadas" o rodeadas debido a los cuartos que se están eliminando. Por otro lado, no todas las habitaciones van a ser cuadradas o rectangulares; estando al corriente de sus vecinos, algunos cuartos pueden fundirse con habitaciones lindantes y así sucesivamente, produciendo esta unión formas y tamaños muy variados para aportar variedad al mapa (Figura 6). De nuevo, la decisión de estas acciones depende de los valores de la semilla. Para que el jugador pueda visitar todas estas habitaciones que se han generado, es necesario que sobre el mapa se apliquen varios métodos, que juntos forman el algoritmo de transitabilidad. Este algoritmo se encarga de que todas las habitaciones estén unidas por medio de puertas, creando varios caminos y retrocediendo para buscar otros nuevos si detecta que aún restan habitaciones desconexas. En caso de ser imposible (o computacionalmente costoso) unir algunos cuartos, dicho algoritmo los eliminará como si de uno de los métodos anteriormente explicados se tratase. Asimismo, la creación de puertas permite "concienciar" a los cuartos de los nexos que las unen a partir de este momento (Figura 7).

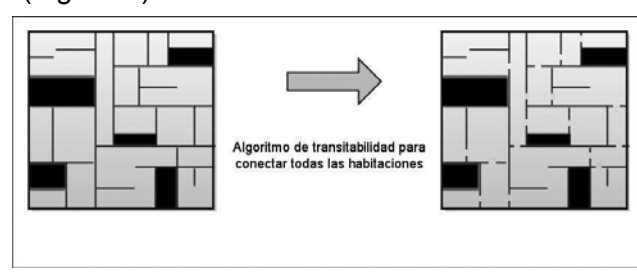

Figura 7 – Algoritmo de transitabilidad que conectará las habitaciones

En este mapa (Figura 8) sustituiremos los vectores por los *tiles* que presentamos bajo estas líneas (Figura 9).

Una vez en este punto, todos estos datos y

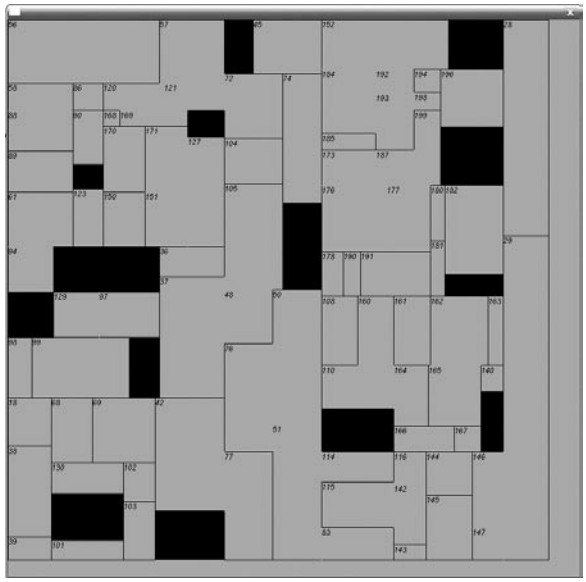

Figura 8 – Un mapa de prueba durante las primeras versiones del generador

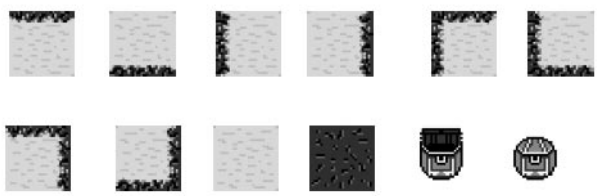

Figura 9 – Los distintos tiles que hemos dibujado para modelar el mapa.

estructuras están listos para convertirse en una imagen tangible. Por medio de nuevas funciones, todos los cuartos son analizados junto con sus vecinos, puertas y datos restantes, seleccionando el tile que les correspondería visualmente y transformando de esta manera la imagen (*tiles* en la página anterior). Para acabar el mapa se han de colocar en él los elementos restantes, dándole sentido y objetivos. Por un lado, se emplazan tres cofres en posiciones decididas por el algoritmo, cuyo contenido podrá ser reclamado por el jugador: en su interior puede haber una llave que permite el avance al siguiente nivel, un elemento de curación o nada en absoluto. También de manera aleatoria, son colocados en el mapa el punto de inicio y el de final. El primero de ellos permitirá retroceder a los niveles anteriores (siempre y cuando existan) y el segundo transportará al jugador hasta el siguiente laberinto (Figura 11), sólo si previamente ha obtenido la llave del mismo nivel. Todo esto está englobado en un sistema mayor al que se ha denominado *World Engine* (Motor del mundo), encargado de invocar la creación de mapas, situar y desplazar al jugador por el ambiente, controlar los datos del usuario o su interacción con los cofres y de manejar los eventos y batallas que suceden.

A partir de aquí, entra en juego el usuario (y

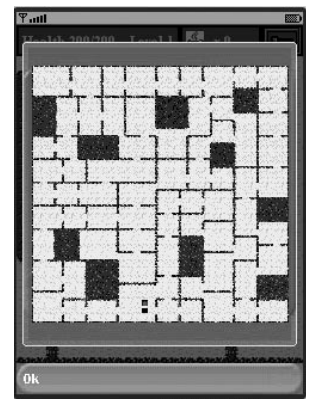

Figura 10 – La herramienta "minimapa" que permite ver todo el nivel laberíntico en el que nos encontramos

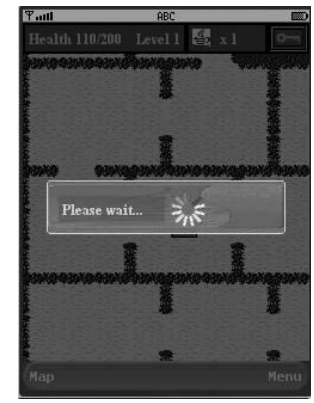

Figura 11 – El widget de carga que aparece mientras cambiamos de nivel

nunca mejor dicho). Tal y como se ha explicado, su objetivo es recorrer estos laberintos buscando la salida, a la vez que aprende el material educativo que ha escogido. Cada cierto número de pasos el motor del mundo generará un evento, en el cual se presenta delante del usuario una pantalla con varios de los desafíos de las fichas que se pretende estudiar. El jugador podrá leer el texto del desafío que él mismo introdujo, así como cualquier contenido multimedia que decidiera adjuntar a la ficha en cuestión. Aquí el cometido es contestar correctamente al mayor número posible de desafíos, teniendo en cuenta que se muestran de uno en uno y sin límite de tiempo. Inmediatamente después de introducir una respuesta, el programa la evalúa e informa al jugador de su veracidad, sumando algunos puntos a la ficha del desafío correspondiente si se ha acertado, o restándoselos y mermando la energía del jugador en caso contrario. Por supuesto, el motivo para penalizar la energía del usuario después de fallar una respuesta no es otra que la de incentivar el estudio, evitando que se realice lo que se conoce en términos de juego como un *rush* (carrera a por todas) a través del juego, algo que impediría aprender los conceptos. En caso de que la energía de

un jugador llegue a cero durante la partida el juego finalizará, sin embargo gracias al gestor de partidas que está implementado en la aplicación se podrán guardar y cargar partidas en cualquier momento.

Para proveer de una aplicación más agradable y completa, se ha añadido un entorno gráfico en el cuál el jugador puede ver la energía que le queda, el número de elementos que reponen energía que tiene en su poder (son pequeños cafés Java) o si dispone de la llave del nivel en el que se encuentra o no. Adicionalmente, se puede acceder a un minimapa (Figura 10) del nivel completo sobre el que se muestra la localización del usuario, y a un conjunto de ventanas que muestran las fichas que se están estudiando, con la finalidad de poder repasar en cualquier momento y no solo en las batallas o eventos. Con la intensión de amoldar el ritmo de estudio a las necesidades de cada usuario (Figura 13), se puede acceder a un menú en el que asignar las puntuaciones que se suman o restan ante aciertos o fallos, los puntos extra que se ganan o pierden si se acierta un desafío que aparece por primera vez o si se falla uno con máxima puntuación (y por consiguiente supuestamente aprendido). También se puede activar un repaso de las fichas trabajadas cada vez que se reanuda el juego y definir el porcentaje de ellas que se van a revisar. Por último, hay dos modos de dificultad, entre los cuales varía la energía máxima del jugador, la frecuencia y el tamaño de las batallas, la energía que se pierde al fallar una respuesta o el número de fichas que se activan para ser aprendidas a cada nivel. En conjunto, se ha prestado especial atención en crear un producto para todas las edades y exigencias, que permite ser aplicado en casi cualquier campo.

CONCLUSIONES

En este artículo se ha descrito el proceso de creación de un videojuego para plataformas móviles, cuyo objetivo es fomentar y hacer más ameno el aprendizaje.

Se ha obtenido un producto acabado y funcional con dos características principales: un gestor de información textual y/o audiovisual y un juego que no brilla por sus gráficos e innovación, sino por su entretenimiento. De este modo, se está tratando de demostrar que la capacidad de retención de las personas varía en función de su estado anímico e interés, a la vez que se aplican los interesantes métodos de Leitner. Por un lado, trayendo los SRS al mercado de los terminales móviles se está avanzando frente a los programas educativos de tirón comercial actuales y dando un paso adelante en materia de herramientas de estudio (Figura 12). Por otro lado, este proyecto trata de sentar las bases para una plataforma de aprendizaje mucho más grande, en la cual pueden coexistir varios juegos o modos de aprendizaje para un mismo gestor de temáticas. La realidad más cercana, el siguiente paso, es la inclusión de un sistema de puntuaciones para incentivar aun más a los jugadores, pudiendo competir o compartir sus puntos *on-line*, participando en eventos con semillas y temarios idénticos a través de un servidor http o una red social como *facebook* por ejemplo.

Figura 12 – Video almacenado en una ficha

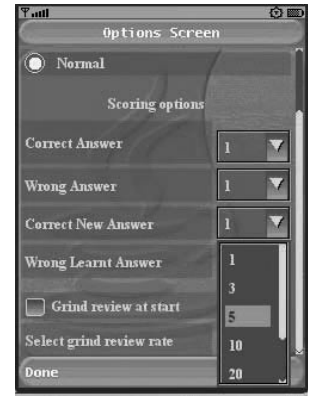

Figura 13 – Opciones del juego

En definitiva, a partir de este punto de inflexión las posibilidades de expansión son enormes, lo que sí está garantizado es que los usuarios dispondrán en todo momento de un software de aprendizaje personal y que se adapta a las necesidades de cada cliente.

BIBLIOGRAFÍA

1. J. WELLSH, Martin. *J2ME Game Programming*, USA: Premier Press, 2004.

2. KEOGH, James. *J2ME The Complete Reference*, USA: McGraw-Hill, 2003.

3. SUN Microsystems, *LWUIT – Developer Guide*, Worldwide: Sun Microsystems Inc., 2008.

4. LWUIT Forum, http://forums.java.net/jive/forum.jspa?forumID=139 , 2009.

REFERENCIAS

[1] http://en.wikipedia.org/wiki/Leitner_System

[2] http://java.sun.com/javase/reference/index.jsp

[3] http://java.sun.com/javame/index.jsp

[4] http://developers.sun.com/mobility/midp/articles/midp20/

[5] http://en.wikipedia.org/wiki/Quicksort | http://www.cs.ubc.ca/~harrison/Java/sorting-demo.html

[6] http://en.wikipedia.org/wiki/Procedural_generation#Contemporary_application

AUTORES

Mario Viktorov Mechoulam es titulado en Ingeniería Técnica de Telecomunicaciones, especializán-dose en Telemática con el proyecto de carrera aquí presente. Cursando sus estudios en la EPSC de Castelldefels, despierto y motivado por proyectos novedosos relacionados con el networking, terminales móviles, plataformas de mensajería, Java o aplicaciones web distribuidas, ha formado parte en varios proyectos de empresas tanto nacionales como extranjeras. Actualmente, desempeña su labor de investigación en el proyecto ViCoMo para el departamento de Sertel en el Campus Nord de Barcelona.

Josep Pegueroles nació en Tortosa (España) en 1974. Recibió el título de Ingeniero de Telecomunicación en 1999 y obtuvo el título de doctor en 2003. Ambos por la Universidad Politécnica de Cataluña (UPC), Barcelona, España. En 1999 entró a formar parte del Grupo de Seguridad de la Información - ISG (http://isg.upc.es) de la Línea de Investigación de Servicios Telemáticos - SERTEL (http://sertel.upc.es) del Departamento de Ingeniería Telemática - ENTEL (http://entel.upc.es) de la UPC (http://www.upc.es). Actualmente es Profesor Titular de Universidad en la Escuela Técnica Superior de Ingenieros de Telecomunicación de Barcelona - ETSETB (http://www.etsetb.upc.es). Sus intereses de investigación incluyen la seguridad para servicios multimedia en red y las comunicaciones seguras de grupo.

Juan Hernández Serrano nació en Salamanca (España) en 1979. Recibió el título de Ingeniero de Telecomunicación en 2002 y obtuvo el título de doctor en 2008. Ambos por la Universidad Politécnica de Cataluña (UPC), Barcelona, España. En 2002 entró a formar parte del Grupo de Seguridad de la Información - ISG (http://isg.upc.es) de la Línea de Investigación de Servicios Telemáticos - SERTEL (http://sertel.upc.es) del Departamento de Ingeniería Telemática - ENTEL (http://entel.upc.es) de la UPC (http://www.upc.es). Actualmente es profesor de Universidad en la Escuela Politécnica Superior de Castelldefels - EPSC (http://epsc.upc.es). Sus intereses de investigación incluyen la seguridad para servicios multimedia en red y las comunicaciones seguras de grupo.