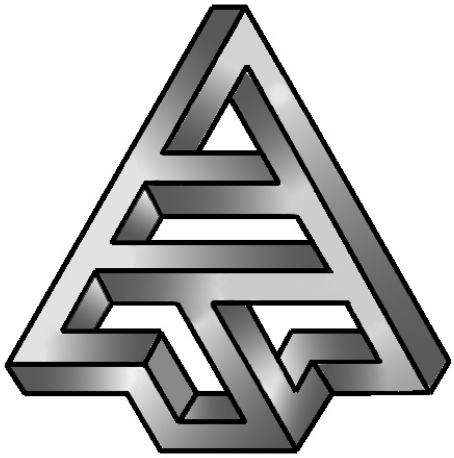
Sistemas Informáticos Curso 2005-2006

Vega Solaris: The Remake



Sergio Díaz Jubera
Javier Gallego Ahijón
José María Sobrinos García

Dirigido por:
Prof. Fernando Sáenz Pérez
Dpto. Sistemas Informáticos y Programación

Facultad de Informática
Universidad Complutense de Madrid

ÍNDICE

Resumen

1.RESUMEN EN CASTELLANO

Miembros integrantes del grupo

- Sergio Díaz Jubera.
- Javier Gallego Ahijón.
- José María Sobrinos García.

Título

Desarrollo de un videojuego para plataformas de telefonía móvil y PC.

Resumen

El propósito de este proyecto fue desarrollar el remake para móviles y ordenadores personales del videojuego Vega Solaris. El lenguaje para el desarrollo del videojuego sería Java debido a su característica de ser un lenguaje multiplataforma.

Vega Solaris fue un juego desarrollado por Fernando Sáenz Pérez y Carlos García en el año 1987. Dinamic Multimedia se interesó por el videojuego y después de dos años, el videojuego se promocionó para Spectrum en el año 1989.

El primer objetivo del proyecto fue desarrollar un prototipo de la aplicación para PC. El segundo objetivo fue la instalación y ejecución correctas del videojuego en teléfonos móviles.

El sistema podría ejecutarse en distintos teléfonos móviles con diferentes características (tamaño de pantalla, sonido, diferente capacidad de memoria para juegos, etc.). Aunque todos los móviles en los que se ejecute el sistema deberían cumplir requisitos mínimos como máquina de Java integrada y gráficos en color.

Se parametrizarían los principales aspectos del videojuego para convertirlo en un motor de desarrollo de videojuegos. Se realizaría un manual de parametrizaciones para que los desarrolladores puedan crear nuevas versiones o nuevos videojuegos configurando determinados archivos de la nueva versión.

Aspectos destacables del desarrollo serían:

- La aplicación de técnicas de inteligencia artificial para el control automático de personajes.

- Control del juego en red (Bluetooth), puesto que el videojuego es multijugador.

- Como resultados del proyecto se tendrán:
- Prototipo del videojuego para PC.
- Versión del videojuego para móviles.
- Documentación: manuales de usuario, manual de diseño (especificación de requisitos, diagramas de análisis...), manual de parametrizaciones (motor de desarrollo de videojuegos).

Los objetivos de este proyecto se han cumplido en su totalidad. El videojuego funciona sobre varias plataformas móviles. Además, se ha ampliado al videojuego con las técnicas más recientes de inteligencia artificial.

El videojuego despertó el interés de los aficionados a la retroinformática desde el primer momento y el profesor Fernando Sáenz Pérez y el grupo fueron invitados a la feria de retroinformática MadriSX&Retro06. El videjuego fue presentado en una sala llena de gente y se aprovechó para realizar una realimentación a partir de las opiniones de los expertos en videojuegos y de los jugadores habituales.

Pensando en la comercialización del videojuego para teléfonos móviles, se realizó un estudio del mercado en el que se podría vender el videojuego, de los beneficios que se obtendrían y de la inversión financiera que se debería llevar a cabo para su propaganda y venta.

El videojuego se ha desarrollado con la misma estructura de clases que utilizan las grandes empresas de desarrollo software. La aplicación tiene una estructura en tres capas (vista, lógica y datos). Se ha utilizado patrones de diseño estándar y otros se han adaptado a las necesidades particulares del proyecto.

El resultado es un videojuego que refleja el espíritu original de Vega Solaris, se ejecuta en teléfonos móviles y permite a futuros desarrolladores configurar multitud de sus aspectos (fondos, gráficos, fotogramas, tamaño del mapa, objetos de cada habitación).

2. ENGLISH SUMMARY

Members of group

- Sergio Díaz Jubera.
- Javier Gallego Ahijón.
- José María Sobrinos García.

Title

Development of a videogame for mobile and personal computer platforms.

Summary

The objective of this project was the development of the remake of the videogame Vega Solaris for mobile phones and personal computers. The videogame would be developed with the Java language due to its capacities as multi platform language.

Vega Solaris was a game developed by Fernando Sáenz Pérez and Carlos García in 1987. Dynamic Multimedia was interested in this videogame and two year later, this videogame was promoting in 1989 for Spectrum.

The first objective of the project was the prototype for PCs development. The second and most important objective was the installation and execution of the videogame on mobile phones.

This system could run in different mobile phones with different characteristics (screen size, sound, memory size for games, etc.). Although all the mobile phones that running this system must carry out minimum requirements, such the Integrated Virtual Java Machine and colour screen.

Main aspects of the videogame would be parametrized for converting the videogame into a game development engine. A parameter's manual would be made to create new versions and new videogames shaping some files from the new version.

Important aspects of development would be:

- Use artificial intelligence techniques to generate automatic control of players.
- Network-game control (Bluetooth) due to this videogame is a multiplayer game.

- The results of this project are:
- Videogame version for mobile phones.

- Prototype for personal computer.
- Documentation: user guide, design guide (requirements specification, analysis diagrams…), parameter manual (videogame development engine).

The objectives of this project have been accomplished totally. The videogame works over mobile phones. Besides, the videogame has been extended with more recent techniques of artificial intelligence.

This videogame woke up interest of retro computing fan from the beginning, and the Professor Fernando Sáenz Pérez and the project group was invited to retro computing party MadriSX&Retro06. This videogame was introduced in a room full of persons and we seized the opportunity to re-feeding with experts and usual players' opinion.

In order to commerce the videogame for mobile phones, we did a market studio where application would be sold, a profits studio that the application would get and financial invert studio to promote and sale the videogame.

This videogame has been developed with the same class structure as big software developed enterprises use. This application has a three level structure (view, logic and dates). We have standard design patterns and other ones have been adapted to the project needs.

The result is a videogame that maintains original spirit of Vega Solaris, it runs over mobile phones and multiple aspects can configured (backgrounds, graphics, frames, map size, objects for each room) as user wants.

Descripción Del Proyecto

1. DESCRIPCIÓN

Este proyecto consiste en generar una nueva versión de un sistema ya existente. Además se deberán ampliar las funciones que tenía el sistema inicialmente siempre que eso sirva para mejorar la jugabilidad de la aplicación y no modifique el espíritu original del juego.

Uno de los aspectos que se han ampliado es la modularidad del diseño del código del videojuego. En las dos versiones nuevas, realizar amplicaciones del juego es un proceso sumamente sencillo y rápido. Ahora se pueden cambiar todos los gráficos, crear nuevos mapas para el juego, introducir nuevas armas, objetos y enemigos; se puede cambiar la inteligencia de los enemigos e introducir algoritmos que hagan a los enemigos más inteligentes...

Las herramientas utilizadas para el desarrollo son:

- J2SE 5.0 JDK (http://java.sun.com/javase/downloads/index.jsp): API de desarrollo de software en Java para PC.
- Sun Java Wireless Toolkit 2.2 (http://java.sun.com/products/sjwtoolkit/): entorno de desarrollo de software en Java para móviles.
- Adobe Photoshop CS2 (http://www.adobe.com/es/products/photoshop): herramienta de tratamiento de imágenes.
- Eclipse 3.1.2 (http://www.eclipse.org): entorno de desarrollo integrado (IDE) para crear aplicaciones clientes de cualquier tipo.
- EclipseUML (http://www.omondo.com/): plugin para Eclipse para la creación de diagramas UML.
- EclipseME (http://eclipseme.org/): plugin para Eclipse que conecta éste con Sun Java Wireless Toolkit 2.2.
- ArtIcons (http://www.aha-soft.com/spanish/articons/index.htm): herramienta de creación y edición de iconos y pequeñas imágenes.
- ZX SPIN (http://www.zxspin.co.uk/): emulador de Spectrum para PC.
- Windows XP Professional Edition (www.microsoft.com/spain/windowsxp/default.asp): sistema operativo para PC.
- Microsoft Word (www.microsoft.com/spain/office/): herramienta de procesamiento de textos.
- Nokia PC Suite (www.nokia.es/soporte/pc_suite.jsp): herramienta de conexión entre PC's y móviles de la marca Nokia.
- SnagIt 8 (www.techsmith.com/snagit.asp): herramienta de capturación de imágenes y videos de pantalla.

- ProGuard (http://proguard.sourceforge.net/): herramienta de ofuscación de código Java.
- *Escaneador de habitaciones:* herramienta web, creada durante el desarrollo y diseñada específicamente para capturar los escenarios de la versión original de Vega Solaris.

2. HISTORIAL DE VERSIONES

12 / Diciembre / 2006 - Versión 0.1 (PC)

Versión Vega Solaris para ordenadro personal con las pantallas de presentación y los controladores. También se habían capturado todos los gráficos del juego original y se había realizado el proceso de obtención de requisitos y la decisión de los aspectos del juego a parametrizar.

14 / Abril / 2006 - Versión 0.2 (PC)

Versión Vega Solaris para ordenador personal al que se le han añadido enemigos, mapa totalmente capturado y pasado a archivos de habitaciones, lógica del juego y del personaje protagonista. Falta la inteligencia artificial del jugador oponente.

13 / Junio / 2006 - Versión 1.0 (PC)

Se corrigen los errores en la versión final. Se introducen los algoritmos de inteligencia artificial para mover al personaje oponente.

10 / Febrero /2006 - Versión 0.1 (teléfono móvil)

Versión vega Solaris para teléfonos móviles conlas pantallas de presentación y controladores.

15 / Junio / 2006 - Versión 0.2 (teléfono móvil)

Se ha añadido el juego monojugador correctamente.

30 / Junio / 2006 - Versión 1.0 (teléfono móvil)

Apliacación completa Vega Solaris para teléfono móvil con partidas multijugador vía Bluetooth incorporadas correctamente.

3. ESPECIFICACIONES TÉCNICAS

3.1. Requisitos mínimos

Versión PC

- Windows ME, 2000, XP, Linux.
- JDK 1.5.
- Pentium 750 MHZ.
- 128 MB RAM.

Versión teléfono móvil

- Pantalla en color.
- Máquina virtual de Java incorporada (soporte para juegos Java) con CLDC 1.0 y MIDP 2.0.
- Bluetooth.

3.2. Requisitos recomendados

Versión PC

- Windows ME, 2000, XP, Linux.
- JDK 1.5 o superior.
- Pentium 3200 MHZ.
- 2036 MB RAM.

Versión teléfono móvil

- Pantalla en color.
- Máquina virtual de Java incorporada (soporte para juegos Java) con CLDC 1.1 y MIDP 2.0.
- Bluetooth.

3.3. Instalación

Versión PC

Para instalar la aplicación en un ordenador personal:

1. Copiar los archivos en la carpeta que se desee.
2. Abrir la carpeta Vega Solaris PC dentro del directorio donde se instaló la aplicación.
3. Ejecutar el archivo por lotes “ejecutar.bat”.

Versión teléfono móvil

Consultar la documentación del dispositivo móvil específico sobre cómo instalar aplicaciones y/o juegos Java en su teléfono.

3.4. Ejecución

Versión PC

Ejecutar el fichero Vega Solaris PC/ejecutar.bat.

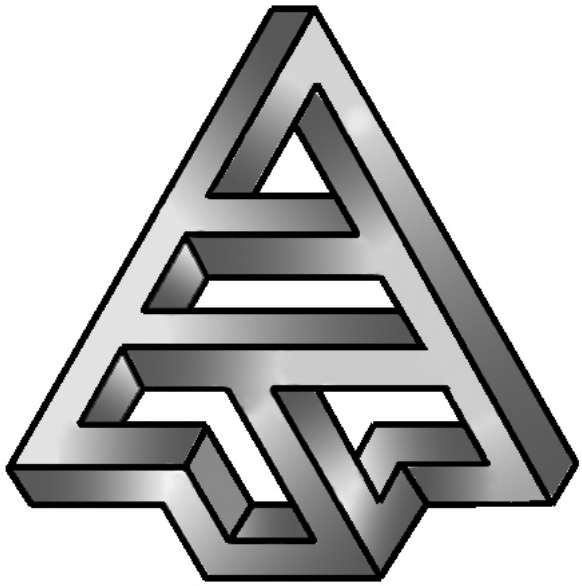
Versión teléfono móvil

Consultar la documentación del dispositivo móvil específico sobre cómo ejecutar aplicaciones y/o juegos Java en su teléfono.

3.5. Archivos de configuración

El juego se configura automáticamente adaptándose al tamaño de la ventana del móvil sobre el que se ejecuta. El videojuego detecta cual es el tamaño máximo de la pantalla y elige el tamaño máximo en el que cabe una pantalla del juego.

Los aspectos relacionados con los cambios en los archivos de gráficos, fondos, tamaño del mapa o contenido de las habitaciones se explican en profundidad en el apartado Manual de parametrizaciones de este mismo documento.



Vega Solaris

Plan de Proyecto

Versión 1.0
04/07/2006

ÍNDICE

1. INTRODUCCIÓN

1.1. Propósito del plan

El objetivo del presente proyecto es la realización de un videojuego para teléfonos móviles. La aplicación será una nueva y actualizada versión del juego para Spectrum "Vega Solaris". Las opciones que se incorporarán a esta versión será la de ejecutarse en teléfonos móviles y jugar partidas multijugador entre varios jugadores mediante Bluetooth.

1.2. Ámbito del proyecto y objetivos

1.2.1. Declaración del ámbito

Contexto

Nuestra aplicación es monousuario para partidas de un solo jugador en un solo móvil y permite partidas multijugador con otras aplicaciones del mismo juego en otros teléfonos móviles. Se ejecutará en teléfonos móviles compatibles con la versión J2ME (configuración CLDC con el perfil MIDP), y recursos de procesamiento que se suponen suficientes. Para el juego en red el teléfono móvil deberá de disponer de capacidad de conectarse con otros móviles mediante Bluetooth. Por tanto este aspecto no impone ninguna restricción a la hora de abordar el proyecto.

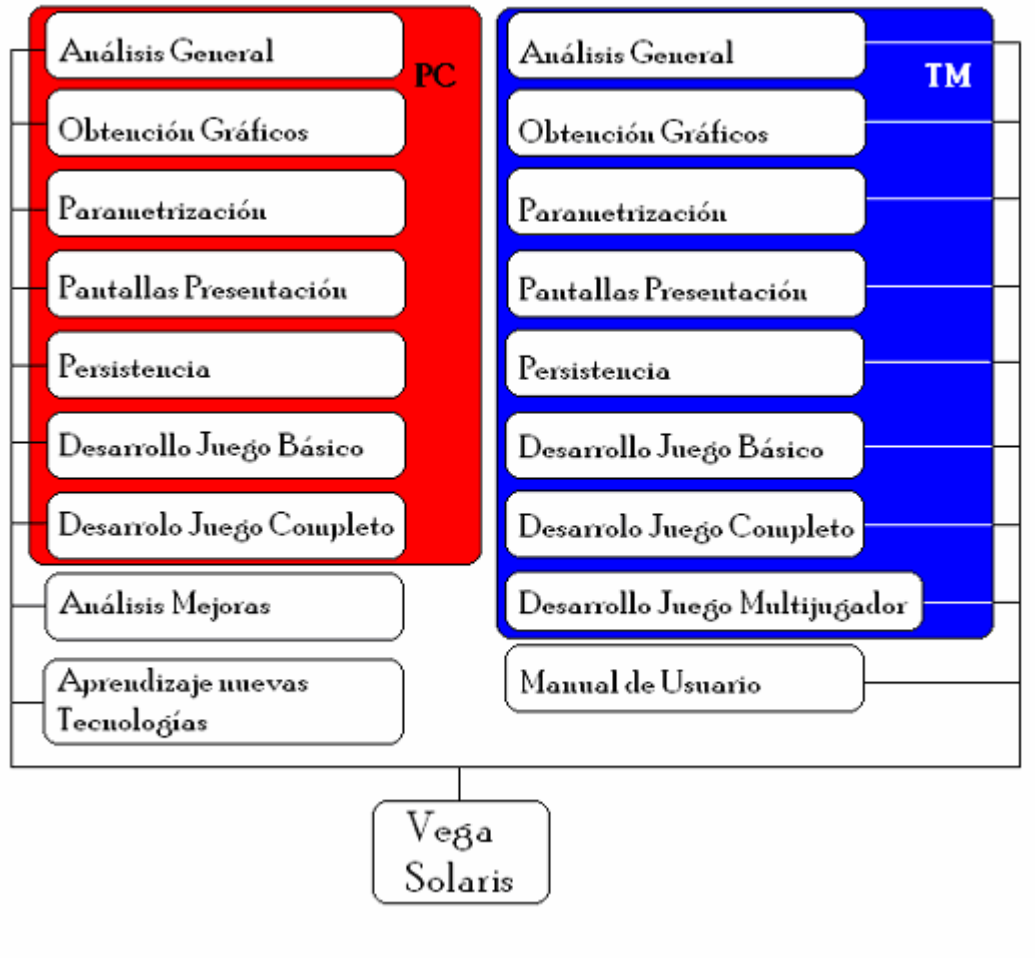
Sin embargo, los recursos de memoria son limitados (las memorias para aplicaciones en los móviles varían alrededor de los 256 KB) y deberán ser tenidos en cuenta a lo largo del desarrollo de la aplicación (definiendo como aceptable un tamaño de la aplicación de menos de entre 40 y 70 KB).

Objetivos de información

Los objetos de entrada serán pulsaciones de teclas introducidos mediante un teclado, correspondientes a las opciones del menú deseadas, los movimientos y las acciones del personaje. Los objetos de salida serán una serie de datos alfanuméricos e imágenes. Los datos alfanuméricos corresponden al resultado de la partida, puntuaciones, pantallas de ayuda, mensajes y opciones de menús. Las imágenes corresponden a los movimientos, escenario y características de los personajes (animaciones, vida, objetos y armas) durante el juego.

Los datos visibles al cliente serán los que se necesiten para una correcta interacción con el juego, su desarrollo y su conclusión.

1.2.2. Funciones principales



El juego Vega Solaris va a ser desarrollado para dos soportes: el prototipo inicial se ejecutará en ordenadores personales, y la versión definitiva se ejecutará en teléfonos móviles. Hay funciones que se van a realizar dos veces, una vez en la versión para PC y otra vez para la versión para móviles. Estas funciones repetidas se han marcado englobándolas en rectángulos de color distinto. El rectángulo rojo son las funciones realizadas en el desarrollo para ordenador personal. Las funciones del teléfono móvil son las englobadas en el rectángulo azul.

Estas funciones se corresponden con las tareas de la Planificación Temporal, y no con las funciones del producto en sí (dichas funciones se tratan en el documento de especificación de requisitos).

Módulo de análisis general (desarrollo del esqueleto básico de la aplicación)

Diseño de la estructura básica del juego. Esta estructura será la que se irá completando en las fases posteriores. Esta función se lleva a cabo en ambas versiones.

Módulo de obtención de gráficos

En este módulo se tomarán del juego original los gráficos para el juego y se tratarán para adaptarlos al desarrollo en J2ME. Esta función se lleva a cabo en la versión para PC y

para teléfono móvil, aunque queda abierta la opción de usar los mismos gráficos en ambas versiones.

Módulo de parametrización de características

Se definirán los elementos del juego que se pueden parametrizar (mapa, personajes, enemigos...). Se establecerán las características de cada uno de estos parámetros y sus efectos en el juego para cada uno de los posibles valores que tomen. En principio esta función se lleva a cabo en la versión para PC y en la versión para teléfono móvil, aunque también se recoge la opción de que los parámetros para ambas versiones sean los mismos.

Módulo de desarrollo de las pantallas de presentación

Se desarrollará una primera versión del juego en el cual aparezcan las pantallas con el título del juego, pantalla de fin de juego ("Game over"), pantalla de opciones, etc. En resumen, todas las pantallas excepto la pantalla de juego propiamente dicha, en la que saldría el muñeco del humano/alienígena luchando uno contra otro por conseguir el talismán Vega Solaris. Esta función se lleva a cabo en ambas versiones.

Módulo de gestión de la persistencia

Aquí se definirán los datos que la aplicación leerá o escribirá en archivos externos. Estos datos se cargarán al lanzar el juego de nuevo y representan características que pueden interesar al usuario guardar (puntuaciones máximas, créditos, mapa del juego, etc.). Esta función se lleva a cabo en ambas versiones.

Módulo de desarrollo del juego básico

Se creará una segunda versión que además de incluir todas las pantallas de presentación y menús, incluirá una pantalla de "interacción" que muestre el mapa y al personaje principal. Se podrá navegar con el personaje por toda la pantalla, aunque no aparecerán enemigos ni el rival alienígena con el que luchar. El tipo de partida corresponderá a una partida de monojugador. Esta función se lleva a cabo en ambas versiones.

Módulo de desarrollo del juego completo

Se incluyen en la segunda versión la aparición de enemigos, el rival alienígena/humano, objetos y armas. En esta tercera versión, estará completa la versión monojugador de PC. Esta función se lleva a cabo en ambas versiones.

Además de estas fases hay una función que se lleva a cabo en la versión del juego para teléfono móvil, ha sido colocada dentro del rectángulo azul y se trata de el desarrollo del jueego multijugador para la versión de teléfono móvil. Esta versión no aparece en la versión para ordenador personal porque no hay herramientas software que permitan emularlo y el desarrollo de una excedería el tiempo de realización del proyecto.

Módulo de desarrollo del juego multijugador

Desarrollo del juego para dos jugadores. La opción multijugador permite el juego entre dos jugadores a través de una conexión Bluetooth entre los teléfonos móviles. Esta función sólo se realiza en la versión para teléfono móvil.

Tras generar la versión monojugador para PC se procederá a desarrollar la versión para teléfono móvil del juego. En esta versión se implementará el juego mediante las mismas fases anteriores de la creación del juego para PC.

Hay otras funciones que no se repiten una vez en cada versión. Estas funciones se han situado aparte, y no dentro de uno de los dos rectángulos, para indicar este hecho.

Análisis de mejoras

Tras acabar la versión final del juego para PC, se realizará esta fase donde se estudiarán los aspectos del juego que se pueden mejorar o actualizar para la versión definitiva en teléfono móvil. Se realizará una lista con los posibles aspectos a mejorar y las posibles alternativas. Los cambios serán recogidos y se incluirán en la versión final para teléfono móvil de Vega Solaris.

Aprendizaje de nuevas tecnologías (J2ME y Bluetooth)

En esta fase, los integrantes del grupo de desarrollo se documentarán sobre la tecnología J2ME y la tecnología Bluettoth. La tecnología J2ME es la que se utilizará para programar el juego para teléfonos móviles. Despúes de recopilar información, los integrantes del grupo se dedicarán a estudiar y aprender todo lo necesario para utilizar J2ME para programar el juego en teléfonos móviles. La tecnología Bluetooth combinada con J2ME permite comunicar dos móviles entre sí. El aprendizaje de cómo llevar a cabo esta conexión también se recogerá en esta fase.

Manual de usuario

Durante todo el desarrollo de la aplicación se irá recogiendo información sobre el funcionamiento y manejo del juego. El objetivo de esto es crear un documento formal que pueda ser utilizado por el usuario final como manual de instrucciones para jugar.

1.2.3. Aspectos de rendimiento

No hay ningún aspecto crítico de rendimiento a tener en cuenta para el desarrollo del sistema. Tan sólo hay que resaltar que la aplicación debe tener tiempos de respuesta razonables y la memoria física consumida debe ser también razonable.

1.3. Modelo de proceso

Para este proyecto se ha elegido el ampliamente usado *modelo en espiral*, en concreto la *variante de Boston*, un tipo de modelo de proceso evolutivo que se caracteriza por

adaptarse a la naturaleza cambiante del desarrollo del software y por ser iterativo, es decir, se realizan varios incrementos en cada uno de los cuales se desarrollan versiones del sistema final más completas.

El modelo en espiral de Boston está compuesto por seis actividades estructurales fijas, a diferencia del modelo en espiral original – de Boehm. Estas actividades son las siguientes: *Comunicación con el Cliente*, *Planificación*, *Análisis de los Riesgos*, *Ingeniería*, *Construcción y Adaptación* y *Evaluación por el Cliente*. Cada una de estas actividades estructurales se dividirá en diversas tareas que servirán como método de adaptación del modelo genérico a este proyecto en particular, y que se aparecerán en la tabla EDT (*véase apartado 4.1*). Cabe destacar que se ha decidido incluir en una misma tarea TUE y Obtención de SRS, pues son dos conceptos íntimamente relacionados, y mucho más en este proyecto concreto.

2. ESTIMACIONES DEL PROYECTO

2.1. Datos históricos

No se dispone de datos históricos para realizar estimaciones.

2.2. Técnicas de estimación

La técnica de estimación utilizada para realizar las estimaciones de esfuerzo del proyecto ha sido por descomposición del proceso. A partir de la tabla EDT (véase apartado 4.1) se ha estimado el esfuerzo para cada tarea, y posteriormente se ha sumado el esfuerzo de todas las tareas del proceso.

2.3. Estimaciones

En las páginas siguientes se presentan las estimaciones realizadas a partir de la *Estructura de Descomposición del Trabajo*, o tabla *EDT*, presentada en el apartado 4. Como puede observarse, se destina el 47,27% del esfuerzo a tareas de ingeniería, y un 37,95% a la construcción del sistema. El hecho de que se dedique más esfuerzo al análisis y al diseño que a la construcción es debido a la necesidad de adquirir los conocimientos necesarios para trabajar con las tecnologías (J2ME y BlueTooth). Un tiempo bastante menor de construcción se justifica por la experiencia del equipo de desarrollo en proyectos anteriores y el reducido código del sistema a construir. Por último resaltar que el esfuerzo restante (14,78%) se dedica a gestión. El hecho de dedicar poco esfuerzo a la gestión se debe a que partimos de una descripción de los requisitos muy bien definida y estable, al ser el proyecto una versión sin cambios de un sistema ya existente.

Hay que resaltar que algunas de las tareas de *Evaluación por el Cliente* se han considerado *hitos* del proyecto, y por lo tanto su esfuerzo se considera 0 p·d.

Como se calcula en la tabla de estimación, se estiman 660 horas (660 p·d o 660 p·h según lo expuesto en el apartado A.2). Según los datos estimados, cada miembro del equipo trabajará una hora cada día de la semana (los siete días). Esta estimación varía tan sólo un 0,32 % respecto a la estimación dada por *Microsoft Project* (662,13 horas).

Descripción de las iteraciones y tareas de cada iteración

Iteración 1 para PC

Recursos: José María, Sergio y Javier (salvo en la evaluación).

Duración: 10 días.

El proyecto comienza realizando una especificación de requisitos y la planificación del proyecto. Las tareas que se llevan a cabo son:

- **1.1. Comunicación con el cliente. TUE y obtención SRS.** En esta tarea se entrevista al director del proyecto (el cliente) y se obtienen los requisitos de la aplicación. La mayor parte de los requisitos están recogidos en la página web de la versión de Vega Solaris para Spectrum. Con la información de estas entrevistas, los datos de la página web y el juego original se confecciona una especificación de requisitos software estable.
- **1.2. Planificación. Estimación.** En esta tarea se realizan las estimaciones de las características del proyecto (duración, recursos necesarios, tareas a realizar…).
- **1.3. Planificación. Planificación temporal.** División del proyecto en tareas con una duración, fecha de inicio y fin fijas. También se asignan de recursos a esas tareas, se establecen hitos, documentación, código y productos a entregar tras cada tarea. Al final resultará una completa división del esfuerzo necesario para desarrollar el proyecto dentro del tiempo requerido.
- **1.4. Análisis de riesgos. Análisis de riesgos del proyecto.** Se identifican los problemas que pueden aparecer a lo largo del desarrollo de la aplicación. Se les asigna un impacto en el proyecto y una posibilidad de que ocurran. A estos problemas se los identifica como riesgos del proyecto.
- **1.5. Análisis de riesgos**. **Plan RSGR**. A los riesgos determinados en la etapa anterior se les aplica el plan RSGR que consiste en determinar como se va a llevar a cabo la reducción, supervisión y gestión.
- **1.6. Evaluación por el cliente. Evaluación.** El cliente (director del proyecto) evaluará la planificación y la especificación de requisitos software. Comunicará los cambios necesarios en caso de que algo no se ajuste adecuadamente al producto.

Iteración 2 para PC

Recursos: José María, Sergio y Javier (salvo en la evaluación).

Duración: 3 días.

Tras la presentación de la planificación y la especificación de requisitos se lleva a cabo una segunda iteración. En esta segunda iteración se corrigen pequeños problemas e incorrecciones encontrados en alguno de los dos documentos. Se llevan a cabo las siguientes tareas:

- **2.1. Comunicación con el cliente. TUE y obtención SRS.** En esta tarea se revisan las entrevistas con el cliente y los requisitos de la aplicación. Se corrigen las incorrecciones e incoherencias encontradas.
- **2.2. Planificación. Estimación.** En esta tarea se revisan las estimaciones de las características del proyecto (duración, recursos necesarios, tareas a realizar…). Si se encuentran incorrecciones también se corrigen.
- **2.3. Planificación. Planificación temporal.** Se lleva a cabo la revisión de la planificación temporal.
- **2.4. Análisis de riesgos. Análisis de riesgos del proyecto.** Se revisa el análisis de riesgos de la iteración anterior. Se corrigen los errores encontrados.

- **2.5. Análisis de riesgos**. **Plan RSGR**. Se revisa el plan RSGR de la iteración anterior. Se corrigen los errores encontrados.
- **2.6. Evaluación por el cliente. Evaluación.** El cliente evalúa la revisión de la planificación y la especificación de requisitos software revisada. Comunicará los cambios necesarios en caso de que algo en alguno de los dos documentos no se ajuste del todo al producto final buscado. Cuando el cliente dé por válidos ambos documentos, se comenzará con la iteración de análisis general.

Módulo de análisis general

Recursos: José María, Sergio y Javier.

Duración: 7 días.

En esta iteración se desarrollorá la estructura básica y mínima de la aplicación. Esta estructura será la que se irá completando en las fases posteriores. Se llevan a cabo las siguientes etapas:

- **3.1. Ingeniería. Análisis.** En esta tarea se analiza qué es lo que va a ser el esqueleto de la aplicación. Se utilizarán herramientas de Ingeniería del Software Asistida por Computador (CASE) para modelar la estructura básica de la aplicación.
- **3.2. Ingeniería. Diseño.** Se define cómo va a ser el esqueleto de la aplicación. También se usarán herramientas CASE para el desarrollo de esta tarea.

Módulo de obtención de gráficos para PC

Recursos: José María, Sergio y Javier (salvo en la evaluación).

Duración: 2 días.

En este módulo se crearán o copiarán del original los gráficos para el juego. Aunque queda abierta la opción de usar los mismos gráficos en ambas versiones. Se llevan a cabo las siguientes tareas:

- **4.1. Ingeniería. Análisis.** En esta tarea se analiza cuáles van a ser los gráficos de la aplicación. Hay que identificar todos los movimientos posibles de cada uno de los dos personajes, de los enemigos y de los objetos del escenario.
- **4.2. Ingeniería. Diseño.** Tras identificar todas las animaciones se procederá a diseñarlas y a captarlas de la versión original del juego.
- **4.3. Evaluación por el cliente. Evaluación.** El cliente evaluará los gráficos para darlos como válidos para el desarrollo de la aplicación.

Módulo de parametrización de características para PC

Recursos: José María, Sergio y Javier (salvo en la evaluación).

Duración: 2 días.

Se definirán los elementos del juego que se pueden parametrizar (mapa, personajes, enemigos...). Se establecerán las características de cada uno de estos parámetros y sus efectos en el juego para cada uno de los posibles valores que tomen. Las tareas que se llevan a cabo son:

- **5.1. Ingeniería. Análisis.** En esta tarea se analiza cuáles van a ser los parámetros de la aplicación y sus efectos. Hay que identificar todos los movimientos posibles de cada uno de los dos personajes, de los enemigos y de los objetos del escenario.
- **5.2. Ingeniería. Diseño.** Tras identificar todos los parámetros se adaptará la estructura básica (esqueleto) de la aplicación para que se puedan introducir los parámetros en la aplicación.
- **5.3. Evaluación por el cliente. Evaluación.** El cliente evaluará los parámetros y cómo han sido introducidos en el sistema. Si ve correctos los parámetros, se continuará con el desarrollo de la aplicación.

Módulo de desarrollo de las pantallas de presentación para PC

Recursos: José María, Sergio y Javier (salvo en la evaluación).

Duración: 15 días.

Se desarrollará una primera versión del juego en el cual aparezcan las pantallas con el título del juego, pantalla de fin de juego ("Game over"), pantalla de opciones, etc. En resumen, todas las pantallas excepto la pantalla de "interacción" en la que saldría el muñeco del humano/alienígena luchando uno contra otro por conseguir el talismán de Vega Solaris. Se llevan a cabo las siguientes tareas:

- **6.1. Ingeniería. Análisis.** En esta tarea se definirán las características que deben cumplir las pantallas de presentación de la aplicación. Se utilizarán herramientas CASE para plasmar el análisis.
- **6.2. Ingeniería. Diseño.** En esta tarea diseñamos las pantallas de presentación y cómo se pasa de unas a otras. Se utilizarán herramientas CASE.
- **6.3. Construcción y adaptación. Codificación.** En esta tarea se codificarán las clases necesarias para implementar las pantallas de presentación.
- **6.4. Construcción y adaptación. Prueba.** En esta tarea se realizarán pruebas para comprobar el correcto funcionamiento de las pantallas de presentación.
- **6.5. Construcción y adaptación. Ensamblaje.** Se integra el código de las pantallas de presentación en el código del resto de la aplicación.
- **6.6. Evaluación por el cliente. Evaluación.** El cliente evaluará las pantallas de presentación y confirmará si las pantallas se ajustan a los requisitos que comunicó al equipo de desarrollo.

Módulo de gestión de la persistencia para PC

Recursos: José María, Sergio y Javier (salvo en la evaluación).

Duración: 6 días.

Se definirán los datos que se almacenarán en archivos externos y que la aplicación deberá leer y escribir. Se llevan a cabo las siguientes tareas:

- **7.1. Ingeniería. Análisis.** En esta tarea se define qué información debe guardar y leer el juego, desde archivos externos.
- **7.2. Ingeniería. Diseño.** En esta tarea se diseña cómo se van a almacenar los datos persistentes.
- **7.3. Construcción y adaptación. Codificación.** En esta tarea se codificarán las clases necesarias para gestionar la persistencia.
- **7.4. Construcción y adaptación. Prueba.** En esta tarea se realizarán pruebas para comprobar el correcto tratamiento de los datos.
- **7.5. Construcción y adaptación. Ensamblaje.** Se unifica todo el código de la aplicación desarrollado hasta ahora: pantallas de presentación, esqueleto de la aplicación y persistencia.
- **7.6. Evaluación por el cliente. Evaluación.** El cliente evaluará el correcto funcionamiento de la persistencia en la aplicación.

Módulo de desarrollo del juego básico para PC

Recursos: José María, Sergio y Javier (salvo en la evaluación).

Duración: 14 días.

Se creará una segunda versión que además de incluir todas las pantallas de presentación y menús, incluirá una pantalla de "interacción" que muestre el mapa y al personaje principal. Se podrá navegar con el personaje por toda la pantalla, aunque no aparecerán enemigos ni el rival con el que luchar. El tipo de partida corresponderá a una partida monojugador. Esta función se lleva a cabo en la versión para teléfono móvil y la versión para ordenador personal. Se llevan a cabo las siguientes tareas:

- **8.1. Ingeniería. Análisis.** En esta tarea se define qué debe suceder en las pantallas del juego interactivo.
- **8.2. Ingeniería. Diseño.** En esta tarea se diseñan la pantalla de interacción o de juego y cómo ocurre la acción en ellas.
- **8.3. Construcción y adaptación. Codificación.** En esta tarea se codificarán las clases necesarias para implementar las pantallas de interacción.
- **8.4. Construcción y adaptación. Prueba.** En esta tarea se realizarán pruebas para comprobar el correcto funcionamiento de la pantalla de interacción.
- **8.5. Construcción y adaptación. Ensamblaje.** Se une el código implementado hasta ahora.
- **8.6. Evaluación por el cliente. Evaluación.** El cliente evaluará el correcto funcionamiento de la pantalla de interacción.

Módulo de desarrollo del juego completo para PC

Recursos: José María, Sergio y Javier (salvo en la evaluación).

Duración: 11 días.

Se incluyen en esta iteración la aparición de enemigos, el rival alienígena/humano, objetos y armas. Al final de esta iteración estará completa la versión monojugador para PC. Se llevan a cabo las siguientes tareas:

- **9.1. Ingeniería. Análisis.** En esta tarea se define qué deben hacer los monstruos, objetos y el rival del personaje elegido.
- **9.2. Ingeniería. Diseño.** En esta tarea se diseña cómo se llevará a cabo la inserción de los monstruos, objetos y el enemigo.
- **9.3. Construcción y adaptación. Codificación.** En esta tarea se codificarán las clases necesarias para implementar lo anterior.
- **9.4. Construcción y adaptación. Prueba.** En esta tarea se realizarán pruebas para comprobar el correcto funcionamiento del juego completo.
- **9.5. Construcción y adaptación. Ensamblaje.** Se une todo el código desarrollado para construir la versión final para PC.
- **9.6. Evaluación por el cliente. Evaluación.** El cliente evaluará el correcto funcionamiento del juego completo para PC.

Análisis de mejoras

Recursos: José María, Sergio y Javier (salvo en la evaluación).

Duración: 14 días.

Tras acabar la versión final del juego para PC, se realizará esta fase donde se estudiarán los aspectos del juego que se pueden mejorar o actualizar para la versión definitiva en teléfono móvil. Se realizará una lista con los posibles aspectos a mejorar y las posibles alternativas. Los cambios serán recogidos y se incluirán en la versión final para teléfono móvil de Vega Solaris. Durante esta etapa se podrán detectar pequeñas actualizaciones de la especificación de requisitos software y de la planificación para ajustarse a los nuevos esfuerzos obtenidos al aumentar el trabajo a realizar. Todas estas actualizaciones se llevarán a cabo en la siguiente iteración.

- **10.1. Comunicación con el cliente. TUE y obtención SRS.** En esta tarea se realiza una lista con las posibles mejoras. Se revisará la especificación de requisitos software y la planificación ajustándolas, si es necesario, al nuevo esfuerzo estimado.
- **10.2. Evaluación por el cliente. Evaluación.** El cliente evaluará la lista de cambios a introducir para comprobar su corrección.

Iteración 1 para teléfono móvil

Recursos: José María, Sergio y Javier (salvo en la evaluación).

Duración: 6 días.

En esta iteración se realiza lo mismo que en la iteración 2 para ordenador personal. Se parte de una especificación de requisitos y una planificación ya realizadas y se realizarán cambios para corregirlas y que se ajusten a la realidad del desarrollo para teléfono móvil.

Las tareas son las mismas pero enfocadas al desarrollo del juego para el teléfono móvil.

Estudio de tecnologías a utilizar (J2ME)

Recursos: José María, Sergio y Javier.

Duración: 16 días.

En esta fase los integrantes del grupo de desarrollo se documentarán sobre la tecnología J2ME. La tecnología J2ME es la que se utilizará para programar el juego para teléfonos móviles. Despúes de recopilar información, los integrantes del grupo se dedicarán a estudiar y aprender todo lo necesario para utilizar J2ME para programar el juego en teléfonos móviles.

- **12.1. Ingeniería. Análisis.** En esta tarea los miembros del grupo se dedicarán a comprender la tecnología J2ME.

Estudio de tecnologías a utilizar (Bluetooth)

Recursos: José María, Sergio y Javier.

Duración: 28 días.

En esta iteración se realiza lo mismo que en la iteración anterior, pero estudiando en este caso la tecnología Bluetooth. Las tareas que se realizan también son idénticas pero con la tecnología Bluetooth como objetivo a comprender.

Módulo de obtención de gráficos para teléfono móvil

Recursos: José María, Sergio y Javier (salvo en la evaluación).

Duración: 3 días.

Esta iteración es idéntica a la iteración del mismo nombre para ordenador personal. Se realizarán las adaptaciones necesarias para obtener los gráficos para la versión en teléfono móvil, ya sea utilizando los mismos que en la versión para ordenador personal o adaptándolos. Se llevan a cabo las siguientes etapas:

- **14.1. Ingeniería. Análisis.** Similar a la tarea de análisis de la versión para ordenador personal.
- **14.2. Ingeniería. Diseño.** Similar a la tarea de diseño de la versión para ordenador personal.
- **14.3. Evaluación por el cliente. Evaluación.** Similar a la tarea de evaluación de la versión para ordenador personal.

Módulo de parametrización de características para teléfono móvil

Recursos: José María, Sergio y Javier (salvo en la evaluación).

Duración: 3 días.

Esta iteración es idéntica a la iteración del mismo nombre para ordenador personal. Las tareas que se llevan a cabo son:

- **15.1. Ingeniería. Análisis.** Similar a la tarea de análisis de la versión para ordenador personal.
- **15.2. Ingeniería. Diseño.** Similar a la tarea de diseño de la versión para ordenador personal.
- **15.3. Evaluación por el cliente. Evaluación.** Similar a la tarea de evaluación de la versión para ordenador personal.

Módulo de desarrollo de las pantallas de presentación, persistencia y juego básico para teléfono móvil

Recursos: José María, Sergio y Javier (salvo en la evaluación).

Duración: 20 días.

Esta iteración es idéntica a las tres iteraciones del mismo nombre para ordenador personal. Al tener mucha experiencia tras el desarrollo para ordenador personal, realizaremos las tres iteraciones para teléfono móvil en una sola. Se llevan a cabo las siguientes etapas:

- **16.1. Ingeniería. Análisis.** Similar a las tareas de análisis de la versión para ordenador personal.
- **16.2. Ingeniería. Diseño.** Similar a las tareas de diseño de la versión para ordenador personal.
- **16.3. Construcción y adaptación. Codificación.** Similar a las tareas de codificación de la versión para ordenador personal.
- **16.4. Construcción y adaptación. Prueba.** Similar a las tareas de prueba de la versión para ordenador personal.
- **16.5. Construcción y adaptación. Ensamblaje.** Similar a las tareas de ensamblaje de la versión para ordenador personal.
- **16.6. Evaluación por el cliente. Evaluación.** Similar a las tareas de evaluación de la versión para ordenador personal.

Módulo de desarrollo del juego monojugador para teléfono móvil

Recursos: José María, Sergio y Javier (salvo en la evaluación).

Duración: 26 días.

Esta iteración es idéntica a la iteración *Módulo de Desarrollo del Juego Completo para Ordenador Personal*. Se llevan a cabo las siguientes etapas:

- **17.1. Ingeniería. Análisis.** Similar a la tarea de análisis de la versión para ordenador personal.
- **17.2. Ingeniería. Diseño.** Similar a la tarea de diseño de la versión para ordenador personal.
- **17.3. Construcción y adaptación. Codificación.** Similar a la tarea de codificación de la versión para ordenador personal.
- **17.4. Construcción y adaptación. Prueba.** Similar a la tarea de prueba de la versión para ordenador personal.
- **17.5. Construcción y adaptación. Ensamblaje.** Similar a la tarea de ensamblaje de la versión para ordenador personal.
- **17.6. Evaluación por el cliente. Evaluación.** Similar a la tarea de evaluación de la versión para ordenador personal.

Desarrollo y prueba del modo monojugador en teléfono móvil

Recursos: José María, Sergio y Javier (salvo en la evaluación).

Duración: 8 días.

En esta iteración se comprobará que los programas en el emulador y en el teléfono móvil producen los mismos resultados. En caso de no ser así se realizarían las correciones oportunas del código.

- **18.1. Ingeniería. Codificación y Prueba.** En caso de no producirse los mismos efectos en la pantalla del emulador y en la pantalla del teléfono móvil, se codificarán en esta tarea las instrucciones necesarias para que la aplicación se comporte de un modo correcto en el teléfono móvil. Se han unido las tareas de pruebas y codificación porque se realizan ambas a la vez.
- **18.2. Evaluación por el cliente. Evaluación.** El cliente evalúa la aplicación resultante para comprobar que los resultados producidos en la pantalla del emulador y la pantallas de los teléfonos móviles coinciden. La evaluación se llevará a cabo hasta en tres teléfonos móviles de diferentes modelos y diferentes marcas.

Módulo de desarrollo del juego multijugador para teléfono móvil

Recursos: José María, Sergio y Javier (salvo en la evaluación).

Duración: 31 días.

Desarrollo del juego para varios jugadores. La opción multijugador permite el juego entre varios jugadores a través de una conexión Bluetooth entre los teléfonos móviles. Se llevan a cabo las siguientes etapas:

- **19.1. Ingeniería. Análisis.** En esta tarea se define en qué consiste el juego multijugador. Se utilizarán herramientas CASE.

- **19.2. Ingeniería. Diseño.** En esta tarea se diseña cómo será el juego multijugador. Se utilizarán herramientas CASE.
- **19.3. Construcción y adaptación. Codificación.** En esta tarea se codificarán las clases necesarias para implementar el juego multijugador.
- **19.4. Construcción y adaptación. Prueba.** En esta tarea se realizarán pruebas para comprobar el correcto funcionamiento de las partidas del juego multijugador.
- **19.5. Construcción y adaptación. Ensamblaje.** Se une la aplicación desarrollada hasta ahora con el código que implementa el juego multijugador.
- **19.6. Evaluación por el cliente. Evaluación.** El cliente evaluará el correcto funcionamiento del juego completo para teléfono móvil.

Manual de usuario

Recursos: José María, Sergio y Javier (salvo en la evaluación).

Duración: 4 días.

Durante todo el desarrollo de la aplicación se irá rellenando información sobre los mandos y teclas del juego. En esta fase todas esas notas se recopilarán para crear un documento formal que pueda ser usado por el usuario final como manual de instrucciones del juego.

- **20.1. Construcción y adaptación. Codificación.** En esta tarea se recopilan todos los documentos y archivos con información sobre el manual de usuario que se han ido desarrollando a lo largo del proyecto y se generará un archivo en formato PDF y otro en Html con el manual de usuario final.
- **20.2. Evaluación por el cliente. Evaluación.** El cliente evaluará el manual de usuario final para decidir si es válido o no.

Revisión final de la aplicación para teléfono móvil

Recursos: José María, Sergio y Javier (salvo en la evaluación).

Duración: 11 días.

Durante esta iteración se revisarán todos los documentos generados durante el desarrollo de la aplicación para generar una versión final estable, correcta y libre de incoherencias.

- **21.1. Comunicación con el cliente. TUE y obtención SRS.** En esta tarea se revisan las entrevistas con el cliente, los requisitos de la aplicación y los datos en la página web de la versión original. También se revisa la última versión de la especificación de requisitos software.
- **21.2. Planificación. Estimación.** En esta tarea se revisan las estimaciones de las características del proyecto (duración, recursos necesarios, tareas a realizar…). Se recogen los datos de las características del sistema para su uso como datos históricos en otro proyecto.

- **21.3. Planificación. Planificación temporal.** Se revisa la división del proyecto en tareas. También se revisa la asignación de recursos a esas tareas e hitos. Los datos recogidos se utilizarán para generar una información histórica que se pueda usar en otros proyectos.
- **21.4. Análisis de riesgos. Análisis de riesgos del proyecto.** Se revisan los riesgos que se identificaron durante la identificación de riesgos y los riesgos reales aparecidos durante el proceso. Los datos recogidos se utilizarán para generar una información histórica que se pueda usar en otros proyectos.
- **21.5. Análisis de riesgos**. **Plan RSGR**. Se revisa el plan RSGR, cómo se ha aplicado y su utilidad en el proyecto. Los datos recogidos se utilizarán para generar una información histórica que se pueda usar en otros proyectos.
- **21.6. Ingeniería. Análisis.** En esta tarea se revisa toda la documentación relativa al análisis del juego para corregir errores.
- **21.7. Ingeniería. Diseño.** En esta tarea se revisa toda la documentación relativa al diseño del juego para corregir errores.
- **21.8. Construcción y adaptación. Codificación.** En esta tarea se revisa toda la documentación relativa al código generado del juego para corregir errores.
- **21.9. Construcción y adaptación. Prueba.** En esta tarea se revisa toda la documentación relativa a las pruebas del juego para corregir errores.
- **21.10. Construcción y adaptación. Ensamblaje.** En esta tarea se revisa toda la documentación relativa al ensamblaje de las partes del juego para corregir errores.
- **21.11. Evaluación por el cliente. Evaluación.** El cliente evaluará la versión final de la aplicación.

A continuación se presenta la tabla de estimación del esfuerzo.

	Comunicación con el Cliente	Planificación		Análisis de Riesgos		Ingeniería		Construcción y Adaptación			Evaluación por el Cliente	Esfuerzo Total
	TUE y Obtención SRS	Estimación	Planificación Temporal	Análisis Riesgos Proyecto	Plan RSGR	Análisis	Diseño	Codificación	Prueba	Ensamblaje	Evaluación	Esfuerzo Total
proyecto PC iteración 1	1.1 3	1.2 6	1.3 6	1.4 6	1.5 6						**1.6 0**	**27,0**
proyecto PC iteración 2	2.1 1,5	2.2 1,5	2.3 1	2.4 1	2.5 1						**2.6 0**	**6,0**
Módulo Análisis General						3.1 9	3.2 12					**21,0**
Módulo Obtención de Gráficos para PC						4.1 1,5	4.2 1,5				**4.3 0**	**3,0**
Parametrizac. de Características para PC						5.1 1,5	5.2 1,5				**5.3 0**	**3,0**
Desarrollo Pantallas Presentación para PC						6.1 9	6.2 9	6.3 9	6.4 9	6.5 6	**6.6 0**	**42,0**
Gestión de la Persistencia para PC						7.1 3	7.2 3	7.3 3	7.4 3	7.5 3	**7.6 0**	**15,0**
Desarrollo Juego Básico para PC						8.1 6	8.2 6	8.3 15	8.4 6	8.5 6	**8.6 0**	**39,0**

Desarrollo Juego Completo para PC						9.1 6	9.2 6	9.3 12	9.4 3	9.5 3	**9.6** **0**	**30,0**
Estudio Perfecciona-miento Aspectos para versión videojuego Móvil	10.1 39										**10.4** **0**	**39,0**
proyecto iteración 1 para Teléfono Móvil	11.1 3	11.2 3	11.3 3	11.4 3	11.5 3						**11.6** **0**	**15,0**
Estudio Tecnologías a utilizar J2ME						12.1 48						**48,0**
Estudio tecnologías a usar BLUETOOTH en TM						13.1 84						**84,0**
Módulo Obtención de Gráficos para Teléfono Móvil						14.1 3	14.2 3				**14.3** **0**	**6,0**
Parametrizac. de Características para Teléfono Móvil						15.1 3	15.2 3				**15.3** **0**	**6,0**
Desarrollo Pantallas Presentación, Gestión Persistencia y Básico para						16.1 12	16.2 12	16.3 21	16.4 6	16.5 6	**16.6** **0**	**57,0**
Desarrollo Juego Monojugador						17.1	17.2	17.3	17.4	17.5	**17.6**	

Monojugador Completo para Teléfono Móvil						15	18	30	9	6	**0**	**78,0**
Desarrollo y Prueba Juego Monojugador en Teléfono Móvil								18.1 21			**18.2** **0**	**21,0**
Desarrollo Juego Multijugador Bluetooth para Teléfono Móvil						19.1 15	19.2 15	19.3 30	19.4 15	19.5 15	**19.6** **0**	**90,0**
Desarrollo Manual de Jugador								20.1 4,5			**20.2** **0**	**4,5**
Revisión Final de la Aplicación para Teléfono Móvil	21.1 1,5	21.2 1,5	21.3 1,5	21.4 3	21.5 3	21.6 3	21.7 3	21.8 3	21.9 3	21.10 3	**21.11** **0**	**25,5**
Esfuerzo (p·d)	48,0	12,0	11,5	13,0	13,0	219,0	93,0	138,0	64,5	48,0	0,0	660,0
	48,0	23,5		26,0		312,0		250,5			0	660
Porcentaje	7,27%	1,82%	1,74%	1,97%	1,97%	33,18%	14,09%	20,91%	9,77%	7,27%	0%	100,0%
	7,27%	3,56%		3,94%		47,27%		37,95%			0%	100,0%

3. ESTRATEGIA DE GESTIÓN DEL RIESGO

Se seguirá una estrategia de gestión de riesgos proactiva. Procediendo según dicha estrategia, se identifican los posibles riesgos del proyecto y se les asigna una probabilidad y una consecuencia a cada uno, clasificándolos y priorizándolos en función de estos parámetros. A los riesgos más prioritarios (20% del total de riesgos identificados) se les aplicará el plan RSGR.

3.1. Análisis del riesgo

Se han identificado los siguientes riesgos en este proyecto:

R1. Especificación de requisitos cambiante (riesgo de rendimiento). Puede darse el caso de que la especificación de requisitos inicial no esté completa o sea incorrectamente interpretada en algunos puntos, por lo que los requisitos podrían aumentar, disminuir o cambiar. Este riesgo puede provocar que la funcionalidad final del sistema no cumpla con la especificación inicial.

R2. Complejidad (riesgo de rendimiento). El sistema se ha divido en módulos, y la interacción entre ellos es pequeña y las funciones que realizan no son excesivamente complejas, por lo que no hay que poner especial atención en las actividades de ingeniería. Podría suceder, si finalmente se cumple este riesgo, que el rendimiento del sistema no sea el óptimo.

R3. Fecha de finalización inalcanzable (riesgo de planificación). La fecha de finalización no se puede aplazar bajo ningún concepto. Teniendo en cuenta que el equipo de desarrollo realiza otras tareas profesionales ajenas al proyecto, el desarrollo tiene que realizarse en paralelo con dichas tareas, lo que puede provocar que los miembros del equipo no le dediquen al proyecto las horas asignadas. También que los objetivos sean demasiado ambiciosos pueden provocar que el esfuerzo estimado sea considerablemente inferior, y este incremento del esfuerzo no pueda llevarse a cabo.

R4. Desconocimiento de las tecnologías de programación en teléfonos móviles (riesgo de planificación). Aunque parte del equipo ya se ha enfrentado anteriormente a la realización de aplicaciones Java que manejan imágenes, ningún miembro tiene experiencia en el desarrollo de aplicaciones para teléfonos móviles, lo que podría provocar un aumento considerable del esfuerzo estimado.

R5. Personal insuficiente (riesgo de planificación). El equipo de desarrollo consta de tres personas, y no puede ser ampliado ni modificado bajo ningún concepto. Esto supone, por tanto, que si hay retrasos y se corre el riesgo de no cumplir la fecha de entrega no se puede añadir personal para paliar la falta de tiempo, lo que se traduce en la necesidad de un incremento del esfuerzo para poder entregar el proyecto a tiempo.

R6. Retrasos (riesgo de planificación). El riesgo a que existan retrasos se debe a que el equipo de desarrollo, como ya se ha comentado anteriormente, tiene que compatibilizar su trabajo en el proyecto con otras actividades profesionales, y por tanto puede que no se cumpla con la planificación inicial. Este riesgo también se debe a que la estimación inicial podría ser demasiado optimista, y requerir un mayor esfuerzo o tiempo en la realidad. De darse retrasos en el proyecto debería aumentarse el esfuerzo.

R7. Tamaño de la aplicación mayor del esperado (riesgo de planificación). El tamaño esperado de la aplicación podría ser menor que el tamaño real debido a la poca experiencia en

estimación del equipo de desarrollo. Esto podría ocasionar retrasos y aumentos en el esfuerzo invertido en el proyecto.

3.2. Estudio de los riesgos

La siguiente tabla contiene los riesgos identificados, así como la categoría a la que pertenecen, la probabilidad de que ocurran y el impacto que tendrían en el proyecto si se hicieran realidad. Los riesgos están ordenados por importancia, y para los más relevantes (20%) se muestra una entrada en el plan RSGR.

Por otro lado, nótese que la mayoría de los riesgos, así como los más relevantes, son riesgos de planificación. El riesgo a que la planificación temporal prevista no se ajuste al desarrollo real tiene un impacto catastrófico y muchas posibilidades de cumplirse, por tanto es de vital importancia considerar en el plan RSGR todos aquellos riesgos que puedan provocar retrasos en la planificación.

Riesgos	Categoría	Impacto	Probabilidad	RSGR
R6	Planificación	Catastrófico (3)	Frecuente (3)	SÍ
R3	Planificación	Catastrófico (3)	Probable (2)	NO
R4	Rendimiento	Crítico (2)	Probable (2)	NO
R5	Planificación	Crítico (2)	Probable (2)	NO
R1	Rendimiento	Crítico (2)	Improbable (1)	NO
R2	Rendimiento	Crítico (2)	Improbable (1)	NO
R7	Planificación temporal	Marginal (1)	Improbable (1)	NO

3.3. Plan de gestión del riesgo

El riesgo más relevante es el *R6*. El resto han sido ignorados, aplicando la regla de Pareto del 80-20. A continuación se presentan el plan RSGR para el riesgo R6:

Riesgo R6

- **Reducción**
  - Cumplir semanalmente con los objetivos fijados en la planificación.
  - Cumplir puntualmente las tareas de evaluación por el cliente.
  - Ajustar la planificación regularmente, en base al estado del proyecto.
  - Descubrir áreas cuya complejidad sea mayor de la estimada inicialmente en esta planificación para poder rediseñar la planificación y tomar las medidas necesarias para que no se produzcan retrasos.
- **Supervisión**
  - Revisión constante de la planificación.
  - Informes de gestión mediante las vías señaladas en el apartado 6.2 que indiquen a los miembros del equipo de desarrollo el cumplimiento o retraso en las tareas planificadas.
- **Gestión**
  - Aumentar el esfuerzo a 2 horas diarias o más, dependiendo del grado en el que se dé el riesgo.
  - Implementar sólo las funcionalidades básicas de los módulos.

- Reducir la documentación producida, para invertir el esfuerzo en el desarrollo.

4. PLANIFICACIÓN TEMPORAL

En la sección de apéndices se incluyen algunas notas relevantes sobre la planificación temporal (ver apartado A).

4.1. Estructura de descomposición del trabajo (EDT)

La planificación temporal se ha descompuesto en un total de 93 tareas, distribuidas en 21 grupos. La dependencia entre tareas está implícita en la tabla, para ver una especificación explícita es conveniente ver los puntos 4.2 y 4.3. En las siguientes páginas se presenta la tabla EDT, en la que la forma de representar cada tarea concreta es la siguiente:

Código EDT
Fecha de inicio
Fecha de fin
Recursos asignados
Entrega / nombre de la tarea

La tabla ordena las tareas concretas en base a dos parámetros: actividades estructurales divididas en tareas de adaptación (columnas) y módulos funcionales (filas). Para ver una descripción de los módulos que componen el proyecto, consultar el documento **Especificación de Requisitos Software**.

	Comunicación con el Cliente	Planificación		Análisis de Riesgos		Ingeniería		Construcción y Adaptación			Evaluación por el Cliente
	TUE y Obtención SRS	Estimación	Planificación Temporal	Análisis Riesgos Proyecto	Plan RSGR	Análisis	Diseño	Codificación	Prueba	Ensamblaje	Evaluación
proyecto PC iteración 1	1.1 ini: 18-10-05 fin: 18-10-05 r: J, JM, S e: TUE y SRS	1.2 ini: 19-10-05 fin: 20-10-05 r: J, JM, S e: tabla EDT	1.3 ini: 21-10-05 fin: 22-10-05 r: J, JM, S e: planificación temporal	1.4 ini: 23-10-05 fin: 24-10-05 r: J, JM, S e: riesgos del proyecto	1.5 ini: 25-10-05 fin: 26-10-05 r: J, JM, S e: RSGR						**1.6** ini: 27-10-05 **fin: 27-10-05** r: **e: evaluación plan proyecto y SRS**
proyecto PC iteración 2	2.1 ini: 28-10-05 fin: 28-10-05 r: J, JM, S e: revisión TUE y SRS	2.2 ini: 28-10-05 fin: 28-10-05 r: J, JM, S e: revisión tabla EDT	2.3 ini: 29-10-05 fin: 29-10-05 r: J, JM, S e: revisión planificación temporal	2.4 ini: 29-10-05 fin: 29-10-05 r: J, JM, S e: revisión riesgos proyecto	2.5 ini: 29-10-05 fin: 29-10-05 r: J, JM, S e: revisión RSGR						**2.6** ini: 30-10-05 **fin: 30-10-05** r: **e: evaluación plan proyecto y SRS revisado**
Módulo Análisis General						3.1 ini: 31-10-05 fin: 2-11-05 r: J, JM, S e: análisis General	3.2 ini: 3-11-05 fin: 6-11-05 r: J, JM, S e: diseño General				
Módulo Obtención de Gráficos para PC						4.1 ini: 7-11-05 fin: 7-11-05 r: J, JM, S e: análisis de Gráficos	4.2 ini: 7-11-05 fin: 7-11-05 r: J, JM, S e: diseño de Gráficos				**4.3** ini: 8-11-05 **fin: 8-11-05** r: **e: Evaluación de Gráficos**
Parametrizac. de						5.1 ini: 9-11-05 fin: 9-11-05 r: J, JM, S	5.2 ini: 9-11-05 fin: 9-11-05 r: J, JM, S				**5.3** ini: 10-11-05 **fin: 10-11-05** r:

de Características para PC						e: análisis de Parametrizac. de Características	e: diseño de Parametrizac. de Características				**e: evaluación de Parametrizac. de Características**
Desarrollo Pantallas Presentación para PC						6.1 ini: 11-11-05 fin: 13-11-05 r: J, JM, S e: análisis Pantallas de Presentación	6.2 ini: 14-11-05 fin: 16-11-05 r: J, JM, S e: diseño Pantallas de Presentación	6.3 ini: 17-11-05 fin: 19-11-05 r: J, JM, S e: código Pantallas de Presentación	6.4 ini: 20-11-05 fin: 22-11-05 r: J, JM, S e: prueba Pantallas de Presentación	6.5 ini: 23-11-05 fin: 24-11-05 r: J, JM, S e: aplicación Pantallas de Presentación	**6.6** ini: 25-11-05 **fin: 25-11-05** r: **e: evaluación de Pantallas de Presentación**
Gestión de la Persistencia para PC						7.1 ini: 26-11-05 fin: 26-11-05 r: J, JM, S e: análisis persistencia	7.2 ini: 27-11-05 fin: 27-11-05 r: J, JM, S e: diseño persistencia	7.3 ini: 28-11-05 fin: 28-11-05 r: J, JM, S e: código persistencia	7.4 ini: 29-11-05 fin: 29-11-05 r: J, JM, S e: prueba persistencia	7.5 ini: 30-11-05 fin: 30-11-05 r: J, JM, S e: aplicación Pantallas Presentación y Persistencia	**7.6** ini: 1-12-05 **fin: 1-12-05** r: **e: evaluación Pantallas Presentación y Persistencia**
Desarrollo Juego Básico para PC						8.1 ini: 2-12-05 fin: 3-12-05 r: J, JM, S e: análisis Desarrollo Juego Básico	8.2 ini: 4-12-05 fin: 5-12-05 r: J, JM, S e: diseño Desarrollo Juego Básico	8.3 ini: 6-12-05 fin: 10-12-05 r: J, JM, S e: código Desarrollo Juego Básico	8.4 ini: 11-12-05 fin: 12-12-05 r: J, JM, S e: prueba Desarrollo Juego Básico	8.5 ini: 13-12-05 fin: 14-12-05 r: J, JM, S e: aplicación Desarrollo Juego Básico	**8.6** ini: 15-12-05 **fin: 15-12-05** r: **e: evaluación Juego Básico PC**
Desarrollo Juego Completo para PC						9.1 ini: 16-12-05 fin: 17-12-05 r: J, JM, S e: análisis Desarrollo Juego Completo	9.2 ini: 18-12-05 fin: 19-12-05 r: J, JM, S e: diseño Desarrollo Juego Completo	9.3 ini: 20-12-05 fin: 23-12-05 r: J, JM, S e: código Desarrollo Juego Completo	9.4 ini: 24-12-05 fin: 24-12-05 r: J, JM, S e: prueba Desarrollo Juego Completo	9.5 ini: 25-12-05 fin: 25-12-05 r: J, JM, S e: aplicación Desarrollo Juego Completo	**9.6** ini: 26-12-05 **fin: 26-12-06** r: **e: evaluación Juego Completo PC**
Estudio Perfecciona-miento Aspectos	10.1 ini: 27-12-05 fin: 8-1-06 r: J, JM, S										**10.2** ini: 9-1-06 **fin: 9-1-06** r:

miento Aspectos para versión videojuego Móvil	e: estudio efectos en planificación y SRS de las mejoras										**e: evaluación mejoras**
proyecto iteración 1 para Teléfono Móvil	11.1 ini: 10-1-06 fin: 10-1-06 r: J, JM, S e: revisión TUE y SRS versión para TM	11.2 ini: 11-1-06 fin: 11-1-06 r: J, JM, S e: revisión tabla EDT versión para TM	11.3 ini: 12-1-06 fin: 12-1-06 r: J, JM, S e: revisión planificación temporal versión para TM	11.4 ini: 13-1-06 fin: 13-1-06 r: J, JM, S e: revisión riesgos proyecto versión para TM	11.5 ini: 14-1-06 fin: 14-1-06 r: J, JM, S e: revisión RSGR versión para TM						**11.6** ini: 15-1-06 **fin: 15-1-06** r: **e: revisión plan proyecto y SRS para TM**
Estudio Tecnologías a utilizar J2ME						12.1 ini: 16-1-06 fin: 31-1-06 r: J, JM, S e: documentos y notas sobre programación en J2ME					
Estudio tecnologías a usar BLUETOOTH en TM						13.1 ini: 1-2-06 fin: 28-2-06 r: J, JM, S e: documentos y notas sobre programación con BLUETOOTH en TM					
Módulo Obtención de Gráficos para Teléfono Móvil						14.1 ini: 1-3-06 fin: 1-3-06 r: J, JM, S e: análisis Gráficos	14.2 ini: 2-3-06 fin: 2-3-06 r: J, JM, S e: diseño Gráficos				**14.3** **ini: 3-3-06** **fin: 3-3-06** **r:** **e: evaluación Gráficos para TM**
Parametrizac. de						15.1 ini: 4-3-06 fin: 4-3-06	15.2 ini: 5-3-06 fin: 5-3-06				**15.3** **ini: 6-3-06** **fin: 6-3-06**

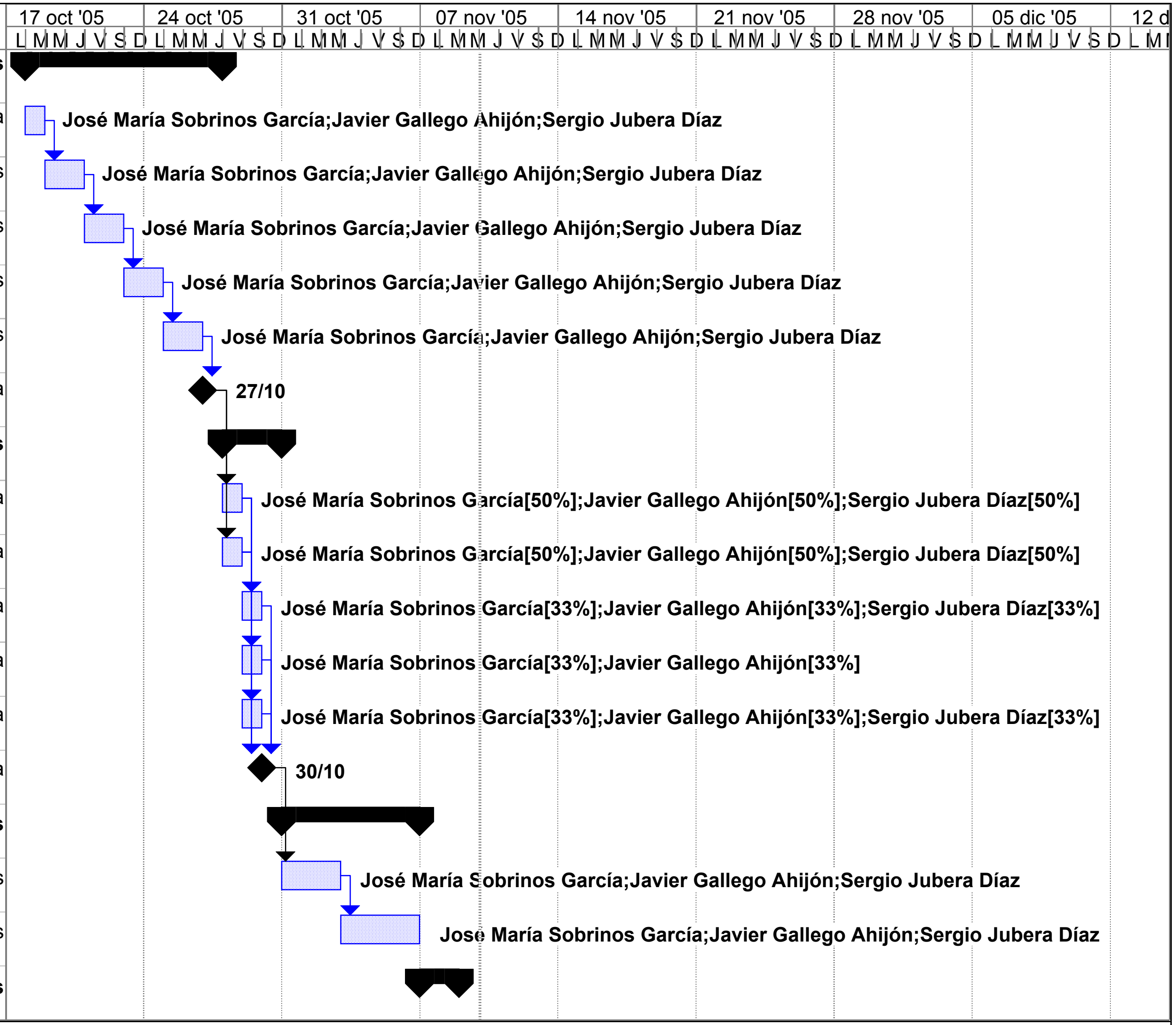
Características para Teléfono Móvil						r: J, JM, S e: análisis Parametrización de Características	r: J, JM, S e: diseño Parametrización de Características				r: **e: evaluación Parametrizac. Caracterist. para TM**
Desarrollo Pantallas Presentación, Gestión Persistencia y Básico para Teléfono Móvil						16.1 ini: 7-3-06 fin: 10-3-06 r: J, JM, S e: análisis de Desarrollo Básico para TM	16.2 ini: 11-3-06 fin: 14-3-06 r: J, JM, S e: diseño de Desarrollo Básico para TM	16.3 ini: 15-3-06 fin: 21-3-06 r: J, JM, S e: código Desarrollo Básico para TM	16.4 ini: 22-3-06 fin: 23-3-06 r: J, JM, S e: prueba Desarrollo Básico para TM	16.5 ini: 24-3-06 fin: 25-3-06 r: J, JM, S e: aplicación Desarrollo Básico para TM	**16.6** **ini: 26-3-06** **fin: 26-3-06** r: **e: evaluación Desarrollo Básico para TM**
Desarrollo Juego Monojugador Completo para Teléfono Móvil						17.1 ini: 27-3-06 fin: 31-3-06 r: J, JM, S e: análisis Juego Monojugador Completo para TM	17.2 ini: 1-4-06 fin: 6-4-06 r: J, JM, S e: diseño Juego Monojugador Completo para TM	17.3 ini: 7-4-06 fin: 16-4-06 r: J, JM, S e: código Juego Monojugador Completo para TM	17.4 ini: 17-4-06 fin: 19-4-06 r: J, JM, S Prueba en emulador Juego Monojugador Completo para TM	17.5 ini: 20-4-06 fin: 21-4-06 r: J, JM, S e: aplicación Juego Monojugador Completo para TM	**17.6** **ini: 22-4-06** **fin: 22-4-06** r: **e: evaluación Juego Monojugador Completo para TM**
Desarrollo y Prueba Juego Monojugador en Teléfono Móvil								18.1 ini: 23-4-06 fin: 29-4-06 r: J, JM, S e: código y prueba en teléfono móvil de Juego Monojugador			**18.2** **ini: 30-4-06** **fin: 30-4-06** r: **e: evaluación Juego Monojugador en Teléfono Móvil**
Desarrollo Juego Multijugador Bluetooth para Teléfono Móvil						19.1 ini: 1-5-06 fin: 5-5-06 r: J, JM, S e: análisis de Juego Multijugador BlueTooth para TM	19.2 ini: 6-5-06 fin: 10-5-06 r: J, JM, S e: diseño de Juego Multijugador BlueTooth para TM	19.3 ini: 11-5-06 fin: 20-5-06 r: J, JM, S e: código de Juego Multijugador BlueToothpara TM	19.4 ini: 21-5-06 fin: 25-5-06 r: J, JM, S e: prueba de Juego Multijugador BlueTooth para TM	19.5 ini: 26-5-06 fin: 30-5-06 r: J, JM, S e: aplicación Juego Multijugador BlueTooth para TM	**19.6** **ini: 31-5-06** **fin: 31-5-06** r: **e: evaluación Juego Multijugador BlueTooth para TM**
								20.1			**20.2**

Desarrollo Manual de Jugador								ini: 1-6-06 fin: 3-6-06 r: J, JM, S e: documentos y archivos Manual del Jugador de Juego Completo para TM			**ini: 4-6-06** **fin: 4-6-06** r: **e: evaluación Manual del Jugador por el profesor**
Revisión Final de la Aplicación para Teléfono Móvil	21.1 ini: 1-6-06 fin: 1-6-06 r: J, JM, S e: revisión final TUE y SRS para TM	21.2 ini: 2-6-06 fin: 2-6-06 r: J, JM, S e: revisión final tabla EDT para TM	21.3 ini: 3-6-06 fin: 3-6-06 r: J, JM, S e: revisión final planificación temporal versión para TM	21.4 ini: 4-6-06 fin: 4-6-06 r: J, JM, S e: revisión final riesgos proyecto versión para TM	21.5 ini: 5-6-06 fin: 5-6-06 r: J, JM, S e: revisión final RSGR versión para TM	21.6 ini: 6-6-06 fin: 6-6-06 r: J, JM, S e: revisión final análisis de Juego Completo para TM	21.7 ini: 7-6-06 fin: 7-6-06 r: J, JM, S e: revisión final diseño de Juego Completo para TM	21.8 ini: 8-6-06 fin: 8-6-06 r: J, JM, S e: revisión final código de Juego Completo para TM	21.9 ini: 9-6-06 fin: 9-6-06 r: J, JM, S e: revisión final prueba de Juego Completo para TM	21.10 ini: 10-6-06 fin: 10-6-06 r: J, JM, S e: revisión final aplicación Juego Completo para TM	**21.11** **ini: 11-6-06** **fin: 11-6-06** r: **e: evaluación final por el profesor**

4.2. Gráfico Gantt

En las páginas siguientes se muestra el gráfico Gantt del proyecto, que es una representación visual de la tabla EDT.

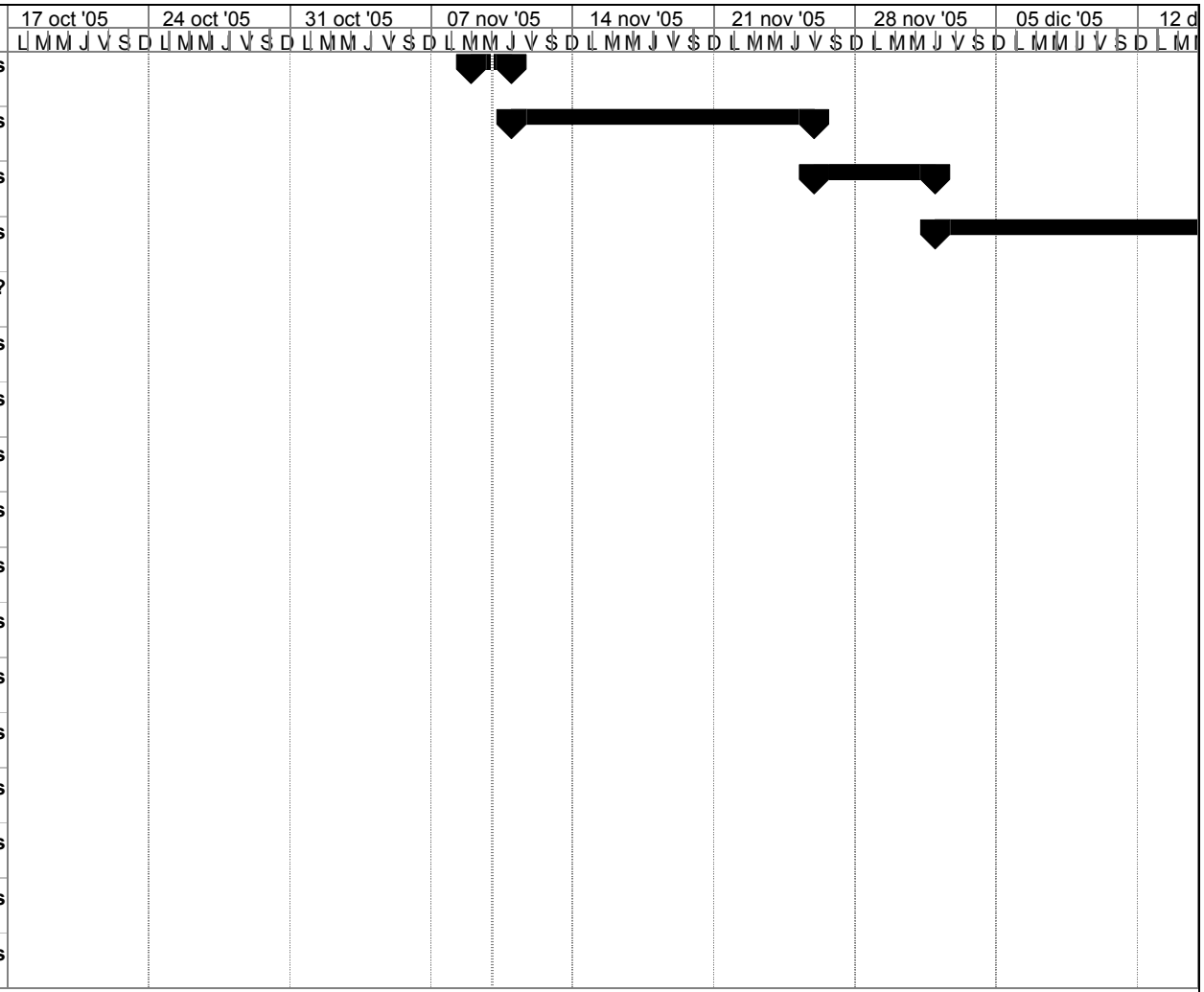
Id		EDT	Nombre de tarea	Duración
1		1	**Proyecto PC iteración 1**	**10 días**
2		1.1	TUE y Obención SRS	1 día
3		1.2	Estimación	2 días
4		1.3	Planificacion Temporal	2 días
5		1.4	Análisis Riesgos Proyecto	2 días
6		1.5	Plan RSGR	2 días
7		1.6	Evaluación	1 día
8		2	**Proyecto PC iteración 2**	**3 días**
9		2.1	Revisión TUE y Obtención	1 día
10		2.2	Revisión Estimación	1 día
11		2.3	Revisión Planificación Ten	1 día
12		2.4	Revisión Análisis Riesgos	1 día
13		2.5	Revisión Plan RSGR	1 día
14		2.6	Revisión Evaluación	1 día
15		3	**Módulo Análisis General**	**42 días**
16		3.1	Análisis General	3 días
17		3.2	Diseño General	4 días
18		4	**Modulo Obtención Gráfico pa**	**9 días**



Proyecto: Vega_Solaris_Project
Fecha: jue 10/11/05

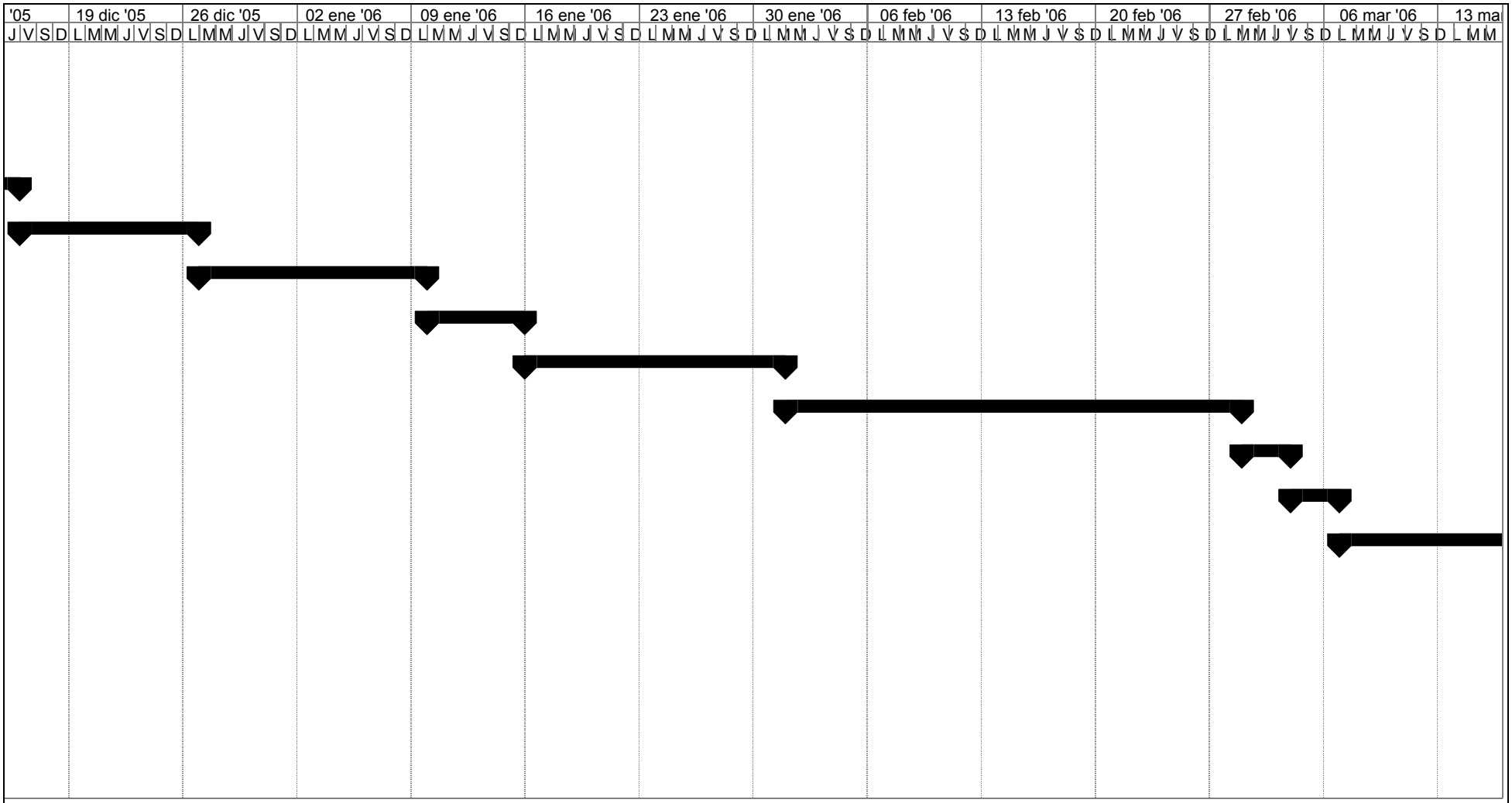
Tarea | Hito | Tareas externas
División | Resumen | Hito externo
Progreso | Resumen del proyecto | Fecha límite

Id	EDT	Nombre de tarea	Duración
22	5	**Parametrización Característic**	**9 días**
26	6	**Desarrollo Pantallas Presenta**	**113 días**
33	7	**Gestión Persistencia PC**	**41 días**
40	8	**Desarrollo Juego Básico PC**	**105 días**
47	9	**Desarrollo Juego Completo P**	**81 días?**
54	10	**Estudio Perfeccionamiento A**	**105 días**
57	11	**Iteración 1 para TM**	**41 días**
64	12	**Estudio Tecnologías a utiliza**	**121 días**
66	13	**Estudio Tecnologías a utiliza**	**217 días**
68	14	**Módulo Obtención de Gráfico**	**17 días**
72	15	**Parametrización Característic**	**17 días**
76	16	**Desarrollo Pantallas Presenta**	**153 días**
83	17	**Desarrollo Juego Monojugad**	**209 días**
90	18	**Desarrollo y Prueba Juego M**	**57 días**
93	19	**Desarrollo Juego Multijugado**	**241 días**
100	20	**Desarrollo Manual de Jugado**	**25 días**
103	21	**Revisión Final de la Aplicació**	**81 días**



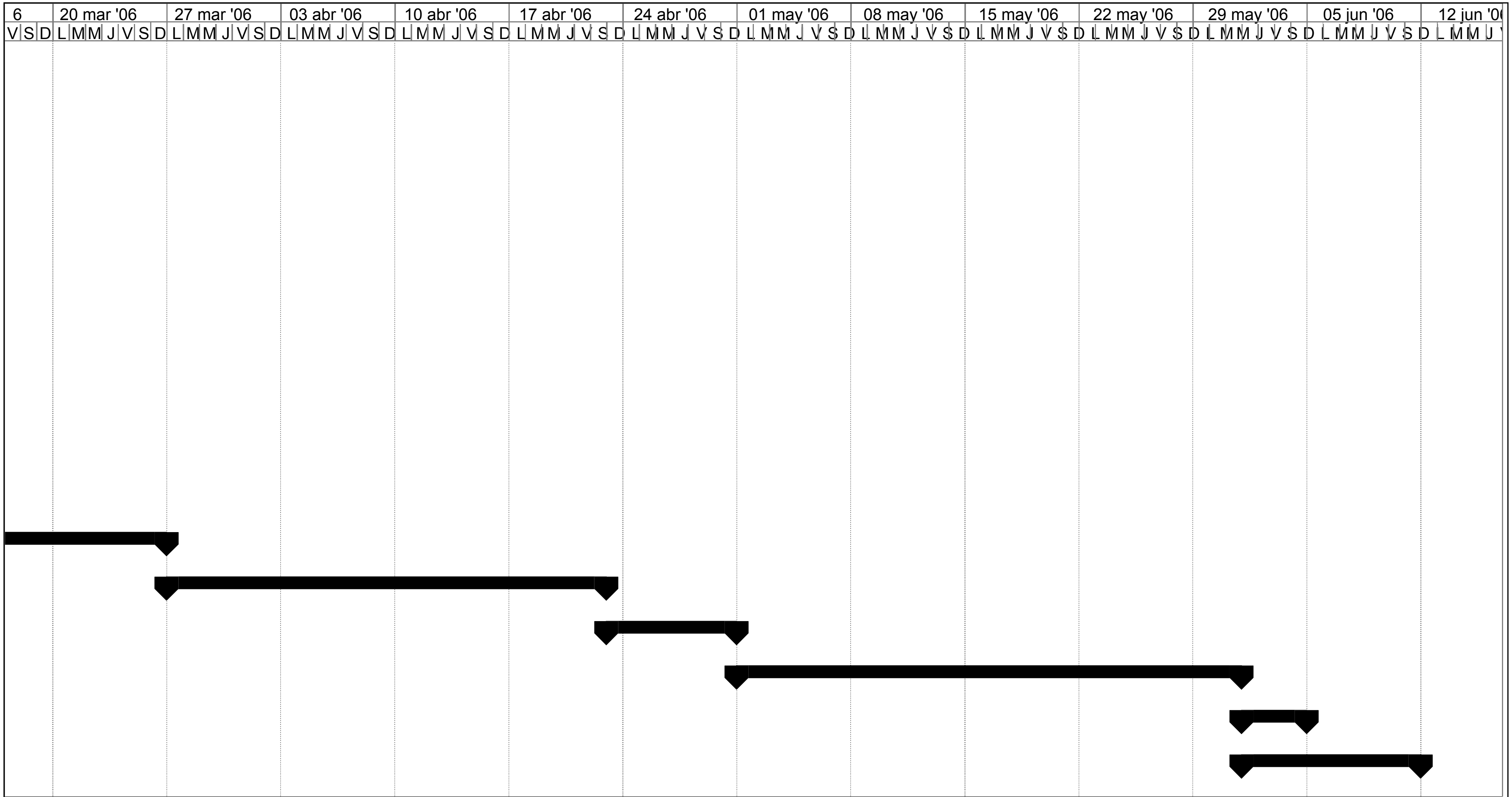
Proyecto: Vega_Solaris_Project
Fecha: jue 10/11/05

Tarea
División
Progreso
Hito
Resumen
Resumen del proyecto
Tareas externas
Hito externo
Fecha límite



Proyecto: Vega_Solaris_Project
Fecha: jue 10/11/05

Tarea		Hito		Tareas externas	
División		Resumen		Hito externo	
Progreso		Resumen del proyecto		Fecha límite	



Proyecto: Vega_Solaris_Project
Fecha: jue 10/11/05

Tarea	Hito	Tareas externas
División	Resumen	Hito externo
Progreso	Resumen del proyecto	Fecha límite

4.3. Red de tareas

En el archivo *Vega_Solaris_Project.mpp* puede verse la red de tareas (diagrama Pert) en la que puede observarse explícitamente la dependencia entre las tareas específicas de la planificación.

4.4. Tabla de uso de recursos

En el archivo *Vega_Solaris_Project.mpp* puede verse la tabla de uso de recursos, que muestra el uso que se hace de los recursos. Para ver una descripción de los recursos, véase el punto 5.

5. RECURSOS DEL PROYECTO

5.1. Personal

Se dispone de un equipo de trabajo formado por 3 personas:

- Javier (J)
- Sergio (S)
- Jose María (JM)

Los integrantes del equipo están familiarizados con la gestión y desarrollo de un proyecto similar y con el manejo de Microsoft Project, Java y Eclipse. También conocen el empleo de UML, Together y Racional Rose en tareas de diseño.

Por otro lado, nunca han trabajado con las herramientas J2ME de desarrollo de aplicaciones para móviles. Por tanto, todos los integrantes necesitarán una toma de contacto previa antes de comenzar con el diseño y la codificación.

Pasamos a describir algunas características del proyecto en relación con el personal:

- **Dificultad**: media-baja. Por la ya comentada experiencia en prácticas anteriores.
- **Tamaño**: medio. El trabajo a realizar no llega a tener la entidad que podría tener una solución de mayor complejidad, pero sí se aproxima.
- **Duración del proyecto**: alta. Aproximadamente 8 meses será la duración.
- **Modularidad**: alta. Se ha llegado a la conclusión de que el proyecto se puede descomponer sin problemas en módulos fácilmente realizables por uno o varios integrantes del grupo.
- **Fecha de entrega**: fija. Marcada por el plan de estudios de Ingeniería Informática de 1.998 de la Universidad Complutense de Madrid. La fecha fijada será en las últimas semanas de Junio.
- **Comunicación**: muy alta. Los integrantes del equipo se verán a diario, se plantearán numerosas reuniones y tendrán la posibilidad de intercambiar información del proyecto dentro de un grupo de trabajo creado en Internet, en la dirección siguiente: http://es.groups.yahoo.com/group/SSII0506/.

5.2. Hardware y Software

El hardware necesario para el desarrollo consiste en: los equipos personales con los que cada integrante del equipo trabajará individualmente; los teléfonos móviles en los que se ejecutará la aplicación; los equipos de los laboratorios de la Facultad de Informática de la Universidad Complutense de Madrid. Por este motivo no se tienen en cuenta en la planificación temporal, pues son suficientes y están siempre disponibles (unos u otros, o todos ellos).

El software necesario será el siguiente:

- *Microsoft Word*, procesador de textos para elaboración de documentos.
- *Acrobat PDF Writer*, generador de documentos en formato PDF para la edición final de los documentos.

- *Microsoft Project* para realizar la planificación temporal del proyecto.
- *IBM Rational Rose* para el diseño del sistema en UML.
- *IBM Eclipse*, herramienta de desarrollo en Java para llevar a cabo la implementación de la aplicación para PC.
- *J2ME* para llevar a cabo la implementación de la aplicación para móvil.
- *Nokia Developer Suite* y otros programas semejantes para la creación de los archivos de la aplicación para el teléfono móvil (archivos `.jar` y `.jad`).

6. ORGANIZACIÓN DEL PERSONAL

6.1. Estructura de equipo

La organización de equipo elegida para este proyecto es la Descentralizada Democrática de Mantei, que se caracteriza por tener bajos niveles de control y jerarquía, pero un alto nivel de creatividad. No hay un jefe fijo, todos los componentes son profesionales y todos aportan ideas, las rebaten, etc. La comunicación es, por tanto, bastante fluida.

Varias son las razones que hacen que ésta sea la opción más adecuada:

- El pequeño número de componentes del equipo y su homogeneidad de conocimientos y experiencia.
- Dentro del equipo no se hará distinción de roles, pudiendo cada integrante llevar a cabo las tareas de analista, de programador o de gestor de proyecto, ya que todos parten con una experiencia y conocimientos similares en el desarrollo de este tipo de proyectos. No habrá un jefe de equipo definido.
- Al no haber un jefe de proyecto, la coordinación se logrará mediante reuniones frecuentes en las que todos tendrán el mismo peso en la toma de decisiones.
- Una comunicación fluida será por tanto de vital importancia. La estructura elegida facilita la comunicación.

6.2. Informes de gestión

Dado que el equipo de trabajo es pequeño y con un alto nivel de comunicación, los informes de gestión se llevarán a cabo por medio de vías informales, aunque no por ello poco importantes: email, llamadas telefónicas, pequeñas reuniones, grupo de trabajo en Internet, etc.

7. MECANISMOS DE SEGUIMIENTO Y CONTROL

7.1. Garantía de calidad y control

No se han definido actividades de la SQA en la tabla EDT, ya que las revisiones se van a realizar inmersas en las actividades de prueba – por parte del equipo de desarrollo – y en las reuniones con el cliente – por parte del cliente. No se harán Revisiones Técnicas Formales, y tampoco se realizará un plan SQA específico ni se aplicarán estándares de calidad.

7.2. Gestión de la configuración software

Identificación de Elementos de Configuración Software

Los *Elementos de Configuración Software* (en adelante ECSs) consistirán en archivos de texto con un nombre identificativo en el que se incluirá su fecha de creación como ECS. Estos archivos de texto contendrán su nombre identificativo, un breve descripción de su contenido y una lista de cambios en sus diferentes variantes o versiones.

En el grupo de Internet del equipo de desarrollo existirá un repositorio en el que guardar todos los ECSs.

Control de Versiones

Existirá un archivo de texto, llamado *control_versiones.txt*, en el que se especificarán explícitamente las versiones válidas de cada uno de los ECS. Este archivo debe estar actualizado en todo momento, y los miembros del equipo de desarrollo lo consultarán frecuentemente para conocer la versión válida de los ECSs con los que trabajen.

Control de Cambios

No existirá una autoridad de control de cambios explícita, la autoridad de control de cambios estará formada por todos los integrantes del equipo de desarrollo. El proceso de control de cambios es el siguiente:

1. Se reconoce la necesidad del cambio.
2. El equipo de desarrollo la evalúa y decide si llevar a cabo el cambio.
3. Identificación de los ECSs implicados en el cambio.
4. Realización del cambio.
5. Revisión de cambio (auditoría).
6. Realización de actividades de garantía de calidad y de prueba.
7. Cambios incluídos en la nueva versión.

Auditoría de la Configuración

Las auditorías de la configuración se llevarán a cabo por parte de los miembros del equipo de desarrollo que participen en el cambio que generó la auditoría, y serán reuniones informales en las que los componentes se preocuparán de si:

- Se han reflejado los cambios en el ECS afectado.
- Se han actualizado convenientemente todos los ECs relacionados.

Informes de Estado

Se crearán y repartirán vía email "informes de estado" que informen sobre:

- En qué consistió el cambio.
- Quién lo realizó.
- Cuándo lo realizó.
- Qué más se vio afectado.

Se generará un "informe de estado" cada vez que se asigne una nueva identificación a un ECS o cada vez que se lleve a cabo una auditoría de configuración. También se generarán "informes de estado" regularmente para mantener informados a los desarrolladores de los cambios importantes.

La generación y distribución de los "informes de estado" correrá a cargo de uno de los responsables de la generación de dicho "informe de estado".

Los informes de estado se archivarán en un repositorio en el grupo de Internet del equipo de desarrollo.

APÉNDICE A. NOTAS SOBRE LA PLANIFICACIÓN TEMPORAL

Fecha de inicio proyecto: 18/10/2005.

Entrega final (implementación): 11/06/2006.

A.1. Breve descripción de las tareas de ensamblaje planificadas

Las tareas de ensamblaje a realizar durante el proyecto consistirán en la adicción de nuevas pantallas o archivos de gráficos al resto de sistema implementado hasta el momento.

Las tareas de ensamblaje planificadas son las siguientes:

1. **23/11/2005**: unión de los gráficos del juego con el módulo de *Pantallas de Presentación de la Aplicación para PC*, en una única aplicación que será la base para el resto de implementaciones de la versión para PC. A esta aplicación se la llamará **Aplicación Base**.
2. **30/11/2005**: se añaden las nuevas funciones obtenidas del módulo *Gestión de la Persistencia* a la Aplicación Base.
3. **14/12/2005**: se añade el *Módulo del Juego Básico* a la Aplicación Base con Persistencia.
4. **25/12/2005**: se reemplaza el *Módulo del Juego Básico* por el *Módulo del Juego Completo* a la aplicación construída hasta ese momento. Se cierra la versión para PC del juego monojugador Vega Solaris.
5. **25/03/2006**: unión de los gráficos del juego, en su versión para teléfonos móviles, con el módulo de *Pantallas de Presentación* de la aplicación para teléfonos móviles y el *Módulo de Gestión de la Persistencia* y el *Módulo del Juego Básico* en una única aplicación que será la base para el resto de implementaciones de la versión para teléfonos móviles. A esta aplicación se la llamará **Juego Básico**.
6. **21/04/2006**: se incorpora al Juego Básico el módulo de *Juego Completo Monojugador*.
7. **30/05/2006**: se incorpora al Juego Completo Monojugador el *Juego Multijugador*. Se cierra la versión para teléfonos móviles del juego monojugador y multijugador Vega Solaris.
8. **10/06/2006**: se corrigen los errores encontrados en la aplicación. Se cierra la versión final, en condiciones de ser comercializada, del juego para teléfonos móviles Vega Solaris.

A.2. Esfuerzo

Se dispone de nueve días para realizar la planificación del proyecto y el documento de especificación de requisitos, desde la segunda reunión con el director del proyecto (18/10/2005) hasta la entrega de la planificación (27/10/2006).

Se dispone de 227 días para la parte de implementación del proyecto (del 28/10/2005 al 11/06/2006).

En total se tienen 236 días.

Para realizar la planificación temporal ha sido necesario tener en cuenta que el grupo de trabajo no dispone de un horario fijo de trabajo para el proyecto a causa de sus actividades docentes y profesionales. Es decir, puede darse el caso de que una tarea que dure una semana, se realice toda durante un sólo día o repartido en varios días.

Se ha buscado abstraerse de este problema estimando que cada integrante dispone de **una hora de trabajo al día** para el proyecto, **incluyendo los 7 días de la semana** (aunque luego no se trabaje en festivos o en otros días). De esta forma, la planificación no resulta tan precisa y será más fácil de seguir.

APÉNDICE B. COSTE DE LA APLICACIÓN

En este punto vamos a mostrar detalladamente el coste estimado de la aplicación. El coste del proyecto se divide en:

1. Coste del personal (impuestos aplicables al personal incluídos).
2. Coste de equipos y licencias.
3. Publicidad.
4. Coste del servicio de mantenimiento y descarga por GPRS.
5. Derechos de autor.
6. Infraestructura.
7. Impuestos.

Coste de personal

Se ha calculado el número de horas que pasan los miembros del equipo de desarrollo realizando funciones de consultor, jefe de proyecto, etc.

Los números que aparecen en la tabla son el número de horas dedicadas por todos los roles de ese tipo, de todos miebros que realizan dicho rol juntos.

Puesto \ Fase	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21
Jefe proyecto	27									15											
Analista			9	1,5	1,5	9	3	6	6	12		16	28	3	3	12	15		15	3	
Consultor (Profesor)	8	8		4	4	8	4	8	8	8	4			4	4	8	8	4	8	4	8
Consultor (Alumnos)		6									15										30
Técnico Sistemas																3	6	3	9		
Diseñador			12	1,5	1,5	9	3	6	6	12		16	28	3	3	12	18		15	3	
Programador						24	9	27	18			16*	28*			30	39	21	51	3**	
Implantador																			3		

*: se reparten los conocimientos adquiridos entre analista, diseñador y programador.
**: el manual de usuario que se ha ido realizando a lo largo del proyecto es terminado y validado por los analistas, diseñadores y programadores.

Suponiendo que los sueldos de la tabla del archivo *Salarios.pdf* son de una jornada de 4 semanas, 5 días laborables a la semana y 8 horas de trabajo por día, se tiene:

Puesto	Coste por Hora* [€ / hora]	Número de Horas	Coste Total
Jefe de Proyecto	16,57	42	696,25 €
Analista Senyor	11,90	133	1583,15 €
Consultor [Profesor]	12,18	112	1364,71 €
Consultor [Alumnos]	12,18	51	621,43 €
Técnico de Sistemas	14,24	21	299,17 €
Diseñador	8,61	139	1196,82 €
Programador Senyor	7,16	266	1904,90 €
Implantador	6,43	3	19,29 €

Programas Producto			
Total			**7685,62 €**

*: incluída Seguridad Social del operario y de la empresa.

Coste de equipos y licencias

Equipo	Precio por Unidad	Número	Coste Total
Ordenadores personales Pentium 4, 3 Ghz, 1.024 MB RAM	549 €	4	2196 €
Teléfonos móviles con Bluetooth (e.g. Nokia 6822)*	332,58 €	4	1330,32 €
Software + Licencia RTU (derecho de uso) Sun Java Studio Standard 5	650 €	4	2600 €
Total			**6126,32 €**

*: ver tabla de referencia de precios de teléfonos móviles con Bluetooth.

Tabla de Precios de Teléfonos Móviles

Precios de algunos teléfonos móviles con tecnología Bluetooth del mercado (año 2005).

Modelo	Precio
Nokia 3230	407,88 €
Nokia 6600	287,40 €
Nokia 6680	608,68 €
Nokia 6822	332,58 €
Nokia 7280	701,55 €
Nokia 7610	626,25 €

A este coste se le añade un 10 % para sufragar desperfectos en el material durante la explotación e incorporar los gastos por amortizaciones de los equipos y teléfonos móviles.

Coste de la aplicación	13811,94 €
Costes amortizaciones	1381,19 €
Coste desarrollo de la aplicación	**15193,13 €**

Este coste es el mínimo para desarrollar la aplicación, pagando sueldos y material. Pero a este coste hay que añadir otros costes adicionales.

Coste de publicidad

El primer coste adicional que se debe añadir al precio es el coste de la publicidad y promoción de la aplicación. Tras un estudio de la cuantía que las empresas de informática dedican a la publicidad, se ha obtenido que las empresas suelen dedicar un 19% del coste de desarrollo del producto a la publicidad y promoción.

Coste de publicidad	**2886,69 €**

A continuación se muestra un análisis del coste de diversas políticas de publicidad. Las diferentes alternativas que existen, en orden decreciente de costes, son:

- **Televisión**: el coste de este tipo de publicidad es el más elevado y alcanza entre uno y diez millones de personas.
- **Radio**: el coste de este tipo de publicidad es muy elevado y alcanza entre uno y dos millones de personas.
- **Prensa**: el coste es muy elevado y alcanza entre cien mil y ochocientas mil personas.
- **Cine**: el coste es alto y dependiendo de la película puede variar entre cien mil personas y cinco millones de personas.
- **Teleoperadores**: el coste de esta publicidad es medio y puede alcanzar de mil a cien mil personas.
- **Publicidad en Internet**: con un coste medio se puede llegar de cien a cien mil personas.
- **Carteles**: esta opción es poco costosa, aunque algo más que los panfletos. Alcanza de cincuenta personas a veinte mil personas.
- **Panfletos**: los panfletos se reparten en sitios concurridos. Es una opción interesante ya que es muy poco costosa y alcanza desde diez a cien mil personas.

1) A continuación se analiza una de las opciones más caras: poner publicidad en la prensa escrita.

La publicidad del producto consiste en anuncios en la prensa durante dos meses tras la finalización del videojuego. El número de publicaciones que incluirán el anuncio del videojuego será de tres.

Se publicarán anuncios con una imagen del videojuego y los datos para poder descargarlo vía GPRS. Como ejemplo y estándar para calcular el coste, hemos elegido un anuncio cuyo tamaño de la imagen será de 75 por 151 milímetros y aparecerá en una sección secundaria, como la sección de la página de televisión.

Para reducir costes, los días laborables se pondrá un recuadro (37 por 37 mm.) y los fines de semana un faldón (75 por 151 mm.).

Hay que incluir los costes del estudio de arte gráfico que desarrolle la imagen del anuncio del videojuego.

Coste diseño anuncio por estudio arte gráfico	**200 €**

Publicidad Periódico	**Número días**	**Coste por día**	Coste total
Laborables Diario Gran Rotativa (e.g. El Mundo*), recuadro	52	1450 €	75400 €
Domingos y festivos Diario Gran Rotativa (e.g. El Mundo*), faldón	8	5540 €	44320 €

*: los costes presentados en este cuadro son los costes reales de poner publicidad en dichas secciones en el periódico El Mundo, a 2 de Noviembre de 2005.

Coste publicidad anuncio en un diario	119720 €

El coste de publicar la publicidad en las otras dos publicaciones de menor importancia se ha estimado en un 50 % del coste de publicar en un diario importante.

Coste publicidad anuncio en las tres publicaciones	239440 €
Coste total de publicidad	**239640 €**

Como puede apreciarse el coste de poner publicidad en la prensa escrita durante dicho periodo de tiempo excede ampliamente el presupuesto destinado a publicidad para la aplicación.

De esta forma se concluye que las opciones de poner publicidad en los medios que alcanzan grandes masas tienen unos costes demasiado elevados en el caso de esta aplicación.

2) Tele-operadores.

En esta opción se presenta el problema de poder llegar a muy poca gente y ser un medio de difusión muy poco usado para la venta de aplicaciones Java para teléfono móvil.

Estimando un coste de 0,15 € por minuto-operador, el capital (2.886 €) alcanza para 19.240 minutos de llamada a teléfonos fijos. Suponiendo 3 minutos con cada cliente y una tasa de cliente no captado del 80% se venderían 1.282 programas.

El coste de las llamadas telefónicas es demasiado elevado y la tasa de ventas demasiada baja. Por tanto estos resultados son insuficientes, dada la inversión de capital realizada. El problema está en el bajo precio del producto que se vende.

3) Carteles.

Esta opción es asequible, aunque algo más costosa que los panfletos. El alcance es bastante elevado comparado con su coste y no se necesitarían elevados gastos de personal para su colocación.

Los precios suelen variar poco y se suele vender un cartel a un precio cercano a 1 € al comprar grupos de cuarenta o sesenta carteles. Cada cartel colocado suele alcanzar a unas cien personas, si está colocado en un lugar concurrido.

Con 2886,69 € se podrían adquirir 2.686 carteles (sin contabilizar los se pueden obtener por volumen de compra) quedando 200 € destinados a pagar al estudio gráfico. Con estos 2.686 carteles, colocándolos durante dos meses, se podría alcanzar a unos 268.600 clientes, lo que supone una cifra muy elevada.

4) Panfletos.

Los panfletos se reparten a domicilio o en sitios muy concurridos, de mano en mano. También se pueden dejar apilados en la puerta de edificios públicos, o repartirlos a la salida del suburbano.

Ésta es una opción interesante, ya que es muy barata y alcanza a bastante gente. El precio de los panfletos es mucho menor que el de los carteles, y por 35 € se pueden imprimir hasta 250 panfletos en muchas imprentas.

Aquí también se invierten 200 euros en un estudio de arte gráfico.

Potencialmente, con 2.686 € se podrían comprar 19.185 panfletos, con los que alcanzar a 21.103 personas (suponiendo que algunos panfletos se dejan en el hogar de los que lo reciben y en un 10% de los casos alguien más accede a ellos).

Conclusión

Tras este análisis se concluye que la solución es una mezcla de panfletos y carteles debido al reducido presupuesto para publicidad.

Se podrían dedicar 200 € al estudio de arte gráfico para que produzcan un diseño para los panfletos y para los carteles. Sería ventajoso obtener varios diseños por la misma cantidad de dinero.

De la cantidad restante (2.686,17 €) se empleará una parte para carteles (1200 €) con los que se tendría 1.200 carteles para colocar en las calles más concurridas de la ciudad, y otra parte (1200 €) se dedicarán a comprar panfletos. Se comprarán 8.751 panfletos que se

repartirán en edificios públicos, centros comerciales, etc. Los 286,17 € restantes se dedicarán a pagar a los individuos que repartan la publicidad y peguen los carteles.

Coste de la aplicación añadiéndole los costes de publicidad	**18079,82 €**

Coste de Mantenimiento del Servicio GPRS, Derechos de Autor e Infraestructura

Al coste anterior hay que añadirle los costes por derecho de autor, servicios de descarga de archivos a teléfono móvil por GPRS y los costes en concepto de infraestructura a la facultad.

Tras estudiar cómo se aplican esos costes en ejemplos de la vida real, se ha obtenido que los costes por derechos de autor son un 7 % del coste calculado hasta ahora. En proyectos como este, en los que no se sabe la cuantía final de las ventas de la aplicación y los autores no saben si cobrar por copia vendida les saldrá rentable, se suele establecer un coste fijo a utilizar su juego como idea base para el desarrollo de una nueva versión.

Por supuesto, siempre pueden negarse y probar suerte obteniendo un 7 % de cada juego vendido (0,21 € por copia sobre el precio de 3 € por juego). Aplicando la tarifa fija ganan algo menos, pero las ganancias son fijas.

En este proyecto, el total que cobrarían los autores por el uso de su idea sería de:

Coste por derechos de autor	**1265,37 €**

Otros costes que también hay que sufragar son los referidos al uso de las infraestructuras de la facultad de Informática de la Universidad Complutense de Madrid. Estos costes suponen un 15 % de los costes de desarrollo del producto (incluída la publicidad).

Coste de infraestructuras	**2711,52 €**

El último aspecto que hay que contabilizar es el coste de mantenimiento de un servicio de descarga de aplicaciones por GPRS. Los costes de este servicio se han estimado en un 3 % del coste del desarrollo de la aplicación más su publicidad.

Coste de servicio de descarga de aplicaciones por GPRS	**542,39 €**

Agrupando los tres costes obtendríamos el coste total de la aplicación tras estos costes adicionales en:

Coste de la aplicación antes de impuestos	**22599,77 €**

Coste de los Impuestos

Sin contabilizar los impuestos de la seguridad social de los trabajadores, que ya están incluidos en el apartado de *Costes del Personal*, se obtiene que los impuestos suelen aumentar el precio de la aplicación en un 16 %, para que la empresa pueda de este modo hacerlos frente. Este 16 % no es el I.V.A., que tiene un tratamiento contable distinto, pero sí incluye los gastos asociados a la diferencia entre el I.V.A. soportado y el I.V.A. repercutido que debe abonar la empresa.

Los costes por impuestos se elevarían a la cuantía de:

Coste de los impuestos	**3841,96 €**

Con lo cual, el coste final de la aplicación sería de:

Coste total de la aplicación	**26441,73 €**

Aquí concluye la estimación del presupuesto detallado de la aplicación.

APÉNDICE C. ESTUDIO DE VIABILIDAD

C.1. Estudio de viabilidad económica de la aplicación

En este apartado presentamos el estudio de viabilidad económica de la aplicación.

Los costes de la aplicación totales suponen una cuantía que asciende a 26441,73 euros.

El precio de nuestro producto está fijado por el mercado en las siguientes cuantías:

Juego para teléfono móvil monojugador	3 euros
Juego para teléfono móvil multijugador	6 euros

El coste del juego que vamos a aplicar será de 6 euros. Aunque cabe la posibilidad de ponerlo a 3 euros para atraer a una mayor cuantía de clientes.

La información relativa a este hecho será recogida en el Anexo 1 tan pronto se reciban datos sobre los ingresos y gastos de empresas reales.

Los costes de pagar todos los gastos menos el personal serían de 18756,11 euros.

Lo primero a pagar es las deudas de los acreedores. Después se pagaría el sueldo del miembro de desarrollo Fernando Sáenz y por último se empezaría a pagar el sueldo de los restantes miembros del equipo y desarrolladores de la aplicación.

- Para pagar a los acreedores deberían venderse 3127 copias del juego.
- Para pagar el sueldo del profesor deberían venderse 321 copias más.

Posteriormente, el resto del dinero que se recaude iría destinado íntegramente a los tres desarrolladores del juego (José María Sobrinos, Javier Gallego, Sergio Díaz). Para pagar sus salarios mínimos deberían venderse 963 copias más.

C.2. Estudio del mercado al que va destinado. Escenario de ventas

El mercado al que debe ir destinado este videojuego es el mercado donde están vendiendo las empresas de venta de videojuegos para teléfonos móviles.

Estas empresas han marcado unas características claras de sus clientes y han determinado dos perfiles de clientes:

1. Hombres adultos entre 30 y 45 años, varones, que disfrutan de los juegos a los que jugaban en su adolescencia (PacMan, Game Over...).
2. Adolescentes entre 12 y 18 años, varones, que disfrutan con juegos en grupo, juegos de acción, simuladores de fútbol o velocidad, aventuras o inteligencia.

Estos son los dos públicos a los que debe ir destinado nuestro producto. Puesto que son los grupos que ocupan mayoritariamente el mercado de juegos para teléfonos móviles.

Vega Solaris tiene las condiciones para atraer a ambos grupos puesto que es un juego de acción que permite jugar en red (para los adolescentes) y está basado en un juego de Spectrum (para los varones adultos).

Vega_Solaris_Project
FDI (UCM)

desde jue 10/11/05

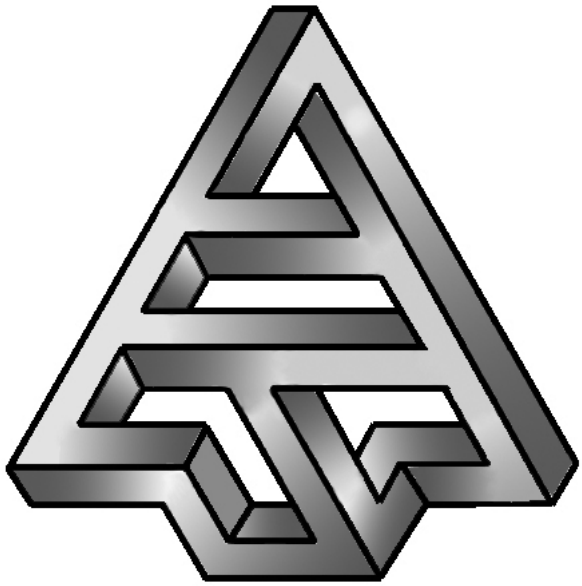
Fechas			
Comienzo:	mar 18/10/05	Fin:	dom 11/06/06
Comienzo previsto:	NA	Fin previsto:	NA
Comienzo real:	NA	Fin real:	NA
Variación de comienzo:	0 días	Variación de fin:	0 días

Duración			
Programada:	1791 días?	Restante:	1791 días?
Prevista:	0 días?	Real:	0 días
Variación:	1791 días?	Porcentaje completado:	0%

Trabajo			
Programado:	662,13 horas	Restante:	662,13 horas
Previsto:	0 horas	Real:	0 horas
Variación:	662,13 horas	Porcentaje completado:	0%

Costos			
Programados:	6.793,56 €	Restantes:	6.793,56 €
Previstos:	0,00 €	Reales:	0,00 €
Variación:	6.793,56 €		

Estado de las tareas		Estado de los recursos	
Tareas aún no comenzadas:	114	Recursos de trabajo:	3
Tareas en curso:	0	Recursos de trabajo sobreasignados:	0
Tareas finalizadas:	0	Recursos materiales:	0
Total de tareas:	114	Total de recursos:	3



Vega Solaris

Especificación de Requisitos Software

Versión 1.0
04/07/2006

ÍNDICE

1. INTRODUCCIÓN

Este es el documento de Especificación de Requisitos Software (ERS o SRS) del videojuego "Vega Solaris" para teléfonos móviles.

Parala creación de este documento de especificación se han seguido las directrices dadas por el estándar "IEEE Recommended Practice for Software Requirements Specification ANSI/IEEE 830 1998".

1.1. Propósito

El objetivo de este documento de especificación es definir de un modo claro y preciso todas las funcionalidades y restricciones del sistema que se va a construir, y va dirigido al equipo de desarrollo y al director del proyecto.

Este documento será el canal de comunicación entre las partes implicadas, tomando parte en su confección miembros de cada parte.

Esta especificación será sometida a varias revisiones, dando lugar a sucesivas versiones del documento hasta alcanzar su aprobación.

Estas revisiones concluirán cuando los requisitos se ajusten a las funcionalidades requeridas por el sistema. Una vez aprobado este documento, servirá de base al equipo de desarrollo para la construcción del nuevo sistema.

1.2. Alcance

Se pretende realizar una versión moderna del juego arcade "Vega Solaris" que apareció en 1989 para Sinclair ZX Spectrum, del que se incluye una ficha técnica en el apéndice del documento.

Esta nueva versión será desarrollada para teléfonos móviles con soporte para juegos Java, y ofrecerá la posibilidad de jugar contra el propio sistema o contra otro jugador mediante Bluetooth.

El objetivo del juego es conseguir el talismán de Vega Solaris. Dicho talismán está formado a su vez por cuatro cristales repartidos en un escenario con diferentes áreas, que incluyen una selva, cuevas y un templo, y que se reparten entre decenas de habitaciones. Estos cuatro cristales se deben llevar hasta el punto de origen en un tiempo limitado. Para ganar el juego es necesario llegar a esta pantalla con los cuatro cristales en los bolsillos de uno de los dos personajes.

Hay dos personajes en la búsqueda de este talismán: un humano y un extraterrestre. Éste es precisamente uno de los alicientes del juego, ya que se puede jugar contra otro jugador o contra la máquina.

Además de representar la parte del mundo en que se mueven, en la parte inferior se encuentra información del contenido de los cuatro bolsillos de los personajes en que pueden guardar y llevar los objetos que encuentren a su paso. El personaje que llegue a la pantalla inicial con los cuatro cristales del talismán de Vega Solaris en sus bolsillos ganará la partida.

Durante el recorrido encontrarán no sólo estos cristales, sino también armas que pueden usar contra el otro personaje y contra los enemigos que aparecen cada vez más insistentemente. Tanto los personajes como los enemigos pueden infligir daños que se

representan con colores en el indicador de estamina del personaje. Estos colores varían de blanco brillante (sin daños) a negro (consumida toda la estamina). Cuando se alcanza el negro, el personaje debe reponerse, quedando inconsciente durante algún tiempo durante el cual el otro personaje puede robarle los objetos que posea.

Obviamente, el juego debe completarse en un límite de tiempo, que se representa por una calavera dentro de una esfera terrestre que va apareciendo lentamente. Cuando aparece totalmente, finaliza la partida.

1.3. Definiciones, Acrónimos y Abreviaturas

Definiciones

- *Bluetooth*: red inalámbrica por radiofrecuencia (2.4 a 2.48Ghz Full Duplex) con un canal de 720Kb/seg y rango óptimo de 10 metros.
- *Contexto software*: instancia del sistema en un teléfono, que está funcionando y que por tanto está siendo utilizada por el usuario y está interactuando con el hardware del dispositivo.
- *Dispositivo*: el teléfono móvil en el que se instala y ejecuta el sistema.
- *Estamina*: es una representación de los daños que ha sufrido el personaje. A mayor estamina, menores daños.
- *Ítem*: objetos que los jugadores pueden recoger durante la partida y que tienen diversas funciones.
- *Jugabilidad*: característica de un juego que refleja el grado en que es agradable jugar al mismo, en base a varios factores: facilidad de manejo, diversión proporcionada, horas de juego (desde la primera vez que el usuario juega hasta la última), calidad de los gráficos y sonidos, etc.
- *Jugador*: usuario del sistema.
- *Mapa*: término usado para referirse al escenario del juego, en el que se desenvuelven los personajes.
- *Modo monojugador*: funcionalidad básica que ofrece el sistema, consistente en permitir al usuario jugar una partida contra el propio sistema.
- *Modo multijugador*: se entenderá por *modo multijugador* la funcionalidad que ofrece el sistema para que dos usuarios jueguen una misma partida mediante la tecnología Bluetooth.
- *Partida*: es el contexto software del sistema desde que el jugador comienza a jugar hasta que consigue el objetivo del juego o da fin al mismo.
- *Sistema*: la aplicación Vega Solaris.

Acrónimos

- *ERS* o *SRS*: Especificación de Requisitos Software (*Software Requirements Specification*).
- *J2ME*: la versión de Java para programación en teléfonos móviles.
- *PC*: Ordenador Personal (*Personal Computer*).

- *VS*: Vega Solaris.
- *[VS89]*: Vega Solaris 1989.

1.4. Referencias

- [SRSA1] SRS Anexo 1 – Ampliaciones al Producto Original, un documento que recoge todas las modificaciones que se han introducido en la aplicación como novedad respecto a [VS89], pero que no se han incluído en el presente documento por no ser requisitos contemplados inicialmente.
- [IE3-830] IEEE Recommended Practice for Software Requirements Specification. ANSI/IEEE std 830,1998.
- [VS89] Vega Solaris, versión de 1989.
- [VS89WEB] Página web de Vega Solaris (1989): http://www.fdi.ucm.es/profesor/fernan/PG/html/vega_solaris.html

1.5. Resumen

Este documento esta dividido en 3 secciones que proporcionan una definición de los requisitos del sistema a distintos niveles de detalle:

1.–Introducción: es la sección en la que nos encontramos y aquí se da una visión general de la ERS.

2.–Descripción General: se proporciona una descripción a alto nivel de detalle de lo que hace el sistema, de sus funcionalidades. Se definen todas las funciones que el sistema debe realizar, los datos asociados y los factores, restricciones, supuestos y dependencias que afectan al desarrollo.

3.–Requisitos Específicos: se definen e identifican todos los requisitos que ha de satisfacer el sistema de forma detallada y exacta. Se indica cuáles son las entradas, las salidas y el proceso necesario para satisfacer cada requisito.

2. DESCRIPCIÓN GENERAL

En esta sección se presenta una descripción de alto nivel del sistema. Se presentarán las principales áreas a las cuales el sistema debe dar soporte, las funciones que el sistema debe realizar, la información utilizada, las restricciones y cualquier otro tipo de factores que afecten al desarrollo del mismo.

2.1. Perspectiva del Producto

El sistema interactuará con el sistema operativo del teléfono móvil en el que se instale, y además interactuará con otros teléfonos móviles en los que el sistema esté también instalado, en el caso del juego multijugador. Esta interacción se realizará mediante Bluetooth.

El sistema no requerirá interactuar con ninguna base de datos externa.

2.2. Funciones del Producto

En términos generales, el sistema debe ser capaz de gestionar los siguientes aspectos relacionados con un videojuego:

- Gráficos.
- Pantallas de Presentación.
- Gestión de la Persistencia.
- Módulo Básico de Juego.
- Módulo Avanzado de Juego.

2.2.1. Gráficos

Este módulo comprende todo el conjunto de gráficos que serán utilizados en el juego. No ofrece ninguna funcionalidad directa al usuario, tan sólo sirve para representar gráficamente los distintos elementos de la aplicación. Todos ellos se tomarán de [VS89], pudiendo ser retocados para adaptarlos a las nuevas capacidades de la tecnología actual. Consistirán en gráficos bidimensionales en formato adecuado para su uso en Java.

2.2.1.1. Mapa

Elementos gráficos que representarán el escenario del juego.

2.2.1.2. Personajes

Elementos gráficos utilizados para los personajes jugadores (humano y alienígena) y para los no jugadores, es decir, los enemigos.

2.2.1.3. Ítems

Elementos gráficos que representarán los distintos elementos que los jugadores pueden encontrar por el escenario y recoger.

2.2.1.4. Marcadores

Elementos gráficos utilizados para los marcadores que representan la vida del personaje, elementos recogidos, tiempo restante, puntuación, etc.

2.2.1.5. Otros

Otros elementos gráficos no contemplados en los puntos anteriores.

2.2.2. Pantallas de Presentación

Este módulo será el encargado de presentar las pantallas de presentación de información al usuario, y responder ante las acciones del mismo.

2.2.2.1. Pantalla de Inicio

- **2.2.2.1.a. Mostrar**: muestra el talismán de Vega Solaris, y las opciones del juego:
    1. Partida contra la máquina – iniciará una partida en modo monojugador.
    2. Partida contra otro jugador – iniciará una partida en modo multijugador.
    3. Ver puntuaciones máximas – mostrará la pantalla de puntuaciones, que contiene el registro de las mejores puntuaciones conseguidas en el teléfono.
    4. Ver créditos – mostrará la pantalla de créditos, que contiene información sobre los creadores del juego.
    5. Ver ayuda – mostrará la pantalla de ayuda, que contiene una pequeña explicación del juego y su manejo.

  Además debe permitir la navegación entre las opciones y ejecutar las acciones asociadas a cada una de ellas.

2.2.2.2. Pantalla de Fin

- **2.2.2.2.a. Mostrar**: muestra un mensaje de fin de la partida, la puntuación de ambos jugadores (tanto en modo monousuario como en modo multiusuario) e informa al jugador de si ha ganado o ha perdido.

2.2.2.3. Pantalla de Puntuaciones

- **2.2.2.3.a. Mostrar**: muestra el registro de las mejores puntuaciones obtenidoas en el teléfono.

2.2.2.4. Pantalla de Créditos

- **2.2.2.4.a. Mostrar**: muestra una breve información sobre los creadores del juego.

2.2.2.5. Pantalla de Ayuda

- **2.2.2.5.a. Mostrar**: explica muy brevemente el funcionamiento del juego y el control del personaje.

2.2.3. Gestión de Persistencia

Este módulo será el encargado de guardar y recuperar datos que la aplicación necesite almacenar en el teléfono. Esta información será: puntuaciones máximas, información textual que se muestre durante el juego[1] (créditos, ayuda y mensajes) e información estructural sobre el mapa y los personajes[2].

- **2.2.3.a. Guardar datos**: esta función almacenará la información en un formato adecuado.
- **2.2.3.b. Recuperar datos**: esta función recuperará la información guardada a través de la función anterior.

2.2.4. Módulo Básico de Juego

Ofrece una funcionalidad muy básica respecto al producto final. Tan sólo representa el escenario y a los personajes jugadores dentro de él.

2.2.4.1. Representación del Mapa

- **2.2.4.1.a. Cargar mapa**: crea el mapa en memoria.
- **2.2.4.1.b. Cargar habitación**: carga en memoria una habitación concreta.

2.2.4.2. Personajes Jugadores

- **2.2.4.2.a. Mover**: mueve al personaje en una de las cuatro direcciones: arriba, abajo, izquierda y derecha.
- **2.2.4.2.b. Selección de acción**: selecciona la acción que realizará el personaje (selección de bolsillo, acción, coger/robar o dejar).
- **2.2.4.2.c. Selección de bolsillo**: selecciona el bolsillo activo del personaje. Si está pegados al personaje contrario inconsciente, selecciona su bolsillo (esto es útil para robarle).
- **2.2.4.2.d. Acción**: hace que el personaje use el objeto que tenga en el bolsillo seleccionado. La consecuencia puede ser lanzar una flecha, un hacha, comer, activar un conjuro, etc., dependiendo del contenido del bolsillo. Si el bolsillo está vacío, el personaje golpea con un puñetazo.
- **2.2.4.2.e. Coger/Robar**: se cogerá el objeto del suelo, o del contrario si está pegado al personaje. Si el bolsillo ya contenía un objeto, se intercambian. En general, para robar hay que: 1) alejarse del contrario para seleccionar el bolsillo propio donde queremos alojar el objeto robado, 2) pegarse a él para seleccionar el bolsillo ajeno y 3) coger el objeto.

[1] Almacenar esta información en archivos de texto facilita el mantenimiento y reutilización de la aplicación (e.g. traducción a otros idiomas).

[2] Para facilitar la implementación del requisito RS0 (ver punto 3.6) se almacenará información sobre la representación interna de cada elemento del juego (e.g. representación de un mapa concreto).

- **2.2.4.2.f. Dejar**: hará que el personaje deje el objeto en el suelo al lado del personaje (siempre que haya espacio para ello).
- **2.2.4.2.g. Suma puntos**: añade puntos al jugador por haber eliminado a un enemigo o por haber herido al personaje contrario.

2.2.4.3. Marcadores

Representa los marcadores de los jugadores, que muestran información sobre las acciones, los ítems, las puntuaciones y el tiempo restante.

La zona de los marcadores, en la parte inferior de la pantalla, contiene tres secciones principales: la izquierda está dedicada a la información sobre el personaje terrestre, la central a la indicación del tiempo transcurrido y la derecha a la información sobre el personaje extraterrestre.

Sección de Información del personaje

Estamina

En el centro de la sección aparece el indicador de estamina, una forma ovalada que cambia de blanco (máxima estamina) a negro (sin estamina) pasando por dieciséis colores. El personaje queda inconsciente cuando se agota la estamina y durante su inconsciencia la va recuperando poco a poco. Otra forma de recuperar estamina es comer.

Bolsillos

Cada uno de los cuatro bolsillos se muestran en el extremo derecho de la sección de información del terrestre y en el izquierdo en el caso del extraterrestre. Sólo uno de los bolsillos está activo (se indica con fondo de otro color).

Iconos

En cada sección de información de los personajes aparecen cuatro iconos que permiten usar, recoger, dejar y robar objetos. Sólo uno de ellos está activo en cada momento (se indica con fondo de otro color). Al pulsar la tecla **Selector**, se selecciona el siguiente icono (de izquierda a derecha y de arriba abajo).

Puntuación

En la parte inferior de cada panel se muestra la puntuación del personaje, que aumenta según se abaten enemigos o se hace disminuir la estamina del personaje contrario.

Sección de indicación de tiempo

Entre las secciones de información de los personajes hay un espacio para mostrar una imagen que se va completando conforme avanza el tiempo. La partida termina cuando se completa la imagen o cuando se alcanza el objetivo del juego.

- **2.2.4.3.a. Activar acción**: cambia la acción del personaje indicado (seleccionar bolsillo, coger, tirar o usar) a otra acción.
- **2.2.4.3.b. Activar bolsillo**: cambia el bolsillo activo del personaje indicado.
- **2.2.4.3.c. Actualizar reloj**: actualiza la imagen que muestra el tiempo restante acorde con dicho tiempo.
- **2.2.4.3.d. Actualizar puntuación**: actualiza la puntuación del personaje indicado.

2.2.5. Módulo Avanzado de Juego

Amplía el módulo anterior para dotar al producto de la funcionalidad total. Representa a los personajes no jugadores y los ítems, e implementa la lógica del juego y de la aplicación.

2.2.5.1. Personajes No Jugadores

Representa a los personajes no jugadores y calcula las colisiones de los mismos con los elementos del mapa y con los personajes jugadores. Además, implementará algún algoritmo básico de inteligencia artificial.

- **2.2.5.1.a. Mover**: mueve al enemigo. Esta función debe estar implementada mediante algún algoritmo básico de inteligencia artificial.
- **2.2.5.1.b. Atacar**: hará que el enemigo ataque a un personaje jugador. Dependiendo del enemigo, este ataque será cuerpo a cuerpo (pegado al personaje) o a distancia.

2.2.5.2. Ítems

Representa los ítems que se pueden recoger a lo largo del mapa, y calcula las colisiones con los personajes jugadores.

En total se pueden encontrar los siguientes objetos:

- Cristales del talismán de Vega Solaris: hay cuatro diferentes que conforman el símbolo completo.
- Armas:
    1. Espadas: proporciona un ataque cuerpo a cuerpo poderoso.
    2. Arcos: lanzan flechas a personajes y enemigos. Cuando el número de flechas llega a cero, el objeto es destruido.
    3. Hachas: se lanzan a personajes y enemigos. Es un ataque a distancia más poderoso que el arco.
    4. Puñetazos: no es un objeto como tal, es el modo de ataque cuando no se utilizan armas.
- Escudos: proporcionan inmunidad por un tiempo determinado hasta que se agotan. El personaje que lo usa cambia su color intermitentemente a amarillo.
- Conjuros (desaparecen una vez usados):

1. Básico: una bola de luz que se lanza y disminuye la estamina de la víctima.
2. Noche: oculta la visión del otro personaje, de manera que sólo puede ver su personaje en la pantalla, y todo lo demás en la oscuridad. Mapa, enemigos y otros objetos no serán visibles durante un tiempo limitado.
3. Teletransporte: permite acudir inmediatamente a la ubicación del otro personaje.
4. Temporal: detiene el tiempo salvo para quien lo activa y hace desaparecer a los enemigos. El otro personaje queda paralizado hasta que terminen los efectos del hechizo.

- Comida: permite recuperar la estamina perdida. La hay de varios tipos, unos recuperan más que otros.

Todos los diferentes tipos de ítems tienen una misma función:

- **2.2.5.2.a. Ejecutar**: el ítem ejecuta su acción, que variará según el tipo de objeto.

2.2.5.3. Lógica del Juego

Implementa la lógica del programa, así como la de las reglas del juego. También implementa los marcadores.

- **2.2.5.3.a. Iniciar partida individual**: inicia una partida monousuario.
- **2.2.5.3.b. Iniciar partida doble**: inicia una partida multiusuario.
- **2.2.5.3.c. Terminar partida**: da por finalizada una partida, mostrando las puntuaciones y los ganadores, si un jugador ha ganado o se ha acabado el tiempo, o volviendo directamente a la pantalla de inicio si la partida se ha cancelado.
- **2.2.5.3.d. Cargar datos iniciales**: carga todos los gráficos y crea todos los objetos de la aplicación.
- **2.2.5.3.e. Mostrar mensaje**: muestra un mensaje por pantalla.

2.3. Características del Usuario

Los usuarios del producto serán todos aquellos que posean un teléfono móvil con soporte para juegos Java y Bluetooth. En la actualidad este tipo de usuario engloba a personas de muy diversas condiciones sociales y formativas. Por ello supondremos un estereotipo de usuario que tan sólo tenga conocimientos básicos en el manejo de un teléfono móvil de las características mencionadas.

2.4. Restricciones

Esta sección describirá aquellas limitaciones que se imponen sobre los desarrolladores del producto.

Restricciones de Fiabilidad

La única restricción en cuanto a fiabilidad es la de que el sistema no tenga errores y, por tanto, funcione correctamente. Es especialmente importante en el caso del modo multijugador, debiendo garantizarse la conexión durante toda la partida, dentro de las restricciones de la tecnología Bluetooth.

Consideraciones Acerca de la Seguridad

En el modo multijugador se enviarán los datos mínimos necesarios para gestionar la partida, y en ningún caso se enviarán datos que comprometan la intimidad del usuario.

Lenguaje de Programación

El sistema Vega Solaris se implementará usando el lenguaje Java, con las características concretas de la programación en teléfonos móviles.

Otras Restricciones

El sistema deberá ofrecer una respuesta rápida y su manejo debe ser sencillo, lo suficiente como para ofrecer al usuario una alta jugabilidad.

2.5. Supuestos y Dependencias

Se supone que todos los requisitos que han sido solicitados serán estables una vez aprobados por el director del proyecto aunque, por supuesto, pueden modificarse a lo largo del desarrollo, siempre y cuando sea aprobado por todas las partes y sea en fases tempranas de desarrollo.

El sistema Vega Solaris se comunicará con otros sistemas Vega Solaris instalados en otros teléfonos móviles, por medio de la tecnología Bluetooth.

2.6. Requisitos Futuros

En un futuro podría ser interesante añadir al sistema la opción de ofrecer varios escenarios (y no sólo uno, como está especificado en este documento) y nuevos enemigos. La idea sería que el usuario pudiera actualizar el sistema con nuevos enemigos, escenarios, armas, etc… Esta actualización podría realizarse, por ejemplo, mediante descarga de archivos por Internet o por WAP.

Para facilitar la creación de nuevos escenarios y enemigos, podría implementarse también un editor de mapas y personajes.

3. REQUISITOS ESPECÍFICOS

En este apartado se presentan los requisitos funcionales que deberán ser cumplidos por el sistema. Todos los requisitos aquí expuestos son **esenciales**, es decir, no sería aceptable un sistema que no satisfaga alguno de los requisitos aquí presentados. Estos requisitos se han especificado teniendo en cuenta, entre otros, el criterio de *estabilidad*: dado un requisito, deberá de ser fácilmente demostrable si es satisfecho o no por el sistema.

Todos los requisitos están codificados siguiendo la siguiente notación: una letra R que indica que se trata de un requisito; después una letra que indica de qué tipo es el requisito –I para Interfaces, F para funciones, R para rendimiento, B para base de datos lógica, D para diseño y A para atributos del sistema; por último se utiliza un código alfanumérico que distingue entre requisitos del mismo tipo. En el caso de las funciones, este código hace referencia a la sección con el mismo código del apartado 2 del presente documento, para facilitar su consulta.

3.1. Interfaces Externas

Interfaces de Usuario

- **RIU0.** La interfaz de usuario ha de ser simple, es decir, sólo presentará la imagen del juego. No tendrá botones, menús, barras de desplazamiento, etc. Esto es debido a que el juego se ejecutará en móviles, donde la pantalla es pequeña y hay que maximizar el uso de la misma para gráficos.
- **RIU1.** Para la interacción con el usuario el sistema responderá a la pulsación de determinadas teclas, que serán las usuales en los juegos para móviles.

Interfaces Hardware

No han sido definidas interfaces hardware.

Interfaces Software

No han sido definidas interfaces software.

Interfaces de Comunicación

- **RIC0.** El sistema interactuará con otro idéntico instalado en otro teléfono móvil, por lo que debe tener soporte para comunicarse con la red local (formada por los dispositivos Bluetooth dentro del radio de alcance).

3.2. Funciones

3.2.1. Pantallas de Presentación

3.2.1.1. Pantalla de Inicio

- **RF2221A. Mostrar**

Descripción de la función: muestra en la pantalla el menú de presentación o de inicio del juego, que ofrece varias opciones: iniciar partida para un jugador (contra la máquina),

iniciar partida contra otro jugador (por Bluetooth), ver puntuaciones máximas, ver los créditos del juego y ver la ayuda. En esta pantalla se mostrará también el talismán de Vega Solaris, en tamaño grande.

Descripción de entrada: ninguna.

Origen de la entrada: ninguno.

Descripción de la salida: pantalla de inicio.

Destino de la salida: terminal.

Requisito: repositorio de imágenes.

Precondiciones: no hay ninguna partida en funcionamiento.

Poscondiciones: el usuario puede ver la pantalla y elegir una opción.

3.2.1.2. Pantalla de Fin

- **RF2222A. Mostrar**

Descripción de la función: muestra la pantalla de fin de partida, que contiene las puntuaciones de ambos jugadores (tanto en modo monousuario como multiusuario), el tiempo transcurrido y el restante e informa al jugador de si ha ganado o ha perdido.

Descripción de entrada: puntuaciones de ambos jugadores, tiempo restante, ganador de la partida.

Origen de la entrada: aplicación.

Descripción de la salida: pantalla de fin de partida.

Destino de la salida: terminal.

Requisito: ninguno.

Precondiciones: el tiempo de juego se ha agotado o uno de los jugadores ha conseguido el objetivo.

Poscondiciones: partida finalizada.

3.2.1.3. Pantalla de Puntuaciones

- **RF2223A. Mostrar**

Descripción de la función: muestra por pantalla el registro de las mejores puntuaciones logradas en el terminal.

Descripción de entrada: ninguna.

Origen de la entrada: ninguno.

Descripción de la salida: pantalla de puntuaciones.

Destino de la salida: terminal.

Requisito: registro de mejores puntuaciones.

Precondiciones: no hay ninguna partida en funcionamiento.

Poscondiciones: ninguna.

3.2.1.4. Pantalla de Créditos

- **RF2224A. Mostrar**

Descripción de la función: muestra la pantalla de créditos, en la que figura diversa información sobre las personas que han participado de alguna u otra forma en el proyecto, y otra información relevante.

Descripción de entrada: ninguna.

Origen de la entrada: ninguno.

Descripción de la salida: pantalla de créditos.

Destino de la salida: terminal.

Requisito: archivo que contenga los datos que se mostrarán, incluyendo información de formato.

Precondiciones: no hay ninguna partida en funcionamiento.

Poscondiciones: ninguna.

3.2.1.5. Pantalla de Ayuda

- **RF2225A. Mostrar**

Descripción de la función: muestra la pantalla de ayuda, que contiene una brevísima descripción del juego y de su manejo.

Descripción de entrada: ninguna.

Origen de la entrada: ninguno.

Descripción de la salida: pantalla de ayuda.

Destino de la salida: terminal.

Requisito: archivo que contenga la información que se mostrará.

Precondiciones: ninguna.

Poscondiciones: partida en pausa, si había una partida en funcionamiento.

3.2.2. Gestión de la Persistencia

- **RF223A. Guardar Datos**

Descripción de la función: guarda una determinada información en un archivo. Esta información corresponde a las puntuaciones máximas de los jugadores.

Descripción de entrada: puntuaciones máximas que se deben guardar.

Origen de la entrada: sistema.

Descripción de la salida: un mensaje indicando si el proceso ha sido realizado correctamente.

Destino de la salida: terminal.

Requisito: registro de puntuaciones máximas.

Precondiciones: las puntuaciones máximas proporcionadas están actualizas. En ningún caso esta función realizará dicha actualización, simplemente eliminará el registro antiguo y creará uno nuevo con la información proporcionada.

Poscondiciones: registro de puntuaciones máximas actualizado.

- **RF223A. Recuperar Datos**

Descripción de la función: recupera datos de diversa índole a partir de archivos almacenados en el terminal. Los tipos de datos que debe recuperar son:

1. Registro de puntuaciones máximas.
2. Información textual mostrada durante el juego: créditos, ayuda y mensajes a los jugadores.
3. Información estructural: representación interna del mapa y de los personajes (jugadores y no jugadores).

Descripción de entrada: nombre del archivo que contiene los datos.

Origen de la entrada: sistema.

Descripción de la salida: un objeto con la información recuperada.

Destino de la salida: sistema.

Requisito: registro de puntuaciones máximas, archivos de texto con los textos que se mostrarán en pantalla, archivos con la información estructural del mapa y de los personajes.

Precondiciones: el archivo consultado tiene un formato adecuado.

Poscondiciones: el sistema tiene un objeto con la información del archivo.

3.2.3. Módulo Básico de Juego

3.2.3.1. Representación del Mapa

- **RF2241A. Cargar Mapa**

Descripción de la función: crea una representación interna del escenario del juego a partir de los archivos que contienen la información para el mapa (gráficos y estructura).

Descripción de entrada: archivos del mapa.

Origen de la entrada: sistema.

Descripción de la salida: indicación de si la carga del mapa ha tenido éxito o no.

Destino de la salida: sistema.

Requisito: archivos que definen los gráficos y estructura del mapa.

Precondiciones: no hay un mapa cargado en memoria.

Poscondiciones: mapa cargado en memoria.

- **RF2241B. Cargar Habitación**

Descripción de la función: crea una representación interna de una habitación concreta del mapa, la que se le indique, con el objetivo de poder representarla gráficamente en pantalla.

Descripción de entrada: identificador de la habitación que se quiere cargar.

Origen de la entrada: sistema.

Descripción de la salida: indicación de si la operación se ha podido realizar con éxito o no.

Destino de la salida: sistema.

Requisito: habitaciones del mapa identificadas unívocamente siguiendo algún tipo de criterio.

Precondiciones: mapa cargado en memoria.

Poscondiciones: habitación cargada en memoria y lista para ser representada gráficamente.

3.2.3.2. Personajes Jugadores

- **RF2242A. Mover**

  Descripción de la función: mueve el personaje en la dirección indicada.

  Descripción de entrada: dirección de movimiento.

  Origen de la entrada: usuario.

  Descripción de la salida: nueva posición del personaje.

  Destino de la salida: pantalla.

  Requisito: ninguno.

  Precondiciones: la partida está activa y el jugador puede moverse en la dirección indicada.

  Poscondiciones: el personaje se encuentra en la nueva posición.

- **RF2242B. Selección de Acción**

  Descripción de la función: cambia la acción activa del personaje.

  Descripción de entrada: ninguna (esta función selecciona automáticamente la siguiente acción activa en función de la actual, según el siguiente orden cíclico: selección de bolsillo, acción, coger/robar y dejar).

  Origen de la entrada: ninguno.

  Descripción de la salida: ninguna.

  Destino de la salida: ninguna.

  Requisito: ninguno.

  Precondiciones: partida activa.

  Poscondiciones: el personaje tiene activa la siguiente acción.

- **RF2242C. Selección de Bolsillo**

  Descripción de la función: cambia el bolsillo activo del personaje, o el del personaje contrario.

Descripción de entrada: ninguna (esta función selecciona automáticamente el siguiente bolsillo activo en función del actual, según el siguiente orden cíclico: bolsillo1, bolsillo2, bolsillo3, bolsillo4).

Origen de la entrada: ninguno.

Descripción de la salida: ninguna.

Destino de la salida: ninguno.

Requisito: ninguno.

Precondiciones: la acción activa del personaje es la de *selección de bolsillo.*

Poscondiciones: el personaje contrario tiene activo el siguiente bolsillo, si se está adyacente a él, o es el propio personaje el que tiene activo el siguiente bolsillo, en caso contrario.

NOTA: para cambiar el bolsillo activo del personaje contrario (éste tiene que estar inconsciente), el jugador tiene que estar adyacente a él.

- **RF2242D. Acción**

  Descripción de la función: utiliza el objeto del bolsillo activo.

  Descripción de entrada: ninguna.

  Origen de la entrada: ninguno.

  Descripción de la salida: ninguna.

  Destino de la salida: ninguno.

  Requisito: ninguno.

  Precondiciones: la acción activa del personaje es la de *acción*.

  Poscondiciones: varían según el tipo de objeto.

- **RF2242E. Coger / Robar**

  Descripción de la función: añade al bolsillo activo del personaje el objeto obtenido, que puede estar en el suelo o en un bolsillo del personaje contrario.

  Descripción de entrada: objeto recogido.

  Origen de la entrada: sistema.

  Descripción de la salida: ninguna.

  Destino de la salida: ninguno.

  Requisito: ninguno.

  Precondiciones: la acción activa del personaje es la de *coger/robar*. El bolsillo activo del jugador está vacío (si no lo está se intercambian los objetos). Además el objeto tiene que estar adyacente al jugador o en un bolsillo del personaje contrario (estando éste adyacente al jugador). En este último caso el objeto recogido es el del bolsillo activo del personaje contrario.

  Poscondiciones: el objeto recogido está en el bolsillo activo del personaje.

- **RF2242F. Dejar**

Descripción de la función: deposita en el suelo el objeto del bolsillo activo del personaje.

Descripción de entrada: ninguna.

Origen de la entrada: ninguno.

Descripción de la salida: ninguna.

Destino de la salida: ninguno.

Requisito: ninguno.

Precondiciones: el personaje tiene un hueco adyacente para depositar el objeto. La acción activa debe ser *dejar*.

Poscondiciones: el bolsillo activo del personaje está vacío, y el objeto depositado se encontarará en el suelo, adyacente al personaje.

- **RF2242G. Suma Puntos**

Descripción de la función: añade una cantidad de puntos al personaje, por haber eliminado a un enemigo o por haber herido al personaje contrario.

Descripción de entrada: puntos obtenidos.

Origen de la entrada: sistema.

Descripción de la salida: ninguna.

Destino de la salida: ninguno.

Requisito: ninguno.

Precondiciones: partida activa.

Poscondiciones: puntuación del personaje actualizada.

3.2.3.3. Marcadores

- **RF2243A. Activar Acción**

Descripción de la función: resalta como activa la siguiente acción a la actual.

Descripción de entrada: ninguna (esta función selecciona automáticamente la siguiente acción activa en función de la actual, según el siguiente orden cíclico: selección de bolsillo, acción, coger/robar y dejar).

Origen de la entrada: ninguno.

Descripción de la salida: ninguna.

Destino de la salida: ninguno.

Requisito: ninguna.

Precondiciones: partida activa.

Poscondiciones: siguiente acción resaltada como actual.

- **RF2243B. Activar Bolsillo**

  Descripción de la función: resalta como activo el siguiente bolsillo al actual.

  Descripción de entrada: ninguna (esta función selecciona automáticamente el siguiente bolsillo activo en función del actual, según el siguiente orden cíclico: bolsillo1, bolsillo2, bolsillo3, bolsillo4).

  Origen de la entrada: ninguno.

  Descripción de la salida: ninguna.

  Destino de la salida: ninguno.

  Requisito: ninguno.

  Precondiciones: partida activa.

  Poscondiciones: siguiente bolsillo resaltado como actual.

- **RF2243C. Actualizar Reloj**

  Descripción de la función: actualiza la imagen del reloj según el tiempo restante.

  Descripción de entrada: tiempo restante.

  Origen de la entrada: sistema.

  Descripción de la salida: ninguna.

  Destino de la salida: ninguno.

  Requisito: ninguno.

  Precondiciones: partida activa.

  Poscondiciones: imagen actualizada en pantalla.

- **RF2243D. Actualizar Puntuación**

  Descripción de la función: actualiza la puntuación del jugador.

  Descripción de entrada: nueva puntuación.

  Origen de la entrada: sistema.

  Descripción de la salida: ninguna.

  Destino de la salida: ninguno.

  Requisito: ninguno.

  Precondiciones: partida activa.

  Poscondiciones: puntuación actualizada en pantalla.

3.2.4. Módulo Avanzado de Juego

3.2.4.1. Personajes No Jugadores

- **RF2251A. Mover**

Descripción de la función: mueve al enemigo en una determinada dirección, según un algoritmo de inteligencia artificial.

Descripción de entrada: ninguna.

Origen de la entrada: ninguno.

Descripción de la salida: ninguna.

Destino de la salida: ninguno.

Requisito: ninguno.

Precondiciones: partida activa.

Poscondiciones: enemigo en la nueva posición.

- **RF2251B. Atacar**

Descripción de la función: realiza un ataque, que variará dependiendo del enemigo (más o menos débil, corta, media o larga distancia, etc.).

Descripción de entrada: ninguna.

Origen de la entrada: ninguno.

Descripción de la salida: ninguna.

Destino de la salida: ninguno.

Requisito: ninguno.

Precondiciones: partida activa.

Poscondiciones: ataque efetuado.

3.2.4.2. Ítems

- **RF2252A. Ejecutar**

Descripción de la función: dependiendo del objeto, causa efectos en el propio personaje que lo utiliza, en el contrario o en los enemigos.

Descripción de entrada: ninguna.

Origen de la entrada: ninguno.

Descripción de la salida: ninguna.

Destino de la salida: ninguno.

Requisito: ninguno.

Precondiciones: partida activa.

Poscondiciones: varían dependiendo del objeto.

3.2.4.3. Lógica del Juego

- **RF2253A. Iniciar Partida Individual**

Descripción de la función: comienza una partida monojugador.

Descripción de entrada: personaje (humano o extraterrestre) elegido por el jugador.

Origen de la entrada: usuario.

Descripción de la salida: ninguna.

Destino de la salida: ninguno.

Requisito: ninguno.

Precondiciones: todavía no se ha iniciado una partida.

Poscondiciones: partida iniciada.

- **RF2253B. Iniciar Partida Doble**

Descripción de la función: comienza una partida multijugador.

Descripción de entrada: personaje (humano o extraterrestre) elegido por cada jugador.

Origen de la entrada: usuarios.

Descripción de la salida: ninguna.

Destino de la salida: ninguno.

Requisito: ninguno.

Precondiciones: terminal conectado con otro vía Bluetooth. El otro terminal también está ejecutando Vega Solaris. En ninguno de ambos terminales hay una partida iniciada.

Poscondiciones: partida iniciada.

- **RF2253C. Terminar Partida**

Descripción de la función: da fin a una partida, liberando recursos (memoria y conexiones) y actualizando el registro de puntuaciones máximas (en ambos terminales, en caso de partida multijugador).

Descripción de entrada: ninguna.

Origen de la entrada: ninguno.

Descripción de la salida: ninguna.

Destino de la salida: ninguno.

Requisito: ninguno.

Precondiciones: partida activa.

Poscondiciones: partida finalizada y, en el caso de partida multijugador, conexión Bluetooth finalizada. Registro de puntuaciones máximas actualizado.

- **RF2253D. Cargar Datos Iniciales**

Descripción de la función: carga en memoria los datos necesarios para el funcionamiento de la aplicación: mapa, enemigos y sus gráficos correspondientes.

Descripción de entrada: ninguna.

Origen de la entrada: ninguno.

Descripción de la salida: ninguna.

Destino de la salida: ninguno.

Requisito: repositorio con todos los datos necesarios.

Precondiciones: todavía no se han cargado los datos.

Poscondiciones: todos los datos necesarios en memoria.

- **RF2253E. Mostrar Mensaje**

  Descripción de la función: muestra un mensaje por pantalla.

  Descripción de entrada: el texto del mensaje que se quiere mostrar.

  Origen de la entrada: sistema.

  Descripción de la salida: ninguna.

  Destino de la salida: ninguno.

  Requisito: ninguno.

  Precondiciones: ninguna.

  Poscondiciones: en pantalla aparece el mensaje indicado.

3.3. Requisitos de Rendimiento

- **RR0**. Sólo habrá que dar servicio a un terminal, en el caso del modo monojugador, y a dos terminales en el caso del modo multijugador.
- **RR1**. El tiempo de respuesta del sistema VS debe ser muy bajo para poder ofrecer una experiencia de juego en tiempo real, tanto en modo monojugador como en modo multijugador. El tiempo de respuesta del sistema debes ser inapreciable para el jugador durante la partida, pudiendo ser de varios segundos (máximo 10) durante la navegación entre menús.
- **RR2**. El sistema debe ser fiable en modo multijugador aún en el caso de fallos en la red local, permitiendo que la partida continúe en ambos extremos aunque se pierda momentáneamente la conexión con el otro terminal. Evidentemente si la conexión no se recuperase en un período de tiempo razonable (1 minuto), el sistema deberá informar a los usuarios y cancelar la partida.

3.4. Requisitos de Base de Datos Lógica

No hay requisitos de esta naturaleza por no haber base de datos lógica.

3.5. Restricciones de Diseño

3.5.1. Requisitos de Desarrollo

- **RDDes0**. Se construirá un prototipo monousuario de la aplicación para PC, que contendrá todos los requisitos recogidos en el presente documento (excepto todos los referentes al modo multiusuario, ya que dicho modo no estará implementado). Es decir, será una versión final monousuario de la aplicación pero funcionando sobre un PC – el sistema operativo es irrelevante, dado que se programará con Java y por tanto será multiplataforma.
- **RDDes1**. Una vez construido el primer prototipo se realizará la migración del sistema a un teléfono móvil.
- **RDDes2**. Una vez que el prototipo funcione correctamente en un teléfono móvil se implementará el modo multiusuario.

3.5.2. Requisitos de Diseño

- **RDDis0.** Las clases construidas deben tener el mayor nivel de abstracción posible, de manera que sea posible modificar, en cualquier momento, la forma de representación de los distintos elementos del juego (mapas, enemigos, ítems, etc.) sin afectar al resto del diseño.

3.5.3. Requisitos de Tecnología

- **RDT0**. Los terminales serán teléfonos móviles de muy distintas características, pero al menos deberán tener soporte para juegos Java y Bluetooth.
- **RDT1**. La implementación se realizará en J2ME.

3.6. Atributos del Sistema Software

- **RS0**. El sistema debe tener una alta mantenibilidad, de manera que en un futuro permita modificar los distintos aspectos del juego (escenarios, gráficos, IA de los personajes…) de manera fácil y rápida.
- **RS1**. El sistema debe ser portable a varios modelos de teléfonos móviles, por lo que el diseño y la implementación no deben ser dependientes del dispositivo hardware en el que se implante. Esto es especialmente importante en cuanto a las dimensiones de la pantalla, muy variables de unos modelos a otros.

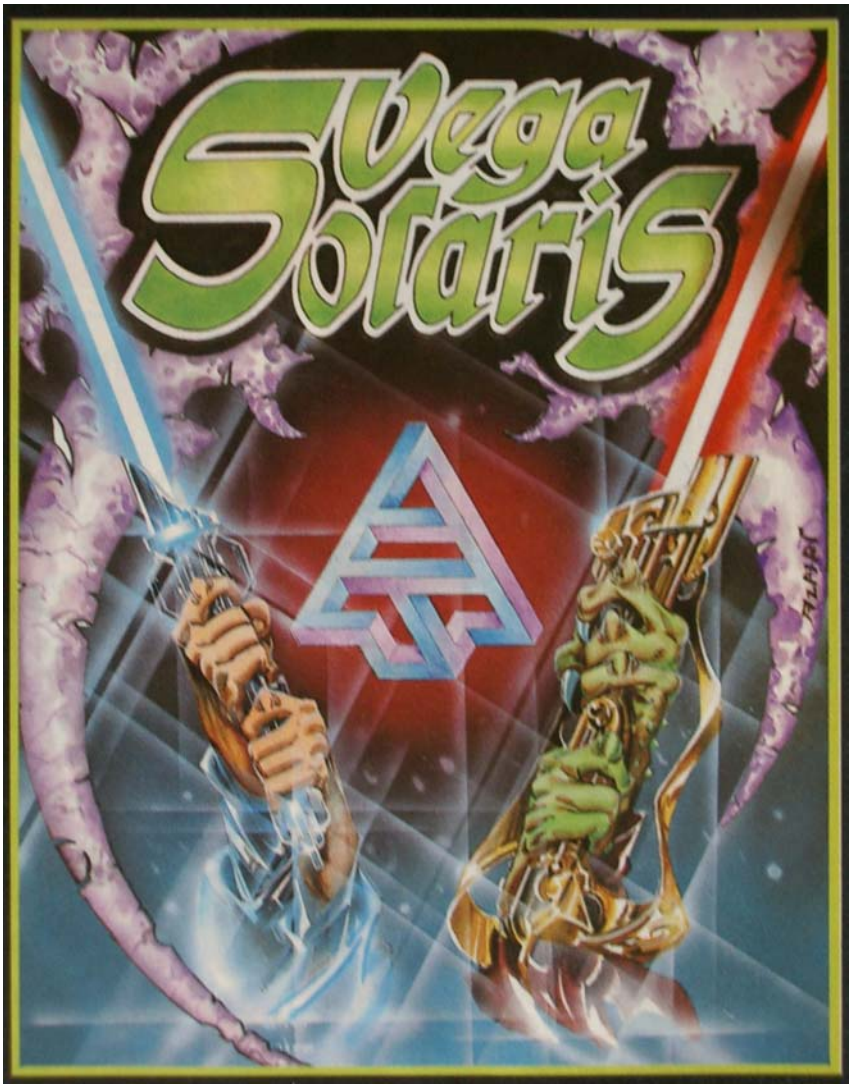
3.7. Otros Requisitos

No hay requisitos que no hayan sido especificados en las secciones anteriores.

APÉNDICES

Ficha Técnica de Vega Solaris (1989)

- Título: Vega Solaris.
- Plataforma: Sinclair ZX Spectrum.
- Formato: TZX y Z80.
- Aparición: 1989.
- Tipo: arcade.
- Autores: Fernando Sáenz Pérez y Carlos García Cordero (Quasar/Eclipse).
- Sinopsis: un planeta recóndito es el escenario de un duelo entre dos seres totalmente distintos: un humano y un extraterrestre. Ambos quieren encontrar las piezas que conforman el símbolo de VEGA SOLARIS. Sólo uno podrá conseguirlo.
- Otros datos de interés: esta versión se presentó en MadriSX&Retro 2005.



Capturas del videojuego original

Pantalla de carga

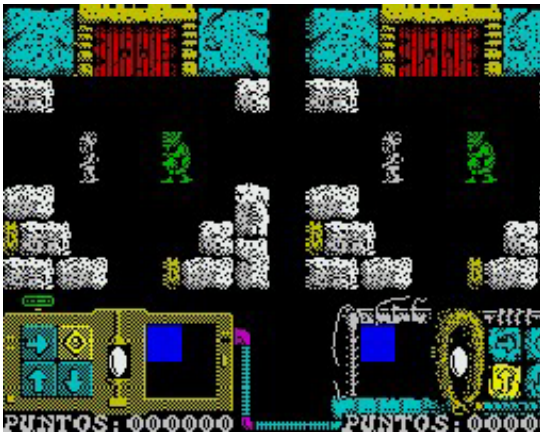
En el videojuego original, el proceso de carga y ejecución del juego es el habitual: escribir LOAD “”, insertar la cinta y pulsar PLAY en el cassete. Si se usa un emulador como Spectaculator o ZX Spin se puede cargar el archivo TZX a velocidad normal o rápida.

En el primer caso las cosas suceden como en el Spectrum 48K original.En concreto, se usa una rutina especial de carga desde casete que dibuja la pantalla por líneas consecutivas, en lugar del proceso de dibujo habitual.

Además, y una vez dibujada la pantalla de carga que aparece en la siguiente figura, aparece una versión de “**Space Invaders**” para jugar mientras dure el resto del proceso de carga. El objetivo de este pequeño juego es simplemente obtener el máximo número de puntos “matando marcianitos”. La puntuación indica el número de ellos abatidos y se hace cero si alguno consigue llegar a nuestra posición. Los patrones de movimiento cambian ligeramente según vienen nuevas hordas de marcianos.

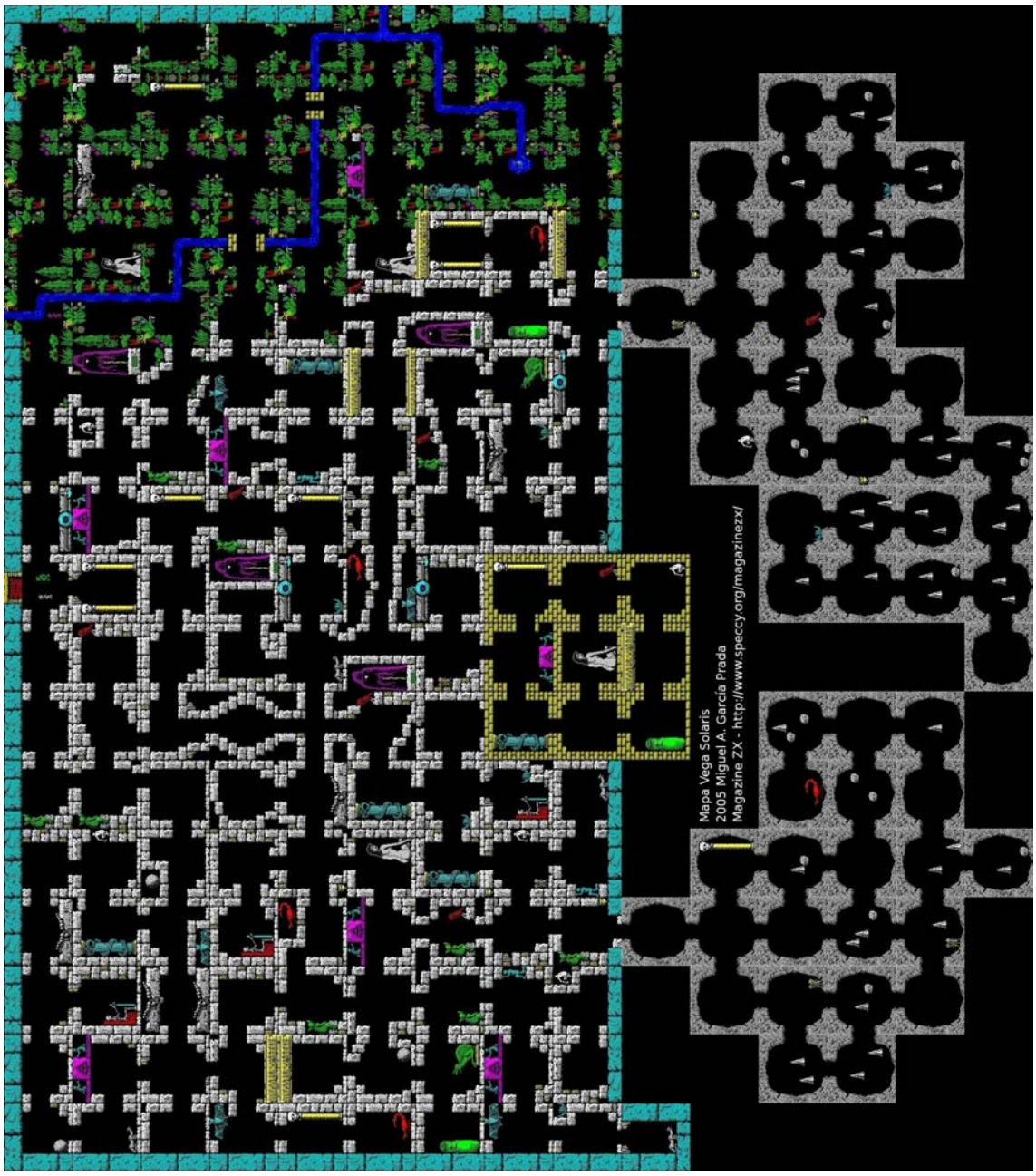
Las teclas de juego son: Q para la izquieda, P para la derecha y Control para disparar. Si se pulsa Break (la barra espaciadora en un PC) se interrumpe el proceso de carga.

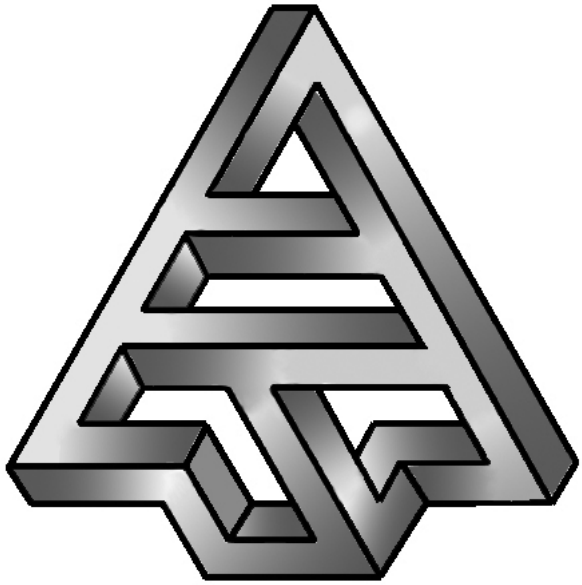

VEGA SOLARIS
1 JUGADOR 1: SPECTRUM
2 JUGADOR 2: SPECTRUM
3 DEFINIR TECLADO 1
4 DEFINIR TECLADO 2
0 EMPEZAR JUEGO
PROGRAMA POR ECLIPSE 1989


PUNTOS: 000000
PUNTOS: 0000


PUNTOS: 001250
PUNTOS: 000000

Mapa del juego





Vega Solaris

SRS Anexo 1 - Ampliaciones al Producto Original

Versión 1.0
04/07/2006

ÍNDICE

1. PREÁMBULO

Este documento recoge todas las modificaciones realizadas en el sistema Vega Solaris respecto a la versión de 1989.

El hecho de incluir estas modificaciones en un documento aparte, y no en la propia SRS, es debido a que todos los aspectos que se recogen aquí no son requisitos, en cuanto a que de su implementación no depende la corrección y completitud del sistema. No obstante son aspectos que contribuyen a aumentar la jugabilidad de Vega Solaris en su versión para un nuevo soporte – como el un teléfono móvil – y un nuevo mercado, muy distinto al de 1989 debido a la gran expansión que ha experimentado el mundo de los videojuegos, en concreto el de juegos para móviles.

2. DESCRIPCIÓN DE LAS AMPLIACIONES

2.1. Introducción de Nuevas Funciones en el Sistema

En algunos casos las ampliaciones realizadas mejoran la funcionalidad del sistema sin añadir nada, pero en otros casos añaden cierta funcionalidad, con lo que aparecen nuevas funciones. Estas funciones se codificarán en lo sucesivo mediante la signatura **AP** (referente a Ampliaciones del Producto) seguida del número de apartado en el presente documento, y terminando en una letra que hace referencia a su posición dentro de un apartado con varias funciones nuevas.

2.2. Ampliaciones

2.2.1. Cambio de vista

En la versión de VS de 1989, para poder localizar (aproximadamente) dónde se encontraba el enemigo, bastaba observar el lado correspondiente de la pantalla (lado derecho para observar al extraterrestre, y lado izquierdo para observar al humano). Dado que en la nueva versión se va a eliminar esa parte de la pantalla, y en la misma sólo aparecerá la vista del propio jugador, la localización del enemigo resulta imposible.

Por este motivo, se ha introducido la posibilidad de cambiar la vista de la aplicación, de manera que mientras se pulse cierto botón, no se verá en pantalla la habitación en la que se encuentra el jugador sino la habitación en la que se encuentra el enemigo.

Esta mejora introduce la necesidad de una nueva función del sistema:

- **AP221a. Mostrar pantalla enemigo**: muestra por pantalla la habitación en la que se encuentra el enemigo, en lugar de la habitación en la que se encuentra el jugador.

2.2.2. Panel de control

Otro aspecto que se ha querido mejorar es el marcador del personaje. Se ha considerado que es más intuitivo indicar cada acción mediante una imagen representativa de cada una ellas en lugar de flechas. Se definen por tanto las siguientes imágenes o iconos para cada una de las nuevas acciones del personaje:

- **Coger / robar**: una mano cerrada (como agarrando algo).
- **Dejar**: una mano abierta (como dejando caer algo).
- **Usar / puñetazo**: un puño.
- **Cambiar de bolsillo**: dos flechas dispuestas circular y cíclicamente.

El hecho de emplear unos iconos distintos a los originales no introduce nuevas funciones en el sistema.

2.2.3. Suelo de las habitaciones

Se desea incluir la posibilidad de que las habitaciones, además de tener objetos de "decorado", puedan tener también una imagen de fondo que represente el suelo de dicha habitación. Caben dos posibilidades:

- Utilizar una imagen fija, que se muestra tal cual.
- Utilizar una imagen que se pintará repetidas veces, en disposición matricial, de manera que se consiga un efecto de "textura".

Esta mejora introduce la necesidad de una nueva función del sistema:

- **AP223a. Embaldosar**: pinta en la pantalla la imagen de fondo de la habitación, repetidas veces y en disposición matricial, de manera que se cubra todo el área visible.

2.3. Nuevas Funciones

- **AP221A. Mostrar Pantalla Enemigo**

Descripción de la función: cambia la vista actual, de manera que en vez de aparecer en pantalla la habitación en la que se encuentra el jugador, aparezca la habitación en la que se encuentra el enemigo.

Descripción de entrada: jugador que ha solicitado el cambio de vista.

Origen de la entrada: aplicación.

Descripción de la salida: ninguna.

Destino de la salida: ninguno.

Requisito: ninguno.

Precondiciones: partida activa.

Poscondiciones: partida activa.

- **AP223A. Embaldosar**

Descripción de la función: cubre todo el área visible dibujando repetidas veces una imagen, en disposición matricial.

Descripción de entrada: imagen que se pintará.

Origen de la entrada: aplicación.

Descripción de la salida: ninguna.

Destino de la salida: ninguno.

Requisito: ninguno.

Precondiciones: partida activa.

Poscondiciones: partida activa.

Diseño

En este apartado se mostrará la estructura de clases de la aplicación. La aplicación ha sido separada en ocho módulos o paquetes:

- **Controlador**: con las clases que implementan los controladores de esta aplicación,
- **Elementos**: con las clases que representan a los objetos, armas, conjuros... del juego,
- **Inteligenciaartificial**: con las clases y algoritmos para simular la inteligencia artificial de los bichos y del enemigo,
- **Mapa**: con las clases que recogen la información del mapa de la partida,
- **Partida**: con las clases que implementan la lógica de la aplicación,
- **Persistencia**: con las clases encargadas de salvar los datos de la aplicación al disco duro y de leerlos posteriormente,
- **Vega_solaris**: con el código principal de la aplicación y clases con sonstantes globales,
- **Vista**: con todas las clases encargadas del interfaz con el usuario.

A continuación, se muestran los diagramas de clases en UML agrupándolos por paquetes para facilitar su comprensión.

Diagramas UML

Diagramas de casos de uso

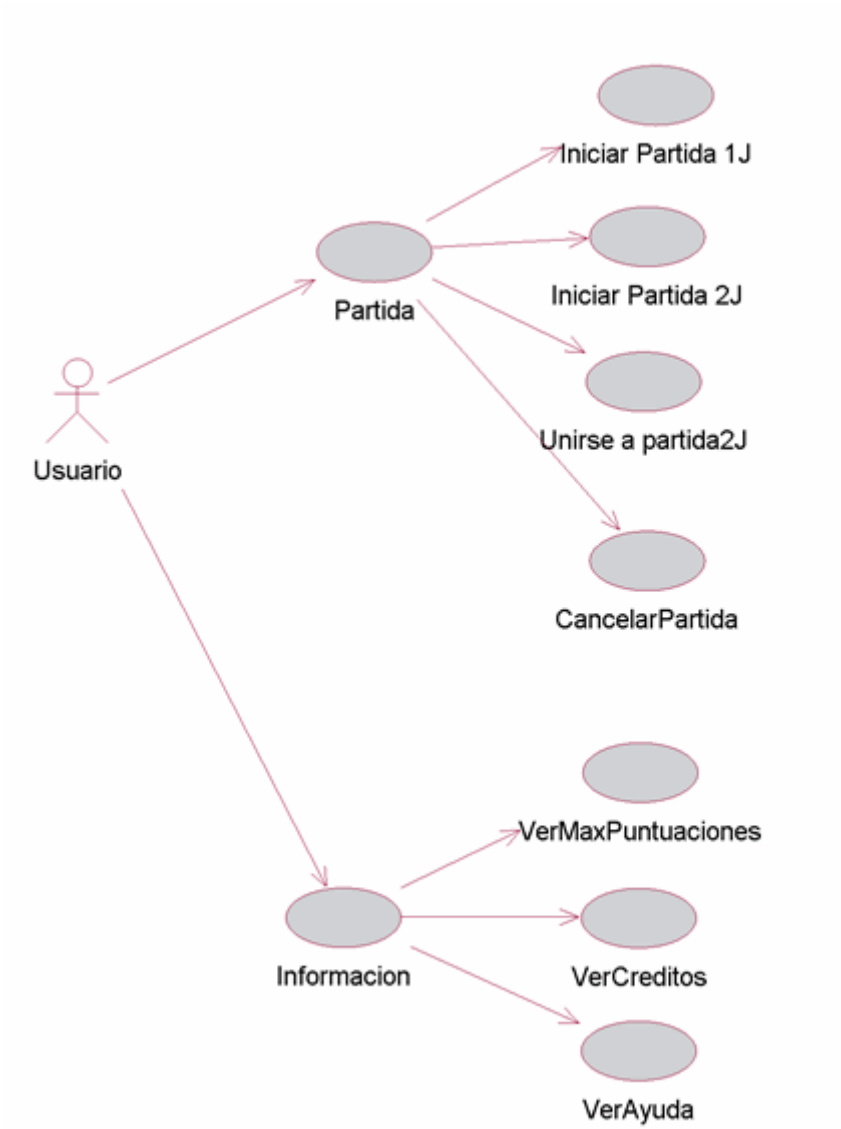


Diagrama de casos de uso principal

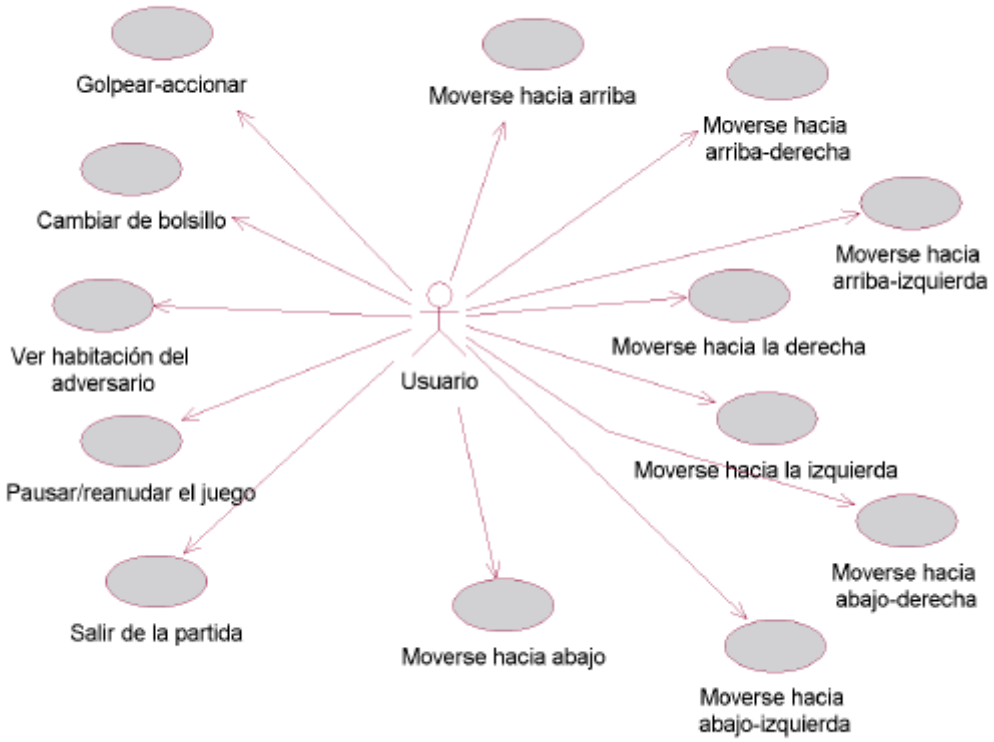


Diagrama de casos de las acciones posibles durante una partida

Diagrama de secuencia

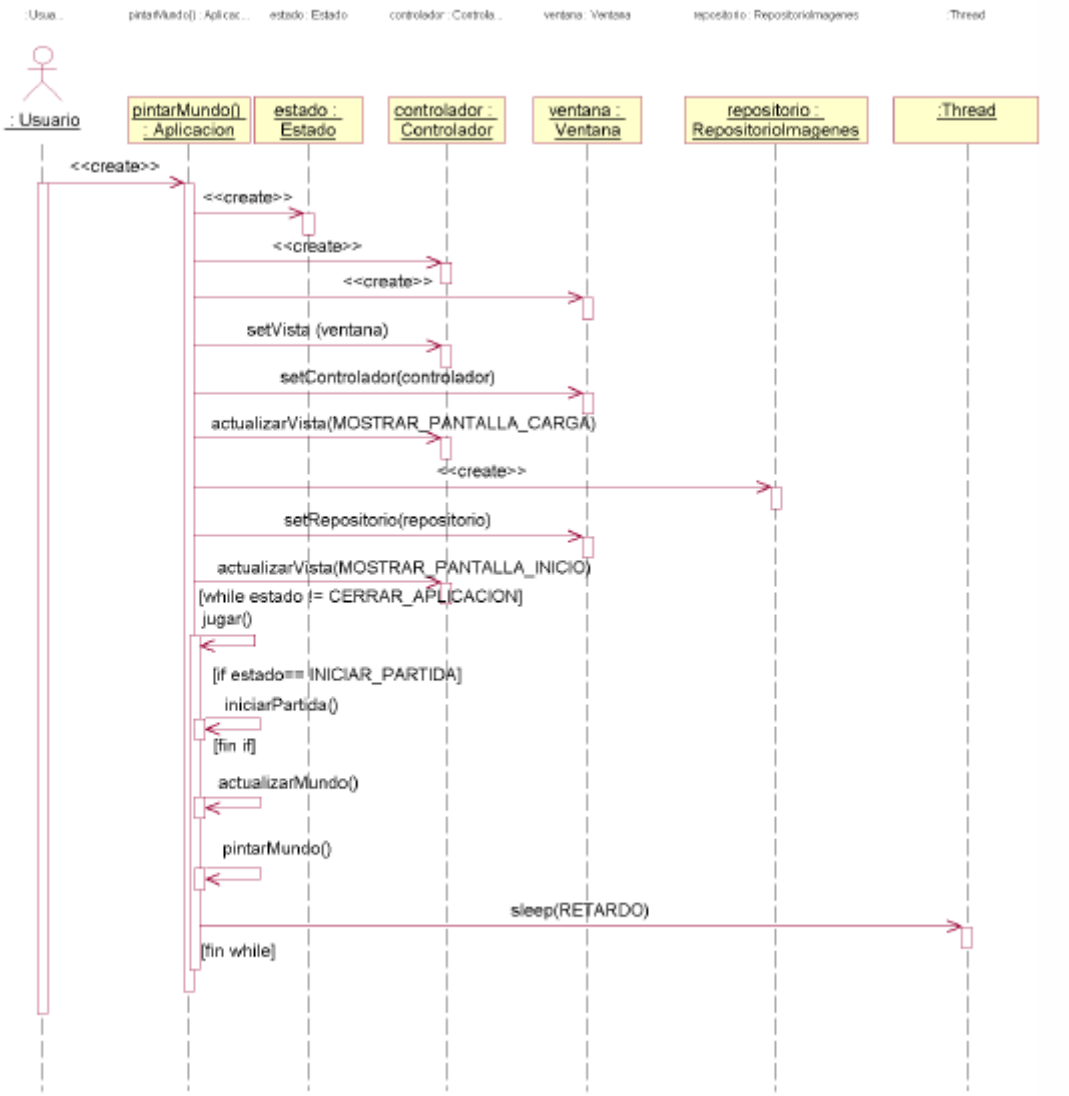


Diagrama de secuencia de la creación de la aplicación y del *game-loop*

Diagramas de clases

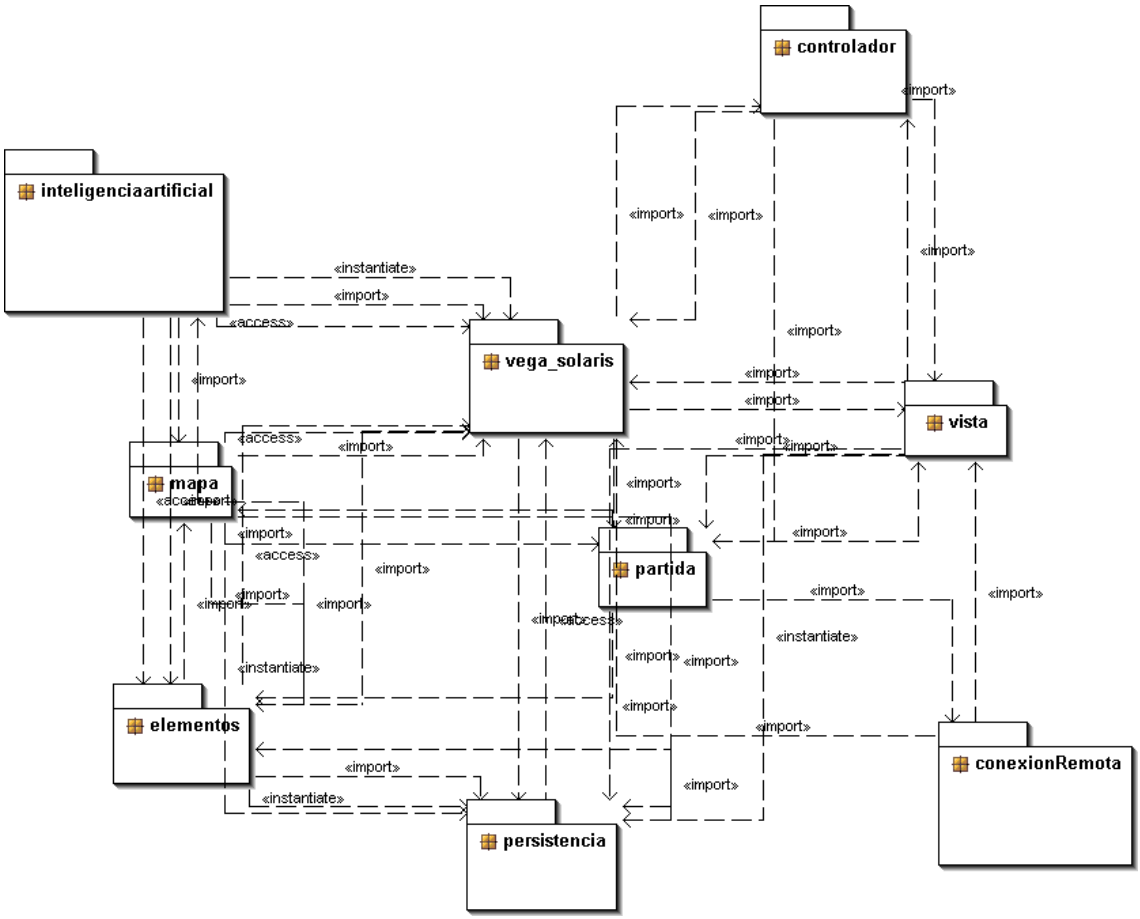


Diagrama de clases de genérico a nivel de paquetes de toda la aplicación

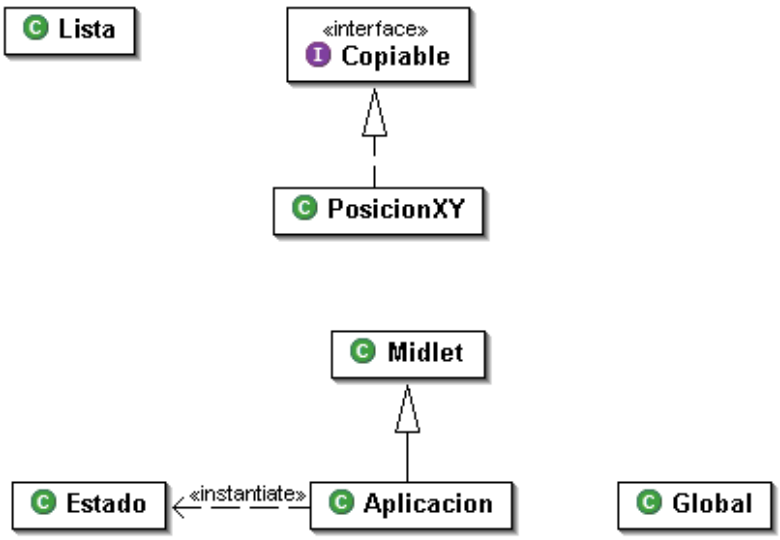


Diagrama de clases del paquete vega_solaris

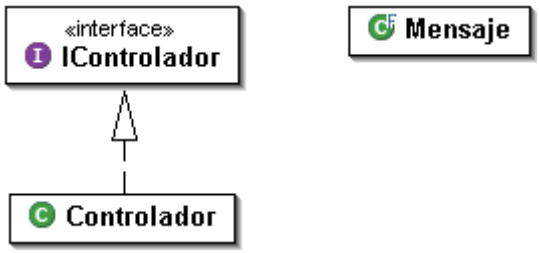


Diagrama de clases del paquete *controlador*

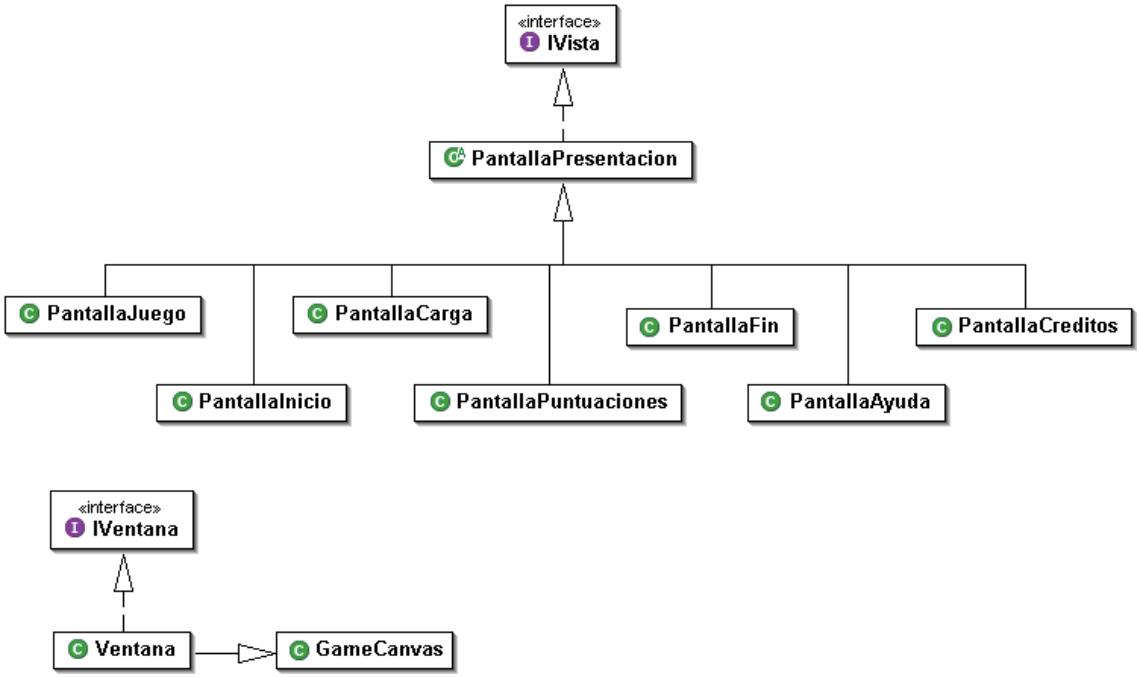


Diagrama de clases del paquete *vista*

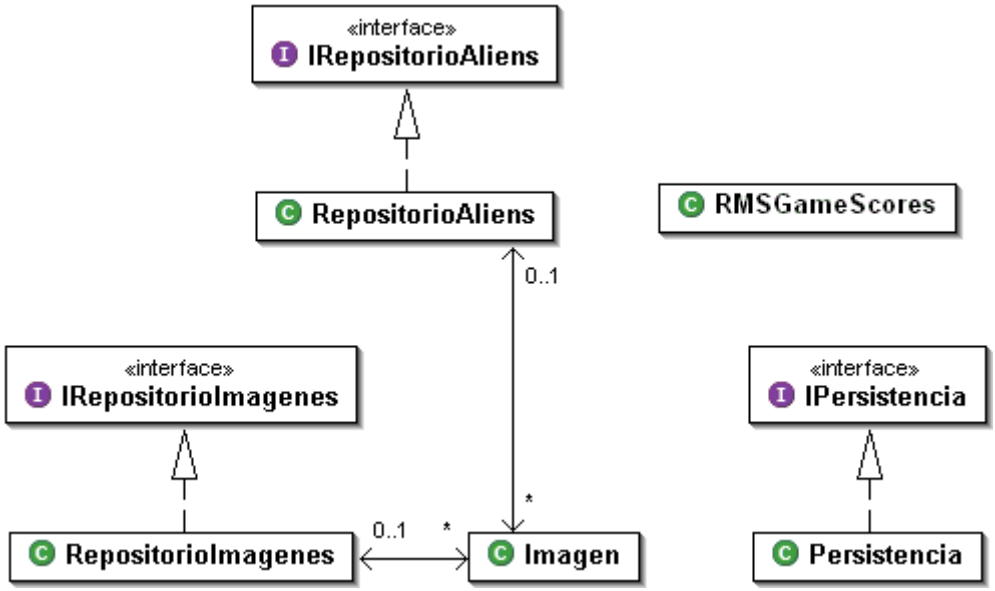


Diagrama de clases del paquete *persistencia*

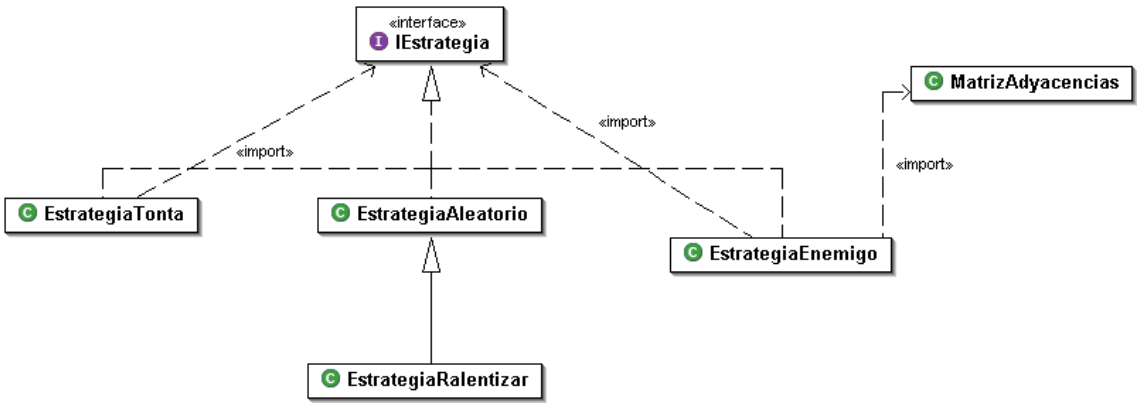


Diagrama de clases del paquete inteligenciaartificial

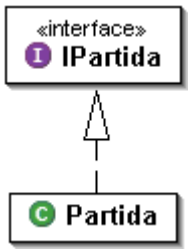


Diagrama de clases del paquete partida

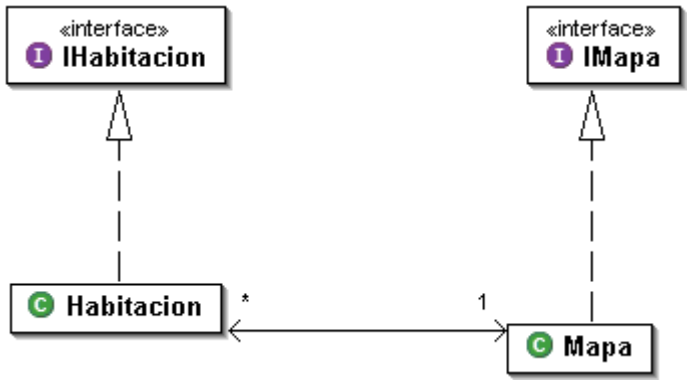


Diagrama de clases del paquete mapa

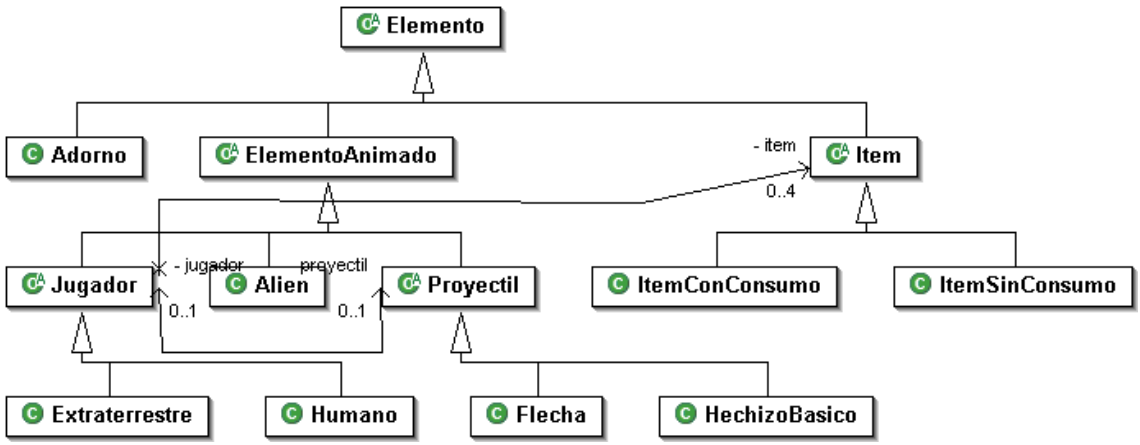


Diagrama de clases del paquete elementos

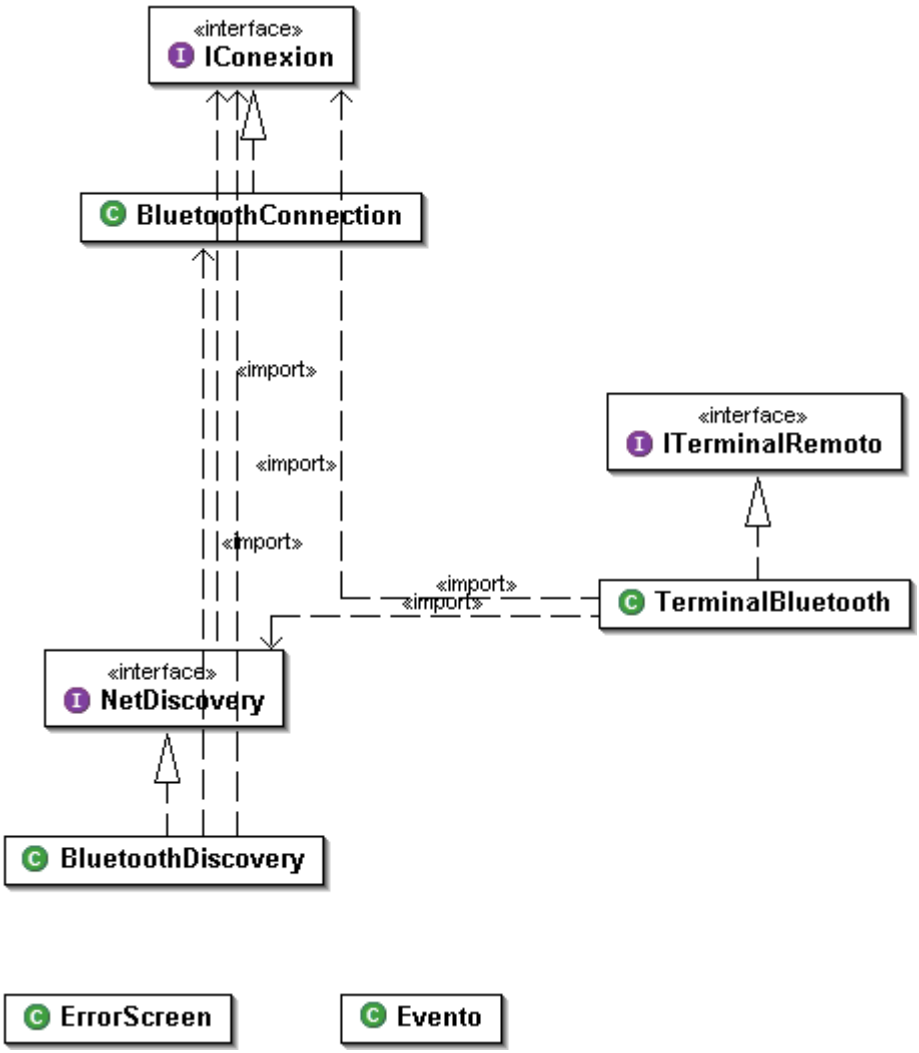
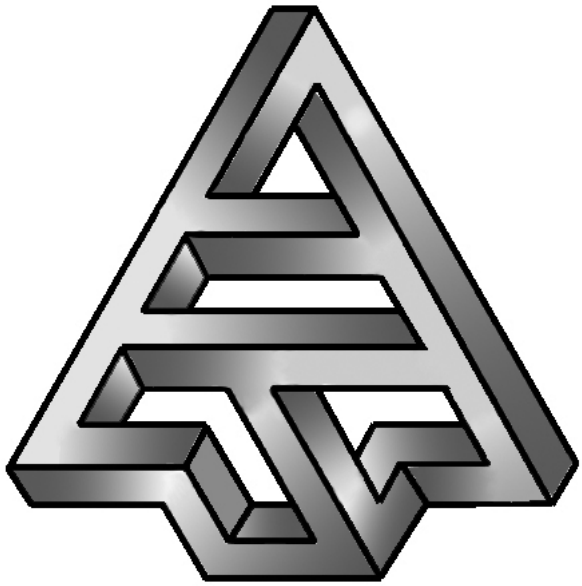


Diagrama de clases del paquete conexionRemota



Vega Solaris

Diseño de la Inteligencia Artificial

Versión 1.0
04/07/2006

ÍNDICE

1. VISIÓN GENERAL

El presente documento describe el funcionamiento y utilización del paquete **inteligenciaartificial** para dotar a la aplicación de un módulo de Inteligencia Artificial para el control automático de los personajes enemigos.

El contenido del paquete es el siguiente:

- class **EstrategiaAleatorio** implements IEstrategia: una implementación de la inteligencia artificial para los alienígenas del juego.
- class **EstrategiaEnemigo** implements IEstrategia: una implementación de la inteligencia artificial para el personaje extraterrestre, el personaje oponente (en modo monojugador).
- class **EstrategiaRalentizar** extends EstrategiaAleatorio: otra implementación de la inteligencia artificial para los alienígenas del juego.
- class **EstrategiaTonta** implements IEstrategia: otra implementación de la inteligencia artificial para los alienígenas del juego.
- interface **IEstrategia**: interfaz para permitir distintas implementaciones de la Inteligencia Artificial.
- class **MatrizAdyacencias**: utilizada por la clase **EstrategiaEnemigo** para realizar búsquedas de caminos en el mapa de juego.

1.1. El interfaz IEstrategia

1.1.1. API

Constantes utilizadas por el archivo **aliens.vs** para decidir la estrategia utilizada por un alien (ver el documento *"Formato de los Archivos de Datos"* para una descripción detallada del archivo de configuración **aliens.vs**):

- public static final String **ESTRATEGIA_RALENTIZAR**
- public static final String **ESTRATEGIA_ALEATORIO**
- public static final String **ESTRATEGIA_TONTA**

Constantes utilizadas por las estrategias de los aliens para la toma de decisiones:

- public static final int **ARRIBA**: decisión de mover hacia arriba.
- public static final int **DERECHA**: decisión de mover hacia la derecha.
- public static final int **ABAJO**: decisión de mover hacia abajo.
- public static final int **IZQUIERDA**: decisión de mover hacia la izquierda.
- public static final int **ATACAR**: decisión de atacar (cuerpo a cuerpo).

Métodos

- public void **actuar**(): evalúa el estado del mundo, toma decisiones sobre la siguiente acción a realizar y cambia el estado interno del personaje de acuerdo a dichas decisiones.
- public void **atacar**(ElementoAnimado j): a
- public IEstrategia **copia**(Alien a): crea una estrategia del mismo tipo a la actual para el alien indicado en el parámetro.
- public void **cambioHabitacion**(): informa a la estrategia de que el personaje ha cambiado de habitación, para que actualice sus atributos internos.
- public void **setPropietario**(Jugador e): establece al jugador (humano o extraterrestre) indicado como parámetro como destinatario de la estrategia actual.

1.1.2. Usar IEstrategia para implementar nuevas estrategias

Cualquier clase que implemente una estrategia para la Inteligencia Artificial, ya sea para un tipo de alien o para el oponente extraterrestre, debe implementar a la interfaz IEstrategia, pero además debe tener en cuenta lo expuesto a continuación.

1.1.2.a. Estrategias para los aliens

Para implementar nuevas estrategias para los aliens, sólo se utilizan los métodos *actuar()*, *atacar(ElementoAnimado j)* y *copia(Alien a)*, de manera que el cuerpo de los métodos *cambioHabitacion()* y *setPropietario(Jugador e)* pueden dejarse vacíos.

El método **actuar()** se utiliza para actualizar la dirección de movimiento del alien, de acuerdo con la estrategia que se desee implementar.

El método **atacar(ElementoAnimado j)** se utiliza para efectuar un ataque sobre un objeto de la clase ElementoAnimado. Al invocar a este método, el objeto j perderá cierta cantidad de puntos de vida y/o sufrirá ciertos efectos secundarios del ataque (ralentización, parálisis, etc.).

El método **copia(Alien a)** debe devolver una nueva estrategia del mismo tipo que la actual, y además debe vincularla al objeto a, de tipo *Alien*.

1.1.2.b. Estrategias para el oponente (extraterrestre)

Para implementar nuevas estrategias para el oponente extraterrestre, sólo se utilizan los métodos *actuar()*, *cambioHabitacion()* y *setPropietario(Jugador e)*. Los métodos *atacar(ElementoAnimado j)* y *copia(Alien a)* pueden implementarse vacíos.

El método **actuar()** engloba todas las posibles decisiones del oponente sobre la siguiente acción a ejecutar, en base al estado del mundo. Todas las clases de la aplicación constan de numerosos métodos de consulta y modificación de atributos, por lo que la percepción del estado de la partida o del mundo puede realizarse de forma sencilla. Una vez percibida y evaluada la situación se tomarán las acciones oportunas, que dependerán de la implementación de la estrategia concreta. Por tanto, una posible implementación de este método podría ser:

```
public void actuar() {
     evaluarSituacion(); //Consulta del estado del mundo
     tomarDecision();    //En base a la consulta anterior,
decidir acción
     efectuarAccion();  //Ejecutar la acción decidida
}
```

El método **cambioHabitacion()** es invocado por la aplicación cada vez que un jugador (humano o extraterrestre) cambia de una habitación a otra. Puede ser utilizado por el método **actuar()** para saber cuándo el personaje ha cambiado de habitación, y tomar así cierto tipo de decisiones.

El método **setPropietario(Jugador e)** es utilizado para establecer el jugador usuario de la estrategia de Inteligencia Artificial. Puede utilizarse, por ejemplo, para intercambiar los roles del personaje humano y el del extraterrestre, de manera que el usuario maneje al extraterrestre y el humano sea controlado mediante la estrategia de Inteligencia Artificial.

1.3. Selección de estrategias para los aliens

En el paquete se incluyen varias clases que implementan la interfaz IEstrategia. Unas implementan la Inteligencia Artificial de los aliens, y otras implementan la Inteligencia Artificial del oponente.

El archivo de datos (o de configuración) *aliens.vs*, que se encuentra en el directorio */res/datos/* de la aplicación, describe los distintos tipos de aliens que aparecerán durante el juego. Una de las características que se incluye en dicha descripción es la estrategia que se debe utilizar para controlar a cada tipo de alien, de manera que editando este archivo se puede seleccionar qué estrategias se le asigna a qué tipo de alien.

Para una información detallada sobre la selección de estrategias para los aliens y el formato y contenido del archivo *aliens.vs*, consultar el documento *"Formato de los Archivos de Datos"*.

2. INTELIGENCIA ARTIFICIAL DE LOS ALIENÍGENAS

2.1. Estrategia "Tonta"

La clase EstrategiaTonta implementa el siguiente comportamiento: el alien elige al azar la dirección en la que se moverá: arriba, abajo, izquierda o derecha. Una vez elegida la dirección de desplazamiento, avanza en dicha dirección hasta encontrar un obstáculo. Entonces, vuelve a elegir aleatoriamente una dirección de desplazamiento que, incluso, puede ser la misma que llevaba hasta ahora (por lo que volverá a encontrarse inmediatamente con el obstáculo y tendrá que elegir una nueva dirección). Es posible, por tanto, que un alien con estrategia "Tonta" se quede a veces golpeándose contra un obstáculo.

Durante el desplazamiento del alien, éste siempre comprueba si es adyacente al humano o al extraterrestre, en cuyo caso le ataca, causándole pérdida de puntos de vida.

2.2. Estrategia "Aleatoria"

La clase EstrategiaAleatoria se comporta de la siguiente manera:

1. El alien elige aleatoriamente si realizará un movimiento horizontal y vertical, o por el contrario realizará un movimiento diagonal.
2. Una vez elegido el tipo de movimiento (horizontal-vertical o diagonal) elige aleatoriamente la dirección de desplazamiento, en base a dicho movimiento. Las posibles direcciones de desplazamiento que puden seguirse, según el tipo de movimiento, son:
    - Movimiento horizontal-vertical: arriba, abajo, izquierda o derecha.
    - Movimiento diagonal: arriba-izquierda, arriba-derecha, abajo-izquierda o abajo-derecha.
3. El alien se mueve en la dirección elegida hasta encontrarse con un obstáculo, en cuyo caso volverá a elegir aleatoriamente un tipo de movimiento y una dirección de desplazamiento en base al tipo de movimiento elegido.

Al igual que en la estrategia "Tonta", si un alien se encuentra adyacente al humano o al extraterrestre, le atacará, infligiéndole daño.

2.3. Estrategia "Ralentizar"

La clase EstrategiaRalentizar hereda de la clase EstrategiaAleatoria, por lo que su comportamiento es el mismo, con una particularidad: al atacar a un jugador no sólo le quita vida, sino que además le hace entrar en estado ralentizado.

NOTA: en el estado ralentizado, el personaje se mueve más despacio de lo normal. Al igual que el resto de estados en los que se puede encontrar un jugador, el estado ralentizado desaparece al cabo de cierto tiempo.

3. INTELIGENCIA ARTIFICIAL DEL ENEMIGO

3.1. Aspectos generales

El comportamiento del personaje enemigo es mucho más complejo que el de los aliens, ya que además de moverse y atacar puede o debe realizar otras muchas acciones: recoger, usar y soltar ítems, cambiar de habitación, buscar los cristales, robárselos al otro jugador, etc. Es por esto que la implementación de la Inteligencia Artificial para controlar a un personaje de estas características se complica también mucho.

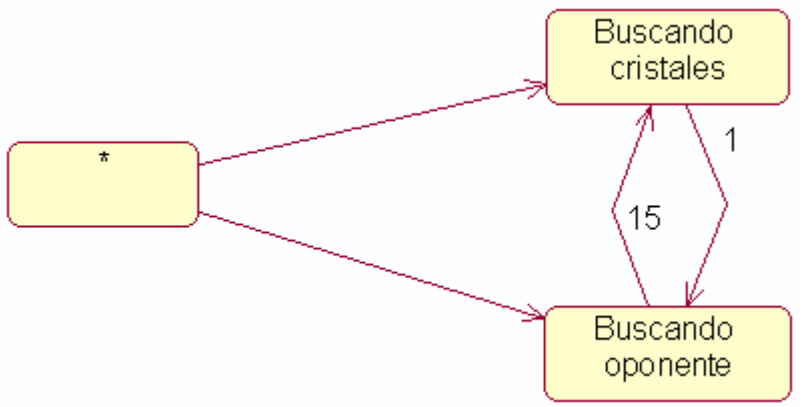
La clase EstrategiaEnemigo proporciona una implementación eficiente – esto es importante ya que el control del enemigo mediante técnicas de Inteligencia Artificial no debe suponer un consumo notable de recursos – de la estrategia a seguir por parte de un jugador, basándose en una máquina de estados.

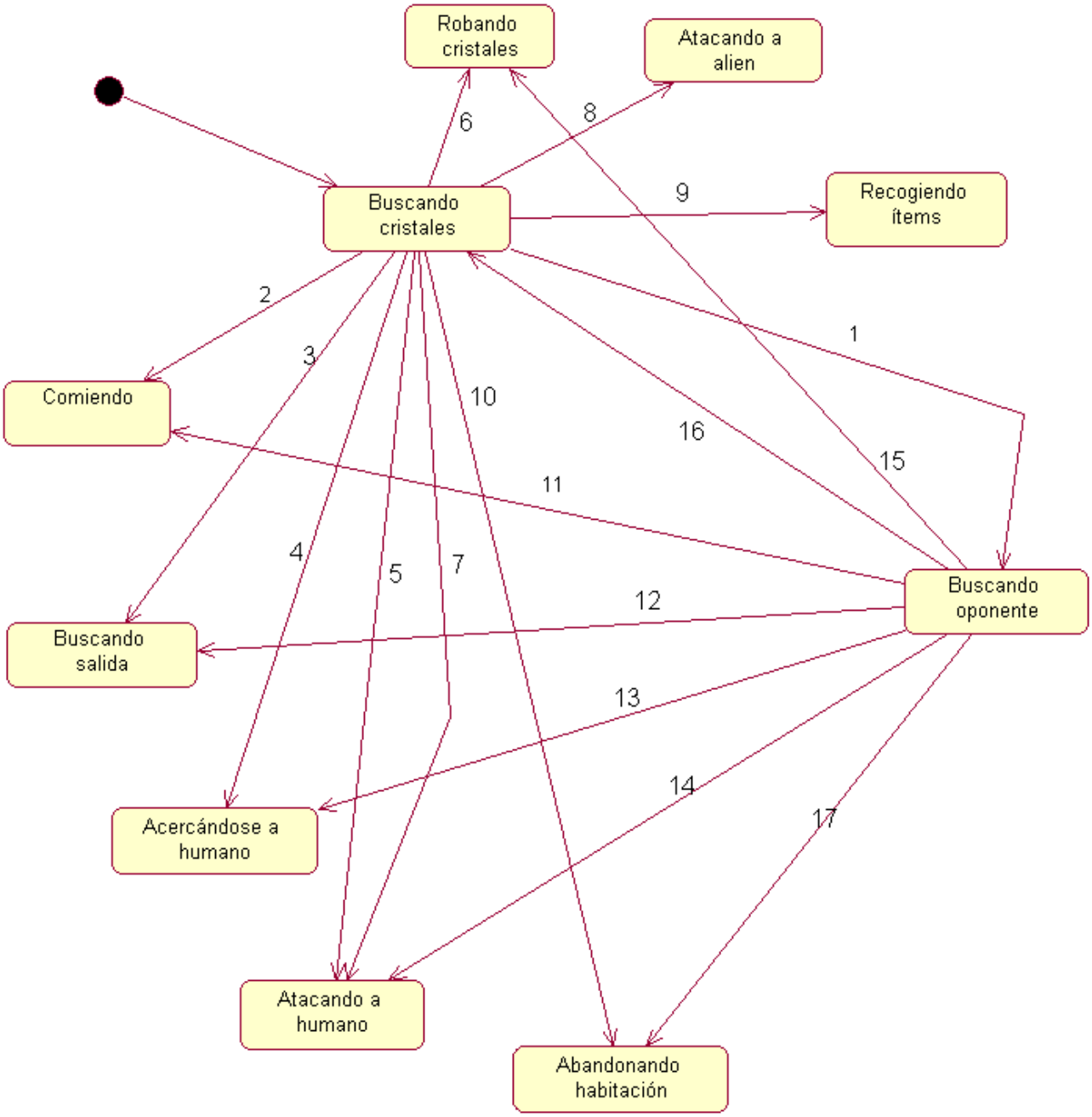
A continuación se expone detalladamente el autómata que controla al enemigo en el caso de jugar una partida monojugador.

3.2. Descripción detallada

3.2.1. Máquina de estados

El siguiente AFN-λ representa la máquina de estados de la clase EstrategiaEnemigo. Por claridad, se han omitido las transiciones λ. Dichas transiciones se producen automáticamente desde cualquier estado (excepto "Buscando cristales" y "Buscando oponente") a uno de los dos siguientes: "Buscando cristales" o "Buscando oponente":





Las transiciones, que aparecen representadas numéricamente en el AFN-λ, se producen cuando los indicadores correspondientes se hacen ciertos:

1. enemigo_tiene_los_que_me_faltan
2. (NOT 1) AND poca_vida AND tengo_item_comida
3. (NOT 2) AND tengo_todos_los_cristales
4. (NOT 3) AND hay_enemigo_con_cristal AND (NOT adyacente_a_enemigo) AND [(NOT tengo_proyectiles) OR enemigo_inconsciente]
5. (NOT 4) AND hay_enemigo_con_cristal AND (tengo_proyectiles OR adyacente_a_enemigo) AND (NOT enemigo_inconsciente)
6. (NOT 5) AND hay_enemigo_con_cristal AND adyacente_a_enemigo AND enemigo_inconsciente
7. (NOT 6) AND adyacente_a_enemigo AND (NOT enemigo_inconsciente)
8. (NOT 7) AND adyacente_a_alien
9. (NOT 8) AND hay_item_interesante

10. NOT 9
11. poca_vida AND tengo_item_comida
12. (NOT 11) AND tengo_todos_los_cristales
13. (NOT 12) AND hay_enemigo_con_cristal AND (NOT adyacente_a_enemigo) AND (NOT tengo_proyectiles OR enemigo_inconsciente)
14. (NOT 13) AND hay_enemigo_con_cristal AND (tengo_proyectiles OR adyacente_a_enemigo) AND (NOT enemigo_inconsciente)
15. (NOT 14) AND hay_enemigo_con_cristal AND adyacente_a_enemigo AND enemigo_inconsciente
16. (NOT 15) AND (NOT enemigo_tiene_los_que_me_faltan)
17. NOT 16

3.2.2. Indicadores

Para realizar las transiciones entre estados se utilizan una serie de indicadores, que no son más que atributos que toman el valor *cierto* o *falso* y que indican si se cumple cierta condición en el momento actual. El valor de estos atributos se calcula en cada llamada al método *actuar()* de la clase EstrategiaEnemigo, utilizando para ello los diversos métodos de consulta del resto de clases de la aplicación.

En esta estrategia, se considera "personaje" al jugador controlado por dicha estrategia, y "enemigo" u "oponente" al jugador contrario (controlado por el usuario en la partida monojugador, aunque también podría ser controlado por otra estrategia distinta).

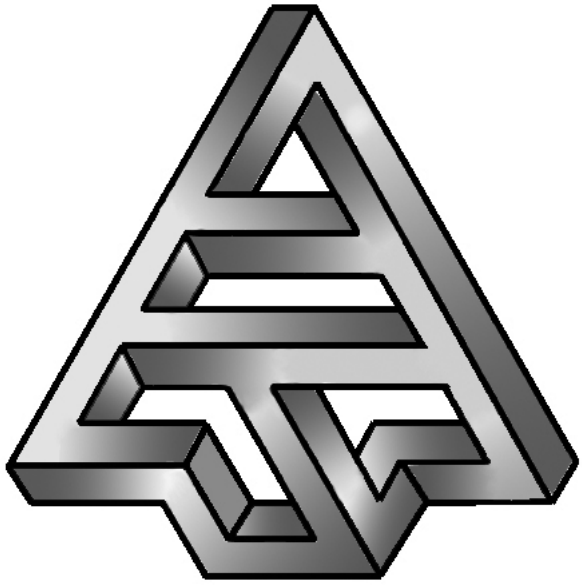
Los indicadores utilizados en la máquina de estados son los siguientes:

- **enemigo_tiene_los_que_me_faltan**: indica si el oponente (el jugador contrario al que posee esta estrategia, normalmente el humano) tiene los cristales que le faltan al propietario de la estrategia para completar el talismán Vega Solaris.
- **poca_vida**: indica si al personaje (propietario de la estrategia) le queda poca vida y, por tanto, corre peligro de quedar inconsciente si es atacado.
- **tengo_item_comida**: indica si el personaje tiene algún ítem de tipo *comida* que le permita restaurar sus puntos de salud.
- **tengo_todos_los_cristales**: indica si el personaje tiene los cuatro cristales del talismán Vega Solaris.
- **hay_enemigo_con_cristal**: indica si, en la habitación en la que se encuentra actualmente el personaje, se encuentra también su oponente y, además, éste posee algún cristal del talismán Vega Solaris.
- **adyacente_a_enemigo**: indica si el personaje se encuentra adyacente a su enemigo.
- **tengo_proyectiles**: indica si el personaje tiene algún ítem que le permita disparar proyectiles (ítem *hechizo básico* o *arco*).
- **enemigo_inconsciente**: indica si el enemigo del personaje está inconsciente.
- **adyacente_a_alien**: indica si el personaje se encuentra adyacente a un alien.
- **hay_item_interesante**: indica si, en la habitación en la que actualmente se encuentra el personaje, hay algún ítem que le interese recoger. Para determinar qué ítems interesa recoger y cuáles no, se ha establecido un sistema de prioridades, de manera que un ítem se considera interesante si tiene mayor prioridad que alguno de los que posee el personaje. La importancia o prioridad, de mayor a menor, de los ítems, viene dada por la siguiente lista:
    1. Cristales.
    2. Comida.
    3. Espada.
    4. Hechizo básico.
    5. Arco.
    6. Escudo.
    7. Hechizo teletransporte.

8. Hechizo temporal.
9. Hechizo nocturno.

3.2.3. Descripción de los estados

- **Abandonando habitación**: en este estado, el personaje está abandonando la habitación en la que se encuentra actualmente.
- **Acercándose a humano**: el personaje se está acercando al enemigo (normalmente el humano).
- **Atacando a alien**: el personaje está atacando a un alien adyacente. Si el personaje tiene un ítem escudo, lo utiliza para protegerse.
- **Atacando a humano**: el personaje está atacando al personaje enemigo, ya sea cuerpo a cuerpo (con los puños o con la espada) o a distancia (con el hechizo básico o con las flechas). Si tiene un escudo, lo utiliza para protegerse.
- **Buscando cristales**: en este estado el personaje no hace nada, ya que se trata de un estado transitorio. Tan sólo comprueba si puede molestar al enemigo con algún hechizo, como el hechizo nocturno o el temporal. Si posee un ítem de cualquiera de esos dos tipos, lo utiliza.
- **Buscando oponente**: en este estado el personaje busca al oponente para robarle los cristales que tenga.
- **Buscando salida**: en este estado el personaje ya tiene los cuatro cristales, y está buscando la habitación inicial para ganar la partida.
- **Comiendo**: el personaje utiliza un ítem comida.
- **Recogiendo ítems**: el personaje ha visto ítems que le interesan en la habitación en la que se encuentra, y está acercándose y/o cogiéndolos.
- **Robando cristales**: el personaje está al lado de su enemigo, y éste está inconsciente, además de tener algún cristal. En este estado, el personaje roba al enemigo los cristales que posea.



Vega Solaris

Formato de los Archivos de Datos

Versión 1.0
04/07/2006

ÍNDICE

1. PREÁMBULO

1.1. Introducción

Este documento describe de forma concisa la función de cada uno de los archivos de datos de la aplicación Vega Solaris, así como su formato.

Los archivos de datos son los archivos que la aplicación utiliza para saber cómo manipular las imágenes de las animaciones, el texto de las pantallas, las colisiones de los objetos gráficos, etc. Estos archivos tienen un formato específicamente diseñado para la aplicación. Si se modifica algún archivo sin seguir exactamente las pautas marcadas en el presente documento, la aplicación fallará.

1.2. Notación

Para explicar el formato de los archivos, se utilizará la siguiente notación:

- <*contenido de la línea*>: indica el uso o significado del texto que se coloque en esa línea.
- <<<*contenido del texto*>>>: similar al anterior, pero para texto que ocupa varias líneas (sin número fijo).
- <<*contenido del texto*>>: listado de una indeterminada cantidad de atributos, separados por punto y coma. Escrito en una única linea.
- <*contenido de la línea*>=*"valor"*: indica el uso o significado del texto que se coloque en esa línea, pero además indica que necesariamente esa línea debe ser igual a "valor". Se utiliza para explicar el significado de las líneas del archivo cuyo contenido no puede ser modificado (o se producirán fallos en el programa). Si "valor" es un intervalo, se indicará como *[inicio .. fin]*.

1.3. Códigos de imagenes

Las imágenes en la mayoría de los ficheros son referenciados mediante un código. Este código identifica de forma unívoca cada imagen. La correspondencia entre este código imagen y la propia imagen está especificada en el fichero imagenes.vs

1.4. Comentarios en los archivos

Se pueden incluir comentarios en todos los archivos de datos, comenzando la linea que se desea ser comentada con una *almohadilla*. Éstas lineas no se contabilizarán para dar valor a la primera linea de cada fichero.

Además las lineas que se encuentren después de la última línea de datos del archivo (indicada por el valor de la primera linea del fichero) no serán tomadas en cuenta. Esto puede resultar útil para hacer aclaraciones sobre el contenido de los archivos, el significado y formato de las líneas, etc.

2. DESCRIPCIÓN DE LOS ARCHIVOS

2.1. Archivos de las Pantallas de Presentación

Las pantallas de presentación manejan una serie de archivos (con extensión VS) que contienen el texto que se muestra por pantalla en cada pantalla de presentación.

El hecho de que el texto mostrado se encuentre en archivos externos facilita la traducción de los textos del juego a otros idiomas.

Los archivos de las pantallas de presentación se encuentran en el directorio **/res/datos/pantallas** y tienen extensión **vs** (de *Vega Solaris*). Estos archivos son:

- *ayuda.vs*: contiene el texto que se mostrará en la pantalla de ayuda, que indica muy brevemente cómo jugar a Vega Solaris.
- *creditos.vs*: contiene los créditos del juego, en los que aparecen los creadores y otra información interesante.
- *fin.vs*: contiene el texto utilizado para indicar, al final de una partida, los puntos de cada jugador y quién ha sido el ganador. También contiene el texto utilizado para pedir al usuario que introduzca su nombre (en caso de haber conseguido una puntuación máxima).
- *inicio.vs*: contiene el texto de las opciones del menú principal.
- *puntuaciones.vs*: contiene el texto que se desea mostrar tras la máximas puntuaciones.
- *textoFinTiempo.vs*: contiene el texto mostrado cuando la partida finaliza por haberse agotado el tiempo.
- *textoTalismanGanador.vs*: contiene el texto mostrado cuando la partida finaliza por haber reunido, el propio jugador, todos los cristales del talismán Vega Solaris.
- *textoTalismanPerdedor.vs*: contiene el texto mostrado cuando la partida finaliza por haber reunido, el oponente, todos los cristales del talismán Vega Solaris.
- *juego_carga.vs*: las rutas de las imágenes que que conforman la pantalla de carga.
- *menus.vs*: contiene el texto que se muestra en ciertos botones del juego.
- *multijugador.vs*: contiene el texto mostrado en los mensajes de la parte de conexión multijugador.

2.2. Archivos del Mapa

Estos archivos definen la estructura del mapa o escenario en el que tiene lugar la acción del juego. Estos archivos se encuentran en el directorio **/res/datos/mapa**, y son los siguientes:

- *mapa.vs*: describe distintos aspectos sobre el mapa, como son el número de habitaciones y el nombre de los archivos de descripción de las habitaciones.
- *habitaciones/libres/habXXYY.vs*: describe las zonas de la habitación que pueden ser ocupadas de forma aleatoria por items o aliens.

- *habitaciones/objetos/habXXYY.vs*: describe los elementos que se deben representar gráficamente en la habitación. Para el fondo, se pueden utilizar texturas (pequeñas imágenes que se mostrarán repetidas cubriendo todo el fondo) o imágenes completas (deberán tener el mismo tamaño que la habitación).
- *habitaciones/puertas/habXXYY.vs*: describe las puertas que tiene la habitación.

Los archivos de descripción de las habitaciones se nombran utilizando un número de cuatro cifras. Este número indica la posición absoluta de la habitación en el mapa, de manera que las dos primeras cifras representan la coordenada horizontal y las dos últimas representan la coordenada vertical, siendo el origen de coordenadas la esquina superior izquierda del mapa. Teniendo en cuenta esto, un mapa puede contener como máximo 10000 habitaciones.

El rango de las XX y las YY es de 0 a 99, ambos incluídos.

2.3. Archivos de los Personajes y Objetos Animados

Cada personaje (jugadores y aliens) y cada proyectil tiene un archivo de datos asociado, que describe las imágenes que deben utilizarse (y en qué orden y situación) para dotar al objeto de animación. Estos archivos se encuentran en el directorio **/res/datos/personajes**, y son:

- *humano.vs*: describe los archivos de imágenes utilizados para realizar la animación del personaje humano.
- *extraterrestre.vs*: describe los archivos de imágenes utilizados para realizar la animación del personaje extraterrestre.
- *aliens/<alien>.vs*: describe los archivos de imágenes utilizados para realizar la animación del alien indicado.

Dado que los objetos del decorado y los ítems no estarán animados, no es necesaria la utilización de este tipo de archivos de datos para estos objetos.

2.4 Otros archivos

Aparte, existen otros ficheros con datos variados pero que no por ello tienen menor importancia. Esos archivos están ubicados en **/res/datos/.**

- *aliens.vs*: indica que tipos de aliens hay y que estrategia tiene cada uno de ellos.
- *items.vs:* indica qué tipo de items hay en el mapa, y para cada uno de ellos, indica su configuración.
- *panel.vs*: indica las imágenes utilizadas para la composición del panel de estado.
- *global.vs*: fichero con información variada de datos del juego.
- *flecha.vs*: fichero que especifica las imágenes que conforman las flechas lanzadas por el arco.

- *hechizoBasico.vs*: fichero que especifica las imágenes que conforman la animación de la bola de hechizo básico.
- *imagenes.vs*: fichero que relaciona los códigos de imágenes con su ruta real.

3. FORMATO DE LOS ARCHIVOS

3.1. Archivos de las Pantallas de Presentación

3.1.1. ayuda.vs

<nº de líneas de texto>

<<<texto>>>

3.1.2. creditos.vs

<nº de líneas de texto>

<<<texto>>>

3.1.3. fin.vs

<nº de líneas de texto>=4

<texto para indicar los puntos del personaje humano>

<texto para indicar los puntos del personaje extraterrestre>

<texto “Pulsar una tecla para continuar”>

<texto para pedir el nombre al jugador>

3.1.4. inicio.vs

<nº de opciones del menú>=6

<opción “Iniciar partida para 1 jugador>

<opción “Iniciar partida para 2 jugadores>

<opción “Unirse a partida para 2 jugadores>

<opción “Ver máximas puntuaciones”>

<opción “Créditos”>

<opción “Ayuda”>

3.1.5. puntuaciones.vs

<nº de líneas de texto>

<<<texto “Pulsar una tecla para continuar”>>>

3.1.6. textoFinTiempo.vs

<nº de líneas de texto>

<<<texto para fin de partida por tiempo agotado>>>

3.1.7. textoTalismanGanador.vs

<nº de líneas de texto>

<<<texto para fin de partida por objetivo logrado por el propio jugador>>>

3.1.8. textoTalismanPerdedor.vs

<nº de líneas de texto>

<<<texto para fin de partida por objetivo logrado por el oponente>>>

3.1.9. juego_carga.vs

<nº de líneas de texto>=2

<ruta de la imagen del texto de la pantalla de carga>

<<rutas de las imágenes que conforman la pantalla de carga>>

3.1.10. menus.vs

<nº de líneas de texto>=4

<texto del boton cerrar aplicacion>

<texto del boton aceptar (menu principal)>

<texto del boton fin de partida (pantalla de juego)>

<texto del boton pausa/continuar>

3.1.11. multijugador.vs

<nº de líneas de texto>=4

<Título de las pantallas de espera de conexión (servidor y cliente)>

<texto mostrado durante la espera a que se conecte un cliente (sólo servidor) >

<texto mostrado durante la búsqueda de un servidor (sólo cliente) >

<titulo del mensaje de error cuando se producen fallos en la conexión (servidor y cliente)>;<cuerpo del mensaje de error cuando se producen fallos en la conexión (servidor y cliente) >

3.2. Archivos del Mapa

3.2.1. mapa.vs

<nº de habitaciones + 1>

<ancho del mapa> <alto del mapa>

<coordenadas habitación 1 (habitación inicial)>

<coordenadas habitación 2>

…

<coordenadas habitación N>

Las coordenadas de cada habitación representan su posición relativa en una matriz de habitaciones, en la que la habitación de coordenadas (0,0) sería la habitación superior izquierda. Las coordenadas de las habitaciones son enteros en el rango [0,99].

La primera habitación indicada se considera la habitación de partida, en la que aparecerán los personajes al comenzar la partida. Es también la habitación a la que los jugadores deberán acudir con los cuatro cristales para ganar la partida.

3.2.2. habXXYY.vs (libres)

<nº celdas>

<posX casilla1> <posY casilla1>

<posX casilla2> <posY casilla2>

….

<posX casillaN> <posY casillaN>

Para representar las habitaciones, éstas se dividen en una matriz de 15 x 15 celdas de 16 x 16 píxeles cada una. El tamaño de las habitaciones es, por tanto, de 240 x 240 píxeles.

El contenido de este archivo es un listado de casillas sobre las que se pueden ubicar de forma aleatoria aliens e items en la habitación. El espíritu de este fichero no es que contenga todas las casillas libres de la habitación, sino simplemente unas pocas. Esto es debido a los dos motivos que llevaron a la creación de este fichero: la eficiencia a la hora de ubicar de forma aleatoria elementos sobre la habitación y poder elegir a priori un conjunto de posibles ubicaciones para los elementos. Dependiendo de la habitación, se recomienda crear un listado cuya longitud esté comprendida entre 3 y 6 elementos.

3.2.3. habXXYY.vs (objetos)

<nº objetos del decorado en la habitación>

<código de imagen de fondo para la habitación> <¿es textura (0) o imagen completa (1)?>=[0..1]

<posX> <posY> <código de imagen para el objeto 1>

<posX> <posY> <código de imagen para el objeto 2>

…

<posX> <posY> <código de imagen para el objeto N>

La segunda línea del archivo indica la imagen que se utilizará para dibujar el fondo de la habitación. Si es una textura, dicha imagen puede tener cualquier tamaño, siempre que el alto y el ancho sean submúltiplos del alto y el ancho de la habitación, respectivamente. Las texturas se dibujarán en mosaico, es decir, se dibujan muchas copias de la imagen de manera que cubran todo el fondo de la habitación. Para indicar que la imagen es una textura, se indicará en la segunda línea del archivo con un 0 detrás del nombre de la imagen, separando ambos con un espacio.

Si la imagen de fondo es una imagen completa, dicha imagen deberá tener las mismas dimensiones que la habitación, o no cubrirá todo el fondo. Para indicar que la imagen es una imagen completa, se indicará en la segunda línea del archivo con un 1 detrás del nombre de la imagen, separando ambos con un espacio.

En caso de este linea esté vacía, la imagen de fondo utilizada para la habitación, será aquella que se especifique en el fichero *global.vs* (especificado en el punto 3.4.4).

En cuanto a las coordenadas de cada objeto, los valores representan la celda en la que el objeto está situado. Si el objeto ocupa varias celdas, se indicará la celda superior izquierda. Las coordenadas de las celdas están en el rango [0..N-1], siendo N el número de celdas de ancho/alto de una habitación.

3.2.4. habXXYY.vs (puertas)

<nº máximo de puertas>=4

<posX puerta superior> <posY puerta superior>

<posX puerta derecha> <posY puerta derecha>

<posX puerta inferior> <posY puerta inferior>

<posX puerta izquierda> <posY puerta izquierda>

Cuando una habitación carezca de alguna puerta, la linea correspondiente a dicha puerta quedará en blanco.

3.3. Archivos de los Personajes y Objetos Animados

3.3.1. humano.vs

<nº movimientos distintos>=9

<<códigos de imagen para formar la animación *andar hacia la derecha*>>

<<códigos de imagen para formar la animación *andar hacia la arriba*>>

<<códigos de imagen para formar la animación *andar hacia la izquierrda*>>

<<códigos de imagen para formar la animación *andar hacia la abajo*>>

<<códigos de imagen para formar la animación *estado inconsciente*>>

<<códigos de imagen para formar la animación *con espada mirando hacia la derecha*>>

<<códigos de imagen para formar la animación *con espada mirando hacia la izquierda*>>

<<códigos de imagen para formar la animación *puñetazo mirando hacia la izquierda*>>

<<códigos de imagen para formar la animación *puñetazo mirando hacia la derecha*>>

<<códigos de imagen para formar la animación del *hechizo nocturno*>>

<<códigos de imagen para formar la animación del *estado invencible*>>

<<códigos de imagen para formar la animación del *estado ralentizado*>>

<<códigos de imagen para formar la animación del *estado paralizado*>>

3.3.2. extraterrestre.vs

<nº movimientos distintos>=9

<<códigos de imagen para formar la animación *andar hacia la derecha*>>

<<códigos de imagen para formar la animación *andar hacia la arriba*>>

<<códigos de imagen para formar la animación *andar hacia la izquierrda*>>

<<códigos de imagen para formar la animación *andar hacia la abajo*>>

<<códigos de imagen para formar la animación *estado inconsciente*>>

<<códigos de imagen para formar la animación *con espada mirando hacia la derecha*>>

<<códigos de imagen para formar la animación *con espada mirando hacia la izquierda*>>

<<códigos de imagen para formar la animación *puñetazo mirando hacia la izquierda*>>

<<códigos de imagen para formar la animación *puñetazo mirando hacia la derecha*>>

<<códigos de imagen para formar la animación del *hechizo nocturno*>>

<<códigos de imagen para formar la animación del *estado invencible*>>

<<códigos de imagen para formar la animación del *estado ralentizado*>>

<<códigos de imagen para formar la animación del *estado paralizado*>>

3.3.3. <alien>.vs

<nº animaciones distintas>=3

<<códigos de imagen para el movimiento normal>>

<<códigos de imagen para el nacimiento>>

<<códigos de imagen para la muerte>>

3.4 Otros Archivos

3.4.1 aliens.vs

<nº de aliens distintos>

<nombre de estrategia alien1>] <código de imagen para alien1>

…

<nombre de estrategia alienN> <código de imagen para alienN>

La correspondencia de estas estrategias con su implementación está explicada en el documento “Diseño de la Inteligencia Artificial”.

3.4.2 items.vs

<nº de items distintos>

<atributo1 item1> <atributo2 item1> <atributo3 item1> <atributo4 item1> <atributo5 item1> <atributo6 item1> <atributo7 item1>

….

<atributo1 itemN> <atributo2 itemN> <atributo3 itemN> <atributo4 itemN> <atributo5 itemN> <atributo6 itemN> <atributo7 itemN>

1er atributo → tipo de item

2º atributo → código de la imagen asociada

3º atributo → nº de elementos de ese tipo que hay en el mapa

*** Los siguientes valores sólo son para COMIDA, ARCO, BASICO y TELETRANSPORTE

4º atributo → cantidad de usos inicial

5º atributo → cantidad del ítem consumida por cada uso

6º atributo → cantidad de vida/daño/tiempo por cada uso

7º atributo (sólo ARCO y BÁSICO)-> fichero que describe las animaciones de los proyectiles del arco y el hechizo basico

Los posibles valores para el tipo de item son: “Cristal”, “Espada”, “Arco, “Basico”, “Teletransporte”, “Comida”, “Escudo”, “Nocturno”, “Temporal”.

3.4.3 panel.vs

<nº de imágenes referidas>=5

<código de imagen del fondo del panel>

<código de imagen de la accion “robar”>

<código de imagen de la accion “soltar”>

<código de imagen de la accion “golpear”>

<código de imagen de la accion “cambiar de bolsillo”>

3.4.4 global.vs

<nº de datos indicados>=3

<código de imagen de fondo para las habitaciones> <¿es textura (0) o imagen completa (1)?>=[0..1]

<archivo que referencia las imágenes del panel de control>

<texto que aparece sobreimpresionado cuando el juego está pausado>

En caso de que la linea de la imagen de fondo aparezca vacía, se tomará como valor por defecto un fondo negro. El tipo y formato de la información de esta linea es la misma que la especificada en el punto 3.2.3.

3.4.5 flecha.vs

<nº de animaciones diferentes>=8

<<códigos de imágenes de la animación moverse hacia *arriba*>>

<<códigos de imágenes de la animación moverse hacia *abajo*>>

<<códigos de imágenes de la animación moverse hacia *izquierda*>>

<<códigos de imágenes de la animación moverse hacia *derecha*>>

<<códigos de imágenes de la animación moverse hacia *arriba-izquierda*>>

<<códigos de imágenes de la animación moverse hacia *arriba-derecha*>>

<<códigos de imágenes de la animación moverse hacia *abajo-izquierda*>>

<<códigos de imágenes de la animación moverse hacia *abajo-derecha*>>

3.4.6 hechizoBasico.vs

<nº de animaciones diferentes>=2

<<códigos de imágenes de la animación *desplazar bola*>>

<<códigos de imágenes de la animación *explosión de la bola*>>

3.4.7 imagenes.vs

<nº de imágenes de la aplicación>

<código de imagen1> <ruta fisica en la que se encuentra el fichero de la imagen>

…

<código de imagenN> <ruta fisica en la que se encuentra el fichero de la imagen>

Cada imagen tiene un código único que lo diferencia del resto de imágenes.

Todos los códigos de imagen que se referencien en cualquier fichero de texto deberán aparecer en este fichero. A su vez, cualquier imagen que aparezca en este fichero deberá existir físicamente

4. CONSIDERACIONES ACERCA DE LOS ARCHIVOS DE IMÁGENES

4.1. Formato

El formato elegido para los archivos de imágenes es el formato PNG (Portable Network Graphics) sin entrelazado. La razón de haber elegido este formato es que la mayoría de los teléfonos móviles con soporte para juegos Java pueden leerlo, mientras que otros formatos de archivos no están tan extendidos en este tipo de terminales. Además, el formato PNG soporta la representación de transparencias en las imágenes, con lo que no es necesaria la gestión de máscaras para las imágenes del juego.

4.2. Dimensiones

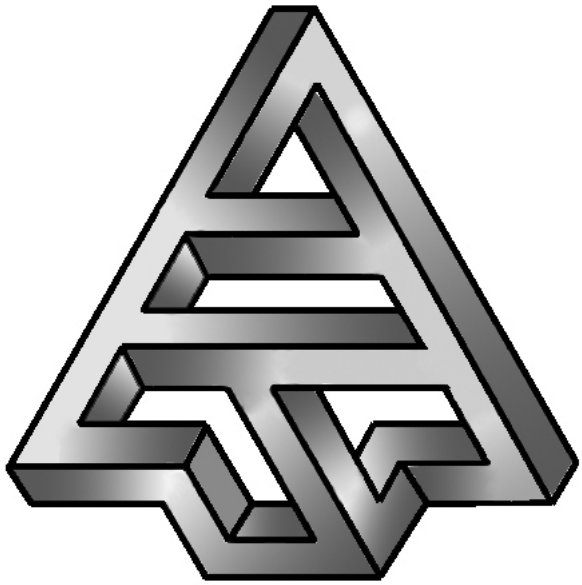
Las dimensiones de los archivos de imágenes varían dependiendo de su uso:

- *Fondos de las pantallas de presentación*: 150 x 150 píxeles.
- *Fondos de las habitaciones (imágenes)*: 120 x 120 píxeles.
- *Fondos de las habitaciones (texturas)*: cualquier tamaño cuyo ancho y alto sean múltiplos de 8.
- *Objetos del decorado*: tanto el ancho como el alto deben ser múltiplos de 8 píxeles.
- *Ítems*: 8x8 píxeles.
- *Personajes jugadores*: no hay restricciones sobre su tamaño, aunque se recomienda que el ancho y el alto sean cercanos a 16 píxeles para garantizar una visualización adecuada.
- *Enemigos*: no hay restricciones sobre su tamaño, aunque se recomienda que el ancho y el alto sean cercanos a 16 píxeles para garantizar una visualización adecuada.
- *Proyectiles*: no hay restricciones sobre su tamaño, aunque se recomienda un tamaño no superior a 16 x 16 píxeles, para garantizar una visualización adecuada.

4.3. Otras Consideraciones

Es recomendable que, en los archivos de imágenes, el objeto o personaje representado ocupe el máximo área posible, quedando alrededor del mismo el mínimo espacio posible. De este modo, al calcular las colisiones de los objetos con los personajes, la precisión será mayor y no se visualizarán efectos extraños (e.g. un personaje que no puede avanzar porque se ha encontrado con una barrera invisible, que no es más que el borde no ocupado de un objeto del decorado). Además, se conseguirá un efecto visual más agradable para las animaciones y movimientos de cámara. En caso de que la imagen sea relativamente grande, y por su forma, el objeto o persanje representado no se pueda ajustar a la imagen, una recomendación es dividir el objeto o persanje en cuestión en varias imágenes.

También es recomendable que para que el efecto visual de los objetos y personajes sobre el fondo sea lo más agradable posible, la parte de la imagen no rellena con el objeto o personaje sea rellenada con color transparente.



Vega Solaris

Implementación – Traducción de J2SE a J2ME

Versión 1.0
04/07/2006

ÍNDICE

1. PREÁMBULO

1.1. Introducción

Este documento pretende dar algunos consejos sobre el proceso de traducción de una aplicación escrita en J2SE al lenguaje J2ME. Estas recomendaciones también son válidas para un programador habituadoa programar en J2SE y que quiere comenzar a desarrollar sus primeros programas en J2ME.

Para ello, se anunciarán cambios sistemáticos que deberá realizar el programador debido al cambio de lenguaje y se darán otras recomendaciones prácticas.

2. CAMBIOS SISTEMÁTICOS

2.1. Importación de clases

- Si consideramos el los entornos de programación J2SE Y J2ME, estamos trabajando con dos API's diferentes, por lo que lógicamente, el uso e importación de ciertas clases varía.
- Por ejemplo en J2ME no existe la clase *AbstractCollection*, ni lógicamente ninguna que herede de ella, por lo que si deseamos trabajar con estructuras de datos que manejen memoria dinámica, deberemos implementarlas por nosotros mismos.
- A la hora de crear formularios, alertas o mensajes al usuario también deberemos tener en cuenta el API específica de J2ME. ya no podremos utilizar las bibliotecas *Swing* o *AWT*. Las clases que podremos utilizar son *Alert, Form, List* y *TextBox*.
- Cuando se desee crear pantallas debajo nivel, es decir, pantallas en las que queremos dejar a nuestro propio gusto todos los detalles de las mismas, al no poder usar el paquete *AWT* ni *Swing*, ya no podremos realizar las siguientes típicas imprtaciones de clases para este fin:
    - java.awt.Color
    - java.awt.Graphics
    - java.awt.Image
    - javax.swing.JFrame
    - java.awt.Canvas

  En su lugar, deberemos realizar las siguientes importaciones:
    - javax.microedition.lcdui.Graphics
    - javax.microedition.lcdui.Image
    - javax.microedition.lcdui.Display
    - javax.microedition.lcdui.Canvas

  La clase Color desaparece definitivamente, por lo que lo códigos de color deberemos indicarlos manualmente.

2.2. Split

Desde la version 1.4.2 de J2SE existe un método muy cómodo en la clase *String* para dividir una cadena de caracteres en varias cadenas mediante un separador.

Pues bien, este meétodo en J2ME desaparece, por lo que si en algún momento deseamos dividir una cadena, deberemos programanos nuestro propio método.

2.3. ImageObserver

En J2SE existe el interfaz *ImageObserver*, necesario para indicar qué objeto gráfico necesita recibir información de la clase *Image*. Un ejemplo habitaul es consultar el ancho o el alto de una imagen.

Pues bien, ya no hay que preocuparse de dicho interfaz en J2ME, ya que desaparece. La información que se desee obtener de un objeto de clase Image, no tendrá problemas de acceso concurrente a pesar de la ausencia delcitado interfaz.

2.4 Lectura y escritura de ficheros de texto

- La lectura de ficheros entre J2ME y J2SE también varía. En J2ME la *java.io* no nos facilita la lectura de ficheros de texto como en J2SE. En su lugar, las lecturas de ficheros de texto se realizan a muy bajo nivel, manejando los ficheros de texto como un conjunto de bytes e implementando nuesros propios buffers para realizar lecturas poco costosas.
- La escritura de ficheros de texto no es posible. En su lugar, existe el llamado *RecordStore*. Consiste en salvar la información que deseemos en la propia memoria del teléfono móvil en lugar de los ficheros de texto. Sobre los *RecordStore* se permite realizar accesos de escritura y de lectura. La biblioteca correspondiente es *javax.microedition.rms*.

3. CONSEJOS PRÁCTICOS

3.1. Carpeta "res"

- En J2SE es habitual colocar todos los ficheros de imagen, texto, etc bajo una carpeta denominada *res*. Debido a ello, todos los accesos a dichos ficheros deberán incluir en su ruta la palabra "res".
- J2ME supone que todos los ficheros de este tipo están bajo la carpeta "res", que automáticamente se genera al crear un proyecto nuevo. Por esto, los accesos a los ficheros ubicados bajo esta carpeta no deberán incluir en su ruta la palabra "res".
- Si estamos migrando una aplicación de J2SE a J2ME y tenemos un referencias a ficheros de esta carpeta que no queremos modificar, una solución es en el proyecto J2ME crear bajo la carpeta "res" una subcarpeta llamada también "res", asi las rutas de acceso a los ficheros quedarán iguales por lo que no habrá necesidad de modificarlas.
- Otra advertencia sobre la lectura de ficheros es que cuando deseemos acceder a un fichero de texto, en J2ME las rutas de dichos ficheros son sensibles a mayúsculas, mientras que en J2SE no lo son. Por lo tanto habrá que cerciorarse de que estamos accediendo al fichero deseado con la ruta exacta y fijándonos bien en las mayúsculas y minúsculas.

3.2. Posición del teclado

- Un pequeño detalle que no tiene mayor importancia es la distribución de teclas en el teclado. La distribución de las teclas en los teléfonos móviles y en el teclado numérico de un teclado de PC es la siguiente:

1	2	3		7	8	9	
4	5	6		4	5	6	
7	8	9		1	2	3	
*	0	#		0		.	
Teléfonos móviles				Teclado numérico PC			

Como puede verse, las posiciones de los números están invertidas verticalmente, por ello, habrá que teener cuidado a la hora de programar eventos de teclado cuando lo que se desee sea subir o bajar algún componente que parezca en la pantalla.

3.3. Strings

- Otro aspecto por el que debe proecuparse un programador de J2ME son los Strings. En J2SE es muy fácil tener almacenardos en memoria largas listas de

Strings y realizar comparaciones sobre ellas, sin que por ello, el rendimiento global de la aplicación se vea afectado.

- Sin embargo realizar este tipo de prácticas en J2ME penalizan duramente el rendimiento de la aplicación, afectando también al tamaño disponible en la memoria de los teléfonos móviles, existiendo la posibilidad real de encontrarnos con que nuestra aplicación se cierre por habernos quedado sin memoria.
- Por ello, realizamos dos recomendaciones, que cuando sea posible, se deberán aplicar:
    - En lugar de almacenar un String que usaremos para realizar la búsqueda de un cierto objeto almacenado en una estructura de datos utilizando dicho String como elemento clave, realizar la búsqueda previamente alamcendo un puntero al objeto deseado.
    - En caso de que el procedimiento anterior no sea aplicable, cuando se desee realizar búsquedas sobre estructuras de tipo clave-valor, en lugar de utilizar claves que sean de tipo String, utilizar claves que sean de tipo entero, ya que las comparaciones entre enteros son mucho más rápidas que entre Strings.

Pruebas

Para la realización de las pruebas de sistema, además de los entornos de desarrollo, se han utilizado dos terminales móviles del mercado, ambos de la marca Nokia. A continuación se exponen las características técnicas de ambos terminales, relativas a la instalación y ejecución de juegos Java.

Nokia 6600

Sistema Operativo: Symbian OS v7.0s.

Plataforma de Desarrollo: Series 60 Developer Platform 2.0.

Tecnología Java:

- CLDC 1.0
- Wireless Messaging API (JSR-120)
- Mobile Media API (JSR-135)
- Bluetooth API (JSR-82 No OBEX)
- MIDP 2.0
- Nokia UI API

Formatos de sonido:

- MIDI (poly 24)
- True Tones (WB-AMR)

Pantalla:

- Profundidad de color: 16 bit
- Resolución: 176 x 208 píxels

Descripción física:

- dimensiones: 109 x 58 x 24 mm.
- peso: 125 g.

Memoria:

- Tamaño de Heap: 3 MB
- Memoria compartida para almacenamiento: 6 MB
- Tamaño ilimitado para aplicaciones .JAR

Conectividad a PC:

- Bluetooth

- Infrarrojos

Nokia 6230i

Sistema Operativo: Nokia OS

Plataforma de Desarrollo: Series 40 Developer Platform 2.0.

Tecnología Java:

- CLDC 1.1
- Wireless Messaging API (JSR-120)
- Mobile Media API (JSR-135)
- Mobile 3D Graphics API (JSR-184)
- JTWI (JSR-185)
- FileConnection and PIM API (JSR-75)
- Bluetooth API (JSR-82 No OBEX)
- MIDP 2.0
- Nokia UI API

Formatos de sonido:

- AAC
- AMR (NB-AMR)
- MIDI Tones (poly 64)
- MP3
- MP4
- True Tones (WB-AMR)

Pantalla:

- Profundidad de color: 16 bit
- Resolución: 208 x 208 píxels

Descripción física:

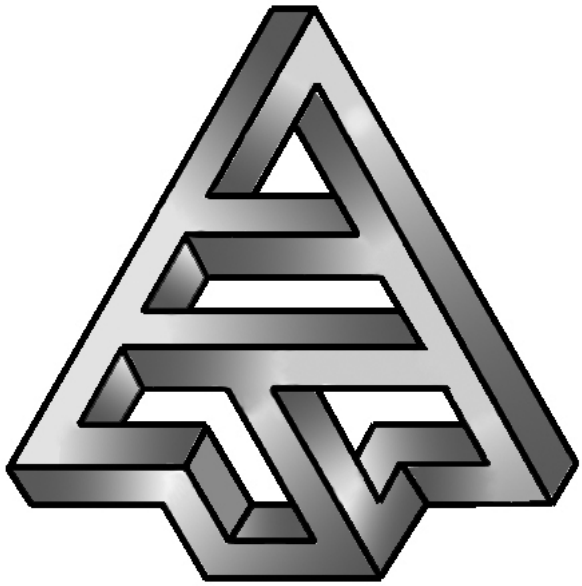
- dimensiones: 103 x 44 x 20 mm.
- peso: 99 g.

Memoria:

- Tamaño de Heap: 2 MB
- Memoria compartida para almacenamiento: 31 MB
- Tamaño limitado para aplicaciones .JAR: 500 KB

Conectividad a PC:

- Bluetooth
- Infrarrojos
- USB



Vega Solaris

Manual de empaquetado y distribución

Versión 1.0
04/07/2006

ÍNDICE

1. INTRODUCCIÓN

El presente documento explica detalladamente el proceso que hay que seguir para, a partir del código fuente de la aplicación, obtener los archivos de la aplicación listos para instalarla y ejecutarla en terminales móviles reales.

Las herramientas utilizadas para realizar el empaquetado de la aplicación han sido:

- J2SE 5.0 (disponible en http://java.sun.com/j2se/1.5.0/).
- J2ME Wireless Toolkit 2.2 (disponible en http://java.sun.com/j2me).

2. EMPAQUETADO Y DISTRIBUCIÓN

2.1. Preliminares

En primer lugar es necesario instalar con éxito, en el equipo en el que se vaya a realizar el empaquetado, el J2ME 5.0 y el J2ME Wireless Toolkit. Una vez hecho esto, hay que copiar el directorio con el código fuente de la aplicación en el directorio *apps* del directorio en el que se haya instalado el J2ME Wireless Toolkit.

La estructura del código fuente de la aplicación Vega Solaris es la siguiente:

VegaSolarisME

\bin: en este directorio se encuentran los archivos necesarios para la distribución de la aplicación.

\lib: librerías externas utilizadas por la aplicación (vacío).

\res: recursos de la aplicación, como archivos de configuración, imágenes, etc.

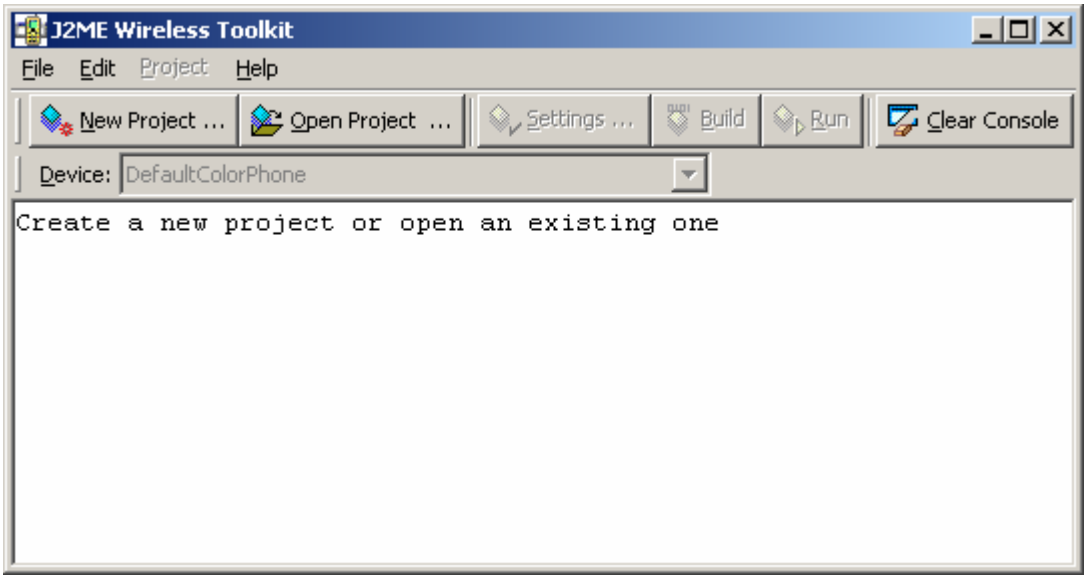
\src: archivos con el código fuente (archivos .java) de la aplicación.

project.properties: archivo que contiene la descripción de las propiedades de la aplicación, necesario para el empaquetado mediante el J2ME Wireless Toolkit.

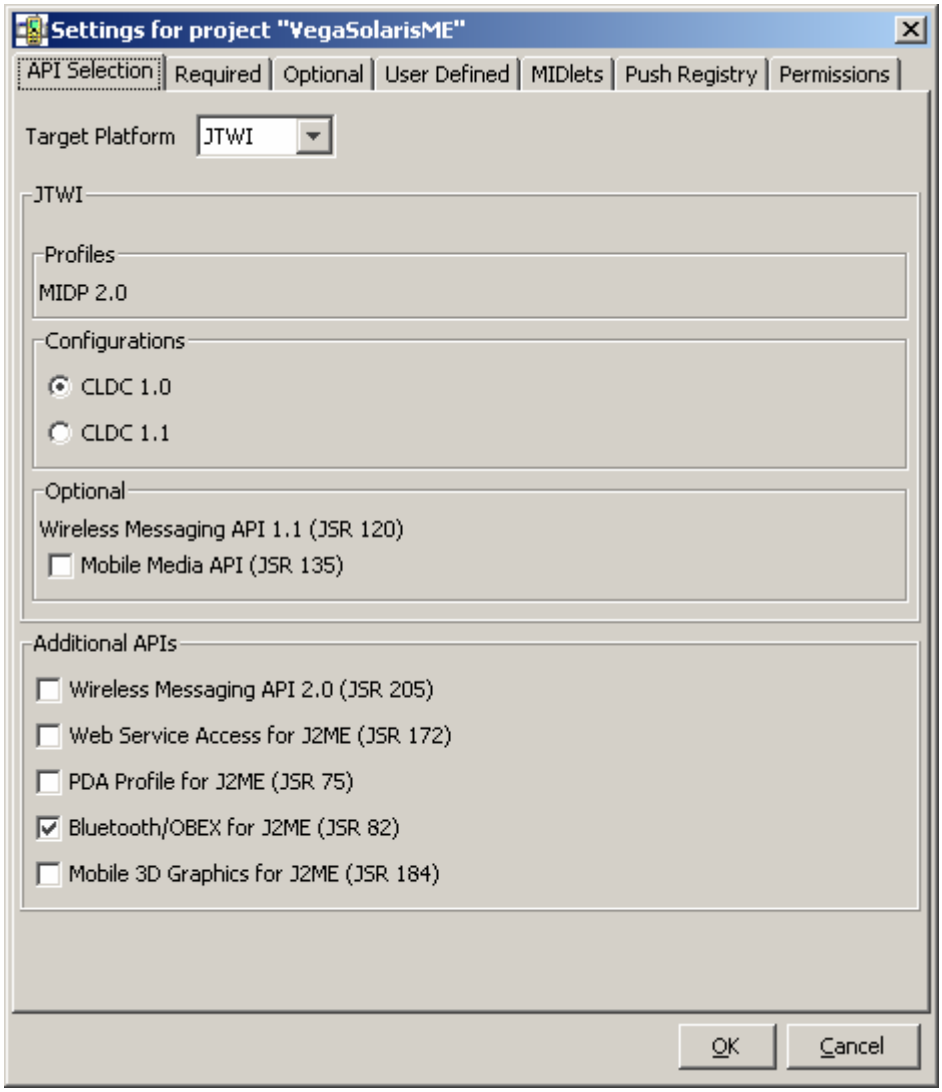
Por tanto, hay que colocar el directorio *VegaSolarisME* en el directorio *apps* de la instalación J2ME Wireless Toolkit.

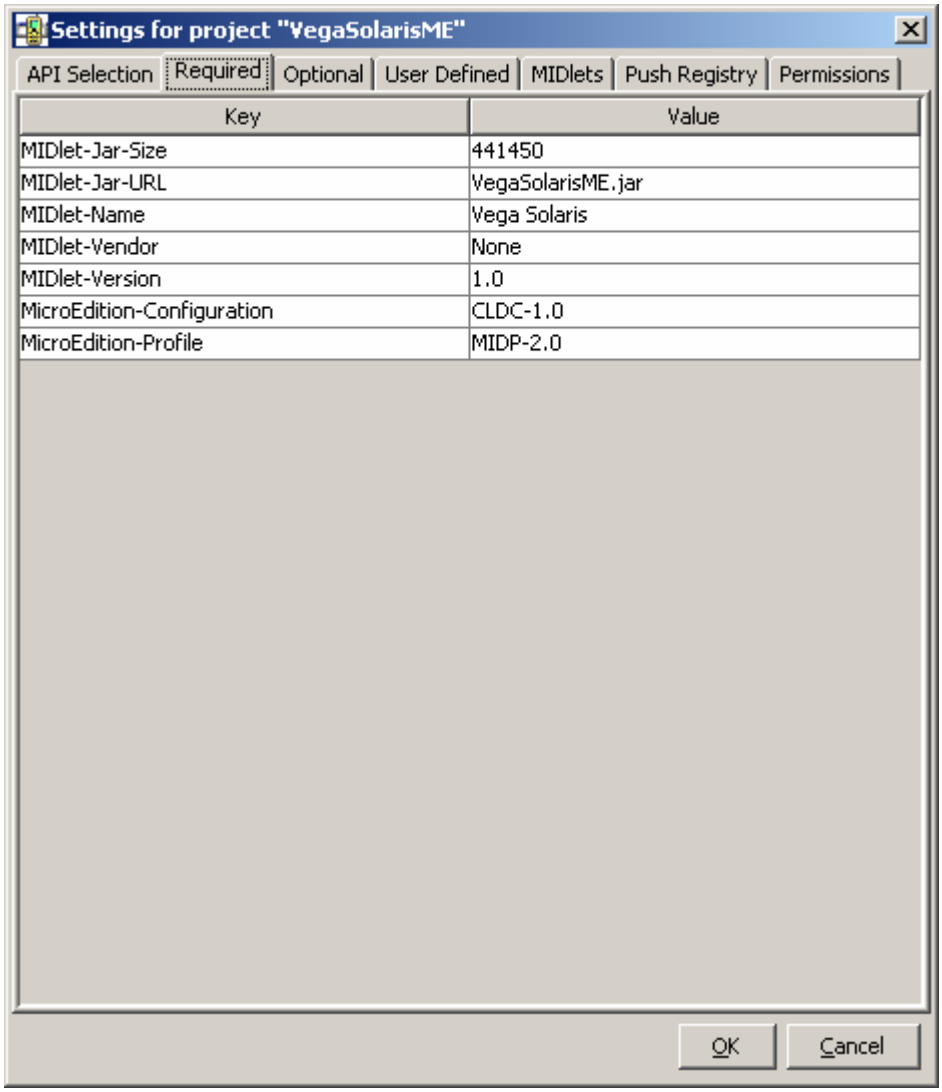
2.2. Configuración del MIDlet

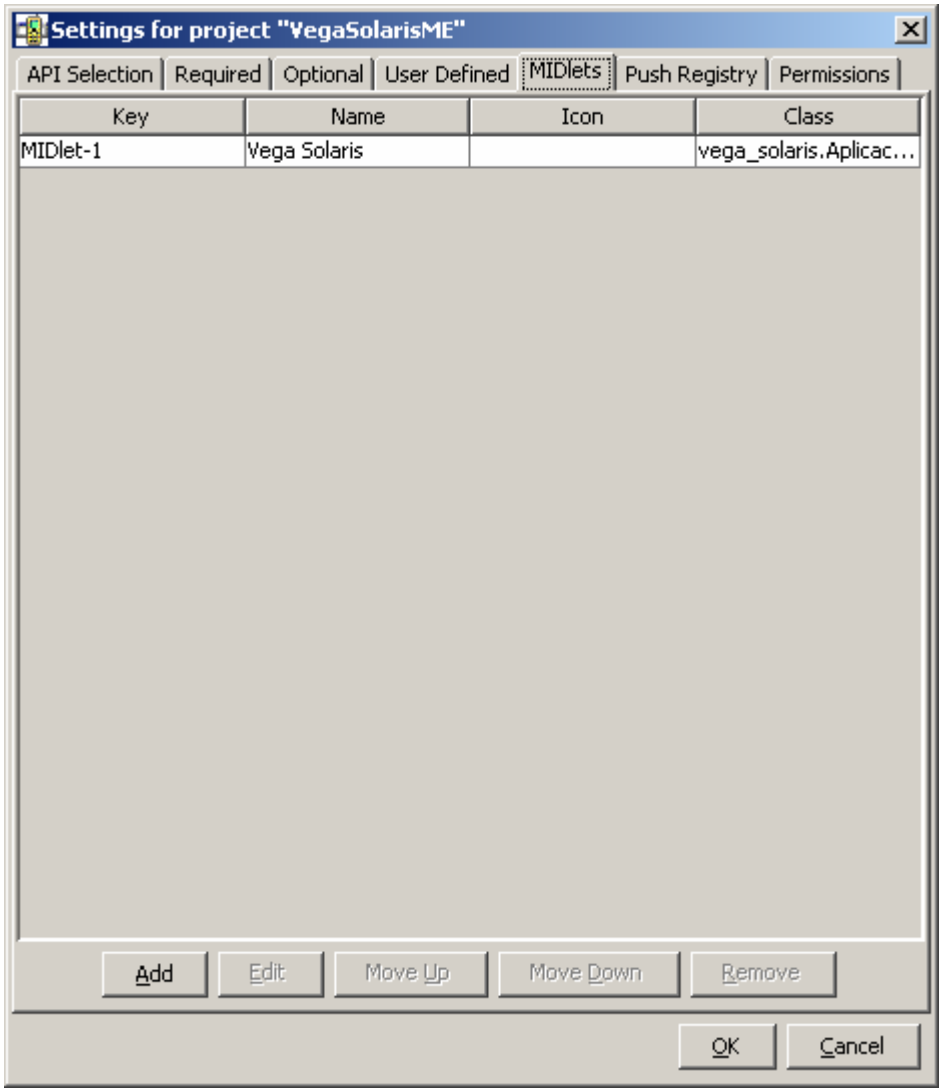
Una vez colocado el código fuente en el directorio *apps*, abrir la utilidad *KToolbar* del J2ME WT. Aparecerá la siguiente ventana:

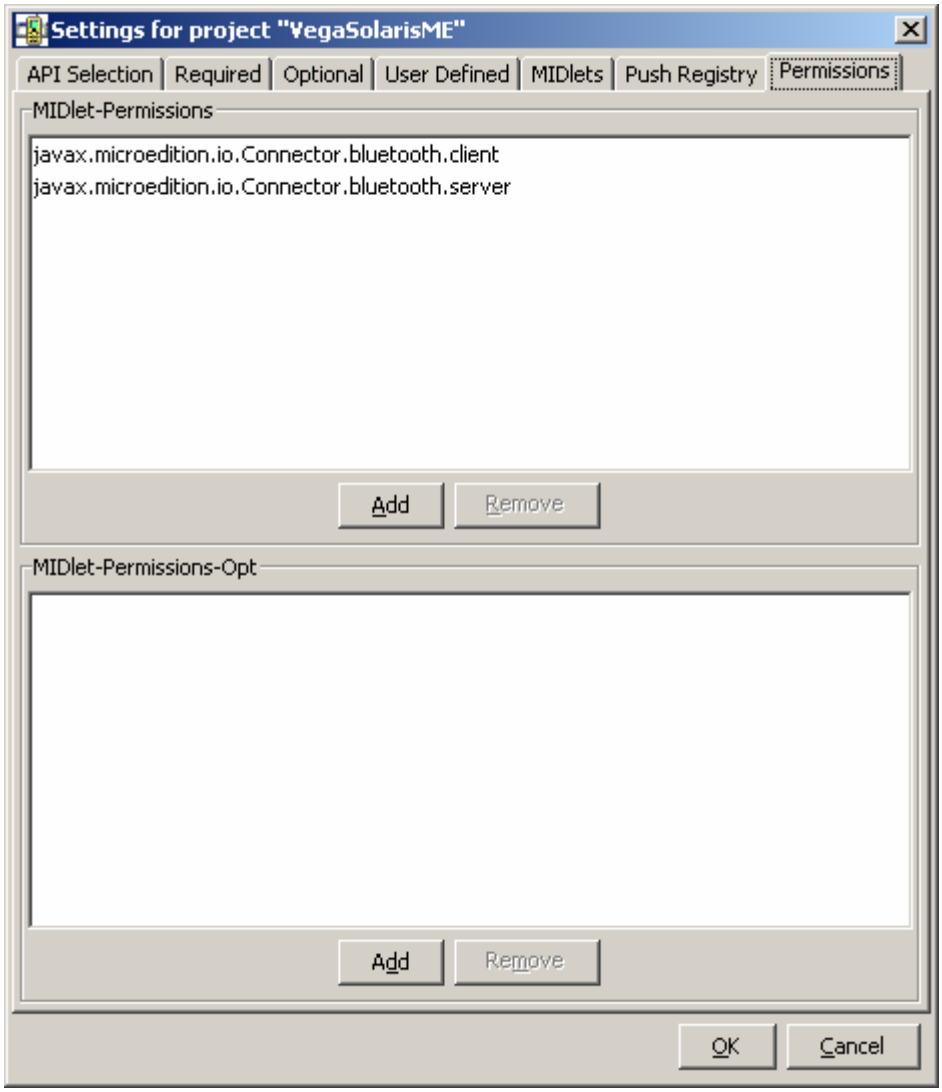


Pulsar el botón *Open Project*, y seleccionar el proyecto VegaSolarisME de la lista que aparece. A continuación, los botones *Settings...*, *Build* y *Run* se activarán. Pulsar el botón *Settings...*, y comprobar que están seleccionadas las siguientes opciones:

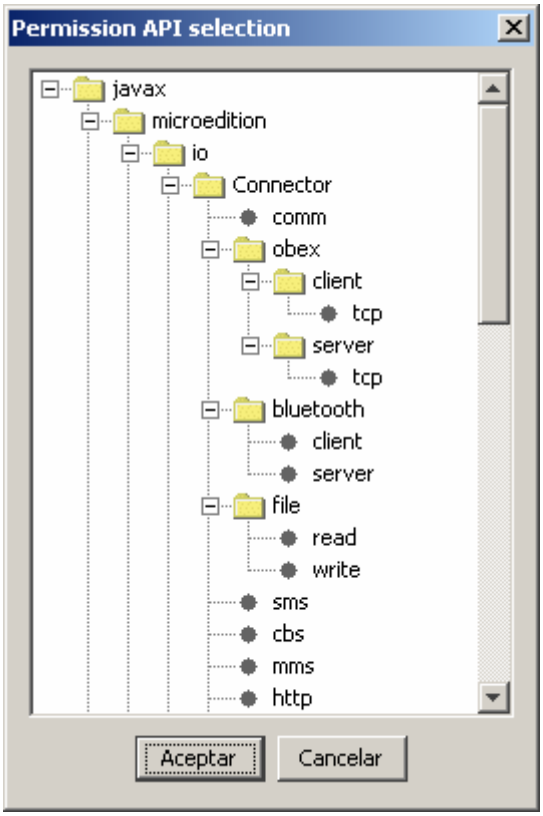

Settings for project "VegaSolarisME"
API Selection
Required
Optional
User Defined
MIDlets
Push Registry
Permissions
Target Platform
JTWI
JTWI
Profiles
MIDP 2.0
Configurations
CLDC 1.0
CLDC 1.1
Optional
Wireless Messaging API 1.1 (JSR 120)
Mobile Media API (JSR 135)
Additional APIs
Wireless Messaging API 2.0 (JSR 205)
Web Service Access for J2ME (JSR 172)
PDA Profile for J2ME (JSR 75)
Bluetooth/OBEX for J2ME (JSR 82)
Mobile 3D Graphics for J2ME (JSR 184)
OK
Cancel


Settings for project "VegaSolarisME"
API Selection
Required
Optional
User Defined
MIDlets
Push Registry
Permissions
Key
Value
MIDlet-Jar-Size
441450
MIDlet-Jar-URL
VegaSolarisME.jar
MIDlet-Name
Vega Solaris
MIDlet-Vendor
None
MIDlet-Version
1.0
MicroEdition-Configuration
CLDC-1.0
MicroEdition-Profile
MIDP-2.0
OK
Cancel


Settings for project "VegaSolarisME"
API Selection
Required
Optional
User Defined
MIDlets
Push Registry
Permissions
Key
Name
Icon
Class
MIDlet-1
Vega Solaris
vega_solaris.Aplicac...
Add
Edit
Move Up
Move Down
Remove
OK
Cancel



En esta última ventana, asegurarse de que están concedidos los permisos de conexión Bluetooth. Para añadir dichos permisos, pulsar el botón *Add* que se encuentra debajo del cuadro de texto *MIDlet-Permissions*. Aparecerá la lista de posibles permisos que se pueden conceder a la aplicación:



Se deben seleccionar los dos permisos siguientes:

- javax.microedition.io.Connector.bluetooth.client
- javax.microedition.io.Connector.bluetooth.server

2.3. Obfuscated package

Una de las utilidades de realizar el prceso de ofuscación es complicar la legibilidad de los programas; pero ésta no es nuestra intención, sino reducir el tamaño del paquete.

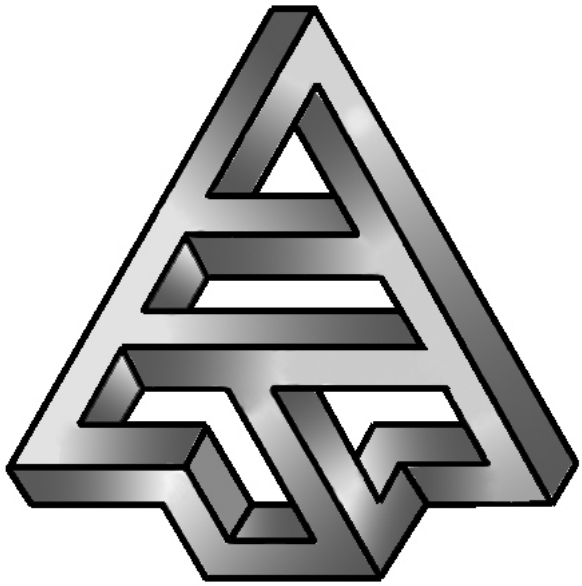
Para su instalación es necesario descargarse desde http://proguard.sourceforge.net el obfuscador.

Posteriormente, será necesario descomprimir el fichero proguard.jar en la carpeta bin del directorio de instalación del J2ME WTK.

2.4. Compilación y empaquetado

Una vez configurado correctamente el MIDlet, pulsar el botón *Build* para compilar el código fuente. Cuando aparezca el mensaje *Build complete*, ya se puede realizar el empaquetado de la aplicación. Para ello, ir al menú *Project* > *Package* > *Create Obfuscated Package*. Cuando aparezca el mensaje *Build complete*, ya estarán listos los archivos de la aplicación para su distribución. Dichos archivos se encuentran en el directorio *VegaSolarisME\bin*, y son:

- VegaSolarisME.jad
- VegaSolarisME.jar



Vega Solaris

Manual de Usuario

Versión 1.0
04/07/2006

ÍNDICE

1. EL JUEGO

El objetivo del juego es conseguir conformar el talismán de Vega Solaris (una figura imposible estilo Escher) a partir de los cuatro cristales en que está fragmentado.

Estos cuatro cristales (figuras también imposibles) están repartidos en un mundo con varios escenarios, que incluyen una selva, cuevas y un templo, y que se reparten entre decenas de pantallas. Se deben llevar hasta la pantalla inicial en un tiempo limitado. El ganador del juego es el personaje (humano o extraterrestre) que haya conseguido llegar a esta pantalla inicial con los cuatro cristales en los bolsillos.

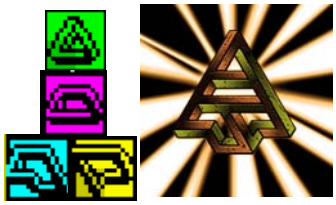
Hay dos personajes en la búsqueda de este talismán: un humano y un extraterrestre a los que se les debería reconocer por su aspecto. Éste es precisamente uno de los alicientes del juego, ya que se puede jugar contra otro jugador o contra la máquina.

En la parte inferior se encuentra información del contenido de los únicos cuatro bolsillos de los personajes en que pueden guardar y llevar los objetos que encuentren a su paso.

La parte izquierda corresponde a nuestro propio personaje mientras que la derecha es la de nuestro adversario.

Durante el recorrido encontrarán no sólo estos cristales, sino también armas que pueden usar contra el otro personaje y contra los bichos que aparecen cada vez más insistentemente. Tanto los personajes como los bichos pueden infligir daños que se representan con el tamaño de la barra de energía. Cuando la barra queda vacía, el personaje debe reponerse, quedando inconsciente durante algún tiempo durante el cual el otro personaje puede robarle los objetos que posea. En total se pueden encontrar los siguientes objetos:

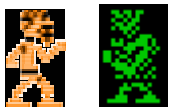
- Cristales del talismán de Vega Solaris. Hay cuatro diferentes que conforman el símbolo completo.



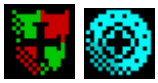
- Armas:
  1. Espadas. Quizás la más llamativa por el modo de lucha.

  2. Arcos. Lanzan flechas a personajes y bichos.

  3. Puñetazos. No hay que olvidar que, aún sin armas, uno puede defenderse de esta guisa.



- Escudos. Proporcionan inmunidad por un tiempo determinado hasta que se agotan. El personaje que lo usa cambia su color intermitentemente a amarillo.



- Conjuros:
  1. Básico. Una bola de luz que se lanza y disminuye la energía de la víctima.

  2. Nocturno. Oculta la visión del otro personaje.

  3. Teletransporte. Permite acudir inmediatamente a la ubicación del otro personaje.

  4. Temporal. Detiene el tiempo salvo para quien lo activa y hace desaparecer a los bichos.

- Comida. Permite recuperar la energía perdida.

Obviamente, el juego debe completarse en un límite de tiempo, que se representa por una calavera dentro de una esfera terrestre que va apareciendo lentamente. Cuando aparece totalmente, finaliza la partida.

2. EL PANEL DE ESTADO

El panel de estado contiene tres secciones principales: la izquierda está dedicada a la información sobre el personaje humano, la central a la indicación del tiempo transcurrido y la derecha a la información sobre el personaje extraterrestre.

2.1. Sección de información del personaje

Energía

En el centro de la sección aparece el indicador de energía, una barra roja que varía de longitud dependiendo de la vida del jugador. El personaje queda inconsciente cuando se agota la energía y durante su inconsciencia la va recuperando poco a poco. Otra forma de recuperar energía es comer.

Bolsillos

Cada uno de los cuatro bolsillos se muestran en el extremo derecho de la sección de información del humano y en el izquierdo en el caso del extraterrestre. Sólo uno de los bolsillos está activo (se indica con fondo naranja).

Iconos

En cada sección de información de los personajes aparecen cuatro iconos que permiten usar, recoger, dejar y robar objetos. Sólo uno de ellos está activo en cada momento (se indica con el color naranja). Al pulsar la tecla Selector, se selecciona el siguiente icono (de izquierda a derecha y de arriba a abajo).

 Selección de bolsillo

Si se selecciona el icono “flechas circulares” a la derecha se puede seleccionar el bolsillo que se desee pulsando las veces que sea necesario Disparo (los bolsillos se seleccionan en el mismo orden que los iconos: de izquierda a derecha y de arriba a abajo). Si estamos pegados al personaje contrario inconsciente, seleccionaremos su bolsillo (esto es útil para robarle).

 Acción

Cuando está seleccionado este icono, al pulsar Disparo se usa el objeto que se tenga en el bolsillo seleccionado. La consecuencia puede ser lanzar una flecha, comer, activar un conjuro, etc., dependiendo del contenido del bolsillo. Si el bolsillo está vacío, el personaje

golpea con un puñetazo. Los bichos se pueden abatir a puñetazos, pero esto consumirá mucha energía. Siempre es mejor usar conjuros o armas.

 Coger/Robar

Cuando está seleccionado este icono, al pulsar Disparo se cogerá el objeto del suelo que esté al lado del personaje. Si el bolsillo ya contenía un objeto, se intercambian.

Este icono también se usa para robar los objetos del otro personaje. Para ello se actúa como si se fuese a coger un objeto del suelo pero pegado al otro personaje, que debe estar inconsciente para poder robarle. En general, para robar hay que: 1) alejarse del contrario para seleccionar el bolsillo propio donde queremos alojar el objeto robado, 2) pegarse a él para seleccionar el bolsillo ajeno y 3) coger el objeto.

 Dejar

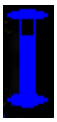
Cuando está seleccionado este icono, al pulsar Disparo se dejará el objeto en el suelo al lado del personaje (siempre que haya espacio para ello).

2.2. Sección de la puntuación

En la parte inferior de cada panel se muestra la puntuación del personaje, que aumenta según se abaten bichos o se hace disminuir la energía del personaje contrario.

2.3. Sección de indicación de tiempo

Entre las secciones de información de los personajes hay un espacio para mostrar una imagen que se va completando conforme avanza el tiempo. La partida termina cuando se vacía la barra o cuando se alcanza el objetivo del juego.



3. CONTROL DE LOS PERSONAJES

Las acciones que pueden realizar los personajes son:

- Avanzar en ocho direcciones: arriba, abajo, izquierda, derecha y las diagonales. Para el movimiento de los personajes se usan las teclas definidas para ello. Para avanzar en diagonal se pulsan simultáneamente dos teclas (por ejemplo, para ir en la diagonal arriba-derecha se pulsarían a la vez las teclas de arriba y derecha).
- Coger objetos (cristales, armas, conjuros y comida). Ésta y las siguientes se describen en la sección "Panel de estado".
- Robar objetos al otro personaje.
- Golpear con los puños al otro personaje y a los bichos.
- Usar las armas y conjuros.
- Comer.

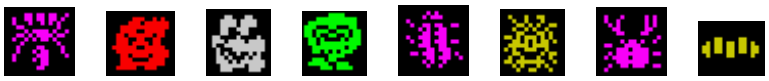
3.1. Teclas predefinidas de control

	Humano
Arriba	KEY_UP
Abajo	KEY_DOWN
Izquierda	KEY LEFT
Derecha	KEY_RIGHT
Selector	GAME_A
Disparo	FIRE
Ver pantalla del enemigo	GAME_B
Pausa	Botón indicado en pantalla
Abandonar	Botón indicado en pantalla

4. ENEMIGOS

El principal enemigo es el contrincante, ya sea humano o extraterrestre, y controlado por otro jugador o por la máquina. No obstante, los bichos son otra serie de enemigos que son principalmente molestos, aunque por su insistencia llegan a aturdir a los personajes. Lo mejor es evitarlos o aniquilarlos con las diferentes armas que se van encontrando a lo largo del mundo. Aunque es posible, no es buena idea tratar de matarlos a puñetazos, porque a cambio se recibirá un daño demasiado elevado.

Uno de los bichos, la libélula (el primer bicho por la izquierda de la serie ilustrada), además de restar energía, provoca un desagradable efecto secundario: hace que el personaje se mueva a la mitad de velocidad. Durante el tiempo que dura esta situación el personaje permanece de color magenta.



5. EL MENÚ INICIAL

El menú del programa aparece al iniciar el juego y sus entradas se describen a continuación:

Nueva partida

Se crea una nueva partida monojugador. Nuestro adversario será la propia máquina.

Crear partida (2J)

Se crea una partida multijugador. A partir de ese momento se espera a que otro usuario se agregue a la partida. Cuando un usuario se agrega, la partida da comienzo.

Unirse a partida creada (2J)

El propio juego buscará alguna partida creada, dándole al usuario la posibilidad de unirse a la que desee para comenzar a jugar.

Récords

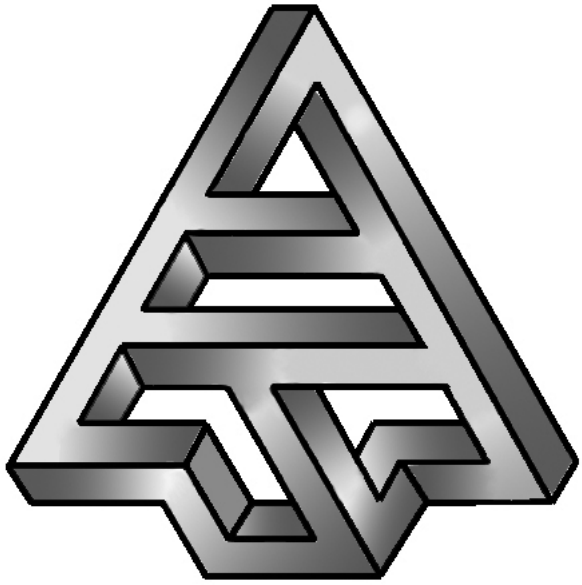
Al pulsar sobre esta opción el usuario puede ver la máximas puntuaciones obtenidas. Pulsando cualquier tecla se retorna a la pantalla principal

Créditos

Al pulsar sobre esta opción el usuario puede ver el nombre de los creadores del videojuego. Pulsando cualquier tecla se retorna a la pantalla principal

Ayuda

Al pulsar sobre esta opción el usuario puede consultar las teclas con las que se maneja a los jugadores. Pulsando cualquier tecla se retorna a la pantalla principal.



Vega Solaris

Parametrización

Versión 1.0
04/07/2006

ÍNDICE

1. PREÁMBULO

1.1. Introducción

Este documento tiene la intención de que junto con el documento "Archivos de Datos", sirva de ayuda a la hora de modificar aquellos parametros del juego que se han permitido modificar sin necesidad de tener que acceder al código fuente.

2. IMAGENES

2.1. Eliminación

- Cuando se desee eliminar una imagen del conjunto de imágenes incluidas en el juego habrá que asegurarse de que dicha imagen no es referenciada en ningún fichero .
- Se deberá borrar el fichero deseado y deberá eliminarse su referencia en el fichero *imágenes.vs.*

2.2. Adición

- Se deberá incluir la nueva imagen en algún subdirectorio bajo la carpeta *imágenes*
- Dicha imagen deberá ser referencia en el documento *imágenes.vs* según el formato de dicho fichero. El código que se le asigne a la imagen deberá ser único evitando que dos imágenes diferentes tengan el mismo código.

2.3. Sustitución

- Este apartado se refiere únicamente a la situación en la que se desea sustituir todas las aparciones de una imagen por otra diferente.
- Se deberá borrar la imagen antigua e incluir la nueva imagen.
- El código correspondiente a la imagen antigua deberá mantenerse, pero la ruta de la propia imagen, lógicamente debe sustituirse por la de la nueva imagen en el fichero *imagenes.vs*
- Se recomienda que la nueva imagen tenga un tamaño lo más similar posible a la original para evitar posibles descolocaciones de la pantalla.
- En caso de que se desee cambiar los objetos que aparecen en el decorado de una habitación, la secuencia de fotogramas que forman algún movimiento, por poner algunos ejemplos, habrá que mirar los apartados consecuentes a este mismo.
- Una excepción a este apartado son las imágenes que forman la pantalla de carga. Para modificar estas imágenes habrá que modificar el fichero *pantalla_carga.vs*, siguiendo el formato descrito en el documento "Archivos de Datos".

3. MAPA

3.1. Añadir una habitación nueva

- Para añadir una habitación nueva habrá que seguir los siguientes pasos:
  - Asegurarse de que las que las coordenadas que se desean dar a la nueva habitación no están ya ocupadas por otra en el mapa.
  - Asegurarse de que la nueva habitación va a ser accesible (de forma directa o indirecta) desde la casilla de salida. Para ello posiblemente sea necesario modificar alguna habitación contigua a la nueva habitación.
  - Incluir su *declaración* en el fichero mapa.vs
  - Crear los los ficheros correspondientes en las carpetas libres, objetos y puertas según se especifica en el documento "Archivos de Datos".

3.2. Eliminar una habitación

- Se deberá eliminar su referencia correspondiente en el fichero *mapa.vs*
- Borrar los 3 ficheros correspondientes en las carpetas libres, objetos y puertas.
- Asegurarse de que la eliminación de esta habitación no implica que otras regiones del mapa queden inaccesibles.

3.3. Modificar una habitación

- Si lo que se desea es modificar los lugares donde eventualmente pueden ubicarse items de forma aleatoria, habrá que modificar el fichero correspondiene a la habitación en ela carpeta *libres*. Habrá que asegurarse de que la posición de los items no está cerca de las puertas de tal manera que pueda provocar que la habitación, e incluso en consecuencia una región del mapa, quede inaccesible.
- Si se desean modificar los elementos del decorado de la habitación habrá que modificar, respetando su formato , los ficheros correspondientes a la habitación en las carpetas *puertas* y *objetos.*
- En caso que se desee por ejemplo, añadir un nuevo elemento de decorado inexistente hasta ese momento en ningún lugar del mapa, también habrá que seguir los pasos del apartado2.2 de esta sección.

3.4. Modificar el fondo de las habitaciones

- Si se desea modificar el fondo de una habitación en concreto, habrá que modificar las 1ª linea de texto del fichero correspondiente a la habitación en la carpeta *objetos*
- Si se desea modificar el fondo por defecto de todas las habitaciones (aplicable en caso de que el fondo especifico de la habitación no exista) deberá modificarse el fichero *global.vs* según se especifica en el documento "Archivos de Datos".

4. MODIFICACIÓN DE LAS ANIMACIONES DEL JUEGO

Para modificar cualquiera de las animaciones existententes en el juego (movimientos del humano, movimientos del extraterrestre, flecha y hechizo básico) simplemente habrá que habrá que modificar su fichero correspondiente según se indica en el documento "Archivos de Datos".

En caso de que estas nuevas animaciones requieran la inclusión, sustitución o eliminación del conjunto de imágenes existentes en el juego, deberán apicarse los pasos indicados en el apartado 2 de esta misma sección.

Los ficheros correspondientes a cada conjunto de animaciones son los siguientes:

- movimientos del humano → humano.vs
- movimientos del extraterrestre → extraterrestre.vs
- movimientos de la flecha → flecha.vs
- movimientos de la bola del hechizo básico → hechizoBasico.vs

5. MODIFICACIÓN DE LOS TEXTOS DE PANTALLAS Y BOTONES

El texto de las pantallas y botones también es modificable. Esto da lugar a que las traducciones a otros idiomas sea sumamente fácil. Para modificar cada pantalla o botón simplemente habrá que modificar su fichero correspondiente atendiendo al formato especificado para dichero fichero en el documento "Archivos de Datos".

El fichero correspondiente a cada pantalla o conjunto de botones es el siguiente:

- Pantalla de ayuda → ayuda.vs
- Pantalla de créditos → creditos.vs
- Pantalla de fin de partida → fin.vs
- Pantalla de inicio → inicio.vs
- Pantalla de puntuaciones → puntuaciones.vs
- Pantalla de fin de partida por tiempo agotado → textoFinTiempo.vs
- Pantalla de fin de partida por ser el ganador → textoTalismaGanador.vs
- Pantalla de fin de partida por ser el perdedor → textoTalismaPerdedor.vs
- Conjunto de botones relativos a la conexión multijugador → multijugador.vs
- Resto de botones → menus.vs

Para modificar el texto que aparece sobreimpresionado cuando el juego está pausado deberá modificarse el fichero *global.vs* según se especifica en el documento "Archivos de Datos".

6. MODIFICACIÓN DE OTROS ASPECTOS DEL JUEGO

6.1. Aliens

Aparte de la posibilidad de modificar las imágenes que conforman la animacion de los aliens actuales, aspecto ya comentado en el punto nº 4, también pueden añadirse aliens nuevos o eliminarse algunos de los existentes.

Para eliminar algún tipo de alien existente simplemente habrá que elimenar el fichero *nombreDeAlien.vs* y quitar la referencia correspondiente en el fichero *aliens.vs*. En caso de que también corresponda que eliminar las imágenes asociadas del conjunto de imágenes de la aplicación, se seguirán tambien los pasos del punto 2.1

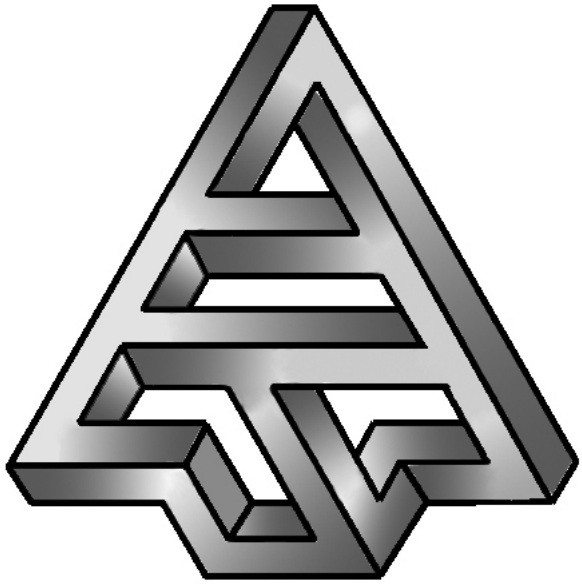
Para añadir un nuevo tipo de alien se llevará el proceso inverso: añadir al conjunto de imágenes de la aplicación aquellas que sean necesarias, crear un fichero con las animaciones del alien, y referenciar dicho fichero en *aliens.vs*

6.2. Items

Pueden añadirse, eliminarse y modificarse items del juego modificando el fichero *items.vs*. En el documento "Archivos de Datos" están claramente especificados los campos y los correspondientes posibles valores que se les puede dar a cada tipo de item.

6.3. Panel de estado

Las imágenes que forman el panel de estado pueden ser reemplazadas por otras siguiendo el apartado 2.3 de esta sección y la especificación del fichero *panel.vs* del documento "Archivos de Datos".



Vega Solaris

Posibles Mejoras

Versión 1.0
04/07/2006

ÍNDICE

1. AMPLIACIÓN DEL REPERTORIO DE OBJETOS

En este "remake" se han respetado los objetos del juego original VS89. Los objetos que contenía el juego son los siguientes:

- Fragmentos del talismán Vega Solares.
- Armas, entre las que se podían encontrar:
  - Espadas.
  - Arcos.
  - Puñetazos.
- Escudos.
- Conjuros, como los siguientes:
  - Básico (una bola de luz que se lanza contra los enemigos).
  - Nocturno (oculta la visión del otro personaje).
  - Teletransporte (para viajar instantáneamente a la habitación en la que está el otro personaje).
  - Temporal (detiene el tiempo salvo para quien lo activa y destruye a los bichos).
- Comida (pollo asado, helado y bebida)

El juego ha sido desarrollado pensando en la posibilidad de ampliar el número de objetos que pueden encontrarse los jugadores. Para ello hay que incorporar la clase con la información del nuevo objeto, su gráfico y el código necesario en el personaje jugador y personaje máquina para realizar la acción que produce dicho nuevo objeto.

Ejemplos de nuevos objetos:

- Armas:
  - Pistola y rifle láser. Sirven para disparar al otro personaje o a los enemigos. La pistola se diferencia del rifle en que sólo puede disparar las balas de una en una mientras que el rifle puede disparar ráfagas de balas consecutivas.
  - Lanza de energía. Similar al arco pero de un solo uso.
  - Carga explosiva. Se coloca en una determinada posición de una habitación y se detonan a voluntad del personaje que la colocó o cuando alguien pasa cerca.
  - Cuchillo. Similar a la espada pero causando un daño menor.
  - Boomerang. Este objeto se lanza y retorna al dueño del arma.
  - Lanzallamas. Este objeto lanza una columna de fuego contra sus enemigos.
  - Cañón BFG. Este objeto dispara una bola de energía que le quita toda la vida al otro personaje o bicho con el que impacta.

- Hechizos:
    - Repelente: cuando lo usa el personaje todos los bichos de la pantalla huirán del personaje.
    - Amistad: tras activarlo, el otro personaje no puede usar ninguna de sus armas.
    - Hambre: tras activarlo, la comida que ingiera el otro jugador no le aumentará su vida.
    - Bolsillo roto: tras activarlo, el otro personaje empezará a perder los objetos que tenía en su inventario uno detrás de otro. Los objetos quedan en el suelo en la casilla en la que estaba el otro personaje antes de que los perdiese.
    - Atontar: tras lanzarlo, los mandos del otro jugador se invierten (por ejemplo, para moverse a la izquierda debe moverse a la derecha)
    - Perder: tras lanzarlo, el otro jugador pierde uno de los fragmentos del talismán que se coloca aleatoriamente en una habitación del mapa. Si no tenía ningún fragmento del talismán, entonces no hace nada.
    - Sanguijuela: mientras que esté en uso, una determinada porción de vida pasa periódicamente del otro jugador al jugador que lanzó el hechizo.

- Objetos:
    - Llaves: para abrir puertas o accesos a partes del mapa que estarían bloqueados de otra forma.
    - Blindaje: reducen una parte del daño sufrido por el personaje.
    - Dinero: para comprar objetos.
    - Botiquín de primeros auxilios.

Otra ampliación que se podría realizar es aumentar el número de objetos que puede tener en el inventario el jugador.

2. TELETRANSPORTES PARA DESPLAZARSE POR EL MAPA

Se podrían incorporar al juego teletransportes que permitiesen moverse más rápidamente a los jugadores por el mapa. Estos teletransportes podrían ser de tres tipos:

- Aleatorios solo ida: mandan al personaje a otro teletransporte aleatoriamente y sin opción de retorno.
- Aleatorios ida y vuelta: mandan al personaje a otro teletrasporte aleatorio pero el teletransporte destino debe poder teletransportar de nuevo al personaje.
- Dirigidos: indican el teletransporte al que se debe enviar el personaje. Son todos de ida y vuelta.

Para implementar esta modificación habría que modificar el modelo del juego para incorporar transportadores. También habría que añadir los gráficos necesarios del sistema de teletransporte y de los personajes siendo teletransportados.

Se podría permitir que los bichos también se teletransportasen al entrar en un teletransportador. En este caso el salto siempre es aleatorio pues los bichos no son lo suficientemente inteligentes como para manipular los mandos del teletransporte.

3. AMPLIACIÓN DE LA FAUNA SALVAJE DEL PLANETA

Otra ampliación que se podría realizar es ampliar el número de criaturas salvajes con las que se pueden encontrar los jugadores.

Para ello, bastaría crear la clase de la nueva criatura con la información de sus atributos, añadir sus gráficos y crear las clases con su inteligencia artificial propia. De esta forma se pueden crear criaturas muy resistentes difíciles de abatir, otras que sean muy inteligentes, criaturas muy rápidas, etc.

Entre las criaturas que se podrían introducir están dinosaurios, mutaciones, humanoides, bestias salvajes, plantas carnivoras, etc.

4. NUEVOS MODOS DE PARTIDA MULTIJUGADOR

Además del modo de juego multijugador "uno-contra-uno" se podrían incorporar nuevos modos de juego multijugador como:

- "Todos-contra-todos": en este modo los personajes son de diferentes razas (humano o alienígena) pero sólo uno de todos ellos puede escapar. La lucha es contra los miembros de la otra especie y contra los de la propia especie.
- "Capturar-la-bandera": en este tipo de partida no gana aquél que consigue los cuatro fragmentos sino aquella especie que consigue mantener los cuatro fragmentos del talismán durante más tiempo en su poder.
- "Por-equipos": en este tipo de partida hay dos bandos (alienígenas o humanos) y gana aquél bando que consigue tener uno o más jugadores del mismo bando en la habitación de salida y entre ellos completan los fragmentos del talismán Vega Solaris.
- "Supervivencia": en este tipo de partida hay un número muy elevado de bichos por pantalla. Gana aquél que consigue llevar todos los jugadores de su equipo antes a la salida.

5. CARACTERÍSTICAS RPG

En el juego actual no hay personajes no jugadores (PNJ) con los que los personajes puedan hablar e interactuar.

En esta mejora, en el mapa se podrán encontrar PNJs con los que los jugadores podrán hablar para recibir pistas sobre el paradero de los fragmentos del talismán. Además podrán comprar armas y objetos a algunos de estos PNJs. Y podrán ofrecerse para ser contratados como mercenarios y que luchen al lado de los personajes..

Inicialmente el personaje empieza con unas características muy reducidas, pero según vaya abatiendo enemigos y bichos del planeta, sus habilidades aumentarán pudiendo realizar más daño con sus golpes, correr más rápido, ser más resistente a los golpes, etc.

Se establecerá un sistema de niveles que requieren una determinada puntuación en experiencia para progresar al siguiente nivel.

6. HERIDAS REALISTAS

En esta mejora, los jugadores son más realistas en sus movimientos cuando son heridos. De esta forma cuando tenga toda la vida, el personaje se moverá correctamente. Pero según reciba más heridas, se moverá más lento y sus golpes harán menos daño. Además los gráficos del personaje se irán llenando de moratones por los golpes recibidos.

Bibliografía

Páginas en Internet

[1] http://www.fdi.ucm.es/profesor/fernan/PG/html/vega_solaris.html
Página web del proyecto original Vega Solaris 89. Realizada por el profesor Fernando Sáenz. Contiene archivos, imágenes, documentación y otra información relevante (mapa original, descripción inteligencia artificial...).

[2] http://www.fdi.ucm.es/profesor/fernan/PG/html/vega_solaris._the_remake.html
Página web oficial del proye cto Vega Solares The Remake. Contiene información sobre la presentación llevada a cabo en MadriSX&Retro06.

[3] http://www.planetalia.com/cursos/index.jsp
Tutoriales sobre cómo desarrollar un videojuego con Java utilizando los paquetes “awt” y “swing”

[4] http://www.programacion.com/java/tutorial/ags_j2me/
Tutoriales y guías sobre la programación de teléfonos móviles utilizando J2ME. Contiene ejemplos explicativos sobre los conceptos explicados.

[5] http://www.speccy.org/vegasolaris/
El Trastero del Spectrum es una página dedicada a mantener con vida videojuegos para Spectrum.Contiene manuales, emnuladores y roms de Spectrum. En este enlace se da acceso a la información sobre el Vega Solaris 89.

[6] http://spa2.speccy.org/
Página de retroinformática dedicada a recuperar juegos para el Sinclair Spectrum.

[7] http://www.todosymbian.com/
Página para desarrolladores de juegos con J2ME y C++/Symbian. Contiene tutoriales, guías, manuales de instalación programas J2ME y foros de preguntas de desarrolladoresde aplicaciones

[8] http://java.sun.com/javame/reference/docs/index.html
Página oficial de Sun de J2ME. Contiene especificaciones y documentación en inglés de los desarrolladores de J2ME.

[9] http://forum.nokia.com/devices
Enlace a la página del foro de Nokia. Esta página contiene información sobre las características de los diferentes tipos de móviles Nokia y la tecnología que soportan (CLDC, MIDP, tamaño del display).

[10] http://www.javaperformancetuning.com/tips/j2me.shtml
Página orientada a proporcionar información sobre J2ME centrándose en aspectos como la optimización de aplicaciones, reducción del tamaño de los programas, benchmarking, capacidades wireless, etc.

[11] http://sourceforge.net/mail/?group_id=86829
Página de SourceForge del grupo de desarrollo del plugin EclipseME. En esta página pueden encontrarse las especificaciones, documentación, archivos necesarios para descargar el módulo, etc.

[12] http://www.j2medev.com/api/midp/index.html
Enlace a la documentación de J2ME de Sun. Contiene información sobre los paquetes y las clases que conforman J2ME.

[13] http://discussion.forum.nokia.com/forum/
Foro de desarrolladores de aplicaciones J2ME. De gran utilidad para plantear dudas difíciles de solucionar y que profesionales de la programación con J2ME la resuelvan.

[14] http://www.forum.nokia.com/info/sw.nokia.com/id/2b17fb6f-b9a4-4cd8-80fd-94b8251a048e/Games_Over_Bluetooth_v1_0_en.zip.html
Ejemplo del funcionamiento de Bluetooth en el desarrollo de juegos multijugador para dispositivos móviles.

[15] http://java.sun.com/j2se/1.4.2/docs/api/
Enlace a la página con la documentación oficial del JDK 1.4.2 de Sun.

[16] http://www.microdevnet.com/articles/techtalk/optimization?PageNo=1
Página con información sobre la optimización en los juegos desarrollados con J2ME.

[17] http://www.club-java.com/TastePhone/J2ME/MIDP_Benchmark.jsp;jsessionid=A28B94B7DB0A9415D4E0448B5436A186
Página con información sobre los diferentes teléfonos móviles que hay en el mercado y sus características (CLDC, MIDP, display, etc)

[18] http://pisuerga.inf.ubu.es/lsi/Invest/Java/Tuto/V_2.htm
Enlace con el código de un ejemplo del uso de los sockets.

[19] http://developers.sun.com/techtopics/mobility/blueprints/articles/game/
Enlace de Sun sobre programación de videojuegos.

[20] http://www.informit.com/articles/article.asp?p=358064&seqNum=1&rl=1
Enlace sobre un programa básico. Explica las API´s, los MIDlets y el esqueleto básico de un juego estándar.

[21] http://grasia.fdi.ucm.es/
Grupo de investigación de Agentes Software de la Universidad Complutense de Madrid.

[22] http://www.gamedev.net/
Página con muchos contenidos sobre el desarrollo de videojuegos.

[23] http://www.mailxmail.com/curso/informatica/j2me
Manual de programación de juegos.

[24] http://arianne.sourceforge.net/
Página de SourceForge con el juego multijugador Stendhal, ejemplo de juego multijugador.

[25] http://www.fdi.ucm.es/profesor/jpavon/
Página del profesor Juan Pavón sobre agentes software. Contiene mucha información sobre los agentes software.

[26] http://www.forum.nokia.com/main/0,6566,034-21,00.html
Página que permite descargarse un simulador de móvil.

[27] http://www.javabluetooth.com/development_kits.html
Página que permite descargar software para programar Bluetooth con java.

[28] http://ingenias.fdi.ucm.es/
Página herramienta INGENIAS desarrollada en la facultad de Informática de la Universidad Complutense de Madrid. Esta herramienta sirve para modelar agentes software.

Libros

[1] "Java 2. Iniciación y Referencia", J. Sánchez Allende, G. Huecas Fernández-Toribio, B. Fernández Manjón, P. Moreno Díaz, Editorial McGraw-Hill, 2001.

[2] "Programación de Juegos en Java", Joel Fan, Eric Ries, Calin Tenitchi, Editorial Madrid Anaya Multimedia, 1998.

[3] "Developing Games In Java", David Brackeen, Editorial Indianapolis: New Riders, 2004.

Palabras Clave

- Vega
- Solaris
- Java
- J2ME
- Bluetooth
- Videojuego
- Multijugador
- Parametrización
- Spectrum
- Retroinformática

Autorización

Los abajo firmantes autorizan a la Universidad Complutense de Madrid a difundir y utilizar con fines académicos, no comerciales y mencionando expresamente a sus autores, tanto la propia memoria, como el código, la documentación y/o el prototipo desarrollado.

Sergio Díaz Jubera Javier Gallego Ahijón José María Sobrinos García

Madrid, a 04 de julio de 2006