



JUEGOS 3D EN J2ME CON LA NUEVA API 3D JSR-184



Alberto Gutiérrez Martínez (alberto@flagsolutions.net)

TABLA DE CONTENIDOS

1.	Introducción al mercado del entretenimiento para teléfonos móviles	3
	El mercado móvil en Europa	3
	El Emergente mercado del entretenimiento móvil.....	4
	Desarrollo de juegos J2ME y “La vuelta al pasado”	7
2.	Estado del arte de juegos en 3D para terminales móviles	8
	Sistema Operativo Symbian.....	8
	Breve historia de los juegos en 3D	9
	La tercera dimensión en teléfonos móviles.....	11
3.	Posibilidades de la nueva API.....	13
	¿Qué es JSR-184?.....	13
	Características de su motor 3D.....	15
4.	Herramientas necesarias:	17
	Herramientas de compilación:.....	17
	Herramientas de modelado 3D y conversión:.....	18
5.	Pruebas prácticas de la api.....	20
	Desarrollo de un juego sencillo:	20
	Estudio crítico y conclusiones	23
6.	Bibliografía	26
7.	Glosario de términos y siglas.....	28
	Ilustración 1: Penetración de la telefonía móvil en Europa.....	4
	Ilustración 2: Uso de servicios avanzados en España	6
	Ilustración 3: ¿Qué servicios móviles estaría dispuesto a usar en un futuro?	6
	Ilustración 4: Castle Master.....	10
	Ilustración 5: Wolfenstein.....	10
	Ilustración 6: Alone in the Dark	11
	Ilustración 7: Tomb Raider para N-Gage	12
	Ilustración 8: Modelo 3D con <i>Bones</i>	15
	Ilustración 9: Técnica de Morphing	16
	Ilustración 10: Conversor de H3T a M3G	19
	Ilustración 11: M3G Viewer	19
	Ilustración 12: Pogoroo en un emulador J2ME	20
	Ilustración 13: Escena M3G creada	21
	Ilustración 14: Con corrección de Perspectiva.....	22
	Ilustración 15: Sin corrección de perspectiva	22
	Ilustración 16: Con texturas transparentes	23
	Ilustración 17: Diferentes opciones	23

1. INTRODUCCIÓN AL MERCADO DEL ENTRETENIMIENTO PARA TELÉFONOS MÓVILES

El año 2002 marcó un hito en la historia de las telecomunicaciones. Por primera vez, el número de usuarios de telefonía móvil superó al número de usuarios de telefonía fija en el conjunto de Europa.

Con un mercado de las telecomunicaciones móviles que se está moviendo hacia su madurez (8 de cada 10 europeos tiene móvil), el crecimiento del mercado va a venir principalmente por la renovación de los terminales y los **nuevos servicios de valor añadido** basados en la transmisión de datos.

A ese respecto, 2003 fue un año decisivo: los terminales multimedia fueron promocionados fuertemente.

En el 2004 estos terminales fueron progresivamente adoptados por los usuarios finales, lo que convirtió al 2004 en el año de los servicios interactivos y multimedia.

Todos estos precedente hacen que en el momento actual, 2005, existan numerosos usuarios potenciales de juegos para móviles. Sin embargo, hay otro actor imprescindible en el mercado de entretenimiento, las operadoras de telefonía. A continuación se analizará la situación actual de la telefonía móvil en Europa, para entender el **interés de las operadoras en el emergente mercado de entretenimiento para móviles**.

El mercado móvil en Europa

Aunque el mercado móvil europeo se está moviendo en una dirección común, como vamos a ver aún se mantienen algunas diferencias entre los diferentes países. Con más de 400 millones de usuarios, Europa es el segundo gran mercado de telefonía móvil del mundo.

Después de un periodo de fuerte crecimiento, el mercado europeo está llegando ahora a un punto de saturación. En un mercado maduro como éste, el número de nuevas subscripciones (y por lo tanto el número total de usuarios) tiende a estabilizarse.

La penetración media de la telefonía móvil alcanzó más del 83% en el 2003. Italia, Suecia y Portugal tienen las mayores tasas de penetración con más del 92%. En aquellos países donde la tasa de penetración es menor, como Polonia y Rusia, el mercado está creciendo de forma importante.

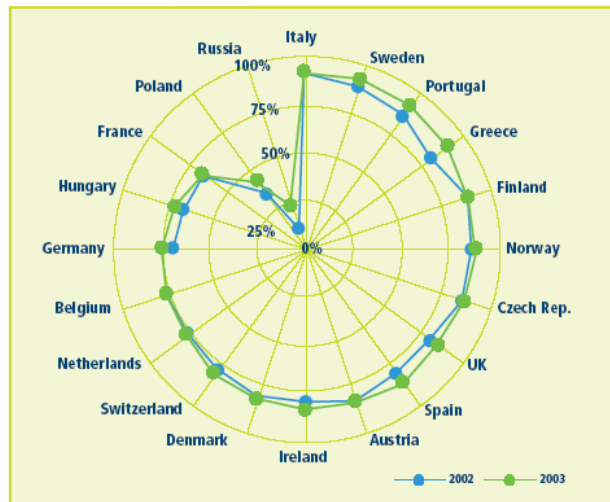


Ilustración 1: Penetración de la telefonía móvil en Europa

El ingreso medio por usuario (**ARPU, Average Revenue Per User**) para los servicios de voz continúa decreciendo, debido a la feroz competencia entre los diferentes operadores de cada país por conseguir los últimos nuevos clientes.

Los operadores necesitan orientar sus estrategias hacia los servicios móviles con transmisión de datos si quieren impulsar sus ingresos. En el 2003, los ingresos no relacionados con la voz aumentaron considerablemente, gracias sobre todo a los SMS y últimamente, aunque sólo tímidamente, a los MMS y a la navegación.

El mercado de los servicios de valor añadido es similar entre los diferentes países europeos, siendo los servicios de personalización del terminal los más populares en toda Europa con una facturación del 40 % del total.

Los operadores están orientando todos sus esfuerzos en conseguir proporcionar servicios avanzados a los usuarios que resulten interesantes y que por lo tanto sean utilizados para conseguir aumentar la cuenta de resultados y simultáneamente diversificar el origen de sus ingresos.

El Emergente mercado del entretenimiento móvil

Cada vez más los usuarios están interesados en formas móviles, portátiles de entretenimiento que ofrezcan nuevas posibilidades de juego. Las modernas tecnologías ya permiten todas estas posibilidades aunque hasta ahora no hayan sido aplicadas en todo su potencial.

Actualmente, hay una amplia gama de dispositivos electrónicos para juegos en el mercado, empezando por las consolas. La popularidad de las consolas de videojuegos está fuera de toda

duda. Por ejemplo, Sony recientemente anunció ventas internacionales de más de 60 millones de consolas PlayStation 2 desde que el producto se lanzó al mercado hace algo más de 5 años. Los ordenadores personales son también otro medio popular para el juego electrónico, especialmente con el creciente aumento de actividades online.

Respecto a las consolas portátiles, la consola Game Boy Advance de Nintendo es el producto líder con más de un millón de consolas vendidas en Europa desde que salió a la venta. Es de destacar que un fabricante de teléfonos móviles como NOKIA lleve ya un par de años intentando colarse en este sector con su móvil/consola portátil NOKIA N-Gage.



El número de usuarios de estos productos está también subiendo, especialmente entre las mujeres, personas por encima de los 30 años, familias y parejas. Casi el 20 % de los jugadores está por encima de los 50 años, las mujeres con más de 18 conforman el segundo grupo de jugadores con el 26 %, ganando al tradicional grupo de niños de 6 a 17 años que representan sólo el 21 %.

En el mundo móvil la revolución de los juegos está empezando a ser un hecho. Ahora ya puedes descargarte juegos en tu teléfono, juegos a color que son mucho más sofisticados que los de hace tan sólo un par de años.

Hasta el 2003, los juegos eran principalmente instalados de fábrica en blanco y negro y de un solo jugador. Pero ahora, el desarrollo de juegos está en plena revolución y **es un mercado con enorme potencial de crecimiento en los próximos años.**

Esta revolución puede ser resumida con los siguientes **avances**:

- Desde los juegos en blanco y negro a los juegos a color
- Desde los juegos de un jugador a los juegos multi-jugador
- Desde los juegos embebidos a los juegos descargables

Estas mejoras sólo están siendo posibles gracias a la mejora de las capacidades de los terminales. Los usuarios ya no tendrán nunca más que jugar con juegos en blanco y negro monojugador, o preinstalados en su terminal. Ahora podrán fácilmente descargar juegos a color e incluso jugar contra usuarios conectados a la misma red.

Aún así hay tres **principales obstáculos** para ofrecer estos servicios de **entretenimiento móvil**:

- **Primero**, ofrecer un servicio de alta calidad que sea capaz de gestionar cómodamente una gran cantidad de peticiones de descarga en cortos espacios de tiempo y que permite realizar rápidas descargas.
- **Segundo**, desarrollar juegos interactivos con funciones para multi-sesión, multi-jugador manteniendo interfaces de usuario atractivos y amigables.

Desarrollo de juegos J2ME y “La vuelta al pasado”

En los comienzos de la industria de los videojuegos, el desarrollo de un juego era el resultado del trabajo de un único programador o como mucho de un grupo de no más de 10 personas.

Sin embargo, los presupuestos que se manejan actualmente para desarrollar software de entretenimiento, nada tiene que ver con aquéllos de los comienzos. Las causas pueden estar en un aumento inmenso de la competencia en este sector, que cada vez ponía el listón más alto para destacar entre los miles de juegos que salían al mercado cada año, y en un aumento paulatino de las posibilidades gráficas y sonoras del hardware y, en consecuencia, de la demanda de los jugadores.

Es muy significativo el hecho de que a principios de los años 90, muchas películas de Hollywood orientadas a los jóvenes se “llevaban” en forma de videojuego a la pantalla de los PCs (*Ghostbusters* fue el pionero), mientras que en los últimos años hemos asistido a lo contrario, Juegos de PC que posteriormente se convierten en películas multimillonarias de Hollywood (*Street Fighter*, *Mortal Kombat*, *Tomb Raider*, *Resident Evil*...).

En consecuencia, los presupuestos para el desarrollo de juegos de ordenador y consolas se asemeja ahora al de una película de Hollywood. Sólo al alcance de grandes compañías y multinacionales.

Centrándonos ahora en el nuevo y emergente mercado de juegos J2ME, es de destacar que los presupuestos que se manejan, nada tienen que ver con el mercado del entretenimiento digital actual. Es por esta razón que se habla de una “**vuelta al pasado**”, pues **un juego J2ME comercial, puede desarrollarse en un tiempo razonable por un único programador.**

El equipo humano ideal debería estar formado por dos o tres personas. Un programador JAVA, a ser posible con conocimientos de las técnicas clásicas de programación de videojuegos, y al menos un diseñador gráfico. El tiempo de desarrollo de un Juego JAVA puede estar entre 3 y 6 meses, dependiendo claro está, de la complejidad de éste.

Actualmente una de las prioridades de cualquier empresa de tecnología es minimizar el *Time to market*. No sólo se trata de realizar productos de calidad sino de que éstos estén en la calle antes que los de la competencia. De este modo se consigue además de la reducción de costes, aprovechar el tirón comercial que tiene un producto innovador, y lo último en juegos J2ME son los **juegos en 3D** que aún están por llegar al mercado.

2. ESTADO DEL ARTE DE JUEGOS EN 3D PARA TERMINALES MÓVILES

Sistema Operativo Symbian

El “Windows” en telefonía móvil se llama Symbian OS (<http://www.symbian.com>). Y recibe el nombre de “Windows”, no sólo porque sea un sistema operativo, sino porque está tan extendido en los teléfonos móviles, como Windows en los ordenadores personales. Actualmente la mayoría de los fabricantes de teléfonos móviles (Nokia, Sony-Ericsson, Siemens, Samsung y Panasonic, entre otros), instalan Symbian OS en sus terminales. Aunque cambian la interfaz gráfica, y parecen Sistemas Operativos distintos, por debajo todos ejecutan el mismo Sistema Operativo. En el momento de elaboración de este trabajo, Symbian OS va por su versión 8, que ya tienen los ultimísimos modelos de Nokia.

Existen actualmente **dos tecnologías para desarrollar aplicaciones para teléfonos móviles** con sistema operativo Symbian, una es (JAVA2ME) y otra es C++. Debe tenerse muy claro que desde el momento en el que se decide emplear tecnología JAVA para programar aplicaciones se deberán asumir no sólo las ventajas sino también las desventajas de usar código interpretado. Éstas son principalmente baja velocidad de ejecución y limitaciones funcionales. Para entender estas desventajas hay que saber distinguir entre código interpretado y código nativo.

El código nativo es el código que se obtendría al compilar un programa en C++. Es el código máquina que puede ser ejecutado por el sistema operativo del teléfono móvil.

El código interpretado es el código que se obtendría al compilar un programa en JAVA. Ese código debe ser traducido a código máquina en el momento de su ejecución, por la Máquina Virtual JAVA. Esa traducción es la responsable de la baja velocidad de ejecución mencionada anteriormente.

Cuando hablamos de limitaciones funcionales se hace referencia a que sólo se dispondrá de los recursos que la implementación de la máquina virtual JAVA permita utilizar. La máquina virtual JAVA por tanto aísla, en la llamada sandbox (caja de arena), las peculiaridades del teléfono que se está programando. Razón por la cual con un programa J2ME no se podrá por ejemplo, usar la conexión por infrarrojos del teléfono, a no ser que el fabricante nos proporcione una API a tal efecto. Por ejemplo, la especificación MIDP 1.0 de JAVA2ME, sólo permite el uso de tonos telefónicos (ring tones) en cuanto a sonido se refiere. Sin embargo, los fabricantes de terminales con capacidad para reproducir sonidos polifónicos o MIDI, como

Nokia o Sony-Ericsson, permiten descargar desde la web las APIs que entre otras, proporcionan funciones para que los programas puedan sacar el máximo partido de sus terminales.

En la Tabla 1 se muestra una comparativa entre las dos tecnologías comentadas:

	J2ME	SYMBIAN OS
Tamaño de aplicación permitido	Decenas de Kb.	Varios MB.
Estándar Abierto	Sí	Sí
Deployment (Despliegue)	Grande	Más pequeño
Soporte de Varios Fabricantes	Sí	Sí
Instalación OTA (Over-the-Air)	Sí	Sí ¹
Ejecución de forma Nativa	No	Sí
Lenguaje de programación	JAVA	C++
Comunicación con Servidores Remotos	Sí	Sí
Animación en 2D	Sí	Sí
Animación en 3D	Normalmente No ⁴	Sí
Mostrar Vídeos	Normalmente No ²	Sí, si lo permite el terminal
Audio MIDI	Normalmente No*	Sí
Audio de Alta Calidad	Normalmente No*	Sí
Acceso a SMS	Normalmente No ³	Sí (Además MMS si lo soporta el terminal)
Acceso a puerto de Infrarrojos y Bluetooth	Normalmente No ⁵	Siempre que lo tenga el terminal
Acceso a la agenda, Calendario, etc	No	Sí
Marcar un teléfono	No	Sí
Complicaciones de Desarrollo Multiplataforma	Menos que SYMBIAN	Sí

Tabla 1: Comparación J2ME y SYMBIAN OS

* Se necesita la API propietaria del fabricante o MIDP 2.0.

¹ Sin embargo, Las aplicaciones de Symbian OS son normalmente tan grandes que es imposible la distribución OTA.

² J2ME puede reproducir vídeos en teléfonos que soporten la JAVA Mobile Media API como el NOKIA 3650 y posteriores o en teléfonos con MIDP 2.0.

³ Acceso a los SMS desde J2ME es posible usando la NOKIA SMS API (Soportada en NOKIA 3410) o la Wireless Messaging API (soportada por los NOKIA 3650 y posteriores).

⁴ **Sólo para los móviles de última generación que soportan la especificación JSR-184. Una completa API 3D basada en Open GL. (Open GL ES)**

⁵ Sólo para terminales con MIDP 2.0 y/o que soportan la *Bluetooth API*.

Breve historia de los juegos en 3D

Quizá uno de los hitos en la Historia de los Videojuegos fue la aparición el 5 de Mayo de 1992 de un juego que dejó a todos los usuarios de PC boquiabiertos, este juego fue el Wolfenstein. El precursor de los llamados Juegos 3D en primera persona o 3D Action Games. Aunque en los tiempos del Spectrum ya existieron juegos que utilizaban mallas vectoriales para representar mundos en 3D (*Freescape games*), éstos eran tan esquemáticos que resultaban confusos de jugar. El que más éxito alcanzó fue un magnífico juego aparecido en 1990 llamado Castle Master. Este juego era una aventura gráfica en 3D y fue probablemente el primero que permitió el movimiento por un mundo virtual en 3D, y precursor de otro juego muy avanzado

para su época como fue el 7th Guest. Sin embargo, ninguno de estos juegos llegó al público de forma tan masiva como lo hizo el mítico Doom. Tanto es así, que desde su aparición se creó un género llamado Juegos tipo Doom. La única gran novedad que aportaron respecto al Wolfenstein fue que ahora se usaba el ratón para apuntar el arma, y no sólo las teclas cursoras. Y es que como se ha mencionado antes el verdadero pionero fue el Wolfenstein. Fue el primero en mostrar gráficos 3D con texturas bastante realistas, (que eran renderizados por software ya que no existía ningún apoyo de las tarjetas 3D como ocurre actualmente). Pero lo más importante es que fue el primero en mostrar una perspectiva de cámara en que sólo se veía el arma del protagonista, consiguiéndose una inmersión en el juego sin precedente hasta esa fecha. Desde la aparición de Doom y sus sucesores (Quake, Half Life, etc) se desató una “fiebre” por los juegos en 3D que revolucionó el mercado del entretenimiento, provocando que los fabricantes de Hardware y Sistemas Operativos tuvieran como principal meta, crear plataformas donde los usuarios pudieran jugar a los cada vez más espectaculares juegos en 3D. En cierto modo, **han colaborado al desarrollo de la tecnología actual**, ya que cada vez demandaban más recursos de las máquinas y éstas se quedaban anticuadas antes, con lo que los usuarios adquirirían nuevo hardware cada menos tiempo, con el consiguiente abaratamiento de costes.



Ilustración 4: Castle Master



Ilustración 5: Wolfenstein

En el mismo año en el que aparecía el Wolfenstein, una empresa francesa llamada Infogrames lanzaba al mercado un juego revolucionario, el Alone in the Dark. Era un juego tipo perspectiva isométrica en el manejo del personaje, pero esta vez con un motor 3D real. La acción transcurría en una mansión encantada, teniendo como novedad técnica que la cámara se encontraba en un sitio distinto en cada estancia de la casa. De modo que el manejo no resultaba sencillo, pero resultaba gráficamente muy atractivo, ya que en todo momento se veía perfectamente las acciones que realizaba nuestro personaje, como si fuera una película. Por esa razón a este tipo de juegos se les llamó Juegos en Tercera Persona.



Ilustración 6: Alone in the Dark

No mucho después de que juegos realmente violentos como el Quake mantuvieran ocupados a buena parte de los adolescentes, empezaron a surgir juegos en 3D en los que la perspectiva del jugador permitía que viéramos a nuestro personaje desde atrás. De este modo en 1996 vio la luz otro juego que ha sido imitado hasta la saciedad, el Tomb Raider. Mucha gente habla de juegos tipo Tomb Raider como una variedad de juegos con características comunes.

La tercera dimensión en teléfonos móviles



En el momento de realización de este trabajo, los juegos J2ME en 3D son una realidad, ya que acaban de aparecer en el mercado un NOKIA (el 6630) y un Sony-Ericsson (K-700) cuyas respectivas máquinas virtuales JAVA soportan la **3D Graphics API (JSR-184)**. Este API se apoya en el motor gráfico OPEN GL ES (Embedded Systems). Por tanto, si se pretende realizar un juego para móviles en 3D usando tecnología JAVA, ya no habrá que crear desde cero un motor 3D propio como se venía haciendo hasta hace poco con escaso éxito, ya que el teléfono no tenía potencia para mover incluso el mundo virtual más sencillo. Si se quiere realizar juegos con este



API bien es verdad que hay que tener en cuenta que todavía no será compatible con el 99% de los terminales del mercado, y de hecho a día de hoy en España no existe en el mercado ningún juego Java 3D para estos teléfonos. Sin embargo a nivel internacional ya existe una compañía que se ha posicionado en este sector con juegos 3D ya listos para distribuir. Esta empresa se llama **Superscape Group Inc.** (<http://www.sperscape.com>) y en dicha web se hace referencia a que han realizado la primera implementación comercial de JSR-184 del mundo. Sin embargo, para la venta de juegos J2ME 3D disponen de un página web a parte (<http://www.swervepowered.com/>). Es de destacar que si se busca información para comprar dichos juegos (<http://www.swervepowered.com/basketball/wheretobuy.asp>) se informa de que la empresa se encuentra aún en negociaciones con las operadoras y fabricantes de teléfonos, lo que da idea del estado de inmadurez de este mercado. Más adelante se volverá a hacer referencia a esta empresa cuando se comenten las herramientas necesarias para el modelado 3D para J2ME.

Aunque para JAVA 2ME los juegos en 3D están empezando, no es así para los juegos nativos para Symbian. Para la serie 60 de Nokia, existen desde sus comienzos, juegos en 3D realmente espectaculares como es el: *Interstellar Flames*, o el mítico *Doom*. Si se posee un Nokia N-Gage, desde hace tiempo se puede jugar a versiones del Tomb Raider o del Fifa 2004 que nada tienen que envidiar a las versiones de PC



Ilustración 7: Tomb Raider para N-Gage

Sin embargo, se están haciendo grandes esfuerzos por parte de los fabricantes de hardware para que la aceleración 3D llegue a los terminales móviles. Hace ya un año que las dos principales firmas de tarjetas aceleradoras, ATI y Nvidia han sacado al mercado sendos procesadores gráficos para teléfonos móviles y PDAs, la **ATI ImageOn 3200** y la **nVIDIA GoForce 4000** respectivamente. No obstante, estos procesadores están más orientados a la aceleración 2D de imágenes de vídeo, que a 3D.



3. POSIBILIDADES DE LA NUEVA API

¿Qué es JSR-184?

La *Mobile 3D Graphics API* es un **paquete opcional** que se complementa con los perfiles existentes de J2ME. La plataforma objetivo de este API opcional es J2ME/CLDC, usado con perfiles tales como MIDP 1,0 o MIDP 2,0. Sin embargo, el API se puede también poner en ejecución encima de J2ME/CDC (para *Smart Phones*), o cualquier plataforma de Java.

La especificación fue definida dentro del proceso de la comunidad de Java ("JCP", JAVA Community Process) bajo petición 184 ("JSR-184 " de la especificación). El líder del proceso fue NOKIA.

Define un API para renderizar gráficos tridimensionales (3D) a *frame rates* interactivos, incluyendo una estructura de *Scene Graph* (una especie de mundo 3D completo metido en un único archivo con formato **M3G**).

La función de esta API es proveer a los programadores Java, medios eficientes y flexibles de exhibir los gráficos animados 3D en tiempo real en los dispositivos que la soporten. Para cumplir con las necesidades de diversos tipos de usos y aplicaciones, se proporcionan una estructura de gráficos de la escena (**scene graph**) fácil de utilizar llamada **modo retenido (Retained mode)** y una interfaz de bajo nivel más compleja pero más potente llamada **modo inmediato (Immediate mode)**.

En el momento de la concepción de este API fueron identificados varios usos: juegos, visualización de mapas, interfaces de usuario, mensajes animados, visualización de productos, y salvapantallas. Cada uno de estos usos tiene diversas necesidades: algunos requieren la creación de contenidos simple, algunos requieren alto rendimiento de procesamiento de polígonos, mientras que otros requieren imágenes de alta calidad con efectos especiales. Es evidente que una gama tan amplia de necesidades no se puede satisfacer con un *Scene Graph* solamente, ni con un "modo inmediato" solamente. Está también claro que tener dos APIs separadas conduciría a la confusión del desarrollador y un uso poco óptimo de la tan escasa memoria del terminal. Por tanto, todo tipo de animación y características de representación están disponibles **simultáneamente** tanto para los objetos del *scene graph* como para los objetos renderizados individualmente si se trabaja a bajo nivel. El desarrollador en consecuencia no necesita elegir entre el modo inmediato y el gráfico de la escena, sino que puede mezclar y emparejar ambos dentro del mismo uso.

El “modo inmediato” es un subconjunto de OpenGL, y el “modo retenido” está construido encima del modo de bajo nivel, de tal modo que cualquier escena representada por un *Scene Graph* siempre podría ser conseguida utilizando exclusivamente métodos de la interfaz de bajo nivel.

Además del API en sí mismo, se define también un **tipo de archivo** para el almacenaje eficiente y la transferencia de todos los datos necesarios. Son los llamados archivos **M3G**. Estos datos incluyen las mallas (*Meshes*), texturas, jerarquías de la escena, características de los materiales, *keyframes* de la animación, etcétera. Los datos son escritos en un archivo con herramientas de creación de contenidos en un ordenador, y cargados en la API a través de la clase **Loader**.

En resumen de todo lo anteriormente comentado podemos resaltar las siguientes características de la API JSR-184:

- El API soporta **Retain Mode** modificando elementos de una *Scene Graph* (es decir, la escena 3D entera metida en un archivo).
- El API soporta el **Immediate Mode** o acceso de bajo nivel, con unas características similares a OpenGL.
- El API soporta mezclar y emparejar el **Retain Mode** y el **Immediate Mode** de tal modo que una misma ejecución la aplicación pueda renderizar de ambas maneras la escena.
- El API no tiene partes opcionales (es decir, todos los métodos deben de estar implementados).
- El API dispone de importadores de las mallas, texturas, gráficos enteros de la escena, etc. A tal efecto se encuentra la clase **Loader**.
- El API está eficientemente implementado sobre **OpenGL ES** (Embedded Systems).
- El API utiliza el tipo de datos nativo *float* de Java que en J2ME no se encuentra hasta la versión 1.1 de CLDC.
- El API funciona eficientemente sin hardware de coma flotante. Aunque puede haber teléfonos que incluyan dicho hardware.

- El API ocupa en torno a los 150 KB en un terminal móvil real.
- El API se encuentra estructurado para reducir al mínimo la recolección de basura (*Garbage collector*).
- El API es “interoperable” con APIs de Java relacionadas, especialmente con MIDP.

El API 3D además se integra fuertemente con el paquete LCDUI de MIDP (paquete de clases relacionado con gráficos en 2D), de modo que los gráficos en 2D y 3D se pueden renderizar indistintamente tanto en la pantalla como en un buffer gráfico cualquiera.

Características de su motor 3D

A pesar de estar limitado a 150Kb, el motor 3D de JSR-184 posee gran parte de las características de su hermano mayor **Open GL**. Quizá la falta más evidente, por otro lado compresible, es el soporte para la **proyección de sombras** de los objetos. Es lógico que se haya suprimido pues si ya en un PC se requiere hardware bastante potente para que el mundo virtual proyecte sombras a tasas interactivas (es decir a unos 15 fps como mínimo), tanto más así en un terminal móvil por muy avanzado que este sea. Por esta misma razón JSR-184 tampoco incluye **sistemas de partículas**. Sin embargo, sorprende ver en la API clases como la *SkinnedMesh* que permiten el uso de *bones*. Técnica con la que el modelo 3D dispone de información adicional que permite el movimiento restringido de su “esqueleto” junto con la textura que “envuelve” a los triángulos afectados por dicho movimiento.

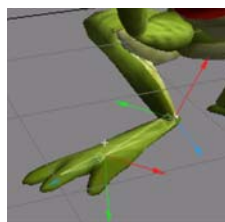


Ilustración 8: Modelo 3D con *Bones*

Muy relacionado con el uso de *bones* también es la técnica de *Morphing* consistente en el uso de unas mallas de referencia para deformar un modelo 3D, encargándose la API de calcular las mallas intermedias entre ellas mediante interpolación. Con esta técnica se puede animar un modelo 3D sin tener que usar para ello una animación hecha con una herramienta de autor o

una ingente cantidad de vértices. La clase usada en JSR-184 para este fin se llama *Morphing Mesh*.



Ilustración 9: Técnica de Morphing

En JSR-184 encontramos además otras clases relacionadas con el soporte y control de animaciones de los modelos 3D. Estas clases son: *AnimationController*, *AnimationTrack* y *KeyFrameSequence*. Existe posibilidad de efecto niebla (clase *Fog*), detección de colisiones e intersecciones (clase *RayIntersection*), distintos tipos de *blending* de imágenes transparentes (incluso sprites transparentes que siempre están paralelos a la cámara, muy útiles para efectos de explosiones, clase *Sprite3D*), materiales, texturas (con posibilidad de corrección de perspectiva e incluso filtrado trilinear), imagen de *Background* que puede ser mostrada por trozos y los tipos de luces típicos de cualquier motor 3D.

4. HERRAMIENTAS NECESARIAS:

Herramientas de compilación:

Aunque hace un par de años la compilación, preverificación (etapa nueva en J2ME) y empaquetado de clases había que hacerlo de forma manual, actualmente Sun proporciona un *kit* de herramientas que facilitan estas tareas. Esta herramienta es la llamada *J2ME Wireless Toolkit*. Lo primero que hay que señalar sobre esta herramienta es que no es un IDE como tal. Realmente por sí sola no abarca todo el ciclo de vida del software, ya que se necesita otra herramienta para editar el código fuente. Valdría con cualquier editor de texto como el bloc de notas, aunque lo usual es cohabitar con el Sun One Studio o Netbeans. De hecho se puede integrar dentro de este, para lo cual es preferible tener instalado previamente el Sun One antes de instalar el Wireless Toolkit. Para el desarrollo del presente trabajo se ha integrado con otro IDE muy potente y gratuito llamado ECLIPSE.

Antes de instalar el *Wireless Toolkit* es necesario tener instalado el JAVA 2SE SDK (<http://java.sun.com/j2se/downloads.html>) (se recomienda el SDK 1.4.x) para posteriormente seguir las instrucciones de instalación del J2ME Wireless Toolkit 2.2 de SUN que se puede descargar de http://java.sun.com/products/j2mewtoolkit/download-2_2.html

Una vez instalado correctamente el software de SUN para compilar un MIDlet, hay que buscar la aplicación *KToolBar*. Pulsar sobre New Project. Dale el nombre al proyecto (sin espacios) y en el campo *MIDlet name* hay que poner el nombre del MIDlet (la aplicación en cuestión). Posteriormente aparecerá otro diálogo con la etiqueta *API Selection* activa. Se debe seleccionar aquí como *Target Platform: MIDP 2.0 All Platforms definition* (ya que es la única plataforma que contiene la JSR-184 API) y al aceptar hay que observar que volvemos al entorno de la *KToolBar* que ahora indica con un texto, dónde deberemos poner los fuentes y los recursos del proyecto.

Fuentes en: [unidad de instalación]/WTK22/[NombreProyecto]/src/

Recursos en: [unidad de instalación]/WTK22/[NombreProyecto]/res/

Una vez se sitúen los ficheros .java en el directorio de fuentes correspondiente, sólo hay que pulsar en *Build* para compilar y/o *Run* para ejecutar en un emulador.

Herramientas de modelado 3D y conversión:

Tras una investigación preliminar sobre el formato **M3G** aparecía como opción ideal para su creación la última versión de **3D Studio MAX**. La versión 7.0. Sin embargo ante la imposibilidad de conseguir dicho software, se procedió a investigar sobre plug-ins para la versión anterior de dicha herramienta de modelado, con la que sí se contaba. La versión 6 SP1. En los foros de NOKIA (<http://forum.nokia.com>) se hacía mención a un plug-in para las versiones 5 y 6 de 3D Studio. Este plug-in además era gratuito y estaba disponible en (<http://www.m3gexporter.com/>). Tras varios intentos fallidos de instalación siguiendo al pie de la letra el proceso de instalación, se comprobó que por alguna razón desconocida la versión de **3D Studio** de la que se disponía era incapaz de cargar una de las tres DLLs necesarias para ejecutar correctamente el plug-in. Esto ocasionó que hubiera que seguir buscando una solución para conseguir una escena 3D con el formato requerido por JSR-184. Se encontró en seguida que la misma empresa que desarrolla el *m3gexporter* (y los primeros juegos en 3D comentados anteriormente) tiene un producto de pago que es sin duda la solución ideal para desarrollar escenas 3D en formato **M3G** de forma profesional. Este producto se llama **Swerve Studio**

(http://www.superscape.com/products/swerve_studio/). Esta solución también era descartada y se siguió buscando en Internet. Llama la atención el hecho de que exista todavía muy poca información en Internet respecto a **M3G**, y tanto es así que las posibles soluciones para conseguir un conversor se cuentan con los dedos de una mano. Se encontraron algunos exportadores de software libre (**Juinness**, <http://sourceforge.net/projects/juinness/>) pero todos estaban en fase experimental. Incluso llama la atención el caso de una empresa británica que conocedora de los problemas para conseguir exportadores de **M3G** ha creado un formato propio que vende junto con un MIDlet que es capaz de cargar y gestionar dichos modelos: (<http://www.jcodeworks.com/demomidp.html>). Esta solución a parte de ser bastante cara tenía un gran *handicap*, no soporta animaciones. Algo casi imprescindible para un juego en 3D profesional. Tras descartar de nuevo esta solución se encontró por fin una empresa (http://www.mascotcapsule.com/M3G/download/e_index.html) que proporcionaba un exportador de un formato de archivo poco usual (**H3T**) a **M3G**. Por suerte también proporcionaban un plug-in que podía exportar de **3D Studio** a **H3T**. En ese momento parecía que se había encontrado una buena solución. Se creaba la escena en 3D MAX, se exportaba a H3T y luego se usaba el conversor a M3G. La **ilustración 10** muestra el aspecto de la sencilla aplicación convertidora. En las primeras pruebas que se hicieron de conversión de modelos de 3D Studio a M3G no hubo ningún problema. Sin embargo, cuando se probó a incluir una textura en un modelo, la conversión daba un error. Se investigó el origen de dicho error y se encontró que el formato **H3T** era muy parecido al formato **OBJ** en cuanto a que se trata de

texto ASCII estructurado que define los elementos y vértices que componen la escena. Analizando dicho texto se encontró que el archivo **H3T** hacía referencia a la imagen de la textura en cuestión, pero le había modificado el nombre. Se procedió entonces a cambiar el nombre del archivo y se consiguió que el programa conversor hiciera la conversión con éxito.

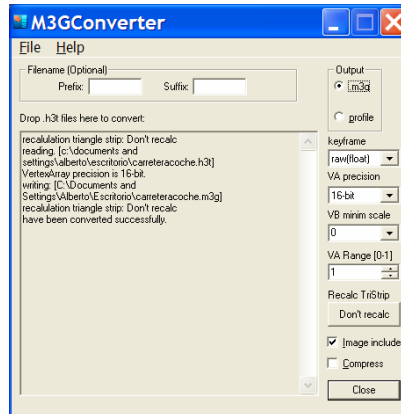


Ilustración 10: Conversor de H3T a M3G

El software de conversión de H3T a M3G incluía también un completo visor de archivos **M3G** que muestra la **Ilustración 11**.

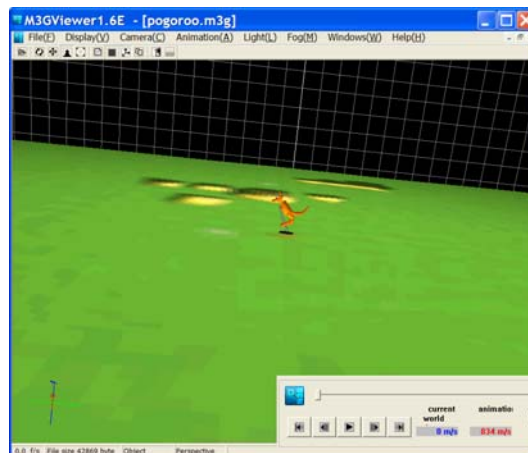


Ilustración 11: M3G Viewer

Cuando se probó el archivo **M3G** con textura que había sido convertido con éxito, se comprobó que el programa era incapaz de cargarlo. Más adelante se descubrió que **la altura y la anchura de las texturas en JSR-184 deben de ser potencias de 2**. Sin embargo no ocurre lo mismo con la imagen que se puede poner de fondo (clase *Background*) o con las imágenes que se pueden utilizar en un *Sprite3D*. Se cambió por tanto el tamaño de la textura y se comprobó que el M3G Viewer lo cargaba sin problemas así que se dio por terminada esta fase de búsqueda de herramientas y se pasó al desarrollo del juego en 3D.

5. PRUEBAS PRÁCTICAS DE LA API

Desarrollo de un juego sencillo:

Llegados a este punto ya se tenía claro la forma de proceder para desarrollar el juego y se tenían grandes conocimientos de la API gracias a lecturas muy recomendables como la del Dr. Andrew Davison (<http://fivedots.coe.psu.ac.th/~ad/jg/objm3g/index.html>) en la que se imparte una especie de curso avanzado sobre el uso de JSR-184 a bajo nivel, e incluso se enseña cómo convertir ficheros en formato **OBJ** a métodos JAVA de la API 3D que crean el mismo modelo. Sin embargo, el conocimiento avanzado mostrado en estos documentos era puramente académico y nada práctico para el desarrollo de juegos 3D en un tiempo razonable. Según se menciona en otro interesante documento (sobre todo para iniciarse en JSR-184) de NOKIA (3-D_Game_Development_On_JSR-184_v1_0), la complejidad de una escena 3D en un juego debe ser abordada usando un enfoque “**Retained Mode**” y usar el modo de bajo nivel sólo para efectos y ocasiones muy concretas.

Antes de empezar a programar el juego, se estudió a fondo un ejemplo sobre JSR-184 que se incluye en el *Wireless Toolkit 2.2* de Sun. Este ejemplo se llama *pogoroo* y tiene la peculiaridad sobre los demás ejemplos incluidos, en que al probarlo sobre un terminal real, el NOKIA 6630, todo funciona según lo previsto. El resto de ejemplos no se ejecutaban en dicho terminal. La **Ilustración 12** muestra dicho programa en ejecución en el emulador.



Ilustración 12: Pogoroo en un emulador J2ME

Una vez estudiado el proceso de carga del fichero M3G que hacía *pogoroo*, se hizo un proceso semejante para cargar fichero M3G creado y convertido según el proceso explicado en el apartado anterior. El M3G en cuestión consistía en un modelo bajo de polígonos de un coche y un plano con una textura en la que se dibujaba un circuito por el que se debería manejar el coche.

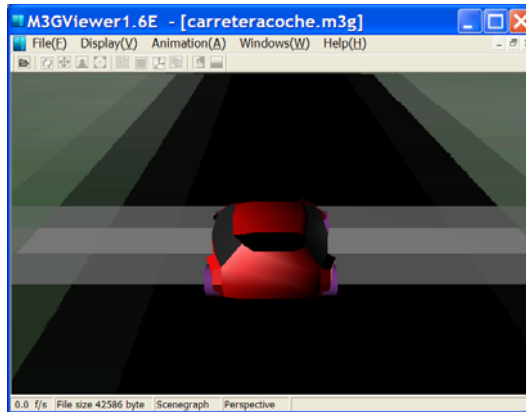


Ilustración 13: Escena M3G creada

Se realizó el programa que se entrega con el nombre **modoRetained.zip** con el que se pretendía cargar la escena contenida en el fichero **M3G** y renderizarlo sobre la pantalla del móvil. El mencionado programa cargaba sin problemas la escena pero en el primer intento de renderizado (método `g3d.render(mundo)`) el programa lanzaba una excepción de tipo *IllegalStateException*, que según la documentación de JSR-184 (http://www.forum.nokia.com/files/nds_disclaimer/1,6673,3959,00.html) podía estar relacionada por la falta de una cámara activa de la escena o por un fallo del contexto gráfico sobre el que se está intentando renderizar. Se descartaron ambas causas y se concluyó tras comprobar que con el mismo código se cargaba correctamente y se mostraba por pantalla cualquiera de los archivos M3G de los ejemplos que se disponían, que el error estaba relacionado con una errónea conversión por parte de las herramientas que se habían utilizado. Esta conclusión tuvo si cabe más fuerza tras hacer numerosas pruebas simplificando al máximo la escena desde 3D Studio MAX y siguiendo el proceso de exportación mencionado anteriormente.

Llegados a este punto, y ante la imposibilidad de contar con un fichero M3G propio para el desarrollo de un juego, se optó por utilizar un fichero M3G de ejemplo con los que se contaba. Se utilizó el fichero del ejemplo del programa de ejemplo *pogoroo* y se observó que era una idea descabellada pues el árbol de objetos que contiene un archivo M3G es críptico para alguien que no tiene acceso al fichero con el que se originó.

Con este nuevo fracaso se optó por utilizar la API a bajo nivel o lo que es lo mismo en “**modo inmediato**”. Por supuesto el desarrollo de un juego en este modo suponía una labor demasiado compleja para el objetivo del presente trabajo de investigación. Se optó por partir de un código de ejemplo que aparece en la propia documentación de JSR-184 en la que se crea un cubo desde cero, definiendo todos sus vértices, los triángulos que forman la figura final, las normales de las caras de dichos triángulos y las coordenadas donde se carga una textura en cada cara del cubo. Se añadieron nuevas funcionalidades a dicho programa como el uso de texturas transparentes, luz tipo *spot light*, activar/desactivar la corrección de la perspectiva y se probaron distintos tipos de filtrado. Como curiosidad se añadió un método que cambia de forma aleatoria el eje de rotación del cubo permitiendo ver el modelo girar de infinitas formas posibles. Las siguientes ilustraciones muestran el programa en ejecución en el emulador de J2ME que se entrega con el nombre **ModoInmediato.zip**:



Ilustración 14: Con corrección de Perspectiva



Ilustración 15: Sin corrección de perspectiva



Ilustración 16: Con texturas transparentes



Ilustración 17: Diferentes opciones

Estudio crítico y conclusiones

El programa desarrollado sobre el emulador se ejecutó en el terminal de prueba (un NOKIA 6630) y se comprobó que a diferencia del emulador, el programa no se ejecutaba a pantalla completa (algo que era lógico pues no se había activado el modo a pantalla completa con el método *setFullScreenMode* de la clase *Canvas* de MIDP 2.0).

Lo primero que llamaba la atención era la suavidad con la que se movía el cubo y la calidad del **filtrado trilineal** que lleva a cabo la implementación de la API de NOKIA, que es muy superior al del emulador, en el que no existe diferencia apreciable al activarlo.

Se probó a ejecutar el programa activando el **antialiasing**. Se comprobó que la implementación de la API de NOKIA no lo soporta, y el programa se ejecuta de forma idéntica a como lo haría sin antialiasing. Según la especificación JSR-184, la implementación del antialiasing es opcional, así que no sorprende que ésta no se encuentre implementada.

Por lo demás no existe ningún fallo apreciable en la ejecución del programa. Sin embargo, este hecho no es indicativo dada la simplicidad del programa desarrollado y sobre todo porque estamos trabajando a bajo nivel. En este sentido se hizo una pequeña investigación por los foros de NOKIA (<http://forum.nokia.com>) y se transcribe a continuación la enumeración de errores encontrados por los desarrolladores en la implementación de JSR-184 en el NOKIA 6630:

(<http://discussion.forum.nokia.com/forum/showthread.php?s=&threadid=53764&highlight=%2Am3g+exporter%2A>)

1. Sprite3D works fine in render(World), but not in render(Node, Transform). Workaround: Put all sprites into a World.
2. Depth offset does not work properly. This bug originates from the OpenGL ES engine, so it also affects Symbian applications.
3. The background image may flicker on and off when scrolled. A wrapping background image is erroneously not drawn if the crop rectangle is outside of the viewport. Workaround: Use a modulo operation to always put the crop rectangle within the viewport.
4. Camera and light transformations in Graphics3D are left in an incorrect state after a render(World). Workaround: Set the camera and lights manually.
5. Transformable.pre/postRotate does not work. This happens after setting an initial orientation with Transformable.setOrientation(0, 0, 0, 0), which is legal according to the spec but results in invalid values internally. Workaround: Any non-zero axis to setOrientation(0, ...) will work.
6. The Loader fails to load compressed M3G files produced with HI's exporter. This is because of a bug in handling of empty sections. Workaround: do not tick the "Compress" checkbox in the M3G Converter tool.

Como se puede observar, los fallos empiezan a ser numerosos y bastante importantes como para “volver loco” a un desarrollador que cree que está haciendo algo mal en su programa cuando en realidad son fallos no documentados de la implementación del teléfono. Lo más preocupante no es que haya *bugs* y que no estén documentado, sino que **los bugs son arreglados por NOKIA en distintas versiones de la máquina virtual del teléfono** (se puede comprobar la versión del *firmware* de la máquina virtual marcando **#0000#*). Este hecho unido a que los usuarios no llevan su teléfono por norma general a que le actualicen el *firmware*, provocan una gran incertidumbre en los desarrolladores de software. En el escenario actual, si se quiere desarrollar un juego en 3D, el desarrollador debe conocer los *bugs* de cada versión del *firmware* sobre la que se va a ejecutar y tomar dos caminos. O bien se decide a no usar en su implementación nada que no esté soportado en todas las versiones del *firmware* (con lo que el arreglo de fallos por parte de NOKIA resulta inútil), o bien debe ser capaz de que su programa funcione de una u otra manera detectando la versión del *firmware* en tiempo de

ejecución. (o creando una nueva compilación por cada versión). Si ya de por sí los teléfonos móviles son completamente heterogéneos y debemos de hacer diferentes compilaciones para distintos modelos, el hecho de contemplar también versiones de *firmware* convierten a la plataforma J2ME en una plataforma nada portable, lo que unido a las limitaciones funcionales que vienen impuestas, pueden provocar que los desarrolladores prefieran desarrollar aplicaciones nativas en Symbian, que aunque más difíciles de programar, son mucho más potentes, y paradójicamente pueden llegar a ser más portables que J2ME.

6. BIBLIOGRAFÍA



LIBROS:

- **“Programación de Videojuegos para teléfonos móviles en JAVA con J2ME”** Ediversitas Multimedia
- **“Micro JAVA Game Development”** Addison-Wesley Personal Education
- **“J2ME Game Programming”** Thomson Course Technology
- **“Manual Avanzado de JAVA 2 Edición 2000”** Anaya Multimedia
- **“De Super Mario a Lara Croft: La Historia Oculta de los Vídeo Juegos”** Editorial Dolmen



DOCUMENTOS ELECTRÓNICOS:

- Curso de Desarrollo de Juegos para Móviles – Universidad de Salamanca - Tema 1 (Javier Pérez Trigo)

Todos los documentos aquí mostrados son descargables de la web de Nokia previo registro gratuito como desarrollador en <http://developer.nokia.com>

- 3-D_Game_Development_On_JSR-184_v1_0.zip
- Audio_Programs.pdf
- BlockGame_v1_0.pdf
- Brief_Introduction_to_MIDP_Graphics_v1_0.pdf
- Brief_Introduction_to_Networked_MIDlets_v1_0.pdf
- Camera_MIDlet_A_Mobile_Media_API_Example_v1_0.pdf
- Color_Management_in_Series_30_Series_40_and_Series_60_Devices.pdf
- Comparison_of_Nokia_7650_and_Nokia_3650_v1_4.pdf
- Designing_Bluetooth_Apps_for_Series_60_v1_4.pdf
- Designing_JAVA_Application_for_Series_60.pdf
- Image_Memory_in_Color-Screen_Phones.pdf
- Installing_MIDlets_into_the_Games_Menu_of_Nokia_Devices_v1_0_en.pdf
- Issues_in_6600.pdf
- J2ME_Symbian_OS_1_01.pdf
- JAVA_Development_and_Nokia_APIs_by_Chiam_Poh_Guan.pdf

- JAVA_MIDP_App_Dev_Guide_v1_0.pdf
- jsr184-specification-1.0.pdf
- Known_Memory_Issues_with_the_Nokia_7650.pdf
- MIDlets_Borland_v1_0.pdf
- MIDP_and_Game_UI_v_1_0.pdf
- MIDP_APIs_and_Nokia_Extension_APIs_Hemant_Madan.pdf
- NDSforJ2ME_InstallationGuide_v2_0.pdf
- NDSforJ2ME_UsersGuide_v2_0.pdf
- Nokia_3650_FAQ_1_02.pdf
- Nokia_Sound_Converter_v1_0.pdf
- Nokia_UIAPI_Guide.pdf
- Nokia_UI_API_public_1_1_JAVAdoc.zip
- nS60_jme_symbian_sdk_1.2_InstallationGuide.pdf
- OTAProvisioning-1.0.pdf
- RI_Binary_License_for_developers_JSR-184.pdf
- Single_Player_Games_v1_01.pdf
- The_Nokia_3650_Mobile_Media_API.pdf
- Tones_v1_0.pdf
- Using_Ant_and_Antenna_MIDP_v1_0.pdf
- Why_J2ME_Projects_Fail.pdf
- Wireless_Messaging_API_by_Chiam_Poh_guan.pdf
- Basic Over the Air-Server Configuration.pdf (Motorola)



ENLACES INTERNET: (15/Marzo/2005)

- <http://forum.nokia.com>
- <http://developers.sun.com>
- <http://www.flipcode.com>
- <http://www.series60.com>
- <http://www.bluej.org/>
- <http://home.rochester.rr.com/ohombres/jScience/>
- http://www.onJAVA.com/pub/a/onJAVA/excerpt/j2menut_3/index4.html?page=1
- <http://www.kickJAVA.com/>
- <http://www.eclipse.org>
- <http://eclipseme.sourceforge.net/>

7. GLOSARIO DE TÉRMINOS Y SIGLAS

JAVA: Lenguaje de programación orientado a objetos de Sun Microsystems aparecida en 1996. <http://java.sun.com>

JAVA 2 SE: Estándar Edition. Es el subconjunto de clases de JAVA de propósito general.

JAVA 2 ME o J2ME: Micro Edition. Es el subconjunto de clases de JAVA orientadas a dispositivos pequeños y portables, como teléfonos móviles y agendas electrónicas.

CLDC: Connected Limited Device Configuration. Define el conjunto de clases JAVA que deben estar presentes en una implementación de la máquina virtual JAVA de un entorno J2ME

MIDP 1.0: Mobile Information Device Profile versión 1.0. Subconjunto de clases de Java que debe implementar toda máquina virtual que cumpla con dicha versión del perfil.

MIDP 2.0: Mobile Information Device Profile versión 2.0. Subconjunto de clases de Java que debe implementar toda máquina virtual que cumpla con dicha versión del perfil.

MIDlet: Aplicación de MIDP.

Bluetooth: Bluetooth es una tecnología de radio de corto alcance, que permite conectividad inalámbrica entre dispositivos remotos. Opera en la banda de los 2,4Ghz.

MMS: Multimedia Messaging Service. Servicio de mensajería multimedia que permite transmisión asíncrona de imágenes, audio y vídeo. Se apoya en WAP y SMIL

OTA (Over-the-Air): Modo de instalación de un MIDlet sobre una red de telefonía móvil, en la que intervienen un servidor WAP, y el gestor de aplicaciones de la implementación J2ME del terminal móvil.

SMS: Short Message Service. Servicio de mensajes cortos de los teléfonos de segunda generación (2G) y en adelante, que permite enviar y recibir de forma asíncrona un texto formado por hasta 160 caracteres.

MIDI: Musical Instrument Digital Interface. Es un estándar de conectividad para instrumentos musicales y ordenadores.

SMIL: Synchronized Multimedia Integration Language. Lenguaje parecido a HTML que permite sincronizar diferentes tipos de medios, como sonido e imagen.

XHTML: Es un subconjunto de HTML que está basado en XML.

GPRS: General Packet Radio Service. Es una tecnología de radio para redes GSM que añade protocolos de intercambio de paquetes.

GSM: Global System for Mobile communications. Es el estándar de facto europeo para los servicios de telefonía celular.

UMTS: Universal Mobile Telecommunications System. Estándar europeo para los teléfonos móviles de tercera generación.

JVM: JAVA Virtual Machine. Siglas para referirse a la máquina virtual JAVA que interpreta y ejecuta las aplicaciones programadas en este lenguaje.

KVM: Kilobyte Virtual Machine. Máquina virtual JAVA adaptada a los terminales móviles, que debe ocupar aproximadamente 1Kb.

Nokia UI API: Nokia User Interface API. Biblioteca de clases para J2ME que añaden funcionalidades demandadas por los desarrolladores de juegos JAVA para móviles Nokia, que no tiene el estándar MIDP 1.0.

Smart Ring Tone: Archivo sonoro de Nokia que permite almacenar melodías polifónicas en sincronización con vibraciones de la batería.