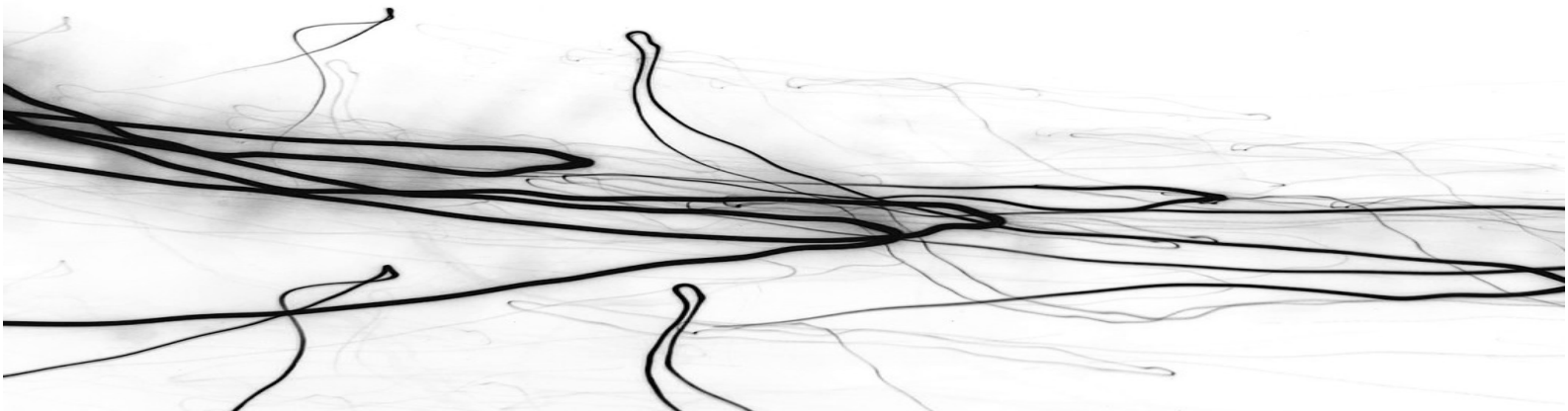
II Máster Universitario en
Comunicaciones y Desarrollo
de Servicios Móviles

Simulación de entornos virtuales en dispositivos móviles.
Implementación en gráficos vectoriales

TESINA

autor: **Luis Leiva Torres**
directora: **Ruth Manzanares Rubio**

fecha: **Julio 2006**



TESINA

Simulación de entornos virtuales en dispositivos móviles.
Implementación en gráficos vectoriales

autor: **Luis Leiva Torres**
directora: **Ruth Manzanares Rubio**
fecha: **Julio 2006**

Contenidos

Ilustraciones

Tablas

1. Introducción

En su relativamente corta pero rápida vida, las páginas web han pasado a ser no ya unos meros documentos textuales a los que se puede acceder por Internet, sino unas verdaderas presentaciones multimedia, que combinan textos con imágenes, sonidos, videos y elementos de Realidad Virtual (RV).

Este escenario se está repitiendo en el mundo de los dispositivos móviles, donde poco a poco observamos nuevos modelos que ofrecen unas prestaciones inimaginables hace escasos meses. La convergencia entre el mundo de la web y el de dichos dispositivos se hace inevitable, en cuanto a la gestión de contenidos y presentación de los mismos. Una línea de investigación relativamente reciente consiste en implementar las tres dimensiones en la web, lo que se conoce como web3D. Dado que se ha fijado este punto de partida para realizar la presente tesina, en repetidas ocasiones se hablará de “web3D” y “entornos 3D para dispositivos móviles” indistintamente para referirse a la misma acepción. Y es que actualmente muchos desarrolladores para la web hacen lo propio en dispositivos portables [LEIVA06], dada la inevitable convergencia entre ambos entornos salvando unas cuantas diferencias, claro está.

Por dispositivo móvil se entiende un computador de mano de alta movilidad funcionando con baterías, el cual es capaz de capturar, procesar, presentar y transmitir datos desde y hacia un servidor o trabajando en modo local. Algunos productos de estas características son los Palms, PDAs, Teléfonos Celulares, Smartphones, etc.

Nótese que un dispositivo se considera móvil cuando el usuario puede usarlo mientras se mueve: un equipo portátil no se debe o puede usar mientras se mueve, por ejemplo los Notebook. La batería de un dispositivo móvil dura todo el día, las de uno portátil entre 2 y 4 horas, el peso de los portátiles es hasta 15 veces el de uno móvil, la resistencia a caídas y golpes de los móviles es hasta 10 veces la de los portátiles, etc.

La navegación en dispositivos móviles cuenta con la gran limitación del procesado de instrucciones y el bajo ancho de banda en las redes móviles actuales, lo que complica el trabajar con archivos voluminosos como tradicionalmente son los archivos de 3D. Otra gran limitación de estos dispositivos es el teclado. Esta limitación se puede ver superada con la incorporación progresiva de tecnologías de reconocimiento de voz y escritura. Quizá hoy día una solución muy interesante para llevar 3D a los dispositivos móviles es crear renderings de objetos y animaciones en formato vectorial [COPESA00].

Actualmente, a nivel de usuario, en todo dispositivo móvil se hace uso de pistas visuales y auditivas únicamente. En algunos modelos de teléfono es posible incorporar además sensaciones táctiles por medio de la vibración del terminal, lo cual enriquece el uso de nuevas aplicaciones. Combinadas adecuadamente con la interactividad y las técnicas de simulación que se van a presentar en este trabajo es posible generar entornos tridimensionales con un mínimo de medios, optimizando así recursos.

2. Motivación

El desarrollar aplicaciones para dispositivos móviles no deja de ser un reto en la actualidad, dadas las limitaciones comentadas anteriormente y las características intrínsecas de estos aparatos. Este tipo de dispositivos se caracteriza por no tener disco duro, en su lugar cuentan con potentes recursos de memoria RAM y la posibilidad de usar tarjetas de almacenamiento externo (Storage Cards) para almacenar los datos. Disponen además de procesadores que actualmente no llegan a los 400 Mhz de frecuencia de reloj.

Otro gran problema al que nos enfrentamos al realizar una aplicación para un dispositivo móvil es el de la resolución de pantalla. En el mundo web, en su momento, se tuvo que lidiar con dicho handicap, pese a que el número de resoluciones diferentes no era muy elevado. Sin embargo, en dispositivos móviles el problema se acentúa increíblemente: en el mercado existen infinidad de displays, cada uno con un tamaño y resolución totalmente distintos el uno del otro. Y parece que no va a llegarse a una armonización, por lo menos a corto y medio plazo. Esta es la gran aportación de los gráficos vectoriales para paliar el problema de la resolución de pantalla, como veremos en el próximo capítulo.

La búsqueda del realismo ha motivado gran parte de la historia del rendering, el proceso de creación de imágenes mediante gráficos por ordenador. En un principio se empezó a trabajar con modelos de iluminación local, esto es, el estudio de la luz en una superficie. Posteriormente se dio un paso más al estudiar cómo influía la luz en un entorno, lo que se llamó modelos de iluminación global.

Sin embargo, a medida que avanzaban y mejoraban las prestaciones, aparece un nuevo problema [NPR02]: la eficiencia en la comunicación, entendida esta como la facilidad para transmitir información a través de la interacción usuario-máquina. Recientemente ha surgido una nueva búsqueda, para algunos autores más interesante que la del realismo, dejando a un lado el mundo físico que nos rodea. Esta búsqueda se basa en las ciencias cognitivas, y sus herramientas son básicamente el arte, el diseño gráfico y las técnicas de comunicación audiovisual. Estas herramientas son el pilar sobre el que se sustenta y estructura la información, que es el modus vivendi de los tiempos que corren.

Esta búsqueda ha creado una interesante área de estudio, denominada No Fotorrealística (del inglés Non-PhotoRealistic, NPR). Y, aunque ha carecido de unos contenidos organizados en un principio, dicho área de estudio ha cambiado nuestra forma de ver las cosas: nuevos interfaces, nuevas técnicas de rendering e impresión, etc. En definitiva, una nueva forma de comunicación.

En la web3D se intentan aunar dos características de dos entornos en un principio diferentes aunque no excluyentes, a saber: el fundamento de Internet es disponer de una información accesible en todo momento y de forma ubicua, y el las 3D es el de proporcionar una visión más similar a la que capta el ojo humano de su propio entorno.

Esto es totalmente extrapolable al mundo de los dispositivos móviles, dada la inminente convergencia entre web3D y las nuevas aplicaciones de entornos tridimensionales en estos aparatos.

Este trabajo no pretende ser un curso sobre diseño 2D y 3D. Se intentará ofrecer una visión general aplicada a los dispositivos móviles y, en particular, a la generación de entornos virtuales. Por entorno virtual se entiende un modelo tridimensional que usará el ordenador para su aplicación de RV [MOLLA06]: puede ser una estructura molecular para un químico, un edificio para un arquitecto o una mesa de mezclas para un DJ. Como toda aplicación de RV, en este caso particularizada a dispositivos móviles, la generación de las imágenes y el sonido debe ocurrir en tiempo real.

3. Fundamentos sobre gráficos vectoriales

En los gráficos vectoriales las imágenes se almacenan y representan por medio de trazos geométricos controlados por cálculos y fórmulas matemáticas, tomando algunos puntos y/o distintos elementos (vectores) de la imagen como referencia para construir el resto [GRADIG01]. Por ejemplo, una línea se define en un gráfico de mapa de bits mediante las propiedades de cada uno de los pixels que la forman, mientras que en un gráfico vectorial se hace por la posición de sus puntos inicial y final y por una función que describe el camino entre ellos. Análogamente, un círculo se define vectorialmente por la posición de su centro (coordenadas x,y) y por su radio (r).

Ilustración 1: Conceptos de gráficos vectoriales

Por lo tanto, las imágenes en los gráficos vectoriales no se construyen píxel a píxel, sino que se forman a partir de vectores, definidos matemáticamente. Esto es de suma importancia de cara a la resolución, en tanto que los gráficos vectoriales son independientes de ésta, ya que no dependen de una retícula de pixels dada. Por ello, tienen siempre la máxima resolución que permite el formato en que se almacena.

Otras ventajas que ofrecen los gráficos vectoriales, derivadas de su naturaleza matemática, son:

- Almacenan las imágenes en archivos muy compactos, ya que sólo se requiere la información (ecuaciones) necesaria para generar cada uno de los vectores. dado que no se ha de almacenar información para definir cada punto de la pantalla, sino una serie de fórmulas matemáticas.

- Permiten modificar el tamaño de las imágenes y de sus objetos componentes sin que se produzca pérdida de información, pues se actualizan de forma matemática todas las nuevas relaciones y posiciones de los elementos geométricos que las componen. Con ello, los cambios de tamaño de las imágenes vectoriales no afectan a la calidad de las mismas, apareciendo siempre con la misma nitidez.
- Cada objeto viene definido por sus propias fórmulas matemáticas y se maneja independientemente del resto, pudiendo escalarse, distorsionarse y cambiarse de forma o de posición sin afectar para nada los otros elementos del dibujo.
- Es posible un control independiente del color, tanto del contorno como del relleno, admitiendo la aplicación de texturas, degradados, transparencias, etc.
- Se puede controlar con gran precisión la forma, orientación y ordenación de los elementos.
- Cualquier efecto que se aplique a los objetos puede rectificarse en cualquier momento, ya que el dibujo es siempre editable. Esto no ocurre en las imágenes de mapas de bits, en las que una vez pintado un elemento ya no es posible modificarlo.

Y por supuesto, algunos inconvenientes de trabajar con gráficos vectoriales son:

- Un aspecto más frío que los gráficos de mapa de bits, con su contorno demasiado perfecto, que los hace a veces poco “naturales”.
- Su falta de eficacia para representar imágenes de tipo fotográfico. Aunque existe la posibilidad de vectorizar dichas imágenes, el resultado nunca es fidedigno.
- Cuando una escena presenta un alto grado de complejidad (esto es, un gran número de objetos y diversas interacciones entre ellos), puede volverse extremadamente difícil de manejar, consumiendo mucho tiempo de procesador.
- La dificultad que presenta para tratar algunos efectos, tales como la iluminación o la proyección de sombras.

Vistos los fundamentos de gráficos vectoriales con sus pros y contras, a lo largo de este trabajo se discutirán algunas técnicas para llevarlos a 3D. Pero antes hay que comentar algunos aspectos que rodean estas y otras definiciones.

4. Requisitos 3D vs. requisitos RV clásicos

Se ha venido marcando una frontera entre los sistemas convencionales y los sistemas de RV tal que en los primeros percibimos la información como una verdad representada mediante caracteres, imágenes, audio o incluso gráficos tridimensionales, y en los segundos nos encontramos cara a cara con la información, tal y como si compartiese el mismo espacio en el que nos encontramos. Nos hallamos inmersos en esa realidad. Esta definición se ha impuesto durante mucho tiempo, pese a que, en última instancia, la frontera depende más bien de los gustos de cada cual y de su capacidad de percepción, que de una ley que marque la diferencia. Un videojuego de lo más sencillo puede resultar una experiencia totalmente realista para un niño de seis años, mientras que la más sofisticada máquina con los más complejos dispositivos puede no dejar de ser un videojuego para un adulto escéptico.

En todo desarrollo de un sistema de RV se diferencian dos grandes etapas: el diseño del sistema y el desarrollo de la aplicación. Básicamente, en el primer caso se han de estudiar los requerimientos que vamos a imponer al sistema, mientras que en el segundo caso se procede a construir el mundo virtual.

A continuación se adjunta un esquema propuesto por L.M. Del Pino [DELPINO95] y que es extensible al desarrollo de todo sistema de RV, si bien de modo puramente informativo.

A. Requerimientos

A.1. Tipo de sistema
- Marco: Inmersivo, proyectivo o de sobremesa
- Mono o multiusuario

A.2. Presupuesto
- Diseño sistema
- Desarrollo aplicación
- Personal involucrado
- Costes de uso

A.3. Soluciones Hardware/Software
- Plataforma
- Periféricos
- Programas

A.4. Objetivos de la aplicación RV
- Alcance del Mundo Virtual
- Nivel de interacción : Dinámico, navegación o paseo virtual.
- Precisión en el tratamiento de los datos

A.5. Usuario Potencial
- Conocimiento/entrenamiento previo
- Peculiaridad física/psíquica
- Limitaciones de uso

A.6. Imagen
- Visión estereoscópica
- Número de Objetos
- Resolución
- Velocidad (fps) y Latencia
- Modelo de Iluminación

A.7. Sonido
- Tipo
- Síntesis de voz/sonido

B. Desarrollo

B.1. Artístico: Creación
- Escena y Objetos: Métodos constructivos o algorítmicos
- Texturas
- Animación: Interpolación, Cinemática Inversa o Sistemas de Partículas
- Sonido

B.2. Funcional: Programación
- Interacciones
- Presentación al usuario
- Optimización

Tabla 1: Requisitos, a grandes rasgos, de un sistema de RV

Cabe decir que cada aplicación tendrá sus propios requerimientos, por ejemplo: ¿es necesario incluir sonido 3D? ¿hay que realizar una detección de posición/orientación del usuario? En entornos tan particulares como son la web o los dispositivos móviles cabrían preguntarse estas y otras preguntas.

5. Evolución y Estado del Arte

El punto de partida histórico lo marcaron, como no, las primeras investigaciones de la DARPA en 1962. En 1989 Tim Berners-Lee crea Internet tal y como la conocemos hoy día, y en 1990 aparece el primer programa cliente (navegador y editor). Una revolución sin precedentes en el mundo de la informática y las comunicaciones estaba a punto de comenzar.

5.1. Evolución de las aplicaciones para la web

Hasta la aparición de Macromedia Flash y los ficheros SWF, allá por el año 2000, la inclusión directa de gráficos vectoriales en navegadores web no era posible. Su introducción en las páginas web supuso un considerable y revolucionario avance, ya que permitieron incluir gráficos de tamaño escalable sin pérdida de calidad, muy útiles en, por ejemplo, creación de logotipos y planos de CAD o GIS. Hoy día el formato SWF posee una penetración de mercado del 98%, es multiplataforma y de SDK abierto -cualquiera puede implementar su reproductor o programa de desarrollo sin tener que pagar royalties-.

Pese a que SWF se ha convertido en un estándar de facto en la web en cuanto a gráficos vectoriales 2D se refiere, tiene en SVG (Scalable Vector Graphics) su claro representante. De hecho, SWF es una aplicación de la especificación SVG en formato binario. Su estructura es más compleja que SWF, está basado en XML y es totalmente Open Source. Se creó en 2000, muy poco antes de aparecer SWF, pero ha evolucionado tan lentamente que prácticamente todos los desarrolladores web han escogido el formato de Macromedia, y quizás debido también a las facilidades que daba la herramienta Flash, para crear sites con nuevos contenidos multimedia.

La idea de incluir material tridimensional en la web no es nueva. Desde hace más de una década existen iniciativas para formatos de archivo que representen objetos y mundos tridimensionales. En general estas iniciativas han fracasado debido al gran tamaño de los archivos o a la gran cantidad de recursos de sistema requeridos para reproducirlos. Aún es posible encontrar formatos tales como VRML o QTVR en sitios web, pero definitivamente no han prosperado lo suficiente [COPESA00].

Las posibilidades parecen ser mejores ahora, con iniciativas de varias compañías comerciales y la incursión de otros desarrolladores para exportar 3D en diferentes formatos compatibles. ~~Macromedia~~ Adobe, además de sus tradicionales programas para el desarrollo de sitios y la creación de animaciones vectoriales e interactividad, ha

producido varios plug-ins gratuitos para visualizar contenido Flash y contenido Shockwave. Este último, junto con su herramienta de desarrollo multimedia Director, permite crear y reproducir contenido 3D de bajo peso en la web.

Ilustración 2: Entornos 3D creados para web

En 2004 surge el sucesor de VRML, el estándar abierto X3D. Este formato, que parece ir ganando terreno en el campo de las tres dimensiones, es capaz de renderizar escenas empleando sintaxis XML, pero actualmente la tecnología es aun incipiente. El tamaño de las escenas, si bien puede considerarse reducido, sigue siendo algo excesivo para algunas aplicaciones, tanto en Internet como en redes móviles. Su auge sin duda llegará, por un lado, cuando se comercialicen tarjetas hardware para realizar los cálculos matemáticos que toda aplicación 3D necesita, y por otra parte cuando sea posible aplicar texturas SVG a estos gráficos X3D.

5.2. Evolución de las aplicaciones para dispositivos móviles

Paralelamente a esta evolución de la web, se tiene la evolución de las aplicaciones en dispositivos móviles [BERBEL00]. Aplicando la Ley de Moore, nos damos cuenta de que dicha evolución es de aproximadamente 9 meses: esto es, la mitad que en los ordenadores de sobremesa.

En 1982 la Conferencia de Administraciones de Correos y Telégrafos (CEPT) formó el grupo de estudio Groupe Spécial Mobile (GSM) para que desarrollase un conjunto de estándares para una futura red inalámbrica de comunicaciones móviles paneuropea. El servicio comercial de GSM empezó a mediados de 1991 y por 1993 ya existían 36 redes en 22 paises. GSM permite la conexión a Internet a una velocidad de 9600 bits por segundo, a través del protocolo WAP (Wireless Application Protocol). Esta velocidad no es suficiente para acceder a la mayoría de las actuales páginas web,

pero permite acceder a algunos servicios de Internet, como el correo electrónico, y también a contenidos e información especialmente pensada para esta baja velocidad de transmisión.

La evolución de GSM es GPRS (Global Packet Radio Service). GPRS permite velocidades de transferencia de hasta 112 kilobits por segundo. Esta velocidad es tres o cuatro veces superior a la que disponemos en nuestro ordenador personal equipado con un módem estándar v90. Permite la navegación por Internet e incluso la transferencia de video de baja calidad. La diferencia básica entre GSM y GPRS, a parte de la velocidad, es que con GPRS estamos siempre conectados (el concepto conocido como “allways-connected”) y, en principio, sólo pagaremos por la información transmitida, no por el tiempo de conexión.

La tecnología posterior a GPRS se llama UMTS (Universal Mobile Telecommunications System). UMTS permite una velocidad de transferencia de hasta 2 megabits por segundo (velocidad 50 veces superior a la tenemos con un módem estándar). La implantación de UMTS se llevará a cabo durante los próximos años y será cuando se produzca la gran difusión de la Internet móvil.

La tecnología Wireless Lan se consolida en 2004. Hoy día supone para algunos una amenaza para las operadoras de telefonía móvil y para otros el complemento perfecto de la idea "All-IP". Tiene una cobertura de 50 a 100 metros y una gran velocidad de transmisión (hasta 54 megabits por segundo). Permite la conexión a la red de dispositivos portátiles y fijos dentro de los edificios. Esta tecnología puede ser una alternativa o un complemento a las actuales redes de los edificios basadas en cables.

Ese mismo año la tecnología Bluetooth se considera totalmente operativa, pudiendo sincronizar todas nuestras fuentes de información sólo con acercar los dispositivos en un radio de aproximadamente 10 metros. Bluetooth es un estándar de la industria y aparece gracias a los acuerdos entre las principales compañías tecnológicas como 3Com, Ericsson, Intel, IBM, Lucent, Microsoft, Motorola, Nokia y Toshiba. Proporciona una velocidad de transferencia de hasta 720 kilobits por segundo y elimina la necesidad de cables para conectar dispositivos fijos, portátiles, móviles y periféricos (impresoras, escáneres, etc).

Con todo esto, las aplicaciones para dispositivos móviles han evolucionado al mismo ritmo que aparecían los nuevos estándares, que como vemos no son pocos. Típicamente estas aplicaciones se centraban en la gestión de noticias, orientación de flotas de vehículos y servicios de localización. Hoy día el ámbito se ha disparado enormemente, encontrando prácticamente las mismas soluciones para empresa y particulares que se ofrecen a través de Internet.

Ilustración 3: Aplicación WAP y nuevo terminal con Windows Mobile

5.3. El panorama actual

Actualmente nos hallamos ante un salto cualitativo comparable al que en su momento produjo el ordenador personal. La manera que utilizaremos y nos conectaremos a la red puede cambiar drásticamente en los próximos años. Este salto vendrá dado por tres factores: utilizaremos nuevos dispositivos para acceder a la red, se crearán nuevas redes inalámbricas y aparecerán nuevos servicios en la Internet móvil.

Los servicios móviles no son una réplica de los servicios web en versión reducida. Tenemos que pensar en términos de movilidad: simplicidad, facilidad y ergonomía. Asimismo, la Internet móvil no es sólo Internet en el móvil.

Todo apunta a que durante los próximos años los terminales móviles se transformarán en completos dispositivos audiovisuales que funcionarán con redes sin cables y de alta velocidad, con capacidades sensiblemente cercanas a las de los ordenadores de sobremesa.

Hoy por hoy, mientras llega este panorama tan alentador, la propuesta que aquí se hace se centra en emplear exclusivamente gráficos vectoriales para crear entornos y aplicaciones 3D. Para ello se expondrán diversos métodos de simulación (conocidos como 2.5D o Fake3D) e implementación de “verdaderas” 3D (motores de renderizado) que, teniendo en cuenta las limitaciones a las que están sujetos los dispositivos móviles, resultan suficientes y de inmediata aplicación. Una web que implementa esta tecnología es el otrora conocido sitio desarrollado por Florian Krüsch (mx3d.com).

Ilustración 4: Ejemplos de web3D con gráficos vectoriales

Cabe señalar que los algoritmos que se van a presentar en este trabajo no requieren grandes cálculos ni altos tiempos de computación: característica muy deseable en este tipo de aparatos.

6. Simulación de entornos virtuales en dispositivos móviles

De los cuatro factores que cita [DELPINO95], existen dos de vital importancia para nuestra aplicación: la capacidad sintética (el mundo virtual se genera en tiempo real según la posición del usuario) y la interactividad (el entorno responde a las acciones y movimientos del usuario). Estos factores son de obligada inclusión en todo entorno de tres dimensiones. Los otros dos que menciona el autor son los que se discuten a continuación.

6.1. Tridimensionalidad

El mundo de los videojuegos se encuentra hoy día muy ligado al mundo de la RV y viceversa. Para que un sistema pueda ser llamado de RV es necesario que proporcione la dimensión de profundidad, que el cerebro humano emplea para deducir las posiciones y tamaños de los objetos. Si el sistema calcula algún tipo de proyección de los objetos sobre la escena, entonces éstos tienen asociada una posición "en profundidad", y aparecerán los fenómenos asociados a los mundos de 3D, que son principalmente la perspectiva lineal, el sombreado y el paralaje.

En 2D, las operaciones de visión transfieren puntos bidimensionales en el plano coordenado mundial a puntos bidimensionales en el plano de coordenadas del dispositivo. Estas coordenadas de dispositivo normalizadas se convierten después en coordenadas de dispositivo y el objeto se despliega en el dispositivo de salida, que en nuestro caso es la pantalla del móvil. La siguiente figura ilustra este concepto.

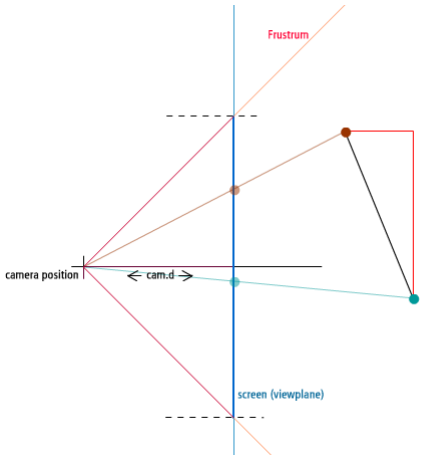


Ilustración 5: Proyección de una escena 3D sobre la pantalla (2D)

Más adelante se incidirá en la nomenclatura y naturaleza de los términos que aparecen en esta figura.

Es importante observar el hecho que, una vez que se ha efectuado la proyección de la imagen sobre la pantalla, lo que pasamos a tener es una imagen en 2D, pese a lo cual somos perfectamente capaces de reconocer la naturaleza 3D del mundo virtual.

6.2. Ilusión de realidad

Al hablar de entornos 3D no se trata sólo de los sistemas que imitan mundos reales (simuladores), sino también de aquellos realizados de forma que el usuario se crea esta ilusión. El sistema generado no es real, sino ficticio, y definido de tal modo que al ser humano le parezca real, su realidad. El que el usuario pueda sentir como real el mundo simulado depende tanto de factores físicos como psicológicos.

Los factores físicos están relacionados con el aspecto del mundo virtual, entendiendo por “aspecto” las percepciones visuales, sonoras, táctiles, etc. El aspecto será tanto más real cuantos más sentidos sea capaz de estimular el sistema al usuario. Algunos mecanismos muy empleados son la estereoscopía o la localización de sonido.

Los factores psicológicos están relacionados con la naturaleza del mundo virtual, tal como el usuario la percibe. Los principales factores son el nivel de inmersión e interactividad, la facilidad de navegación y el uso de periféricos de entrada. Cabe decir que en un sistema multiusuario la sensación de realidad se intensifica, dado de que hay otros objetos en el sistema (otros usuarios) con un “comportamiento” que de hecho es real.

En general, es el propio comportamiento del mundo virtual lo que el usuario va a

valorar como verosímil o no. Resulta curioso observar que la ilusión de realidad se intensifica cuanto más fantástico es el mundo virtual: cuantas menos equivalencias con el mundo real, menos referencias tiene el usuario para cuestionar la "realidad" de lo que percibe, y pierde importancia la calidad de la imagen en sí.

6.3. Colores

Desde siempre el uso del color ha sido simbólico y emocional, basado en la mitología y la religión, y no necesariamente con fines estéticos. Por lo menos hasta el Renacimiento. Pese a que cada persona puede reaccionar de forma diferente a un determinado color, lo que sí es cierto es que los colores evocan emociones psicológicas y fisiológicas. Por ello, su importancia en la construcción de entornos tridimensionales es fundamental.

Normalmente se hace una clasificación en colores cálidos, fríos y neutros. Estos últimos se refieren a la escala de grises, mientras que los primeros se asocian al espectro del rojo (rojo-naranja-amarillo) y los segundos al rango de azules (verde-azul-violeta). Esto es interesante a la hora de inferir en el usuario, reforzando la capacidad de inmersión en el entorno de la aplicación. Por ejemplo, usando colores fríos estamos induciendo calma, mientras que con tintas cálidas estimulamos el sistema nervioso. Cuanto más saturado es un color, más excitación produce en el usuario y viceversa. Algunos colores evocan asociaciones más específicas: el rojo se asocia a la pasión, violencia, sexo, creatividad, coraje, o es usado como símbolo universal para denotar un error o detenerse (ante una señal de stop, por ejemplo).

6.4. Características básicas de las tres dimensiones

Los siguientes apartados se van a ilustrar con un ejemplo de las técnicas para implementar 3D en dispositivos móviles. Se intentará aproximarse a un sistema de RV, en base a las definiciones expuestas anteriormente y las que a continuación se presentan.

Todos estos ejemplos pueden ser consultados en el directorio ***/ejemplos*** que acompaña la versión digital de este documento.

6.4.1. Definición de vista 3D estática: el origen de coordenadas (0,0,0)

Una vez que tenemos un modelo matemático sobre el que trabajar (los gráficos vectoriales), aún nos queda por resolver el impedimento de representar modelos 3D en dispositivos 2D. De nuevo, recordar que la pantalla es una superficie 2D donde se proyecta la luz, generando las imágenes. Así pues, toda sensación de profundidad y 3D es puramente una ilusión. Incluso en las películas de cine la profundidad debe ser creada, aunque los actores sean materia tridimensional.

Podemos suponer que en una vista 2D existe teóricamente un eje Z. El problema es que nosotros vemos un plano XY paralelo a nuestro campo de visión, y por lo tanto sólo recibimos información de movimiento en dichos ejes. La forma más simple de realimentación visual es crear la sensación de profundidad, “descubriendo” ese eje Z.

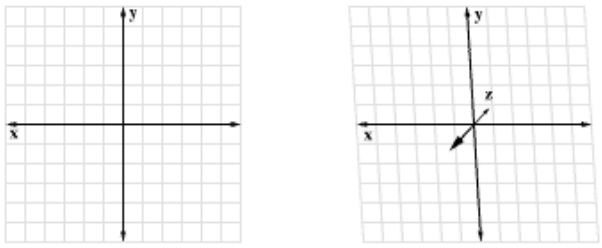


Ilustración 6: El eje Z define las 3D en un plano XY

Toda escena 3D está basada, pues, en sus 3 ejes coordenados XYZ. Esta será la referencia de toda representación que se realice posteriormente. Al trabajar con gráficos vectoriales, el punto (0,0) es la esquina superior izquierda del monitor, por convenio universal. La coordenada Z=0, dado que debe ser normal al plano XY, estará por tanto en ese mismo punto. Con esta consideración, habrá que reposicionar (o no) los elementos de nuestra escena al centro visual de la pantalla.

6.4.2. Profundidad

La sensación visual de tridimensionalidad más evidente es la profundidad. ¿Cómo determinarla a partir de 2D? La solución, como ya adelantábamos anteriormente, son las proyecciones. La proyección en paralelo sirve para obtener vistas exactas de varios lados de un objeto, pero no ofrece una representación realista de un objeto tridimensional. Una proyección en perspectiva produce vistas realistas pero no preserva las dimensiones relativas. Las líneas distantes se proyectan como menores que aquellas que están más próximas al plano de proyección.

La convergencia de líneas en el horizonte son muy útiles, pero pueden ser alteradas

dependiendo de la lente de la cámara -siempre y cuando simulemos una lente de cámara, claro está-. Por ejemplo, un gran angular deforma la imagen para conseguir el máximo campo de visión, haciendo que objetos muy cercanos al punto de visión parezcan más lejanos.

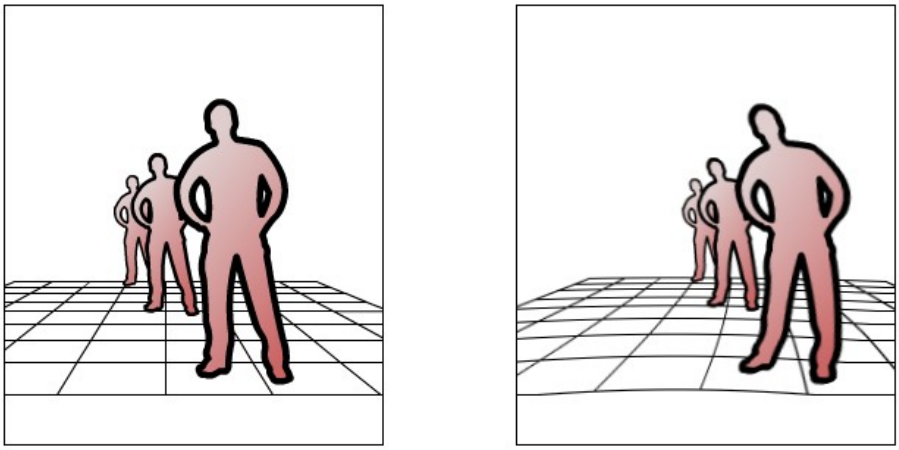


Ilustración 7: Vista estática simple y vista con lente simulada

Muchas veces se compara la pantalla con una ventana a través de la cual vemos el mundo [PIXEL96]. El usuario hace comparaciones intuitivas para determinar lo que está viendo. Algunas de estas comparaciones son sencillas: si observamos dos figuras semejantes y una de ellas es más grande, seguramente estará más cerca. Si un objeto se superpone con otro, es probable que estén muy juntos. Decimos “seguramente” y “probablemente” porque el cerebro humano utiliza sus propios mecanismos para establecer medidas y distancias.

6.4.3. Objetos y Fondo

El fondo (background) que usemos en nuestra aplicación va a jugar un papel fundamental en la recreación más o menos verosímil del mundo virtual. Así, en la ilustración anterior se acentúa la sensación de profundidad dibujando una rejilla en perspectiva.

Algunas técnicas para simular ambientes de exterior (outdoor) consisten en marcar una línea de horizonte o dibujar el plano de referencia con un color degradado.

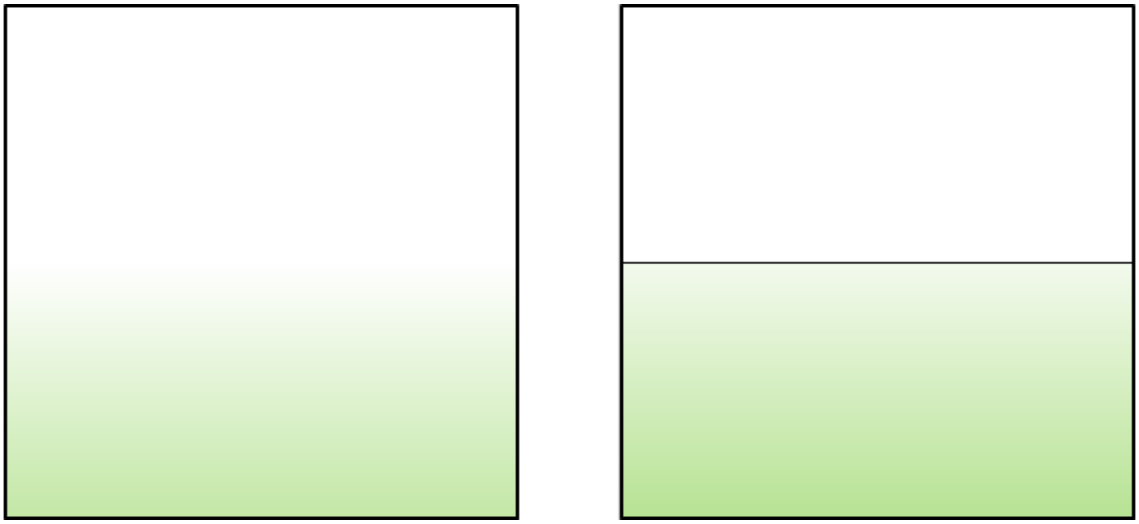


Ilustración 8: Fondos para ambientes exteriores

Es importante destacar que el fondo se encuentra siempre a menor nivel de brillo que los objetos en primer plano. En condiciones normales de iluminación, el fondo presenta un nivel de brillo uniforme. En el caso de incluir iluminación lateral, el histograma muestra un gradiente del fondo, pero siempre por debajo del nivel de brillo de los objetos enfocados.

6.4.4. Overlapping

Inevitablemente, al trabajar en 2.5D ocurre un fenómeno de solapamiento o transposición entre capas, conocido como overlapping. Esto ocurre porque en gráficos vectoriales dos objetos que se están escalando dinámicamente son considerados independientes y están situados en dos niveles de profundidad (capas) distintos. Por ello, al modificar la profundidad, cada objeto reaccionará modificando su escala y pueden llegar a superponerse por un corto periodo de tiempo que, aunque despreciable, puede influir en ciertas aplicaciones.

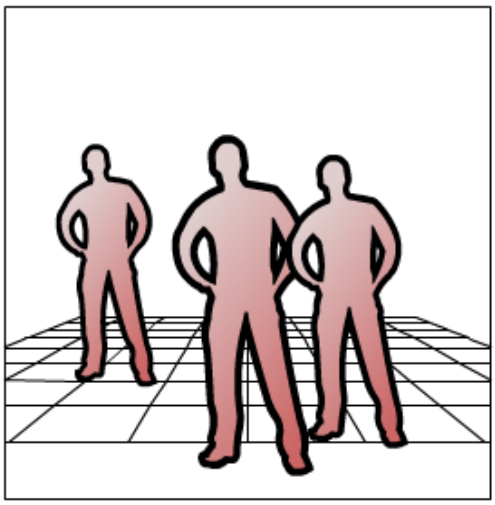


Ilustración 9: El fenómeno de overlapping

Una forma de evitar este fenómeno es incrementar la tasa de fotogramas por segundo (fps). Teóricamente a partir de 60 fps el ojo humano no distingue cambios apreciables, así que podríamos enviar la escena a, por ejemplo, 120 fps. Por supuesto siendo conscientes que ese aumento en la velocidad de imágenes va a influir en el rendimiento del resto de parámetros de nuestra aplicación, y llega a ser inviable en algunos dispositivos móviles. El otro método (más apropiado) se discutirá más adelante, en el Capítulo dedicado al rendering de la escena.

6.5. El concepto de Cámara

6.5.1. Movimiento de objetos vs. movimiento de cámara

Hasta ahora los ejemplos precedentes se han realizado para una vista fija en un aparente entorno 3D. Esto no tiene más utilidad sino definimos un punto de vista, el cual se consigue simulando una cámara. Realmente una cámara representa un desplazamiento de los objetos en el entorno 3D, así que es exactamente lo mismo afirmar que el origen es estático y que los objetos se mueven en torno a él, o que la escena permanece estática y es la cámara la que se va desplazando por la misma. De hecho, dado que una pantalla es un entorno 2D, técnicamente las cámaras no se mueven: “sólo” reposicionan los objetos en el espacio para mostrar lo que pasaría en el mundo real.

6.5.2. Frustum

El Frustum es el campo de visión. Se representa por un volumen 3D, una pirámide truncada, que abarca todo lo que es visible, entendiendo como visible todos aquellos elementos que se verían en pantalla si otros elementos no los taparan. Esta pirámide tiene, pues, 6 lados y se representa mediante 6 planos infinitos. Todo lo que se encuentre dentro de esos seis planos es visible. El Frustum está definido por un FOV (Field Of View o ángulo de visión) y por las distancias de los planos frontal o cercano (zNear) y posterior o lejano (zFar), cuya posición es relativa a la posición de la cámara.

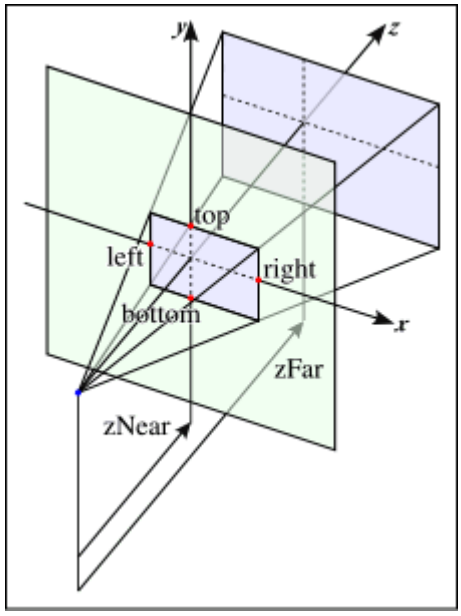


Ilustración 10: Volumen de visión (frustum)

La distancia zNear se conoce como distancia focal, y sirve para determinar el escalado de los objetos en perspectiva. Hay que procurar no posicionar objetos por debajo de `z=0`. Si en el eje Z se alcanzan valores negativos, tendríamos en pantalla la imagen especular del objeto moviéndose en dirección contraria al sentido que llevaba inicialmente.

La forma del volumen afecta la forma en que los objetos son proyectados en la pantalla, debido a que el plano frontal es interpretado como la pantalla. La forma más común de proyección es en perspectiva. La proyección en perspectiva es la responsable de hacer que un objeto cercano a la cámara se vea más grande que un objeto alejado de ella.

6.5.3. Aspect Ratio

El Aspect Ratio es la relación de aspecto que tiene la ventana de proyección, es decir es el cociente del ancho/alto en pixels. Dado que el FOV se expresa en radianes, si

por ejemplo especificamos `PI/2` de FOV, la cámara podrá "mirar" 90 grados de amplitud sin importar la resolución que tenga la pantalla, por tanto el Frustum va a tener una amplitud de 90 grados entre sus planos laterales. Como el Frustum está en coordenadas de cámara, se debe asumir que la posición de la cámara es el origen (0,0,0) y que se está mirando en la dirección del eje Z. La siguiente figura ilustra la ecuación:

```
aspectRatio = tan(fovX/2)/tan(fovY/2)
```

Ilustración 11: FOV vertical (plano ZY) y Aspect Ratio

A la hora de redibujar los objetos en pantalla podemos encontrar problemas con polígonos que están muy cerca de la cámara y que son bastante más largos que los elementos de su alrededor. Considérese esta figura como ejemplo:

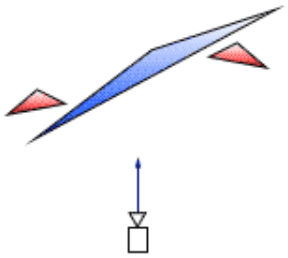


Ilustración 12: Posicionamiento complicado de polígonos

En este caso tan particular, es complicado determinar la posición de cada elemento en relación a la pantalla. Conviene definir un scale ratio (relación de escala), para evitar perspectivas extrañas, del tipo:

```
scaleRatio = distanciaFocal/(distanciaFocal+z)
```

Como comentábamos anteriormente, hay que procurar no posicionar objetos por debajo de `z=0`. Si en el eje Z se alcanzan valores negativos, tendríamos en pantalla la imagen especular del objeto moviéndose en dirección contraria al sentido que llevaba inicialmente.

6.5.4. Traslaciones y rotaciones

Para cambiar el punto de vista hay que efectuar una traslación y/o rotación angular de la cámara. El desplazamiento de la cámara se denomina panning y la rotación se descompone en los tres ejes coordenados: en X se llama pitch (cabeceo), en Y es yaw (giro) y en Z es roll (balanceo).

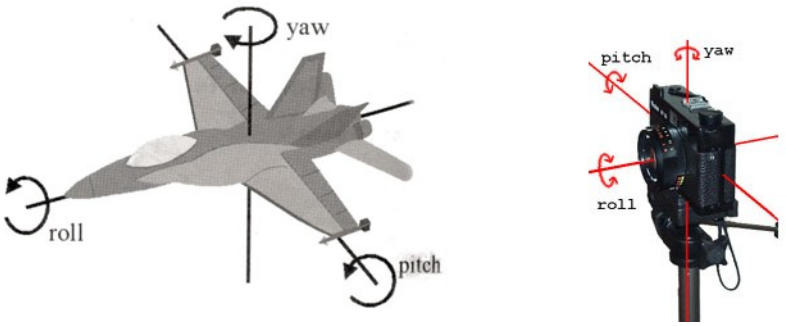


Ilustración 13: Rotaciones en objetos y cámara

La herramienta matemática que debemos emplear para conseguir este efecto es, como ya nos estaremos imaginando, la trigonometría.

Recordemos que lo que ocurre es una transformación de todos los objetos alrededor de una referencia. Podemos representar un volumen tridimensional centrado en la pantalla si dicha referencia es el centro de la aplicación y no el punto de vista (la cámara). Esto puede ser útil para, por ejemplo, manipular objetos. Al emplear la cámara como referencia tendremos la sensación de estar desplazándonos por el entorno, lo cual es requerido en los paseos virtuales.

Una rotación en el eje Z sobre un objeto se puede traducir en un desplazamiento en los ejes X e Y proporcional al ángulo que forman. Esa proporcionalidad es el coseno y el seno de dicho ángulo, respectivamente, y es lo que (por ejemplo) permitirá avanzar o retroceder en la escena.

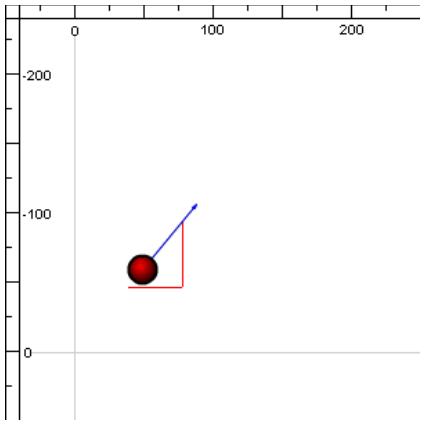


Ilustración 14: Desplazamiento basado en trigonometría

Puesto que la trigonometría juega un papel fundamental en la caracterización de estos entornos 3D, en el siguiente Capítulo se van a exponer unos conceptos necesarios para poder poner en práctica toda la teoría.

6.6. Herramientas matemáticas

6.6.1. Trigonometría

Dado que esta rama de las matemáticas estudia las relaciones entre ángulos, vamos a ver cómo aplicarla en los cálculos de 3D. Suponiendo una recta definida en 2D con uno de sus extremos en el origen y el otro en una coordenada genérica (x,y), podemos definir sus proyecciones cartesianas sobre los ejes coordenados como se observa en la siguiente figura.

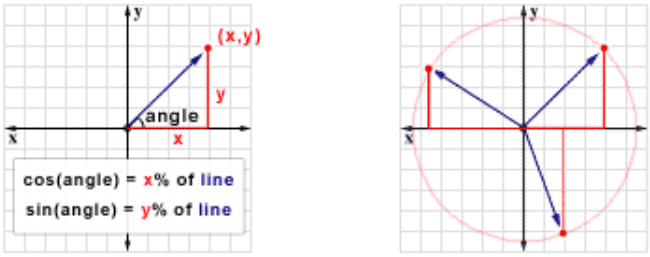


Ilustración 15: Principios básicos de trigonometría

Al rotar esta recta y variar por lo tanto su ángulo, las proyecciones cambiarán su tamaño dentro del rango `[0,L]` siendo `L` la longitud de dicha linea, la cual representa la hipotenusa de un triángulo definido en sentido horario por los segmentos `xLy`. Otra forma de referirse a `L` es como el radio de la circunferencia que describe (de hecho, es la forma más común de notación). Las relaciones básicas que debemos tener en mente son:

```
x = cos(angulo) * L;
y = sin(angulo) * L;
angulo = arctan (y / x);
L = radio = sqrt(x² + y²);
```

Estas proyecciones serán las que representen la posición y rotación de los objetos en la escena. Las rotaciones se expresan en radianes, lo cual habrá que considerar para ajustar algunos parámetros de nuestra aplicación. Trasladar estos cálculos a una dimensión más es directo, teniendo siempre en cuenta el origen (0,0,0) y los offsets en cada eje coordenado:

```
// rotación en el eje X
y = origenY + cos(anguloEjeX)* radio;
z = origenZ + sin(anguloEjeX)* radio;

// rotación en el eje Y
x = origenX + cos(anguloEjeY)* radio;
z = origenZ + sin(anguloEjeY)* radio;

// rotación en el eje Z
x = origenX + cos(anguloEjeZ)* radio;
y = origenY + sin(anguloEjeZ)* radio;
```

Cada vez que se efectúa una rotación en uno de los ejes, habrá que recalcular el nuevo radio `L`, por lo que estas ecuaciones anteriores sólo serán válidas para mover objetos en el entorno. Si lo que queremos es desplazarnos en él, habrá que reescribir dichas expresiones no en función del ángulo, sino en función del cambio del ángulo:

```
// rotación en eje X: pitch
y = cos(anguloEjeX)*y – sin(anguloEjeX)*z;
z = sin(anguloEjeX)*y + cos(anguloEjeX)*z;

// rotación en eje Y: yaw
z = cos(anguloEjeY)*z - sin(anguloEjeY)*x;
x = sin(anguloEjeY)*z + cos(anguloEjeY)*x;

// rotación en eje Z: roll
x = cos(anguloEjeZ)*x - sin(anguloEjeZ)*y;
y = sin(anguloEjeZ)*x + cos(anguloEjeZ)*y;
```

Con estos conceptos, muchos autores han escrito sus propios motores de rendering (3D engines). Algunos ejemplos se incluyen en la versión digital de este documento.

6.6.2. Optimización de parámetros

Los dispositivos móviles tienen mucha menos memoria y capacidad de procesado que los ordenadores tradicionales de sobremesa y portátiles. La velocidad de la CPU varía considerablemente entre los diferentes modelos. Por lo tanto, es importante considerar varios parámetros de optimización y rendimiento en toda aplicación para dispositivos móviles.

El mayor problema que afectará a nuestra aplicación es determinar el frame rate, esto es, los fps (fotogramas por segundo) a los que se ejecutará el programa. En muy pocas ocasiones se consiguen los fps de diseño, dado que cada terminal tiene un ratio nativo. Por ejemplo, en los teléfonos de Symbian Series 60v2 el refresco de pantalla es de 1/64 segundos. Esto significa que las funciones que usemos pueden llamarse a intervalos de 1/64, 2/64, 3/64, 4/64, 5/64, y así sucesivamente. De esta forma, si hemos preparado nuestra aplicación para funcionar a 20 fps, teóricamente los eventos deberían ser ejecutados por el dispositivo cada 1/20 segundos. Para los teléfonos de este ejemplo, 1/20 se encuentra entre los intervalos de 3/64 y 4/64, que son los que proporciona dicho terminal. Dado que la ejecución de instrucciones es conservativa, nuestra animación se ejecutará a una velocidad de 4/64 segundos, es decir, a 16 fps. Este "ajuste" del 20% se percibe como pérdida de rendimiento, haciendo por ejemplo que una suave transición que habíamos creado se convierta en algo más que un mero refresco de pantalla.

A continuación, se presentan dos listas de aspectos a recordar de cara al diseño de aplicaciones móviles basadas en gráficos vectoriales. Se deben tener en mayor consideración en tanto en cuanto la complejidad del código se dispare. Cabe señalar que algunas de estas recomendaciones son perfectamente extrapolables para diseñar otro(s) servicio(s) móvil(es) en diferentes lenguajes de programación.

Este primer checklist hace referencia al tratamiento explícito de gráficos vectoriales, mientras que el segundo sugiere algunas técnicas de programación.

- Típicamente, al crear una aplicación con gráficos vectoriales, el motor de render puede emplear tres niveles de calidad diferente para renderizar los objetos (`LOW`, `MEDIUM` y `HIGH` quality). Dependiendo de la situación, podemos ir alternando entre uno y otro nivel.
- Evitar el abuso de transparencias, gradientes y máscaras; son operaciones que hacen trabajar mucho el procesador.
- El uso de líneas y ángulos rectos es más eficiente computacionalmente que los círculos y curvaturas. Aunque ello no quita, por supuesto, que se dejen de utilizar splines o curvas bezier.
- Las líneas de contorno (strokes) incrementan notablemente el número de

vectores a renderizar.

- Cuando queramos incluir mapas de bits (bitmaps), conviene usarlos a la misma resolución que nuestra aplicación. Es ineficiente importar imágenes a gran resolución y después disminuir su escala. Asimismo, los formatos de imagen que usan compresión con pérdidas (tipo JPG) restan rendimiento: se recomienda emplear en su lugar el formato PNG.
- La gran mayoría de los teléfonos móviles actuales no soportan colores de más de 16 bit (miles de colores). Por lo tanto es recomendable diseñar nuestra aplicación como mucho a esta resolución, mejorando así el rendimiento de la misma.
- La tasa de frames por segundo no debe superar los 15 - 30 fps, pese a que existen dispositivos que soportan mayores ratios, pero no son mayoría.
- Si un código tarda más de 1 segundo en ejecutarse, se recomienda dividirlo en múltiples instantes de ejecución, acordes a la tasa de refresco del dispositivo.

La siguiente lista de sugerencias se basa en el principio de “simplifica y vencerás”, con el ánimo de mejorar la aplicación a nivel de programación. Es recomendable seguirla, siempre por supuesto en la medida de lo posible.

- Como ya vimos anteriormente, para obtener una proyección en perspectiva de un objeto tridimensional, se proyectan los puntos a lo largo de líneas de proyección que se interceptan en el centro de proyección. Puede elegirse cualquier posición para el centro de proyección, pero la elección de una posición a lo largo del eje Z facilita los cálculos en las ecuaciones de transformación.
- Para conversiones angulares de radianes a grados es computacionalmente más eficiente el usar el multiplicador PI/180 en lugar de 2*PI/360. A la inversa, para convertir de grados a radianes usaremos el factor 180/PI.
- Es conveniente agrupar en clases o librerías las acciones que van a realizar los objetos, teniendo así ordenada de forma sistemática nuestra aplicación.
- En ocasiones podemos reutilizar los resultados de un cálculo, en lugar de volver a recalcularlo todo. Parece algo obvio, pero es innumerable la de ocasiones en que no se hace así.
- En general, la operación de sustracción (resta) es más rápida computacionalmente que la de adición (suma). Por esta misma razón, los decrementos de variables son mejores que los incrementos.
- Todos los cálculos de transformaciones y posterior reposicionado deben hacerse al mismo tiempo. Hay autores que realizan un primer ciclo para ver qué objetos han variado su posición, y un segundo ciclo para asignarles su posición en pantalla. Esto puede ralentizar la aplicación cuando estamos usando muchos elementos en la misma escena.
- Los nombres de las variables que usemos deben ser descriptivos, sí, pero lo

más cortos posibles. Sobretodo si vamos a acceder a ellas muchas veces a lo largo del código. Es conveniente registrar las variables en modo local en vez de global, pues el rendimiento es más eficiente.

- Hay que evitar guardar valores de un elemento en un array si se puede acceder a ellos como una propiedad de objeto. Por ejemplo, prácticamente todo software gráfico ofrece información acerca de las coordenadas de un objeto, su visibilidad o su transparencia.
- Hay que tratar de mantener el código lo más limpio y claro posible, con vistas a una mejor comprensión (pasado cierto tiempo), para trabajos en grupo, etc.

6.7. Rendering

6.7.1. Shading y shadowing

Se ha preferido referirse a la técnica del sombreado por su equivalente del inglés para evitar la ambigüedad del lenguaje que supone el método de aplicar sombras a los objetos (dicha técnica se denomina, de nuevo con su traducción inglesa, shadowing). Se entiende por shading la forma de colorear los objetos, tanto en brillo como en saturación, que toman al ser presentados en pantalla. Sin tener ahora en cuenta los significados del color que mencionábamos en el Capítulo 6.3, esta técnica nos puede proporcionar gran información acerca de la profundidad de la escena.

La separación en planos de luz es una de las técnicas más usadas para crear la sensación de 3D y espacialidad. Esto se acentúa intercalando áreas u objetos oscuros entre dichos planos. Dada la naturaleza ondulatoria de la luz, podemos implementar un simple algoritmo que relacione la intensidad luminosa de manera inversamente cuadrática a la distancia de la fuente emisora. En el Capítulo 6.8.2 se discute la ecuación en cuestión a emplear, que de forma breve es:

$$Intensidad = 1 / distancia^2$$

El uso de sombras proyectadas nos ayuda a determinar la proximidad de los objetos, tanto hacia nosotros mismos como hacia el suelo (en el caso de que estuvieran suspendidos). Una forma sencilla de simulación de sombras crear una máscara del objeto a proyectar y aplicarle un desenfoque (blur). El color que lleve la sombra debería ser proporcional a la intensidad luminosa, pero sin ser tan estrictos podemos suponer sombreado negro o, ”afinando” un poco, en la escala de grises.

6.7.2. Fog

El uso de efectos atmosféricos realza la profundidad y el ambiente. Una técnica muy usada en 2.5D es disminuir (aumentar) linealmente el contraste y la saturación del color a medida que vamos alejándonos (acercándonos) al plano zNear, creando una sensación de "niebla en el horizonte" (fog). La siguiente figura muestra este concepto.

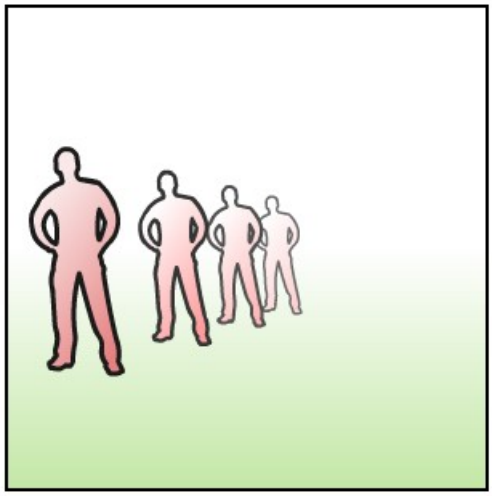


Ilustración 16: Aplicación de niebla a la escena

6.7.3. Desenfoques

En la primera especificación de SVG ya se incluía el filtro blur en numerosas versiones (gausiano, especular, convolucional, etc.), con lo que es posible simular distancias focales de cámara. La idea es sencilla: un determinado objeto se encuentra en una determinada capa, con su valor de cota `z` asociado. Al enfocar dicho objeto, todos los elementos que queden por debajo o por encima de su capa deberían quedar algo desenfocados. Este algoritmo puede ser intensivo para el procesador, dependiendo del número de objetos que integren la escena. Por ello, para disminuir el coste computacional, en ocasiones se procede a intercalar un plano de desenfoque por detrás del objeto enfocado.

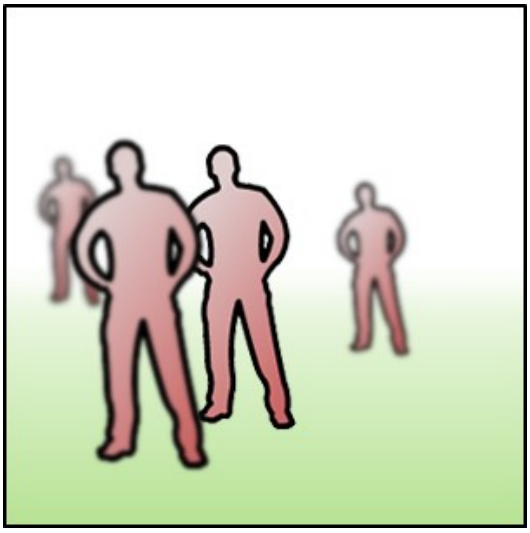


Ilustración 17: Aplicación de desenfoques

6.7.4. Integración de elementos pre-renderizados

Un recurso para añadir contenidos a nuestro entorno 2.5D es integrar objetos ya renderizados previamente en una aplicación 3D. Esto se hace así dado un objeto que por su complejidad o naturaleza (imaginemos un avatar configurable por cada usuario) nos pueda resultar más cómodo incluir de esta manera. Esto se emplea para disponer de todas las vistas de un elemento en torno a su eje de rotación, por lo que deberemos realizar un render frame-by-frame de 360 grados. De esta manera, al incluirlo en el entorno, siempre estaremos viendo la perspectiva correcta de dicho objeto.

La desventaja de este método es que al haber renderizado un objeto previamente, esa rotación de 360 grados se ha hecho en base a una cámara. Al integrarlo en nuestro entorno y situarnos muy cerca de él tendremos una variación angular muy grande en una distancia focal muy corta, lo que puede implicar perspectivas visualmente incorrectas. Pero no deja de ser un recurso más que suficiente en dispositivos móviles.

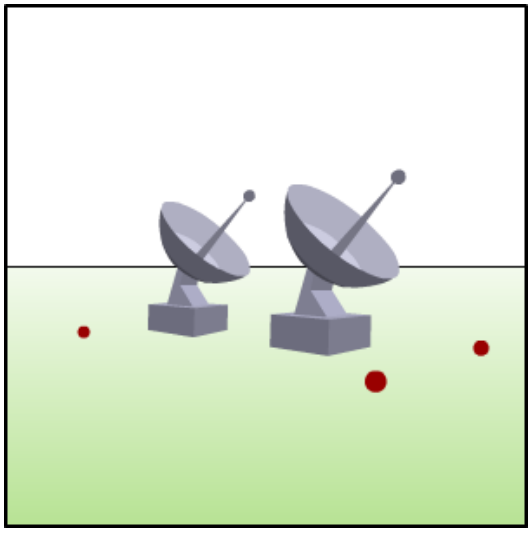


Ilustración 18: Integración de elementos pre-renderizados

Cuantos más fotogramas tenga el objeto pre-renderizado, mayor es el rango de variación angular y por lo tanto más suave serán los cambios entre perspectivas, pero a mayor número de fotogramas el tamaño de la animación aumenta. Deberemos escoger una solución de compromiso dependiendo de la aplicación.

6.7.5. Wire Frame

Para definir una escena de 3D técnicamente “real” hay que definir los objetos por una serie de puntos en el espacio, que a su vez definirán unos polígonos y serán estos los que definan la superficie de los objetos.

Por tanto, las transformaciones deben realizarse sobre dichos puntos. Al unir varios puntos por líneas (curvas o rectas) estaremos definiendo el contorno de un determinado objeto. Esa representación sin relleno ni textura se conoce como wireframe o representación alámbrica.

Hay que definir también una función que transforme los puntos que definen el objeto de 3D a 2D, empleando para ello las ecuaciones presentadas en el Capítulo 6.6.1, o, de forma más avanzada, los conocidos cuaterniones (matrices de transformación), que por su complejidad algorítmica quedan fuera de los propósitos de este trabajo.

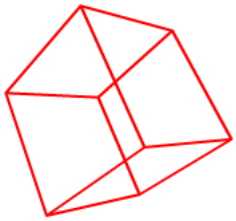


Ilustración 19: Representación alámbrica (Wireframe)

En determinadas aplicaciones de RV es muy útil trabajar con esta representación de modelos. Sin embargo, en dispositivos móviles el trabajar con nubes de puntos disminuye considerablemente el rendimiento de la aplicación, pues el número de operaciones a realizar por segundo es bastante elevado.

Está claro que si nuestra aplicación no tiene objetos muy complicados, este recurso puede enriquecer el aspecto visual de la misma.

6.7.6. Rellenos

Para crear caras y dar color a la representación alámbrica de un objeto, se emplean los rellenos. Ya que que nuestra aplicación se va a generar en tiempo real, no hay que perder de vista el overlapping entre capas, lo cual puede complicar el proceso de dibujado. Afortunadamente, simplemente basta con dividir cada objeto en las caras que lo compone para superar este handicap.

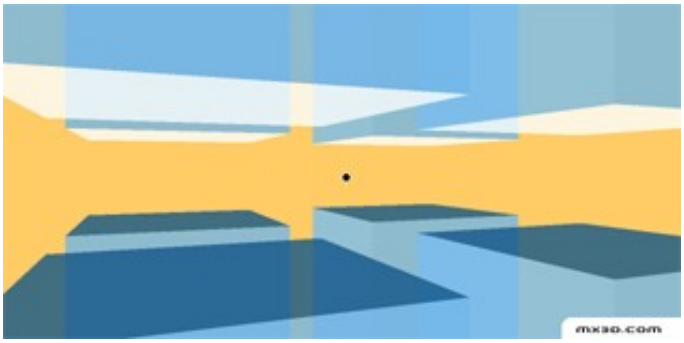


Ilustración 20: Relleno de caras con transparencia

A la hora de redibujar la escena, sólo debemos ver las caras del objeto que apuntan hacia la cámara. En la figura anterior se emplean transparencias para no tener que calcular las caras ocultas al punto de vista en cuestión. Pero ¿Cómo hacerlo de forma eficiente? Si hemos usado el método de separar en capas cada polígono que forma un objeto, habrá que determinar cuáles son visibles según su posición en el eje Z (de

forma similar al funcionamiento del z-buffer). Pero existen otras técnicas más adecuadas, como se verá a continuación.

6.7.7. Backface Culling

Las caras de un objeto sólo se pueden ver desde un lado, el otro es invisible a la cámara. Existen dos formas para mostrar la parte visible de una superficie: definiendo las normales o definiendo las pendientes. Ambas técnicas se conocen como backface culling (podrían ser traducidas como recorte de caras ocultas). Usándolas correctamente podemos obviar el problema del overlapping, si bien hay que implementar unos algoritmos más avanzados.

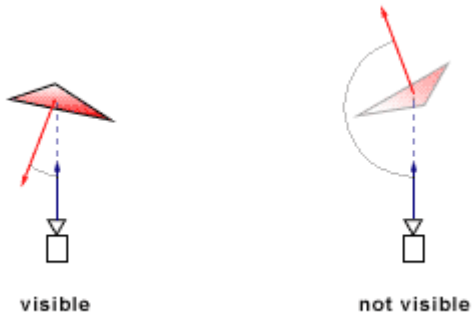


Ilustración 21: Comparación de las normales de un polígono

Una normal es un vector perpendicular a una superficie, cuyo sentido define el lado visible de los objetos. A no ser que lo queramos expresamente, este vector no se dibuja, sólo actúa como referencia. Con dicha referencia, si el ángulo que forma con la dirección de la cámara es mayor de 90 grados, la cara estará oculta.

Una forma equivalente más sencilla es comparar la pendiente de una arista en 2D. Para ello, se definen los puntos de un polígono en sentido horario, por ejemplo, y se establece como visible esa numeración. Si en la actualización de la escena la numeración es antihoraria, ese polígono estará oculto. En la siguiente figura se clarifica esta idea: en el instante inicial la pendiente del segmento AB está definida como positiva respecto de un eje vertical que pase por B; en el instante final está definida negativa bajo el mismo criterio, por lo que ese polígono no se pinta en pantalla.

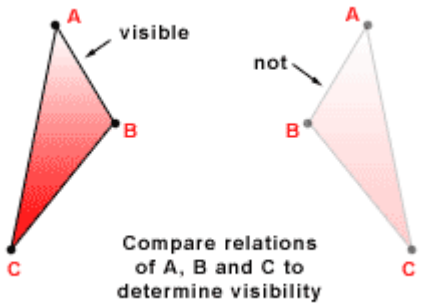


Ilustración 22: Comparación de la pendiente de una arista

Llevando estas consideraciones a la práctica, esta función determina si un triángulo definido por los puntos `ABC` es visible:

```
if (((B.y-A.y)/(B.x-A.x) - (C.y-A.y)/(C.x-A.x) < 0) ...
^ (A.x<=B.x == A.x>C.x )) {
 return true;
} else {
 return false;
}
```

Donde se accede a las coordenadas de un punto en X e Y (plano 2D como consecuencia de haber proyectado 3D sobre la pantalla del dispositivo) por medio de las propiedades `.x` y `.y`. El operador `^` indica `BITWISE OR` (O Lógico).

6.7.8. Frustum Culling

En numerosas ocasiones estamos calculando elementos que no aparecerán en pantalla por el simple hecho de no estar en el campo de visión. Eso supone mayor tiempo de procesamiento o, en otras palabras, menos fps para actualizar la escena.

El Frustum Culling es la técnica que nos permite averiguar qué se encuentra dentro del volumen de visión o Frustum. Esos elementos se descartan o se aceptan por grupos. Normalmente los grupos que queremos comprobar se agrupan dentro de volúmenes simples; como por ejemplo una esfera (Bounding Sphere) o una caja alineada con los ejes de coordenadas (Axis Aligned Bounding Box o AABB). Si la esfera o la caja no están fuera del Frustum, dibujaremos los elementos en cuestión.

Por supuesto el Frustum Culling también se puede utilizar para el escenario, pero antes debemos dividirlo en trozos (Space Partitioning). Así, si partimos el escenario en octrees o sectores, el Frustum Culling nos permite averiguar qué cubos del octree son visibles o cuales de los portales que unen los sectores son visibles.

En 2.5D se puede representar como una máscara aplicada sobre la aplicación: todo lo

que salga de esa máscara no saldrá por pantalla, aunque la opción más empleada es borrar directamente de la memoria los objetos que han alcanzado un determinado valor en los ejes coordenados.

6.7.9. Otras técnicas

Hay muchos más métodos para implementar algoritmos de 3D en gráficos 2D. Actualmente se emplea álgebra matricial para controlar las cámaras y posicionar objetos, pero incrementa notablemente el número de operaciones a realizar por el procesador, así que se han obviado por quedar fuera del propósito de este trabajo. Lo que se ha pretendido es presentar unas técnicas muy sencillas que simulan eficazmente entornos virtuales. Además, una aplicación móvil puede ser consultada bajo diversas condiciones de procesador, memoria o resolución de pantalla, debiendo siempre intentar optimizar recursos. Hasta la fecha existen proyectos muy avanzados en este campo, con verdaderas librerías de funciones 3D para gráficos vectoriales. En el Capítulo 8 pueden consultarse algunas referencias.

6.8. Sonido

El problema más complicado al que se enfrentan los sistemas de RV, en cuanto a sonido se refiere, es la validación de los modelos matemáticos empleados. Muchas veces la precisión de estos pueden degenerar en excesivos tiempos de latencia o retardos en la señal. En dispositivos móviles no es necesario llegar a tanta precisión, a costa de ganar tiempo de carga y minimizar retrasos a la hora de reproducir un sonido.

Como se cita en [CAÑADA04], la única manera “real” que tiene una interfaz de “acercarse” al usuario es mediante el sonido. Y es muy cierto: todos los elementos interactivos de un sistema informático son bidimensionales. Pero el sonido es tridimensional, crea ambientes, rodea al usuario. Es interesante comprobar cómo difiere la compresión de un texto si se acompaña con una u otra música de fondo. Entonces, ¿por qué tan poca gente se atreve a usar sonido en las aplicaciones web? Los detractores de esta solución suelen negarse por dos razones: la primera es que los archivos de sonido añaden peso, prolongando el tiempo de descarga; la segunda es que no todo el mundo dispone de altavoces o bien consulta webs escuchando la música que le gusta (o simplemente los tiene desconectados), por lo que la señal se malgastaría. Pero existen ocasiones, por cuestiones de accesibilidad a personas minusválidas, por ejemplo, en la que es fundamental usar audio.

El tiempo de descarga no tiene por qué ser un impedimento. En todo desarrollo de front-end hay que saber cómo y cuándo precargar archivos, que es una de las claves para que todo esté a punto para cuando el usuario llegue al lugar en el que vaya a recibir feedback sonoro. A la segunda pregunta contestaría algo de sentido común: el uso de sonido en una interfaz web no debería ser excluyente, sino complementario

respecto a los métodos visuales.

A continuación se citarán unas nociones básicas de conversión analógico-digital para poder ser tomadas como referencia para aplicaciones en dispositivos móviles. Básicamente este proceso consiste en un muestreo frecuencial y posterior resolución (cuantificación + codificación).

6.8.1. Consideraciones básicas

La frecuencia de muestreo, medida en Hertz (Hz), es la cantidad de muestras de un sonido que se toman por segundo, el número de veces que se mide la amplitud de la onda con que se está trabajando en un segundo. Así, si trabajamos con calidad de CD y estamos grabando un sonido, nuestro ordenador tomará 44100 medidas por cada segundo grabado, ya que la frecuencia de muestreo en esta calidad es de 44.1 KHz.

La frecuencia de un sonido que percibimos los seres vivos es el número de veces que un determinado sonido oscila en cada segundo. El oído humano es captar de captar frecuencias (oscilaciones por segundo) en el rango 20Hz-20KHz aproximadamente.

La frecuencia de muestreo limita la frecuencia máxima que puede almacenar un fichero de sonido digital. Así sucede que un archivo con una cierta frecuencia de muestreo podrá almacenar como máximo un sonido de frecuencia la mitad que esa frecuencia de muestreo. Por ejemplo, un archivo con calidad de CD, 44100 Hz de frecuencia de muestreo, puede almacenar como máximo sonidos con una frecuencia de 22050 Hz (44100 / 2 = 22050).

La resolución hace referencia a la exactitud (precisión) de las medidas efectuadas, en bits: si la resolución es de 8 bits tenemos 256 niveles posibles. Si ampliamos a 16 bit, cada medida puede estar en un rango de 0 a 65535. Así, tanto el número de niveles de volumen (entre el silencio y el volumen máximo) como la calidad y riqueza de matices de la reproducción están directamente relacionadas con la precisión utilizada para la medida de la amplitud y el almacenamiento de los resultados.

Si la señal a muestrear contiene información en frecuencias superiores a la mitad de la frecuencia del muestreo, la toma de muestras no se hará de manera correcta y la versión de la señal basada en las muestras contendrá componentes falsos debido al fenómeno de aliases. La frecuencia máxima de la que se puede tomar muestras se llama frecuencia Nyquist y es igual a la mitad del ritmo de muestreo.

Con estas consideraciones, y teniendo en cuenta las restricciones en los dispositivos móviles, se recomienda emplear una de estas cuatro frecuencias de muestreo (Hz): [11.025, 22.050, 44.100] y para niveles de volumen o precisión: [8, 16] bits. Cada entorno 3D tendrá unos requerimientos. En web, por ejemplo, es común emplear 11.025 Hz y 8 bits para sonidos de tipo telefónico, mientras que en una tienda de venta de discos online podría interesar ofrecer archivos a 44.100 Hz y 16 bits.

Los formatos de audio más empleados en dispositivos móviles son AMR, WAV, MP3 y MID, aunque existen bastantes modelos que también reproducen RA y AAC.

Por último, queda hablar del sonido sintetizado, en el que no se realiza ninguna

captura de ondas sonoras reales, sino que es sonido totalmente digital, generado directamente en el equipo informático por el reproductor digital, como por ejemplo el conocido MIDI (Music Instrument Digital Interface). Cuando se desea reproducir un sonido concreto, se envía un comando al chip sintetizador, que se encarga de traducir ese comando en una vibración especial que produce el sonido. Mediante este sistema es posible crear melodías bastante aceptables, aunque nunca tendrán la calidad ni riqueza de una onda sonora natural capturada. Pero en entornos móviles puede ser una opción más que suficiente.

6.8.2. Sonido tridimensional

Un sencillo algoritmo consiste en modificar el volumen de un sonido asociado a una fuente puntual en base a su valor de cota Z. Así, los objetos más cercanos a la pantalla que lleven un sonido asociado se escucharán a mayor volumen.

Aunque la velocidad del sonido no depende del tono (frecuencia) ni de la longitud de onda de la onda sonora, sí es importante su atenuación. Este fenómeno sigue una ley cuadrática inversa, que explica que la intensidad sonora I disminuye con el cuadrado de la distancia (r) a la fuente sonora. Aplicando la ecuación de ondas y la definición de intensidad en promedio temporal se tiene:

$$I = A^2 / r^2$$

Donde A es una constante tras resolver la ecuación de ondas para $c=w/k$. En una aplicación para dispositivos móviles no necesitamos emular el fenómeno físico a la perfección, por lo tanto tomaremos $A=1$.

La mayoría de dispositivos no implementan sonido estéreo, por razones de espacio en memoria y porque normalmente sólo incorporan un altavoz. Para simular entonces sonido envolvente en los ejes X e Y podemos emplear la ecuación de atenuación anterior o simplemente reducir de forma lineal inversa el sonido a una fuente puntual en X e Y.

7. Líneas de investigación actuales

7.1. La nueva Realidad Virtual

Podría parecer que, tras su apogeo a finales de los 80, el término “Realidad Virtual” y todos los conceptos que encierra hubieran caído en el pozo del olvido a fuerza de promesas incumplidas. Sin embargo actualmente vivimos una época en la que la RV resurge lentamente de sus cenizas [GORKA01] y se va consolidando como una de las tecnologías que probablemente regirán nuestro futuro.

Uno de los mayores problemas a resolver en la actualidad es el de proporcionar una interfaz adecuada entre el mundo virtual y el usuario que lo habita. Acciones cotidianas como tomar algo que está sobre la mesa y dejarlo sobre el suelo pueden no ser tan evidentes en un entorno de RV, en el que a lo mejor la acción de tomar un objeto se realiza señalando con el dedo, y la de dejarlo abriendo la palma de la mano. Una cosa es el posicionarse en un escenario y otra bien distinta es poder interactuar con los propios objetos y cómo realizarlo.

Otro área en que se investiga actualmente es en las interfaces gráficas de usuario, campo en el que parece evidente que algún día deberemos migrar desde nuestras interfaces “físicas” planas actuales a nuevas y espectaculares interfaces gráficas tridimensionales. Llegados a este punto habrá que reescribir muchas de las analogías que se utilizan en los Sistemas Operativos actuales y que todos damos por válidas (por ejemplo, la acción de “cortar” y “pegar” es algo que se realiza sobre el plano, mientras que tal vez sobre el espacio deberíamos empezar a hablar de “coger” y “dejar”).

Por último, también el concepto de “comunidad virtual” empieza a tomar importancia. Tal vez en un futuro bastante más cercano de lo que nos creemos la videoconferencia sea sustituida por salas virtuales en la que personajes que representan a usuarios del mundo real (conocidos como avatares) charlen e intercambien información de una forma tan natural como si todos ellos estuviesen presentes en la misma estancia.

La evolución es inevitable... sólo falta que la sociedad esté preparada para recibir el cambio.

7.2. Las nuevas aplicaciones móviles

Según [AJIT06], en los próximos 5 años, el número de internautas en el mundo se cuadruplicará hasta llegar a los dos billones de personas. Del billón y medio de incremento, uno se corresponde con dispositivos webs baratos de bolsillo y muñeca utilizando contenido multimedia.

Más del 66% de los nuevos dispositivos que se conectarán a Internet, según esta previsión, no se corresponden con el prototipo de ordenadores de sobremesa o portátiles -con pantallas de muchas pulgadas y procesadores de gran capacidad-.

Si relacionamos estas predicciones (que el tiempo se encargará de confirmar o rechazar) con la situación actual y la dirección a medio o largo plazo que parece tener, tenemos argumentos de sobra para tener presentes algunos principios básicos del desarrollo de aplicaciones para web y dispositivos móviles, y como ya sabemos, son:

- Accesibilidad: capacidad de acceder e interactuar cualquier persona desde cualquier dispositivo.
- Codificar respetando las recomendaciones del Consorcio de la Web (lo que muchos hemos llamado erróneamente “estándares web“)
- Utilizando un código estructural, mediante el uso de etiquetas apropiadas para cada fin (títulos, párrafos, listas, citas, etc.)
- Separación del contenido y la presentación, utilizando formatos de intercambio y hojas de estilos adaptadas a cada tipo de dispositivo: ordenadores de sobremesa, impresoras, navegadores, pequeños dispositivos...
- Separación del contenido y el comportamiento, dejando, en la medida de lo posible, que el servidor se encargue del comportamiento, ya que no se puede tener la certeza que el agente de usuario interprete correctamente los lenguajes de scripts.

7.3. Los nuevos dispositivos móviles

Dice [BRAVO06] en su Blog:

- Se espera comercializar dispositivos con pantallas desplegables, olvidándonos así de los ciento y pocos pixels de ancho de muchos teléfonos móviles.
- Aumentan las posibilidades de interoperar con otros dispositivos, mediante teclados virtuales y la interpretación por comandos de voz. Ya no estamos limitados a un minúsculo teclado o área de escritura.
- Sigue aumentando la capacidad del procesador.

- Sigue aumentando la capacidad del "disco duro" de los pequeños dispositivos. Ahora mismo ya hay PDAs y teléfonos móviles con 4 GB de memoria.
- Nos olvidamos de la conexión GPRS y UMTS, dando paso a rápidas conexiones Wi-Fi libres y gratuitas.
- La Web Semántica, más o menos como la había imaginado Tim Berners-Lee: las máquinas beneficiando a las personas gracias a un eficiente intercambio de información automatizado.

Los dispositivos móviles están avanzando a una velocidad sorprendente. Pero ¿es esta rápida evolución del otrora humilde teléfono celular una cuestión de marketing impuesto o una progresión racional? La respuesta es probablemente una combinación de ambos. Pero el hecho es que los dispositivos que ahora están en etapa de diseño llevarán a los servicios móviles del mañana a lugares que sólo podemos soñar hoy.

8. Bibliografía

A continuación se citan las fuentes bibliográficas empleadas para elaborar el presente documento. Nótese que las URL aquí mostradas estaban operativas hasta la fecha de publicación de este trabajo (Julio de 2006), pero inevitablemente es posible que dejen de existir dentro de un tiempo. Por ello, se recomienda acceder a dichas referencias desde un buscador para consultar las futuras actualizaciones.

[LEIVA06] Luis Leiva Torres. "**Web3D. Técnicas de simulación e implementación en formato vectorial**". Curso Doctorado en RFIA. Valencia. 2006.

[COPESA00] Medios Digitales Copesa. "**Moviéndose a la web**". URL. Chile. 2000. http://mouse.latercera.cl/2001/rep/11/07/04.htm

[NPR02] T. Strothotte, S. Schlechtweg. "**Non-Photorealistic Computer Graphics: Modelling, Rendering and Animation.**". Ed. Morgan Kaufmann. San Francisco. 2002.

[MOLLA06] Ramón Mollá Vayá. "**Realidad Virtual**". Curso Doctorado en RFIA. Valencia. 2006.

[GRADIG01] Luciano Moreno. "**Gráficos Digitales (VIII)**". URL. Madrid. 2003. http://www.htmlweb.net/diseno/graficos_digitales/graficos_8.html

[DELPINO95] L.M. Del Pino González. "**Realidad Virtual**". Ed. Paraninfo. Madrid. 1995.

[BERBEL00] Genís Berbel. "**Una introducción a la Internet móvil** ". URL. Barcelona. 2000. http://www.uoc.edu/web/esp/articles/berbel/Internet_i_mobilitat.htm

[PIXEL96] John Kahrs et al.. "**Pixel Cinematography. A Lighting Approach for Computer Graphics**". Siggraph '96 Course#30. New York. 1996.

[CAÑADA04] Javier Cañada. "**Una apología del sonido en las interfaces web**". URL. Granada. 2004. http://www.terremoto.net/x/archivos/000096.html

[GORKA01] Gorka Elexgaray. "**Virtualidad Real. Sistemas, aplicaciones y futuro de la Realidad Virtual**". URL. Madrid. 2001. http://www.redcientifica.com/doc/doc200106100001.html

[AJIT06] Ajit Jaokar. "**Mobile web applications - do they need the browser?**". URL. Londres. 2006. http://www.readwriteweb.com/archives/mobile_web_appl.php

[BRAVO06] Alejandro Gonzalo Bravo. "**El futuro de la web está en los dispositivos móviles**". URL. Madrid. 2006. http://www.webposible.com/blog/?p=209

9. Enlaces en Internet

Para concluir, se presentan algunas direcciones de consulta recomendada en la web. Se han citado únicamente los enlaces más interesantes relacionados con este trabajo, dejando de lado las soluciones comerciales desarrolladas por diversas empresas.

sobre gráficos vectoriales -
www.svg.org
www.w3.org/Graphics/SVG/About

referencia básica en 3D y SWF -
www.kirupa.com/developer/actionscript/3dexplore.htm

numerosas definiciones -
www.wikipedia.org

mucha información sobre NPR -
www.red3d.com/cwr/npr/

el todopoderoso buscador -
www.google.es

Tabla 2: Enlaces de interés en Internet.

Glosario de términos

A

B

C

E

F

G

H

I

M

N

O

P

Q

R

S

T

U

V

W

X

Y

Z

Anexo. Nokia N70

Esta es la hoja de especificaciones técnicas del teléfono móvil en el que se han implementado los distintos algoritmos propuestos en la tesina.

Especificaciones
Tecnologías: XHTML, HTML, MMS, Java MIDP, Symbian OS 8.1 a, Series 60 Platform 2nd Edition
Procesador: 32-bit RISC CPU based on ARM-9 series, 220 Mhz
Firmware: v5.0609.0.2.1
Batería: Ion Litio 970mAh BL-5C
Dimensiones: 108.8 x 53 x 21.8 mm
Peso: 126 g

Pantalla
Resolución: 176 x 208 pixels
Profundidad de Color: 18 bit (262144 Colores)

Memoria
35 MB de almacenamiento interno
Tarjeta de memoria miniSD de 128 MB incluida*
Lector de tarjetas compatible con RS-MMC (Reduced Size Multimedia Memory Card) y/o MMC Mobile cards (Multimedia Memory Cards)

* *pueden existir diferentes configuraciones y capacidades disponibles de la tarjeta de memoria miniSD dependiendo del país o de la configuración del operador.*

Teclado
2 soft keys
5-way scrolling

Formatos de Video
3GPP (H.263)
MPEG-4
RealVideo

Cámara
Cámara de 2 megapíxeles (1.600 x 1.200 px) con zoom digital 20x
Cámara VGA secundaria (640 x 480 px) con zoom digital 2x

Conectividad Local
Bluetooth 1.2
USB 2.0
Infrarrojos

Configuración

CodePage = EN
DeviceImageFormats = image/gif, image/jpg, image/png
DeviceSoundKeyOnly = No
DeviceSoundsCached = No
DeviceSoundsOrdered = DeviceSound_MIDI_Generic, DeviceSound_AAC, DeviceSound_AAC_Plus, DeviceSound_3GP, DeviceSound_3GP2
DeviceVideoFormats = video/3gpp, video/mp4, video/mov
DynamicHeapSize = 1280
DynamicMemAvailable = Yes
Email = Yes
FlashLiteSVG = No
FSCommand = FivePerKeyPerFrame
GetSetMaxScroll = yes
getUrl = FivePerKeyPerFrame
GetURL_UTF8 = Default_NoConversion
GetURLOnLocalSWFOpensInBrowser = No
InputText = Yes
KeyRestrictions = No
keySet = Phone
LoadMovie = FivePerKeyPerFrame
LoadVars = FivePerKeyPerFrame
Loop = Yes
MMS = No
mouseType = None
MultipleDeviceSound = No
NativeSounds = NativeSound_ADPCM, NativeSound_MP3, NativeSound_PCM
NavigationType = 4Way
PersistentStoragePerMovieSize = 4
PixelType = RGB565_16
PlayerRendersText = No
SetSoftKeys = Yes
SharedObjects = Yes
SMS = Yes
SoftKeyCount = 2
SoundChannels = mono
soundEnabled = Yes
SoundSampleRate = 22
StreamingSound = Yes
TextEncodingForm = UTF16
TotalPersistantStorageSize = 256
UsePlayerEmbeddedVectorFont = No
UseSingleFontFace = No
VideoClipProportionallyToFitHeight = No
VideoClipProportionallyToFitWidth = Yes
VideoForceSizeDownToWindow = Yes
VideoForceSizeUpToWindow = No
VideoMaintainAspectRatio = Yes
VideoScaleInPowersOfTwo = Yes
VideoScalingDownLimitPct = 50

UNIVERSITAT
POLITÈCNICA
DE VALÈNCIA