

INTERFACES 3D EN LOS NAVEGADORES DE INTERNET

David Daniel Pérez Sosa

Ciudad del Este, Paraguay

www.david5.com

contact@david5.com

Resumen

Hoy en día existen posibilidades para el desarrollo de interfaces gráficas de usuario tridimensionales en los navegadores de internet, capaces de llegar al 99% de los internautas sin ninguna instalación adicional, a través de la máquina virtual Adobe Flash Player, que posee un formato multimedia nativo.

PaperVision3D es un motor de renderizado 3D en tiempo real, hecho en ActionScript 3, de código abierto, que posee las funcionalidades básicas de la computación gráfica tridimensional, creando una ilusión 3D en el motor de renderizado que posee el Flash Player.

Es posible realizar una solución tecnológica 3D basada totalmente en herramientas de código libre, utilizando Linux como plataforma para Flex SDK, que compila el código ActionScript, basado en las bibliotecas de PaperVision3D.

Palabras claves: 3D, Interfaces, Adobe Flash Player, PaperVision3D

Datos del investigador

Alumno:

David Daniel Pérez Sosa

C.I. 3.828.834

Tel. 0973-821224

Orientador:

Lic. José Eduardo Rojas

Área de expertise del orientador: Ingeniería de software – Profesor de la cátedra de Ingeniería de Software II de la Facultad Politécnica UNE. Desarrollo de aplicaciones en ambiente web – profesor de la cátedra de Seminario V de la Facultad Politécnica. Ha publicado estudios especializados sobre RIA (Rich Internet Applications) renderizados en Adobe Flash Player.

Area de investigación: Informática – Internet.

Tema de investigación: Utilización de un motor de renderización 3D en tiempo real.

Título del Proyecto: Interfaces tridimensionales renderizadas en tiempo real en los navegadores de internet

Etapas concluidas durante estos 6 meses

- Análisis de las Clases de ActioScript 3.0 a utilizar. 1 mes
Se utilizaron las clases [Papervision3D 2.0.869](#) y [Tweener 1.31.74.as3](#)
- Análisis de Blender. Primeros ejemplos 1 mes
Se utilizó para el modelado la versión gratuita de [Google Sketchup](#). La elección de este software se debe a su facilidad de uso. La exportación a formato [COLLADA](#) de Google Sketchup cumple las normas internacionales de intercambio de archivos 3D.
- Análisis de Papervision3D. Primeros ejemplos 1 mes
El resultado de esta etapa fue publicado en internet. Los primeros ejemplos consistieron en la construcción de un triángulo, un plano, la creación de un juego 3D simple, la construcción dinámica de superficies algebraicas, pruebas con luces y materiales, integración con sonido, y la utilización de materiales primitivos.
- Integración de Papervision3D y Blender. 15 días
Como fue explicado arriba, se utilizó Google Sketchup, y la integración entre Papervision3D y Google Sketchup se realizó a través del formato COLLADA

- Creación del primer ejemplo de espacio 3D simple 10 días
Ver <http://david5.com/labs/pv3d/investigacion/Panorama1.html>
- Desarrollo de un espacio virtual 3D integrando todo lo investigado 1 mes
Ver <http://david5.com/labs/pv3d/investigacion/Kmz1.html>
- Introducción de datos dinámicos al espacio virtual, via XML u otros 5 días
Se preparó el código para facilitar la carga de contenido HTML en el plano verde del espacio virtual 3D.
- Configuración del servidor y publicación 1 día
Todos los archivos fuentes pueden ser encontrados en esta dirección:
<http://david5.com/labs/pv3d/investigacion/srcInvestigacion.zip>
Y también aquí <http://www.david5.com/mis-primeros-experimentos-en-papervision3d/>
Se pueden ver los ejemplos en las siguientes direcciones:
-Construcción de un triángulo:
<http://david5.com/labs/pv3d/investigacion/ConstruccionDeUnTriangulo.html>
-Construcción de un Plano con la unión de dos triángulos
<http://david5.com/labs/pv3d/firststeps/planocontriangulos/MainPlano.html>
- Construcción de superficies algebraicas, implementando por primera vez la clase Tweener.
<http://david5.com/labs/pv3d/firststeps/algebraicsurfaces1/MainAlgebraicSurface1.html>
<http://david5.com/labs/pv3d/firststeps/algebraicsurfaces2/MainAlgebraicSurface2.html>
-Prueba con materiales e iluminación
<http://david5.com/labs/pv3d/firststeps/materialstest/MainSuperficieMontanosa.html>
-Prueba de mapeado de imágenes de mapa de bits en objetos primitivos (cubos y esferas)
<http://david5.com/labs/pv3d/firststeps/materialCubeAndSphere/TestMaterialCube2.html>

Presentación y discusión de los principales resultados, aspectos positivos y negativos.

El proceso de modelado, scripting y renderización en Papervision3D ha sido un gran desafío, debido a la poca documentación existente. El 95% del material de estudio se encuentra en el idioma inglés. No existe ningún libro, ya sea gratuito o de pago, las demostraciones están distribuidas por diversos blogs.

En el momento que superé esta etapa de dar los primeros pasos en el aprendizaje, ya fue fácil explorar las distintas clases que posee Papervision3D, fue de gran valor el hecho de que el código sea Open Source, si no fuese así, sería imposible aprender sin la documentación adecuada.

Si duda alguna con esta tecnología se abre la posibilidad para la construcción de una nueva gama de front-ends tridimensionales en internet.

El principal desafío es la optimización del código. Se obtienen framerates aceptables con un máximo de 4000 a 6000 triángulos, dependiendo del procesador.

Para el desarrollo, se necesita como mínimo una computadora con procesador de doble núcleo, memoria de 2Gb, y un buen acelerador gráfico.

Es destacable el tamaño reducido de los archivos compilados.

Entidades que me dieron apoyo para la realización del Proyecto.

El principal apoyo lo he recibido del Profesor Orientador Lic. José Eduardo Rojas, quien me ha apoyado siempre en el proyecto. También la comunidad de desarrolladores de Papervision3D, a través de: foros, mailing-list, salas de chat y blogs. Los principales los cito abajo:

<http://www.nabble.com/Papervision3D-f22855.html>

<http://pv3dchat.flashbookmarks.com/>

<http://dev.papervision3d.org/>

<http://www.unitzeroone.com/blog/>

<http://blog.zupko.info/>

<http://pv3d.org/>

<http://forum.papervision3d.org/>

<http://groups.google.com/group/ESPapervision3D>

Principales resultados:

Introducción

La experiencia de usuario es un factor que últimamente ha cobrado gran importancia en la red. Existe una clara tendencia a que las aplicaciones de escritorio se trasladen al navegador: procesadores de texto, planillas electrónicas, software de tratamiento fotográfico, clientes de correo electrónico, y un sinnúmero de aplicaciones han sido construidas para ejecutarse en los navegadores de internet. La pregunta es, ¿por qué no han aparecido muchas interfaces tridimensionales para ciertas aplicaciones o páginas web complejas con una mínima cantidad de tiempo de descarga? La respuesta es simple: la falta de [plataforma de software](#) adecuada para el desarrollo de las aplicaciones tridimensionales online. Este punto será analizado con profundidad en este artículo.

Para simples demostraciones e interfaces, Papervision3D abre un mundo largamente descuidado en la red, ahora es potencialmente posible crear interfaces tridimensionales impactantes, e interactivas, con un mínimo tiempo de descarga. Esto es algo que los desarrolladores de internet han querido por un largo tiempo. Aún tiene grandes limitaciones, pues la aceleración es realizada por software y no por hardware, pero está en desarrollo [una nueva versión de Papervision3D](#), basado en la nueva API del Flash Player 10, que ya incorpora aceleración por hardware, y con ello se espera aumentar dramáticamente la calidad de las páginas web tridimensionales de internet. Por este camino, se abre toda una vía de interfaces, esto hace a Papervision3D muy importante en el desarrollo web.

Uno de los objetivos será mostrar los principales elementos necesarios para el renderizado 3D usando las clases de Papervision3D.

Análisis de la tecnología existente para el desarrollo de interfaces 3D en los navegadores

Cuando el conjunto de tecnologías y recursos para el desarrollo de interfaces tridimensionales y el fácil acceso de usuarios (Kit de Desarrollo de Software, motor de renderizado 3D en tiempo real vía internet, utilización del GPU por un motor de renderizado, alta penetración la tecnología en los navegadores de internet y ancho de banda de alta velocidad) haya madurado, será inminente un cambio de paradigma en la navegación por internet.

Los ingenieros de software de numerosas organizaciones y numerosas comunidades de desarrollo de código abierto están trabajando hoy en día para hacer posible la renderización de 3D en tiempo real en los navegadores de internet. Es posible demostrar que el entorno de desarrollo basado en la máquina virtual Adobe Flash Player es actualmente la infraestructura líder para ejecución de interfaces 3D en los navegadores de internet.

Las ventajas de Flash sobre otras alternativas:

En este análisis simplemente se resaltan aspectos técnicos y la ubicuidad, que van excluyendo alternativas para desarrollo 3D para navegadores, con respecto a la plataforma Flash.

HTML vs Flash para 3D

HTML es un formato de documento, Flash tiene en su núcleo un formato multimedia. Entonces, a diferencia de HTML, que tiene exactamente 1 fotograma, el contenido de FLASH puede tener una infinita cantidad de fotogramas con respecto al tiempo. HTML es estático, FLASH es dinámico. HTML no posee Drag and Drop, Flash si.

AJAX vs Flash para 3D

En la mayoría de los casos, AJAX es más promocionado para el desarrollo de RIA, los ejemplos incluyen: Gmail, Google Maps, entre otros. Pero una prueba de de AJAX está llegando a sus límites, es que Google Earth no se ha podido llevar al navegador con AJAX, quedando demostrado que el desarrollo de software 3D va más allá de lo que AJAX puede ofrecer.

SVG vs FLASH para 3D

Flash y SVG fueron concebidos inicialmente para gráficos bidimensionales, pero el Core Team de Papervision3D ha desafiado los rendimientos del Flash Player y lo ha convertido en un motor de renderizado 3D. Esto también podría ser posible con SVG, pero las etiquetas aplicadas nada mas que a un fotograma de contenido real (no a páginas de demostración) son demasiadas. La productividad es demasiado baja. En el navegador Internet Explorer, es necesario instalar un plug-in para la visualización (la mayoría de las PCs que tienen instalados Internet Explorer no poseen este plugin actualmente).

Máquina virtual Java vs Flash para 3D

La Máquina Virtual Java posee diversas versiones, incompatibles entre sí. Flash no posee incompatibilidades entre una versión y otra.

Silverlight vs Flash para 3D

Silverlight posee más potencia que Flash para el desarrollo de entornos tridimensionales, debido a que tiene acceso a DirectX, pero aún posee un índice de penetración demasiado bajo. Es necesario hacer un download del plugin para poder utilizarlo. [Flash Player está presente en el 98% de los navegadores.](#)

Shockwave vs Flash para 3D

Shockwave es un plugin para navegadores especializado en la creación de contenido 3D en tiempo real. Técnicamente es la mejor opción para desarrollo 3D, pero posee un índice de penetración bajo (50%).

Unity3D vs Flash para 3D

Unity3D también es un plugin para navegadores especializado en la creación de contenido 3D

en tiempo real. Técnicamente es la mejor opción para desarrollo 3D, pero también posee un índice de penetración muy bajo (1%).

Por lo tanto, Adobe Flash Player podría ser actualmente la mejor opción para la creación de interfaces tridimensionales en los navegadores de internet.

El punto más importante para el buen posicionamiento del Flash Player en este análisis, es por estar presente en tan alto porcentaje en los navegadores de internet. Esto tiene consecuencias realmente importantes. El ejemplo más claro es Second Life, un mundo virtual que no ha tenido tanto éxito como se esperaba, y gran parte de este fracaso se debe al download del software (21.2MB) que debe realizarse obligatoriamente para poder acceder a dicho mundo virtual. Bajar un plugin para visualizar un sitio web suele ser una gran limitación para que un sitio alcance gran popularidad, los internautas generalmente no desean instalar nuevos plugins. [Flash es el plugin más extendido en todo el mundo.](#)

Pero el Flash Player no es Open Source, eso es bueno o malo?

Es bueno. FLASH no es Open Source ni lo será. Esto se debe a la mala experiencia que ha tenido SUN con la Máquina Virtual Java. El hecho que sea Open Source hará que surjan versiones alternativas, y lo más probable es que sean incompatibles entre si, y con ello, se perderá un gran virtud del Flash Player, que es programar un solo código para diversos navegadores, y en distintas plataformas, sin preocuparse por las incompatibilidades. Java ha fracasado en este sentido. (y JavaFx no posee alta ubicuidad).

El Flash player está diseñado y optimizado para renderización en 2D. No en 3D.

Este es un punto crítico por el cual Flash no es técnicamente la mejor opción. La cantidad de fotogramas de contenido 3D renderizados por segundo suele ser media o baja, en casos reales. La comunidad de desarrolladores del Flash Player está desarrollando varios proyectos para poder renderizar 3D en el Flash Player, existen proyectos privados y de código abierto, entre los que se encuentran:

[Alternativa3D](#)

[Away 3D](#)

[Five3D](#)

[ND3D](#)

[Papervision3D](#)

[Sandy3D](#)

[Wire Engine 3D](#)

Para este artículo, he decidido hablar sobre Papervision3D, por ser el motor de renderización 3D en tiempo real más adoptado entre los citados arriba.

Nociones Básicas de diseño 3D con Papervision3D

En términos de visualizaciones en ordenador, más específicamente en 3D, la [renderización](#) es un proceso de cálculo complejo desarrollado por un ordenador destinado a generar una imagen 2D a partir de una escena 3D.

La técnica más utilizada hoy en día para la producción de gráficos 3D en tiempo real es la rasterización. La rasterización es básicamente un proceso de transformación de datos de vectores (datos) se convierten en un conjunto de píxeles (imágenes). Papervision3D utiliza esta técnica, debido a que es la más rápida.

En pocas palabras, el motor de renderizado de Papervision3D utiliza bastante lógica para producir una ilusión 3D sobre 2D. Dicha lógica posee básicamente lo siguiente:

Proyección ([project](#))

La proyección ortogonal se realiza a través de transformaciones matemáticas. Dicha técnica no es del todo satisfactoria, por ello, en realidad se utiliza la proyección en perspectiva. En Papervision3D una matriz 4x4 ([Matrix3D](#)) representa la translación, rotación y escala para todos los ejes. También existen otras tres matrices para representar escenas 3D: La matriz global, la matriz de cámaras, y la matriz de transformación de objetos.

Más explícitamente, cada vértice 3D que exista en cada objeto 3D tendrá que ser transformada por todas estas matrices para terminar como un punto 2D en el Flash Player.

Geometría

Básicamente, todo punto ubicado en el espacio es almacenado en forma de vértices de datos ([Vertex3D](#)). El conjunto de tres vértices puede formar una face ([Triangle3D](#)). Matrices de vértices y faces se almacenan en instancias de [GeometryObject3D](#). Cada clase [DisplayObject3D](#) contiene información del objeto (ubicación, transformación, orientación, ejes, etc) y una referencia a la instancia de GeometryObject3D. Los datos de DisplayObject3D son utilizados por el motor de renderizado [BasicRenderEngine](#) y los materiales de los objetos, que son utilizados para dibujar el objeto dentro de un objeto [Viewport3D](#).

Mapeado de texturas

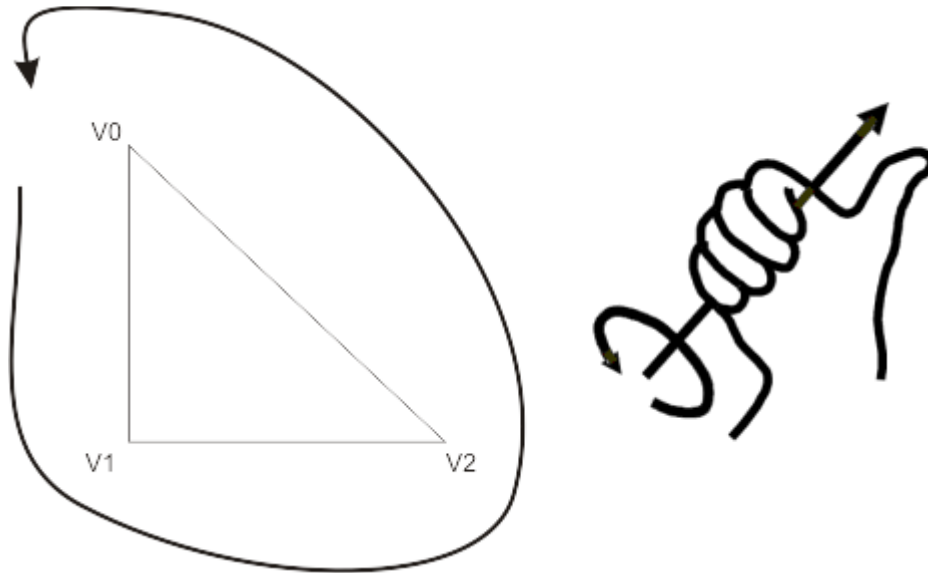
Es el proceso más utilizado, y el que consume mayor cantidad de ciclos del CPU. El [mapeo UV](#) es utilizado en este proceso.

El mapeado, de un triángulo, en el caso más básico, se realiza de la siguiente forma:

Se crea una matriz de vertices, en cada posición de la matriz almacenamos una instancia de Vertex3D que será la coordenada x, y, z del vértice del triángulo. En este ejemplo llamémosle v0, v1, y v2 a los vértices del triángulo.

Una face (cara) está compuesto por tres vértices (v0, v1, v2). Se crea una instancia de [Triangle3D](#), donde se introduce el material a ser mapeado, los vértices, y las coordenadas UV

de cada vértice. El array de 3 vértices introducidos en la instancia de Triangle3D debe poseer la misma secuencia de ubicación el array de las 3 coordenadas UV. La coordenada UV de cada punto corresponde a la coordenada del [versor bidimensional](#) de la proyección del punto sobre el objeto Viewport3D. La secuencia elegida puede ser arbitraria, pero debe recordarse que la orientación de la cara estará regida por la [regla de la mano derecha](#).



```
v0 = new Vertex3D(x0,y0,z0); vertices.push(v0);
```

```
v1 = new Vertex3D(x0,y0,z0); vertices.push(v1);
```

```
v2 = new Vertex3D(x0,y0,z0); vertices.push(v2);
```

```
faces.push( new Triangle3D(triangleMesh, [v0,v1,v2], material, [UV0, UV1, UV2]).
```

Se obtiene los mismos resultados siguiendo las secuencias: [v0,v1,v2] o también [v1,v2,v0] e incluso [v2,v0,v1]

Si la face se ubica en sentido contrario, la cara tendrá sentido contrario. Se mapeará la otra cara del triángulo siguiendo las secuencias contrarias: [v0,v2,v1], o también [v1,v0,v2], e incluso [v2,v1,v0]

El array "vertices" y el array "faces" se almacenan en una instancia de GeometryObject3D.

Los objetos geométricos complejos se crean a partir de la combinación de triángulos, que son el ente geométrico básico en el proceso de renderizado. Aquí posteé unos [ejemplos de construcción de un triángulo, y de instancias de TriangleMesh3D más complejas](#).

Materiales

Los materiales son una colección de propiedades que determinan como será renderizada una superficie 3D. Podríamos llamar a los vértices esqueletos, y los materiales serían la piel del

esqueleto. Los distintos tipos de materiales heredan la clase [MaterialObject3D](#), esto implica que se encontraran las siguientes propiedades en cada material:

name (string): el nombre del material

interactive(Boolean): habilita la asignación de listeners, eventos de click por ejemplo.

oneSide o doubleSided(Boolean): Mapea el material en una, o en ambas caras, repectivamente.

smooth(Boolean): aplica un algoritmo de suavización al material.

Los tipos básico de materiales son:

WireframeMaterial

Como dijimos anteriormente, un objeto 3D está compuesto por varios triángulos. Los lados de cada triángulo, de todo el objeto, son visualizados con WireframeMaterial.

ColorMaterial

Los materiales de color proyectan un color sólido sobre los objetos 3D.

BitmapMaterial, y BitmapFileMaterial

Es posible usar una imagen jpg, gif o png, o cualquier bitmapData sobre el objeto

BitmapColorMaterial

Es parecido a ColorMaterial, la diferencia es que se puede acceder a los pixeles del material.

VideoStreamMaterial

Se puede hacer streaming de video como materiales de objetos 3D.

Iluminación y sombreado

La simulación de luz es uno de los aspectos más complicados en el proceso de rasterización 3D debido la alta cantidad de ciclos del CPU que ocupa. Por ello, la iluminación el Papervision3D es muy básica. Se puede crear una sola luz instanciando la clase [PointLight3D](#). Las luces afectan a los siguientes materiales: [CellMaterial](#), [EnvMapMaterial](#), [FlatShadeMaterial](#), [GouraudMaterial](#), y [PhongMaterial](#).

Collada

En Papervision3D es posible importar objetos tridimensionales desde cualquier software de modelado 3D que soporte la exportación a COLLADA. COLLADA es un formato XML normalizado para intercambio de archivos para aplicaciones 3D interactivas.

Cámaras

En Papervision3D hay tres cuatros de cámaras: free, target, spring y debug.

La cámara posee tres propiedades específicas que pueden ser manipuladas: zoom, enfoque y FOV

Creación del proyecto en ActionScript 3 usando Papervision3D

Flex 3 SDK es multiplataforma, y para estos ejemplos, han sido compilados utilizando el modo consola de Linux. Existe un gran potencial para el desarrollo de RIAs al compilar de esta manera, ya que son incluidas las bibliotecas del Flex. Éstas pueden ser aprovechadas convenientemente, fusionándolas con la interfaz tridimensional que puede crearse con Papervision3D. Esto abre la posibilidad de contruir interfaces innovadoras.

La versión de Papervision3D más utilizada es Papervision3D v2.0 Great White, que actualmente se encuentra en fase beta. Posee varias novedades con respecto a las versiones anteriores, entre ellas, la implementación de iluminación. También posee mayor optimización de código.

PaperVision3D + Tweeners + Flex SDK + Linux: Una solución Open Source

Carlos Ulloa ha liberado Papervision3D a la comunidad a finales del 2006 y ha tenido un gran avance desde entonces. Para la transición de los valores de las propiedades numéricas utilizo las clases Tweeners de Fernando Zeh.

Adobe Flex 3 SDK se ha convertido a Open Source, existe un IDE llamado Adobe Flex Builder, que es propietario, pero es de uso opcional.

Es importante aclarar que Adobe Flash Player sigue siendo propietario, pero es totalmente lícito utilizarlo gratuitamente a partir de las herramientas mencionadas arriba.

Conclusión

Con la utilización de todas las nuevas características de la API del Flash Player 10, se obtendrá mayor capacidad para el renderizado 3D en tiempo real, gracias a la renderización por hardware. Esto optimizará notablemente las prestaciones de Papervision3D y mejorará notablemente la experiencia de los usuarios de Internet gracias a estas tecnologías.