Universidad de las Ciencias Informáticas

Facultad 5

Optimización de la visualización en la herramienta gráfica SceneToolKit

Informe de los Aportes Personales al Proyecto Herramientas de Desarrollo para Sistemas de Realidad Virtual

Presentado en opción al título de

Máster en Informática Aplicada

Autor: Ing. Yanoski Rogelio Camacho Román

Tutor: MSc. Yuniesky Coca Bergolla

Asesor: MSc. Pedro Carlos Pérez Martinto

Ciudad de la Habana

Junio de 2009

Declaración jurada de autoría

Yo, Yanoski Rogelio Camacho Román, con carné de identidad 80041120627, declaro que soy el autor principal del resultado que expongo en la presente memoria titulada "Optimización de la visualización en la herramienta gráfica SceneToolKit", para optar por el título de Máster en Informática Aplicada.

Este trabajo fue desarrollado durante el período 2003–2009 en colaboración con mis colegas de equipo, quienes me reconocen la autoría principal del resultado expuesto en esta memoria.

Autorizo a la Universidad de las Cencias Informáticas a hacer el uso que estime pertinente de los resultados aquí presentados, como propietaria de los derechos legales de este proyecto.

Finalmente declaro que todo lo anteriormente expuesto se ajusta a la verdad, y asumo la responsabilidad moral y jurídica que se derive de este juramento profesional.

Y para que así conste, firmo la presente declaración jurada de autoría en Ciudad de la Habana a los 15 días del mes de junio del año 2009.

Yanoski Rogelio Camacho Román

Agradecimientos

A quienes han contribuido a mi formación y mis resultados profesionales como Ingeniero en Informática.

A Mayra, por impulsarnos y apoyarnos.

A Coca, excelente tutor y amigo.

A la Facultad 5 por confiar en mí para esta actividad tan importante para el país como es la producción.

A los que han compartido conmigo en el proyecto y han dado su aporte, a la SceneToolKit, y a mi persona.

A la pequeña pero importante comunidad de desarrollo.

A los que siempre han confiado en la STK.

A mi familia y amigos, a todos, gracias...

Síntesis

Con la creación de un perfil de realidad virtual en la Universidad de las Ciencias Informáticas (UCI), surgieron nuevos retos para la Facultad 5, encargada de esta línea de investigación-producción.

Una necesidad primaria ante el desarrollo de simuladores y videojuegos, fue reutilizar código a través del uso de algún motor de simulación, que permitiera desarrollar aplicaciones de realidad virtual con calidad y en el menor tiempo posible.

De esta manera, surge un proyecto con la misión de brindar herramientas informáticas de apoyo a la producción, el cual lleva más de cuatro años de trabajo y cuenta ya con un cúmulo importante de experiencias.

En el presente documento, se hacen reflexiones sobre los resultados del proyecto y de la **SceneToolKit** (**STK**) como producto principal del mismo. Se exponen además algunas contribuciones del autor a dichos resultados.

Palabras Claves: Optimización, motor gráfico, motor de simulación, motor de videojuego, realidad virtual, simuladores, videojuegos, *graphic engine*, *game engine*.

Índice de contenidos

1. Introducción **1**

2. Desarrollo **5**
2.1. El proyecto 5
2.1.1. La SceneToolKit 5
2.2. Contribución personal 8
2.2.1. Grafo de escena 10
2.2.2. Sistema de cámaras 12
2.2.3. Oclusión estática 14
2.2.4. Niveles de detalles (LOD) 16

3. Conclusiones y recomendaciones **18**

A. Glosario de términos **20**

B. Glosario de abreviaturas **24**

Referencias **25**

1. Introducción

La Realidad Virtual (RV) ha alcanzado la madurez tecnológica suficiente como para convertirse en una valiosa herramienta al servicio de la industria. Los Sistemas de Realidad Virtual (SRV) comienzan a popularizarse. Muchos productos empiezan a invadir el mercado en forma de simuladores, videojuegos, así como en la televisión y el cine, entre otras esferas.

En la Universidad de las Ciencias Informáticas (UCI), se comenzó a trabajar en el tema de la Realidad Virtual desde el 2003, con el desarrollo de un simulador de conducción de auto en asociación con la empresa SIMPRO[1]. A partir de ese momento surgieron nuevos compromisos de sistemas a desarrollar, tanto simuladores como videojuegos. Por tal motivo, se necesitó de algún sistema de trabajo que garantizara eficiencia productiva y calidad, minimizando los esfuerzos de los desarrolladores.

La respuesta a este problema mundialmente es la reutilización de código, unificando las funcionalidades comunes en sistemas conocidos en este caso como herramientas o motores gráficos (en inglés *engines*), sobre los cuales se montan las aplicaciones finales. Asociadas a estos motores se suelen brindar otras herramientas auxiliares y editores que agilizan el proceso de desarrollo [Alo08].

En la UCI se probaron algunas herramientas existentes (libres en internet o compradas) y no dieron el resultado esperado, pues en todos los casos hubo que modificar su código. Este es un proceso realmente engorroso y lento, sobre todo cuando no se tiene una documentación clara de su arquitectura con los

[1]SIMPRO, Empresa de SIMuladores PROfesionales: alias comercial del Centro de Investigaciones y Desarrollo #2 (CID-2) de la Unión de Industrias Militares (UIM) de las FAR.

detalles necesarios. Se identificó entonces la necesidad de concebir herramientas propias para el trabajo en los proyectos, en especial un motor de simulación sobre el cual desarrollar las aplicaciones finales.

Entre las diversas problemáticas a enfrentar para la implementación de este motor, se tuvo la siguiente: ¿Cómo optimizar el proceso de visualización en la Herramienta principal a desarrollar para el Polo Productivo de Realidad Virtual de la UCI?

Entiéndase por optimización de la visualización, lograr una frecuencia de refrescamiento de alrededor de 30 imágenes o fotogramas por segundo (FPS) [Boo08], es decir, aumentar en todo lo posible la velocidad con que la aplicación hace el redibujado del *frame buffer*[2], manteniendo una fluidez visual acorde con los conceptos de respuesta en tiempo real, a través de la implementación de técnicas que aceleren dicho proceso de redibujado.

Dicho problema se enmarca dentro del proceso de desarrollo de motores de simulación y videojuegos en el Polo Productivo de Realidad Virtual, como objeto de estudio.

Analizando este problema en específico, se planteó como objetivo: Incorporar técnicas que optimicen la visualización de los entornos a la Herramienta en desarrollo.

El campo de acción sería el proceso de desarrollo de técnicas para la optimi-

[2]Porción separada o seleccionada de una memoria del *display* que contiene un *frame* o fotograma completo de datos de representación. Algunas veces se denomina *bit map* o mapa de *bits*. Los algoritmos de síntesis de imágenes calculan la imagen en el *frame buffer* con los parámetros de la cámara virtual correspondientes a la posición y orientación de la misma. La información digital del *frame buffer* se traduce en una señal de video por la tarjeta gráfica y esta señal se traduce en una imagen física en el dispositivo de visualización.

zación de la visualización en motores de simulación y videojuegos.

Para dar cumplimiento a este objetivo se plantearon las siguientes tareas:

- Estudiar algoritmos, técnicas y tendencias para la visualización optimizada de entornos virtuales.
- Constatar la presencia de algoritmos, técnicas y tendencias para la visualización optimizada, en los motores más populares.
- Adaptar y desarrollar soluciones técnicas acorde a las condiciones y características de la Herramienta a desarrollar.

Con el cumplimiento de estas tareas se esperaba implementar algunas técnicas básicas para la optimización de la visualización, y garantizar arquitectónicamente la flexibilidad para adicionar otras técnicas.

Así, se dotaría al Polo de una herramienta con buenas prestaciones en cuanto a velocidad de visualización, que abstraería[3] a los programadores de tener que preocuparse por la optimización de sus aplicaciones, enfocándose solamente en desarrollar una solución final en un corto espacio de tiempo. Esto conllevaría lógicamente a una mayor productividad.

[3]Esta abstracción, desde el punto de vista informático, no implica que el programador no conozca la existencia de estas técnicas de optimización, sino que se ocultan los detalles de implementación de ciertas funcionalidades.

Síntesis de los resultados del proyecto

Como resultado de este proyecto surge la **SceneToolKit** (**STK**), un paquete de herramientas para desarrollo de sistemas de realidad virtual, que actualmente se encuentra en una versión bastante avanzada —denominada STK 8.06 "Newton"— la cual ha respondido a las necesidades más significativas de los proyectos que la han estado utilizando.

En el informe final del proyecto [CJ09], se dedica el primer capítulo a la descripción del proyecto y se enuncian resultados negativos y positivos del mismo. En el segundo capítulo se describe la **SceneToolKit**, señalando también sus deficiencias y aspectos positivos. Además, en el presente documento se hacen algunas valoraciones respecto a estos resultados.

Síntesis de la contribución personal

Desde el mismo comienzo de la concepción de esta Herramienta, el autor de la presente memoria trabajó no sólo en la concepción general del motor, sino que se especializó en las áreas que influían en la visualización eficiente, como son: la estructura base para el almacenamiento de los datos (grafo de escena), optimización de la navegación por dicho grafo, aplicación de técnicas de optimización basadas en esta estructura (*frustum culling* jerárquico[4], oclusión y niveles de detalles). Dichos aportes están explicados más adelante en este documento y referenciados en el informe del proyecto [CJ09].

[4]Técnica de selección de objetos visibles determinando si están o no en el interior de la pirámide de visión de la cámara. La variante jerárquica aprovecha la semántica de los objetos de la escena, que básicamente plantea que si un objeto no se ve, ninguna de sus partes se ve.

2. Desarrollo

A continuación se presentan tres epígrafes: uno para la descripción y valoración del proyecto, otro dedicado al producto desarrollado, y un tercero donde se describen los aportes del autor en dicho proyecto.

2.1. El proyecto

El proyecto "Herramientas de desarrollo para sistemas de realidad virtual" (HDSRV), como se describe en el capítulo 1 del informe final del proyecto [CJ09], ha encontrado en su camino dificultades que se han ido superando con el tiempo, y en la medida que en la Universidad se han adquirido experiencias en la gestión de proyectos informáticos. Actualmente, se destaca por su gestión y buenos resultados en auditorías y chequeos, así como por su papel en la transmisión de experiencias a los nuevos proyectos. Ha resultado vital en la garantía de las bases para el desarrollo de los productos del Polo, por lo que se considera que ha sido efectivo y ha cumplido con la misión planteada para el mismo.

2.1.1. La SceneToolKit

Como se expone en el epígrafe 1.2.1 del informe final del proyecto [CJ09], en la Facultad 5 de la UCI se comenzó a trabajar en el tema de los simuladores en el curso 2003–2004. Los motores de simulación utilizados inicialmente no brindaron todas las prestaciones necesarias resultando inflexibles en algunos puntos, por lo que hubo que hacer modificaciones en el código con algunas

dificultades. La empresa SIMPRO —asociada a la Universidad en el desarrollo de estos simuladores— tuvo experiencias similares desde 1998, por lo que ya venían desarrollando su propio motor gráfico.

Es por esto que se toma la decisión —por parte de los líderes del proyecto en ese entonces[5] (2003), tanto de la UCI como de SIMPRO— de implementar una herramienta propia para desarrollar las aplicaciones de realidad virtual de la Universidad.

En el 2004 se reanaliza el estado del arte (ver epígrafe 1.2.2 del informe final del proyecto [CJ09]). Como resultado, se decide continuar con el desarrollo de la Herramienta, pero orientado a la implementación de un núcleo capaz de interactuar con módulos o bibliotecas existentes con fines específicos, de manera que se conozca su arquitectura y se puedan hacer rápidamente las modificaciones pertinentes para los nuevos requerimientos que puedan aparecer en los productos encargados, cada vez más complejos.

El intento de desarrollar un motor genérico que respondiera a las necesidades de cualquier aplicación final trajo algunas dificultades, por lo que en el 2008 se reanaliza el alcance y la estructura de la Herramienta, tomando como base los conceptos de motor gráfico (*graphic engine*) y motor de juego (*game engine*) [Alo08].

La Herramienta actualmente tiene implementadas las funcionalidades necesarias para desarrollar un simulador de conducción de modestas dimensiones[6], como lo muestran los productos realizados con ésta (ver en el informe del pro-

[5]En la Facultad 5 existía un único proyecto (UCI-SIMPRO) para el desarrollo de simuladores, que asumió inicialmente el desarrollo del *engine*.

[6]Se han montado entornos de ciudad de hasta 14 manzanas.

yecto el epígrafe 2.7). Si bien requiere de otras especializaciones necesarias para la realización de videojuegos (como la inclusión de mejores efectos especiales), y a pesar del relativamente corto tiempo de desarrollo[7], ha cumplido con los requerimientos más importantes de los clientes, por lo que se puede evaluar como aceptable.

En el sitio web de producción de la Universidad, se tiene publicado un estudio del mercado actual[8] en el cual se analizan las características principales de los motores más populares en el mundo. Éste servirá como referencia en el proceso de desarrollo de las próximas versiones.

De este estudio se identifica la ventaja comparativa de que esta Herramienta es completamente desarrollada en la Universidad, por lo que no existen interferencias legales de ningún tipo, así como que se dispone del código fuente y el *Know-How* de la misma.

La disponibilidad de recursos humanos en la Universidad, con una cultura científica bastante amplia en el tema de la gráfica y la realidad virtual, está permitiendo extrapolar el desarrollo de la Herramienta más allá del proyecto designado para esto. Así, se ha ido formando una comunidad de desarrollo que permitirá evolucionarla en un buen grado, convirtiéndose ésta en una vía para aplicar soluciones creativas y novedosas de investigadores del tema.

[7]Se llevan trabajando cuatro años y medio con un equipo de desarrollo pequeño, formado en su mayoría por estudiantes, mientras que las herramientas más populares y estables mundialmente llevan alrededor de 8–10 años de desarrollo (*Ogre3D* comienza como proyecto en 1999, *Irrlicht* en 2002, *Nebula Device* en 1998, *Fly3D* en el 2000, *Crystal Space* en 1997, y *Delta3D* libera su primera versión en el 2004), y generalmente tienen un trabajo en comunidad muy fuerte (ver http://ucipedia.uci.cu/index.php/Estudio_de_marketing_Herramientas_de_Desarrollo_para_Sistemas_de_Realidad_Virtual).

[8]Ver link en: http://ucipedia.uci.cu/index.php/Estudio_de_marketing_Herramientas_de_Desarrollo_para_Sistemas_de_Realidad_Virtual

2.2. Contribución personal

La contribución personal del autor de este trabajo, comienza con la investigación "Biblioteca gráfica para sistemas de realidad virtual" [CJ04] (2003–2004), la cual dio como resultado el diseño de la Herramienta para el Polo Productivo de Realidad Virtual.

Apoyado en el proceso de investigación del proyecto y a través del tránsito por diversos roles, ha sido autor de varios módulos de dicha Herramienta, fundamentalmente los relacionados con la visualización eficiente. En el epígrafe 1.5.1 del informe de proyecto [CJ09], se enumeran las publicaciones (6) e investigaciones (18) realizadas por el autor, algunas de éstas relacionadas con la optimización de la visualización.

Para cada tipo de aplicación de realidad virtual, se tienen diferentes optimizaciones posibles[9]. Un motor de videojuegos incluye más o menos optimizaciones, según el abanico de tipos de juegos al que se quiera dar soporte [Alo08]. Para la Herramienta se decidió implementar las optimizaciones correspondientes a la disminución de la carga de la tubería gráfica[10] (*graphic pipeline*) [CF08].

Las fases principales de la tubería gráfica son: [CF08]

1. Seleccionar los objetos que deben dibujarse en cada fotograma.
    - Convertir los objetos a primitivas poligonales.

[9] Aparte de la optimización de la visualización, se suele trabajar en mejoras en los aspectos matemáticos que conciernen a la aritmética, geometría o cálculos numéricos, así como optimizaciones en la gestión de memoria, tratamiento de ficheros, etc.

[10] Proceso computacional que va desde la descripción de los objetos en la escena hasta la imagen final. Se produce, en los sistemas en tiempo real, mediante una serie de fases consecutivas y conectadas entre sí (la salida de datos de una fase constituye la entrada de la siguiente).

- Producir listas de vértices a partir de las primitivas.

2. Calcular el color para cada vértice según un método de sombreado local.

3. Proyección 3D → 2D de cada vértice, que además proporciona su profundidad. Incluye el recorte o *clipping* del triángulo si alguna parte de él cae fuera de la ventana.

4. Rellenado de los triángulos (*rastering*). Para ello se realizan varias fases de procesamiento por *píxel*.

Partiendo de la recopilación y la evaluación de información sobre las técnicas de optimización básicas más utilizadas en el mundo (ver epígrafe 1.2.8 en [CJ04]), se determinó trabajar primeramente en la primera fase de la tubería, en la cual se intenta optimizar la calidad y cantidad de objetos que se van a utilizar. La selección de objetos se basa en dos técnicas principales [CF08]:

- Selección del detalle: un mismo objeto puede estar representado con más de un nivel de detalle, y por tanto, se puede escoger el que sea más adecuado, según su calidad y costo, en cada momento.

- Selección por visibilidad: Se puede dejar de enviar los objetos que estén ocultos por otros objetos, y los que queden fuera del campo de visión.

El trabajo del autor abarcó tres áreas de la optimización de la visualización: selección de objetos visibles en el cono de la cámara, selección de objetos no ocluidos, y modificación de los objetos por niveles de detalles. A continuación se describen algunos de los aportes hechos por el autor a la **STK** en estos elementos.

2.2.1. Grafo de escena

La primera problemática enfrentada a la hora de optimizar la visualización en la Herramienta, fue seleccionar e implementar estructuras de datos para el control de los objetos de la escena. A partir del estudio realizado se seleccionó el grafo general de escena, una estructura jerárquica muy abarcadora empleada en la mayoría de las herramientas actuales. En el epígrafe 3.1.5 "Grafo de Escena" del informe de proyecto [CJ09], se explica a *grosso modo* las características del grafo implementado en la Herramienta, así como las razones para su selección inicial. En el epígrafe 1.2.8 de la tesis de pregrado [CJ04] del autor de este documento, se referencian algunos materiales estudiados para la selección de dicha estructura.

Mientras que algunos motores, por ejemplo *G3D*[11] [McG07], optan por no brindar ninguna estructura de datos o ninguna base común para dichas estructuras, otros plantan las bases para la implementación de estructuras por parte del usuario, como *Ogre3D*[12] [Ogr08]. Algunos ofrecen además un grupo de nodos especializados como parte del motor, por ejemplo *WildMagic*[13] [Ebe04]. En la **SceneToolKit**, se decidió brindar la clase base para la construcción del grafo de escena, así como algunos nodos especializados para objetos genéricos (geometrías, luces y cámaras), dejando en manos de usuarios o de la comunidad, la implementación de otros nodos.

[11] *G3D* no brinda un grafo de escena pero tiene por ejemplo un *Kd-Tree*. Ver además wiki del proyecto en http://ucipedia.uci.cu/index.php/Engine_grafico_G3D.

[12] *Ogre3D* brinda un nodo base para el grafo y algunas implementaciones en forma de *plugins*, dejando el resto en manos de los usuarios. Ver además wiki del proyecto en http://ucipedia.uci.cu/index.php/Engine_grafico_Ogre3D.

[13] *WildMagic* ofrece nodos para luces, cámaras, geometrías y algunos para estructuras de datos espaciales y técnicas de optimización como portales, etc.

Una adaptación interesante del grafo de escena en la **STK**, es que los objetos están independientes y los nodos del grafo hacen referencias a ellos. Esto permite que varios objetos en la escena sean nodos que referencian a una malla cargada una única vez[14], lo que garantiza además la modularización de los datos y la definición de las responsabilidades de las clases entidades y controladoras. También se hicieron algunas adaptaciones para minimizar el recorrido por el grafo en cada fotograma, documentados en la sección "Navegación por el grafo" del epígrafe 3.1.5 del informe de proyecto.

Como resultado, el uso del grafo de escena ha permitido la aplicación tanto de técnicas de optimización (*frustum culling*, oclusión y niveles de detalles), como de otras soluciones de diversas índoles, basándose en su estructura. Mediante el mismo, se pudo adaptar la estructura de huesos para los objetos con animación por esqueletos; se logró implementar una clase para el control de autos virtuales con controladores propios. También se implementó un *QuadTree* para el seguimiento de terrenos, por lo que se estima que no será difícil acoplar las estructuras *Octree*, *BSPTree* y otras de este tipo.

Se recomienda hacer un uso más exhaustivo del grafo de escena como lo hace por ejemplo *OpenSceneGraph*[15] [OSG08]. Su fortaleza radica en su rendimiento, escalabilidad y portabilidad asociados al empleo de un grafo de escena, que soporta no solamente estructuras y técnicas de optimización, sino que está muy orientado a las características de *OpenGL* (por ejemplo, está en función de los *vertex arrays*, *vertex buffer objects* y *display lists*) para hacer un uso óptimo de sus potencialidades.

[14] En el caso de *WildMagic* [Ebe04] por ejemplo, las geometrías, luces, cámaras y otros heredan de la clase *Spatial* que hereda de *Node*.

[15] Ver además wiki del proyecto en
http://ucipedia.uci.cu/index.php/Engine_grafico_OpenSceneGraph.

2.2.2. Sistema de cámaras

Otras de las contribuciones del autor de este documento, fue una solución como respuesta a una necesidad práctica en los simuladores de auto que se comenzaban a desarrollar en el Polo. Se requería de más de una pantalla para aumentar el área de visión del chofer. Esto demandó el uso de más de una PC con conexión por red para cada simulador, pues las salidas de las tarjetas de video estaban limitadas a una o dos.

Tratando de minimizar la cantidad de PC a utilizar, se manejó la variante de que desde una misma PC se representaran más de una vista simultáneamente, de aquí que se debía optimizar el recorrido por el grafo y determinar los objetos visibles, pero orientado al uso de varias cámaras en una misma escena. En la sección "Sistema de cámaras" del epígrafe 3.1.5 del informe de proyecto [CJ09] se expone brevemente las características de la solución, disponible con más detalles en una publicación sobre el tema[16] [Cam07].

Con este sistema se pueden atachar varias cámaras en un único nodo y ser manejadas como un único objeto, a diferencia de otros motores (ver artículo referenciado), donde a lo sumo se brinda un nodo para el control de cada cámara. El aporte principal está en una optimización hecha a partir de que la presencia de más de una cámara, multiplica normalmente los recorridos por el grafo. Se propone una forma para hacer un único recorrido con todas las cámaras existentes, al final del cual se tiene, para cada una, los objetos a representar por ella.

[16] Versión digital disponible en: http://10.5.2.200:5800/svn/hdsrv/trunk/Expediente%20Proyecto/investigaciones/documentos/Sistema%20de%20camaras%20(Yanoski).pdf

Durante el desarrollo de la propuesta se hicieron pruebas para medir las mejoras, apreciándose un aumento de fotogramas por segundo (FPS) de 55.1 FPS sin la optimización a 58.3 FPS con la optimización, usando un sistema de tres cámaras. El mismo entorno se representó a 87.2 FPS con una única cámara.[17]

Este sistema ha sido utilizado en los tres simuladores producidos a partir del 2006 (ver epígrafe 2.7 del informe de proyecto), brindando las comodidades y eficiencia esperados. No se le han detectado problemas por lo que no se propone ninguna mejora en el mismo. Hasta donde se pudo estudiar, no se ha encontrado información sobre ningún sistema optimizado como este en los *engines* actuales.

[17]Estas mediciones se hicieron en un entorno que por características de su diseño, se cargó en un grafo de muy poca profundidad (prácticamente una lista). La ventaja de usar este sistema deberá apreciarse con mayor diferencia a medida que el grafo de escena sea más profundo y jerarquizado.

2.2.3. Oclusión estática

Como un paso importante en la visualización eficiente, se necesitaba eliminar del proceso de dibujado aquellos objetos que quedaban ocultos por otros, por su tamaño y posición. En el proyecto se dio una solución a esta necesidad, consistente en una herramienta[18] generadora de un fichero con información útil para oclusión entre objetos estáticos, que básicamente divide el entorno en cuadrículas y determina, para cada celda, la lista de objetos visibles desde ella —es decir, los no ocluidos— en los 360°.

La adaptación de este sistema a la **STK** fue hecha por el autor de esta memoria. La misma consistió en la creación de una rejilla regular acoplada al grafo de escena, cuyos datos se cargan del fichero generado por la herramienta de oclusión antes mencionada. Como adición, basado en el mismo sistema de determinación de objetos visibles por *view frustum culling* implementado en el grafo de escena, se toman los objetos visibles de una celda y se les da una segunda pasada determinando los visibles en el *frustum* de la cámara.

En el epígrafe 3.3.1 "Oclusión estática" del informe de proyecto [CJ09], se describe la adaptación de esta solución a la Herramienta. En un documento utilitario con el mismo nombre como parte de la documentación del proyecto[19], se describe la base teórica de la técnica, características del fichero generado, adaptación a la **STK**, etc.

[18] La herramienta para generación de información de oclusión se llama *VSM_Builder*, de Denys Almaral Rodríguez, disponible en http://10.5.2.200:5800/svn/hdsrv/trunk/Expediente%20Proyecto/ingenieria/implementacion%20y%20pruebas/tutoriales%20y%20herramientas%20de%20apoyo/Oclusion%20estatica/VSM_Builder/. Genera un fichero de extensión .VSM con el mismo nombre que el fichero de la escena .3DX.

[19] Versión digital disponible en: http://10.5.2.200:5800/svn/hdsrv/trunk/Expediente%20Proyecto/ingenieria/implementacion%20y%20pruebas/tutoriales%20y%20herramientas%20de%20apoyo/Oclusion%20estatica/Oclusion%20Estatica.pdf.

El módulo de oclusión estática ha resultado muy eficaz en los entornos de los simuladores realizados, aumentando los FPS hasta en un 30 % en zonas de mucha carga de polígonos.

2.2.4. Niveles de detalles (LOD)

Como un tercer paso de optimización de la visualización, luego del *view frustum culling* y la oclusión, se consideró la modificación del detalle de los objetos en dependencia de su importancia visual. Para incluir esta funcionalidad en la Herramienta, se realizó la investigación "Visualización de sistemas de realidad virtual. Optimización por niveles de detalles" [GG07].

De las tres técnicas básicas para el manejo de los niveles de detalles (LOD discreto, continuo y dependiente de la vista), y de las dos formas principales de determinación del detalle (por distancia a la cámara y por área en pantalla) [Lue03], se seleccionaron para una primera versión los métodos discreto y continuo, con determinación por distancia a la cámara, aunque se creó una base arquitectónica para la incorporación de las demás técnicas.

El módulo desarrollado es configurable y completamente transparente al usuario, pues el propio sistema determina a qué objetos se les debería aplicar optimización, activándose la misma si la malla correspondiente sobrepasa el límite de polígonos por defecto o por configuración. Además, permite configurar la técnica de aplicación del detalle y el método de determinación del detalle.

Una característica novedosa de esta solución respecto a otras existentes en el mundo [GG07], es que para los niveles discretos[20] normalmente se deja en manos de los diseñadores gráficos la modelación de los objetos con diferentes detalles, mientras que la **STK** es capaz de generar estos modelos a partir del modelo original, aunque también se le pueden especificar los modelos por diseño. Para esto se usa el mismo mecanismo de generación del espectro de

[20] Generalmente se usan tres modelos aplicables a las tres zonas: cercana, media y lejana.

modelos para LOD continuo, con la diferencia de que se seleccionan los modelos discretos cada cierta "distancia" en el espectro, en un procesamiento *offline*, es decir, antes de la ejecución de la aplicación final.

Se recomienda en esta parte incorporar el LOD dependiente de la vista, principalmente para la visualización de objetos grandes como terrenos, así como el uso de impostores[21]. Además, determinar el detalle del objeto según el área que ocupa en pantalla.

Ver más en el epígrafe 3.3.2 relacionado en el informe de proyecto, así como la tesis antes mencionada.

[21]Sustitución de objetos complejos por otros más simples, generalmente planos con una imagen del objeto, aplicable en largas distancias.

3. Conclusiones y recomendaciones

Partiendo de la situación problémica expuesta y del estudio del estado del arte en cuanto al uso de los motores gráficos, se llegó a la conclusión de que la creación de un paquete de herramientas propio (**SceneToolKit**), para el trabajo en el Polo Productivo de Realidad Virtual de la Facultad 5, es la solución más factible para acelerar el proceso de producción.

A lo largo de la vida del proyecto se han hecho análisis que le han dado a la **SceneToolKit** mayor flexibilidad, acercándola cada vez más a las tendencias mundiales en cuanto al trabajo con motores de simulación y videojuegos. Tanto los aportes del autor como el producto en sí contienen elementos novedosos, no sólo desde el punto de vista técnico, sino además por el hecho de lograr un *software* de este tipo sin precedentes en la Universidad. Su desarrollo ha exigido adaptaciones acorde a las condiciones del Polo, a partir de las soluciones a nivel mundial.

Durante cuatro años y medio de trabajo, se ha ganado en experiencia en métodos y metodologías de trabajo, así como en cultura científica en el área temática de la realidad virtual. Esto, unido a la concepción arquitectónica modular del producto, ha influido favorablemente en nuevas estrategias de trabajo que conllevarán al fortalecimiento y consolidación de este producto.

La **SceneToolKit** ha permitido en un buen grado sustituir el uso de motores gráficos[22], con la importancia que esto implica desde el punto de vista de la independencia y la soberanía tecnológica, además del ahorro de tiempo en el

[22]De ocho productos gráficos terminados en el Polo, seis fueron desarrollados con esta Herramienta, y dos productos nuevos se están desarrollando sobre ésta.

desarrollo de aplicaciones finales.

En cuanto a la optimización de la visualización, se puede decir que dicho aspecto ha sido resuelto en un gran porciento, pues se implementaron algunas técnicas básicas y se garantizó una arquitectura flexible para la adición de otras técnicas. Esto ya ha sido probado en los dos últimos simuladores hechos con la **SceneToolKit**.

Finalmente, se recomienda estudiar las experiencias expuestas en este documento y en el Informe Final del Proyecto, como puntos de análisis para la continuación del trabajo en el proyecto, y para el perfeccionamiento de futuros proyectos productivos y de investigación o innovación.

A. Glosario de términos

Términos en inglés:

***Engine*:** Forma corta de referirse al *game engine*.

***Frame Buffer*:** Porción separada o seleccionada de una memoria del *display* que contiene un fotograma completo de datos de representación.

***Frustum*:** Volumen formado por los 6 planos que definen los límites de la cámara: cercano (*near*), lejano (*far*), izquierdo (*left*), derecho (*right*), arriba (*top*) y abajo (*bottom*).

***Frustum culling*:** Técnica de selección de objetos visibles determinando si están o no en el interior de *frustum* de la cámara.

***Game engine*:** Es el sistema operativo de un videojuego, el núcleo. Se encarga de dirigir y coordinar cada uno de los aspectos tecnológicos ligados a él. Capta eventos de entrada, computa física, reproduce sonidos, simula IA, pinta, etc.

***Graphic engine*:** (motor gráfico) Módulo gráfico independiente lo bastante potente para poder tratar toda la información que hay en él, así como su visualización en tiempo real. Este concepto surge por la complejidad gráfica añadida a la hora de tratar con escenarios 3D.

***Graphic pipeline*:** (tubería gráfica) Proceso computacional que va desde la descripción de los objetos en la escena hasta la imagen final. Se produce, en los sistemas en tiempo real, mediante una serie de fases consecutivas

y conectadas entre sí (la salida de datos de una fase constituye la entrada de la siguiente).

***Render*:** (rendering) Crear en forma automática una imagen de acuerdo al modelo tridimensional que existe en el ordenador.

***Shaders*:** Conjunto de instrucciones gráficas destinadas para el acelerador gráfico, estas instrucciones dan el aspecto final de un objeto. Los *shaders* determinan materiales, efectos, color, luz, sombra, etc.

Términos en español:

Fotogramas: (*Frames*) Cada una de las imágenes que se muestran como proceso final del renderizado de una escena 3D a imagen 2D. La secuencia de estas imágenes conforma la parte visual de un videojuego. La velocidad con que se muestran es conocida como *frames* por segundo, especificando la cantidad de imágenes mostradas por cada segundo de aplicación.

Grafo de escena: Estructura jerárquica donde se almacenan y organizan los objetos de la escena para el control de su información y determinación de la visibilidad, donde las hojas contienen los objetos dibujables de la escena.

Herramienta: (en gráficos por computadora) Ver *engine*, *graphic engine* y *game engine*. Está mayormente referido al *game engine*: herramienta de videojuegos. Usada en este documento con letra inicial mayúscula como sinónimo del *game engine* que forma parte del paquete **SceneToolKit**.

Impostores: Técnica de optimización de la visualización por niveles de detalles (LOD), donde se sustituyen objetos de poca importancia visual

(pequeños o alejados de la cámara) y de muchos polígonos, por objetos más sencillos (por ejemplo, un plano con una fotografía del objeto original) sin afectar la calidad visual.

Motor gráfico: Ver *graphic engine*.

Oclusión: Técnica de optimización de la visualización que descarta a aquellos objetos que son ocultados por otros. La oclusión estática se aplica solamente entre objetos que no cambian su información espacial. La oclusión dinámica implica oclusión entre objetos dinámicos, y entre dinámicos y estáticos.

Polo Productivo de Realidad Virtual: Estructura organizativa existente en la UCI, que agrupa a los proyectos dedicados al desarrollo de sistemas de realidad virtual.

Realidad Virtual: Representación de escenas u objetos producidos por un sistema informático, dando la sensación de su existencia real.

SceneToolKit: (STK) Alias del paquete de herramientas desarrollado por el proyecto, formada por *scene*, escena, y *toolkit*, "paquete de herramientas". El nombre completo es "Herramientas para Desarrollo de Sistemas de Realidad Virtual", y debe constar del *game engine* y otras herramientas utilitarias y de edición.

Simuladores: Un simulador es un aparato que permite la simulación de un sistema, reproduciendo su comportamiento. Los simuladores reproducen sensaciones que en realidad no están sucediendo.

Sistemas de Realidad Virtual: Aplicaciones informáticas basadas en técnicas de realidad virtual (ver Realidad Virtual).

Tiempo real: (*Real time*) En gráficos por computadoras, las aplicaciones en tiempo real son aquellas que generan los fotogramas a medida que son necesarios durante la ejecución de la aplicación. El tiempo de respuesta de la aplicación debe ser lo suficientemente corto como para que la percepción del fenómeno se corresponda con fidelidad al sistema real.

Videojuego: Programa informático normalmente asociado a un hardware específico, que recrea un ejercicio sometido a reglas, se debe lograr uno o varios objetivos, donde los jugadores pueden interactuar y tomar decisiones.

B. Glosario de abreviaturas

FPS: *Frames* por segundo.

GPU: Unidad de Procesamiento Gráfico (*Graphics Processing Unit*).

HDSRV: Nombre del proyecto: Herramientas de Desarrollo para Sistemas de Realidad Virtual.

LOD: *Level of Detail*, niveles de detalles, técnica de optimización de la visualización.

RV: Realidad Virtual.

SIMPRO: Empresa de SIMuladores PROfesionales: alias comercial del Centro de Investigaciones y Desarrollo #2 (CID–2) de la Unión de Industrias Militares (UIM) de las FAR.

SRV: Sistemas de Realidad Virtual.

STK: SceneToolKit.

UCI: Universidad de las Ciencias Informáticas.

Referencias

[Alo08] Alonso Alonso, J.: *Fundamentos y programación de videojuegos*, capítulo VideoJuegos 3D, páginas 7–30. UOC, Barcelona, 2008.

[Boo08] Boo, Javi: *Sistemas Gráficos Interactivos (SGI)*, capítulo 2 Introducción a la Realidad Virtual. Departamento de Lenguajes y Sistemas Informáticos, Universidad Politécnica de Catalunya, 2008. http://www.lsi.upc.edu/~pere/SGI/guions/ArquitecturaRV.pdf.

[Cam07] Camacho Román, Yanoski: *Sistema de cámaras para la visualización de entornos de Realidad Virtual.* III Congreso Internacional de Tecnologías, Contenidos Multimedia y Realidad Virtual, en XII Convención y Expo Internacional Informática, 2007.

[CF08] Coma, Inmaculada y Marcos Fernández: *Sistemas de visualización en tiempo real. Visualización por hardware*, 2007–2008. Tema 3 de la asignatura Ampliación de Informática Gráfica, Departamento de Informática, Universidad de Valencia. http://informatica.uv.es/iiguia/AIG/web_teoria/tema3.pdf.

[CJ04] Camacho Román, Yanoski y Fernando Jiménez López: *Biblioteca Gráfica Para Sistemas de Realidad Virtual.* Tesis de pregrado, CUJAE, 2004. http://10.5.2.200:5800/svn/hdsrv/trunk/Expediente%20Proyecto/investigaciones/tesis%202003-2004/Yanoski-Fernando%20(Biblioteca%20Grafica%20para%20SRV).pdf.

[CJ09] Camacho Román, Yanoski y Fernando Jiménez López: *Informe Final del Proyecto Herramientas de Desarrollo para Sistemas*

de Realidad Virtual. Informe técnico, Polo Productivo de Realidad Virtual Facultad 5, UCI, 2009. http://10.5.2.200:5800/svn/hdsrv/trunk/Expediente%20Proyecto/informe%20de%20proyecto/InformeFinalProyecto.pdf.

[Ebe04] Eberly, D. H: *3D Game Engine Architecture: Engineering Real-Time Applications with Wild Magic*. Morgan Kaufmann Publishers Inc., San Francisco, 2004.

[GG07] Gómez Cuello, Katherine y Ariangna Garcés Gilart: *Visualización de sistemas de realidad virtual. Optimización por niveles de detalles*. Tesis de pregrado, UCI, 2007. http://10.5.2.200:5800/svn/hdsrv/trunk/Expediente%20Proyecto/investigaciones/tesis%202006-2007/Niveles%20detalles%20(Katherine-Ariangna).pdf.

[Lue03] Luebke, David: *Level of Detail for 3D Graphics*. Morgan Kaufmann Publishers, USA, 2003.

[McG07] McGuire, M.: *G3D Overview*. http://g3d-cpp.sourceforge.net/manual/guideoverview.html, Agosto 2007.

[Ogr08] Ogre Community: *Documentation Architecture*. http://www.ogre3d.org/wiki/index.php/Documentation_Architecture, Octubre 2008.

[OSG08] OSG Community: *OpenSceneGraph Documentation*. http://www.openscenegraph.org/projects/osg/wiki/Support, Agosto 2008.

Fecha de impresión: 7 de octubre de 2009

Generado con LaTeX