Revista de investigación
Editada por Área de Innovación y Desarrollo, S.L.

Envío: 08-07-2013

Aceptación: 4-08-2013

Publicación: 30-09-2013

ALGORITMO DE RECORTES Y DE NIVELES DE DETALLES PARA EL INCREMENTO DE LA VELOCIDAD DE VISUALIZACIÓN DE MODELOS 3D EN DISPOSITIVOS DE BAJO COSTE.

ALGORITHMS CLIPPING AND DETAIL LEVELS FOR INCREASING THE 3D MODELS VISUALIZATION IN LOW COST DEVICES

Gendrys Espinosa Vega[1]

1. Ingeniero en Ciencias Informáticas. Facultad Regional Granma de la Universidad de Granma. Cuba. E-mail: gvega@grm.uci.cu

RESUMEN

La lenta visualización de los modelos 3D afecta a los recorridos virtuales desarrollados en la Facultad Regional Granma, pues son soportados por computadoras con bajas prestaciones. Este trabajo tiene como objetivo general el diseño de un algoritmo que permita el incremento de la velocidad de visualización de los modelos 3D. Posterior al diseño del algoritmo se realizaron pruebas mediante aplicaciones desarrolladas con el motor gráfico Panda3d, comprobándose que incrementa la velocidad de visualización de los modelos 3D en los entornos virtuales.

ABSTRACT

The slow displaying of 3D models affects the virtual tours developed at Regional Faculty of Granma, because they are not supported by low performance computers. The objective of this research is to design an algorithm that allows increasing the speed of displaying 3D models. After the algorithm was designed, it was tested through applications developed with the Panda3D graphic engine and the increase of the display speed of 3D models was achieved in virtual environments.

PALABRAS CLAVE

3D, virtuales, algoritmo, velocidad, visualización.

KEYWORDS

3D, virtual, algorithm, speed, display.

INTRODUCCIÓN

En la informática se han desarrollado diversos campos, un ejemplo de ellos es la realidad virtual que surge como una necesidad de explorar la realidad, y permite una interacción sin límite con el mundo virtual. Sus aplicaciones han experimentado un vertiginoso salto en los últimos tiempos y no basta solo con observar mundos completamente realistas, sino que los usuarios sean parte de ellos e interactúen dentro del mismo, asumiendo roles y responsabilidades. Su alcance es hoy mucho mayor que los juegos interactivos, la representación estática de objetos o la realización de imágenes. [1]

Una de las ramas del conocimiento que abarca la realidad virtual es la visualización de modelos tridimensionales (3D) en tiempo real. Esta se encarga de todo lo relacionado con la representación de superficies complejas sin afectar el nivel de interactividad y dinamismo de los modelos en una escena virtual.

Al interactuar con objetos en una escena virtual es necesario un proceso de optimización para poder realizar diversas funciones gráficas. La optimización de la visualización trata de lograr a través de técnicas y algoritmos que se muestre la mayor cantidad de información en el menor tiempo posible. Además, posibilita que los entornos virtuales sean lo más fieles posibles a la realidad, de manera que la interacción con estos les resulte fluida a los usuarios, independientemente de que la carga de objetos en la escena sea muy grande, o que la cantidad de procesos que se necesiten visualizar sea muy elevada.

En el grupo Realidad Virtual del Centro de Desarrollo (CEDES) de la Facultad Regional de la Universidad de Granma se trabaja con proyectos de recorridos virtuales 3D. Para el desarrollo de estos proyectos es necesario el modelado de objetos virtuales con un alto nivel de realismo, donde los usuarios experimenten sensaciones muy similares a las del mundo real y que los modelos sean mostrados de la manera más rápida posible.

Generalmente los modelos obtenidos a partir del software de modelado 3D son considerablemente complejos, y para poder procesarlos, analizarlos y visualizarlos es necesario contar con altas prestaciones de hardware. Con el avance de las tarjetas gráficas se ha incrementado la utilización de las mismas para aumentar el realismo de la visualización en los sistemas de realidad virtual, permitiendo visualizar cientos de miles de primitivas por segundo. Con la utilización de una tarjeta aceleradora gráfica podrían solucionarse algunos de estos problemas, aunque no en su totalidad, por lo que podría darse el caso que se sobrepasen las capacidades gráficas de estas tarjetas y resulte insuficiente el hardware. Debido al alto costo del software y del hardware involucrado en los procesos de visualización de modelos 3D, en muchas ocasiones los recorridos virtuales que se realizan en la FRG son soportados por computadoras que no cuentan con la tecnología necesaria.

Técnicas de recorte o técnicas de culling

Las técnicas de recorte son aquellas que se utilizan para eliminar de la escena la geometría que no es necesaria dibujar en pantalla porque no aparece en la imagen final o no aporta casi nada a la misma. Con esto se logra un gran incremento de velocidad, sobre todo en escenas de interiores donde se puede desechar gran cantidad de la geometría al mostrarse espacios más reducidos. Son usadas ampliamente en el mundo de los videojuegos, así como en la realidad virtual y en herramientas para producción de películas. Las técnicas de recorte utilizadas son:

- Backface culling o recorte de cara de atrás
- Frustum culling
- Portal culling
- Detail culling o recorte de detalle
- Oclussion culling

Se consideró que las más convenientes a utilizar para satisfacer las necesidades planteadas en este trabajo son: backface culling, frustum culling y portal culling. Estas técnicas cuentan con las características necesarias para ser aplicadas a los recorridos virtuales que se realizan en la FRG. Estos recorridos son mayormente desarrollados en escenas divididas por salones y los objetos que se modelan en su mayoría no son penetrados por la cámara.

Para agilizar el proceso de búsqueda de un objeto es recomendable hacer uso de las estructuras de datos espaciales, ya que estas permiten organizar los objetos en forma jerárquica. Al organizar los objetos en un árbol, la complejidad en tiempo de búsqueda puede ser reducida a un orden logarítmico a diferencia de una lista o un arreglo que tomaría un tiempo O(n). Para comprobar la visibilidad de un objeto haciendo uso de alguna estructura de datos espaciales, los volúmenes que encierran los hijos no tienen que ser examinados si el volumen que encierra al padre no es visible.

Estructuras de datos espaciales

Las estructuras de datos espaciales organizan la geometría en 2D, 3D o más dimensiones y son necesarias en la mayoría de las técnicas de aceleración [2]. Estas estructuras permiten ordenar de forma jerárquica los elementos de una escena, con esto se logra que se pueda acceder a ellos y realizar cualquier acción sobre los mismos de una manera más rápida. Las estructuras que existen son las que se explican a continuación.

- Jerarquía de volúmenes acotantes (Bounding Volume Hierarchy (BVH))
- Partición Binaria del Espacio (Binary Space Partitioning (BSP))
- Octrees

Niveles de detalle (Levels Of Detail, LODs)

Los niveles de detalle van a ser las simplificaciones que se le realizan a un modelo complejo, buscando una versión simple para representarlo, cuando este se encuentre a una larga distancia del observador.

Tienen como objetivo la simplificación de la imagen sin crear una degradación visual, que permita una mayor optimización del escenario virtual, pues no es necesario visualizar un objeto de múltiples polígonos si este se encuentra lejos del observador. Esto permite ahorrar capacidad para cargar otros elementos e incrementar la velocidad en la escena. [3]

Los niveles de detalle almacenan varias versiones de un mismo objeto en dependencia del modelo que vayan a utilizar. Esto permite que según el área que ocupe, se pueda elegir un modelo u otro para ser representado en pantalla. Y es lo que se le conoce como usar diferentes niveles de detalle, entre los que existen algoritmos que se clasifican en niveles de detalles discretos o estáticos, continuos o dinámicos, o dependientes del punto de vista.

PROPUESTA DEL ALGORITMO

El desarrollo del algoritmo consta de 3 fases que se relacionan entre sí, de tal forma que los datos de salida de una fase van a ser los datos de entrada de la siguiente. Las fases en que se desglosa el algoritmo son:

1. Localización de la celda a recortar y obtención de los objetos que se encuentran dentro del campo de visión de la cámara.
2. Aplicación de las técnicas de niveles de detalle.
3. Obtención de los polígonos que miran hacia el espectador.

La primera fase hace uso de la técnica de recorte frustum culling, esta permite eliminar los objetos que son innecesarios cargar porque no se encuentran dentro del frustum. Tiene como dato de entrada la celda o salón donde se encuentra ubicado el espectador. Luego haciendo uso de la estructura de datos espaciales BVH se genera un árbol jerarquizado con todos los objetos que contiene dicha celda. Se calcula el tamaño del frustum y se comprueba si dentro del mismo existen portales (puertas o ventanas). Para cada portal visible se calcula el tamaño del frustum que pasa a través de él. Finalmente, se genera una lista que contendrá todos los objetos que se encuentran dentro del frustum que sale de la celda que se está analizando más los que se encuentran dentro de los pequeños frustum que pasan por los portales visibles.

En la segunda fase se aplica la técnica de LOD continuos a los objetos que quedaron de la fase anterior. Se parte de la lista que se obtuvo en la fase 1, para verificar la distancia que tiene cada uno de los objetos y aplicarle el nivel de detalle correspondiente. A estos modelos se les van aplicar tres versiones diferentes de LOD; la primera representación del objeto, siendo el más cercano al observador tendrá un alto nivel de detalle, que permite que el objeto sea lo más realista posible. La segunda representación del objeto se encuentra a una distancia media, donde se podrá representar con menos detalle que el anterior. Y por último se tiene la tercera salida del objeto, aún más alejado, que tendrá un bajo nivel de detalle con respecto a los demás. El objeto con el nivel de detalle correspondiente se guardará en una nueva lista.

En la tercera fase se parte de la lista de objetos 3D que fue generada en la fase 2 y se aplica la técnica de recorte backface culling. Se accede a cada uno de los objetos de esa lista. Para cada objeto deben ordenarse todos sus polígonos en una misma dirección, puede ser en sentido horario o anti horario. Se comprueba para cada uno de los polígonos su visibilidad, si es visible se muestra en pantalla, en caso contrario se recorta.

Fase 1: Localización de la celda a recortar y obtención de los objetos que se encuentran dentro del campo de visión de la cámara

Inicialmente para poder cargar la escena donde se encuentra ubicado el espectador hay que subdividir la misma en sectores (salones/celdas). Luego buscar en qué celda está ubicado el observador, esto es posible comprobando si la cámara está dentro del sector que se esté verificando. Conociendo ya donde se encuentra ubicado el espectador se prosigue a realizar la construcción del árbol BVH de los objetos 3D que se encuentran en ese sector. Para ello se debe comenzar inicializando un BV en el rectángulo de la pantalla, haciendo uso de un volumen acotante llamado caja envolvente alineada con los ejes (Axis Aligned Bounding Box (AABB)). Esta caja es rectangular y sus caras están orientadas de manera que sus normales son paralelas a los ejes coordenados. Cada caja contendrá en su interior un objeto o grupo de objetos y se podrá acceder a cada uno de ellos de una manera más rápida. Luego se calcula el frustum (F) que se expande desde el espectador. Todos los objetos que se encuentren dentro del frustum serán almacenados en una lista. Para verificar si el objeto se encuentra dentro del frustum se hace uso de la técnica de recorte frustum culling.

La idea básica del frustum culling será aplicar un pequeño algoritmo para verificar si un determinado objeto se encuentra dentro o fuera del frustum y de este modo saber si se debe dibujar o no. Para ello se deben tener en cuenta los 6 planos con que cuenta el frustum (plano de recorte lejano, plano de recorte cercano, plano superior, plano inferior, plano izquierdo y plano derecho) y realizar la verificación contra los AABB que contendrán a los objetos.

En primer lugar, se reciben los 8 vértices (esquinas) del AABB y se verifica cada plano del frustum contra ellos. En función de la clasificación de cada vértice contra el plano del frustum que se está analizando se pueden extraer conclusiones tempranas. Se toma el primer plano del frustum y se clasifica cada vértice para saber si está delante o detrás del mismo. Si los 8 vértices de la caja están detrás de alguno de los planos, se puede concluir que el AABB está fuera del frustum (nuevamente el mejor caso), por otro lado, si los 8 vértices se encuentran delante del plano, se incrementa un contador que luego servirá para determinar si esta situación se repite 6 veces más, situación en la cual se puede establecer que el AABB está totalmente dentro del frustum. El caso que queda indica que el AABB está parte dentro y parte fuera del frustum. Haciendo uso del árbol BVH formado se podrá descartar una gran cantidad de geometría de un solo cálculo.

Luego de haber calculado el frustum F y verificado cuáles son los objetos que se encuentran dentro de él, los mismos se guardan en una lista denominada (L). Se verifica si dentro de F existen portales visibles, esto se realiza verificando si existe intersección entre el AABB inicial del salón y el AABB que se construye al proyectar cada portal en pantalla. En caso de ser afirmativo se calcula el frustum reducido (Fi) que pasa por esos portales. Para ello tiene que haberse construido otro árbol BVH de la celda que se ve a través de los portales visibles. Luego se agrega a L los objetos que se encuentran dentro de Fi. Finalmente, se obtiene una lista de objetos 3D que contiene todos los objetos que se encuentran dentro del campo de visión de la cámara.

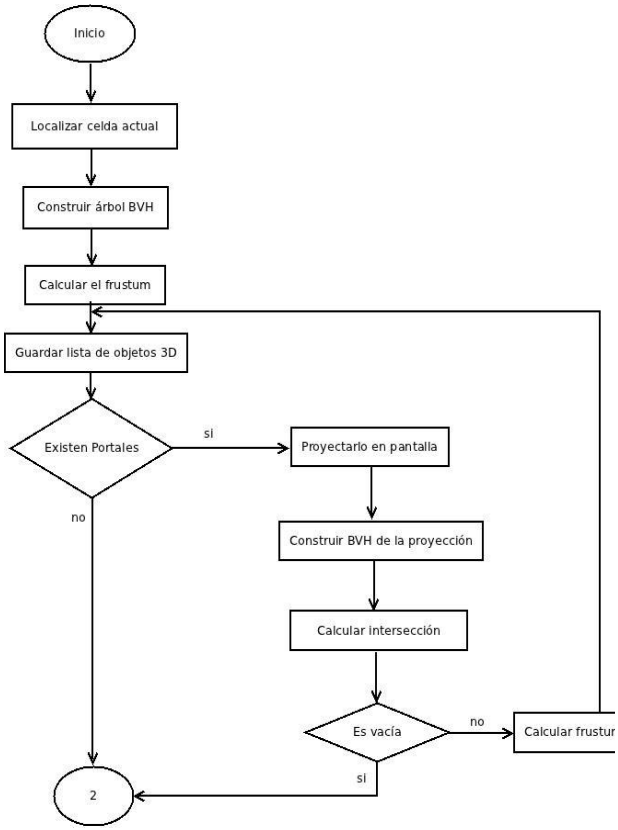

Figura 1: Diagrama de flujo para la primera fase del algoritmo. Fuente: Elaboración propia.

Fase 2: Aplicación de las técnicas de niveles de detalle

Se tiene como dato de entrada la lista (L) de objetos 3D que se generó en la fase anterior, se escoge cada uno de los objetos por separado y se comprueba a qué distancia se encuentra del observador, para poder aplicar el nivel de detalle correspondiente. Además, se tiene en cuenta el tamaño del objeto a ver si es preciso aplicarle el LOD. Para estimar cada LOD se va a definir un rango de distancia, para que a la hora de verificar cada objeto se compruebe si este se encuentra dentro de ese rango, y si lo supera el objeto no se mostraría en la escena. Además se va a verificar qué prioridad tiene el objeto para salir en la escena, si es alta se comprueba lo dicho anteriormente, sino se escoge el que tenga mayor prioridad que él. Para cada objeto de la lista se verifica la distancia que se encuentra el mismo del espectador (cámara virtual), teniendo en cuenta el rango de distancia cerca, medio y lejos definido por el programador. Primeramente se comprueba si el elemento E se encuentra cerca del espectador, si esto se cumple se le aplica al objeto el nivel de detalle con mayor fidelidad (mayor carga poligonal) para que tenga un alto grado de realismo en la escena final. Si no cumple con la primera condición se verifica si E se encuentra dentro del segundo rango establecido, si se cumple se le

aplica el nivel de detalle medio, y en caso contrario, entonces se le aplicaría el nivel de detalle bajo. Luego de aplicarle a E el nivel de detalle correspondiente se almacena en otra lista (Le) que contendrá los objetos visibles con el nivel detalle correspondiente. Los objetos que no se encuentren dentro de ninguno de los rangos predefinidos no se van a mostrar en la escena Este proceso se realizará para cada uno de los objetos de la lista L.

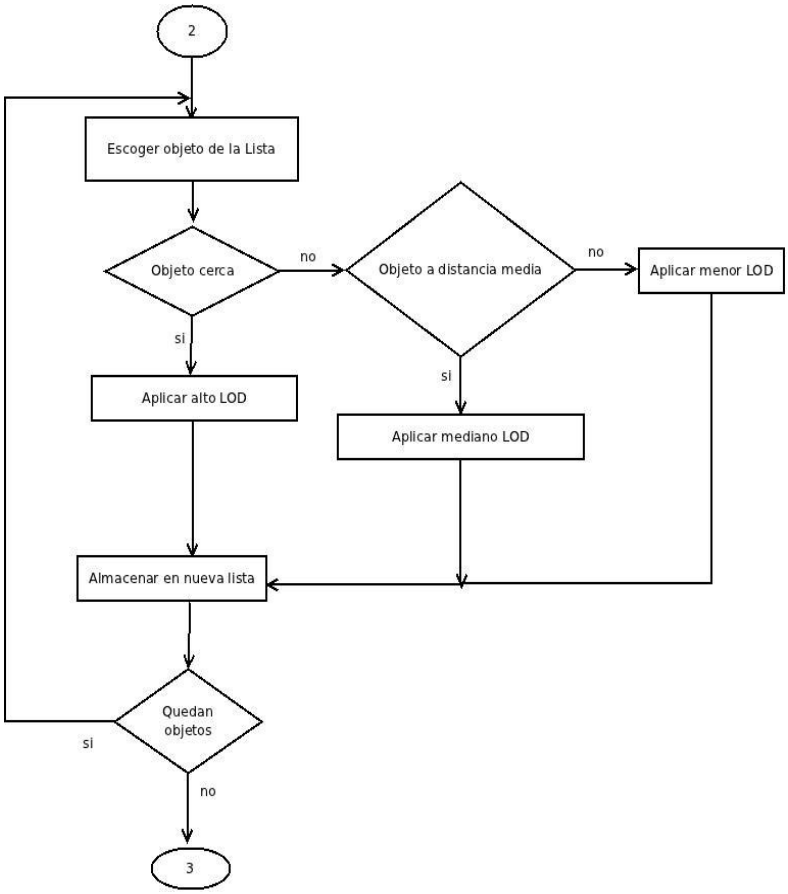

Figura 2: Diagrama de flujo para la segunda fase del algoritmo. Fuente: Elaboración propia.

Fase 3: Obtención de los polígonos que miran hacia el espectador

En esta fase se permitirá recortarle a los objetos que son visibles por el espectador las caras que el mismo no puede ver y de esta manera solo se pintarán en pantalla los objetos con las caras visibles. A continuación se muestra en la figura 3 el diagrama de flujo de la fase 3.

Se partirá de la lista (Le) de objetos 3D generada en la fase anterior. Se irán analizando cada uno de los objetos (Oi) de la lista. Para cada uno de ellos se deben organizar sus polígonos (caras) en una misma dirección, para ello debe definirse una función que devuelva 3 posibles valores que tengan que ver con el sentido de las normales de las caras de un modelo. Estos valores pueden ser: sentido horario, sentido antihorario o ninguno (None). El caso del valor None se usa cuando se quiere dibujar los polígonos por ambos lados, por tanto, se desactivaría el backface culling. Luego de tener ordenadas las caras del objeto se prosigue a verificar la visibilidad de cada una de ellas.

Para determinar qué polígonos son visibles desde el punto de vista del observador se deben realizar los siguientes pasos: a partir de los vértices del polígono (V1, V2, V3) se calcula el vector normal al polígono n= (v2-v1) x (v3-v1) = (0, 0,+/-n). Se calcula el producto escalar de dicho vector con el vector que va desde el centro del polígono hasta el punto de observación (vector de la vista del polígono). Si el resultado de esa operación es negativo, el ángulo entre dichos vectores es superior a 90 grados, por lo que la superficie no es visible y se le aplica la técnica de recorte backface culling para recortarla.

En caso que la superficie sea visible se mostraría la misma en pantalla. Este proceso se realizará con todas las caras del objeto Oi y con todos los objetos que se encuentran en Le.

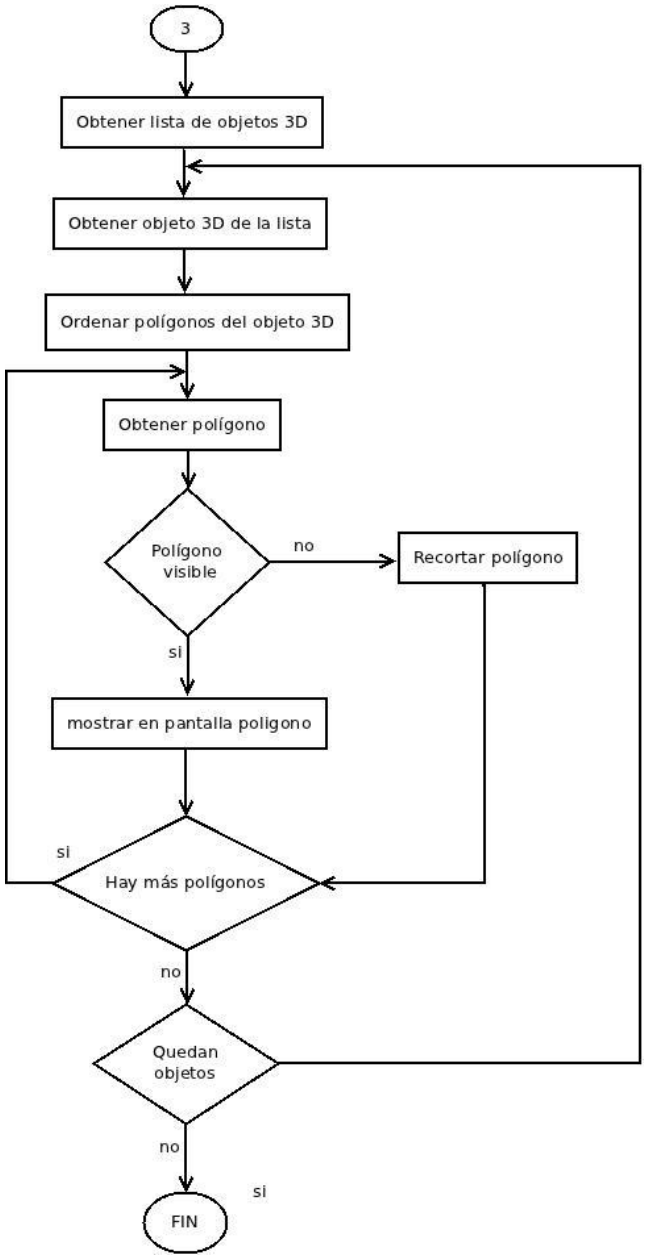

Figura 3: Diagrama de flujo para la última fase del algoritmo. Fuente: Elaboración propia.

RESULTADOS Y DISCUSIÓN

Para el análisis de los resultados se crearon dos aplicaciones que contribuyeron a la realización de las pruebas del algoritmo propuesto. Una aplica el algoritmo diseñado, y la otra solamente carga los objetos con todos sus polígonos correspondientes. Para la realización de estas pruebas se escogieron cuatro puntos de control para establecer la comparación y verificar si cumple con las expectativas previstas. Estos puntos fueron: en el momento que se carga la aplicación, otro cuando se cambia al primer nivel de detalle, el tercero es cuando se muestra el segundo nivel de detalle y por último cuando se observan todos los objetos.

En la tabla 1 se realiza una comparación para verificar el grado de cumplimiento del algoritmo, según la velocidad de cada aplicación mediante la cantidad de frames por segundo.

Aplicaciones	Velocidad	Distancia			
		Al iniciar	Menor	Media	Mayor
1	Con técnicas (fps)	42.0 – 50.2	59.0 – 59.9	59.0 – 59.9	50.0 – 58.0
2	Sin técnicas (fps)	42.0 – 50.2	31.5 – 34.0	30.5 – 32.0	24.8 – 28.2

Tabla 1: Análisis de la velocidad (fps). Fuente: Elaboración propia.

La velocidad varía a medida que se comience a interactuar con cada una de las aplicaciones. En el caso de la aplicación 1, que va a ser al que se le aplicó las técnicas de recortes y de niveles de detalle, cuando la distancia es menor y media va a mantener una velocidad entre 59.0 - 59.9 fps. La misma va a decaer a un valor de 50.0 - 58.0 fps cuando la distancia sea mayor y se vean todos los objetos en el escenario. Para la aplicación 2, que es al que no se le aplicó ninguna técnica, la velocidad va a decrementar considerablemente según aumente la distancia. Cuando esta sea menor va a tener una velocidad de 31.5 - 34.0 fps, cuando sea media va a oscilar entre los 30.0 - 32.0 fps y cuando el observador se encuentre a una mayor distancia va a estar entre 24.8 – 28.2 fps. Analizando todos estos valores se puede observar que al aplicarle el algoritmo al demo, este resulta más eficiente ya que aumenta la velocidad de visualización.

En la tabla 2 se realiza una comparación entre la cantidad de triángulos y de vértices de cada aplicación, según la distancia en que se encuentra la escena 3D.

Aplicaciones	Triángulos y vértices	Distancia			
		Al iniciar	Menor	Media	Mayor
1	No. de Triángulos	51035	51035	31874	7092
	No. de Vértices	61381	61381	31957	11879
2	No. de Triángulos	49434	49434	49434	49434
	No. de Vértices	113522	113522	113522	113522

Tabla 2: Análisis del número de triángulos y de vértices. Fuente: Elaboración propia.

Para la aplicación 2, estos dos factores se van a mantener constante para cualquier distancia en que se encuentre el observador, y va a tener 49434 triángulos y 113522 vértices. Mientras que para la aplicación 1, estos van a variar según la distancia en que se encuentre cada objeto del observador. Para cuando cargue la aplicación y cuando la distancia sea cerca va a tener la misma cantidad de triángulos y vértices (51035 y 61381 respectivamente), ya que el nivel de detalle no va a variar hasta que el observador se encuentre a una distancia media donde se va a tener 31874 triángulos y 31957 vértices (Ver anexo 4). Y para cuando el observador se encuentre a una mayor distancia va a tener 7092 triángulos y 11879 vértices, lo que provoca que aumente la velocidad de la escena.

Para las pruebas de rendimiento del algoritmo se desarrolló una gráfica que muestra información acerca del comportamiento de la distancia y los frames por segundos (fps). Los resultados obtenidos se pueden observar en la figura 4. En la aplicación sin técnicas la velocidad disminuye según vaya aumentando la distancia. Porque aunque no se aprecie en la figura, aquí va a intervenir el número de triángulos de cada objeto, y a una mayor distancia el número de triángulos en el modelo se va a mantener constante, aunque si se incorporan otros modelos dentro del frustum la cantidad total de triángulos dentro de la escena incrementará sustancialmente.

Y para la aplicación con técnicas, según aumente la distancia mayor va a ser la velocidad, porque a mayor distancia menor será el número de triángulos y de nivel de detalle de los objetos.

CONCLUSIONES

Se desarrolló un estudio sobre las técnicas de recorte y de niveles de detalle, donde se seleccionaron las más convenientes para el desarrollo del algoritmo. De las técnicas de recorte se escogieron portal culling, frustum culling y backface culling y los niveles de detalle continuos para realizarles las simplificaciones a los modelos que se encuentren en escena. Para organizar la geometría en 3D y optimizar el trabajo de las técnicas de recorte se seleccionó la estructura de datos espaciales BVH.

Se diseñó un algoritmo con la aplicación de las técnicas de recortes y niveles de detalle, compuesto por tres fases que a su vez están desglosadas por una serie de pasos lógicos.

Se implementaron dos aplicaciones, una en la que se aplica el algoritmo diseñado y otra que no lo aplica. Las mismas permitieron verificar la validez del algoritmo diseñado.

Se realizaron pruebas que demostraron que el algoritmo diseñado cumple con los objetivos propuestos, y permite incrementar la velocidad de visualización de los modelos 3D.

REFERENCIAS

[1] **HIDALGO NAVARRO, José Alejandro.** *Animación en Tiempo Real (Técnicas de Incremento de Velocidad)*. [Fecha de consulta: 26 de febrero 2012], Digital, 2003.

[2] **MÖLLER, Tomas y HAINES, Eric.** *Real-Time Rende ring*. [Fecha de consulta: 26 de febrero 2012], 1999.

[3] **VEGA INFANTE, Dailen y FERNÁNDEZ Balbuena, Celina.** *Algoritmo de Niveles de Detalles para la Visualización de Modelos 3D*. [Fecha de consulta: 20 de febrero 2012].