# Módulo para optimización por niveles de detalles

***A module to the optimization by level of details***

**Katherine Gómez, Yanoski Camacho Román**

**Universidad de las Ciencias Informáticas**

{kgomezc, rcamacho}@uci.cu

**Resumen**

Una meta perenne-aunque no la única-, de los gráficos por computadora es producir evocaciones visuales de mundos virtuales que parezcan verdaderas.

Este trabajo aborda el diseño de una solución para optimizar la visualización a través de los niveles de detalle, garantizando que las primitivas que alimentarán los gráficos en la tubería de visualización lleven el mínimo de polígonos necesarios, y eliminando aquellos que según su distancia al punto de visión, tengan una proyección en la imagen final substancialmente menor que la resolución de píxeles de la exhibición.

Para alcanzar esta meta se trabaja en la investigación y desarrollo de algoritmos eficientes para aplicar niveles de detalles en los entornos virtuales, y se realiza una exposición de los conceptos y algoritmos para el trabajo con dichos niveles.

El módulo implementado como resultado de esta investigación, incorpora mayor calidad en la visualización de las aplicaciones finales de los proyectos de realidad virtual que hagan uso del mismo.

**Palabras clave:** entornos, gráficos por computadora, niveles de detalles, optimización, sistemas de realidad virtual, visualización.

***Abstract***

*A continuous goal not only the one_ of the computer graphics is to produce visual evokes of virtual worlds make them seem true.*

*This paper is about a solution design to optimize the visualization through the level of details, guaranteeing that the primitive aspects which feed this graphics in the visualization piping can carry the minimum polygon needed and deleting those that according to the its distance to the vision point, have a projection in the final substance image less than the exhibition pixels resolution.*

*To achieve this goal it is worked on the investigation and development of efficient algorithms to be applied in level of details on the virtual environment, and t works on an exposition of concepts and algorithms to the work of these levels.*

*The implemented module, as a result of this investigation, incorporates the highest quality in the visualization of the final applications on the real virtual projects to make them use of the same.*

***Key words**: computer graphics, environments, levels of detail, LOD, optimization, virtual reality systems, visualization.*

## Introducción

Con el surgimiento y desarrollo de las técnicas de Realidad Virtual (RV), la Informática Gráfica ha dado un salto en la calidad de sus aplicaciones en cuanto a interactividad y realismo, dando paso al surgimiento de los Sistemas de Realidad Virtual (SRV), sistemas orientados fundamentalmente al desarrollo de habilidades en la manipulación de equipos mediante una ambientación digital. La Universidad de las Ciencias Informáticas (UCI), proyecto que desde sus inicios ha estado dirigido a la formación de profesionales y a la producción de software, tiene entre sus líneas de investigación-producción el tema de la Realidad Virtual, contando para esto con varios proyectos entre los que se encuentra el de Herramientas de desarrollo para Sistemas de Realidad

Virtual, cuya misión es producir herramientas para uso de otros proyectos para el desarrollo de aplicaciones finales, haciendo así más ágil su trabajo.

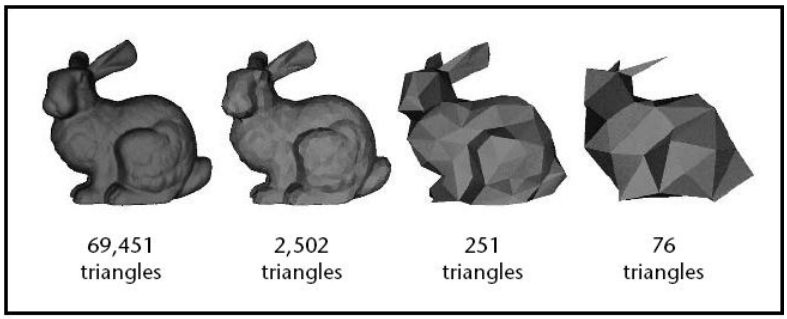
Este proyecto ya tiene una versión inicial de una herramienta básica (SceneToolKit), pero ésta no garantiza completamente la optimización de la visualización (lograr que se muestre una cantidad adecuada de fotogramas por segundo (FPS) de manera que la visualización le resulte fluida a los usuarios finales), ya que de los tres pasos esenciales del visibility pipeline o tubería de visualización (1: selección por prisma de visión, 2: selección por oclusión y 3: selección por nivel de detalle), solamente se tienen acopladas las dos primeras técnicas y la visualización es bastante fluida (se sobrepasa actualmente los 25 FPS mínimos necesarios, incluso los 60 deseados), pero es requerimiento de los proyectos finales que se incluyan determinados efectos visuales de considerable peso (luces, sombras, brillo, reflejos, fuego…), así como cálculos físico-matemáticos que deben disminuir considerablemente la calidad de la visualización, por tanto, paralelo a la implementación de estos nuevos requerimientos, se necesita incorporar otras técnicas de visualización de manera que la biblioteca esté preparada para soportar los nuevos módulos.

Por tanto se desea incorporar un módulo de optimización por niveles de detalles a dicha herramienta, con lo que se debe mejorar considerablemente la calidad de la visualización de los entornos virtuales creados, independientemente de si la carga de objetos en la escena es muy grande, o de la cantidad de procesos o cálculos paralelos que se necesiten hacer durante la visualización.

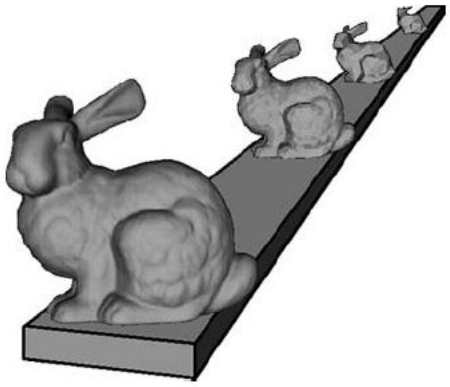
**Estudio preliminar**

Los objetos son representados en la gráfica computacional como modelos geométricos. Dichos modelos pueden ser creados a diferentes niveles de detalles (LOD), con más polígonos y mayores texturas para modelos de mayor detalle, y con menos polígonos y texturas pequeñas para modelos de menor detalle. [3].

Para implementar el rendering por niveles de detalles, se crean múltiples modelos con diferentes niveles de detalles (tradicionalmente tres: cercano, medio y lejano), y se selecciona uno de ellos cada frame en dependencia de la distancia a la cámara. Por regla general, cada nivel suele tener el doble de polígonos que el nivel precedente a medida que se acerca a la cámara. [3]. La Figura 2 representa el concepto fundamental del LOD: [2].




(a)



(b)

Fig. 1. Modelos a diferentes niveles de detalles.

a) Modelos a diferentes LOD. b) Visualización de los modelos LOD.

A continuación se hace un análisis de las técnicas de manejo de los niveles de detalles, así como de los principales algoritmos y tendencias que se están empleando en el mundo para la selección de los niveles de detalle y la deformación de las mallas.

**Técnicas de manejo de los niveles de detalles**

Las tres técnicas básicas para el manejo de los niveles de detalles son: LOD discreto, LOD continuo y LOD dependiente de la vista. [2].

El LOD discreto es el tradicional y más conocido (llamado también LOD basado en rangos), consistente en crear múltiples versiones (usualmente 3) para cada objeto durante un proceso offline o de preprocesamiento, cada una de ellas correspondiente a cada nivel de detalle, y en tiempo de ejecución se selecciona la versión adecuada. Como esta es una operación precalculada, no se puede predecir desde qué dirección se estará viendo el objeto, por lo que este tipo de LOD se conoce también como isotrópico o independiente de la vista.

El LOD continuo, conocido también como LOD progresivo, parte de la idea del LOD discreto, con la diferencia de que el nivel de detalle de cada objeto es calculado en tiempo real cuando es necesitado, en lugar de ser preprocesado. El funcionamiento real consiste en preprocesar el objeto para crear una estructura asociada al mismo que contiene el espectro de detalles, la cual será utilizada en tiempo real para extraer los parámetros necesarios para el nivel deseado y modificar el objeto. [6]

El LOD dependiente de la vista es una extensión del continuo, usándose un criterio de simplificación dependiente del punto de vista del observador, donde se selecciona dinámicamente el LOD más apropiado para la vista actual. Como es un modelo anisotrópico, un simple objeto puede abarcar múltiples niveles de simplificación; por ejemplo, las partes más cercanas del objeto se pueden representar con mejor detalle que las más lejanas (óptimo para objetos grandes y complejos como terrenos), o la silueta se puede representar con mejor resolución que las regiones interiores. [1].

**Selección del nivel de detalle**

No todos los objetos requieren optimización por niveles de detalles, por ejemplo, aquellos que permanecen a una distancia relativamente constante de la cámara, como el personaje principal en un juego, se mantienen siempre a un mismo nivel de detalle. [3]

La selección del detalle adecuado debe observar el aporte visual del objeto en la escena para cada frame, para esto, se suelen usar dos técnicas fundamentales: selección por distancia a la cámara, y selección por área en pantalla.

**Selección del detalle por distancia a la cámara**

La vía más simple de seleccionar el nivel de detalle es por la distancia a la cámara, por ejemplo, utilizando el detalle alto para modelos más cercanos que 500 unidades, el detalle medio para modelos entre 500 y 1500 unidades, y el detalle bajo para los modelos más lejanos que 1500 unidades de la cámara. Sin embargo, este método presenta algunos problemas: [3]

Primeramente, no tiene en cuenta el área que el objeto ocupa en pantalla. Si se hacen variaciones al lente de la cámara, provocará cambios en la visualización de los objetos: por ejemplo, si un objeto está lejos de la cámara pero el campo de visión o lente es muy estrecho (por ejemplo en un zoom), la imagen aparecerá grande en la pantalla y se necesitará un alto nivel de detalle; similarmente, aunque el objeto esté relativamente cerca de la cámara, si el lente es muy ancho el objeto parecerá pequeño y se necesitará menor nivel de detalle.

Comúnmente existen objetos que a pesar de estar lejos de la cámara, son demasiado grandes como para eliminarles detalles, por ejemplo, un edificio que ocupa un área importante de la pantalla; y a su vez, objetos que están cercanos pero son muy pequeños como para representarlos con mucho detalle, por ejemplo, una piedra pequeña; en estos casos tampoco es funcional ajustarles el detalle según la distancia a la cámara.

**Selección del detalle por área en pantalla**

Una mejor alternativa a la distancia a la cámara, es el criterio de área en pantalla, el cual selecciona un mayor nivel de detalle a medida que el radio aumenta, como se explica en el siguiente epígrafe.

La selección basada en el área de pantalla utiliza el objeto entero, siendo un criterio más realista de selección, pero a la vez más caro computacionalmente, por lo que se suele usar el volumen frontera del objeto para calcular su importancia visual en lugar del objeto en sí, lo cual optimiza bastante los cálculos. [2].

Muchos sistemas utilizan las cajas fronteras, proyectándolas en la pantalla y obteniendo una aproximación del área ocupada por el objeto, presentando el inconveniente de que dichas cajas dependen de la orientación del objeto. Una alternativa consiste en utilizar como volumen frontera una esfera, y calcular el radio proyectado por dicha esfera en la pantalla. Éste radio será el mismo independientemente de la orientación del objeto, pero es más pobre en cuanto al ajuste a la forma del objeto. Existe la posibilidad de usar otros volúmenes frontera como elipsoides y cajas orientadas.

**Histéresis de umbral**

Existe un problema durante la selección y aplicación de los niveles de detalles, cuando un objeto permanece cercano a las distancias límite, y está cambiando rápidamente de una profundidad a otra, por ejemplo, cuando un personaje esta corriendo en zigzag entre los niveles cercano y medio, alternando de niveles de detalles, provocando un efecto indeseable y de distracción. Este efecto se conoce como “back and forth popping”, o “salto alante y atrás”. De manera general, cuando un objeto pasa de un nivel de detalle a otro, los cambios de polígonos (sobre todo en la silueta) y texturas no deben ser muy drásticos.

Este problema puede ser resuelto con la histéresis de umbral (“hysteresis thresholding”) [4], que no es más que un retardo en la transición de niveles de detalles, en la cual los objetos pasan con mayor suavidad de un nivel a otro, con el objetivo de evitar el parpadeo (“back and forth popping”) cuando un objeto constantemente salta entre dos niveles.

Se basa en el uso de un umbral superior e inferior construyéndose una franja de transición entre los límites de profundidades, haciéndose dentro de dicha franja un efecto de transición de un LOD a otro.

**Operadores de colapsamiento**

Una malla en gráficos 3D tiene dos componentes principales: la geometría de la malla, representada por los vértices; y la conectividad de la malla, representada por las aristas y caras que conectan a los vértices. Esta conectividad codifica la topología de la malla, es decir, el número de huecos, túneles y cavidades en la malla. La simplificación de la geometría de la malla puede conllevar o no a la simplificación de la topología de la malla. [2].

**Colapsamiento de aristas**

Este operador [2] colapsa una arista (va, vb) a un solo vértice vNew, causando la eliminación de la arista (va, vb) y de los triángulos que contienen a esa arista. El operador inverso de un colapsamiento de arista es la partición de vértices, que agrega la arista (va, vb) y los triángulos adyacentes a ella. Así, el operador de colapsamiento de arista simplifica la malla y el operador partición de vértices agrega detalle a la malla.

Existen dos variantes del operador de colapsamiento de arista: el colapsamiento de media arista y el colapsamiento de arista completa. En el colapsamiento de media arista, el vértice resultante del colapsamiento es uno de sus puntos finales, es decir, vNew = va ó vNew = vb. En el colapsamiento de arista completa (más general, abreviado a menudo como simplemente colapsamiento de arista) el vértice colapsado vNew puede ser un vértice nuevamente computado.

La ventaja del operador de colapsamiento de media arista es que los vértices de la malla simplificada son un subconjunto de la malla de entrada, por lo que no se almacena ningún vértice nuevo. Finalmente, el número de triángulos modificados por un colapsamiento de media arista es más pequeño que el número modificado por un colapsamiento de arista completa. Esto puede conducir a una simplificación más eficiente puesto que pocos triángulos necesitan ser actualizados. [2].

**Grafos de escena en la SceneToolkit**

Para lograr un rendering eficiente de la escena, primeramente se necesita tener el control de los objetos que en ella se encuentran.

En las aplicaciones 3D en tiempo real se suele emplear una estructura jerárquica para la gestión de la escena denominada grafo de escena, el cual contiene información sobre la organización lógica y espacial de los objetos. El árbol que representa la escena está compuesto por una serie de nodos y se recorre en profundidad partiendo del nodo raíz hasta llegar a los nodos hojas realizando algún tipo de operación sobre estos nodos. El árbol se recurre de arriba abajo y de izquierda a derecha. [5].

En la SceneToolkit, el grafo de escena presenta nodos intermedios (que contienen información de los objetos como posición, orientación y escala, volúmenes fronteras (utilizados para la selección jerárquica de objetos visibles y la detección de intersecciones), o cualquier otro estado geométrico o de render que se desee aplicar), y nodos hojas (objetos en cuestión). Al atravesar una rama y llegar a un nodo hoja, se tiene toda la información necesaria para el rendering.

**Solución propuesta**

El sistema implementado cuenta con las siguientes características:

Se implementaron las técnicas de niveles de detalles discreto (con tres modelos), y continuo. Los modelos discretos se construyen en tiempo de preprocesamiento en un proceso semejante al del LOD continuo. No se implementó el LOD dependiente de la vista, pero existe el soporte para ser añadido con posterioridad.

No se crearon nodos específicos para el manejo del LOD en el grafo de escena existente –como suelen hacer las aplicaciones de RV normalmente–, sino que a los nodos que almacenan las geometrías se les puede atachar modificadores de la información de las geometrías que contengan dichos nodos. Básicamente, estos nodos contienen la malla en sí, y podrán tener un modificador del detalle de la malla, el cual contendrá los parámetros y datos (por ejemplo, el espectro de modelos) necesarios para esta modificación. Esto tiene como ventaja que el programador que utilice este módulo, no tendrá que hacer acciones diferentes para LOD discreto o continuo, ni crear nodos para un caso y no para el otro... sino que crea un modificador indicando el tipo de LOD, y se lo atacha al nodo que desee optimizar, y dicho manipulador internamente realiza las acciones pertinentes, quedando completamente transparente al programador.

La selección del detalle se hace por distancia a la cámara para esta versión, aunque existe el soporte para implementar fácilmente la selección por área en pantalla. Para esto, cada modelo del espectro contendrá la distancia a la cámara a la cual es aplicable, y el área en pantalla para la cual es aplicable también, aunque solamente se calculen las distancias a la cámara en esta versión.

En el caso del LOD discreto se tiene en cuenta la histéresis de umbral (hysteresis thresholding) en las zonas de cambio, con un valor por defecto del 10%. La misma se aplica actualmente para la distancia a la cámara, pero se podrá aplicar de manera semejante al área en pantalla.

No se tuvieron en cuenta otros factores de selección como las condiciones ambientales de niebla, lluvia, oscuridad o brillo, ni factores perceptuales del ojo. Dentro de los parámetros a modificar está solamente la malla, es decir que no se hace uso de impostores, morphing, o el cambio de calidad en las imágenes.

La malla se deforma con el operador local de colapsamiento de aristas (colapsamiento de media arista), conservándose la topología y género de la malla, y basado en la fidelidad (se introduce una métrica para minimizar los cambios drásticos entre modelos continuos del espectro) y el costo (se configura la menor cantidad de polígonos a la que se podrá deformar una malla). El operador de colapsamiento de aristas es uno de los más simples y de menos requerimientos de memoria y procesamiento, además de que partiendo de este se pueden implementar la mayoría de los operadores de colapsamiento restantes.

En el módulo se puede además definir la cantidad de polígonos a partir de la cual a los objetos se les asociará modificadores de detalles, esto podrá ser utilizado para hacer esta acción automáticamente por ejemplo durante la carga de los objetos de la escena, de manera que se le aplique a todos los que tengan más polígonos que los especificados. En caso de que se desee asociar un modificador de detalles a un objeto que tenga menos polígonos, se deberá especificar por parámetro que se ignore esta comparación.

Se pueden configurar otros parámetros que regulan el comportamiento de la aplicación, con los siguientes valores por defecto:

- Límite para aplicar LOD: 2000 triángulos.
- Máxima compresión que tendrán las mallas: 500 triángulos.
- La zona cercana estará entre los valores 100.0f y 500.0f.
- La zona media estará entre los valores 500.0f y 1500.0f.
- La zona lejana estará entre los valores 1500.0f y 3000.0f.
- Histéresis: 10%.

Se validará la lógica de los datos anteriores de manera que la histéresis no sobrepase el 100%, y la zona cercana tenga valores menores que la zona media, y la zona media tenga valores menores que la zona lejana. En caso de entrar datos incorrectos se establecerán los valores por defecto o se mantendrán los últimos datos correctos establecidos. Cuando un objeto esté más cerca que el límite inferior de la zona cercana, o más lejos que el límite superior de la zona lejana, se le mantendrá el detalle correspondiente a dichos límites respectivamente.

El módulo está programado en C++ estándar siguiendo la filosofía Orientada a Objeto y la nomenclatura acorde con el trabajo hecho en la SceneToolKit y el proyecto “Herramientas de Desarrollo para Sistemas de Realidad Virtual”.

**Resultados obtenidos**

Se realizaron pruebas con un modelo de 3966 polígonos, haciéndose 10 mediciones de tiempo de dibujado de un frame en milisegundos sin aplicar la técnica LOD y aplicando la misma (columnas 1 y 2 de la tabla 1); con estos valores se determinó la diferencia para cada medición (columna 3) y la ganancia (columna 4).

**Tabla 1. Cálculo de la ganancia con la aplicación de Niveles de Detalles.**

| ~LOD | LOD | Delta (~LOD-LOD) | % (Delta%~LOD) |
|---|---|---|---|
| 3.967285 | 3.906520 | 0.060765 | 0.015316 |
| 3.038574 | 2.977539 | 0.061035 | 0.020086 |
| 3.010254 | 3.019531 | -0.009277 | -0.003081 |
| 4.028320 | 3.950131 | 0.078189 | 0.019409 |
| 2.949219 | 2.391398 | 0.557821 | 0.189141 |
| 2.832013 | 2.920898 | -0.088885 | -0.031385 |
| 3.051758 | 2.991211 | 0.060547 | 0.019840 |
| 2.925781 | 2.139392 | 0.786389 | 0.268779 |
| 3.139794 | 2.934570 | 0.205224 | 0.065362 |
| 3.906250 | 3.909912 | -0.003662 | -0.000937 |
| **3.2849248** | **3.1141102** | **0.1708146** | **0.056253** |

Nótese que como promedio la aplicación del LOD ahorra 0.1708146 milisegundos por cada frame para un 5.6253% de ganancia promedio. Si este valor se multiplica para valores estándares de representación de escenas (25 FPS, 60 FPS, 85 FPS), se obtienen los siguientes resultados:

0.056253 * 25 frame/seg = 1.406325

0.056253 * 60 frame/seg = 3.37518

0.056253 * 85 frame/seg = 4.781505

Lo cual se puede interpretar como que por cada 25 frames se ahorran 1.4 frames, por cada 60 frames se ahorran 3.4 frames, y por cada 85 frames se ahorran 4.8 frames. Por ejemplo para 85 frames, se dispondrán de 4.8 frames (casi 5 ciclos de escena) para cálculos físicos o de otro tipo.

Todas estas mediciones se realizaron para LOD discreto para un nivel medio; en caso de que el modelo se posicione a un nivel lejano la ganancia será mucho mayor, esperándose que normalmente en nuestros entornos la presencia de modelos cargados esté normalmente lejos del punto de visión; a su vez, entre más cargados de polígonos estén los modelos mayor será la diferencia al aplicar esta técnica, por tanto se considera factible la aplicación de la misma.

**Conclusiones**

A partir de esa investigación se propusieron las características técnicas de la solución, obteniéndose un módulo que responde a todas las necesidades planteadas, logrando una interfaz de programación sencilla con la cual el usuario programador puede optimizar sus aplicaciones finales, sin preocuparse por los detalles internos de procesamiento.

Se implementaron, de las técnicas más usadas internacionalmente para la optimización por niveles de detalles, las que resuelven este problema con mejor velocidad de procesamiento y buena calidad visual, de esta manera se incrementan las prestaciones de la SceneToolKit, herramienta netamente de la Universidad, constituyendo un aporte más a la disminución de la dependencia de herramientas gráficas extranjeras, que habría que comprar o que probablemente no aportarían todas las prestaciones necesarias.

Con la aplicación de los niveles de detalles se eleva la calidad de la visualización y por tanto la calidad de los entornos virtuales de los simuladores y juegos, tributando al mejoramiento de estos productos y por tanto a la mayor comercialización de éstos. Esto, como política de la producción de Sistemas de Realidad Virtual de la facultad 5 de la UCI, daría un aporte social mayor en cuanto al entrenamiento y formación de habilidades de los usuarios de los productos.

Como recomendaciones para futuras versiones del módulo LOD creado se tienen las siguientes:

- Crear los modelos en un proceso offline o de preprocesamiento y no durante la carga de las escenas, y salvar los mismo a un fichero para la carga posterior.
- Incluir la técnica de manejo de niveles de detalle dependiente de la vista del observador.
- Desarrollar la selección del detalle según el área en pantalla que ocupe el objeto, así como considerar la presencia de otros factores como niebla.
- Extender la optimización del detalle a las texturas de los objetos, incorporando también el uso de impostores.

**Referencias Bibliográficas**


[1] CHOVER, M. Variable Level of Detail Strips. Italy, Springer, 2004. pp. 622-630

[2] GONZÁLEZ, Carlos L. Tutorial de VRML. Tema 14: Niveles de detalles, 2006 [Disponible en: http://wwwdi.ujaen.es/~rsegura/igai/vrml/documentos/tema14.htm]

[3] KRUS Mike, BOURDOT Patrick, and THIBAULT Guillaume. Niveles de Detalle y Simplificación Poligonal, 2007. [Disponible en: http://acm.org/crossroads/espanol/xrds3-4/levdet.html]

[4] LUEBKE, D., REDDY, M., COHEN, J., VARSHNEY, A., WATSON, B., and HUEBNER, R. Level of Detail for 3D Graphics, 2002. [Disponible en: http://lodbook.com/]

[5] PELECHANO GÓMEZ, N. Estudio y desarrollo de un Sistema de Simulación de Deformaciones Tridimensionales en Figuras Flexibles, orientado a su uso en la Visualización Interactiva de Personajes Virtuales. Universidad de Valencia. Valencia, 2001. 98 pp.

[6] ULRICH, Thatcher. Continuous LOD Terrain Meshing Using Adaptive Quadtrees, 2007 [Disponible en: http://www.gamasutra.com/features/20000228/ulrich_01.htm]