SOLUCIONES INFORMÁTICAS PARA LA VISUALIZACIÓN TRIDIMENSIONAL EN TIEMPO REAL

Autor: MSC. Jose Ignacio Guzmán Montoto

Dirección: 16% 11 y 13 casa 105 Apto 9. Plaza de la Revolución.

Ciudad de la Habana.

Teléfono: 8312974.

Institución: SIMPRO. Centro de Investigación y Desarrollo de Simuladores.

Resumen

Los avances alcanzados en el desarrollo del hardware y en los métodos de adquisición de datos como scanners tomográficos y sistemas de análisis de imágenes, han llevado a la obtención de modelos geométricos de elementos biomédicos con la propiedad de ser manipulados a través de la visualización tridimensional (3D). Hoy día, esta visualización abarca desde aplicaciones biológicas, incluyendo análisis de estructuras y sus relaciones funcionales, hasta aplicaciones médicas que comprenden exactitudes anatómicas y la planificación o el entrenamiento para operaciones quirúrgicas complejas.

En este trabajo se proponen soluciones informáticas para satisfacer requerimientos de visualización en tiempo real. Los algoritmos desarrollados se encuentran agrupados en una biblioteca gráfica que facilitará el desarrollo de futuros trabajos. Los resultados obtenidos permiten enfrentar problemas actuales de representación tridimensional de superficies complejas, se alcanza realismo en las imágenes y tienen posible aplicación en bioinformática y medicina.

Palabras clave: *Visualización tridimensional, Realidad Virtual, Optimización 3D.*

Abstract

The advances reached in the development of the hardware and in the methods of acquisition of data like tomographic scanners and systems of analysis of images, have allowed obtaining geometric models of biomedical elements with the property of being manipulated through the three-dimensional visualization (3D). Nowadays, this visualization embraces from biological applications, including analysis of structures and its functional relationships, until medical applications that include anatomical accuracies and the planning or the training for complex surgical operations.

This work proposes computer solutions to satisfy visualization requirements in real time. The developed algorithms are contained in a graphic library that will facilitate the development of future works. The obtained results allow facing current problems of three-dimensional representation of complex surfaces, realism is reached in the images and they have possible application in bioinformatics and medicine.

Keywords: *Three-dimensional visualization, Virtual Reality, 3D Optimization.*

.

Introducción

Con el advenimiento de nuevas técnicas para la adquisición de datos biomédicos, y el incremento de las capacidades de las computadoras para el procesamiento y la reconstrucción de esa información, se presentan nuevas oportunidades para el diagnóstico y el tratamiento médico, así como para las investigaciones biológicas. De manera especial, la visualización en tiempo real, abre nuevas posibilidades al permitir interactuar con inmediatez, sobre modelos de suficiente detalle [1].

Generalmente los datos de partida conforman modelos geométricos considerablemente complejos y se necesitan para su procesamiento, análisis y visualización, de la computación de altas prestaciones. Aplicaciones muy útiles han sido implementadas sobre estaciones de trabajo con hardware diseñado para la aceleración gráfica.

Dado el alto costo del hardware y software involucrado en la visualización de tiempo real existe la necesidad de soportar el proceso sobre computadoras personales (PC). El empleo de estas computadoras constituye un reto a la ciencia y precisa de la inclusión de un hardware reforzado [2] y de algoritmos optimizados para lograr las prestaciones requeridas.

Es por tanto objetivo de este trabajo la obtención de soluciones para la visualización en tiempo real utilizando computadoras personales. De manera específica se pretenden desarrollar algoritmos optimizados sobre PC, evaluar

la visualización obtenida y explorar otras posibilidades de aumentar las capacidades de estos equipos.

En este artículo se exponen los resultados obtenidos en el campo de la visualización 3D utilizando la biblioteca gráfica OpenGL. Se muestran algoritmos desarrollados para la visualización de superficies, que aprovechando al máximo el rendimiento de la tarjeta gráfica logran una representación en tiempo real aún incluyendo un modelo de iluminación.

Como resultado del trabajo se desarrolló un programa de visualización que aplica los algoritmos planteados y permite representar superficies complejas en tiempo real.

Optimización en el uso de OpenGL sobre PC.

Hoy día se hace imprescindible el empleo de una biblioteca gráfica (API) como OpenGL o DirectX, para el desarrollo de aplicaciones de visualización en tiempo real. Esta API es soportada por las tarjetas aceleradoras que se comercializan y con esta combinación se pueden alcanzar resultados muy convincentes.

Podría pensarse que con el uso de una tarjeta aceleradora gráfica todo esta resuelto, sin embargo, si no se aplican algoritmos de optimización, muy fácilmente se sobrepasan las capacidades gráficas y resulta, erróneamente, insuficiente el hardware.

Por otra parte OpenGL, biblioteca que se utilizó en este trabajo, contiene soluciones para el dibujo de primitivas básicas, pero no tiene funciones de

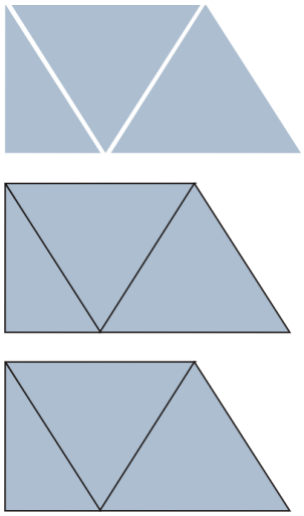
importar modelos obtenidos desde editores gráficos u otras fuentes, tampoco presenta facilidades específicas de observación del mundo o algoritmos de optimización por citar ejemplos. Estos aspectos adicionales se incluyen en bibliotecas a más alto nivel.

Un ejemplo de estas bibliotecas lo constituye el conjunto de clases que soporta este trabajo [3]. A este soporte se le añadió un algoritmo de optimización (Octree) recomendado para casos de aceleración por hardware, para lo cual se realizaron adaptaciones en las estructuras de datos utilizadas.

Método del Particionamiento (Octree)

El objetivo de este algoritmo es reducir la cantidad de triángulos que se van a dibujar a partir de una división del terreno en varias regiones [4]. Cada región es asociada con una lista de OpenGL (glList) la cual es determinada en una etapa de precalculo. Llegado el momento, se dibujan solo aquellas regiones que se encuentran en la zona de visibilidad.

Con esta técnica se minimiza la carga del microprocesador del PC, ya que no se utilizan las sucesivas llamadas a funciones para la definición de los datos de los vértices del tipo glvertex(). La figura 1 muestra la ventaja de evitar la carga que producen sobre el procesador las diferentes formas de invocar el dibujo de triángulos [5, 6].



Triángulos independientes

(xyzw, rgba, uvs, xyz) * 9 vértices = 36 llamadas

Arreglos de vértices

5 llamadas a funciones

Listas

Solo una llamada a función.

Figura 1. Llamadas a funciones que provocan 3 métodos de dibujo de grupos de triángulos.

El método se basa en la formación de manera recursiva de un árbol de 8 nodos. El límite de crecimiento del árbol es hasta el nivel 5 y el máximo de polígonos definido para cada nodo es 1000. Cada triángulo en este procesamiento previo, se ubica en el nodo que le corresponde según las coordenadas de sus vértices.

Para aumentar la velocidad de representación se utilizaron las listas de OpenGL creadas de una manera igualmente recursiva. Esta recursividad va por todos los nodos finales asignándoles una identificación de Lista. De esta forma se llama a esta identificación cuando se dibuja, incrementando considerablemente la velocidad en cuadros por segundo.

Corte de Volumen (Frustum culling)

El volumen de visión se define por seis planos que forman una pirámide truncada [7, 8, 9]. Cuatro planos constituyen los laterales de la pirámide y los restantes constituyen los límites de visión hacia atrás y hacia delante. Lo que no se observa en la pantalla está fuera de este campo de visión y por tanto resulta ineficiente enviarlo a la tarjeta gráfica.

Es por esta razón que en el momento de dibujar el modelo se hace necesario verificar cada uno de estos nodos contra el volumen de visión para determinar si debe dibujarse o no. Si un nodo no se encuentra no habrá que chequear sus hijos y se pasa al próximo.

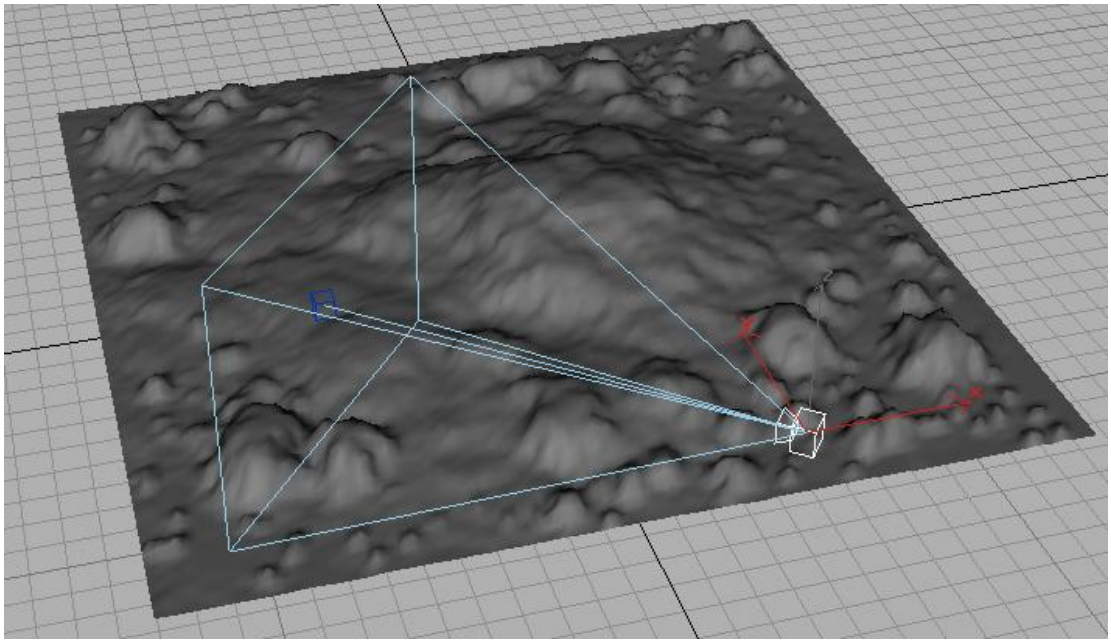


Figura 2. Representación del cono de visión.

El siguiente fragmento de código muestra la secuencia necesaria para el dibujo de la escena por métodos optimizados. Se aprecia como el cono de visión es calculado para cada cuadro y la llamada al dibujo del árbol. Este último se dibuja recursivamente y comprueba los nodos contra el cono de visión.

```
void RenderScene()
{
	glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);
	glLoadIdentity();
	Camera.Actualize();

	// En cada cuadro se verifica si hay movimiento de la cámara y
	//de ocurrir se calcula el nuevo volumen de visión.
	MyFrustum.Calculate();

	//Dibujo del Octree
	MyOctree.Draw(&MyOctree, &MyWorld);
	..........
}
```

Ensayos realizados

Las optimizaciones implementadas se comprobaron en dos configuraciones de hardware con las siguientes características:

1. PC Pentium III a 550 Mhz. 128 Mb de RAM. Tarjeta aceleradora gráfica AccelStar II.

2. PC Pentium IV a 1.4 GHz. 256 Mb de RAM. Tarjeta aceleradora gráfica GeForce IV.

Para las pruebas de rendimiento del algoritmo se desarrolló un programa de visualización que, a partir de implementaciones de OpenGL, aplica o no la optimización. Se diseñó la presentación de la información de cantidad de polígonos y cuadros por segundo (fps). Además se construyó un modelo de superficie con resoluciones desde 10 000 hasta 60 000 triángulos. Los resultados obtenidos para la primera configuración se muestran en la figura 3.

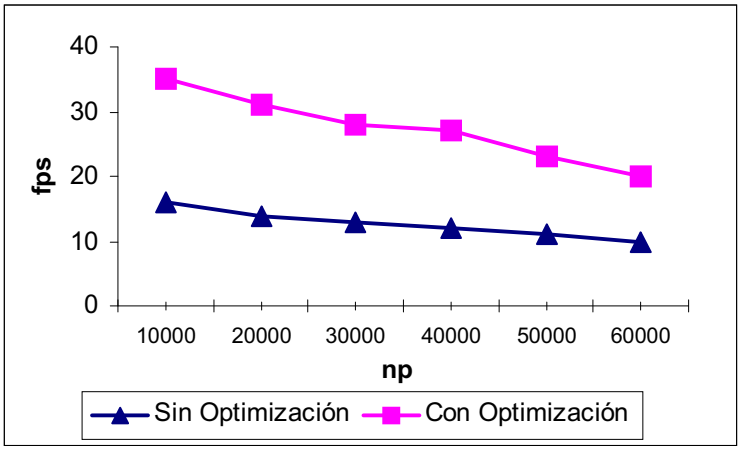


Figura 3. Comportamiento de velocidad contra número de triángulos para la configuración 1

Analizando los resultados para la primera configuración apreciamos que sin optimizaciones, desde las pruebas con menor carga, no hay una respuesta eficiente. Esto se explica por ser el número de triángulos inicial superior a la capacidad de respuesta del sistema gráfico. Se trata de una configuración típica de 2 años atrás, que pone de manifiesto sus limitaciones.

Al aplicar la optimización se aprecia en este mismo caso una mejoría considerable en la velocidad de refrescamiento del sistema gráfico. Esto se explica por el papel que desempeñan las técnicas de optimización aplicadas en este trabajo. Recuérdese que se está controlando en este caso la cantidad de triángulos a dibujar, así incluso para las mayores cargas (60 000 triángulos) solo se dibujan los que están en el cono de visión.

Esta primera prueba demuestra como, aplicando optimizaciones, se puede obtener de una configuración no tan potente, buenos resultados.

Este ensayo se repitió para la configuración número 2 y los resultados se muestran en la figura 4.

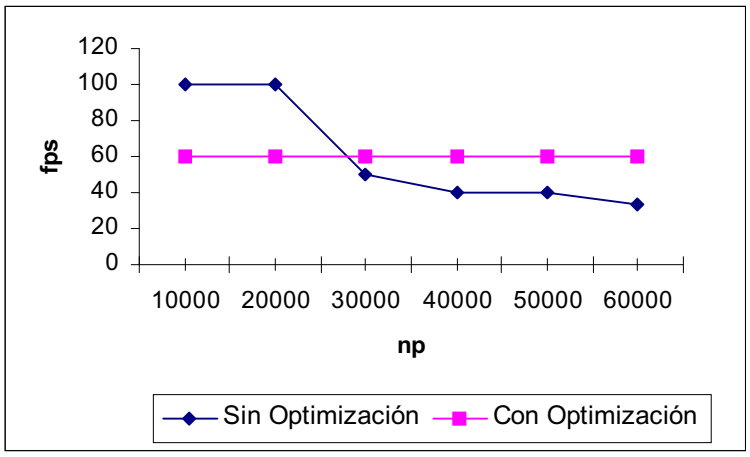


Figura 4. Comportamiento de velocidad contra número de triángulos para la configuración 2.

En este caso aparece un alto rendimiento inicial sin optimización (100 fps) que se explica por la alta capacidad de la configuración. Esta tarjeta gráfica es de las últimas del mercado y se caracteriza por altas prestaciones. La carga inicial, hasta 20 000 triángulos, es soportada sin dificultad. Al aumentar la carga sin embargo, se aprecia la reducción de la velocidad y aún así utilizando la fuerza bruta (sin optimizar), esta velocidad es de 33 fps 60000 triángulos.

Al aplicar el algoritmo de optimización se aprecia una constante de 60 fps durante toda la prueba, lo que indica un rendimiento excelente para cargas altas. Para las cargas mínimas se nota sin embargo, una disminución con respecto al uso de la fuerza bruta. La explicación de este fenómeno es que para optimizar se realizan cálculos adicionales que influyen en el ciclo de dibujo, afectando su velocidad. El uso de estos cálculos se ve recompensado en la representación de mallas de alta densidad de polígonos como muestra la gráfica.

Conclusiones

En este trabajo se han mostrado soluciones para la representación de superficies tridimensionales utilizando computadoras personales. Se ha creado un soporte para la implementación de las funciones de OpenGL de manera eficiente y se han introducido algoritmos que aumentan las capacidades de representación de estas computadoras.

Los algoritmos aplicados han sido probados y se encuentran en proceso de perfeccionamiento para obtener mejores rendimientos. Los resultados expuestos en este trabajo muestran la eficiencia de la optimización y cuanto lo necesita la PC a pesar de contar con aceleración por hardware.

Las funciones y algoritmos implementadas son capaces de representar superficies complejas y por tanto de adecuarse a aplicaciones biomédicas. El empleo de estas técnicas en esta esfera puede contribuir de manera significativa a la obtención de sistemas de visualización eficientes sobre PC conduciendo al avance científico y a la reducción de costos.

Referencias bibliográficas

1. R.A Robb,.: Three-Dimensional Visualization in Medicine and Biology. En el Libro: Handbook of Medical Imaging: Processing and Analysis, ed. Isaac N. Bankman, Academic Press, San Diego, CA, Capítulo 42, pp. 685-712, 2000.
2. B.Grigore, C.Philippe. Virtual Reality Technology. John Wiley & Sons, Inc. 1994.
3. J. I. Guzmán, J. Becquet. Biblioteca de clases para sistemas de realidad virtual. Informática 2002. 2002.
4. J. Blow. Terrain Rendering Research for Games. Slides for Siggraph 2000 Course 39, 2000.
5. K.Cok Developing Efficient Graphics Software: The Yin and Yang of Graphics. SIGGRAPH 2000 Course.
6. R.S.Wright, M.Sweet. OPENGL SuperBible. Waite Group Press. 1999
7. J. D.Foley, A.van Dam, S. K.Feiner, Hughes J. F. Computer Graphics Principles and Practice. Addison – Wesley Publishing Company. Second Edition. 1990.
8. R. Fosner. OpenGL Programming for Windows 95 and Windows NT. Addison – Wesley Developers Press. 1996.
9. M.Woo, J.Neider, T.Davis and D.Shreiner. OpenGL® Programming Guide: The Official Guide to Learning OpenGL, Version 1.2. Waite Group Press. Second Edition. 1999.