Serie Científica de la Universidad de las Ciencias Informáticas
Vol. 9, No. 3, Marzo, 2016
ISSN: 2306-2495 | RNPS: 2343
http://publicaciones.uci.cu
Pág. 13-27

Tipo de artículo: Artículo original
Temática: Realidad virtual
Recibido: 28/02/2016 | Aceptado: 28/03/2016

Visualización avanzada de volúmenes empleando hardware gráfico

Advanced volume visualization using graphics hardware

Luis Guillermo Silva Rojas [1*], Rubén Alcolea Núñez [1], Alina D. Rodríguez Peña[1]

[1] Universidad de las Ciencias Informáticas, Carretera a San Antonio de los Baños, km 2 ½, Boyeros, La Habana, Cuba. lgsilva, ralcolea, alina@ uci.cu

[*] Autor para correspondencia: lgsilva@uci.cu

Resumen

El hardware gráfico actual ofrece un amplio espectro de aplicaciones a desarrollar, desde la representación de millones de primitivas geométricas hasta la realización de operaciones computacionalmente costosas como cálculos numéricos, compresión de datos y visualización de volúmenes. El algoritmo *Raycasting* basado en GPU se ha convertido en el estándar para la visualización de volúmenes de forma interactiva y con alta calidad. El algoritmo original puede modificarse o extenderse para aplicaciones específicas, obteniendo buenos resultados, siempre que se empleen eficientemente las potencialidades de las tarjetas gráficas. En este artículo se presenta una implementación del algoritmo *Raycasting* basado en GPU con el objetivo de realizar una representación avanzada de datos volumétricos. Se empleó la estructura de datos *octree* para saltar los espacios en blanco y se adicionó so-porte para incluir geometría poligonal dentro de la representación volumétrica. Se adicionó la navegación libre por el interior del volumen en forma de endoscopia virtual. Para mejorar la calidad de la visualización se utilizó el algoritmo Refinamiento de los puntos de intersección. Se propone un editor de funciones de transferencia unidimensionales y otro para las bidimensionales basadas en gradiente, además se incluyó el modelo de Phong para iluminación local y el algoritmo Rayos de sombra para iluminación global. Con las modificaciones propuestas al algoritmo *Raycasting*, se obtienen representaciones volumétricas con alta calidad visual y sin perder el tiempo real de la visualización.

Palabras clave: GPU, *Raycasting*, visualización, volumen

Abstract

The current graphics hardware offers a wide range of applications to be developed, since the representation of millions of geometric primitives to performing computationally expensive operations such as numerical computation,

Serie Científica de la Universidad de las Ciencias Informáticas
Vol. 9, No. 3, Marzo, 2016
ISSN: 2306-2495 | RNPS: 2343
http://publicaciones.uci.cu
Pág. 13-27

data compression and volume visualization. The GPU-based Raycasting algorithm has become the standard for interactive high quality volume visualization. The original algorithm can be modified or extended for specific applications, obtaining good results, if the potentialities of graphic cards are used efficiently. This paper presents an implementation of GPU-based Raycasting algorithm in order to perform advanced representations of volumetric data. We used the octree data structure for empty space skipping and added support to include polygonal geometry within the volumetric representation. The proposed algorithm allows the user to move the viewpoint into the dataset using fly-through mode like virtual endoscopy applications. The Hitpoint Refinement algorithm was added to improve the display quality. We propose two transfer function editors, the first one for one-dimensional transfer functions and the second for two-dimensional gradient based transfer functions, the algorithm also includes the Phong model for local illumination and Shadow-rays for global illumination. With the proposed changes to the Raycasting algorithm, can be obtained volumetric representations with high visual quality without losing real-time performance.

Keywords: *GPU, Raycasting, visualization, volume.*

Introducción

En el campo de la visualización de volúmenes asistida por hardware gráfico, se destacan dos aproximaciones que alcanzan representaciones interactivas de conjuntos de datos volumétricos. La primera aproximación, presentada originalmente por (Cullip and Neumann, 1993), emplea directamente las potencialidades del hardware gráfico para el mapeo de texturas, creando superficies de muestreo (geometría intermedia) usualmente planas y alineadas al vector de visión para el caso de una textura 3D o alineadas a los ejes de coordenadas para el caso de texturas 2D. Esta técnica es ampliamente usada para la representación interactiva de volúmenes de mediano tamaño.

Esta aproximación presenta una serie de desventajas que limitan su eficiencia y versatilidad: primero, todo lo que se necesita calcular para el resultado final, cada acceso a textura, cálculo de gradiente o iluminación, debe realizarse para cada fragmento, sin tener en cuenta si contribuye o no a la imagen final. Segundo, se torna compleja la implementación de técnicas avanzadas como Saltar los espacios en blanco (*Empty space skipping*), debido a la inflexible naturaleza del algoritmo. Y finalmente, la reducción de artefactos de muestreo para la proyección en perspectiva, impone dificultades en la mayoría de los casos (Scharsach, 2006).

La segunda aproximación consiste en implementar el algoritmo *Raycasting* en el *fragment shader* de la GPU[1], como se propone en (Krüger and Westermann, 2003). Esta variante, mucho más flexible, puede extenderse de diferentes

[1] Del inglés: *Graphics processing unit.*

Serie Científica de la Universidad de las Ciencias Informáticas
Vol. 9, No. 3, Marzo, 2016
ISSN: 2306-2495 | RNPS: 2343
http://publicaciones.uci.cu
Pág. 13-27

maneras y posibilita emplear directamente las ventajas que ofrece la GPU con respecto a la CPU[2] , entre las que se encuentran:

- Arquitectura masivamente paralela.
- Separación en dos unidades de procesamiento (*vertex shader* y *fragment shader*) que puede duplicar el rendimiento si el flujo de trabajo se puede dividir.
- Instrucciones dedicadas a tareas gráficas.
- Operaciones con vectores de cuatro elementos tan rápidas como operaciones escalares.
- Interpolación trilineal automática para texturas 3D (implementada en hardware).

Teniendo en cuenta estas ventajas y otras que aparecen en dependencia de la naturaleza específica del algoritmo, se selecciona la segunda variante de implementación para el desarrollo del presente trabajo. En las siguientes secciones se introducen los conceptos relacionados con el algoritmo *Raycasting* basado en GPU, las extensiones realizadas para mejorar el rendimiento y la calidad de la imagen, los resultados alcanzados y los trabajos futuros.

Materiales y métodos o Metodología computacional

Raycasting basado en GPU

Para implementar el algoritmo *Raycasting* basado en GPU, como se propone en (Krüger and Westermann, 2003), es necesario almacenar el volumen de datos en una textura 3D, con el objetivo de emplear la interpolación trilineal implementada en hardware; luego, se representa una geometría envolvente en forma de cubo, donde el color de cada vértice coincide con su posición espacial. De esta forma, la representación de las caras delanteras y las caras traseras, codifica la posición de entrada y salida del volumen respectivamente, ver Figura 1.

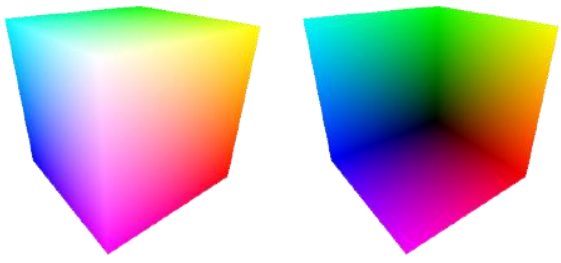

Figura 1. Representación de las caras delanteras y traseras de la geometría envolvente

[2] Del inglés: *Central processing unit*

Serie Científica de la Universidad de las Ciencias Informáticas
Vol. 9, No. 3, Marzo, 2016
ISSN: 2306-2495 | RNPS: 2343
http://publicaciones.uci.cu
Pág. 13-27

En este punto, se calculan los vectores de visión para cada píxel, restando el color de las caras delanteras con el de las caras traseras pertenecientes al píxel actual. El resultado se normaliza y guarda en una textura bidimensional que representa los vectores de visión para cada píxel de la imagen a representar, ver Figura 2.

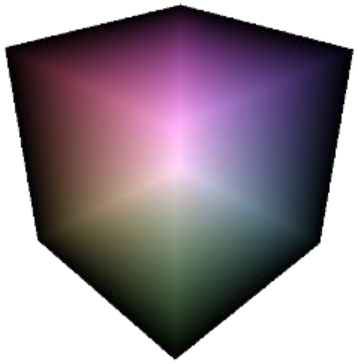

Figura 2. Textura direccional, cada píxel representa el vector de visión para el fragmento actual

Luego, se representan las caras delanteras nuevamente y se avanza en la dirección del vector de visión para el píxel actual, hasta que el rayo abandone el volumen; en cada paso, se evalúa el color de la textura 3D y se mezcla con los valores obtenido a lo largo del rayo, finalmente, el color resultante se asigna al fragmento actual.

El *fragment shader* ejecutado en la GPU provee soporte natural para la implementación del *Raycasting*, al representar las caras delanteras como una geometría normal de OpenGL, se llama automáticamente al *fragment shader* por cada píxel perteneciente a las caras frontales.

Básicamente el algoritmo *Raycasting* basado en GPU es un proceso de cuatro pasos (Scharsach, 2006):

1. Generación de las caras delanteras: Se representan las caras delanteras de la caja envolvente del volumen y se almacena en una textura.
2. Generación de la textura direccional: Se representan las caras traseras de la caja envolvente del volumen, se guarda en otra textura y se le sustrae a la textura generada en el paso 1. Los vectores direccionales resultantes son almacenados en otra textura (Figura 2).
3. *Raycasting*: Se toma la posición inicial de la textura generada en el paso 1 y a lo largo del vector direccional se toman muestras mientras el rayo no se salga del volumen y cumpliendo que la opacidad acumulada sea menor que 1 (terminación temprana del rayo).
4. El resultado del *Raycasting* se presenta en la pantalla.

Serie Científica de la Universidad de las Ciencias Informáticas
Vol. 9, No. 3, Marzo, 2016
ISSN: 2306-2495 | RNPS: 2343
http://publicaciones.uci.cu
Pág. 13-27

Saltando los espacios vacíos

El algoritmo inicial, presentado en el subepígrafe *Raycasting* basado en GPU, permite mejoras y extensiones de diferentes tipos como por ejemplo saltar los espacios en blanco, la cual acelera considerablemente su tiempo de ejecución (Scharsach, 2006) (Zhang et al, 2011). En esta sección se propone el empleo de un *octree* como estructura de datos para la generación de la geometría limitante, con el objetivo de evadir la representación de las zonas no significativas del volumen (Rodríguez, 2013).

Siguiendo el algoritmo tradicional, el procesador de vértices (*vertex processor*) se encuentra parcialmente inactivo, debido a que sólo se representan 12 triángulos para la geometría limitante, mientras que el procesador de fragmentos (*fragment processor*) está sobrecargado con la representación de todo el volumen.

La arquitectura actual de las tarjetas gráficas está diseñada para recibir cantidades masivas de geometría poligonal sin impactar considerablemente en el rendimiento de la visualización. Cuando las tarjetas gráficas modernas representan una escena, intentan procesar los vértices y los píxeles en paralelo, el mejor de los casos equilibra la carga de trabajo entre el procesador de vértices y de fragmentos (Scharsach, 2006). Por tanto, la idea consiste en aumentar la complejidad de la geometría limitante haciéndola dependiente del volumen de datos a visualizar.

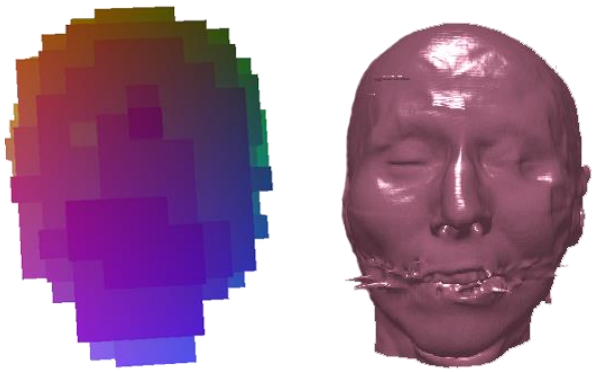

Figura 3. Representación de la geometría limitante empleando un *octree*. Solo se representan los bloques significativos

En el presente trabajo se empleó un *octree* donde cada bloque representa una porción del volumen original, ver Figura 3. Para la visualización sólo se tienen en cuenta los bloques que superan el umbral (*isovalue*) o son de interés según la función de transferencia actual.

Con esta modificación al algoritmo, se saltan los espacios en blanco que rodean el volumen y los que se encuentran en su interior, además, se simplifica el proceso de representación de la geometría limitante que se propone en (Scharsach, 2006), específicamente en la sección *Empty space skipping*.

Serie Científica de la Universidad de las Ciencias Informáticas
Vol. 9, No. 3, Marzo, 2016
ISSN: 2306-2495 | RNPS: 2343
http://publicaciones.uci.cu
Pág. 13-27

Navegación dentro del volumen

Para muchas aplicaciones resulta interesante mover el punto de visión dentro del volumen y explorarlo en modo de vuelo libre, como por ejemplo las endoscopias virtuales y los simuladores quirúrgicos (Preim and Botha, 2013).

Mientras la cámara virtual no intersecte la geometría limitante representada, no se observa ningún problema. Pero en cuanto el plano de corte cercano de la cámara (*near clipping plane*) intersecta la geometría limitante, se generan huecos en la representación de las caras delanteras, trayendo como resultado posiciones iniciales incorrectas para la creación de los rayos. La Figura 4 ilustra este comportamiento.

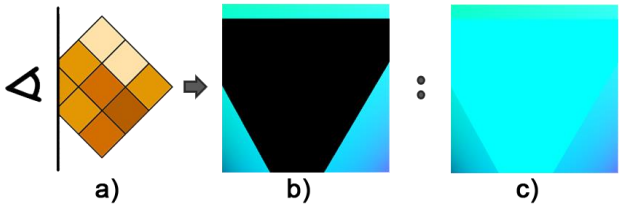

Figura 4. a) Intersección con las caras delanteras, b) huecos resultantes, c) corrección de la representación.

Los huecos que se generan deben ser rellenados con el color correspondiente a la posición donde ocurre la intersección con el plano cercano de la cámara en el píxel actual, ver Figura 4 c).

Cuando el valor correspondiente a la textura que representa las caras delanteras coincide con el color de fondo, se realiza una proyección inversa para obtener la posición en coordenadas de volumen que ocupa el plano cercano de la cámara. Para esto es necesario conocer las coordenadas (X, Y) del píxel o fragmento actual, luego se transforman las coordenadas de pantalla obtenidas a coordenadas espaciales, multiplicando la posición resultante del paso anterior por las inversas de las matrices de modelado y proyección. Esta posición se toma como el color de las caras delanteras.

Con esta modificación al algoritmo *Raycasting*, se obtienen posiciones iniciales válidas para todos los rayos y se permite la exploración desde cualquier parte, interior o exterior, del volumen de datos.

Refinamiento de los puntos de intersección

El principal problema que presenta cualquier algoritmo basado en *Raycasting* es la aparición de artefactos en forma de anillos debido a la baja frecuencia para la toma de muestras (Scharsach, 2006), estos artefactos son visibles en la Figura 6 a). *Raycasting* es solo una aproximación al cálculo real de la integral a lo largo del rayo, por tanto, siempre existe al menos una pequeña diferencia entre el valor del cálculo real y el aproximado de forma discreta. Las diferencias disminuyen cuando se toma una distancia entre las muestras (delta) muy pequeña, trayendo consigo un impacto negativo en el rendimiento del sistema.

Serie Científica de la Universidad de las Ciencias Informáticas
Vol. 9, No. 3, Marzo, 2016
ISSN: 2306-2495 | RNPS: 2343
http://publicaciones.uci.cu
Pág. 13-27

Una posible solución puede ser el refinamiento de los puntos de intersección (*Hitpoint Refinement*), que se basa en mantener un paso largo y constante durante las zonas del volumen que no precisan de gran nivel de detalle. En cambio, donde el rayo intersecta objetos de interés, se disminuye el paso para tomar un número mayor de muestras (Scharsach, 2006).

Cuando se detecta la primera intersección, el algoritmo retrocede la mitad del paso actual en dirección contraria, si aún se encuentra dentro de una zona de interés, el próximo paso lo ejecuta hacia atrás igualmente, siempre a una distancia de la mitad del paso actual. Si por el contrario, se encuentra fuera de una zona de interés, se desplaza hacia adelante de la misma forma, siempre tomando la mitad de la distancia anterior para su avance, la Figura 5 ilustra gráficamente este procedimiento.

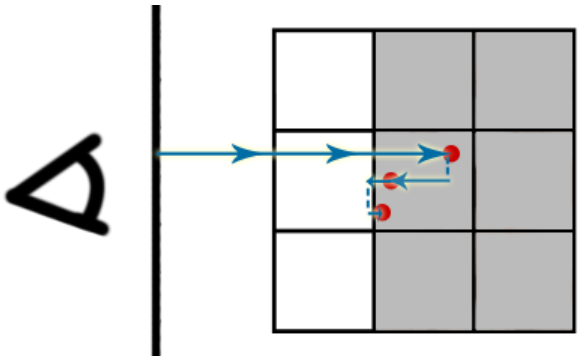

Figura 5. Algoritmo Refinamiento de los puntos de intersección

La Figura 6 b), muestra los resultados obtenidos refinando los puntos de intersección con 6 iteraciones. Se puede observar que los artefactos presentes en la Figura 6 a) han sido eliminados y su calidad visual es superior.

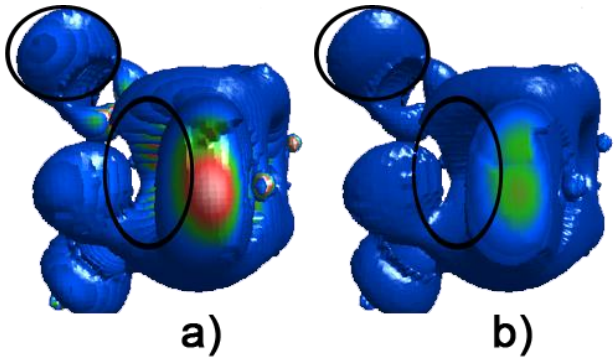

Figura 6. Resultados del algoritmo Refinamiento de los puntos de intersección

Serie Científica de la Universidad de las Ciencias Informáticas
Vol. 9, No. 3, Marzo, 2016
ISSN: 2306-2495 | RNPS: 2343
http://publicaciones.uci.cu
Pág. 13-27

Intersección con geometría

La combinación de las técnicas de visualización de volumen con las técnicas tradicionales para la representación de los objetos (mallas triangulares, NURBS, etc.) amplía en gran medida la variedad de aplicaciones posibles a desarrollar mediante el uso del *Raycasting*. Para las aplicaciones de endoscopias virtuales, por ejemplo, resulta necesario combinar la representación volumétrica con el endoscopio y las demás herramientas que se emplean para estos procedimientos. Para aplicaciones más avanzadas, como simuladores quirúrgicos, se deben representar, además: las pinzas de corte, cauterizadores, agujas e hilo para suturas, etc.

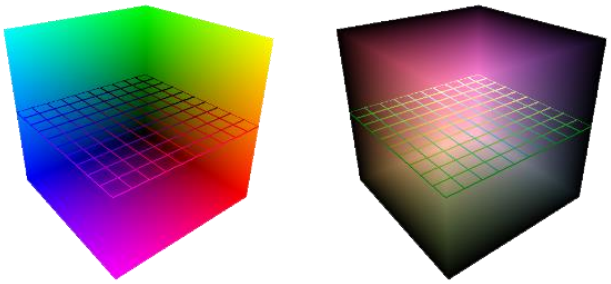

Figura 7. Representación de las caras traseras y la textura direccional adicionando en ambos casos una red (*grid)* de orientación

Como se observa en la Figura 7, cuando ocurra una intersección entre el volumen y un objeto formado por geometría poligonal, la representación de las caras traseras debe modificarse. Este comportamiento se consigue adicionando los nuevos objetos a la representación de las caras traseras de forma que, si el objeto se encuentra más cerca del punto de vista, este es el que se representa, con un ligero cambio: los vértices tendrán como color, la posición espacial que ocupan dentro del volumen, de la misma forma que se realiza la representación de la geometría limitante (Scharsach, 2006).

Los resultados de esta modificación al algoritmo se aprecian en la Figura 8. Aunque el objetivo del trabajo no persigue en esta etapa establecer una comparación competitiva con resultados alcanzados por los grupos de avanzada en la disciplina, si es posible tener en alguna medida la magnitud del progreso logrado y el grado de terminación del mismo para lograr la identificación de las posibles soluciones posteriores. Con la adición de la nueva geometría en forma de red se obtiene una percepción aceptable de la profundidad y del sentido de la distancia.

Serie Científica de la Universidad de las Ciencias Informáticas
Vol. 9, No. 3, Marzo, 2016
ISSN: 2306-2495 | RNPS: 2343
http://publicaciones.uci.cu
Pág. 13-27

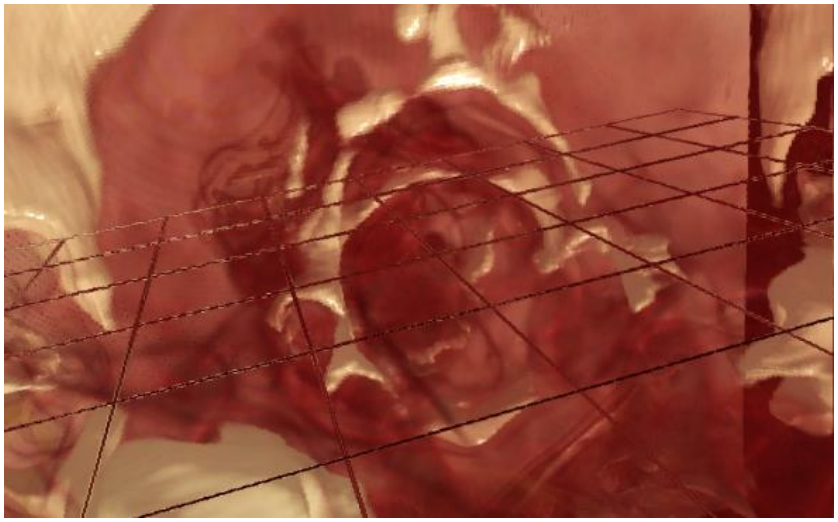

Figura 8. Endoscopia virtual con una red para mejorar la percepción de profundidad

Funciones de transferencia

Una característica importante de la visualización directa de volumen, especialmente en imágenes médicas, es la habilidad de distinguir las diferentes estructuras en el volumen. Este proceso se lleva a cabo por la función de transferencia, la cual asigna propiedades ópticas, como color y opacidad, a los datos extraídos o calculados del volumen (Engel et al., 2006), (Gravrilescu, 2011).

Mientras la transformación de valores del volumen a propiedades ópticas es un proceso sencillo, la creación de una buena función de transferencia es una tarea compleja, por esa razón, el usuario interactúa con un editor de funciones de transferencia.

Para el presente trabajo se implementaron dos editores de funciones de transferencia, el primero emplea el dominio 1D (ver Figura 9), representando valores escalares del volumen. En este caso, el eje horizontal representa los valores en escala de grises en el rango de 0 a 255 y el eje vertical la opacidad de 0 a 1. El editor brinda al usuario una serie de puntos de control que definen la función opacidad. La parte inferior permite seleccionar por cada valor del eje horizontal el color deseado, que unido a la opacidad A, obtenida de la altura desde la base hasta la función opacidad, forma un color RGBA. Estos colores se interpolan linealmente y se envían al *fragment shader* como una textura 1D, donde son usados en el tercer paso del algoritmo *Raycasting* (Silva et al, 2016).

Serie Científica de la Universidad de las Ciencias Informáticas
Vol. 9, No. 3, Marzo, 2016
ISSN: 2306-2495 | RNPS: 2343
http://publicaciones.uci.cu
Pág. 13-27

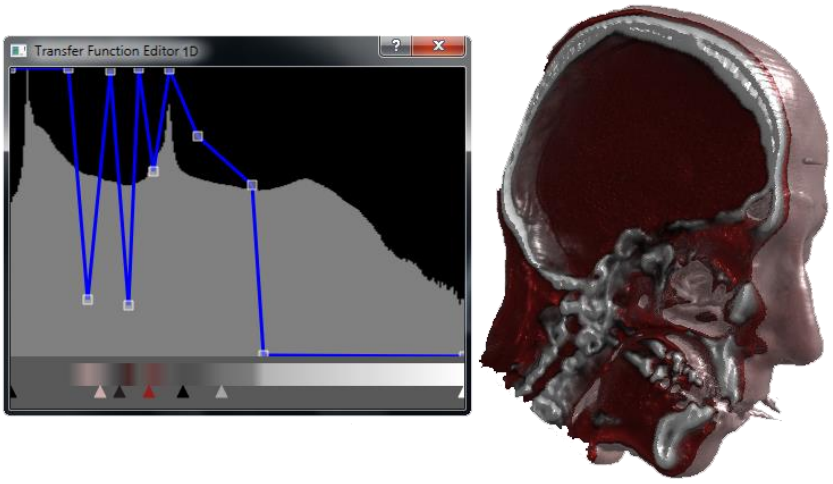

Figura 9. Editor de funciones de transferencia unidimensionales y resultado de su aplicación en la visualización

El segundo editor (ver Figura 1) emplea un dominio 2D, donde los valores de las muestras y el vector gradiente son los ejes de la función. La adición del vector gradiente de un volumen de datos escalar a la función de transferencia mejora su habilidad de distinguir materiales de áreas envolventes (Urra and Pereira, 2011) (Gavrilescu, M. et al, 2011).

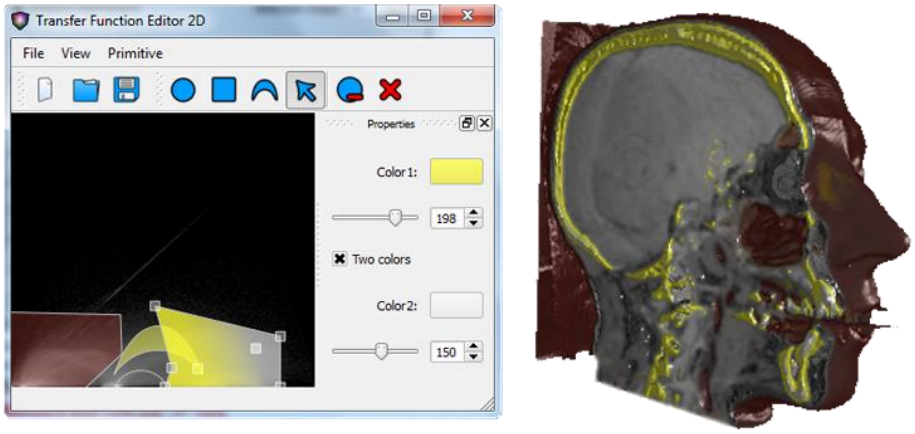

Figura 10. Editor de funciones de transferencia bidimensionales basadas en gradiente y resultado de su aplicación en la visualización

Iluminación y sombreado

La iluminación y el sombreado influyen decisivamente en el nivel de realismo que se obtiene en la visualización de volumen. Estos elementos posibilitan adicionar la contribución de fuentes de luces externas a la escena con el objetivo

Serie Científica de la Universidad de las Ciencias Informáticas
Vol. 9, No. 3, Marzo, 2016
ISSN: 2306-2495 | RNPS: 2343
http://publicaciones.uci.cu
Pág. 13-27

de mejorar la percepción de profundidad, el sentido volumétrico y la apreciación de detalles que no son visibles sin la presencia de la fuente de luz.

Para calcular la iluminación se utilizan los modelos de iluminación (Gallardo, 2001), (Alcolea and Pereira, 2011), estos pueden ser locales o globales (Engel et al., 2006). En la presente investigación se utilizó el modelo de iluminación local de Phong y el vector gradiente del volumen de datos como vector normal (Engel et al., 2006), (Alcolea and Pereira, 2011), (Preim and Bartz, 2007). Para calcular el vector gradiente se escogió el método de diferencias centrales (Preim and Bartz, 2007), (Hadwiger et al., 2009) por la rapidez y calidad de sus resultados, este se implementó en el *fragment shader*, la Figura 11 muestra los resultados alcanzados aplicando iluminación local.

Para el cálculo de las sombras, se empleó el algoritmo Rayos de Sombra (Alcolea and Pereira, 2011), (Ropinski et al., 2008). Este tiene un mayor impacto computacional en el rendimiento de la visualización, debido a que traza un segundo rayo desde la posición actual del rayo del Raycasting hasta la fuente de luz para determinar si existe algún objeto que genere sombra para el vóxel actual, los resultados visuales se muestran en la Figura 12.

Figura 11. Izquierda: Visualización sin luces. Derecha: Visualización empleando Phong como modelo de iluminación local

Serie Científica de la Universidad de las Ciencias Informáticas
Vol. 9, No. 3, Marzo, 2016
ISSN: 2306-2495 | RNPS: 2343
http://publicaciones.uci.cu
Pág. 13-27

Figura 12. Izquierda: Visualización sin luces. Derecha: Visualización empleando Rayos de Sombra como algoritmo de iluminación global.

Resultados y discusión

Con el objetivo de evaluar los resultados de la implementación del algoritmo propuesto se tendrán en cuenta dos criterios fundamentales: el rendimiento del sistema y la calidad de la representación. El primer caso se expresa mediante la cantidad de cuadros que es capaz de representar en un segundo (fps) y por el consumo de memoria RAM (Meïsner, 2000). El segundo caso, aunque con elementos de subjetividad en su apreciación, se evalúa por la precisión y realismo visual de la imagen final.

Para la realización de las pruebas se tomaron tres volúmenes de datos con diferentes dimensiones, cada volumen emplea 8 bits para sus vóxeles. La resolución de la imagen que se representó en todos los casos fue de 512x512 píxeles y como distancia para la toma de muestras se estableció la longitud de un vóxel. Las pruebas se realizaron en una computadora personal con un procesador Intel Core 2 Quad Q6600 a 2.40 GHz, 4GB de RAM y tarjeta de video NVidia GeForce 9800 GT con 512 MB para video.

En todos los casos el rendimiento del algoritmo alcanzó el tiempo real en la representación, lo que demuestra la eficiencia de la propuesta. Para el Cráneo y la Molécula, la frecuencia del monitor empleado (85 Hz) limitó la cantidad de cuadros por segundo. La siguiente tabla muestra el rendimiento normal del sistema:

Tabla 1. Resultados de saltar los espacios en blanco en el rendimiento del algoritmo

Volumen	Dimensiones	Normal (fps)	Saltando los espacios en blanco (fps)
Bonal	512x512x346	53	61
Cráneo	256x256x256	85	85
Molécula	64x64x64	85	85

La Tabla muestra los efectos que produce el Refinamiento de los puntos de intersección en el rendimiento del sistema, aunque este disminuye, aún conserva el tiempo real.

Tabla 2: Impacto del refinamiento de los puntos de intersección en el rendimiento del algoritmo

Volumen	Dimensiones	Normal (fps)	Refinando los puntos de intersección (fps)
Bonal	512x512x346	53	61
Cráneo	256x256x256	85	61
Molécula	64x64x64	85	75

Los conjuntos de datos volumétricos empleados para las pruebas se encuentran disponibles en (Roettger, 2016), (Engel, 2016). Excepto el primero, que es original del proyecto de investigación de los autores, los restantes se emplean en la mayoría de las pruebas realizadas a este tipo de algoritmos (Meïsner, 2000).

Conclusiones

La implementación del algoritmo *Raycasting* con las modificaciones realizadas permite la visualización de volúmenes, con alta calidad visual y rendimientos aceptables.

El uso de la estructura de datos *octree* resulta eficiente para saltar los espacios en blanco. Los efectos son apreciables en mayor medida cuando las dimensiones del volumen son mayores.

Se demostró la eficacia del Refinamiento de los Puntos de Intersección como técnica para mejorar la calidad de la representación, obteniéndose en todas las pruebas una ejecución en tiempo real.

Como trabajo futuro se incluye la ampliación del algoritmo para representar grandes volúmenes de datos que superan la cantidad de memoria de la GPU y brindar soporte para la segmentación previa de los volúmenes.

Referencias

ALCOLEA, R. AND PEREIRA, O: Modelos de Iluminación para Visualización Directa de Volumen. En: XIV Convención y Feria Internacional Informática 2011, Evento V Congreso Internacional de Tecnologías, Contenidos Multimedia y Realidad Virtual, Salón Virtual, 2011.

CULLIP, T.; NEUMANN, U. Accelerating volume reconstruction with 3D texture mapping hardware, Technical Report TR93-027, Department of Computer Science, University of North Carolina, Chapel Hill, 1993.

ENGEL, K. Datasets Library. [En línea] Universidad de Stuttgart. [Consultado el: 10 de enero de 2016]. Disponible en: http://www.vis.uni-stuttgart.de/~engel/pre-integrated/data.html

ENGEL, K.; HADWIGER, M.; KNISS J. AND REZK-SALAMA, C. Real-Time Volume Graphics. En: Eurographics 2006. Eurographics 2006 Tutorial Notes T7. Eurographics, 2006, p. 1-153.

GALLARDO, A.: 3D Lighting: History, Concepts and Techniques, Massachusetts, Charles River Media, 2001. 489 p.

GAVRILESCU, M. Visualization and Graphical Processing of Volume Data. PhD Thesis. Vienna University of Thechnology, Vienna, 2011.

GAVRILESCU, M.; MANTA, V.; AND GROLLER, E.: "Gradient-based classification and representation of features from volume data". En: 15th International Conference on System Theory, Control, and Computing (ICSTCC), IEEE, 2011, p. 1–6.

HADWIGER, M.; LJUNG, P.; SALAMA, C. AND ROPINSKI, T. Advanced Illumination Techniques for GPU-Based Volume Rendering. En: ACM SIGGRAPH 2009. ACM SIGGRAPH 2009 Courses. ACM,.2009, p. 1-166.

KRÜGER, J.; WESTERMANN, R. Acceleration techniques for GPU-based volume rendering. En: IEEE Transaction and Visualization. Proceedings of the 14th IEEE Visualization 2003 (VIS'03), IEEE Computer Society, 2003, p. 287–292.

MEÏSNER, M.; HUANG, J.; BARTZ, D.; MUELLER, K. AND CRAWFIS, R. A practical evaluation of popular Volume Rendering Algorithms. En: IEEE Symposium on Volume Visualization. Proceedings of the 2000 IEEE Symposium on Volume Visualization, Salt Lake City, ACM, 2000, p. 81-90.

PREIM, B. AND BARTZ, D. Visualization in Medicine. Theory, Algorithms, and Applications, San Francisco, Elsevier, Morgan Kaufmann Publishers, 2007. 663 p.

PREIM, B. AND BOTHA, C. Visual Computing for Medicine: Theory, Algorithms, and Applications. San Francisco, Elsevier, Morgan Kaufmann Publishers, 2013. 964 p.

ROETTGER, S. The Volume Library. [En línea] [Consultado el: 10 de enero de 2016]. Disponible en: http://www9.informatik.uni-erlangen.de/External/vollib/.

RODRÍGUEZ, M.; GOBBETTI, E.; GUITIÁN, J.; MAKHINYA, M.; MARTON, F.; PAJAROLA, R. AND SUTER, S. A survey of compressed GPU-based direct volume rendering. En: Eurographics 2013-STARs, 2013, p. 117-136.

ROPINSKI, T.; KASTEN, J. AND HINRICHS, K. Efficient Shadows for GPU-Based Volume Raycasting. En: WSCG 2008. Proceedings of the 16th International Conference in Central Europe of Computer Graphics. WSCG, 2008, p. 17-24.

SILVA, L. G.; RODRÍGUEZ, A.; ALCOLEA, R. AND CARRASCO, R.: "Vismedic – Illustration: Sistema para la generación de ilustraciones volumétricas". En: XVI Convención y Feria Internacional Informática 2016, 7mo Congreso Internacional de Tecnologías y Contenidos Multimedia, 2016.

SCHARSACH, H. Advanced Raycasting for Virtual Endoscopy on Consumer Graphics Hardware, Tesis de maestría, Instituto de Computación Gráfica y Algoritmos, Universidad Politécnica de Viena, 2006.

URRA, L. AND PEREIRA, O. Funciones de Transferencia para Visualización Directa de Volumen, En: XIV Convención y Feria Internacional Informática 2011, Evento V Congreso Internacional de Tecnologías, Contenidos Multimedia y Realidad Virtual, Salón Virtual, 2011.

ZHANG, Q.; EAGLESON, R. AND PETERS, T. Volume visualization: a technical overview with a focus on medical applications. Journal of Digital Imaging, 2011, 24(4): p.640-664.