resumen

Este artículo presenta el proceso de renderizado distribuido de ambientes gráficos tridimensionales usando como mecanismo de comunicación entre procesos a RMI y el sistema de nombres y servicios JNDI. Se usaron dos técnicas de renderizado distribuido: Sort-First y Sort-Last, para comparar los resultados.

Renderizado distribuido usando RMI-JNDI

Diana Lucía Reyes R. • Alfonso Barbosa L.

introducción

La computación gráfica ha evolucionado rápidamente gracias al aporte de otras líneas de investigación, tales como los sistemas distribuidos y los diferentes mecanismos de comunicación. En la actualidad existen varias arquitecturas que permiten realizar el renderizado distribuido, pero por lo general usan solo como mecanismo de comunicación los Sockets. En este artículo se presenta una arquitectura flexible que permite realizar de forma distribuida el proceso de renderizado configurando de manera dinámica tanto la técnica de renderizado como el mecanismo de comunicación.

Para definir los requerimientos mínimos de la arquitectura del sistema de renderizado distribuido para ambientes gráficos 3D (REDAG3D) se tuvieron en cuenta conceptos de sistemas distribuidos, computación gráfica y técnicas de renderizado distribuido. Para probar la arquitectura se realizó la implementación de una aplicación que puso en funcionamiento cada una de las características de los componentes definidos. Así mismo, se creó una escena constituida por modelos 3D complejos para validar las características de desempeño y flexibilidad de la arquitectura. La aplicación REDAG3D permite al usuario final seleccionar el mecanismo de comunicación (Sockets y RMI-JNDI), así como la técnica de renderizado distribuido (Sort-First – Sort-Last), en tiempo real. Según la comparación preliminar de las técnicas de distribución usando los dos mecanismos de comunicación (realizando escalamiento horizontal de las máquinas visualizadoras), RMI-JNDI presentó un mejor rendimiento.

Este artículo se enfoca en el comportamiento de las técnicas de renderizado distribuido usando el mecanismo de comunicación RMI-JNDI.

Antecedentes

JNDI es un API que proporciona funcionalidades de nombrado y directorio a las aplicaciones, definido para ser independiente de cualquier implementación de servicio de directorio. Las aplicaciones usan RMI-JNDI para acceder a una gran variedad de servicios de nombres y directorios. Se debe definir un SPI (Service Provider Interface) para que se puedan conectar varios servicios de nombres y directorios, esto permite que las aplicaciones accedan a los servicios de JNDI [1]. Los Sockets son usados dentro de la arquitectura para establecer la comunicación inicial entre los diferentes procesos encargados de realizar el renderizado distribuido [2]. Otro punto importante, es la existencia de la arquitectura de tubos y filtros (pipeline) de renderizado de gráficos 3D, que convierte la representación geométrica de un mundo virtual 3D en una imagen 2D foto-realista [3], la Figura 1 muestra las diferentes etapas del proceso. La entrada al pipeline es una escena conformada por un conjunto de objetos típicamente representados como una malla de triángulos. Las dos etapas principales son la transformación geométrica y la rasterización. En la etapa de transformación geométrica se mapean triángulos de un sistema de coordenadas 3D (espacio de objeto) en un sistema de coordenadas 2D (espacio de imagen) llevando a cabo unas series de transformaciones. En la etapa de rasterización se convierte los triángulos transformados en píxeles [4] [7].

Figura 1. Pipeline de gráficos 3D

Un algoritmo computacional que permite generar imágenes "realistas" es el raytracing, que se basa en la trayectoria seguida por rayos de luz individuales trazados desde la posición de la cámara hacia la escena. Estos rayos se intersectan con los objetos 3D de manera que se determinan las superficies visibles al observador [5] [8] [9].

Renderizado distribuido

Sort-First y Sort-Last son técnicas de renderizado distribuido. La diferencia radica en la forma como se distribuye la geometría y en la etapa del proceso de renderizado en donde

se realiza el ordenamiento. En Sort-First, el espacio de pantalla se divide por regiones, las primitivas que pertenecen a cada región se clasifican según su ubicación, a cada procesador se le asigna una región y éste se encarga de renderizar las primitivas respectivas. Luego se realiza la transformación geométrica y la rasterización [6]. En la Figura 2 se pueden ver las etapas del proceso de renderizado en la arquitectura Sort-First.

En Sort-Last cada procesador realiza el proceso de transformación geométrica y rasterización de las primitivas asignadas, posteriormente se realiza un proceso de composición de cada representación para formar la imagen final [6]. En la Figura 3 se pueden ver las etapas del proceso de renderizado en la arquitectura Sort-Last.

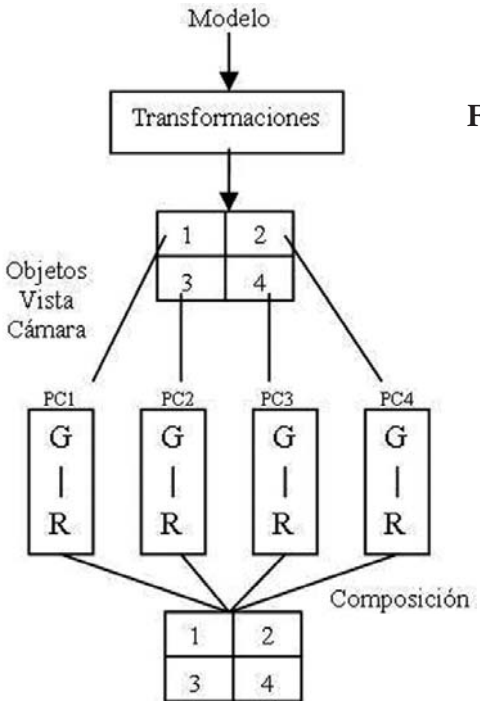


Figura 2. Etapas de Sort-First

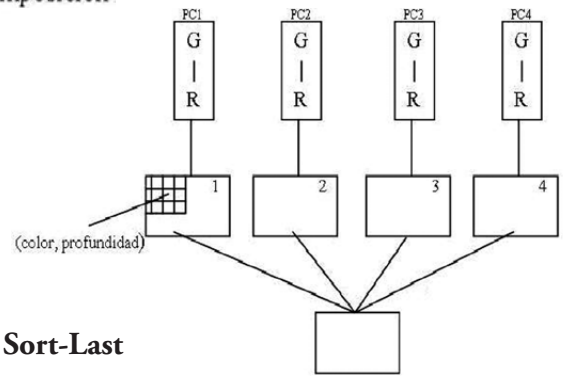
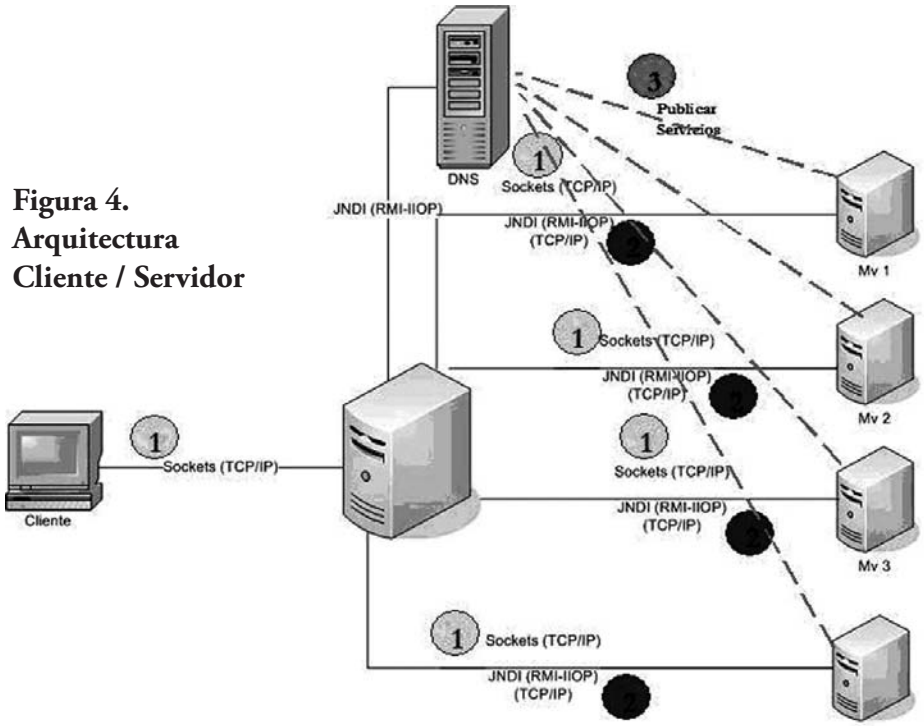


Figura 3. Etapas de Sort-Last

Diseño e implementación de REDAG3D

La arquitectura propuesta para REDAG3D se definió de acuerdo con los requerimientos del sistema. En la Figura 4 se muestra una vista de despliegue, donde se especifica la forma en que las máquinas están organizadas físicamente y el mecanismo de comunicación entre ellas. Existe un servidor encargado de la administración del sistema y varios clientes que pueden realizar solicitudes al mismo (máquinas visualizadoras). Sin embargo, el servidor actúa como cliente durante gran parte del proceso de renderizado, toda vez que utiliza los servicios prestados por las máquinas visualizadoras que se convierten en servidores de renderizado.

Figura 4.
Arquitectura
Cliente / Servidor



Inicialmente, el cliente establece la comunicación con el servidor administrador mediante el uso de Sockets, el cual se conecta con las máquinas visualizadoras (Ver Figura 4 Numeral 1). Luego el servidor envía solicitudes a cada máquina para configurar la técnica de renderizado que será utilizada en el proceso. Posteriormente, el servidor envía una solicitud de activación para que cada máquina visualizadora publique los servicios disponibles ante un Directorio de Nombres y Servicios (DNS) (Ver Figura 4 Numeral 2). A partir de ese momento las máquinas se convierten en servidores y el servidor administrador consumirá sus servicios. Después de publicados los servicios, el servidor recupera el stub del servicio que va a solicitar y delega a cada máquina la carga correspondiente (Ver Figura 4 Numeral 3).

Si se selecciona la técnica Sort-Last, las mallas se distribuyen a cada máquina visualizadora de manera aleatoria. Al seleccionar la técnica Sort-First, cada malla se distribuye en las máquinas visualizadoras según una clasificación previa, dependiendo de su ubicación en el espacio de pantalla. Una vez se ha distribuido la carga, el servidor la envía a cada máquina. En cada máquina visualizadora se administra la geometría manipulando las mallas por medio de funciones que ofrece el toolkit VSDK (Vitral Software Development Kit) [10], y posteriormente se renderiza la escena de acuerdo a la técnica seleccionada, retornando la imagen resultado al servidor administrador. Con Sort-Last el resultado del proceso de renderizado por cada máquina es una imagen y un buffer de profundidad, mientras que con Sort-First el resultado del proceso de renderizado por cada máquina es una sección de la imagen final. Una vez terminada la fase de renderizado el servidor administrador debe componer la imagen resultado para que sea presentada al Cliente. Con Sort-Last, esta imagen se construye aplicando el algoritmo de z-buffer y con Sort-First uniendo las imágenes parciales (secciones). En las Figuras 5 y 6, se muestra un esquema general del proceso de renderizado distribuido para cada una de las técnicas (ver Figura 6 página 83).

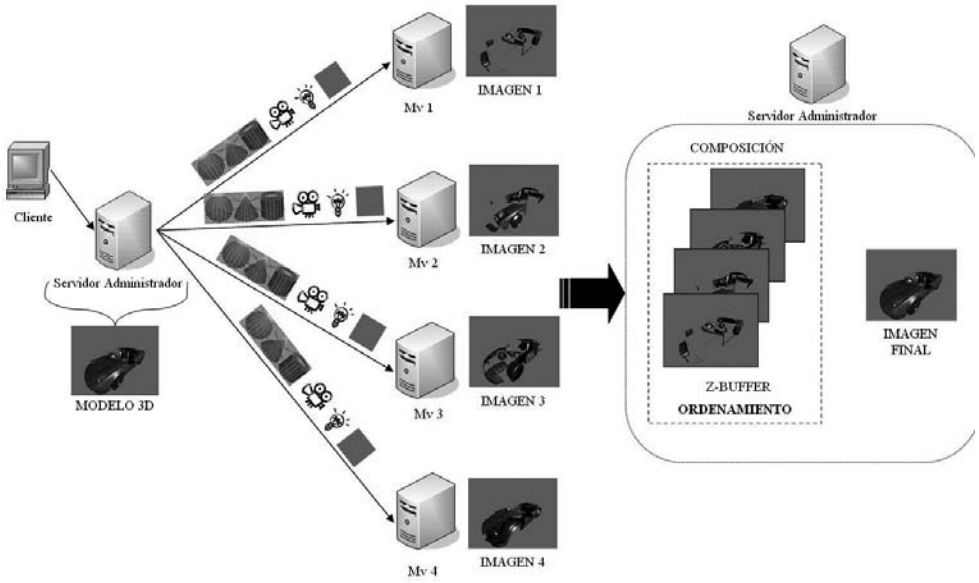


Figura 5. Proceso de Renderizado Distribuido en REDAG3D con Sort-Last

Pruebas de Renderizado

Se verificó la eficiencia de cada técnica de renderizado y para cada máquina renderizadora se tomó el tiempo de inicio y fin del renderizado más el tiempo de envío de cada imagen al servidor administrador y de composición de la imagen final. En la Figura 7 se muestran

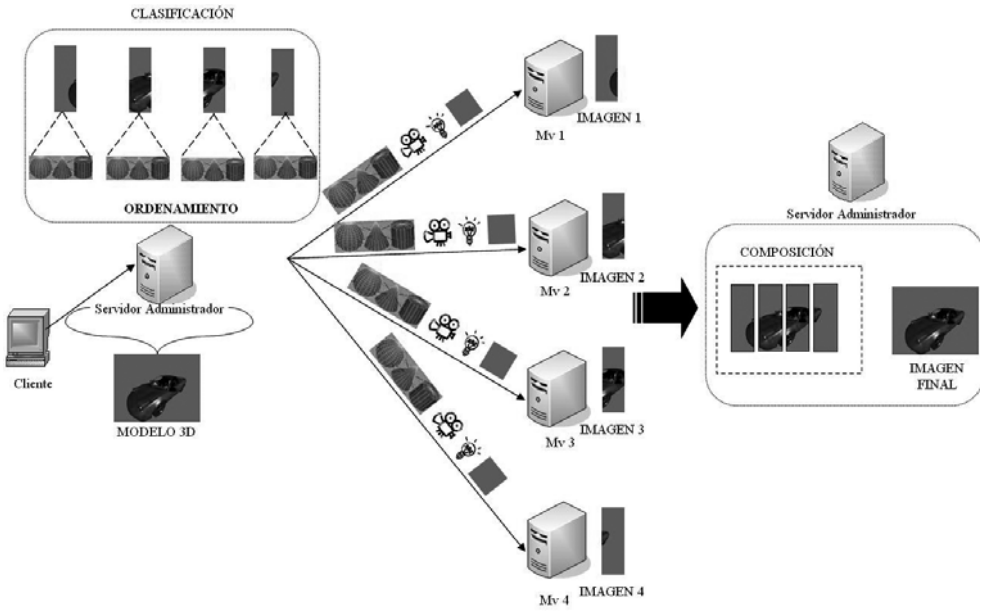


Figura 6. Proceso de Renderizado Distribuido en REDAG3D con Sort-First

los resultados de estas pruebas y se observa que el comportamiento de las técnicas es similar. Con 2 y 4 máquinas visualizadoras Sort-First es un poco más eficiente, pero a partir de 8 máquinas mejora el desempeño de Sort-Last. Con pocas máquinas Sort-First tiene un mejor comportamiento debido a que no se generan imágenes completas, por lo cual el envío de estas a través de la red tarda menos tiempo que con Sort-Last.

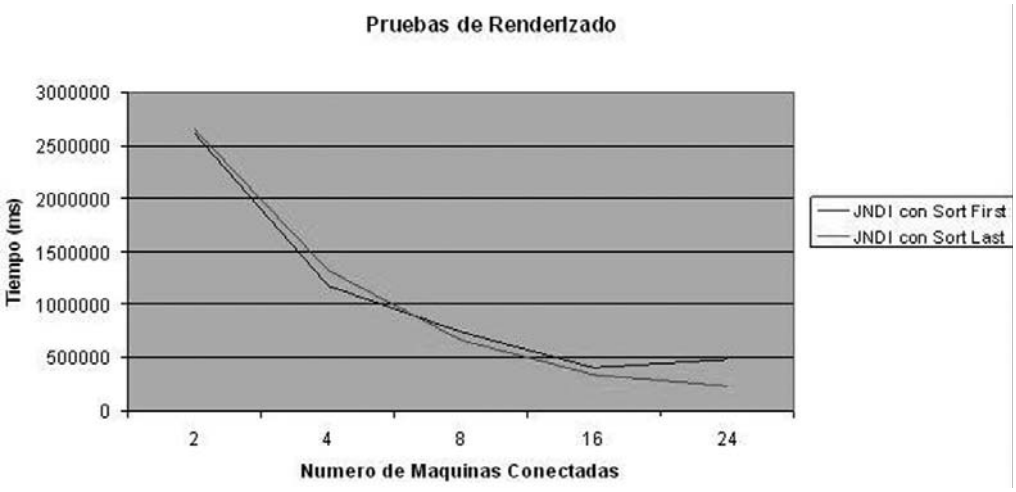


Figura 7. Sort-Last y Sort-First con el mecanismo JNDI

Conclusiones

Con un número de máquinas considerable, a partir de 8 para la aplicación REDAG3D, es más alto el impacto del ordenamiento en Sort-First que el ordenamiento realizado en Sort-Last.

La técnica Sort-First, implica un mayor tiempo de procesamiento, debido a la duplicidad en el renderizado de mallas que ocupan gran parte del espacio de pantalla, por la clasificación previa que se debe realizar.

Es más exigente la aplicación del algoritmo de raytracing al usar una escena con mayor número de mallas, que aplicar el algoritmo de z-buffer con una gran cantidad de imágenes.

En la técnica Sort-First se presenta un problema de escalabilidad al aumentar el número de máquinas visualizadoras. Algunas máquinas tuvieron mayor carga de procesamiento debido a que muchas mallas pertenecen a su región (subdivisión) respectiva. También influye el hecho de que algunas mallas que ocupan gran parte del espacio de pantalla son comunes a varias subdivisiones.

Referencias

[1] Tutorial JNDI, Url: http://www.programacion.com/tutorial/jndi/6, Última consulta: 7 de Mayo de 2007.

[2] Los Sockets en Java, Url: http://pisuerga.inf.ubu.es/lsi/Invest/Java/Tuto/V_2.htm, Última consulta: 18 de Abril de 2007.

[3] Iosif Antochi, Ben Juurlink, Stamatis Vassiliadis. A Flexible Simulator for Exploring Hardware Rasterizers. Computer Engineering Laboratory, Delft University, 2002.

[4] Tulika Mitra, Tzi-cker Chiueh. Compression-Domain Parallel Rendering. Proceedings of the 16th International Symposium on Parallel and Distributed Processing, IEEE Computer Society. 2002.

[5] J.Foley, A. VanDam, S. Feiner, J. Hughes. "Computer Graphics. Principles and practive." Addison-Wesley Publishing Company, Inc. 1991.

[6] Steven Molnar, Michael Cox, David Ellsworth, Henry Fuchs. "A Sorting Classification of Parallel Rendering". IEEE Computer Graphics and Applications. Volume 14, Issue 4, July 1994 Page(s):23 - 32 1994.

[7] Oliver Staadt, Justin Walker, Christof Nuber, Bernd Hamann. "A Survey and Performance Analysis of Software Platforms for Interactive Cluster-Based Multi-Screen Rendering". The Eurographics Association. 2003.

[8] Rudrajit Samanta, Thomas Funkhouser, Kai Li, and Jaswinder Pal Singh, Hybrid Sort-First and Sort-Last, Parallel Rendering with a Cluster of PCs, In Eurographics/SIGGRAPH workshop on Graphics hardware, pages 99-108. ACM Press, August 2000

[9] Carl Mueller, The Sort-First Rendering Architecture for High-Performance Graphics, Computer Graphics, ACM SIGGRAPH Special Issue on 1995 Symposium on Interactive 3-D Graphics, April 1995.

[10] Toolkit para realidad virtual - Vitral, grupo de investigación TAKINA, url: http://sophia.javeriana.edu.co/~ochavarr/vsdk/html_doxygen/ Última consulta: 21 de Junio de 2007.

autores

Diana Lucía Reyes R. Candidata al título de Ingeniería de Sistemas de la Pontificia Universidad Javeriana de Bogotá, Colombia. Trabajó como analista en atención y soporte a usuarios en el Banco De Occidente Credencial. Se desempeñó como Directora del Proyecto Atlas a nivel nacional de la empresa ICELL. LTDA. Participó en el Diseño y Desarrollo de un Sistema de Información para la Administración de la Cartera del Departamento de Cobranzas de la Empresa Ecoaser y Cía. LTDA. Actualmente, es analista de negocios de Visión Software.

Alfonso Barbosa L. Candidato al título de Ingeniería de Sistemas de la Pontificia Universidad Javeriana de Bogotá, Colombia. Trabajó en el Diseño y Desarrollo de un Sistema de Información para la Administración de la Cartera del Departamento de Cobranzas de la Empresa Ecoaser y Cía. LTDA. Actualmente se desempeña como eTalent en el departamento de Preventa de la empresa Oracle Colombia.