UNIVERSIDAD DE LAS CIENCIAS INFORMÁTICAS

Título: “Sistema de Gestión de Renderizado Distribuido.”

Trabajo de Diploma para optar por el título de
Ingeniero en Ciencias Informáticas

Autor: Nelio Véliz Pedraza

Tutores: Ing. Yunier Vega Rodriguez
Msc. Yenisleydi Cariaga Cristo

Ciudad de La Habana
Mayo del 2010

Investigar es ver lo que todo el mundo ha visto, y pensar lo que nadie más ha pensado.

Albert Szent Gyorgi

DECLARACIÓN DE AUTORÍA

Declaro que soy el único autor de la presente tesis y reconozco a la Universidad de las Ciencias Informáticas los derechos patrimoniales de la misma, con carácter no exclusivo.

Para que así conste firmo la presente a los 22 días del mes de Mayo del año 2010

_____ _____

Nelio Véliz Pedraza Msc. Yenisleydi Cariaga Cristo

_____

Ing. Yunier Vega Rodriguez

Agradecimentos

A mi madre…maa. Por la confianza que en mi ha depositado, por entenderme como soy y por no faltarme nunca una gota de su cariño.

A mi padre...pep. Por sus consejos y por ser mi ejemplo a seguir en todo momento.

A mi hermano...nee el cual espero ver realizar lo mismo algún día.

A mi abuela Carmen por ser mi segunda madre, por todo su cariño y por hacerme reír tanto.

A mis tíos y primos por todo el apoyo y ayuda que recibí de ellos.

A mi tía Milagro de la cual nunca me faltó su apoyo en todo este tiempo.

A mis tutores, Yeni y Yunier, sin ellos no hubiera sido posible esta tesis, son dos personas excepcionales tanto como tutores como amigos.

A los profesores y amigos que nos acompañaron en este largo viaje que ha sido la UCI.

Dedicatoria

Dedico especialmente este trabajo de diploma a mis padres por estar siempre conmigo y apoyarme, por su amor incondicional y confianza, por sus consejos y su ejemplo, por guiarme por el buen camino para hacer de mí una mejor persona y un profesional, espero que siempre estén orgullosos de mí y no defraudarlos nunca.

A mi familia, por todo el apoyo que siempre me brindan, por hacerme sentir tan especial para ustedes, por confiar en mí y por estar ahí cuando más lo necesitaba.

RESUMEN

En la actualidad, la renderización se ha convertido en una de las industrias más importantes del mundo, gracias a lo cual se puede disfrutar en pantalla de las producciones audiovisuales realizadas con los programas de infografía[1], que tan buena acogida han tenido entre los telespectadores de todo el planeta.

Los Sistemas de Gestión de Renderizado Distribuido (SGRD) o granjas de render (*render farm* en inglés) están siendo la solución más conveniente para mejorar el tiempo, que demora el renderizado de una película. Lamentablemente la mayoría de los software creados para estos fines, son privativos y bajo licencias de pago, al alcance solo de los países desarrollados.

El avance que ha tenido el Software Libre (SWL) y los programas Open Source[2], permiten en la actualidad contar con SGRD de alta calidad al alcance de todos, como alternativa a los privativos, fundamentalmente para los países del Tercer Mundo como Cuba. Estos no obstante presentan deficiencias tanto en los procesos de gestión como en su arquitectura.

El presente trabajo de diploma tiene como propósito desarrollar un SRGD libre, que posibilite gestionar eficientemente cada uno de sus componentes, implemente diferentes algoritmos y proporcione distintas variantes de distribución.

Palabras Claves: 3D, Granja de render, Open Source, Software Libre.

1 Son programas que tratan de imitar el mundo tridimensional mediante el cálculo del comportamiento de la luz, los volúmenes, la atmósfera, las sombras, las texturas, la cámara, el movimiento, etc. Ejemplo: Blender, 3D Max etc.

2 *Open Source* (Código abierto en español) es el término con el que se conoce al software distribuido y desarrollado libremente. El código abierto tiene un punto de vista más orientado a los beneficios prácticos de compartir el código que a las cuestiones morales y/o filosóficas las cuales destacan en el llamado Software Libre.

ÍNDICE GENERAL

INTRODUCCIÓN....1
CAPÍTULO 1: FUNDAMENTACIÓN TEÓRICA....5
1.2 Sistema de Renderizado Distribuido....9
1.2.2 Principales motores de render....12
Métodos de Render....12
Motores Comerciales....14
Motores Open Source....15
1.2.3 Sistemas de Gestión de Renderizado Distribuido....17
Render Distribuido....17
Sistemas de gestión de Renderizado Distribuido Libres. 23
1.2.4 DrQueue....25
1.3 Principales compañías que ofrecen servicio de renderización:....32
CAPÍTULO 2: DISEÑO DE ARQUITECTURA PARA SISTEMA DE GESTIÓN DE RENDERIZADO DISTRIBUIDO. 35
2.1 Arquitectura de Software....35
2.1.2 Arquitectura Cliente-Servidor....38
2.2 Lenguajes, herramientas, librerías y metodología para el desarrollo....39
2.2.1 Lenguajes de programación....39
Lenguaje de programación C....40
Lenguaje de programación Python....41
Bash....42
2.2.2 Librerías....43
GTK....43
2.2.3 Herramientas:....43
GForge....43
IDE....43
Metodología de desarrollo....44
2.3 Descripción de la arquitectura 45
2.3.1 Componentes de la arquitectura....46
Cliente/Usuario (C/U)....46
Gestor de Renderizado Distribuido (GRD)....46
Servidor de Render (SR)....47
2.3.2 Variantes de distribución....48
CAPÍTULO 3: IMPLEMENTACIÓN DE SISTEMA DE RENDERIZADO DISTRIBUIDO . .50
3.1 Historias de Usuario....51
3.2 Algoritmo....51
3.3 Desventajas del algoritmo anterior y ventajas del nuevo algoritmo....53
3.4 Pruebas realizadas....56
3.4.1 Hardware....56
3.4.2 Trabajos Realizados....57
3.4.3 Comparación entre los algoritmos....59
3.4.4 Historias Nativas....59

3.5 Análisis de los resultados.....61
RECOMENDACIONES.....64
REFERENCIAS BIBLIOGRÁFICAS.....65
BIBLIOGRAFÍA.....74
ANEXOS.....75
GLOSARIO DE TÉRMINOS.....86

INTRODUCCIÓN

A nivel mundial, la producción de audiovisuales con tecnología 3D, ha alcanzado una gran popularidad en estos tiempos. Productoras como *Pixar* (Pixar 2010) o *Fox Animation* (Fox Animation 2010) han cosechado numerosos reconocimientos por sus cortometrajes, largometrajes y logros técnicos, al tiempo que películas como "*The incredibles*", "*Ice Age*" o "*Toy Story*", y más recientemente "*Avatar*", y "*2012*", han hecho de la producción 3D un éxito rotundo.

En Cuba la producción de animaciones 3D aún no ha alcanzado un gran avance, no obstante ya se comienzan a dar los primeros pasos. La Universidad de Ciencias Informáticas (UCI) en conjunto con el Instituto Cubano de Arte e Industria Cinematográfico (ICAIC), tienen en producción el primer largometraje en 3D nombrado: Meñique.

Toda producción de animaciones 3D consta de tres etapas fundamentales que son Pre-producción, Producción y Post-Producción (I. Victor Kerlow 2000). Dentro de la producción, el renderizado[3] juega un papel importante, pues se encargará de realizar una serie de cálculos y generar imágenes 2D a partir de los modelos 3D, o sea, la computadora "interpretará" las escenas en tres dimensiones y la plasmará en imágenes bidimensionales. Normalmente esta es la fase más intensa del proceso completo en lo que a cómputo y tiempo se refiere. Producto de ello se han investigado muchas formas de optimizar esta fase, enfocándose la mayoría en el proceso de render distribuido, particularmente en los SGRD.

Los SGRD son una agrupación de varias computadoras repartiéndose el trabajo de renderizar las imágenes o animaciones realizadas en un programa de infografía, lo que permite disminuir notablemente el tiempo que demoraría una sola computadora. En la película "*Monstruos S.A.*", la renderización de algunos fotogramas podía durar hasta 80 horas, debido a los efectos complejos de la iluminación, es decir, aproximadamente 3 días por cada fotograma de

3 Ver capítulo 1.

película. Esta película cuya duración es de 92 minutos, habría tardado en ser renderizada aproximadamente 1000 años en una sola máquina. *Pixar* utilizó centenares de máquinas para paralelizar el proceso y terminar el proyecto en apenas dos años (Cediant 2009).

Lamentablemente la mayoría de los *software* que se realizan para las granjas de render son de manera privativa y bajo licencias de pago, siendo estos utilizados principalmente por los países desarrollados, no ocurriendo lo mismo para los del Tercer Mundo.

El crecimiento del SWL como alternativa al uso de Software Privativo (SP), es cada vez mayor. Grupos de personas sin fines lucrativos y mediante el trabajo colaborativo aumentan constantemente el desarrollo de programas libres. Los SGRD no han quedado exentos de esta alternativa, existiendo en la actualidad un gran número y al tener su código fuente accesible, contar con comunidades que los mejoran continuamente.

En la UCI, más específicamente en la facultad 10, fue creado el grupo de proyectos Unicornios cuyo propósito es la migración y soporte de aplicaciones y servicios con fundamento en el SWL. Uno de los sub-proyectos que integra a este grupo es FreeViUX, que surge por la necesidad de crear un grupo de diseñadores que desarrollara productos audiovisuales utilizando sólo SWL, enfocándose en la eliminación acelerada de las herramientas privativas en todas las áreas donde programas como: Photoshop, 3D Max, Maya, Softimage|XSI, Nuendo y Adobe Premiere, aumentaban la dependencia tecnológica que tiene el país con otros del primer mundo. Este ha sido la posible solución para un estudio de diseño y animación basado en herramientas Open Source.

Producto de la realización constante de animaciones en FreeViUX el renderizado juega un papel fundamental, por ese motivo cuenta con un SGRD que ayuda a reducir los tiempos que este proceso tarda. El SGRD usado actualmente es SWL pero este está presentando ciertos inconvenientes.

La **situación problémica** se evidencia en la necesidad de optimizar el proceso de gestión de renderizado distribuido para la producción de audiovisuales con tecnología 3D utilizando Software Libre.

La creación de un SGRD eficiente en la UCI, teniendo en cuenta la capacidad de cómputo que posee, permitirá reducir los tiempos de finalización y hará rentable pequeños y grandes proyectos de animación tanto nacionales como internacionales. Además por ser SWL permitirá el ahorro de importación por adquisición de *software.*

El **problema científico** a resolver entonces es: ¿Cómo mejorar el proceso de gestión de renderizado distribuido para la producción de audiovisuales con tecnología 3D utilizando Software Libre?

Se determina como **objeto de estudio** son los Sistemas de Gestión de Renderizado Distribuido y **campo de acción** los Sistemas de Gestión de Renderizado Distribuido con Software Libre.

El **objetivo general** de este trabajo es desarrollar un Sistema de Gestión de Renderizado Distribuido para la producción de audiovisuales con tecnología 3D utilizando Software Libre, teniendo como **objetivos específicos**:

- Sistematizar en los Sistemas de Gestión de Renderizado Distribuido, haciendo énfasis en aquellos basados en Software Libre.
- Diseñar Sistema de Gestión de Renderizado Distribuido.
- Implementar Sistema de Gestión de Renderizado Distribuido.

Con el propósito de dar cumplimiento a los objetivos planteados se hace necesario realizar las siguientes **tareas**:

- Revisión de la bibliografía disponible en formato duro y digital.
- Modelación arquitectónica del Sistema de Gestión de Renderizado Distribuido.
- Implementación de Sistema de Gestión de Renderizado Distribuido.

Esta investigación está sustentada sobre la base de la utilización de diferentes **métodos científicos** para la realización de la misma. Como métodos científicos teóricos se emplearon el Analítico-Sintético que permitió dividir un SRGD distribuido en partes y así comprenderse mejor su funcionamiento y la Modelación para la realización de un nuevo modelo como propuesta para la arquitectura. Como métodos científicos empíricos se utilizaron la Entrevista para conocer los errores que estaba presentando el sistema y la Observación para ver en las pruebas si se alcanzaba el resultado esperado.

La **idea a defender** es: un Sistema de Gestión de Renderizado Distribuido eficiente permitiría el desarrollo de audiovisuales 3D en un tiempo razonable.

El contenido de este documento está estructurado en capítulos, así como las secciones Glosario de términos, Referencias bibliográficas, Bibliografía y Anexos; organizados de la siguiente forma:

CAPÍTULO I: “Fundamentación teórica” donde se incluyen todos los aspectos teóricos que soportan este proyecto.
CAPÍTULO II: “Diseño de arquitectura para sistema de gestión de renderizado distribuido” donde se muestra la nueva arquitectura a usar.
CAPÍTULO III: “Implementación de sistema de renderizado distribuido” donde se explica la inclusión de un algoritmo de fragmentación de frame.

CAPÍTULO 1: FUNDAMENTACIÓN TEÓRICA

En la actualidad, las grandes productoras cinematográficas hacen uso de las computadoras para obtener escenas e incluso películas completas de animación. Los gráficos por computadora también están siendo muy utilizados en la industria de los videojuegos, destacándose en este aspecto la compañía *Sony Computer Entertainment* (Sony 2010).
Muchos están siendo los campos hacia los que los gráficos 3D están jugando un papel importante. Por lo tanto es primordial el estudio del proceso de producción 3D y con esto el análisis de donde puede ser optimizado.

Para el proceso de producción de gráficos y animaciones 3D se pueden adoptar diferentes estrategias, pero siempre se debe tener en cuenta tres etapas básicas (I. Victor Kerlow 2000). Ver Figura 1.

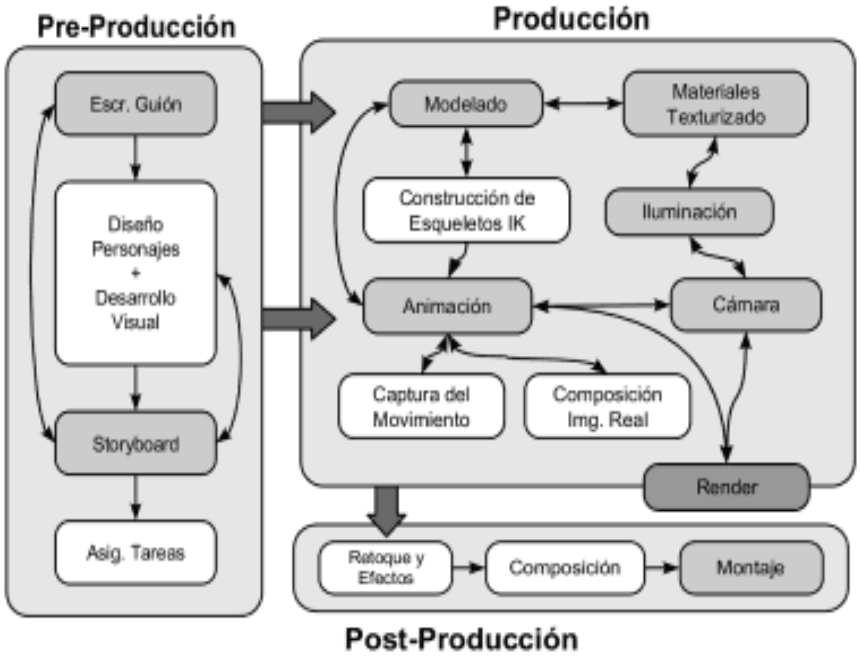


Figura 1: *Etapas de la producción de gráficos y animaciones 3D*

1. Pre-producción

La fase de pre-producción comienza habitualmente con la escritura del guión. Tanto en pequeños como en grandes proyectos, suele ser la primera tarea. Con el guión definido, el equipo de desarrollo visual, habitualmente formado por ilustradores, establece la dirección visual y el estilo del proyecto. De igual forma, se desarrollan las hojas de personajes, con los bocetos del aspecto que tendrán los personajes que serán incluidos en el proyecto.

La fase de pre-producción finaliza con la construcción del *Storyboard*[4] que mediante el mismo trasladamos el guión a imágenes.

2. Producción

En esta etapa existen una serie de fases que serán claves las cuales mencionamos a continuación:

Etapas básicas del proceso de producción:

- **Modelado:** En esta etapa se obtiene una representación tridimensional de los objetos que intervendrán en la escena.
- **Materiales y Texturas:** Mediante los materiales, aplicamos propiedades básicas de reflexión de la luz, color, transparencia a las superficies de nuestros modelos. El material se aplica de forma constante a lo largo de toda la superficie del modelo. Las texturas permiten variar las propiedades del material.
- **Iluminación:** En la búsqueda de la generación de imagen *fotorrealista*[5] un punto clave es la simulación de la luz.
- **Animación:** Tanto las herramientas de animación 2D como las 3D suelen emplear *curvas de interpolación*[6] para calcular la posición clave del parámetro a animar entre

4 Un storyboard o guión gráfico es un conjunto de ilustraciones mostradas en secuencia con el objetivo de servir de guía para entender una historia.

5 El fotorrealismo es la cualidad de una imagen generada por computadora que trata de imitar las imágenes generadas por cámaras fotográficas.

6 En la matemática se denomina interpolación a la construcción de nuevos puntos partiendo del conocimiento de un conjunto discreto de puntos. En estos problemas de interpolación, se utiliza a menudo la interpolación

frames[7]. En animación de personajes suelen emplearse esqueletos internos de animación. Así, el animador establece las rotaciones de estos huesos y el programa calcula la deformación de debe aplicar a la superficie exterior.

➢**Renderización:** En este paso se realiza el cálculo de la imagen 2D correspondiente a la escena que hemos definido (Ver Figura 2). El motor de Render tiene en cuenta todos los parámetros definidos en las etapas anteriores, y trata de realizar una simulación física de la interacción de la luz en la escena. Existen varios métodos de render, siendo norma habitual que a mayor realismo, mayor tiempo de cómputo. El tiempo de render es un parámetro importantísimo a tener en cuenta a la hora de afrontar proyectos complejos.

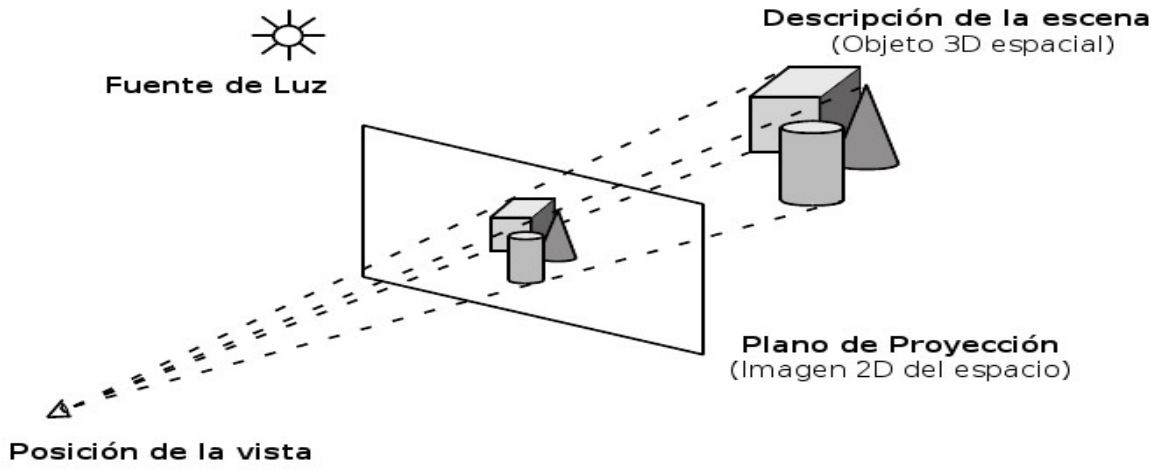


Figura 2: *Renderización*

3. Post-producción

La etapa de post-producción toma como entrada las imágenes generadas en la etapa de renderización de la fase anterior y las compone, aplicándoles una serie de filtros y modificadores antes de generar las imágenes definitivas en el formato de publicación final.

mediante curvas porque da lugar a resultados similares requiriendo solamente el uso de polinomios de bajo grado.

7 Cuadro o imagen individual que componen una animación. La sucesión de frames consecutivos con pequeñas diferencias producen la ilusión de movimiento. La velocidad de la animación se mide en frames por segundo.

Estas imágenes suelen emplearse como *capas* para la composición del fotograma final. En esta fase además se suelen retocar cada una de las capas que formarán el fotograma final, ajustando niveles, brillo, contraste, etc.

Se puede notar que la etapa de producción es principalmente costosa por el **renderizado**. Este proceso se centra básicamente en generar una imagen 2D a partir de una descripción abstracta, que involucra la geometría de la escena, la definición de las fuentes de luz, las posiciones de las cámaras y el uso de materiales. El renderizado supone normalmente la fase más intensa del proceso completo en lo que a cómputo se refiere y requiere también una gran cantidad de tiempo para llevarse a cabo, más aún, si la escena a renderizar es compleja o se necesitan imágenes realistas de alta calidad. Por estas razones el proceso de renderizado se convierte en un "cuello de botella"[8] en la producción de gráficos 3D.

Muchas han sido las ideas, que se han investigado para reducir el tiempo de renderizado, aunque según Chalmers (A. Chalmers 2002) se siguen tres líneas principales de investigación:

1. Optimización mediante hardware:

Existen varias formas de lograr el renderizado por hardware, una de ellas es el uso de unidades de procesamiento gráfico o GPUs (por el acrónimo en inglés *Graphics Processing Unit*), es decir, procesadores dedicados exclusivamente al procesamiento de gráficos, aligerando la carga de trabajo del procesador central o CPU (por el acrónimo en inglés de *Central Processing Unit*) , o sea, mientras gran parte de lo relacionado con los gráficos se procesa en la GPU, la CPU puede dedicarse a otro tipo de cálculos (como la inteligencia artificial o los cálculos mecánicos en el caso de los videojuegos).

8 Cuando se menciona cuellos de botella se refiere a diferentes actividades que disminuyen la velocidad de los procesos, incrementan los tiempos de espera y reducen la productividad, trayendo como consecuencia final el aumento en los costos.

Algunos ejemplos de este tipo de optimización son la existencia del motor de render Parthenon Renderer (T. Hachisuka. 2005) el cual fue diseñado específicamente para ser utilizado con aceleración GPU; y Gelato (Nvidia Corporation 2008), que es un software de renderizado, que permite a cualquier persona que disponga de una GPU NVIDIA crear imágenes espectaculares a toda velocidad.

Otra opción es la de utilizar hardware diseñado especialmente para tareas de renderizado pero esto genera el problema de que sería específico para un tipo de arquitectura.

El uso de GPUs programables supera a las CPUs de una estación de trabajo estándar en un factor de siete (I. Buck 2004), pero su adquisición es muy costosa (ver Figura 2 Anexos) por lo que resulta no ser la mejor opción con la cual reducir el tiempo de renderizado.

2. Mejoras en los algoritmos:

Los algoritmos de render constantemente han sido optimizados a tal punto de que actualmente es bastante complicado lograr obtener una mejora en el tiempo de renderización. Estos se han estandarizados en el mundo.

3. Optimización utilizando computación distribuida:

Lo más común en este punto es el uso de los SGRD o granja de render (*render farm* en inglés) para acelerar la producción de fotogramas.

1.2 Sistema de Renderizado Distribuido

Se llama Sistema de Renderizado Distribuido a todos los componentes, que intervienen desde que se comienza la modelación gráfica hasta el proceso de renderización.
Este consta de 3 partes fundamentales. Ver Figura 3.

- El software de Infografía.
- El SGRD*.
- El motor de render.

*El SGRD[9] esta formado por el cliente, el gestionador o gestor y el servidor de render.

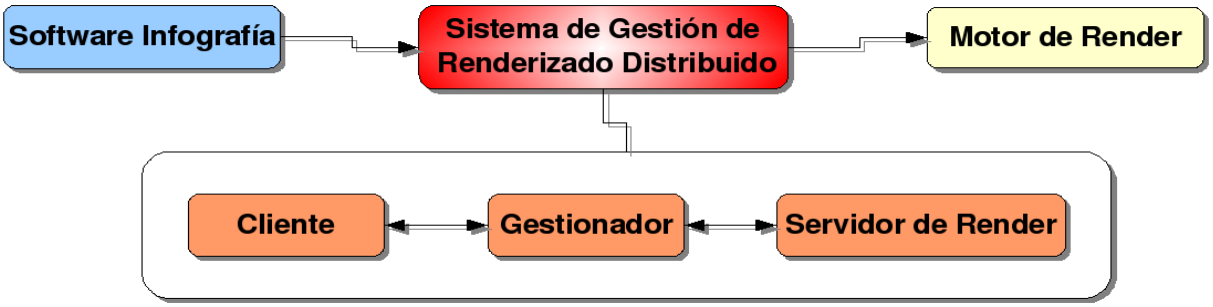


Figura 3: *Sistema de Renderizado Distribuido*

1.2.1 Principales software de infografía

Para realizar una escena 3D, se modelan los objetos, personajes o entornos, se texturizan y se ilumina la escena, de una forma muy similar a la de un estudio de fotografía. Para esto existen un abanico de *software* de infografía en el mercado, de los cuales hay que destacar por la cantidad de artistas que los manejan y reconocidos por sus altas prestaciones y prestigio en el mundo cinematográfico o arquitectural los que a continuación se presentan:

➢Maya: Es quizás el software más popular en la industria, por lo menos hasta el 2003. Factible para infoarquitectura[10], modelado en general, animación de personajes, efectos y partículas, simulaciones físicas de cuerpos rígidos y blandos. Simulación de pelos y telas. Es un software comercial, no tiene versión de aprendizaje gratuita. Es utilizado por multitud de importantes estudios de efectos visuales en combinación con RenderMan, el motor de render

9 Ver Capítulo 2
10 La infoarquitectura es una rama de la arquitectura que pretende mostrar en formato 3d, con acabados realistas, como será un proyecto de obra cuando esté finalizado.

fotorrealista de Pixar. Última versión a diciembre de 2008: Maya 2009.

➢3D Studio Max: Software reconocido por su infoarquitectura y su uso en videojuegos. Fue originalmente escrito por Kinetix (una división de Autodesk) como el sucesor de 3D Studio para DOS. Más tarde Kinetix se fusionaría con la última adquisición de Autodesk, Discreet Logic. La versión a diciembre de 2008 era la 2009. Es el líder en el desarrollo 3D de la industria del videojuego y es muy utilizado a nivel amateur.

➢Lightwave 3D: Fue originalmente desarrollado por Amiga Computers a principios de la década de los 90. Más tarde evolucionó en un avanzado paquete gráfico y animación 3D. Actualmente disponible para Windows, Mac OS[11] y Mac OS X[12]. La versión a finales del 2008 era la 9.5. El programa consiste en dos componentes: el modelador y el editor de escena. Es utilizado en multitud de productoras de efectos visuales como Digital Domain.

➢Softimage XSI: Este es un software comercial y cuenta con una versión de aprendizaje gratuita (XSI Mod Tool). En 1987, Softimage Inc, una compañía situada en Montreal, escribió Softimage|3D, que se convirtió rápidamente en el programa de 3D más popular de ese período. En 1994, Microsoft compró Softimage Inc. y comenzaron a reescribir SoftImage|3D para Windows NT. El resultado se llamó Softimage|XSI. En 1998 Microsoft vendió Softimage a Avid. La versión a finales del 2008 era la 7.01.

➢Cinema 4D: Factible para infoarquitectura, modelado en general, animación de personajes, efectos y partículas, simulaciones físicas de cuerpos rígidos y blandos. Simulación de pelos y telas. Permite adquisición por módulos. Software comercial.

➢Blender: Programa de creación de contenido 3D que abarca desde el modelado y animación hasta la composición y renderización de complejas escenas en 3D. Es software libre, y cuenta con características como soporte para programación bajo Python con una amplia gama de script[13] en constante desarrollo, posee un motor robusto para la programación de juegos, un

11 Mac OS (del inglés Macintosh Operating System, en español Sistema Operativo de Macintosh) es el nombre del sistema operativo creado por Apple para su línea de computadoras Macintosh.

12 Mac OS X es un sistema operativo desarrollado y comercializado por Apple Inc. que ha sido incluido en los sistemas Macintosh desde 2002.[2] [3] Es el sucesor del Mac OS 9.

13 Un script (cuya traducción literal es guión) o archivo de órdenes es un programa usualmente simple, que

motor de render propio y una comunidad de usuarios totalmente abierta y dispuesta a colaborar.

Blender es en el presente la suite 3D más completa y potente que existe gratuitamente frente a las costosas herramientas privativas como Houdini, Lightwave, Softimage XSI, Maya, Cinema 4D y 3D Max que se usan en la producción profesional de materiales audiovisuales. Blender ha sido usado filmes, cortometrajes y spots publicitarios como: *"Spiderman 2", "Friday or Another Day", "Elephant Dreams", "Big Buck Bunny", "Plumíferos", "Soft Boy",* entre otros. Según las estadísticas publicadas en abril del 2009, las descargas desde el sitio http://www.blender.org sobrepasan los 3.4 millones anualmente. (Plohman 2009)

1.2.2 Principales motores de render

El motor de render puede estar integrado en el propio software 3D o externo a este (como plugin[14]). Su función es renderizar los frames o sección de frame mediante el uso de los métodos de renderizado que tienen implementados.

- **Métodos de Render**

El proceso de render es el encargado de convertir la descripción de una escena tridimensional en una imagen bidimensional. Durante dicho proceso, se determinará el color de cada uno de los píxeles que forman parte de la imagen resultado.

A continuación se proporcionará una breve descripción de algunos de los métodos de renderizado.

Raytracing: El *raytracing* (trazado de rayos en español) es un algoritmo para síntesis de imágenes tridimensionales. Propuesto inicialmente por Turner Whitted en 1980, está basado en el algoritmo de determinación de superficies visibles de Arthur Appel denominado Ray Casting (1968).

En el algoritmo Ray Casting se determinan las superficies visibles en la escena que se quiere

generalmente se almacena en un archivo de texto plano.

14 Plugin (del inglés "enchufable") es una aplicación que se relaciona con otra para aportarle una función nueva y generalmente muy específica.

sintetizar trazando rayos desde el observador (cámara) hasta la escena a través del plano de la imagen. Se calculan las intersecciones del rayo con los diferentes objetos de la escena y aquella intersección que esté más cerca del observador determina cuál es el objeto visible.

El algoritmo de trazado de rayos extiende la idea de trazar los rayos para determinar las superficies visibles con un proceso de sombreado (cálculo de la intensidad del píxel) que tiene en cuenta efectos globales de iluminación como pueden ser reflexiones, refracciones o sombras arrojadas.

Para simular los efectos de reflexión y refracción se trazan rayos recursivamente desde el punto de intersección que se está sombreando dependiendo de las características del material del objeto intersectado.

Para simular las sombras arrojadas se lanzan rayos desde el punto de intersección hasta las fuentes de luz. Estos rayos se conocen con el nombre de rayos de sombra (shadow rays).

En la actualidad, el algoritmo de trazado de rayos es la base de otros algoritmos más complejos para síntesis de imágenes, que son capaces de simular efectos de iluminación global complejos como la mezcla de colores (color bleeding) o las cáusticas.

Radiosidad: La radiosidad es un conjunto de técnicas para el cálculo de la iluminación global, (ver abajo) que tratan de resolver, el problema básico de la renderización de la forma más realista posible, en el campo de los gráficos 3D por computadora. Dicho problema es:
El transporte de la luz sólo se puede modelar de forma óptima considerando que cada fuente luminosa emite un número enorme de fotones[15], que rebotan al chocar contra una superficie describiendo una cantidad de trayectorias imposibles de simular en un computador.

Una de las técnicas empleadas en el cálculo de la radiosidad es el Método de Monte Carlo[16] para resolver este problema mediante números aleatorios y de forma estadística.

El auge de la radiosidad y otros métodos eficientes de renderización han posibilitado un auge en la infografía, siendo muy habitual encontrar por ejemplo películas que aprovechan estas

15 Para muchos científicos la luz es un conjunto de partículas y a estas las denominan "fotones".
16 El método de Monte Carlo es un método estadístico numérico usado para aproximar expresiones matemáticas complejas y costosas de evaluar con exactitud.

técnicas para realizar efectos especiales.

Global Illumination: *Global Illumination* (iluminación global en español) es un nombre general para un grupo de algoritmos utilizados en los gráficos por ordenador en 3D que se pretende añadir una iluminación más realista a las escenas 3D. Estos algoritmos tienen en cuenta no sólo la luz que viene directamente de una fuente de luz (iluminación directa), sino también a los casos en los que los rayos de luz de la misma fuente se reflejan en otras superficies en la escena (iluminación indirecta).
En teoría, las reflexiones, las refracciones, y las sombras son todos ejemplos de iluminación global, porque cuando los simulas a ellos, un objeto afecta a la representación de otro objeto (a diferencia de que un objeto se vea afectado solamente por una luz directa).

Las imágenes renderizadas usando algoritmos de iluminación global a menudo parecen más realistas, que las imágenes renderizadas usando solamente los algoritmos de iluminación directa. Sin embargo, esas imágenes son computacionalmente más costosas y por lo tanto mucho más lento de generar. Un enfoque común es calcular la iluminación global de una escena y almacenar esa información con la geometría generada por la radiosidad. Estos datos almacenados pueden ser utilizados para generar imágenes desde diferentes puntos de vista de una escena sin tener que pasar a través de cálculos de iluminación varias veces.

Entre los principales motores comerciales y *Open Source* se encuentran:

- **Motores Comerciales**

Mental Ray (incluido en 3Ds MAX, Viz,Maya, XSI).
Mental Ray es un programa de render, desarrollado por Mental Images en Berlín (Alemania). Este motor de render (realmente un plugin de otros programas de 3D), es uno de los pocos software desarrollados en Europa que compiten con los desarrollados en EEUU o Canadá.
Mental Ray tiene un motor muy completo, que calcula desde sombras simples hasta

segmentadas[17], compilación fotónica emitida por luces conocido como "Photon Map"[18] y cuenta con su propio sistema de aceleración que le permite hacer algunas cosas en tiempos bastante rápidos para el nivel de fotorrealismo que maneja.(Mental Images GmbH 2009)

V-Ray:

Desarrollado por la compañía Chaos Group (Bulgaria) es un renderizador basado en tecnología que implementa Raytracing de uso extendido en visualización arquitectónica, permite generar simulaciones físicamente correctas de los efectos de iluminación, incluida la reflexión y refracción, efectos cáusticos e iluminación global.

Plugins de 3D Studio Max, especialmente usado para arquitectura, exteriores e interiores. Gran calidad pero a costa de un tiempo elevado de render. Actualmente disponen de una versión para Maya pero esta aún no es estable. Extraordinarios shader[19], calidad excelente en 3D Max. (Chaos Group 2009)

FinalRender

Final Render es un sistema avanzado desarrollado por Cebas (2002). Permite emular la dispersión de la luz.

Plugins de 3D Studio Max, especialmente usado para arquitectura, exteriores e interiores. Gran calidad pero a costa de unos tiempo elevados de render. Actualmente disponen de una versión para integrar en Maya, bastante avanzada. Con este motor se renderizó el filme 2012 (Ver Figura 3 Anexos) del director Roland Emmerich. (Cebas VISUAL TECHNOLOGY Inc. 2009)

➢**Motores Open Source**

Luxrender

17 Partido en segmento, seccionado.

18 Método que se construye a partir de las trayectorias de los fotones individuales emitidos por cada fuente de luz

19 Los shaders son utilizados para realizar transformaciones y crear efectos especiales, como por ejemplo iluminación, fuego o niebla.

Con base en el estado de los algoritmos de arte, LuxRender simula el flujo de luz de acuerdo a las ecuaciones físicas, lo que produce imágenes realistas de calidad fotográfica.

LuxRender es Software Libre - tanto para uso personal y comercial - y está licenciado bajo la GPL. El programa se ejecuta en Windows, Mac OS X y Linux. Plenamente exportadores funcionales están disponibles para Blender y Maya, mientras exportadores para un número creciente de aplicaciones 3D están en desarrollo. (LuxRender 2009)

Yafray

YafRay (*Yet Another Free Raytracer* en inglés) es opensource y multiplataforma que surge como proyecto personal de Alejandro Conty Estévez, miembro de Co.A.L.A, en Julio del 2002. Inicialmente se trataba de un trazador de rayos básico para el sistema operativo GNU/Linux. Posteriormente se le fueron añadiendo características, una de las más importantes fue el cargador de escenas en un formato propio utilizando XML. De este modo fue posible la exportación de escenas desde programas de modelado para la posterior generación de la imagen utilizando YafRay. (Wikipedia 2009)

Yaf(a)ray

YafaRay es un motor de render Open Source libre. El trazado de rayos es una técnica de representación para generar imágenes realistas trazando el camino de la luz a través de una escena 3D.

Un motor de procesamiento consta de un programa "sin rostro", equipo que interactúa con una aplicación 3D de acogida específicas para proporcionar las capacidades de trazado de rayos "en demanda". Blender 3D es la aplicación *host* de YafaRay. YafaRay está bajo la licencia LGPL 2.1. (ALTASIS 2009)

En la actualidad muchos son los motores de renderizado que se ofrecen en el mercado, ya sea gratuitamente o pagando con anterioridad los derechos de la licencia correspondiente. Estos caracterizan la realidad de estos tiempos, matizada por un ascenso en la complejidad de las producciones cinematográficas, que cada día requieren que sean más superiores y versátiles.

Para el conocimiento de otros motores de render tanto comerciales como Open Source consultar el **Anexo 1**.

1.2.3 Sistemas de Gestión de Renderizado Distribuido

Un SGRD, es una agrupación de varias computadoras que se reparten el trabajo de **renderizar** las imágenes o animaciones realizadas con un programa de infografía.

Las imágenes de una película de animación o de un cortometraje poseen varios **frames**, la sucesión de las mismas hacen la escena completa. Un SGRD divide el trabajo entre varios ordenadores para que el tiempo de render sea el menor posible.

Esta es una práctica muy conocida en la industria del cine 3D y, es difícil imaginarse que una sola computadora pueda generar tantas imágenes por si sola sabiendo que cada frame (casos como Pixar) tarda 6 horas aproximadamente en generarse y que para hacer un segundo de animación son necesarios de 24 a 30 frames. El tiempo de render es promedio, ya que en películas como “Los Increíbles” algunos frames tardaron hasta 90 horas en generarse.

➢**Render Distribuido**

Al instalarse un SGRD en el gestionador de rendererizado debe existir un algoritmo que permita distribuir a las PC *slave* (esclavas) el trabajo que le tocará renderizar a cada una.

Este “tipo” de distribución de renderizado puede clasificarse en 2 formas:

1. Render distribuido de Granularidad Gruesa
2. Render distribuido de Granularidad Fina.

Granularidad Gruesa: Es donde cada frame de la animación se envía a un procesador para ser renderizado (Ver Figura 4 Izquierda). También se conoce como granularidad *interframe.* (José Angel Mateos Ramos 2007)

Granularidad Fina: Es cuando se divide cada frame en fragmentos más pequeños y estos son

enviados a distintos procesadores para que sean renderizados de forma independiente (Ver Figura 4 Derecha). También se conoce como granularidad *intraframe.* (José Angel Mateos Ramos 2007)

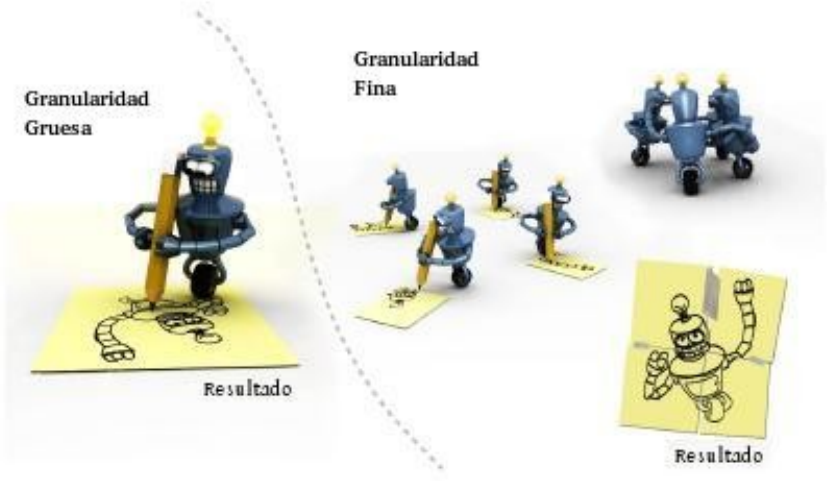


Figura 4: *Diferencia entre granularidades*

Para cada tipo de granularidad existen distintos tipos de algoritmos. Ver Figura 5.

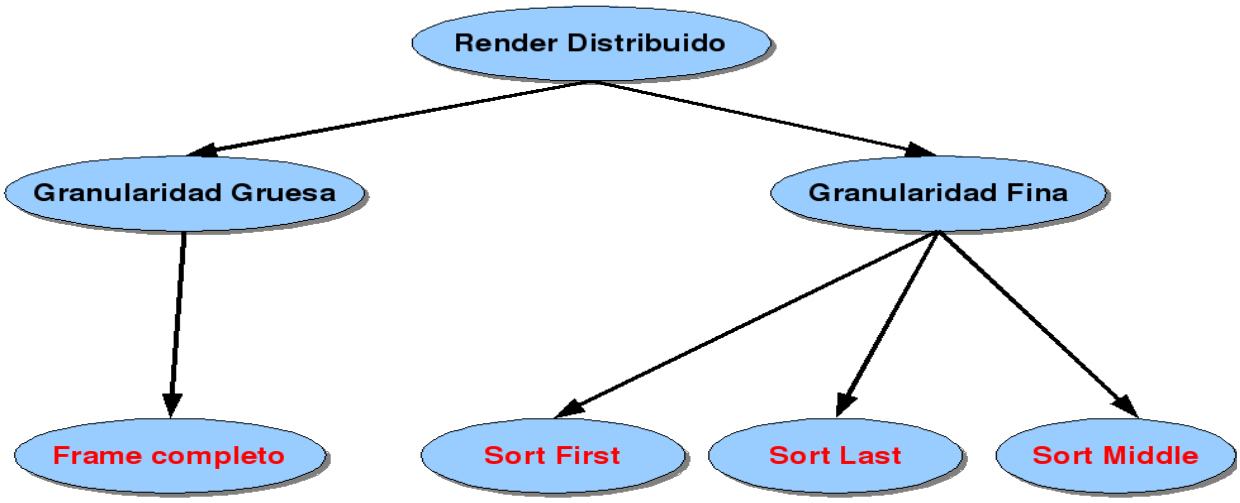


Figura 5: *Ejemplo de granularidades*

➢ **Frame Completo:**

Es donde un frame completo se va a renderizar en cada PC, o sea, varios frames pueden ser procesados debido a que hay varias PC.

Ventaja: La principal ventaja es la calidad ya que al dividir una imagen puede crear un margen de error.
Desventajas: Demora en tiempo pues existen frames los cuales por su contenido tienden a demorar hasta días.

➢ **Sort Middle:**

El procesamiento de las primitivas gráficas[20] se divide por igual entre los procesadores, mientras el procesamiento de píxeles se divide entre los procesadores de rasterización[21] según se superponen con el espacio de los recuadros de la pantalla.

Ventaja: Bueno para sistemas de PC con alto ancho de banda.
Desventaja: Difícil para un sistema de clúster pues requiere alto rendimiento de la red.

➢ **Sort First:**

En Sort-First, el espacio de pantalla se divide por regiones, las primitivas gráficas que pertenecen a cada región se clasifican según su ubicación (Ver Figura 6). 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.

20 Ejemplo de primitivas gráficas son los puntos, los segmentos de recta, las circunferencias, los polígonos, etc

21 La rasterización es el proceso por el cual una imagen descrita en un formato gráfico vectorial se convierte en un conjunto de píxeles o puntos para ser desplegados en un medio de salida digital, como una pantalla de computadora.

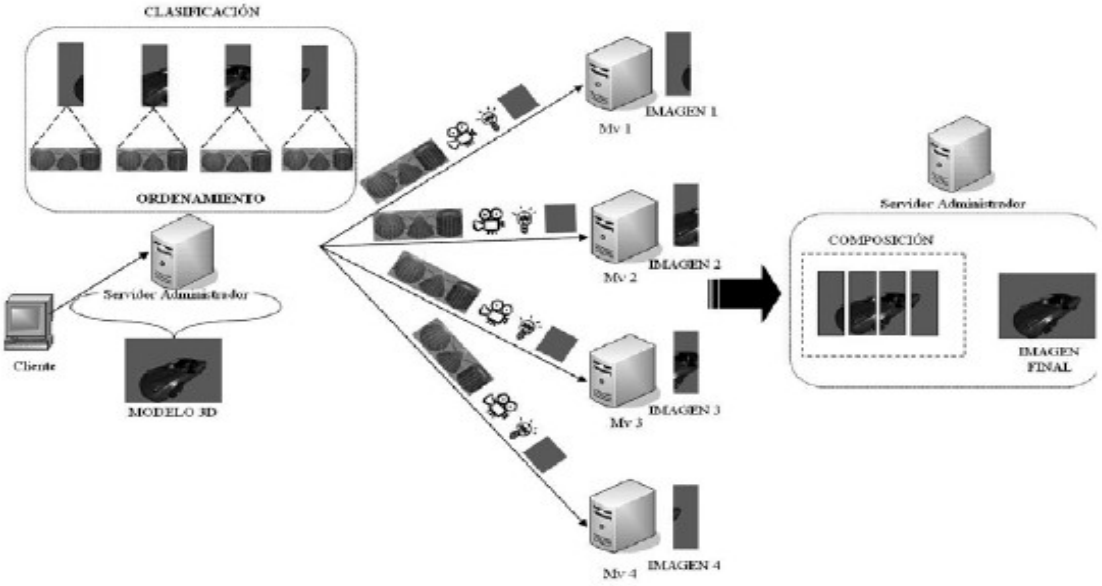


Figura 6: *Sort First*

Ventaja: Necesidad de comunicación relativamente pequeña. Toma ventaja en la coherencia de frame, por lo que es potencialmente atractiva en aplicaciones de animación.
Desventaja: Trabajo adicional que debe hacerse para transformar las primitivas, calcular una adecuada partición de espacio. Desequilibrios de carga debido a la desigual distribución. Tiene poca escalabilidad a medida que aumentan los procesadores.

- **Sort Last**

En el sistema Sort-Last, cada procesador hace una imagen separada que contiene una porción de las primitivas gráficas, y después las imágenes resultantes son mezcladas (con las comparaciones de profundidad) en una sola imagen para su visualización. Ver Figura 7.

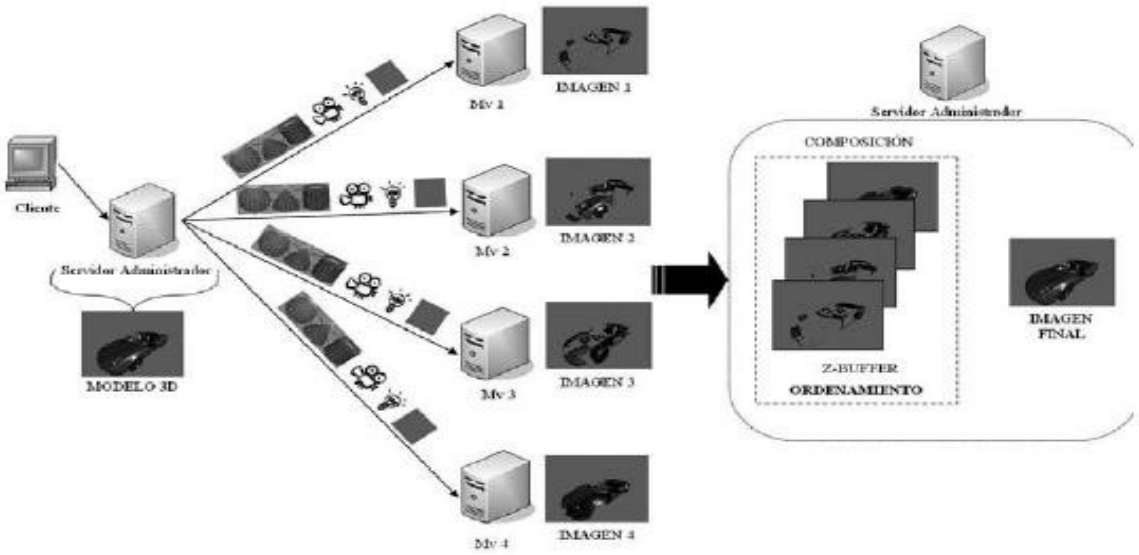


Figura 7: *Sort Last*

Ventaja: La principal ventaja es su escalabilidad. Dado que cada primitiva gráfica es renderizada por exactamente un procesador, el factor de superposición es siempre.

Desventaja: El principal inconveniente es que normalmente requiere de una red de composición de imágenes con gran ancho de banda y capacidad de procesamiento para apoyar la composición de imágenes de fondo se superponen. Se incurre en la latencia.

Teniendo en cuenta que la red de los laboratorios no es de un gran ancho de banda y que muchas computadoras tienen solo 256 de RAM se llega a la conclusión de que el algoritmo de granularidad fina acorde a usar como método para la renderización es el Sort-First. De contarse con PCs más potentes (1 GB RAM) y no siendo mucha la complejidad los frames a renderizar puede usarse también la granularidad gruesa, o sea, el envío del frame completo.

Sistemas de Gestión de Renderizado Distribuido Comerciales

Backburner (incluido con 3D Max)

Backburner es el gestor de cola (*queue manager*) de Autodesk para la transformación de fondo y el procesamiento distribuido en la red. Proporciona los medios para enviar, monitorizar y controlar la renderización de trabajos. Backburner se introduce en las siguientes

aplicaciones de Autodesk:

• Inferno® • Flint® • Smoke® • Lustre® • Toxik™ • Cleaner®

• Flame® • Fire® • Backdraft® • 3ds Max® • Combustion® • Burn™

(Autodesk 2009)

Virtual Vertex Muster

Muster 6 es Virtual Vertex, nueva generación de gestionador de granja de render para la producción de películas, el desarrollo de juegos, y la industria del efecto visual. Construido en la parte superior de estado de las tecnologías procedentes de diez años de desarrollo y pruebas de tensión. Muster 6 maximiza su granja de render y le permiten delegar la vigilancia de los recursos a un entorno de automatización completa. (Virtual Vertex 2009)

Deadline

DEADLINE® es libre de problemas de administración y renderiza con una caja de herramientas para Windows y Mac OSX. Basado en las granjas de render de Linux ofrece un ambiente de flexibilidad y una amplia gama de opciones de gestión de granjas de render de todos los tamaños. Creado por la Prime Focus. (Prime Focus 2010)

RenderPal

RenderPal V2, es el sistema de render profesional de más de 25 renderizaciones y composiciones de aplicaciones, como Maya, Mental Ray, XSI, 3D Studio MAX, Nuke, Cinema 4D, Shake, After Effects, Realflow, Lightwave y muchos más. (Shoran Software 2010)

Maya Satellite

El plugin mental ray que se integra en Maya siempre le permite hacer en un máximo de 4 CPUs locales. Con Maya Complete, de representación Mental Satellite de rayos puede tener lugar el 2 de CPU adicionales a distancia. Con Maya Unlimited, de representación Mental Satellite de rayos puede tener lugar el 8 de CPU adicionales a distancia. Este gestionador permite distribuir cada fotograma a través de las máquinas de la red. (Alias Systems Corp

2010)

ScreamerNet

El Contralor ScreamerNet para Mac OS X, desarrollado por J. David Baker, de Catalyst Productions, es un controlador front-end[22] para Lightwave ScreamerNet en el sistema de renderización por red. Después de haber construido sus escenas en Lightwave, puede utilizar el controlador ScreamerNet para alimentar a la escena a medida que los CPUs serán de distancia, mientras que se puede trabajar en otra escena en Lightwave. (J. David Baker 2010)

➢**Sistemas de gestión de Renderizado Distribuido Libres.**

Se sometieron a pruebas de usabilidad (evaluando la facilidad de uso) y estabilidad (monitoreando el comportamiento dentro del Sistema Operativo para detectar posibles bugs o inconsistencias) a los siguientes programas: Farmerjoe, Distriblend, Yadra, LOKI Render, TrueFarm y DrQueue. (David Padrón Alvarez 2009)

Farmerjoe

Farmerjoe es un sistema de renderizado distribuido para Blender, lo hace tanto en el frame basado en la distribución y el cubo de base (solo frame), la distribución, tiene una interfaz gráfica de usuario web y es bastante fácil de configurar. (Formworks 2010)

Distriblend

Distriblend es una aplicación Java diseñado para establecer y administrar una granja de renderizado para Blender. Está escrito en Java y, como tal, el entorno de ejecución Java es necesario. Aparte de tener instalado Blender, este es el único requisito para la creación de una granja de renderizado con Distriblend.

El sistema funciona sobre una variedad de plataformas - siempre que sea Java puede ser instalado. Ha sido probado principalmente en Windows, pero ha sido utilizado por muchos en

22 Parte del software que convierte los datos a un formato específico de un hardware o de otro software.

Linux y otras plataformas. (Sourceforge.net 2010)

Yadra

Oliver Schulze escribió acerca de su herramienta que le permite hacer animaciones Blender a través de una red de muchos equipos, destacándose su fácil configuración y funcionamiento estable.

Esta es de código abierto y escrito en Java. Se puede obtener información sobre la instalación y uso en la wiki. Una de sus características interesantes es la herramienta web de gestión, para tener un registro de su nueva granja de render. (Oliver Schulze 2010)

Loki Render

Loki Render distribuye las imágenes de Blender 3D (frames) a través de varios ordenadores, reduciendo así el tiempo total necesario para la representación. El renderizado distribuido es particularmente útil cuando muchas imágenes deben ser renderizadas (animación, por ejemplo) o cuando tenemos un complejo de imagen fija, se puede dividir la imagen en varias partes, hacer que cada parte en un equipo diferente de la granja, a continuación, volver a montar el partes en una imagen final.

Loki Render utiliza archivos de proyecto Blender como entrada, y utiliza el motor de render interno de Blender o Yafray para la salida. Apoyo a otros insumos y motores de render se espera en el futuro. (Loki Render 2010).

1.2.4 DrQueue

DrQueue es una herramienta de software de código abierto utilizado para administrar una granja de renderizado. Proporciona render distribuido en función de cada cuadro y la gestión de estas tareas. Se utiliza principalmente para las animaciones y efectos visuales en el cine y la publicidad. DrQueue está licenciado bajo la GPL versión 3. (DrQueue 2010).

De todos los gestionadores a los cuales se le realizaron pruebas en el proyecto resultó ser

DrQueue el que brindó los mejores resultados (David Padrón Alvarez 2009). Por sus potencialidades para lograr productos comerciales de alta calidad, se propuso su utilización en una granja de render de la UCI.

Características de DrQueue:

- Es Open Source y GPL.
- Está programado en C.
- Puede ser usado en los Sistemas Operativos: FreeBSD, IRIX, Linux, Mac OS, Windows.
- Soporta los principales motores de render: Maya Software, Mental Ray, Blender Internal, XSI,Lightwave, Mantra (en Houdini), Turtle (Illuminate Labs), BMRT, Shake, After Effects, Aqsis, Nuke, Terragen, 3Delight y Pixie.
- Posee enlaces Ruby (Comunidad Ruby 2010) y Phyton (Python Software Foundation 2010)
- Tiene funcionalidades de administración y control avanzadas.
- Es escalable.
- Incluye interfaz web alternativa.

Requisitos de hardware mínimos recomendados para DrQueue

No hay requisitos de hardware especiales para DrQueue, solo se necesitan equipos que utilicen como protocolo de comunicación TCP/IP. Estos equipos requieren unos pocos megabytes de RAM y espacio en disco (DrQueue Master y esclavo). Para una mejor renderización es importante hacer que las máquinas donde los esclavos se están ejecutando

tengan algún poder de la CPU, siendo recomendable la utilización de tarjetas gráficas, sobre todo en aquellos casos en los que la escena a renderizar y los efectos deseados sean más complejos y realistas.

Algunas de las empresas, instituciones y usuarios que han trabajado con DrQueue (DrQueue 2010), se relacionan en la siguiente lista:

- Martian Labs (Linux/IRIX farm running Houdini & other general computation)
- NASA Goddard Space Flight Center
- Bren Entertainment (Filmax & Filmax Animation)
- Swedish Film Effect
- Indiana University (Advanced Visualization Lab)
- Gener8Xion Entertainment, Inc.
- Christopher R Green
- Verint Systems Canada Inc.
- New Machine Studios (Mac OS X and Suse Linux Server Farm Running Maya)
- Basecamp VFX
- Leyenda|VFX Producciones
- Alan Jones (who also provided XSI support and XSI-Splash)
- Ainkaboot Ltd UK (Clustering solutions)
- CRAVE (DBIT, Mumbai)
- Digital Graphics (Belgium)
- University of Wisconsin - Parkside (Animation Concentration)
- preset.de (Linux farm running Houdini)

◆Hochschule Wismar - University of Technology, Business and Design

◆The Monk Studio, Thailand (Linux farm running Maya and 3delight)

Producciones realizadas con DrQueue

- "One night with the King" - (by Gener8Xion Entertainment, Inc.)
- "Elephants Dream" - (by Orange Open Movie Project)
- "Pirates of the Caribbean: Dead Man's Chest" - Martian Labs
- "Exorcist, The Beginning" - Martian Labs
- "Bee Season" - Martian Labs
- "Dark City" (HD DVD) - Martian Labs
- "Harsh Times" - Martian Labs
- "Santa Clause 3" - Martian Labs
- "Friday or another day" / "Vendredi ou un autre jour" - (by Digital Graphics (Belgium))
- "Donkey Xote" - (by Bren Entertainment)
- "The Hairy Tooth Fairy" (Spanish title: "Pérez, el ratoncito de tus sueños" - (by Bren Entertainment)
- "Nocturna" - (by Bren Entertainment)
- A slurry of commercials - Toyota, Hummer, Nike, Tylenol, Sketchers, Seaworld, Goodyear, and others (by Martian Labs)

Arquitectura de DrQueue

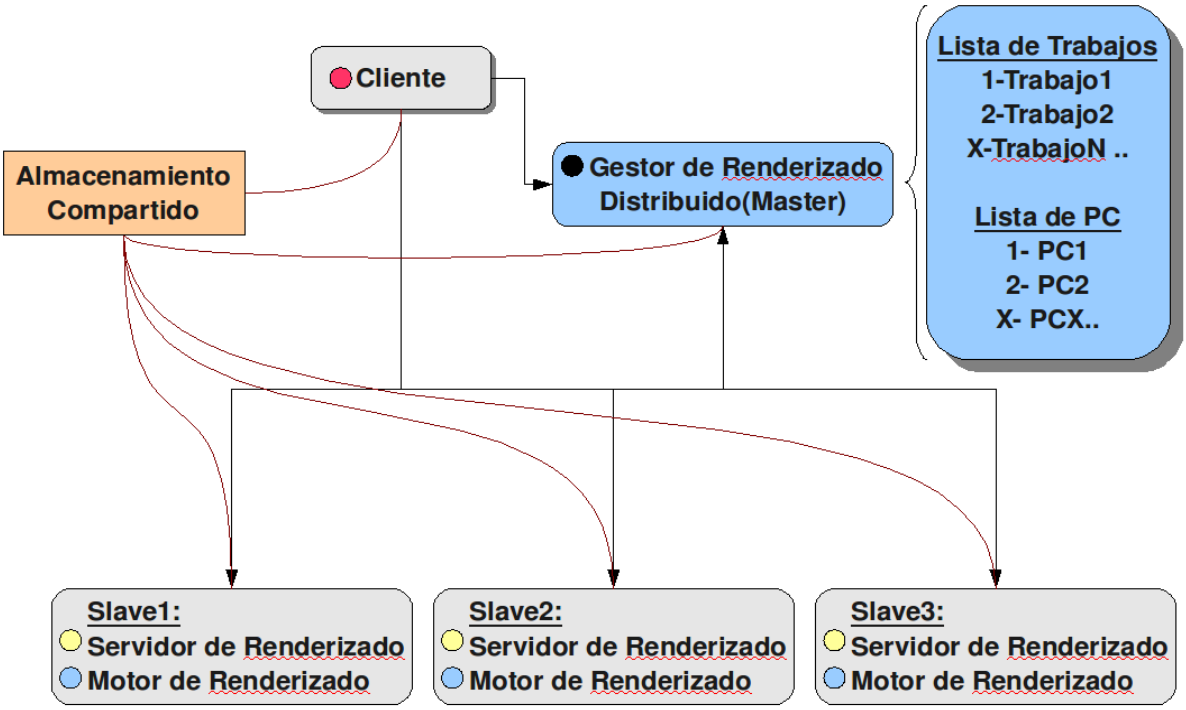


Figura 8: *Arquitectura DrQueue*

En la Figura 8, se puede apreciar la existencia de un Master (que contiene la lista de trabajos y los esclavos), un número de esclavos (slave), el cliente y luego un almacenamiento compartido que podría ser necesario por todos o algunos de ellos.

Este almacenamiento compartido no es parte de DrQueue en sí mismo. El almacenamiento compartido tiene que ser administrado por el sistema operativo subyacente como cualquier tipo de sistema de archivos de red (NFS, CIFS/samba, etc). Puede haber muchos almacenamientos compartidos, no uno solo. DrQueue se basa en el sistema operativo, por lo que puede tener tantos y de tantos tipos como el sistema operativo pueda manejar.

Componentes de DrQueue

Master

El Master se pregunta por los esclavos y por los clientes. Los esclavos también actualizarán su información de estado propio (puestos de trabajo en ejecución, promedio de carga, CPUs disponibles, ...). El Master tiene toda la información sobre los trabajos y los esclavos por lo que debe estar corriendo todo el tiempo.

Hay muchos tipos de peticiones que pueden ser emitidas al Master. Por ejemplo, los clientes pueden solicitar la lista de trabajos almacenados en ella, la lista de los esclavos, los detalles del trabajo de una u otra y así sucesivamente.

Este Master tendrá un puerto de red TCP/IP abierto para recibir las solicitudes. Todas las solicitudes (también las enviadas a los esclavos) son no autenticados por lo que el puerto no debe estar abierto a zonas de no confianza.

El Master accederá al almacenamiento compartido para guardar los logs y que estos puedan ser leídos y analizados desde cualquier lugar con acceso a esa ubicación.

Esclavos

Cada esclavo tiene su propia información, el estado, los límites, las tareas y periódicamente actualiza al Master.

Este cuando este inactivo y esté disponible para la renderización le pedirá al Maestro una tarea. Una vez que la tarea se asigna al esclavo por el maestro, el esclavo recibirá toda la información relativa al trabajo y la tarea (la ubicación del script de trabajo (jobscript), el tipo de trabajo, tarea/el número de frame, etc).

Con esa información, el esclavo se creará un nuevo proceso con las variables de entorno establecidas y ejecutará directamente el jobscript señalado por ese trabajo. El jobscript tiene que estar situado en cualquier lugar en el almacenamiento compartido para que todos los esclavos puede leerlo en la misma dirección.

Ese proceso de nueva creación comenzará a correr y almacenará el archivo de log de tareas en el directorio de registros de DrQueue. Mientras que el proceso no ha terminado, el esclavo periódicamente comprobará su existencia para confirmar que no ha muerto de forma anormal.

Cuando termina el proceso del esclavo se envía toda la información para el Master. Este recibirá la tarea que ha finalizado la ejecución y el estado de salida. Dependiendo de los valores que la tarea podría ser requeued (volver a enviar el frame) automáticamente, marcado como "final" o "Error". Esa información estará disponible para el cliente. Los frames una vez renderizados se guardan en el almacenamiento compartido.

➔El proceso de Render en el Esclavo.

El esclavo crea un nuevo proceso hijo que ejecutará el jobscript, se debe considerar que el jobscript será ejecutado localmente en ese esclavo y así todas las variables de empleo han de ser válidos a nivel local. Esto incluye las direcciones, los ejecutables que el script puede ejecutar (como el ejecutable de motor de renderizado adecuado), plugins, texturas, escenas, directorios de salida, etc. Todo lo que podría ser necesario para el éxito del render tiene que estar disponible para cada nodo en ese mismo lugar. Esa es la razón principal del almacenamiento compartido. Visto que, toda la información guardada no estará disponible para los esclavos y será capaz de ejecutar el jobscripts sin ningún problema.

En un entorno de plataformas cruzadas de renderización, el jobscript será responsable de "transformar" o "traducir" los valores que recibe para ese sistema operativo de esclavos y las necesidades de configuración.

Clientes

Los clientes son todos los otros programas que podría solicitar para recibir información acerca de la renderización o enviar solicitudes de otras para realizar diferentes acciones como volver a renderizar el frame, habilitar esclavos, eliminar trabajos y así sucesivamente.

Drqman (Ver Figura 5 Anexos) es uno de esos clientes, pero hay otros muchos como todas las herramientas de línea de comandos (sendjob, jobinfo, ctask, etc), así como cualquier otro programa que pueda utilizar la biblioteca DrQueue (libDrQueue), como scripts python usando el módulo DrQueue (por ejemplo DrKeewee).

Los clientes pueden realizar todas las acciones que están disponibles en la biblioteca de DrQueue. No hay ninguna restricción sobre el tipo de solicitudes que podría ser realizada por cualquier cliente.

➔**Como funciona:**

De un modo genérico, DrQueue es una herramienta para distribuir las tareas a través de una granja de ordenadores en red. Hay una cola de trabajos que están compuestas por un número diferente de las tareas representadas por números.

Desde el punto de vista de renderizado distribuido, las tareas suelen representar los frames, y el número de tareas es también el número de frame.

Por lo tanto, todos los fotogramas que componen los diferentes puestos de trabajo (mediante script de distintos puestos de trabajo) están en la cola esperando para ser asignados a un esclavo. Una vez asignado, el esclavo se ejecuta el script de trabajo para producir el marco resultante. Después de eso, la tarea será marcada como realizada y el esclavo se liberará para renderizar otros frames.

Desventajas de DrQueue

- El cliente y el Master están físicamente en la misma PC, de lo que se deriva que:
  - No puede gestionar los trabajos de varios usuarios.
  - Cada usuario tendría que instalar el Master en su máquina.
- La estrategia que sigue se basa en ordenar el renderizado de un frame por cada esclavo, por lo que no aplica ninguno de los métodos de fragmentación de frame conocidos.

- Realiza una gestión ineficiente de los clientes.
- Solo es capaz de gestionar 100 computadoras.
- Complicada gestión de los trabajos a renderizar.
- Presenta problemas al ser instalado en Windows.

1.3 Principales compañías que ofrecen servicio de renderización:

Las superproducciones de secuencias 3D y efectos especiales sólo pueden lograrse utilizando granjas de render. Existen muchas compañías por todo el mundo que se han dedicado a brindar el servicio de renderización. Entre las principales por sus prestaciones, precios y potencia de render se encuentran:

PurePowua

Compañía italiana dedicada a la renderización. El Renderizado lo realizan con LuxRender 0.6rc4. Actualmente se encuentran mejorando sus servicios. Las próximas mejoras son: mejor usabilidad, creación de cuentas y selección de zona horaria, implementación de un mecanismo para en caso de existir interrupción de la conectividad, posibilidad de comprar PowuaCredits a través de PayPal, distintos de los anteriormente disponibles como Visa, Mastercard y transferencia bancaria.

Powua Remote Desktop optimizado: todas las aplicaciones Powua puede ser lanzadas fácilmente de la barra de tareas o de una ventana de terminal. (Powua 2010)

RebusFarm

La Granja de Render REBUS ofrece 250 computadoras para hacer sus imágenes fijas y animaciones. No importa la aplicación 3D que esté utilizando.

Tiene un servicio de renderizado rápido, seguro y fácil, gracias a una tecnología única, el Software Farminizer desarrollado por la granja de renderizado REBUS. El Farminizer es una

pieza muy robusta de software en el trabajo con 3D Studio Max, Cinema 4D, MODO, Lightwave y Maxwell. Comprueba automáticamente la integridad y compatibilidad de los datos. (Rebus 2010)

Render Nation

Render Nation, ubicada en el Reino Unido. Ofrecen un servicio completamente gestionado y dedicado a asegurarse de que se obtienen los resultados esperados a tiempo. Utilizan las últimas tecnologías de la informática. Son la única granja de render remota en el mundo. (Render Nation 2010)

Es de destacar la compañía **Data Center** de Weta Digital con la que se ha renderizado Avatar, El Señor de los Anillos y X-Men entre otras. La compañía tiene a su disposición 4,000 HP BL2×220c blades y una red de 10 Gigabits. Todo el sistema esta ocupando el rango entre los números del 193 al 197 entre los 500 sistemas de supercomputadores del mundo. Son 32 máquinas con 40.000 procesadores y 104 Terabytes de memoria. Para la renderización de Avatar (Ver Figura 6 Anexos) según uno de sus administradores afirmó que utilizaron Ubuntu como sistema operativo.

Para el conocimiento de otras compañías que se dedican al renderizado ver **Anexo 2**.

A lo largo de este capítulo se ha expuesto un estudio acerca de las etapas de producciones 3D, donde pudo notarse la importancia que tiene el renderizado por lo costoso que resulta en cuanto a consumo de tiempo y de cómputo, lo que hace necesario su optimización. Se realizó un estudio de las posibles formas de optimización demostrando ser el uso del renderizado distribuido la más admisible.

También se exponen los principales software de infografía, motores de render y gestionadores de render observándose como la mayoría pertenecen al software privativo. No obstante puede notarse que para proyectos que no cuentan con grandes recursos va surgiendo también una

alternativa en el Software Libre.

DrQueue ha demostrado ser la mejor alternativa de SGRD “libre” para la UCI. Este aún presenta ciertos problemas que deben ser resueltos, no obstante por pruebas realizadas ha comenzado a demostrar su competencia.

Cuba aunque no cuenta con el hardware que poseen las grandes compañías de renderizado del mundo si podría formar parte de esta industria. Podría ser una alternativa para el renderizado de producciones realizadas en los países del tercer mundo, principalmente latinoamericanos.

CAPÍTULO 2: DISEÑO DE ARQUITECTURA PARA SISTEMA DE GESTIÓN DE RENDERIZADO DISTRIBUIDO.

El objetivo principal del grupo de proyecto FreeViUX es la creación de audiovisuales utilizando Software Libre y esto hizo evidente la necesidad de un sistema de renderizado distribuido. De todos los gestionadores a los cuales se le realizaron pruebas en el proyecto resultó ser DrQueue el que brindó los mejores resultados y por el reconocimiento que ha tenido en todo orbe y sus potencialidades para lograr productos comerciales de alta calidad, se propuso su utilización.
Como todo sistema distribuido su arquitectura se basa en un esquema cliente-servidor, pero presenta una serie de desventajas, que conllevan a la necesidad de replantearse su modelo de funcionamiento.

El objetivo de este capítulo es basado en los errores de DrQueue expuestos anteriormente hacer una nueva propuesta de **arquitectura** para un futuro sistema de renderizado distribuido. La arquitectura serviría para definir un sistema eficiente que permita un mejor uso del renderizado en este proyecto.

2.1 Arquitectura de Software

Según RUP el concepto de Arquitectura de Software (AS) involucra los aspectos estáticos y dinámicos más significativos del sistema. La arquitectura surge de las necesidades de la empresa, tal y como las interpretan los usuarios y otros stakeholders[23], y tal y como están reflejadas en los casos de uso. Sin embargo, también está influenciada por muchos otros factores, tales como la plataforma de software en la que se ejecutará, la disponibilidad de componentes reutilizables, las consideraciones de instalación, los requerimientos no funcionales (ej. desempeño, confiabilidad) etc. La arquitectura es la vista del diseño completo

23 Stakeholders es aquella persona o entidad que esta interesada en la realización de un proyecto o tarea.

con las características más importantes hechas más visibles y dejando los detalles a un lado. Ya que lo importante depende en parte del criterio, el cual a su vez viene con la experiencia, el valor de la arquitectura depende del personal asignado a esta tarea. Sin embargo, el proceso ayuda al arquitecto a enfocarse en las metas correctas, tales como claridad (*understandability*), flexibilidad en los cambios futuros (*resilience*) y reuso. (RUP 2010)

Según Josep Casanovas (Josep Casanovas 2004) la Arquitectura del Software es el diseño de más alto nivel de la estructura de un sistema, programa o aplicación y tiene la responsabilidad de:

- Definir los módulos principales.
- Definir las responsabilidades que tendrá cada uno de estos módulos.
- Definir la interacción que existirá entre dichos módulos.
- Control y flujo de datos.
- Secuenciación de la información.
- Protocolos de interacción y comunicación.
- Ubicación en el hardware.

La Arquitectura del Software aporta una visión abstracta de alto nivel, posponiendo el detalle de cada uno de los módulos definidos a pasos posteriores del diseño.

La definición teórica de AS según el Instituto de Ingenieros Electricistas y Electrónicos (IEEE)[24] en su clásica guía de recomendaciones IEEE 1471 es: "La arquitectura de un sistema de software es la estructura o estructuras del sistema, que incluye los elementos de software, las propiedades externamente visibles de esos elementos y la relación entre ellos." (Diego Gomez 2008)

Varias han sido las definiciones que se le han asignado a la AS en el mundo. No es novedad

24 Asociación técnico-profesional mundial dedicada principalmente a la estandarización.

que ninguna definición de la AS es respaldada unánimemente por la totalidad de los arquitectos. El número de definiciones circulantes alcanza un orden de tres dígitos, amenazando llegar a cuatro. No obstante la abundancia de definiciones del campo de la AS, existe en general un acuerdo de que ella se refiere a la estructura a grandes rasgos del sistema, estructura consistente en componentes y relaciones entre ellos.

Uno de los aspectos fundamentales dentro de la Arquitectura de Software son los estilos arquitectónicos. Un estilo es un concepto descriptivo que define una forma de articulación u organización arquitectónica. El conjunto de los estilos cataloga las formas básicas posibles de estructuras de software, mientras que las formas complejas se articulan mediante composición de los estilos fundamentales. (Carlos Billy Reynoso 2004)

Un estilo arquitectónico o variante arquitectónica define a una familia de sistemas informáticos en términos de su organización estructural. Este describe los componentes y las relaciones entre ellos con las restricciones de su aplicación, la composición asociada y el diseño para su construcción. Los sistemas empresariales distribuidos pueden agrupar los siguientes estilos arquitectónicos, entre otros:

- Modelo-Vista-Controlador (MVC).
- Arquitecturas en Capas.
- Arquitecturas Orientadas a Objetos.
- Arquitecturas Basadas en Componentes.
- Arquitecturas Orientadas a Servicios.
- Arquitectura Cliente-Servidor.

Cada uno de estos estilos, plantea su propia estructura de componentes, teniendo un propósito específico y por lo tanto un área de aplicabilidad bien determinada. (Aneli Valdes Acosta y María de Dolores Guardia Macias 2007)

2.1.2 Arquitectura Cliente-Servidor

Esta arquitectura consiste básicamente en un cliente que realiza peticiones a otro programa (el servidor) que le da respuesta.

En esta arquitectura la capacidad de proceso está repartida entre los clientes y los servidores, aunque son más importantes las ventajas de tipo organizativo debidas a la centralización de la gestión de la información y la separación de responsabilidades, lo que facilita y clarifica el diseño del sistema.

La *International Business Machines* (IBM)[25] define al modelo Cliente-Servidor: "Es la tecnología que proporciona al usuario final el acceso transparente a las aplicaciones, datos, servicios de cómputo o cualquier otro recurso del grupo de trabajo y/o a través de la organización, en múltiples plataformas. El modelo soporta un medio ambiente distribuido en el cual los requerimientos de servicio hechos por estaciones de trabajo inteligentes o "clientes", resultan en un trabajo realizado por otros computadores llamados servidores".

Según Sara Alvarez (Sara Alvarez 2007) algunas de las características de esta arquitectura son:

- Esta arquitectura se divide en dos partes claramente diferenciadas, la primera es la parte del servidor y la segunda la de un conjunto de clientes.
- Normalmente el servidor es una máquina bastante potente que actúa de depósito de datos y funciona como un sistema gestor de base de datos (SGBD).
- Por otro lado los clientes suelen ser estaciones de trabajo que solicitan varios servicios al servidor.
- Ambas partes deben estar conectadas entre sí mediante una red.

Una representación gráfica de este tipo de arquitectura sería la siguiente:

25 Empresa multinacional que fabrica y comercializa herramientas, programas y servicios relacionados con la informática.

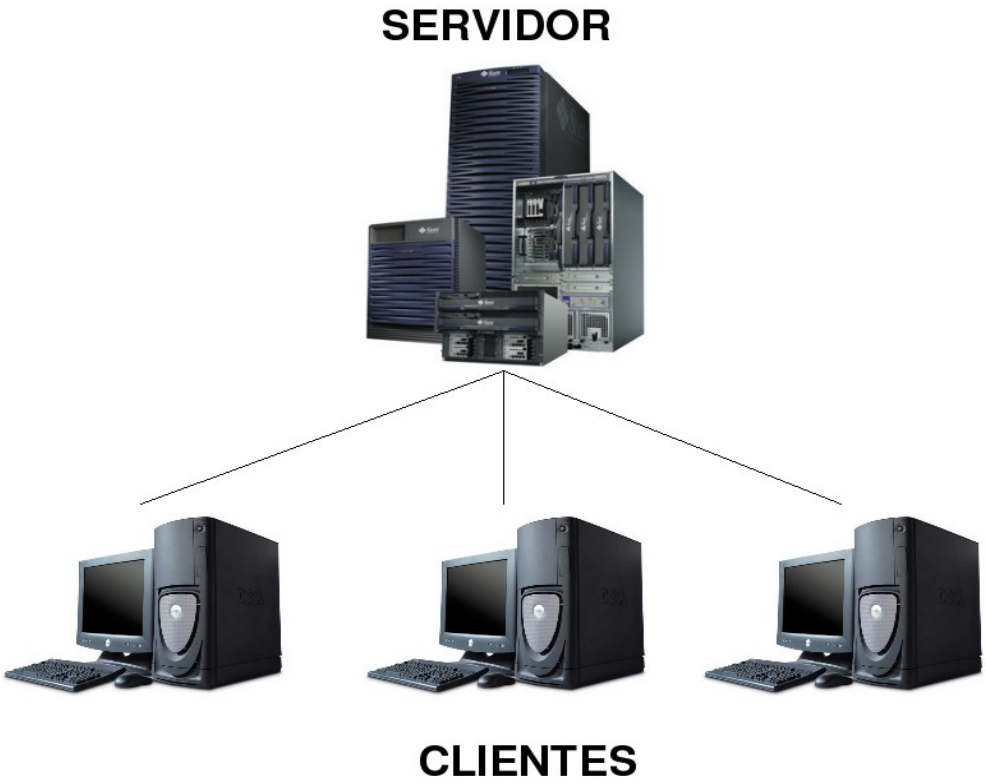


Figura 9: *Arquitectura Cliente-Servidor*

2.2 Lenguajes, herramientas, librerías y metodología para el desarrollo

A continuación se dará una breve explicación de los lenguajes, herramientas, librerías y metodología usada para el desarrollo.

2.2.1 Lenguajes de programación

En informática, un programa es una secuencia de instrucciones que permiten a un ordenador procesar una información conocida como datos de entrada (*input*) para producir una información de salida (*ouput*) o resultados.

Esas instrucciones pertenecen a (o están escritas en) un lenguaje de programación

determinado.

Un lenguaje de programación es una construcción mental del ser humano para expresar programas. Está constituido por un grupo de reglas gramaticales, un grupo de símbolos utilizables, un grupo de términos con sentido único y una regla principal que resume las demás. Para que ésta construcción mental sea operable en un computador debe existir otro programa que controle la validez o no de lo escrito. A éste se le llama traductor.

A continuación se muestran los lenguajes de programación empleados.

- **Lenguaje de programación C**

C es un lenguaje de programación creado en 1969 por Dennis M. Ritchie en los Laboratorios Bell. Es un lenguaje orientado a la implementación de sistemas operativos, concretamente Unix. C es apreciado por la eficiencia del código que produce, especialmente para el procesamiento de algoritmos complejos, como los relacionados con los gráficos por computador.

Entre sus características se encuentran:

- Cuenta con un núcleo del lenguaje simple, con funcionalidades añadidas importantes, como funciones matemáticas y de manejo de archivos, proporcionadas por bibliotecas.
- Es un lenguaje muy flexible que permite programar con múltiples estilos. Uno de los más empleados es el estructurado "no llevado al extremo" (permitiendo ciertas licencias de ruptura).
- Tiene un sistema de tipos que impide operaciones sin sentido.
- Usa un lenguaje de preprocesado. El preprocesador de C es usado para tareas como definir macros e incluir múltiples archivos de código fuente.
- Acceso a memoria de bajo nivel mediante el uso de punteros.

- Interrupciones al procesador con uniones.
- Un conjunto reducido de palabras clave.
- Por defecto, el paso de parámetros a una función se realiza por valor. El paso por referencia se consigue pasando explícitamente a las funciones las direcciones de memoria de dichos parámetros.
- Punteros a funciones y variables estáticas, que permiten una forma rudimentaria de encapsulado y polimorfismo.
- Tipos de datos agregados (*struct*) que permiten que datos relacionados (como un empleado, que tiene un id, un nombre y un salario) se combinen y se manipulen como un todo (en una única variable "empleado").

➢ Lenguaje de programación Python

Python es un lenguaje de programación interpretado creado por Guido van Rossum en el año 1991. En la actualidad Python se desarrolla como un proyecto de código abierto, administrado por la Python Software Foundation.
Python permite dividir el programa en módulos reutilizables desde otros programas Python. Se utiliza como lenguaje de programación interpretado, lo que ahorra un tiempo considerable en el desarrollo del programa, pues no es necesario compilar ni enlazar.

Entre sus características se encuentran:

- Python es un lenguaje de programación multiparadigma, esto significa que más que forzar a los programadores a adoptar un estilo particular de programación, permite varios estilos: programación orientada a objetos, programación estructurada y programación funcional. Otros paradigmas están soportados mediante el uso de extensiones.
- Python usa tipo de dato dinámico y *reference counting*[26] para la utilización de la memoria.

26 Técnica para contabilizar las veces que un determinado recurso está siendo referido.

- Una característica importante de Python es la resolución dinámica de nombres; es decir, lo que enlaza un método y un nombre de variable durante la ejecución del programa (también llamado ligadura dinámica de métodos).

- Otro objetivo del diseño del lenguaje era la facilidad de extensión. Se pueden escribir nuevos módulos fácilmente en C o C++. Python puede incluirse en aplicaciones que necesitan una interfaz programable.

Se propone el uso de este lenguaje para la realización de los script que ejecutarán los servidores de render.

➢ **Bash**

Bash (GNU Project) es un intérprete de comandos de tipo Unix (*shell*) escrito para el proyecto GNU. Su nombre es un acrónimo de Bourne-Again Shell. Es el shell por defecto en la mayoría de las distribuciones de GNU/Linux de hoy en día. Se encarga de interpretar las órdenes para su proceso por el kernel. Al escribir un comando, es Bash el responsable de interpretarlo y ordenar que se ejecute.

Se propone el uso de Bash para mejorar la rapidez en la instalación de los esclavos.

2.2.2 Librerías

➢ **GTK**

GTK es una librería construida sobre GDK (el conjunto de herramientas de dibujo del Gimp). GTK es un API[27] orientado a objetos. Está completamente escrita en C y contiene los objetos y funciones para crear la interfaz gráfica de usuario. Maneja widgets como ventanas, botones, menús, etiquetas, deslizadores, pestañas, etc. (Ian Main 1998).

27 Interfaz de Programación de Aplicaciones o API (del inglés Application Programming Interface) es el conjunto de funciones y procedimientos que ofrece cierta librería para ser utilizado por otro software como una capa de abstracción.

2.2.3 Herramientas:

- **GForge**

GForge es un software que ofrece alojamiento de proyectos, control de versiones (CVS y Subversion), seguimiento de fallos, y mensajería. Es Software Libre licenciado bajo la GPL (GNU General Public License), aunque no su última versión GForge Advance Server. En la actualidad GForge es la plataforma implementada para los proyectos de la facultad 10 de la UCI.

- **IDE**[28]

SCIntilla based Text Editor o SciTE es un editor de textos multiplataforma escrito por Neil Hodgson usando el componente de edición Scintilla[29]. La versión actual es la 1.76, lanzada el 16 de marzo de 2008.

Ligero y hecho para ser veloz, está diseñado principalmente para edición de código fuente, y resaltado de sintaxis, y tiene referencia de función en línea para muchos lenguajes de computación. También hay una versión con un solo .exe ejecutable que no necesita archivos adicionales, siendo ideal para transportarlo y usarlo desde memorias USB en forma de aplicación portátil. SciTE comparte algunas características con Notepad++ que también está basado en el componente de edición Scintilla.

2.2.4 Metodología

- **Metodología de desarrollo**

Por la envergadura del proyecto a realizar se requiere de una metodología de desarrollo rápida y flexible, que describa un proceso sencillo y bien documentado de los elementos

28 Entorno de desarrollo integrado (acrónimo en inglés de *integrated development environment*), es un programa informático compuesto por un conjunto de herramientas de programación.
29 Scintilla es un componente libre de edición de código fuente.

imprescindibles tanto para los clientes como para los desarrolladores, proponiéndose SXP, la cual reúne muchas de estas características, y fue desarrollada en la UCI para proyectos de Software Libre.

SXP es una metodología compuesta por las metodologías SCRUM y XP, que ofrece una estrategia tecnológica, a partir de la introducción de procedimientos ágiles, que permite actualizar los procesos de software para el mejoramiento de la actividad productiva, propicia el desarrollo de la creatividad, eleva el nivel de preocupación y responsabilidad de los miembros del equipo y ayuda al líder del proyecto a tener un mejor control de este.

SCRUM es una metodología para gestionar un equipo de manera que trabaje eficientemente, proporcionando información sobre los progresos alcanzados.

XP es una metodología encaminada al proceso de desarrollo, que se centra en el principio de una programación rápida o extrema, cuya particularidad es tener como parte del equipo al usuario final, lo cual es requisito indispensable para el éxito del proyecto.

Consta de 4 fases principales:

1. Planificación - Definición: donde se establece la visión, se fijan las expectativas y se realiza el aseguramiento del financiamiento del proyecto.
2. Desarrollo: es donde se realiza la implementación del sistema hasta que este listo para ser entregado.
3. Entrega: es la puesta en marcha.
4. Mantenimiento: es la fase donde se realiza el soporte para el cliente.

Por cada una de estas fases se realizan numerosas actividades tales como el levantamiento de requisitos, la priorización de la Lista de Reserva del Producto, definición de las Historias de Usuario, Diseño, Implementación, Pruebas, entre otras; de donde se generan artefactos para

documentar todo el proceso.

Las entregas son frecuentes, y existe una refactorización continua, lo que nos permite mejorar el diseño cada vez que se le añada una nueva funcionalidad. SXP está especialmente orientada a proyectos de pequeños equipos de trabajo, rápido cambio de requisitos o requisitos imprecisos, muy cambiantes, donde existe un alto riesgo técnico y se orienta a una entrega rápida de resultados y una alta flexibilidad.

Ayuda a que trabajen todos juntos, en la misma dirección, con un objetivo bien delimitado, permitiendo además seguir de forma clara el avance de las tareas a realizar, de forma que los jefes puedan ver día a día cómo progresa el trabajo. (Gladys Marsi 2008)

2.3 Descripción de la arquitectura

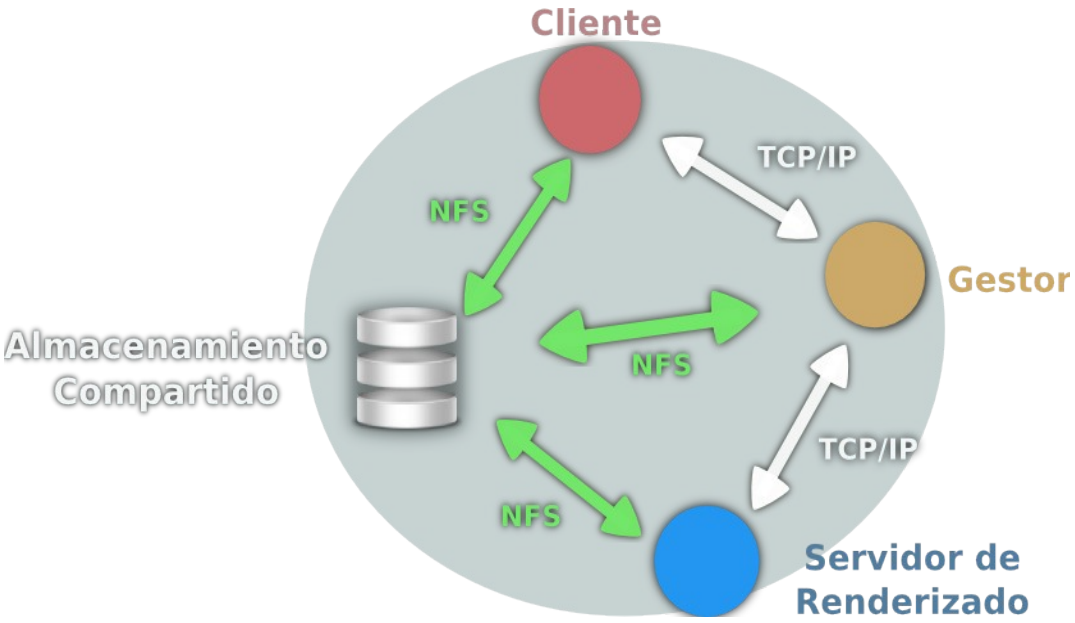


Figura 10: *Arquitectura del sistema propuesto*

La solución propuesta se basa, como se describe en la Figura 10, en la arquitectura Cliente-Servidor, que caracteriza a los sistemas distribuidos, compuesta por varias capas.

2.3.1 Componentes de la arquitectura

- **Cliente/Usuario (C/U)**

El Cliente o Usuario, es la entidad que solicita el renderizado de trabajos al Gestor de Renderizado Distribuido. Este es la GUI (Interfaz Gráfica de Usuario) la cual hace posible la interacción con el Gestor. Los trabajos que son solicitados para su renderización son modelos implementados en alguna aplicación de infografía, como Blender, Maya etc.

- **Gestor de Renderizado Distribuido (GRD)**

El Gestor de Renderizado Distribuido sería lo que para DrQueue es el Master, este es el centro del Sistema, puede afirmarse que el cerebro encargado de organizar y distribuir en forma de tareas a los servidores de renderizado los trabajos solicitados por los clientes, es decir que gestiona tanto los clientes, los cuales pueden solicitar varios trabajos, como los servidores de renderizado y las tareas asignadas a estos.

Para organizar y distribuir las tareas de renderizado entre los diferentes servidores de render disponibles, el Gestor de Renderizado Distribuido se basa en estrategias de distribución, que forman parte de su metodología de trabajo. El Gestor puede lo mismo utilizar alguno de los métodos de fragmentación de frame existentes o enviar, tal cual hace DrQueue los frames completos para su renderización. En la propuesta el gestor de renderizado debe ser capaz de contar con todos los métodos de distribución.
El Gestor de Renderizado Distribuido es único, lo cual evita redundancias innecesarias y ralentización de los procesadores de los ordenadores involucrados, como consecuencia del posible acceso simultáneo de dos procesos similares.

- **Servidor de Render (SR)**

El Servidor de Render o de Renderizado, sería lo que para DrQueue es el servidor esclavo (*slave*), y por tanto siempre estaría a la escucha de las peticiones del Gestor de Renderizado

Distribuido. Este componente no solo tiene el propósito de gestionar las tareas de renderizado de aquel con el Motor de Renderizado, sino también monitorear automáticamente la CPU del host donde se hospeda y brindar las informaciones que le sean solicitadas. Importante a tener en cuenta es que un Servidor de Render maneja tareas de renderizado de forma secuencial, es decir que no podrá atender ninguna tarea en paralelo, negándose a ello cuando el Gestor de Renderizado Distribuido se lo pida.

Protocolo

En informática, un protocolo es un conjunto de reglas usadas por computadoras para comunicarse unas con otras a través de una red. Un protocolo es una convención o estándar que controla o permite la conexión, comunicación, y transferencia de datos entre dos puntos finales. En su forma más simple, un protocolo puede ser definido como las reglas que dominan la sintaxis, semántica y sincronización de la comunicación. Los protocolos pueden ser implementados por hardware, software, o una combinación de ambos. A su más bajo nivel, un protocolo define el comportamiento de una conexión de hardware.

El protocolo es el componente lógico responsable del trasiego a través de la red de la información entre los clientes y el Gestor de Renderizado Distribuido, y entre este último y los servidores de renderizado.

En la propuesta el protocolo de comunicación con el almacenamiento compartido será por NFS y entre los diferentes componentes por TCP/IP.

TCP/IP: Son 2 protocolos, el primero es el Protocolo de Control de Transmisión (TCP) y el segundo el Protocolo de Internet (IP).

El TCP/IP es la base de Internet, y sirve para enlazar computadoras que utilizan diferentes sistemas operativos, incluyendo PC, minicomputadoras y computadoras centrales sobre redes de área local (LAN) y área extensa (WAN).

El conjunto TCP/IP está diseñado para enrutar y tiene un grado muy elevado de fiabilidad, es adecuado para redes grandes y medianas, así como en redes empresariales. Se utiliza a nivel mundial para conectarse a Internet y a los servidores web. Es compatible con las herramientas

estándar para analizar el funcionamiento de la red.

NFS: El Network File System (Sistema de archivos de red), o NFS, es un protocolo de nivel de aplicación, según el Modelo OSI. Es utilizado para sistemas de archivos distribuido en un entorno de red de computadoras de área local. Posibilita que distintos sistemas conectados a una misma red accedan a ficheros remotos como si se tratara de locales.
El protocolo NFS está diseñado para ser independiente de la máquina, el sistema operativo y el protocolo de transporte.

2.3.2 Variantes de distribución

Cada una de las entidades involucradas en el SGRD es independiente desde el punto de vista lógico, comunicándose entre ellas a través de la red, de manera que pueden existir físicamente en lugares distintos, por lo que varios serían los modelos posibles de distribución que podrían implementarse, como puede verse en la Figura 11.

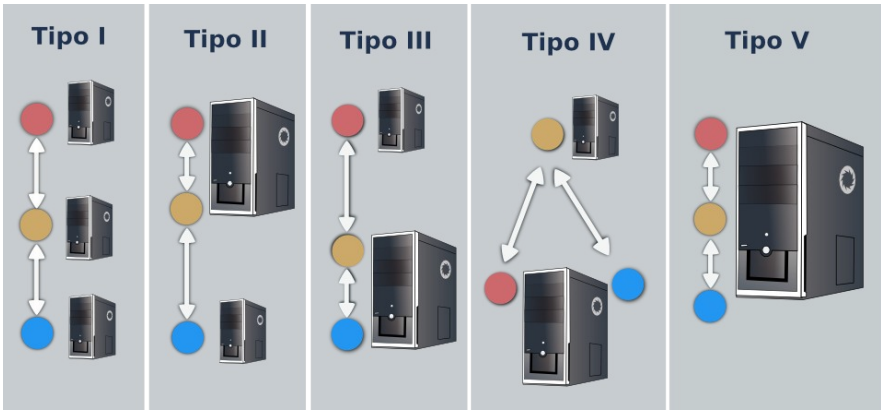


Figura 11: *Variantes de distribución del Sistema*

De las cinco posibles combinaciones de distribución, la más recomendable a usar sería la primera, pues el descentralizar físicamente las actividades, permitiría una mayor dedicación de los componentes así como una disminución de la latencia en los servidores, entre otros beneficios, también de carácter organizativo. En ese caso los clientes y los servidores estarían separados desde el punto de vista físico del gestor.

Lamentablemente la universidad en estos momentos no cuenta con máquinas de alta potencia en lo que a microprocesador y memoria RAM se refiere. Esto hace que el renderizado se realice con las máquinas de proyecto y de docencia. En ese caso, la distribución más adecuada al entorno UCI, sería cualquiera de los tipos presentados menos el primero.

Ventajas

- La principal ventaja de esta arquitectura está en la posibilidad de que desde cualquier PC (Cliente) puedan ser enviados trabajos al gestor para ser renderizados por los diferentes servidores de render.
- A través de un grupo de políticas, el gestor define el mejor método de distribuir los trabajos entre los servidores de render, haciendo un control riguroso del proceso, para lo cual se nutre de las informaciones periódicas recibidas tanto de los servidores de render como de los clientes.
- El cliente podría escoger si usar el envío del frame completo o si desearía usar algún método de fragmentación de frame.

Desventajas

- En la actualidad, la arquitectura no contempla mecanismos en el gestor para recuperación ante fallas, lo cual será un objetivo a tener en cuenta para versiones posteriores del Sistema.

CAPÍTULO 3: IMPLEMENTACIÓN DE SISTEMA DE RENDERIZADO DISTRIBUIDO

Para realizar la implementación se tomó en cuenta DrQueue para obtener un sistema básico y no partir completamente desde 0, máxime cuando la aplicación goza internacionalmente de gran aceptación pese a sus deficiencias.

DrQueue distribuye una tarea en una granja de computadoras conectadas mediante una red. Los frames repartidos se renderizan por dichas computadoras para después juntarlos y obtener el resultado final del proceso. DrQueue soporta varios motores de renderizado, también permite la posibilidad de escribir algún script si se necesita algún otro motor de render o si fuera necesario modificar la forma de trabajo.
El siguiente capítulo trata sobre la modificación de un script para lograr realizar la renderización por partes.

3.1 Historias de Usuario

Historia de Usuario	
Número: ***1***	**Nombre Historia de Usuario:** ***Renderizar imagen por partes.***
Modificación de Historia de Usuario Número:	
Usuario: ***Nelio Véliz Pedraza***	**Iteración Asignada:** ***1***
Prioridad en Negocio: ***Alta***	**Puntos Estimados:** ***2***
Riesgo en Desarrollo: ***Bajo***	**Puntos Reales:** ***1***
Descripción: ***A la hora de renderizarse un trabajo el software tiene que ser capaz de***	

enviar a renderizar una imagen en varias partes y no completa como hace comunmente.
Observaciones:
Prototipo de interfaz: ***Ver figura 3 Anexos.***

3.2 Algoritmo

```
maxparts = 3  #Cantidad de partes en las que se dividirá el frame.

curpart =  int(os.getenv('curpart'))  #Número que irá incrementándose, toma el valor de $DRQUEUE_FRAME.

if curpart > 0:

        scn = Scene.GetCurrent()                        #Tomar la escena actual.

        context = scn.getRenderingContext()             #Obtener el contexto de renderización para esta escena.

        context.startFrame(curpart)                     #Establecer la imagen de comienzo de la renderización.

        context.endFrame(curpart)                       #Establecer la imagen de fin de la renderización.

        context.enableBorderRender(True)                #Permitir la renderización por regiones.

        scenefile ="/var/lib/drqueue/pool/part1_"       #Lugar donde se guardarán los resultados.

        context.setRenderPath(scenefile)                #Ruta donde se escribirá el resultado.

        height = context.imageSizeY()                   #Altura de la imagen.

        width = context.imageSizeX()                    #Ancho de la imagen.

        part_height = float(height) / float(maxparts)   #Cálculo del tamaño de cada región en px.
```

```
bottom_px = float(height) - (1 * part_height)     #Cálculo desde abajo.
top_px = float(height)                            #Cálculo de la altura.

if top_px < height:                          #Validación para los píxeles de unión.
        top_px = top_px + 5
if bottom_px > 0.0:
        bottom_px = bottom_px - 5

bottom = bottom_px / float(height)               #Llevar de píxeles a escala de 0 a 1.
top = top_px / float(height)                     #Llevar de píxeles a escala de 0 a 1.

print 'curpart: ' + str(curpart)             #Área de impresión para los logs.
print 'maxparts: ' + str(maxparts)
print 'scenefile: ' + scenefile
print 'width: ' + str(width)
print 'height: ' + str(height)
print 'part_height: ' + str(part_height)
print 'bottom_px: ' + str(bottom_px)
print 'top_px: ' + str(top_px)
print 'bottom: ' + str(bottom)
print 'top: ' + str(top) + '\n'
```

```
        context.setBorder(0.0, bottom, 1.0, top)   #Bordes para renderizar:"setBorder(left,
bottom, right, top)"

        context.renderAnim()                              #Renderizar los fotogramas.
```

3.3 Desventajas del algoritmo anterior y ventajas del nuevo algoritmo

Al realizarse la renderización en DrQueue el Master envía un script a cada PC esclava donde le indica la entre otros parámetros la dirección de donde se encuentra el archivo a renderizar (ej: .blend[30]) y las funciones que debe realizar la PC con el mismo. Esto es posible teniendo en cuenta que existe un recurso compartido al cual todas las PCs del sistema de gestión pueden acceder.

DrQueue por defecto envía un script el cual hace que cada PC renderice una imagen completa. También cuenta con un script para renderizar por regiones pero el mismo no funciona correctamente. A este script se le realizaron algunas modificaciones logrando funcionar y cumplía su objetivo (enviar una porción de imagen a cada máquina) solo que al terminar con el primer frame los cálculos que el mismo realizaba para calcular el área de renderización comenzaban a dar negativos y por lo tanto el área a ser demasiado pequeña para ser renderizada mostrando el siguiente error en los logs: *ERROR: Image too small* (ERROR: Imagen demasiado pequeña) y en los resultados después de haber renderizado el primer frame, mostraba una imagen de color negra completamente, no continuando la renderización con el resto de los frames.

En respuesta a este problema se arregló el script de tal forma que actualmente una PC sigue renderizando una imagen completa pero por fragmentos, o sea, ella toma la primera parte de la imagen, la renderiza y la entrega en un .png, luego toma la 2da y así sucesivamente. Ver Figura 12.

30 Extensión con la cual salen los archivos una vez trabajados en el software de infografía: blender.

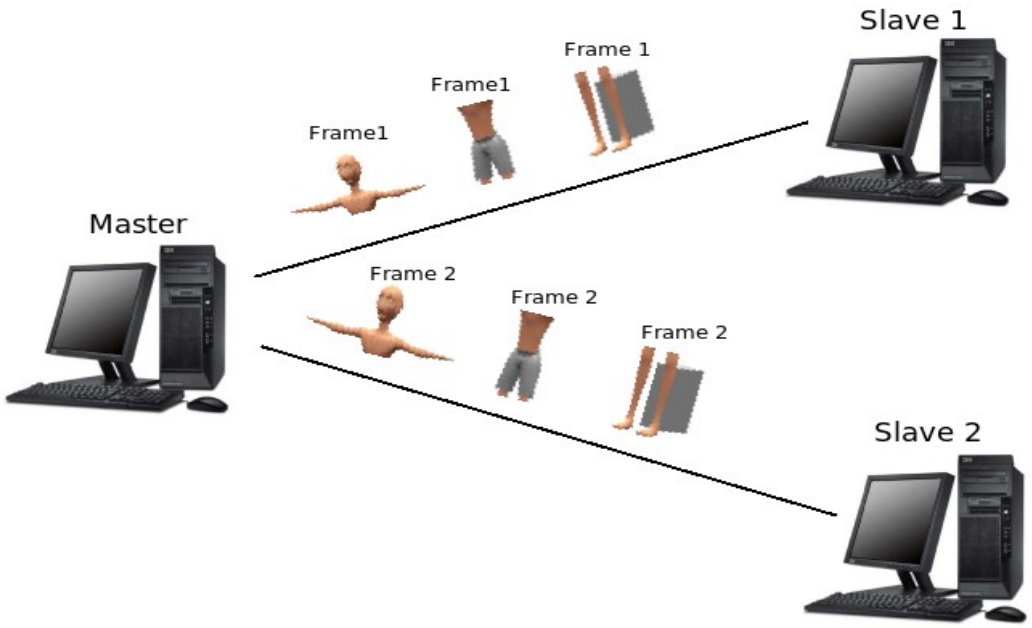


Figura 12: *Distribución que realiza la PC Master al renderizar con el nuevo script*

En el script anterior cada fragmento renderizado era una tarea, por lo tanto de necesitarse renderizar 90 frames se tendría que calcular, suponiendo que la imagen la dividiera en 6, 90x6, y poner este cálculo en la cantidad de frame a renderizar, cuestión esta que podría provocar equivocaciones. También en caso de necesitar renderizar solamente por ejemplo del frame 10 al 25 se tendrían que realizar otra cantidad de cálculos para poder entrar esos datos y comenzar así la renderización.

Sin embargo con el script actual se puede poner los 90 frames (en caso de que este fuera el total) como tarea ya que el hasta que no termine cada porción de un frame completo no pondrá la tarea como realizada. En caso de necesitar solamente renderizar una parte lo pone tal cual es, del Frame 10 al Frame 25.

La otra ventaja que existe es que en los cálculos nunca el algoritmo dará negativo, siempre cumplirá la cantidad de frames que le sea enviado.

Otra ventaja que presenta el mismo es que resulta fácil saber cuales son las imágenes a unir pues la 1ra parte la muestra como Part1_0001, la 2da: Part2_0001 y la 3ra: Part3_0001. Es tan solo fijarse en el número "0001" y unir las imágenes.

Esto conlleva a un menor consumo de memoria y abuso del CPU teniendo en cuenta que las PCs usadas para la renderización son las de proyecto y muchas veces se realiza este proceso en horario de trabajo.

3.4 Pruebas realizadas

En esta sección se realizará una comparativa de los tiempos obtenidos y el por ciento de memoria usado, entre la manera tradicional que DrQueue realizar el renderizado y mediante el uso del nuevo algoritmo. Es importante destacar que todas estas pruebas fueron realizadas en horario de trabajo teniendo en cuenta lo explicado en el capítulo 2.

➢3.4.1 Hardware

Las pruebas se realizaron en **4** computadoras pertenecientes al laboratorio 102, Docente 3, UCI. Sus características son:

Propiedad	Valor
Procesador	Intel Core 2 Duo 2.20GHz
Memoria RAM	1GB
Disco Duro	150GB
Targeta Madre	Intel
Sistema Operativo	GNU/Linux

Tabla 1: Características de las PC

➢3.4.2 Trabajos Realizados

Las renderizaciones realizadas que fueron tomadas como prueba puede verlas en la siguiente tabla.

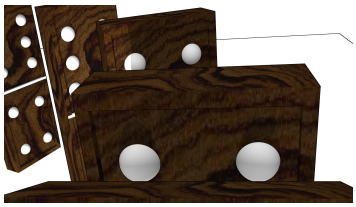
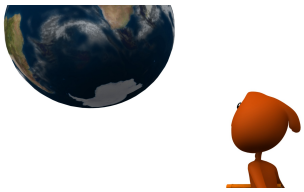
Nombre	Escena	Descripción de la animación	Tamaño del archivo
1. Viejo		Señor que se sienta y luego levanta un pie.	6.6MB
2. Carro		Carro recorriendo un túnel.	5.3MB
4. Domino		Varias fichas de dominó cayendo.	8.1MB
5. Enel Preview		Dibujo que ve la aparición de la tierra.	2.3MB
6. Historias Nativas.		La cámara recorriendo un árbol.	118.0MB

Tabla 2: *Trabajos renderizados*

➢3.4.3 Comparación entre los algoritmos

Nombre	Cant Frames	Tiempo 1 (min)*	Tiempo 2 (min)*	% de Memoria 1*	% de Memoria 2*	Resolución
1. Viejo	90	20	15	10	6	1280x1024
2. Carro	200	10	20	9	5	1920x1080
3. Dominó	200	92	101	10	6	1920x1080
4. Enel Preview	50	13	12	10	6	1920x1080
5. Historias Nativas	1		66			1920x1080

Tabla 3: *Comparación entre algoritmos*

*El tiempo 1 y el % de memoria 1 se refiere al tiempo consumido y el por ciento de memoria promedio usado cuando es enviada a renderizar a cada computadora un frame completo. Ya el tiempo 2 y el % de memoria 2 es similar solo que esta vez es cuando es enviada a renderizar un frame por fragmentos.

➢3.4.4 Historias Nativas

Historias Nativas (HN) es un proyecto creado en Venezuela para incursionar en el terreno de las producciones animadas en 3D promocionando historias de alto valor cultural, como las leyendas de sus indígenas.

Como objetivo general, HN se trazó el fortalecimiento del ámbito de producciones animadas en Venezuela, realizadas con Software Libre siendo el principal software de trabajo: Blender.

Contacto con FreeViUX

Con los autores de “El pequeño gran libro” se establecieron las primeros contactos para renderizar sus producciones. La idea era que se renderizarían sus trabajos gratis con la condición de que dejaran transmitirlos en la televisión cubana. Este intercambio nunca llegó a crearse por razones desconocidas. Una vez que los productores de “Historias Nativas”

mediante el vínculo con los autores de “El pequeño gran libro” supieron de la existencia de una granja de render en Cuba entablaron comunicación para renderizar sus trabajos (Para leer el correo del productor de Historias Nativas referirse a los Anexos).

Renderización de HN

El equipo de Historias Nativas envió un preliminar de 23 segundos (552 frames = 24 frames por segundo) de alta definición (1920x1080), el material pudo ser renderizado en el laboratorio de proyecto cumpliendo las siguientes condiciones:

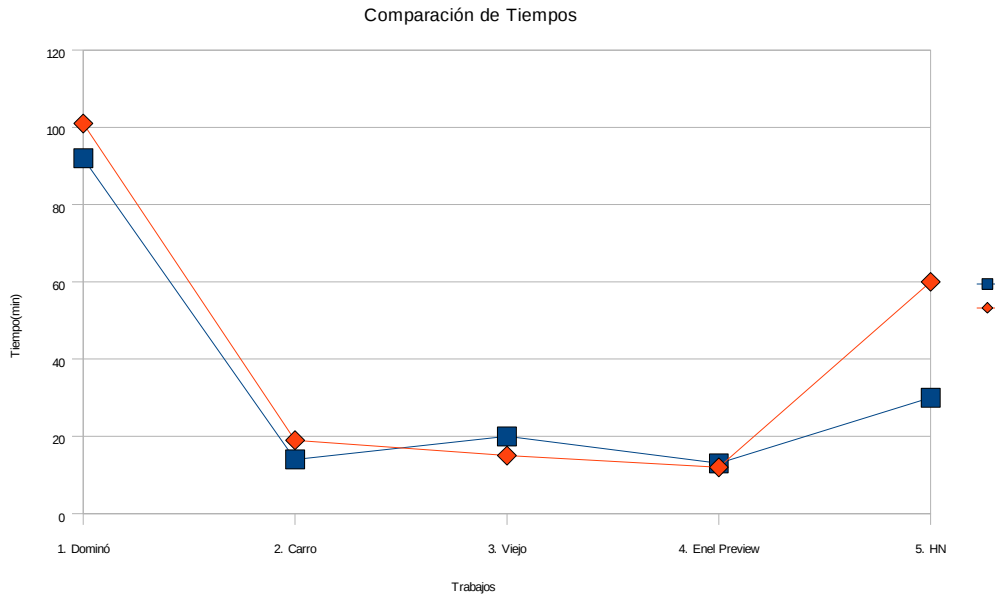
- Cada estación de trabajo solo podía realizar 1 frame al mismo tiempo (aunque las PCs poseen 2 cores y sería realizar dos tareas simultáneas, la limitación en RAM obligaba a esto).
- Cada estación tenía que tener la Interfaz Gráfica Inactiva (puede estar instalada pero había que parar el xorg para disponer de RAM).
- El vm.swappiness tenía que estar al menos a 70 para mayor uso de la swap.
- Se tenía que incrementar la prioridad del "swapeo" (prioridad de swap al menos a 25000, esto ayudaba a que se acelerara un poco el trabajo con la swap.

Con esta configuración el render de cada frame demoró aproximadamente 30 minutos.

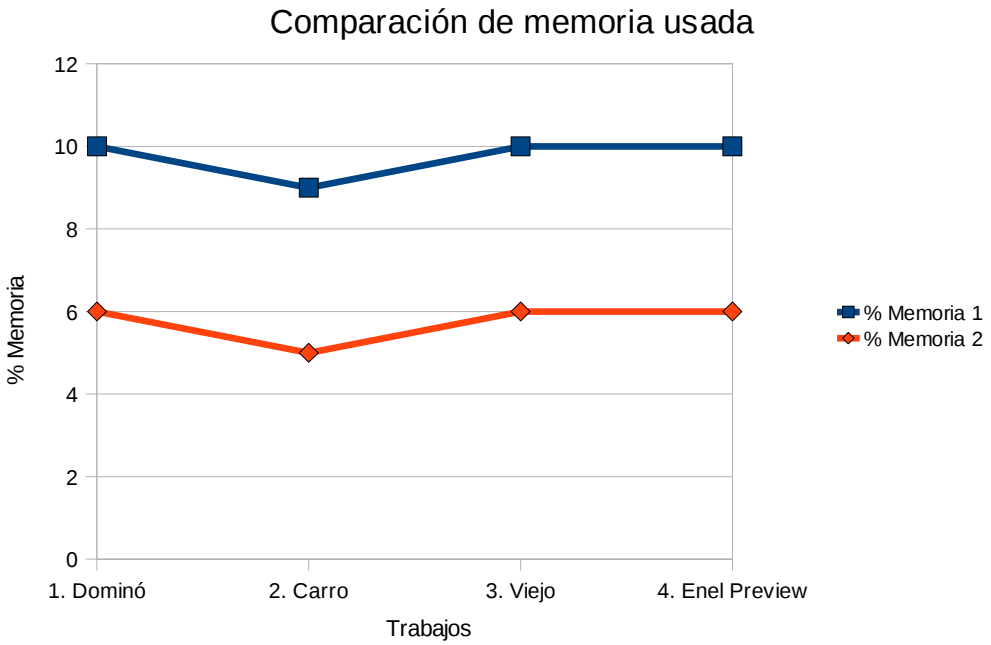
Historias Nativas demostró ser el primer trabajo internacional que se realizó con el SGRD del proyecto FreeViUX y aunque el resultado por cuestiones de “desacuerdos” nunca les fue entregado si demostró que puede existir un intercambio con los países para la realización de esta tarea.

Al probarse el nuevo algoritmo en esta renderización tan compleja el mismo tuvo resultados satisfactorios pues al disminuir el consumo de memoria no fue necesario detener la interfaz gráfica, no obstante cada frame demoró para realizarse alrededor de una hora y media.

3.5 Análisis de los resultados



Gráfica 1: *Comparación entre tiempos*



Gráfica 2: *Comparación de memoria usada*

Es posible apreciar según las gráficas que cuando se renderiza enviando un frame completo a cada PC los tiempos tienden a ser muy similares, esto exceptuando la renderización de Historias Nativas para la cual se tomaron en cuenta ciertas medidas descritas anteriormente. No obstante en el por ciento de memoria usado pudo notarse que al desfragmentar el frame el consumo es menor dando espacio a la mejor realización de las tareas que se estén realizando en cada PC.

CONCLUSIONES

A partir de la investigación realizada y para darle cumplimiento a los objetivos del trabajo de diploma se arriban a las siguientes conclusiones:

- Se logró profundizar en una serie de conceptos referentes a los sistemas de render distribuidos orientándose principalmente en los SGRD libres.
- Se demostró que la forma más factible de optimizar el tiempo de renderizado es mediante el uso de la computación distribuida. Lamentablemente la mayoría de los software para la renderización distribuida se crean de manera privativa y con licencias altas en costo.
- Los países del tercer mundo, como por ejemplo Cuba, tienen una alternativa para la renderización mediante la computación distribuida en el uso de Software Libre. DrQueue ha demostrado ser ejemplo de ello.
- Se propuso el uso de una nueva arquitectura para DrQueue teniendo en cuenta principalmente el no contar con computadoras específicamente para la renderización y realizarse esta en las de trabajo.
- Se implementó un algoritmo que permite renderizar los frames por parte siendo el consumo de memoria a la hora del renderizado mucho menor en cada PC y por tanto disminuye la lentitud que causa la renderización en las computadoras de trabajo.

De crear una buena infraestructura de intercambio por Internet, Cuba mediante el uso de Software Libre pudiera ser la alternativa para realizar las renderizaciones a los países del 3er mundo, principalmente los latinoamericanos, demostrando esto en la realización de la renderización para Venezuela de “Historias Nativas”.

RECOMENDACIONES

Los objetivos de este trabajo han sido logrados, teniendo en cuenta que se cumplieron todos requerimientos planteados. No obstante se hacen las siguientes recomendaciones:

- Validar la arquitectura propuesta en el capítulo 2.
- Realizar un estudio que permita unir de forma automática las imágenes cuando son renderizadas por parte.
- Estudiar en conjunto con la *Blender Foundation* la posible inclusión de la granja de render DrQueue en Blender.

REFERENCIAS BIBLIOGRÁFICAS

A. Chalmers, T.A. Davis, y E. Reinhard. *Practical Parallel Rendering*. 2002.

ABVENT . Artlantis : the fastest 3D rendering application developed especially for architects and designers. *Artlantis* 2009. [cited 2 Diciembre 2009]. Available from world wide web: <http://www.artlantis.com/>.

Alias Systems Corp. mental ray network rendering: Satellite and standalone. 2010. [cited 26 Enero 2010]. Available from world wide web: <http://download.novedge.com/Brands/Alias/Helps/Maya6.5/en_US/Rendering/mentalraynetworkrenderingsatelliteandstandalone.html>.

ALTASIS. Home | YafaRay. *YafaRay* 2009. [cited 2 Diciembre 2009]. Available from world wide web: <http://www.yafaray.org/>.

Aneli Valdes Acosta, y María de Dolores Guardia Macias . Sistema automatizado para la gestión de horarios docentes. . 2007.

Autodesk. Autodesk - 2D and 3D Design and Engineering Software for Architecture, Manufacturing, and Digital Entertainment. 2010. [cited 5 Febrero 2010]. Available from world wide web: <http://usa.autodesk.com/>.

Autodesk. Autodesk - Autodesk Toxik Services & Support - Documentation. *Backburner* 2009. [cited 3 Diciembre 2009]. Available from world wide web: <http://images.autodesk.com/adsk/files/backburner_2008.1_install_guide.pdf>.

Busy Images Inc. BusyRay Project. *BusyRay* 2009. [cited 2 Diciembre 2009]. Available from world wide web: <http://www.busyray.com/>.

Carlos Billy Reynoso. Introducción a la Arquitectura de Software. 2004. [cited 17 Febrero 2010]. Available from world wide web: <http://www.microsoft.com/spanish/msdn/arquitectura/roadmap_arq/intro.mspx#EAG>.

Cebas VISUAL TECHNOLOGY Inc. Welcome to cebas VISUAL TECHNOLOGY Inc. *Final Render* 2009. [cited 1 Diciembre 2009]. Available from world wide web: <http://www.finalrender.com/>.

Cediant. www.rendergrid.es - Grid Render. 2009. [cited 27 Noviembre 2009]. Available from world wide web: <http://www.rendergrid.es/index.php?option=com_content&task=view&id=37&Itemid=62>.

Chaos Group. VRay.info | Front Page. *V-Ray* 2009. [cited 1 Diciembre 2009]. Available from world wide web: <http://www.vray.info/>.

David Padrón Alvarez. Modelo de Negocio basado en Software Libre para la producción de audiovisuales en Cuba.
Junio 2009, 64.

Diego Gomez. Como documentar la arquitectura de software. 2008. [cited 17 Febrero 2010]. Available from world wide web: <http://www.dosideas.com/noticias/metodologias/298-como-documentar-la-arquitectura-de-software.html>.

DrQueue. Welcome to DrQueue Commercial Website. 2010. [cited 26 Enero 2010]. Available from world wide web: <http://www.DrQueue.org/cwebsite/>.

Formworks. Lobo's Blender Page. 2010. [cited 26 Enero 2010]. Available from world wide web: <http://farmerjoe.info/>.

Fox Animation. fox animation. 2010. [cited 18 Mayo 2010]. Available from world wide web: <http://www.tu.tv/videotag/fox-animation>.

Gladys Marsi. Metodología ágil para proyectos de software libre. 2008.

Glare Technologies Limited. Indigo (re)presents light | Indigo Renderer. *Indigo* 2009. [cited 2 Diciembre 2009]. Available from world wide web: <http://www.indigorenderer.com/>.

I. Buck, T. Foley, D. Horn, J. Sugerman, K. Fatahalian, M. Houston, P. Hanrahan. *Brook for GPUs: Stream Computing on Graphics Hardware.* . 2004.

I. Victor Kerlow. *The art of 3D computer animation and imaging. John Wiley & Sons, 2000*. 2000.

Ian Main, Tony Gale. GTK Tutorial. 1998. [cited 2 Mayo 2010]. Available from world wide web: <http://www.linuxlots.com/~barreiro/spanish/gtk/tutorial/gtk_tut.es.html>.

llluminate Labs. Turtle — llluminate Labs. *Turtle* 2009. [cited 2 Diciembre 2009]. Available from world wide web: <http://www.illuminatelabs.com/products/turtle>.

Integra Inc. INSPIRER : Simulation software for lighting design and analysis. *Inspirer* 2009. [cited 2 Diciembre 2009]. Available from world wide web: <http://www.integra.jp/en/inspirer/index.html>.

J. David Baker. ScreamerNet Controller for OS X. 2010. [cited 26 Enero 2010]. Available from world wide web: <http://www.catalystproductions.cc/screamernet/>.

José Angel Mateos Ramos. Yafrid-NG: Mejora de un sistema Grid Computacional para el render de escenas 3D. 2007, 63.

Josep Casanovas. Usabilidad y arquitectura del software. 2004. [cited 17 Febrero 2010]. Available from world wide web: <http://www.desarrolloweb.com/articulos/1622.php>.

Kevin Beason. Pane - a ray tracer by Kevin Beason. *Pane* 2009. [cited 2 Diciembre 2009]. Available from world wide web: <http://www.kevinbeason.com/scs/pane/>.

kinetic vision. kinetic vision. 2010. [cited 5 Febrero 2010]. Available from world wide web: <http://kinetic-vision.com/>.

Kristoffer Berg (MUX medialab). Sunflow - Global Illumination Rendering System. *Sunflow* 2009. [cited 2 Diciembre 2009]. Available from world wide web: <http://sunflow.sourceforge.net/>.

Loki Render. Loki Render. *Loki 0.6.2 released* 2010. [cited 26 Enero 2010]. Available from world wide web: <http://loki-render.berlios.de/>.

LuxRender. LuxRender - introduction. *LuxRender* 2009. [cited 2 Diciembre 2009]. Available from world wide web: <http://www.luxrender.net/>.

Mental Images GmbH. mental images: Home. *Mental Ray* 2009. [cited 28 Noviembre 2009]. Available from world wide web: <http://www.mentalimages.com/index.php>.

Next Limit. Maxwell Render :: The next generation in rendering technology capable of simulating light exactly as it behaves in the real world. *Maxwell Render* 2009. [cited 1 Diciembre 2009]. Available from world wide web: <http://www.maxwellrender.com/>.

Nvidia Corporation. Noticias: NVIDIA Gelato Pro GPU potencia la prestación de software libremente disponible ahora - Noticias y Comunicados de Prensa - Creative COW.

2008. [cited 2 Febrero 2010]. Available from world wide web: <http://geo.creativecow.net/es/th/105/859939/1>.

NVIDIA Corporation. NVIDIA Gelato. *Nvidia Gelato* 2009. [cited 2 Diciembre 2009]. Available from world wide web: <http://www.nvidia.es/page/gelato.html>.

Oliver Schulze. Yadra – Yet Another Distributed Rendering App | BlenderNation. 2010. [cited 26 Enero 2010]. Available from world wide web: <http://www.blendernation.com/yadra-yet-another-distributed-rendering-app/>.

Persistence of Vision Raytracer. POV-Ray - The Persistence of Vision Raytracer. *Pov-Ray* 2009. [cited 2 Diciembre 2009]. Available from world wide web: <http://www.povray.org/>.

Pixar. Pixar Animation Studios. 2010. [cited 18 Mayo 2010]. Available from world wide web: <http://www.pixar.com/>.

Pixar. Pixar's RenderMan®. *Renderman* 2009a. [cited 2 Diciembre 2009]. Available from world wide web: <https://renderman.pixar.com/>.

Pixar. Pixie - Open Source RenderMan. *Pixie* 2009b. [cited 2 Diciembre 2009]. Available from world wide web: <http://www.renderpixie.com/>.

Pixar . Aqsis Renderer : Freedom to Dream. *Aqsis* 2009. [cited 2 Diciembre 2009]. Available from world wide web: <http://www.aqsis.org/>.

Plohman, Angela. Website Statics. Blender.org April 2009. [cited 6 May 2009]. Available from world wide web: <http://www.blender.org/blenderorg/blender-foundation/press/website-statistics/>.

Powua. PurePowua | Cloud computing made easy. 2010. [cited 4 Febrero 2010]. Available from world wide web: <http://www.powua.com/>.

Prime Focus. VFX Software. | Prime Focus. 2010. [cited 26 Enero 2010]. Available from world wide web: <http://www.primefocusworld.com./services/products/vfx-software>.

Ranch Computing. Ranch Computing. 2010. [cited 4 Febrero 2010]. Available from world wide web: <http://www.ranchcomputing.com/I_benchmark_results_uk.php>.

RandomControl. fryrender. *Fryrender* 2009. [cited 2 Diciembre 2009]. Available from world wide web: <http://randomcontrol.com/index.php?option=com_content&view=article&id=3&Itemid=16>.

rdelvalle@macuare.org. Apoyo para render proyecto libre Historias Nativas. Diciembre 2009.

Rebus. Render Farm 3d Renderfarm Rebusfarm Render Farm Service. 2010. [cited 4 Febrero 2010]. Available from world wide web: <http://www.rebusfarm.com/>.

Remote-Render. Main. 2010. [cited 4 Febrero 2010]. Available from world wide web: <http://www.remote-render.com/>.

RenderCore. RenderCore.com online render farm. 2010. [cited 4 Febrero 2010]. Available from world wide web: <http://www.rendercore.com/>.

Renderfarm. renderfarm. 2010. [cited 4 Febrero 2010]. Available from world wide web: <http://www.renderfarm.es/>.

Renderfarm24. Renderfarm24 | Fast - cheap - competent. 2010. [cited 4 Febrero 2010]. Available from world wide web: <http://renderfarm24.com/en/index.php>.

renderfriend. RenderFriend - online render farm for Cinema4D, Vray, Modo, Maxwell Render, After Effects and Blender. 2010. [cited 4 Febrero 2010]. Available from world wide web: <http://www.renderfriend.com/>.

Render Nation. Render Nation - remote render farm service for 3d rendering 3dsmax, Maya & VRay. 2010. [cited 4 Febrero 2010]. Available from world wide web: <http://www.rendernation.com/>.

Render Rocket. Render Rocket Render Farm - Home. 2010. [cited 4 Febrero 2010]. Available from world wide web: <http://www.renderrocket.com/>.

RenderTITAN. RenderTITAN | Remote Render Farm Service. 2009. [cited 4 Febrero 2010]. Available from world wide web: <http://www.rendertitan.com/main>.

RocketTheme. Kerkythea - About Kerkythea. *Kerkythea* 2009. [cited 2 Diciembre 2009]. Available from world wide web: <http://www.kerkythea.net/joomla/index.php?option=com_content&task=view&id=21&Itemid=57>.

RUP. El Proceso Unificado de Desarrollo de Software (RUP). 2010. [cited 17 Febrero 2010]. Available from world wide web: <http://yaqui.mxl.uabc.mx/~molguin/as/RUP.htm>.

Sara Alvarez. Arquitectura cliente-servidor. 2007. [cited 18 Febrero 2010]. Available from world wide web: <http://www.desarrolloweb.com/articulos/arquitectura-cliente-servidor.html>.

Side Effects Software Inc. . cmiVFX Rendering Class - Side Effects Software Inc. *Mantra* 2009. [cited 2 Diciembre 2009]. Available from world wide web:

<http://www.sidefx.com/index.php?
option=com_content&task=view&id=1567&Itemid=66>.

Shoran Software. RenderPal V2 | Render Management System :: News. 2010. [cited 26 Enero 2010]. Available from world wide web: <http://www.renderpal.com/>.

SiTex Graphics, Inc. AIR. *Air* 2009. [cited 2 Diciembre 2009]. Available from world wide web: <http://www.sitexgraphics.com/html/air.html>.

Sony. Website oficial de PlayStation. 2010. [cited 18 Mayo 2010]. Available from world wide web: <http://es.playstation.com/>.

Sourceforge.net. Distriblend. 2010. [cited 26 Enero 2010]. Available from world wide web: <http://netblend.sourceforge.net/distriblend/>.

Syoyo Fujita. lucille | Global Illumination Renderer. *Lucille* 2009. [cited 2 Diciembre 2009]. Available from world wide web: <http://lucille.sourceforge.net/>.

T. Hachisuka. *High-Quality Global Illumination Rendering using Rasterization. GPU Gems 2: Programming Techniques for High Performance Graphics and General-Purpose Computation. Addison-Wesley Professional.* 2005.

The toxic Project. The toxic Open Source Renderer. *Toxic* 2009. [cited 2 Diciembre 2009]. Available from world wide web: <http://www.toxicengine.org/>.

Trinity Animation Inc. . Brazil r/s V2 - Brazil Renderer. *Brazil* 2009. [cited 1 Diciembre 2009]. Available from world wide web: <http://www.brazilrenderingsystem.com/>.

Virtual Vertex. Virtual Vertex - Home page. 2009. [cited 26 Enero 2010]. Available from world wide web: <http://www.vvertex.com/index.html>.

Wikipedia. YafRay - Wikipedia, la enciclopedia libre. *YafRay* 2009. [cited 2 Diciembre 2009]. Available from world wide web: <http://es.wikipedia.org/wiki/YafRay>.

Worley Labs. FPrime Overview. *FPrime* 2009. [cited 2 Diciembre 2009]. Available from world wide web: <http://www.worley.com/E/Products/fprime/fprime.html>.

BIBLIOGRAFÍA

A. Chalmers, T.A. Davis, y E. Reinhard. *Practical Parallel Rendering*. 2002.

Carlos González Morcillo, Gerhard Weiss, Luis Jiménez Linares David Vallejo Fernández, y Francisco Javier Albusac Jiménez. *Optimización del proceso de render 3D distribuido con software libre*. 2007, 41-48.

David Padrón Alvarez. Modelo de Negocio basado en Software Libre para la producción de audiovisuales en Cuba. Junio 2009, 64.

I. Buck, T. Foley, D. Horn, J. Sugerman, K. Fatahalian, M. Houston, P. Hanrahan. *Brook for GPUs: Stream Computing on Graphics Hardware.* . 2004.

I. Victor Kerlow. *The art of 3D computer animation and imaging. John Wiley & Sons, 2000*. 2000.

José Ángel Mateo Ramos. Yafrid-NG: Mejora de un sistema Grid Computacional para el render de escenas 3D. 2007.

NVIDIA Corporation. NVIDIA Gelato. *Nvidia Gelato* 2009. [cited 2 Diciembre 2009]. Available from world wide web: <http://www.nvidia.es/page/gelato.html>.

Rudrajit Samanta, Thomas Funkhouser, Kai Li, y Jaswinder Pal Singh. *Hybrid Sort-First and Sort-Last. Parallel Rendering with a Cluster of PCs*. 2000. [cited 2 Febrero 2010]. Available from world wide web: <www.cs.princeton.edu/~funk/gh2k.pdf>.

T. Hachisuka. *High-Quality Global Illumination Rendering using Rasterization. GPU Gems 2: Programming Techniques for High Performance Graphics and General-Purpose Computation. Addison-Wesley Professional.* 2005.

ANEXOS

Anexo 1: Motores de render comerciales y Open Source.

Motores Comerciales:	
Nombre:	*Descripción:*
Brazil	Elaborado por la empresa SplutterFish es un sistema avanzado de renderización de iluminación global y raytrace de gran calidad para 3ds MAX y VIZ, permite obtener efectos cáusticos. (Trinity Animation Inc. 2009)
Maxwell render	Es un motor de render que trabaja con luz físicamente realista, sus algoritmos y ecuaciones reproducen todos los comportamientos de la luz. Es desarrollado por la compañía Next Limit. (Next Limit 2009)
Fprime	FPrime es un motor de renderizado con un enfoque de diseño exclusivo. A diferencia de extracción de grasas tradicionales, FPrime no es sólo un paso de salida final. En su lugar, es su compañero durante todo el proceso de creación de la configuración inicial, a través de texturas e iluminación, fotogramas clave, y por supuesto, el renderizado final. Se revoluciona el flujo de trabajo al darle una retroalimentación constante sobre la escena actual y el efecto de los cambios. Siempre se puede ver a tus render. (Worley Labs 2009)
Mantra	Es el motor nativo de Houdini. Es un poderoso motor de renderizado, uno de los mejores en la industria. En muchos aspectos es muy parecido a RenderMan. Houdini fue usado para renderizar una amplia variedad de efectos en el filme Monsters vs Aliens, de DreamWorks Animation. (Side Effects Software Inc. 2009)
Renderman	El original, incluye gestores de render y es ampliamente usado en los estudios de post-producción Norteamericanos. Pixar RenderMan Pro Server ® es el mismo software que Pixar utiliza para renderizar sus propias películas de animación. Tiene un alto rendimiento de renderizado diseñado para hacer frente a los 3D más complicados con la producción de tecnología reforzado que es estable, fiable y flexible. Pixar RenderMan Pro Server ® ofrece resultados impresionantes. (Pixar 2009a)
Indigo	Indigo es un motor de render fotorrealistas que simula la física de la luz para alcanzar el realismo de una imagen. La incorporación de un avanzado modelo de la cámara física, unos super-realista materiales del sistema y la capacidad de simular situaciones de iluminación extremadamente compleja a través de Metropolis Light Transport,

	Indigo Renderer es capaz de producir los más altos niveles de realismo exigido por la arquitectura y la visualización de productos.(Glare Technologies Limited 2009)
Fryrender	Fryrender es el motor de render desarrollado por RandomControl. Todos los elementos implicados en el render (materiales, luces, cámaras, etc) se basan en modelos físicamente exactos hasta el punto de que cuando de hacer una simulación del comportamiento real de la luz que pasa por dentro. Por otra parte, fryrender es el primer motor de su clase capaz de transformar escenas en un formato que se puede navegar en tiempo real con la ayuda de los RC5, tecnología de Realidad Virtual. (RandomControl 2009)
Air	AIR es un avanzado motor de render con una arquitectura única y amplias características diseñadas para la producción rápida de imágenes de alta calidad. AIR es un procesador híbrido, que combina las ventajas de la prestación scanline - renderizado rápido de escenas complejas, efectos de movimiento y la profundidad de campo - con la flexibilidad de la demanda-ray en el seguimiento para la reflexión precisa, sombras suaves, iluminación global y cáusticos. AIR es totalmente multiproceso en Windows y Linux. AIR es compatible con el estándar de RenderMan ® para la descripción de escenas 3D. (SiTex Graphics, Inc 2009)
Artlantis	Artlantis es el más rápido independiente la aplicación de renderizado 3D desarrollado especialmente para arquitectos y diseñadores, ideal para crear rápida y fácilmente versiones de alta resolución en 3D, QuickTime VR Panorámicas, QuickTime VR de objetos y animaciones. Artlantis es el software de renderizado usado por arquitectos, diseñadores y profesionales del diseño urbano en más de 80 países. (ABVENT 2009)
Inspirer	Inspirer es un paquete de software para la simulación de la iluminación física exacta y la renderización de imágenes. Inspirer puede utilizarse eficazmente en las oficinas de arquitectura, por la iluminación, fabricantes de automóviles y el sector aeroespacial. Al presentar Inspirer en el proceso de diseño permite a los usuarios reproducir virtualmente los espacios y objetos y para simular los efectos de iluminación para el análisis de las características de la iluminación. (Integra Inc 2009)
POV-Ray™	El POV-Ray (*Persistence of Vision Raytracer*) es de alta calidad, una herramienta totalmente libre para la creación de tres impresionantes gráficos en tres dimensiones. Está disponible en versiones oficiales para Windows, Mac OS / Mac OS X y Linux i86. El código fuente está

	disponible para aquellos que quieran hacer sus propios puertos. (Persistence of Vision Raytracer 2009)
Turtle	Turtle es el único producto en el mercado que combina una avanzada tecnología de renderización con la funcionalidad de hornear en un sofisticado conjunto de herramientas integradas. Es solo para maya. (Illuminate Labs 2009)
Motores Open Source	
Nombre:	*Descripción:*
Lucille	Motor de render de código abierto de iluminación global paralela compatible con las especificaciones técnicas de Renderman. (Syoyo Fujita 2009)
Pane	Pane es un motor de render basado en *global illumination* y *ray tracer* (ver Algoritmos de Renderización). Pane es un trabajo en progreso. Las características básicas son: * Mapeado de fotones y de Monte-Carlo camino trazado * Lee inventor / archivos VRML 1.0 * *Octree-R Ray-triangle* intersección triángulo (Kevin Beason 2009)
Toxic	Toxic es un motor de render que usa la iluminación global con el fin de producir imágenes fotorrealistas y animaciones. La meta de Toxic es ofrecer a los artistas que cuenten con una herramienta gratuita, lo que hace de gran alcance que es mantenido activamente, desarrollado y ampliado. Toxic es un proyecto de código abierto, disponible en SourceForge, y licenciado bajo la licencia GPL. (The toxic Project 2009)
Sunflow	Sunflow es un motor de render de código abierto para la foto-síntesis de imágenes realistas. Está escrito en Java y construido alrededor de un trazado de rayos flexibles básico y un objeto extensible de diseño orientado. (Kristoffer Berg (MUX medialab) 2009)
Pixie	Pixie es un renderizador de codigo abierto RenderMan para generar imágenes fotorrealistas. Puede compilar Pixie en Windows (Visual Studio 2005), Linux y OSX (utilizando XCode o Unix script de configuración de estilo). (Pixar 2009)
Kerkythea	Kerkythea es un procesador independiente, utilizando materiales con precisión física y luces, con el objetivo para la prestación de la mejor calidad en el plazo más eficiente. El objetivo de Kerkythea es simplificar la tarea de hacer la calidad, proporcionando las herramientas necesarias para automatizar la configuración de escena, como la puesta

	en escena utilizando el real GL-visor de tiempo, editor de materiales, General / Configuración de renderizado editores, etc, en virtud de una interfaz común. (RocketTheme 2009)

Anexo 2: Compañías que se dedican al renderizado.

Nombre	*Descripción*
RANCH Computing	En RANCH, cada proyecto de Maxwell Render funciona en 512 núcleos de Intel @ 3 GHz para un rendimiento sin precedentes. Especial de hardware y optimizaciones de software le permiten fusionar 256 GB de datos de alto rango dinámico (archivos MXI) en tan sólo 20/25 minutos: perfecta para proyectos de MultiLight! RANCH ofrece: - Capacidad para reanudar los proyectos de cooperación. - Capacidad para llegar a un nivel de muestreo de cooperación (SL). - Un pase de eliminación de ruido se aplica automáticamente al render. - Capacidad para renderizar varias vistas de la misma escena con una sola carga (Multicam). - El Blitz! Fórmula para los procesos de prueba en baja resolución. - Aparece la lista de espera en tiempo real una aproximación del nivel de la corriente mundial de muestreo. - Vistas previas de imágenes en tiempo real permiten verificar visualmente su proyecto mientras se está representado. (Ranch Computing 2010)
Remote-Render	Compañía de render especializada en 3ds Max y V-Ray. Ubicada en Costa Rica. Cuenta con una granja de render compuesta por: * 140 Intel Q6600, 2.4 GHz Quads (Total: 1.267 GHz combinado) * 8 GB de RAM por nodo de render * Windows XP Pro 64 en todos los nodos * 3ds Max y Vray rendering de 64 bits (Remote-Render 2010)
Render Rocket	Render Rocket es un servicio de granja de render 3D con apoyo en Maya, Mental Ray, 3ds Max, V-Ray, Cinema 4D y Maxwell Render. Este servicio se utiliza para una sola imagen o una película de animación entera. Tiene opciones de instalación rápida. Su visión

	es entregar el poder de representación a nivel profesional a la carta a cualquier equipo de producción en 3D, independientemente de su tamaño o ubicación en el planeta. Su misión es que la solución de la prestación a distancia sea lo más accesible y poderoso como cualquier local de granja de render. (Render Rocket 2010)
RenderCore	Ubicado en Los Ángeles, California, Estados Unidos. Rendercore proporciona una alta velocidad de la prestación de servicios a distancia diseñado exclusivamente para el artista gráfico de computadora. Tiene la automatización total del sistema de renderizado remoto. Cuenta con un análisis automático de escena, es fácil de trabajo y control de la presentación y tiene la automatización de carga y descarga para archivos. Esta compuesto por: • 6,5 Tera Hz Capacidad. • 500 CPUs de 8Core o 16Core Cpu • 16 GB de memoria Soporta Maya, Brazil, VRay, XSI, 3D Studio Max, LightWave, Cinema4D y Turtle. Trabaja con los sistemas operativos: A. linux (32,64bit) B. windows (32,64bit) C. macosx (32,64bit) (RenderCore 2010)
Renderfarm24	Debido a una configuración de 64 bits esta granja de renderizado puede manejar fácilmente escenas “pesadas” con millones de polígonos y texturas de alta resolución. En esta granja las escenas son revisadas por miembros del equipo con experiencia antes de que se realice el renderizado final. Esto asegura que no hay problemas comunes como las texturas que faltan, apoderados o no los ajustes de procesamiento aplicada. De encontrar un problema se ponen en contacto inmediato con el cliente para resolver las cuestiones conjuntamente. (Renderfarm24 2010)
RenderFriend	RenderFriend es una filial de fotogramas clave Multimedia, Inc., un productor de imágenes 3D en sí mismo y para comprender las necesidades y preocupaciones de otros estudios en 3D. Soporta los software Maxon Cinema4D, Luxology MODO, Adobe After Effects CS4, Blender 3D.

	Granja de render compuesta por: *Más de 100 procesadores Xeon de ejecutar al menos 3.0 Ghz * Los procesadores totalmente homogénea * 8 GB de RAM en todas las estaciones de trabajo * 2 terabytes de almacenamiento para los proyectos * Fuente de alimentación redundante * Conectividad de red redundante * Conexión por switch Gigabit segmentado * Gran potencia de UPS 5000VA de APC * Windows Server 2003 @ 64-bit (renderfriend 2010)
Renderfarm.es	Esta granja de render esta compuesta por servidores de alto rendimiento dedicados al cálculo intensivo las 24h, equipados con dual Intel Xeon serie 5400 (8 núcleos por servidor) y 8 Gb de memoria DDR2. Todas las máquinas están conectadas mediante una red Gigabit Ethernet. Cada nodo renderiza una secuencia de imágenes de forma distribuida o mediante «split-frame», lo que permite calcular el mismo frame usando varias máquinas, ideal para imágenes de muy alta resolución. Dispone de los siguientes programas y motores de render. ◆3ds Max 2010 *(V-Ray 1.50SP4a, mental ray & RPC)* ◆Maya 2009 SP1a *(vector, mental ray)* ◆LightWave 3D 9.6 *(G2 & FPrime 3.50)* ◆Cinema 4D R11.5 ◆Dependiendo de la disponibilidad, se suelen asignar más de 250 GHz de potencia de cálculo. (Renderfarm 2010)
Alternate Perspective 3D	Radica en Inglaterra. Ofrece el servicio de render con el motor de Lightwave. Publica los resultados parciales a sus clientes a través de un área segura en la red, y el resultado final a través de la misma web o enviando las imágenes en otro soporte de almacenamiento. Ofrecen facilidades de render a través de la red a proyectos de corto y largo tiempo. La red de computadoras para hacer el render posee procesadores Intel y en todas está instalado el Lightwave 3D Screamnet. Si el cliente lo necesita, pueden hacer el renderizado con 3D Studio Max o SOFTIMAGE Mental Ray.
Autodesk	Es uno de los servicios de granja de render más profesionales de los que se brindan actualmente en la industria audiovisual. Brindan

	acceso a un equipo de consultores de gran experiencia con la profundidad de conocimientos necesarios para ofrecer una amplia gama de soluciones especializadas que incluyen: programación de soluciones de render distribuido, desarrollo de aplicaciones web, programación de scripts personalizados, administración de proyectos, evaluaciones de software y hardware, administración de la granja de render, administración de sistemas, así como servicios de soporte y formación. (Autodesk 2010)
Kinetic Vision	Fue fundado en 1992, en Ohio, Estados Unidos. Actualmente brindan un servicio de renderizado donde además de entregar las imágenes por separado, las unen en un video formando la animación, e incluyen música y transiciones. Actualmente soportan 3D Max, VIZ 2005, Vue5, Mental Ray, V-Ray y RPC. (kinetic vision 2010)

Figura 1: Granja de Render Pixar

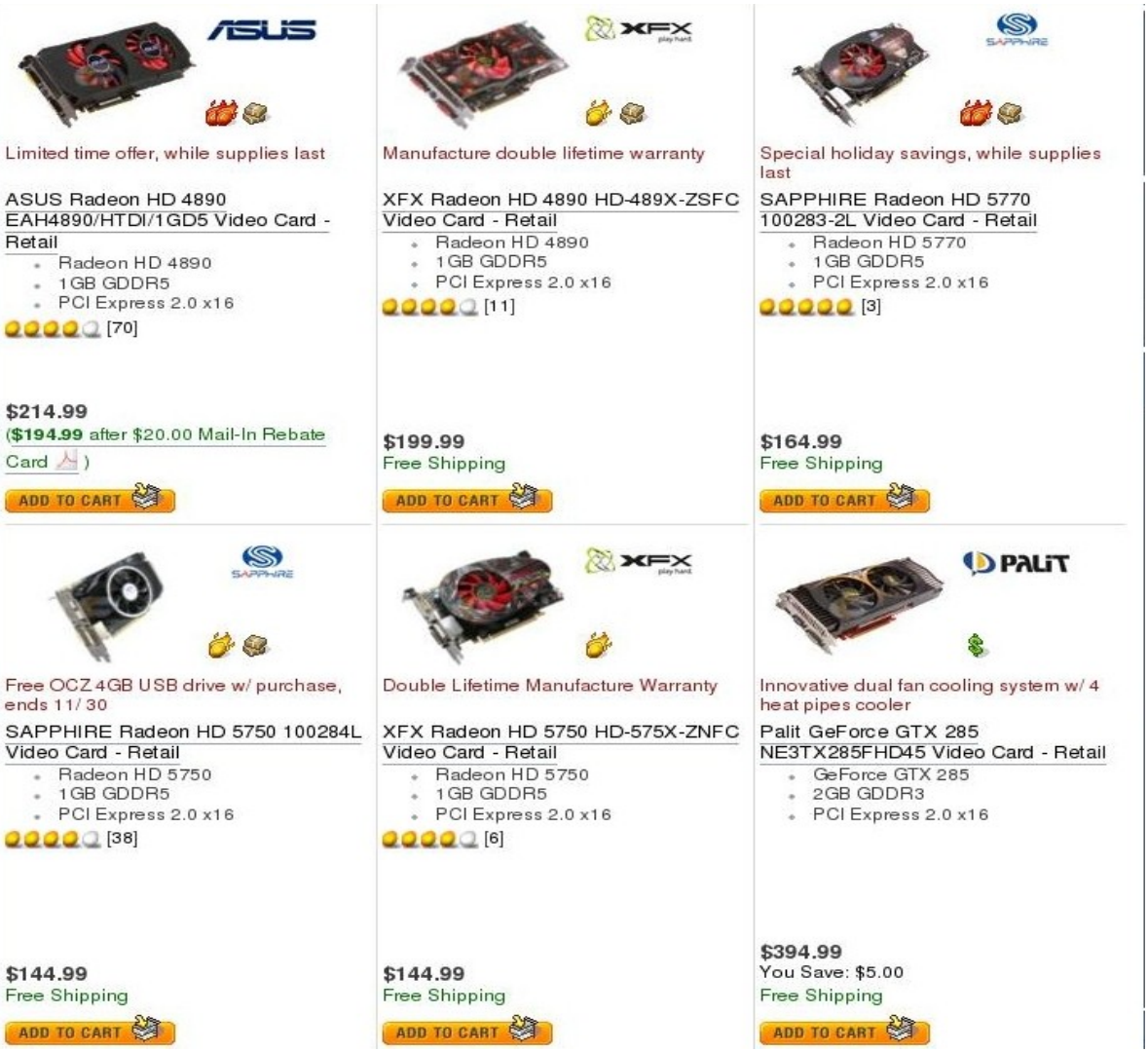


Figura 2. Precios de algunas GPUs (en dólares).

Fuente: www.newegg.com [cited 28 Noviembre 2009]

Figura 3. Escena del filme 2012 renderizado con FinalRender

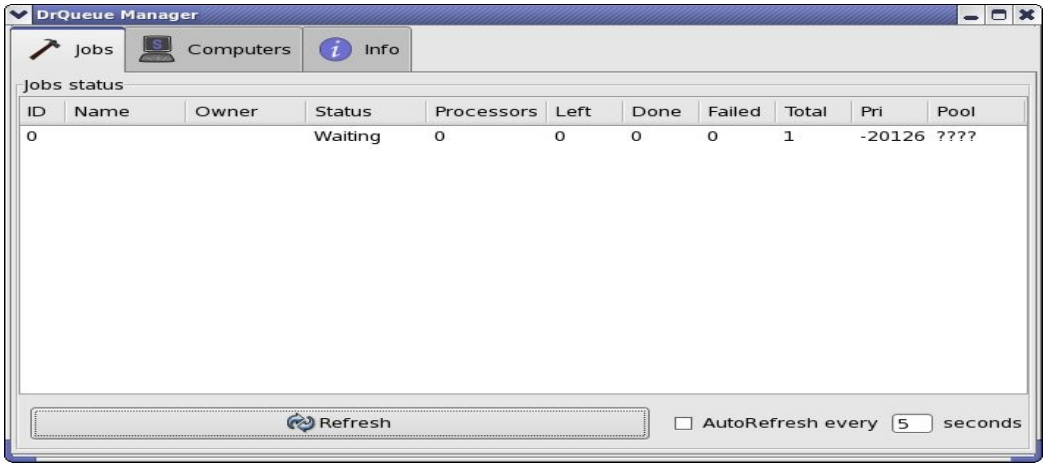


Figura 4. Cilente de DrQueue “Drqman”

Figura 5. Escena del filme Avatar. Para su renderización se usó Ubuntu como sistema operativo.

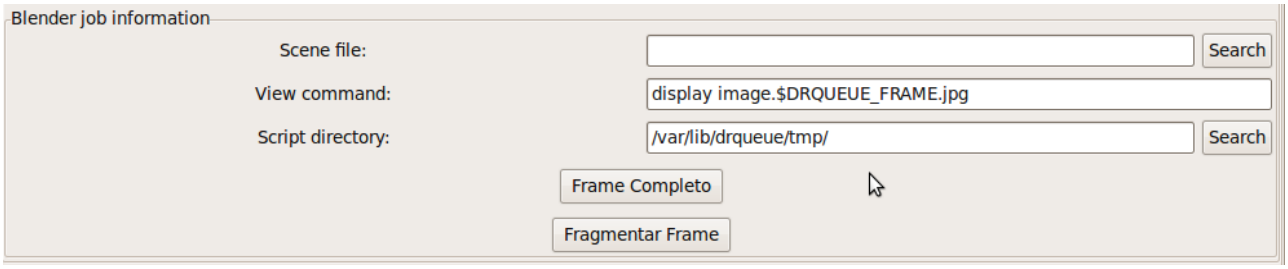


Figura 6. Renderización por frame completo y fragmentado.

Correo del productor de Historias Nativas

Estimados amigos de FreeViUX.

Les escribimos desde Venezuela los realizadores de la producción Historias Nativas, un proyecto libre en animación 3D sobre leyendas indígenas para niños, realizado con Blender y otras herramientas

libres, el cual será liberado bajo licencia Creative Commons. Información general sobre este proyecto la pueden obtener en http://historiasnativas.macuare.org.

David Rodriguez, uno de los autores de "El Pequeño Gran Libro", y quien ahora forma parte del equipo de trabajo de Historias Nativas, nos facilitó el contacto con ustedes.

El presente es para consultarles sobre la posibilidad de hacer el renderizado en la granja de render de FreeViUX del primer capítulo, de aproximadamente 3 min, el cual aspiramos culminar durante el transcurso de esta semana.

Un placer haber hecho contacto con ustedes,

Saludos,

Rómulo Del Valle.
Productor General de Historias Nativas.
Fundación Macuare. (rdelvalle@macuare.org 2009)

GLOSARIO DE TÉRMINOS

Modelo OSI: El modelo de referencia de Interconexión de Sistemas Abiertos (OSI, Open System Interconnection) fue el modelo de red descriptivo creado por la Organización Internacional para la Estandarización lanzado en 1984. Es decir, fue un marco de referencia para la definición de arquitecturas de interconexión de sistemas de comunicaciones.

Pixar: Pixar Animation Studios es una compañía especializada en la producción de gráficos en 3D y con sede en Emeryville, California (Estados Unidos).

Sony Computer Entertainment: Es una empresa multinacional dedicada a los videojuegos y subsidiaria de Sony Corporation. Fue fundada el 16 de noviembre de 1993 en Minato (Tokio), con la intención de lanzar al mercado la videoconsola PlayStation un año después.

Computación Distribuida: La computación distribuida o informática en malla, es un nuevo modelo para resolver problemas de computación masiva utilizando un gran número de computadoras organizadas en racimos incrustados en una infraestructura de telecomunicaciones distribuida.

Cáustico: En óptica, se dice que una superficie es cáustica cuando es tangente a los rayos que se reflejan o se refractan por un sistema óptico.

Infografía: La infografía es una representación más visual que la propia de los textos, en la que intervienen descripciones, narraciones o interpretaciones, presentadas de manera gráfica normalmente figurativa, que pueden o no coincidir con grafismos abstractos y/o sonidos. La infografía nació como un medio de trasmitir información gráficamente. Los mapas, gráficos, viñetas, etc. son infogramas, es decir unidades menores de la infografía, con la que se presenta una información completa aunque pueda ser complementaria o de síntesis.

Refracción: La refracción es el cambio de dirección que experimenta una onda al pasar de un medio material a otro. Sólo se produce si la onda incide oblicuamente sobre la superficie de separación de los dos medios y si éstos tienen índices de refracción distintos. La refracción se origina en el cambio de velocidad que experimenta la onda. El índice de refracción es precisamente la relación entre la velocidad de la onda en un medio de referencia (el vacío para las ondas electromagnéticas) y su velocidad en el medio de que se trate.

Un ejemplo de este fenómeno se ve cuando se sumerge un lápiz en un vaso con agua: el lápiz parece quebrado. También se produce refracción cuando la luz atraviesa capas de aire a distinta temperatura, de la que depende el índice de refracción. Los espejismos son producidos por un caso extremo de refracción, denominado reflexión total.