



“Clúster de renderizado para la generación de modelos tridimensionales”

Proyecto que presenta

Omar Ordóñez Toledo
Matricula: 02611036

Como requisito final para el curso de Desarrollo de proyecto de Campo

Clave: ED09505

Maestría en Gestión de Tecnologías de Información

Profesor: Mtro. Mario Saavedra Raya

Morelia, Michoacán

16/08/2011

Índice

CONTENIDO

CONTENIDO.....	2
ÍNDICE DE TABLAS	4
ÍNDICE DE GRÁFICOS.....	5
ÍNDICE DE IMÁGENES	5
ÍNDICE DE ECUACIONES	5
RESUMEN.....	6
INTRODUCCIÓN.....	7
1.- PROBLEMA DE INVESTIGACIÓN.....	10
1.1 Contexto y Justificación	10
1.2 Objetivos	20
1.2.1 General:	20
1.2.2 Específicos:	20
1.3 Alcance.....	21
1.4 Viabilidad.....	21
1.5 Planteamiento del Problema.....	21
2. MARCO TEÓRICO.....	22
2.1 Conceptos Generales de Renderizado	22
2.1.1 Renderizado	22
2.2.2. Parámetros de render:.....	24
2.2.3. Animación:.....	25
2.2.4. Fotograma:	25
2.2.5. Gráficos de computadora (Computer Graphics, CG):	25
2.2.6. Imagen Fotorrealistas:	26
2.2.7. Modelado:	26
2.2.8. Modelos 3D:	26
2.2.9. Recorrido Virtual:.....	27
2.2.10. Plug-in:.....	27

2.2.11. Parámetros de render.....	27
2.2 Investigaciones Previas	27
2.2.1 REDAG3D: Renderizado Distribuido de un Ambiente Gráfico 3D.....	27
2.2.1.1 Pruebas de Renderizado	30
2.2.2. Yafrid: sistema de renderizado basado en Grid	33
2.2.2.1. Esquema general de la arquitectura	33
2.2.3. Un sistema difuso para la optimización del renderizado	38
2.2.4. Optimización mediante Hardware	40
3. VARIABLES DE ESTUDIO	41
3.1 Identificación de variables potenciales.....	41
3.2 Desarrollo teórico de variables potenciales	43
3.2.1. Motor de renderizado.....	43
3.2.2 Cantidad de Fotogramas.....	44
3.3.3. Técnicas de Renderizado.....	46
3.3.4.1. Scanline	46
3.3.4.2. Raytracing	47
3.3.4.3. Pathtracing	48
3.3.4.4. Photon Mapping	49
3.3.4.5. Pathtracing Bidireccional	49
3.6 Definición de variables.....	50
4. HIPÓTESIS	52
4.1 Hipótesis General	52
4.2 Hipótesis de Trabajo	52
5. DISEÑO DE LA INVESTIGACIÓN Y TRABAJO EMPÍRICO	53
5.1 Factores y nivel de experimentación	53
5.2 Diseño Experimental	54
5.3 Resultados	56
5.5 Análisis de datos y resultados.	58
5.6 Contrastación de hipótesis	59
6. PROPUESTA DE INTERVENCIÓN	60
6.1 Estado de la cuestión	61
6.1.1. BackBurner renderizado por lotes utilizando 3D's MAX	61

6.1.2. Optimización utilizando computación distribuida.....	62
6.2 Modelo de Propuesta	63
6.2.1. Software Open Mosix.....	67
6.2.2. Características de Open Mosix	68
6.2.3. Componentes necesarios para construir el clúster son:	69
6.3 Inversión y periodo de recuperación.....	70
7. CONCLUSIONES.....	72
8. RECOMENDACIONES	74
BIBLIOGRAFÍA.....	76

ÍNDICE DE TABLAS

Tabla 1. Tiempos de renderizado en la creación de una imagen 3D.	15
Tabla 2. Tiempos de renderizado en la creación de una imagen con 3D MAX.	16
Tabla 3.- Costo de Servicios Tecnológicos y costo de penalización.	20
Tabla 4.- Variables Independientes.....	42
Tabla 5.- Nombres de las Variables y Frecuencia.	43
Tabla 6.- Definición de Variables	51
Tabla 7- Descripción de Variables.....	52
Tabla 8.- Factores establecidos para realizar el experimento.....	54
Tabla 9.- Datos Obtenidos del experimento.....	55
Tabla 10.- Tabla de resultados del tiempo de renderizado	59
Tabla 11.- Tabla Contrastación de hipótesis.	60
Tabla 12. Herramientas para render en red más utilizadas.	65
Tabla 13.- Costo de implementación del clúster de renderizado.....	70
Tabla 14.- Costo total de penalizaciones.	71

ÍNDICE DE GRÁFICOS

Gráfico 1.- Tiempos de renderizado en la creación de una imagen 3D.	15
Gráfico 2.- Tiempos de renderizado en la creación de una imagen con 3D MAX.	16
Gráfico 3.- Diagrama de Pareto de las Variables Independientes	42
Gráfico 4.- Efectos principales de tiempo.	56
Gráfico 5.- Semi normal de los efectos	57
Gráfico 6.- Intervalos de experimentos.	58

ÍNDICE DE IMÁGENES

Imagen 1.- Renderizado Distribuido	29
Imagen 2 Sort-Last y Sort-First usando JNDI	30
Imagen 3.- Malla que ocupa gran parte del viewport (16 regiones)	31
Imagen 4.- Sort-Last con JNDI (24 máquinas)	32
Imagen 5.- Sort-First con JNDI (24 máquinas)	32
Imagen 6.- Capas que forman la arquitectura del servidor Yafrid	34
Imagen 7.- Flujo de Trabajo de y principales roles en la arquitectura	37
Imagen 8.- Requisitos para implementar el clúster de renderizado.	66

ÍNDICE DE ECUACIONES

Ecuación 1.- Fórmula para calcular el número de fotogramas necesarios para crear movimiento.	45
---	----

RESUMEN

El último paso en el proceso para la generación de imágenes y animaciones 3D por computadora es el llamado render. En esta fase se genera una imagen bidimensional o un conjunto de imágenes en el caso de las animaciones a partir de la descripción de una escena 3D. Para la obtención de imágenes fotorrealistas se utilizan algoritmos computacionalmente que exigen demasiado procesamiento de cómputo, en el presente trabajo se realiza una investigación cuyo objetivo es identificar las causas que determinan los factores que determina el tiempo en renderizar un modelo tridimensional, en el desarrollo teórico considero algunos autores como: **Carlos González Morcillo, I. Foster, C. Kesselman, S. Tuecke**, pues ellos coinciden en que la etapa de render se considera el cuello de botella debido al tiempo de renderizado necesario para llevar a cabo este proceso, por otro lado las variables potenciales fueron, motor de render, cantidad de fotogramas y técnica de renderizado, en donde generalmente, el problema se resuelve usando granjas o clúster de render sector en las que los frames de una animación se distribuyen en distintas computadoras. Como propuesta de solución el presente proyecto ofrece una alternativa para atacar el render tanto de animaciones como de imágenes basada en computación grid o clúster. Para ello, se ha desarrollado un sistema en el que tienen cabida computadoras heterogéneas tanto en software como en hardware.

INTRODUCCIÓN

Se puede entender el proceso de Render como el mecanismo utilizado para obtener una imagen a partir de la descripción de una escena tridimensional. Actualmente dicho proceso ha alcanzado un alto grado de realismo, mientras que el gran problema sigue siendo el tiempo de ejecución empleado para el mismo.

Se puede entender el proceso de síntesis de **imagen fotorrealistas** como aquel proceso que persigue la creación de imágenes sintéticas que no se puedan distinguir de aquellas captadas en el mundo real. Dicho proceso está dividido en las siguientes fases: **modelado, asignación de materiales y texturas, iluminación, y render.**

La última de las fases mencionadas, es decir, la que se refiere al proceso de renderizado, es la que determina el color más apropiado que se asigna a cada pixel de la escena, y cuyo resultado es una imagen que representa la escena tridimensional. Dicho color vendrá determinado por diversos factores, como por ejemplo la geometría del objeto, la posición, el color de las fuentes de luz, la posición y orientación de la cámara, las propiedades de las superficies, etc.

La fase de renderizado ha sufrido una gran evolución a lo largo del tiempo. En los primeros años de estudio de este campo, la investigación centró su foco en cómo resolver problemas básicos, como por ejemplo la detección de superficies visibles, o el sombreado básico. Conforme el tiempo iba pasando y dichos problemas se iban solventando, el estudio se fue centrando en la creación de algoritmos de síntesis más realistas, que simularan el comportamiento de la luz de la forma más fidedigna posible. Entender la naturaleza de la luz y cómo se dispersa sobre el entorno es esencial para simular correctamente la iluminación.

La etapa de renderizado, dentro del proceso de síntesis de imagen fotorrealistas, supone uno de los principales cuellos de botella en lo que a desarrollo se refiere. La razón se debe a que los algoritmos empleados requieren una gran cantidad de **procesamiento de cómputo**. Además, estos algoritmos han sido ya muy estudiados, por lo que resulta complicado conseguir mejoras sustanciales en lo que a resultados se refiere. Por lo tanto, resulta interesante plantear métodos que reduzcan el tiempo empleado en el proceso de renderizado desde otro enfoque, es decir, desde otro punto de vista que no se centre en el método de simulación.

Un aspecto que resulta interesante resaltar, ya en lo relativo a los parámetros que intervienen en la etapa de renderizado, es que existen ciertos parámetros cuyos valores hacen que el tiempo empleado en el renderizado se incremente, mientras que la diferencia de calidad con respecto a valores distintos de esos parámetros resulta inapreciable.

Visto de otro modo, existen ciertas configuraciones que hacen que el tiempo final de renderizado sea alto, sin mejorar sustancialmente la calidad del resultado final en lo referente a la percepción del usuario.

En el presente trabajo se aborda la problemática referente al tiempo de renderizado y la forma de cómo se puede disminuir este tiempo mediante el uso de diversas técnicas para dicho proceso. Este documento está estructurado de la siguiente forma:

Apartado 1. Se aborda el contexto donde se llevará a cabo el proyecto, hace referencia a la problemática que supone la fase de render en los proyectos relacionados con la síntesis de imagen 3D. Así mismo se declaran los objetivos, tanto generales como específicos, que se persiguen con la realización de este proyecto y la viabilidad del proyecto.

Apartado 2. Se hace una revisión de los Antecedentes y estudio de aquellas áreas que guardan relación con el proyecto, haciendo mención de aquellos autores más relevantes en esta área.

Apartado 3. Se definen las variables de estudio, tanto variable dependiente como independientes con la finalidad de mejorar los procesos de renderizado de imágenes y que estos puedan optimizar el tiempo que tarda en realizar dicho proceso.

Apartado 4. En este apartado se establecen las hipótesis tanto general como de trabajo que se generaron de acuerdo a las variables independientes y a su respectivo análisis desde la parte teórica y su relación con la variable dependiente Y= Tiempo de Renderizado.

Apartado 5. En este apartado se realiza el análisis de los datos que incluye el resumen por variable, los factores y nivel de experimentación, resumen de resultados así como la contrastación de hipótesis en donde se muestran las hipótesis aceptadas o rechazadas.

Apartado 6. Aquí se describe la propuesta de intervención, en donde se desarrolla una solución tecnológica para la solución de la problemática presentada, se narra el estado de la cuestión, el modelo de la propuesta en donde se muestra el funcionamiento tanto operativo como técnico del clúster de renderizado propuesto así como la inversión requerida y el periodo de recuperación.

Por último en este documento se encuentran los apartados de conclusiones y recomendaciones de la investigación presentada.

1.- PROBLEMA DE INVESTIGACIÓN

1.1 Contexto y Justificación

En 1990 la Secretaría de Educación Pública emprendió una serie de estudios y análisis de experiencias académicas en varias naciones para definir un modelo pedagógico de educación superior vinculado con nuestro contexto socioeconómico, como resultado se crea el Sistema de Universidades Tecnológicas, cuya característica fundamental es su relación con el sector productivo y con la sociedad. (Coordinación General de Universidades Tecnológicas, 2011).

Como consecuencia de lo anterior, se concibió un sistema de educación tecnológica superior que prestara servicios al sector productivo de bienes y servicios, así como a la sociedad en general y que, al mismo tiempo, ampliara las expectativas de los jóvenes mexicanos. Este sistema se materializó en lo que hoy conocemos como Universidades Tecnológicas, las cuales ofrecen el Título de Técnico Superior Universitario y actualmente ofrecen a su vez Títulos de Ingeniería.

La Universidad Tecnológica de Morelia se basa en este modelo educativo, ofrece:

- **Un sistema polivalente**, lo que quiere decir que en tu formación profesional abarcarás uno o varios grupos de actividades generales aplicables a todas las ramas del sector productivo de bienes y servicios, de tal forma que al egresar tengas la capacidad de adaptarte a diferentes formas de trabajo.
- **Continuidad**, como egresado puedes continuar estudios de Licenciatura o Ingeniería en otras Instituciones, contando para ello con un gran avance al revalidarte las materias ya cursadas.
- **Intensidad**, el plan de estudios se imparte en 2 años cubriendo 3,000 horas de estudio en promedio.

- **Flexibilidad**, cuenta con planes y programas de estudios flexibles que se revisan y adaptan continuamente a las necesidades del sector productivo de la zona de influencia de la Universidad.
- **Eficacia**, permite que los planes y programas de estudio estén en relación con las necesidades reales de la planta.

La Universidad Tecnológica de Morelia es un organismo público descentralizado que pertenece al Sector Educativo Superior y Medio Superior del Estado de Michoacán de Ocampo. Entre sus atribuciones están las siguientes:

- Formar técnicos superiores universitarios y/o ingenieros a partir de egresados de Bachillerato, en las carreras de Mantenimiento Industrial, Biotecnología, Diseño y Producción Textil y Tecnologías de Información y Comunicación;
- Proporcionar educación superior tecnológica, de acuerdo a los planes y programas de estudios, carreras y modalidades educativas establecidas por la Coordinación General de Universidades Tecnológicas, para garantizar una formación profesional y una cultura científica y tecnológica adecuada;
- Organizar y desarrollar programas de intercambio académico y colaboración profesional con organismos e instituciones culturales, educativas, científicas o de investigación nacionales y extranjeras;
- Establecer la organización administrativa necesaria y contratar los recursos humanos adecuados para su operación, de conformidad con las disposiciones aplicables, el presupuesto de la Universidad y su Decreto de creación;
- Adoptar su estructura orgánica básica, de conformidad con los lineamientos que para tal efecto se expidan;
- Planear y programar la enseñanza e incorporar en sus planes y programas de estudio, los contenidos particulares, locales o regionales necesarios que contribuyan a satisfacer las necesidades de la comunidad, elevando su calidad de vida;

- Proponer los planes y programas de estudio, así como sus adiciones o reformas a la aprobación del Consejo Directivo y de la autoridad correspondiente;
- Planear y desarrollar programas de actualización y superación académica, dirigidos a los miembros de la comunidad universitaria y de la población en general;
- Reglamentar los procedimientos de selección, ingreso y permanencia de los alumnos en la institución, de conformidad con lo dispuesto para el consejo Directivo;
- Regular los procedimientos de ingreso, permanencia y promoción, en su caso, del personal académico atendiendo las recomendaciones que sugieran en el seno del Consejo Directivo;
- Expedir constancias, certificados de estudio, títulos, diplomas, reconocimientos, distinciones especiales que se requieran, conforme a las disposiciones aplicables;
- Establecer los procedimientos y requisitos de acreditación y certificación de estudios, de conformidad con las disposiciones aplicables;
- Promover la congruencia del calendario escolar con el aprobado por la autoridad educativa competente;
- Implementar programas de investigación y vinculación;
- Organizar actividades culturales y deportivas que permitan a la comunidad el acceso a las diversas manifestaciones culturales y contribuir al desarrollo integral del educando;
- Impulsar estrategias de participación y concertación con los sectores público, privado y social para la realización de actividades productivas, con los más altos niveles de eficiencia y sentido social; y
- Las demás necesarias para el cumplimiento de su objeto.

La Universidad Tecnológica de Morelia, se basa en un modelo educativo de 70% práctica y 30% teoría que es una de sus fortalezas frente a sus competidores.

Para poder lograr lo anterior, la universidad cuenta con infraestructura adecuada como aulas, laboratorios que apoyan a la realización de prácticas en cada una de las carreras, además de contar con un centro de información en el cual los usuarios pueden realizar consultas por Internet en la sala de consulta electrónica o bien consultar libros especializados en la biblioteca.

Una de las bondades de este sistema es la atención personalizada que se le brinda al alumno a través de un tutor, el cual es asignado a cada grupo y él es el encargado de dar seguimiento académico a cada alumno, el alumno cuenta con el apoyo de los técnicos académicos para la realización de prácticas en los diferentes laboratorios con los que cuenta la Universidad además el alumno tiene el derecho de recibir asesoría tanto por parte del profesor como del tutor asignado, lo cual ayuda a su crecimiento académico.

En las Universidades Tecnológicas se imparten programas educativos con duración de dos años, Nivel 5B, con los que se obtiene el título de Técnico Superior Universitario.

La Universidad desde su decreto de creación inició una serie de estudios de mercado laboral, dando como resultado definir las carreras profesionales requeridas por el sector productivo. Derivado de los estudios de viabilidad se concluyó como necesaria la implementación de cuatro carreras: Biotecnología, Diseño y Producción Industrial, Mantenimiento Industrial y Tecnologías de la Información y Comunicación, Actualmente y derivado de las tendencias y necesidades de contar con nuevos programas educativos que satisfagan la demanda de los usuarios se contempla que para el ciclo escolar 2011-2012 se integren dos carreras nuevas: Gastronomía y Energías renovables, carreras que sin lugar a duda harán de la Universidad Tecnológica de Morelia una de las mejores opciones de estudio para los jóvenes.

En la carrera de Tecnologías de Información y Comunicación, se ofertan dos especialidades Sistemas Informáticos y Multimedia y Comercio Electrónico, esta última cuenta con laboratorios de computo en donde se imparten materias de especialización como Multimedia I, Multimedia II, Desarrollo de Aplicaciones Web y Animación 3D, las cuales requieren de equipo de computo especializado para realizar aplicaciones multimedia.

Uno de los principales problemas al trabajar con estas materias consiste en el tiempo que tarda el crear la escena final en 3D, es decir construir cada fotograma de la animación o película en 3D, dicho proceso puede tardar dos o tres minutos tomando en cuenta requerimientos de baja calidad, por cada segundo de película animada necesitamos al menos 24 cuadros y si tomamos en cuenta que un cortometraje puede durar minutos entonces una sola PC tardaría semanas o meses durante todo el día construyendo los cuadros ya que esto requiere una potencia de cálculo demasiado elevada lo que resulta generar el lento proceso de renderización para conseguir los resultados finales deseados.

Aunado a esto con las nuevas tecnologías como la alta definición que requiere de mayores características para obtener el resultado final. El tiempo de *render* depende en gran medida de los parámetros establecidos en los materiales, luces y texturas utilizadas, así como de las configuraciones del programa de renderizado.

En la **tabla 1** se muestran los tiempos empleados en renderizar una misma escena de prueba de poca complejidad en distintas máquinas. La imagen resultado que se ha obtenido es de resolución aproximada de (720x576), en formato **NTSC** (National Television System Committee) (Finder, 2010). El motor de render utilizado ha sido Vray, utilizando una de las técnicas que obtienen resultados fotorrealistas.

Procesador	Memoria RAM	SO	Tiempo
Core Duo	2 Gb	Leopard	12 minutos
Intel Pentium4 2.8 GHz	1 Gb	Windows XP	19 minutos
Intel PIII 667 MHz	256 Mb	Windows XP	64 minutos
AMD Athlon 1 GHz	256 Mb	Windows XP	78 minutos
Intel Pentium4 1.8 GHz	512 Mb	Windows XP	38 minutos

Tabla 1. Tiempos de renderizado en la creación de una imagen 3D.

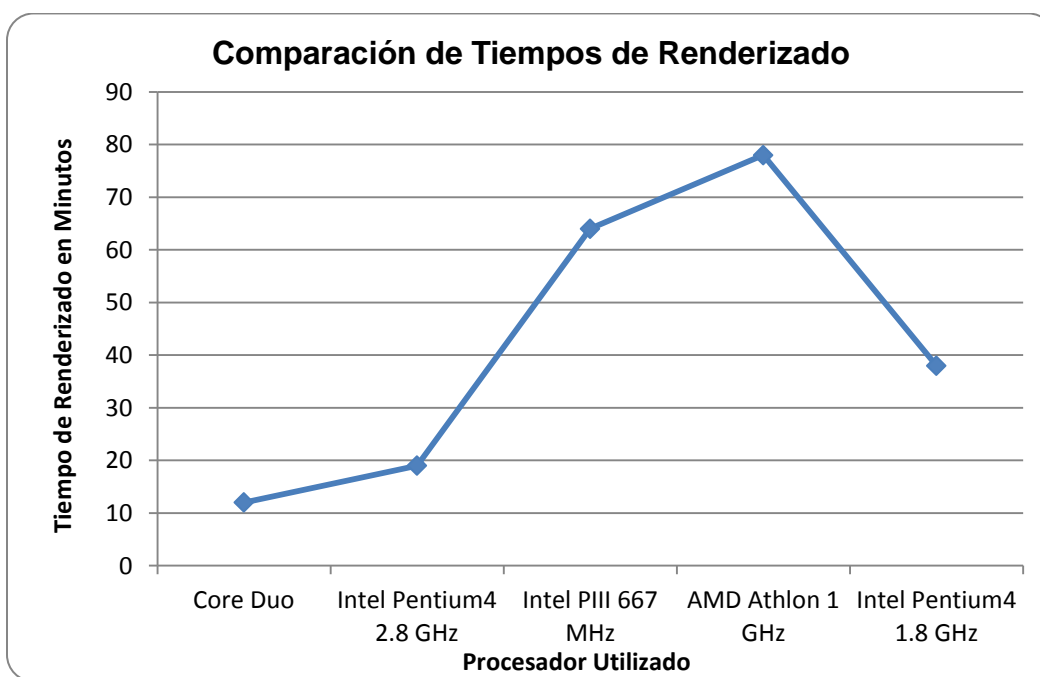


Gráfico 1.- Tiempos de renderizado en la creación de una imagen 3D.

En la **tabla 2** se muestran los tiempos que se tardó en renderizar una escena de prueba de poca complejidad en distintas máquinas. La imagen resultado que se ha obtenido es de resolución aproximada de (640 x 480), usando parámetros propios del software 3D MAX, como son reflexión, desplazamiento, bump, transparencia, sombras de área e iluminación global. El motor de render utilizado ha sido Vray.

Procesador	Memoria RAM	SO	Tiempo
2 xeon	2G	Win XP	4.7 min
Portátil Dual core 2.1 Ghz	1G	Win XP	4.10 min
Pentium 4 2.8 Ghz	2G	Win XP	5.6 min
PowerMac G5 Dual 2.5Ghz	4G	OSX Tiger	5.36 min
Athlon XP 2600+,	1G	Win XP	11.17 min
AMD Athlon 1.2 GHZ	1G	Win XP	22.30 min
Portatil eMac G4	768Mb	OSX	34.43 min

Tabla 2. Tiempos de renderizado en la creación de una imagen con 3D MAX.

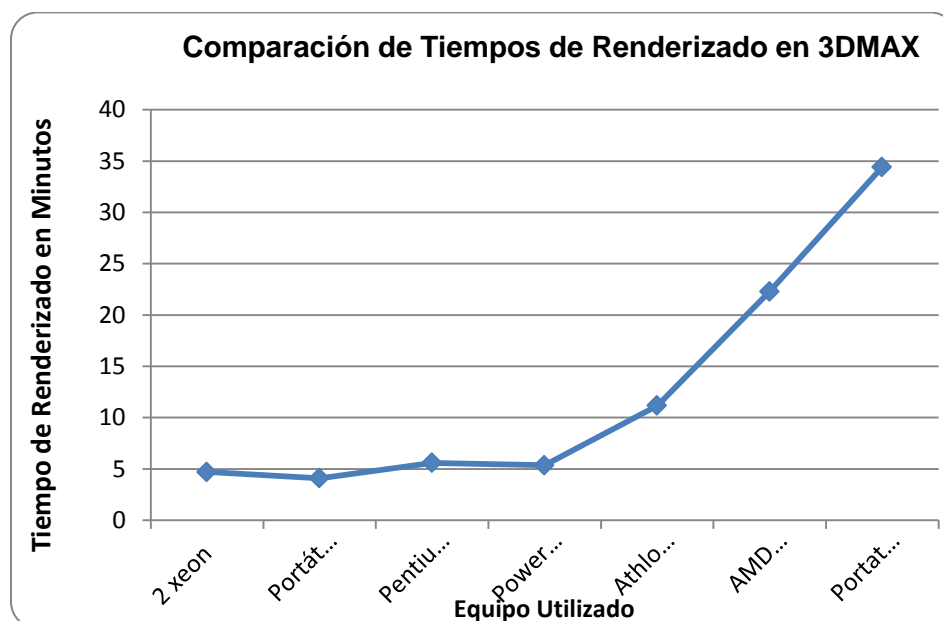


Gráfico 2.- Tiempos de renderizado en la creación de una imagen con 3D MAX.

Una vez hechas estas pruebas se tiene como conclusiones lo siguiente: para dar la sensación de movimiento, las imágenes se tendrían que mostrar a una periodicidad de 24 fotogramas por segundo ante el usuario. De este modo, para un cortometraje de dos minutos de duración habría que obtener 3000 fotogramas, en cuyo render se invertirían 36,000 minutos usando el primero de los equipos de la tabla 1 y 2. Por lo anterior y en base a las pruebas que se han llevado a cabo

podemos determinar, que el principal problema en este tipo de proyectos es claramente el tiempo de respuesta del procesador. (Electrónico, 2010)

Por lo tanto la utilización de computadoras hoy en día se ha vuelto una práctica común en las actividades de la vida diaria, y debido al gran número de tareas que se tienen que procesar, se vuelve necesario contar con equipo de cómputo de mayor capacidad técnica.

Para que además con la incorporación de nuevas tecnologías permitan realizar este trabajo con mayor rapidez sin la necesidad de sacrificar aspectos como la velocidad de proceso y la calidad de los resultados. La incorporación de dispositivos de cómputo con mayores capacidades implica realizar un gasto excesivo, factor que hace que muchas organizaciones vean limitadas sus expectativas de realizar proyectos de mayor relevancia.

Existen varias razones para utilizarlo una de ellas es en el diseño y desarrollo de hardware o software, en el ámbito de la animación 3D que es el tema trascendental de este trabajo, el uso de **técnicas de renderizado** de modelos tridimensionales y recorridos virtuales por medio de imágenes reales, requieren de mayor velocidad de procesamiento para obtener un alto rendimiento o mayor velocidad de ejecución del programa. (flytech, 2007)

En esencia, las técnicas de simulación de modelos matemáticos y la tecnología computacional actual ofrecen una combinación excelente que ayudan a hacer realidad los conceptos de manipulación, exploración y descubrimiento. Los medios múltiples con que cuenta la computadora permiten agregar claridad; sin embargo, se ha explorado poco sobre la integración de métodos didácticos, técnicas de simulación y facilidades de cómputo; es aquí en donde nuestro proyecto tendría un alcance mayor.

De lo anterior podemos determinar que en la Universidad Tecnológica de Morelia, se determinaron las **siguientes problemáticas** respecto al tiempo en que tarda el renderizado de imágenes o la creación de video:

1. En cuarto y quinto cuatrimestre los alumnos tienen en su retícula las materias de **Desarrollo de Aplicaciones WEB, Modelado 3D, Multimedia I y Multimedia II**, materias que ocupan de computadoras con mayores requerimientos tanto de hardware como de software para la realización de prácticas y trabajos de modelado y animación, por lo que al momento de realizar los renderizados se tarda mucho en procesar estas actividades, por lo que es necesario que los equipos de computo se queden prendidos por horas e inclusive hasta días enteros, lo que ocasiona que cuando se utiliza el laboratorio de multimedia para impartir otras clases esas computadoras no se puedan utilizar por que están realizando un proceso de renderizado por lo que se considera tiempo muerto; otro problema es el aumento en el consumo de energía y en ocasiones la descompostura de los equipos.
2. La Universidad Tecnológica de Morelia, dentro de sus actividades ofrece servicios tecnológicos, es decir, ofrece servicios de capacitación, consultorías y últimamente se está ofertando el servicio de realización de modelos tridimensionales y recorridos virtuales; lo que conlleva a que los tiempos de entrega de los productos anteriormente mencionados en ocasiones no se puedan determinar con precisión, debido al tiempo que tardan los equipos en realizar el proceso de renderizado, lo que incurre en la penalización por parte de los contratantes o hasta la cancelación del proyecto.

Para calcular el costo de renderizado de un proyecto tomamos en cuentas las siguientes variables:

1. **Punto de vista**, se debe de determinar si es exterior o interior
2. **Iluminación**, se debe de definir el tipo de iluminación que se utilizará ya sea diurna o nocturna.
3. **Tipo de Diseño**, minimalista o de estilo
4. **Grado de Definición**, si es un renderizado de tipo esquemático o hiperrealista

5. **Entorno**, establecer si es un modelado es decir creación de figuras a base de prismas y polígonos o si el modelo es en base a un foto modelado; es decir modelar todo el entorno.

Tomando en cuenta estas variables para elaborar un modelo tridimensional y recorrido virtual, el primer paso es construir una maqueta digital en base a polígonos, mediante la utilización del procesamiento de equipo de computo, para lo cual son necesarios los planos de plantas, fachadas, cortes y otros detalles especiales; enseguida se busca el punto óptimo para visualizar la mejor perspectiva del modelo, cabe señalar que esto se hace sin la utilización de materiales y sombras, una vez que es aprobado el punto de vista se procede a realizar el armado de la escena con vegetación, accesorios, gente, vehículos, movimientos, materiales, transparencias, luces, etc. Por lo tanto podemos decir que el costo por segundo de renderizado es de \$250.00, si el renderizado tiene una duración de 1 minuto el costo por este proceso es de \$15,000.00.

Respecto a las penalizaciones se tiene contemplado que en caso de no entregar en los tiempos establecidos, el cliente haga una penalización del 20% del precio total del producto renderizado, lo que conlleva a la universidad a tener pérdidas por la no entrega del producto.

La justificación de este trabajo reside en utilizar un enfoque distinto a la hora de abordar la etapa de renderizado, con el objetivo principal de reducir el tiempo empleado para ello. Debido a que existe una gran comunidad de desarrollo en el diseño y creación de modelos tridimensionales y recorridos virtuales se refiere, proyectos como este son importantes porque las mejoras conseguidas afectan beneficiosamente a un gran número de personas. De hecho, existen muchas empresas y entidades que utilizan mecanismos de representación realista, como centros de investigación científica, empresas de publicidad, y entidades relacionadas con el sector del ocio (videojuegos. películas de animación y recorridos virtuales).

En la siguiente tabla se muestran algunos de los proyectos que se han realizado como parte de los servicios tecnológicos que ofrece la Universidad Tecnológica de Morelia, así como su costo, el tiempo de renderizado y la penalización por proyecto, cabe señalar que estos proyectos no se entregaron a tiempo debido al proceso de renderizado, en donde la penalización es del 20% sobre el costo total del proyecto.

Proyecto	Duración del Proyecto	Costo total del Proyecto	Costo de Penalización	Diferencia
1	7 min	\$105,000	\$ 21,000	\$ 84,000
2	4.5 min	\$67,500	\$13, 500	\$ 54,000
3	13 min.	\$ 195,000	\$ 39,000	\$ 156,000

Tabla 3.- Costo de Servicios Tecnológicos y costo de penalización.

1.2 Objetivos

1.2.1 General:

Identificar los factores que determinan el tiempo de renderizado de modelos 3D y recorridos virtuales.

1.2.2 Específicos:

- Identificar teóricamente los factores que determinan el tiempo de renderizado y Validar empíricamente los factores teóricos que determinan el tiempo de renderizado.
- Realizar experimentos que nos ayuden a validar los conceptos teóricos planteados.

1.3 Alcance

- El trabajo presentado es de tipo explicativo y la investigación que se llevará a cabo es de tipo experimental, de tal forma que los resultados no pueden ser generalizados fuera de las condiciones del experimento realizado.
- Tendrá implementación en el laboratorio de Multimedia de la Universidad Tecnológica de Morelia.
- Si el proyecto es factible, se pretende implementarlo en otras Instituciones Educativas.
- Los resultados obtenidos no podrán ser replicados fuera de la Universidad Tecnológica de Morelia.

1.4 Viabilidad

Es viable realizar este proyecto debido a que es una institución pública descentralizada, De lo anterior podemos deducir que para la realización del proyecto se cuenta con toda la información necesaria para su elaboración, es decir se cuentan con las estadísticas del tiempo en que se tarda en renderizar un modelado 3D en los distintos equipos de computo de la Institución Educativa, se tienen los elementos que se involucran para hacer un modelo tridimensional, como son iluminación, calidad, texturas y animación y se cuenta con el personal con el conocimiento adecuado para la realización de modelos y recorridos virtuales, además de contar los equipos necesarios y requerimientos de hardware para la implementación y elaboración del clúster de renderizado.

1.5 Planteamiento del Problema

¿Cuáles son los factores que determinan el tiempo de renderizado en la elaboración de un modelo tridimensional?.

2. MARCO TEÓRICO

2.1 Conceptos Generales de Renderizado

En esta sección se presentan algunos conceptos y definiciones relacionados con el tema de renderizado de imágenes para su mejor comprensión. A continuación se transcriben algunas de las definiciones que se acercan a la idea del significado de renderizado:

2.1.1 Renderizado es un término usado en informática para referirse al proceso de generar una imagen desde un modelo tridimensional. Este término técnico es utilizado por los animadores o productores audiovisuales y en programas de diseño en 3D.

En términos de visualizaciones por medio de una computadora, más específicamente en 3D, la renderización es un proceso de cálculo complejo desarrollado por una computadora destinado a generar una imagen 2D a partir de una escena 3D. La traducción más fidedigna es *interpretación*, aunque se suele usar el término inglés. Así podría decirse que en el proceso de renderización la computadora *interpreta* la escena en tres dimensiones y la plasma en una imagen bidimensional, mediante el **procesamiento de cálculo** de la computadora. (MEDIAactive, 2009)

Según **Castell Cebolla**, autor del libro 3D Studio Max; La renderización se aplica en la computación gráfica, más comúnmente a la infografía. En infografía este proceso se desarrolla con el fin de imitar un espacio 3D formado por estructuras poligonales, comportamiento de luces, texturas, materiales (agua, madera, metal, plástico, tela, etcétera) y animación, simulando ambientes y estructuras físicas verosímiles. Una de las partes más importantes de los programas dedicados a la infografía son los **motores de renderizado**, los cuales son capaces de realizar

técnicas complejas como radiosidad, *raytrace* (trazador de rayos), canal alfa, reflexión, refracción o iluminación global. (Castell, 2006)

Carlos González Morcillo, Catedrático de la Escuela Superior de Informática, de la Universidad de Castilla, España; presenta la siguiente definición: La etapa de render toma como entrada los elementos definidos en las fases anteriores (Modelado, Materiales y Texturas, Iluminación y Animación) y produce como salida una imagen bidimensional que representa a la escena. Dependiendo de la **Técnica de Renderizado** o método de generación, la simulación de la luz se realizará de una forma más o menos realista. El nivel de realismo va típicamente relacionado con el **tiempo de cómputo** y por el **motor de render** que se utilice. (González, 2009)

El diccionario de Informática **ALEGSA.com.ar**. Define el renderizado como: (Del inglés rendering, renderizar, renderizado, renderización o interpretación en español). La renderización es el proceso de generar una imagen (imagen en 3D o una animación en 3D) a partir de un modelo, usando una aplicación de computadora.

El modelo es una descripción en tres dimensiones de objetos en un lenguaje o estructura de datos estrictamente definidos. El modelo debería contener geometría, punto de vista, textura e información de iluminación. La imagen resultado de la renderización es una imagen digital.

La renderización se utiliza en la producción de imágenes en 3D para juegos, diseño computacional, efectos especiales del cine y la TV, etc. En el caso de los gráficos en 3D, el renderizado puede hacerse lentamente (pre-renderizado) o en tiempo real.

El pre-renderizado es un proceso computacional intensivo que es utilizado generalmente para la creación de películas y su resultado es de altísima calidad. Además, en el prerenderizado, todos los movimientos y cambios en las escenas en 3D ya fueron prefijados antes del inicio de la renderización.

En cambio, el **renderizado en tiempo real** es más usado en los juegos en 3D y suele procesarse a través de tarjetas aceleradoras de 3D, por ser un proceso sumamente pesado. En este caso, todos los movimientos y cambios en la escena son calculados en tiempo real, pues los movimientos del jugador no son predecibles.

Son millones los cálculos matemáticos que deben realizarse para procesar un modelo en 3D y resultar en una imagen renderizada. En general, en el proceso de cálculo se pueden tener en cuenta tonalidades, texturas, sombras, reflejos, transparencias, translucidez, refracciones, iluminación (directa, indirecta y global), profundidad de campo, desenfoques por movimiento, ambiente, etc. Además a todo eso hay que agregarle los distintos objetos poligonales en 3D de la escena.

Todos estos cálculos producen una simple imagen final. Por esta razón el proceso de creación de películas en 3D, necesita mucho tiempo y gran capacidad de **procesamiento computacional**. Un sólo segundo de película suele estar constituido por 24 cuadros de imagen. (ALEGSA.com.arg, 2010)

El término de **Motor de renderizado** también se utiliza en los navegadores WEB y está definido como; la parte de un navegador que toma el contenido marcado, (como XML o HTML), lo interpreta de manera visual y lo presenta visualmente a los usuarios. **Todos los navegadores web incluyen algún motor de renderizado**. El término motor de renderizado, (layout engine o rendering engine en ingles), se hizo popular cuanto **Mozilla**, desarrolló el suyo de manera libre y diferenciado del propio navegador, siendo posible de esta manera reutilizarlo para otros navegadores. Uno de los motores más importantes es: **Gecko** es un motor multiplataforma y libre originalmente desarrollado por **Netscape**. Actualmente su desarrollo es gestionado por la Fundación Mozilla. (López, 2010)

2.2.2. Parámetros de render: Conjunto de características definidas por cada software o técnica de render que especifican ciertos aspectos relativos a su implementación. Estas características afectan en gran medida al tiempo requerido por el proceso de render y a la calidad del resultado obtenido. Aunque existen

ciertas características cuyo significado es ampliamente aceptado, cada software particular define un conjunto propio de parámetros y el significado completo de cada uno de ellos es muy dependiente de cada implementación en particular, dependiente del número de muestras por píxel o profundidad en la recursividad para rayos de iluminación directa o indirecta. (video2brain, 2011).

2.2.3. Animación: Se refiere al proceso en el cual cada fotograma de una película es producido y generado individualmente, ya sea como gráfico de computadora o mediante la fotografía de una imagen dibujada, o imprimiéndole una secuencia de movimientos a un modelo y retratando aparte cada uno de estos mínimos movimientos. Cuando los fotogramas son proyectados a una velocidad de 16 o más cuadros por segundo, aparece la ilusión de un movimiento continuo, debido a la persistencia de la visión. La animación es un proceso meticuloso y difícil, que se ha aligerado con el desarrollo de la computación aplicada a esta área. (Alegsa, 2011)

2.2.4. Fotograma: es cada una de las imágenes impresionadas químicamente en la tira de celuloide del cinematógrafo o bien en la película fotográfica; por extensión también se llama de ese modo a cada una de las imágenes individuales captadas por cámaras de video y registradas analógica o digitalmente. Cuando una secuencia de fotogramas es visualizada de acuerdo a una determinada frecuencia de imágenes por segundo se logra generar la sensación de movimiento en el espectador. (ABC, 2007).

2.2.5. Gráficos de computadora (Computer Graphics, CG): Es el campo de la computación visual donde se utilizan computadoras para generar imágenes sintéticamente o para integrar las muestras de la información visual o espacial tomadas del mundo real. Los gráficos de computadora pueden expresarse en dos o en tres dimensiones: en el caso de los segundos, son los llamados, en inglés, 3D Computer Graphics, que requieren de programas especiales. Los gráficos de computadora bidimensionales son los que comprenden imágenes digitales de

modelos geométricos, textos e imágenes bidimensionales en general. (Fetter, 2005)

2.2.6. Imagen Fotorrealistas: imagen generada por computadora que trata de imitar las imágenes generadas por cámaras fotográficas mediante complejos cálculos y algoritmos matemáticos que simulan los efectos/defectos que la luz (halos, destellos) las sombras (coloreado de sombras, difusión), las texturas (aspereza, brillo, reflejos, refracción) y la radiosidad (coloreado de la luz ambiente) producen en las imágenes resultantes. (Wikipedia, Imgen Fotorrealista, 2011)

2.2.7. Modelado: La etapa de modelado consta de ir dando forma a objetos individuales que luego serán usados en la escena. Existen diversas técnicas de modelado; Constructive Solid Geometry, modelado con NURBS y modelado poligonal son algunos ejemplos. Los procesos de modelado puede incluir la edición de la superficie del objeto o las propiedades del material (color, luminosidad, difusión, especularidad, características de reflexión, transparencia u opacidad, o el índice de refracción), agregar texturas, mapas de relieve (bump-maps) y otras características.

El proceso de modelado puede incluir algunas actividades relacionadas con la preparación del modelo 3D para su posterior animación. A los objetos se les puede asignar un esqueleto, una estructura central con la capacidad de afectar la forma y movimientos de ese objeto. Esto ayuda al proceso de animación, en el cual el movimiento del esqueleto automáticamente afectara las porciones correspondientes del modelo. El modelado puede ser realizado por programas dedicados (Lightwave, Rhinoceros 3D, Maya) (Serrano, 2011)

2.2.8. Modelos 3D: Desde un punto de vista visual, es una representación esquemática visible a través de un conjunto de objetos, elementos y propiedades que, una vez procesados (renderización), se convertirán en una imagen 3D o una animación 3D. (Serrano, 2011)

2.2.9. Recorrido Virtual: Los recorridos virtuales son una forma fácil e interactiva de ver un espacio en todas las direcciones con sólo mover el Mouse ratón, por medio de las "fotografías panorámicas esféricas", que permiten observar el espacio fotografiado en 360°x180°. (Gonzalez, 2009)

2.2.10. Plug-in: Programa que puede anexarse a otro para aumentar sus funcionalidades (generalmente sin afectar otras funciones ni afectar la aplicación principal). No se trata de un parche ni de una actualización, es un módulo aparte que se incluye opcionalmente en una aplicación, en este caso es aplicable a los distintos motores de render que existen. (Techterms, 2009).

2.2.11. Parámetros de render: Conjunto de características definidas por cada software de render que especifican ciertos aspectos relativos a su implementación. Estas características afectan en gran medida al tiempo requerido por el proceso de render y a la calidad del resultado obtenido. Aunque existen ciertas características cuyo significado es ampliamente aceptado, cada software particular define un conjunto propio de parámetros y el significado completo de cada uno de ellos es muy dependiente de cada implementación en particular.

2.2 Investigaciones Previas

A continuación se presentan algunas investigaciones previas que se han realizado por diversos autores y que tienen relación con el tema de esta investigación, con la finalidad de que se comprenda mejor el tema tratado e este trabajo.

2.2.1 REDAG3D: Renderizado Distribuido de un Ambiente Gráfico 3D

La computación gráfica ha evolucionado en gran medida desde sus orígenes hasta hoy, gracias al aporte de otras líneas de investigación, tales como los

sistemas distribuidos y los mecanismos de comunicación. Actualmente existen varias arquitecturas que permiten realizar el renderizado distribuido, generalmente solo usan un mecanismo de comunicación:

Es por ello, que en este artículo se presenta una arquitectura flexible que permite realizar de forma distribuida el renderizado (proceso para generar una imagen 2D a partir de objetos 3D), el cual exige un alto consumo de recursos de procesamiento. La arquitectura permite configurar de manera dinámica tanto la técnica de renderizado como el mecanismo de comunicación.

Para realizar el renderizado distribuido de una escena 3D se deben revisar conceptos relevantes sobre sistemas distribuidos, computación gráfica y las técnicas de renderizado distribuido existentes además de conocer los elementos que incluirá el renderizado es decir movimientos, texturas, luces, etc, con el fin de determinar los requerimientos mínimos de la arquitectura de REDAG3D.

Para probar dicha arquitectura, se debe implementar una aplicación prototipo que ponga en funcionamiento cada una de las características de los componentes definidos. Adicionalmente crear una escena constituida por modelos 3D complejos, que permitan validar las características de desempeño y flexibilidad de la arquitectura. Por otro lado, la aplicación debe tener un diseño consistente que ofrezca ventajas para la implementación, mantenimiento y puesta en marcha.

Las técnicas de renderizado distribuido fueron **Sort-First** y **Sort-Last**. Sus principales diferencias radican en la forma como se distribuye la geometría y en qué etapa del proceso de renderizado se hace el ordenamiento (sort). Sort-First implica redistribuir primitivas de espacio de pantalla a cada procesador, en donde se hace una clasificación de éstas antes de la etapa de transformación geométrica.

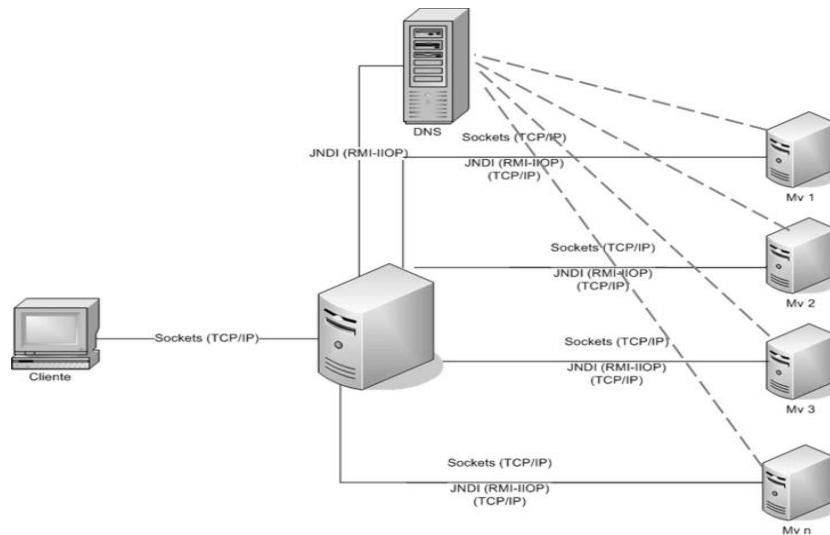


Imagen 1.- Renderizado Distribuido

En Sort-Last, las primitivas se envían a cada procesador sin una previa clasificación y se hace un ordenamiento de píxeles en la etapa de composición de la imagen final. Como mecanismos para la distribución se seleccionaron Sockets y JNDI-RMI. Sockets también fue utilizado como mecanismo de comunicación para el envío y recepción de mensajes entre las máquinas, y JNDI-RMI permitió la publicación de servicios para ser usados por la máquina que administra la aplicación.

La elección de Sockets para el envío y recepción de mensajes se basó en unas pruebas cuyos resultados fueron analizados y arrojaron como conclusión que era el mejor mecanismo para esta labor.

Por otra parte, cuando se hace referencia al renderizado distribuido es necesario contar con escenas 3D que justifiquen la presencia de varias máquinas para realizar dicho proceso, por lo cual, se decidió crear una escena lo suficientemente compleja. Esta complejidad se mide según el número de mallas que conforman los modelos 3D de la escena, el número de triángulos de las mallas, las texturas asociadas a los modelos y efectos adicionales.

Se definió un plan de pruebas, el cual evalúa los aspectos más relevantes asociados al comportamiento de las técnicas de renderizado, los mecanismos de comunicación, así como el rendimiento y la funcionalidad de los gestores de distribución en la aplicación REDAG3D. (Diana L. REYES, Alfonso BARBOSA, César J. BUSTACARA)

2.2.1.1 Pruebas de Renderizado

El propósito de las pruebas de renderizado es verificar la eficiencia de cada técnica de renderizado con el mismo mecanismo de comunicación. Para cada máquina renderizadora se tomó el tiempo de inicio y final del renderizado de las mallas asignadas, más el tiempo de envío de cada imagen al servidor administrador y el tiempo de composición de la imagen final. En el imagen 2 se muestran los resultados de estas pruebas.

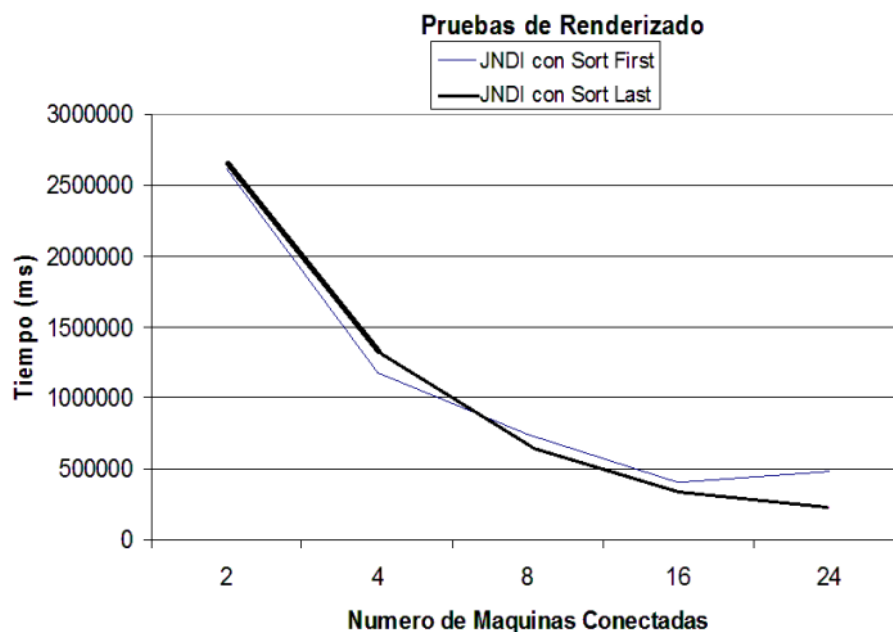


Imagen 2 Sort-Last y Sort-First usando JNDI

Como se observa en la Imagen 2, el comportamiento en general de las técnicas de renderizado es similar. Con 2 y 4 máquinas visualizadoras Sort-First es un poco más eficiente; a partir de 8 máquinas mejora el desempeño de Sort-Last. Con pocas máquinas Sort-First tiene un mejor comportamiento debido a que no se generan imágenes completas, por lo cual el envío de éstas a través de la red tarda menos tiempo que con Sort-Last. Cuando el número de máquinas es mayor a 4, el espacio de pantalla está subdividido en un gran número de regiones con Sort-First, por lo que se renderizan más de una vez mallas que ocupan varias regiones (ver Imagen 3), debido a esto se observa un mejor comportamiento con Sort-Last.

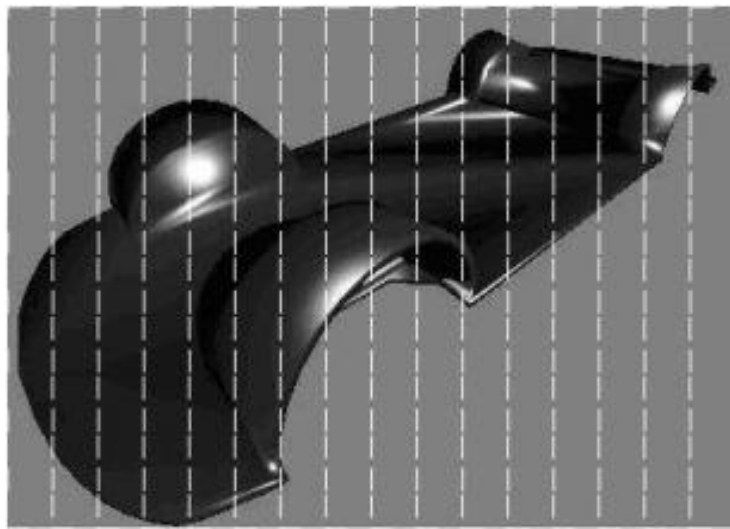


Imagen 3.- Malla que ocupa gran parte del viewport (16 regiones)

También se tuvieron en cuenta tiempos por cada máquina renderizadora de manera que se pudiera observar la distribución de la carga. En las Imágenes 4 y 5 respectivamente, se pueden observar los resultados de estas pruebas para 24 máquinas con las dos técnicas de renderizado distribuido.

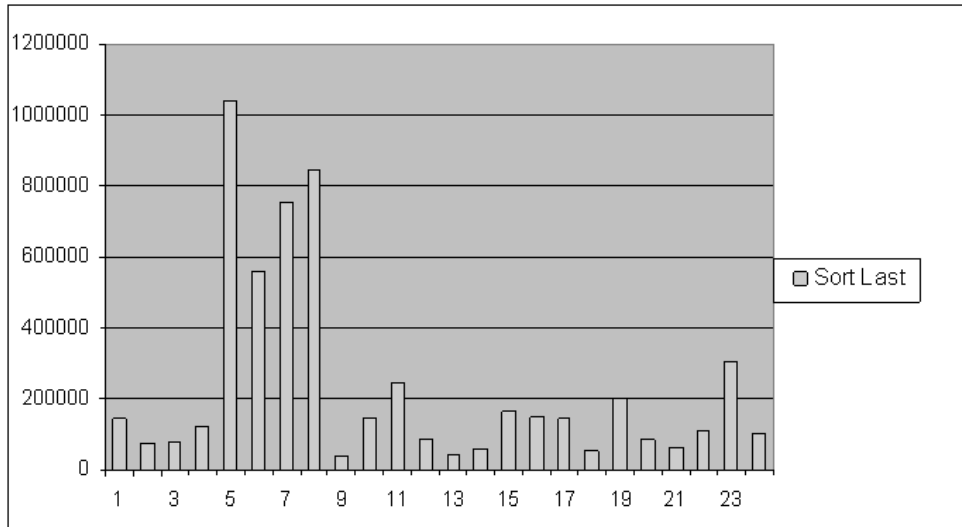


Imagen 4.- Sort-Last con JNDI (24 máquinas)

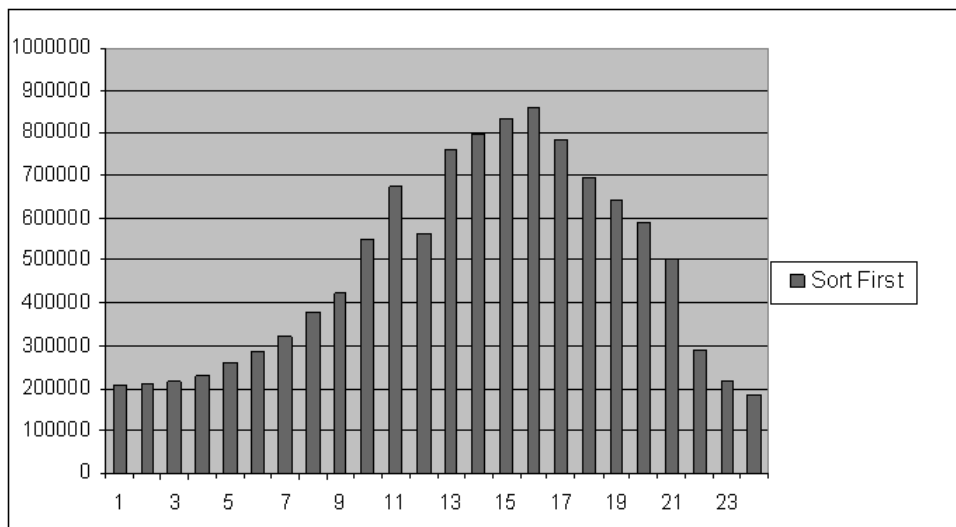


Imagen 5.- Sort-First con JNDI (24 máquinas)

Al observar los dos gráficos se puede concluir que la distribución es más uniforme con Sort-First. A las máquinas que demandaron más tiempo de renderizando les fueron asignadas una cantidad mayor de mallas, o mallas con un mayor nivel de complejidad respecto al número de triángulos.

2.2.2. Yafrid: sistema de renderizado basado en Grid

Yafrid es un sistema que aprovecha las ventajas de los sistemas de computación Grid, permitiendo la distribución de una escena a través de Internet para su renderizado. Además, el sistema soporta otras tareas importantes como son la gestión de las unidades de trabajo y el uso controlado del grid.

2.2.2.1. Esquema general de la arquitectura

Los componentes de Yafrid que se encuentran en la capa de más alto nivel de la arquitectura son:

Servidor: Es el núcleo de Yafrid. Su componente básico es el Distribuidor, encargado de recoger los trabajos de una cola y enviarlos a los proveedores.

Proveedor: Esta entidad se encarga de procesar las peticiones de los clientes.

Cliente: Un cliente es una entidad externa que no pertenece al sistema en un sentido estricto. La función principal del cliente es añadir trabajos al sistema grid, para que posteriormente sean completados por los proveedores.

A continuación se describen los tres roles de usuario utilizados en Yafrid:

- **Cliente.** Este rol permite a un usuario añadir nuevos trabajos al grid. Un cliente también puede crear y gestionar grupos asociados a los proyectos (clientes y proveedores pueden suscribirse a estos grupos). Únicamente los proveedores que pertenecen al mismo grupo pueden participar en el proceso de renderizado del proyecto.

- **Administrador.** Este usuario es necesario para operar con cualquiera de los componentes que constituye el sistema, y además, posee todos los privilegios para acceder a la información relacionada con cualquier usuario del sistema.
- **Proveedor.** El proveedor es un usuario que tiene instalado el software necesario para que el grid pueda enviarle trabajos.

Servidor Yafrid. El servidor es el nodo fundamental ya que todo el sistema gira en torno a él. Cada uno de los proveedores se conecta a este nodo para ceder al grid sus ciclos de CPU, con el objetivo de renderizar las escenas que los clientes han añadido.

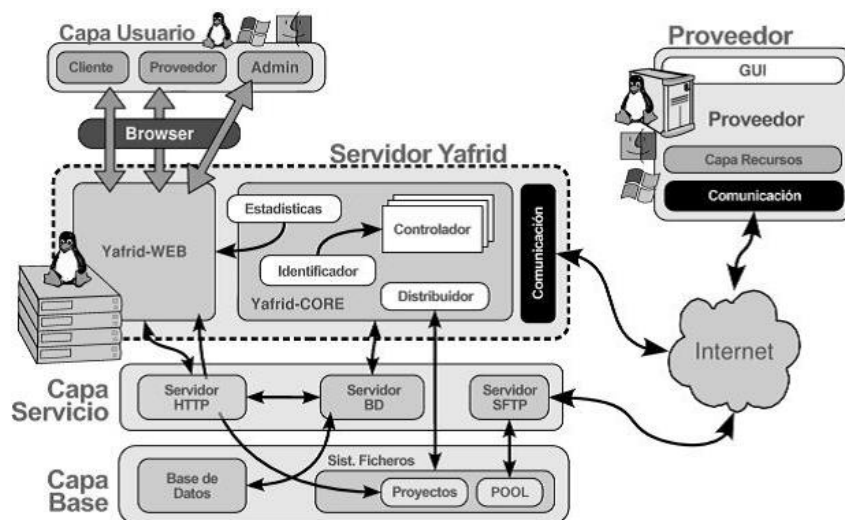


Imagen 6.- Capas que forman la arquitectura del servidor Yafrid

En la figura se muestran las cuatro capas que forman la arquitectura del servidor Yafrid. Este diseño está basado en la arquitectura de (I. Foster, C. Kesselman, S. Tuecke.). Las capas mencionadas anteriormente son Capa Base (o de Recursos, Capa de Servicio, Servidor Yafrid y Capa de usuario).

Capa de recursos. Ésta es la capa de menor nivel de abstracción. La capa de recursos tiene los siguientes componentes:

- **Sistema de base de datos.** En las bases de datos de esta capa se encuentran las tablas donde se almacena la información necesaria para el correcto funcionamiento del sistema. Algunas de estas tablas se utilizan para obtener estadísticas de la ejecución del sistema, mientras que otras almacenan datos asociados a usuarios, grupos, proyectos, etc. Los niveles que se encuentran sobre esta capa en la jerarquía pueden acceder a las bases de datos a través del servidor de bases de datos. La implementación actual del sistema utiliza MySQL como servidor para las bases de datos.
- **Sistema de archivos.** En algunas ocasiones es necesario acceder directamente al sistema de archivos desde las capas superiores. Básicamente el sistema distingue dos tipos de directorios: uno de ellos se utiliza para almacenar las unidades de trabajo de los proyectos creados, y el segundo tipo contiene información relativa a los usuarios y sus proyectos.
- **Sistema de red.** El módulo dedicado a las comunicaciones, que pertenece a la capa principal, oculta la utilización de los recursos de red de los ordenadores mediante el uso de un middleware.
- **Capa de Servicio.** Esta capa contiene los servidores que permiten a los módulos de las capas superiores acceder a los recursos que pertenecen a las capas inferiores. Los servidores que actúan en esta capa son los siguientes:
 - **Servidor HTTP.** El módulo web de Yafrid trabaja sobre este servidor. Como este módulo ha sido implementado con lenguajes de programación que permiten la construcción de páginas dinámicas (la implementación actual está escrita en lenguaje PHP), el servidor web

debe proporcionar soporte para este tipo de lenguajes. También, es necesario tener soporte para la composición dinámica de gráficos y el acceso a la base de datos.

- **Servidor de bases de datos.** Este servidor es utilizado por los módulos de Yafrid para acceder a los datos que son indispensables para el correcto funcionamiento del sistema.
- **Servidor SFTP.** Los proveedores utilizan este servicio para obtener los archivos necesarios para el renderizado de las unidades de trabajo. Una vez que el proceso de renderizado ha finalizado, se utilizará el servidor SFTP (Simple File Transfer Protocol) para enviar al servidor de Yafrid la imagen resultante.
- **Capa Yafrid.** Esta es la capa principal del servidor y está compuesta de dos módulos diferentes (Yafrid-WEB y Yafrid-CORE) que trabajan independientemente el uno del otro. Yafrid-WEB es el módulo interactivo del servidor y ha sido implementado mediante un conjunto de páginas dinámicas. Yafrid CORE es la parte no interactiva del servidor. Este módulo ha sido principalmente desarrollado utilizando el lenguaje Python. Además, Yafrid-CORE está compuesto por tres submódulos; Distribuidor, Identificador Estadísticas.
 - El Distribuidor es la parte activa del servidor encargado de realizar las siguientes tareas indispensables: generación de unidades de trabajo, asignación de las unidades de trabajo a los proveedores, control del Timeout, finalización de proyectos y composición de resultados. A partir de los resultados generados por los proveedores, el distribuidor compone la imagen final. Este proceso no es trivial ya que se pueden distinguir pequeñas diferencias entre los fragmentos obtenidos de diferentes computadores, debido a la componente

aleatoria de los métodos basados en Monte Carlo. Por esta razón es necesario suavizar la zona de unión entre los fragmentos mediante una máscara de interpolación lineal.

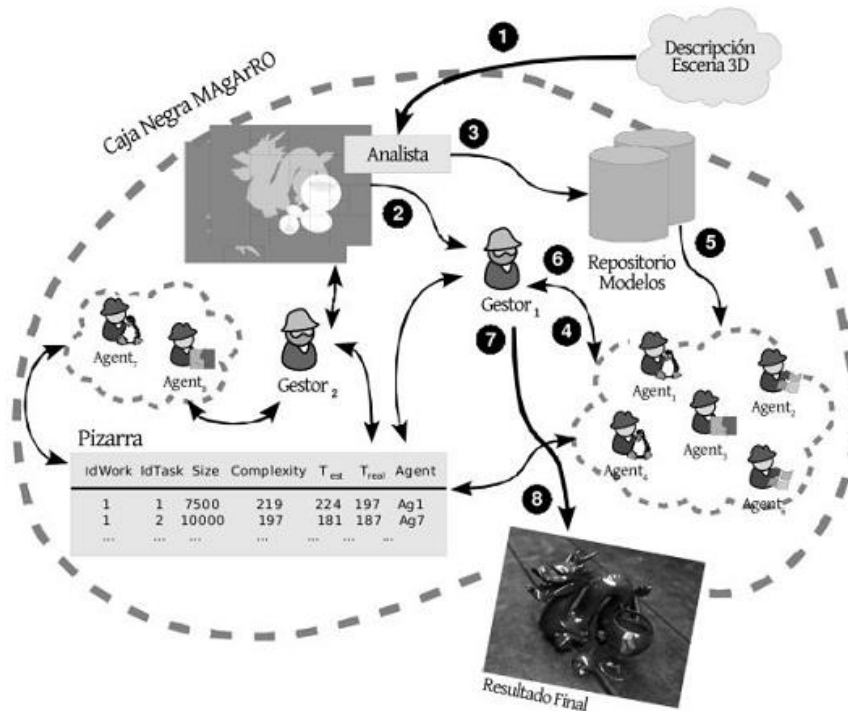


Imagen 7.- Flujo de Trabajo de y principales roles en la arquitectura

- La parte pasiva de Yafrid-CORE se denomina módulo Identificador cuya misión principal consiste en establecer las comunicaciones de los proveedores. La primera vez que el proveedor intenta conectar con el servidor Yafrid, el Identificador genera un objeto (controlador del proveedor) y le devuelve un proxy. Es importante destacar que cada proveedor tiene su propio objeto controlador.
- Proveedor. El proveedor es el software que un usuario debe instalar en su ordenador para que el sistema grid pueda utilizar sus ciclos de CPU ociosos. La primera tarea que debe llevar a cabo un proveedor es establecer la conexión con el grid. Una vez activado, el proveedor espera

hasta que el servidor le envía una unidad de trabajo para procesarla. Cuando finaliza el proceso de renderizado, el proveedor envía el archivo vía SFTP e informa al controlador que el trabajo ha finalizado. (I. Foster, C. Kesselman, S. Tuecke.)

2.2.3.Un sistema difuso para la optimización del renderizado

La gran cantidad de tiempo empleado en el proceso de renderizado implica que todas las optimizaciones relacionadas persigan la reducción del tiempo final requerido. Estas optimizaciones se pueden clasificar en tres tipos generales (o en una mezcla de los mismos) (Debattista, K., Longhurst, P., Chalmers, A., and Troscianko, T. , 2005)

1. Optimizaciones relacionadas con el uso de elementos de procesamiento potentes (optimizaciones de hardware).
2. Optimizaciones vinculadas al uso de sistemas distribuidos (optimizaciones paralelas y distribuidas).
3. Optimizaciones asociadas al uso de sistemas multiagente.

El primer tipo de optimización está relacionado con el uso de potentes elementos de procesamiento. De esta forma, algunos investigadores utilizan GPUs (Graphics Processing Units) programables con el objetivo de que estos procesadores potentes ejecuten ciertas porciones de código del algoritmo de renderizado. El principal problema de este enfoque es el bajo nivel de abstracción empleado, es decir, los algoritmos han de estar específicamente diseñados para cada arquitectura hardware.

El segundo tipo de optimización está basada en el principio de divide y vencerás. Si se adapta este principio al problema específico del renderizado, se pueden utilizar varias unidades de cómputo para reducir el tiempo final de cálculo. Existen

diversos enfoques asociados a esta línea de investigación, como el trabajo realizado por Fernández-Sorribes, que hace uso de una arquitectura grid para gestionar el renderizado distribuido. Sin embargo, la mayoría de estos enfoques no logran un balanceado eficiente de la carga de trabajo y, por lo tanto, el tiempo final requerido está siempre condicionado por la tarea de mayor complejidad.

El tercer tipo de optimización está basado en el uso de sistemas multiagente. En este campo se han propuesto diferentes alternativas, como el trabajo presentado por Rangel-Kuoppa. Sin embargo, este enfoque no hace uso de la tecnología de agentes propiamente dicha, ya que no existen mecanismos de interacción entre los agentes. Otro trabajo relacionado es el propuesto por Schlechtweg, el cual hace uso de un sistema multiagente para renderizar diversos estilos artísticos.

El enfoque descrito en este artículo está basado en un sistema difuso. De este modo, se puede representar el conocimiento experto cualitativo empleando un conjunto de reglas lingüísticas, las cuales permiten a cada unidad computacional ajustar un determinado número de parámetros vinculados directamente al proceso de renderizado. A través de estos ajustes, los tiempos finales de renderizado se reducen, resolviendo el problema relacionado a la tendencia del usuario a utilizar valores de los parámetros del renderizado que no son óptimos, debido a que estos ajustes incrementan el tiempo final sin obtener beneficios importantes de calidad.

Este sistema ha sido usado satisfactoriamente en MAgArRO (Multi-Agent Architecture for Rendering Optimizations). Este sistema multi-agente (diseñado bajo los principios del conjunto de estándares de FIPA (FIPA, 2002) está compuesto por un número indeterminado de agentes especializados, los cuales compiten por la adquisición de unidades de trabajo que componen el trabajo completo. Una unidad de trabajo se puede interpretar como una parte de una escena o como un fotograma en una animación.

El enfoque utilizado en este trabajo implica ciertas mejoras con respecto a otras líneas de investigación:

- Proporciona un alto nivel de abstracción debido al uso de un sistema difuso.
- El poder descriptivo del sistema difuso empleado permite que el experto modele el conocimiento relativo al renderizado de una forma sencilla (Tanaka, 1998)
- El uso de un sistema difuso combinado con un sistema multi-agente permite que cada agente utilice diferentes modelos de conocimiento experto. De esta forma, se podrían emplear distintos agentes especializados manejando diferentes conjuntos de reglas.
- Complementa el uso de conocimiento experto en alternativas basadas en hardware o en renderizado paralelo.

2.2.4.Optimización mediante Hardware

Una alternativa para reducir el tiempo de renderizado consiste en realizar optimizaciones particulares mediante hardware. En esta línea de investigación existen diferentes aproximaciones; algunos métodos utilizan GPUs (Graphics Processing Units) programables como sistemas de paralelización masiva. De esta forma se dispone de potentes procesadores para la ejecución en paralelo de diferentes fragmentos de código de un trazador de rayos.

El uso de GPUs programables superan a las CPUs (Central Processing Unit) de una estación de trabajo estándar en un factor de siete (I. Buck, T. Foley, D. Horn, J. Sugerman, K.Fatahalian, M. Houston, P. Hanrahan., 2004). El uso de la CPU en conjunción con la GPU requiere nuevos paradigmas y alternativas a las (R. Rajagopalan, D. Goswami, S.P. Mudur., 2005) muestra el uso de un GPU para

renderizar imágenes en tiempo real a partir de conjuntos de datos complejos, los cuales requieren tareas de cómputo de gran complejidad.

Existen algunos motores de render diseñados específicamente para ser utilizados con aceleración GPU, tales como **Parthenon Renderer**, el cual es un sistema de render de iluminación global que hace uso de la potencia de cálculo conjunto de las CPUs y GPUs instaladas. (Hachisuka., 2005) o el motor de render **Gelato®** que trabaja con tarjetas gráficas **NVIDIA®**, es un renderizador offline (no de tiempo real) acelerado por la GPU que se diseñó inicialmente para crear efectos visuales y de animación 3D para películas, pero también constituye una potente herramienta para numerosos sectores industriales, entre ellos, los del cine y la televisión, el diseño CAD, la arquitectura o las artes gráficas. (NVIDIA, 2011).

3. VARIABLES DE ESTUDIO

3.1 Identificación de variables potenciales

A continuación se presenta el conjunto de variables que son consideradas como relevantes para la solución del problema de estudio, por considerarse factores potenciales en el tiempo de renderizado.

Dentro del marco teórico y planteamiento del problema encuentro que tengo definido que la mejora de procesos en renderizado es una variable dependiente de varios factores, por lo tanto la variable dependiente que defino es:

Y= Tiempo de Renderizado

Así mismo encontré que tenemos diferentes aspectos que inciden para mejorar los procesos de renderizado de imágenes y que estos puedan optimizar el tiempo que tarda en realizar dicho render. En la siguiente tabla se presenta la frecuencia con que diversos autores e investigaciones previas hacen mención de cada variable como una causa potencial del problema en cuestión:

Nombre de variable	Frecuencia
Motor de Renderizado	5
Cantidad de Fotogramas	4
Técnicas de Renderizado	3

Tabla 4.- Variables Independientes

A continuación, se presenta de forma gráfica mediante un Diagrama de Pareto las variables independientes seleccionadas para realizar la investigación.

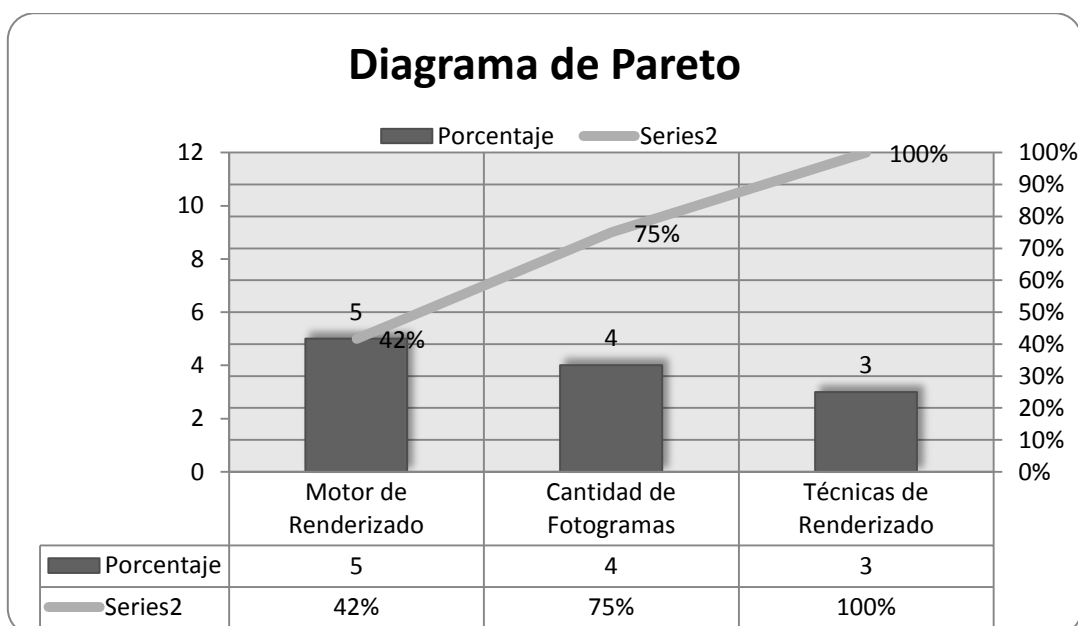


Gráfico 3.- Diagrama de Pareto de las Variables Independientes

X	Y	Y porcentaje	Y Acum	Y Acum Porcentaje
Motor de Renderizado	5	42%	5	42%
Cantidad de Fotogramas	4	33%	9	75%
Técnicas de Renderizado	3	25%	12	100%

Tabla 5.- Nombres de las Variables y Frecuencia.

En seguida, se demuestra el fundamento teórico de cada una de las variables independientes seleccionadas, con el fin de sustentar teóricamente su importancia y peso dentro de la investigación de manera que se justifique de manera clara la elección de cada variable.

3.2 Desarrollo teórico de variables potenciales

En seguida, se presenta el fundamento teórico de cada una de las variables independientes seleccionadas, con el fin de sustentar teóricamente su importancia y peso dentro de la investigación de manera que se justifique de manera clara la elección de cada variable.

3.2.1. Motor de renderizado.

Software que se encarga de interpretar y representar contenido, en el caso de animación 3D, consiste en tomar un modelo virtual en 3D dimensiones, colocar las cámaras objetivo, y mediante una configuración previa de iluminación y materiales de la escena, la computadora RENDERIZA, todos los cálculos de rebotes de luz en los materiales, entregándonos una imagen “Real” lo más parecida a la física que tenía cada material en la escena, el tiempo de demora de cada Render dependerá del procesador que se utilice. (Noticias3D, 2010).

Para fines de realizar pruebas de renderizado con los distintos modelos 3D, se eligió utilizar dos motores de renderizado el primero es V-Ray.

V-Ray motor de render para el procesamiento de escenas con un alto nivel de realidad. Un motor de render que ofrece los recursos necesarios para aplicar a las escenas materiales avanzados y fuentes de iluminación global, este motor de render permite aplicar a los modelos 3D todo tipo de materiales de carácter avanzado: reflexiones y refracciones, soluciones borrosas, materiales con iluminación propia, elementos traslúcidos (ropa o papel) o utilizar texturas de dispersión, este motor de render tiene gran potencial a la hora de calcular la iluminación de las escenas, pues permite trabajar con iluminación global, luces indirectas, mapas irradiantes, luces cáusticas, entre otras fuentes de luz para dar a las escenas un mayor realismo. (Kuhlo, 2010)

El segundo motor de render que se utilizará para realizar pruebas de renderizado es **MentalRay**, es un motor completo y complejo que calcula desde sombras simples hasta segmentadas, compilación fotónica emitida por luces que se conoce como "Photon Map" y que posee su propio sistema de aceleración, una de sus principales características es que es un software Multiplataforma, Mental ray es completamente programable mediante shaders (escrito en C o C++). Utiliza Múltiples geometrías: Puede trabajar con polígonos, NURBS y subdivision de surfaces.

3.2.2 Cantidad de Fotogramas

Se denomina frame en inglés, a un fotograma o cuadro, una imagen particular dentro de una sucesión de imágenes que componen una animación. La continua sucesión de estos fotogramas producen a la vista la sensación de movimiento, fenómeno dado por las pequeñas diferencias que hay entre cada uno de ellos.

La frecuencia es el número de fotogramas por segundo que se necesitan para crear movimiento. Su fórmula es la siguiente:

$$f(\text{frames}) = \frac{1}{T(s)}$$

Ecuación 1.- Fórmula para calcular el número de fotogramas necesarios para crear movimiento.

Se expresa en fotogramas o *frames* por segundo (fps) o en hercios (Hz). Para conseguir que el sistema visual humano vea movimiento hemos de tener en cuenta que:

- Para no observar parpadeo se ha de tener una frecuencia de fotograma > 50 Hz.
- La discontinuidad de movimiento tiene una frecuencia de fotograma < 12–15 Hz.

Las frecuencias de fotograma de algunos de los sistemas más conocidos son las siguientes:

- Cine mudo = 16–18 Hz.
- Cine = 24 Hz.
- Televisión, normas europeas (PAL & SECAM) = 25 Hz.
- Televisión, norma estadounidense (NTSC) = 29,97 Hz.

Estas frecuencias van en relación a la frecuencia de la red eléctrica. En Europa es de 50 Hz, es decir el doble de la frecuencia de la televisión que es de 25 fotogramas cada segundo o, lo que es lo mismo, 25 Hz; en EE.UU. y México es de 60 Hz, el doble de la frecuencia de la televisión que es de 30 fotogramas cada segundo o, lo que es prácticamente lo mismo, 29,97 Hz. (Wikipedia, 2010)

3.3.3. Técnicas de Renderizado

El término **gráficos 3D por computadora** (3D computer graphics) se refiere a trabajos de arte gráfico que son creados con ayuda de computadoras y programas especiales 3D. En general, el término puede referirse también al proceso de crear dichos gráficos, o el campo de estudio de técnicas y tecnología relacionadas con los gráficos 3D. Un gráfico 3D difiere de uno 2D principalmente por la forma en que ha sido generado. Este tipo de gráficos se originan mediante un proceso de cálculos matemáticos sobre entidades geométricas tridimensionales producidas en una computadora y cuyo propósito es conseguir una proyección visual en dos dimensiones para ser mostrada en una pantalla.

Para generar el renderizado de un modelo 3D, existen varias técnicas dependiendo de los elementos que contenga el modelo, para generar imágenes fotorrealistas se utilizan técnicas denominadas de **iluminación global** que tratan de simular la luz dispersa en un modelo 3D ateniéndose a las leyes físicas. Estas técnicas calculan el valor de intensidad en cada punto de la escena teniendo en cuenta las interacciones de la luz con los objetos de la escena.

Algunas de las técnicas más utilizadas para el proceso de renderizado son las siguientes:

3.3.4.1. Scanline

Es quizá uno de los métodos de render más básicos y fue introducido por Bouknight en el año 1970.

Algoritmo 1 Bucle general de Scanline.

```
for all pixel de la imagen do
  línea ← trazar una línea desde la cámara al pixel
  color ← Scanline (línea) {Algoritmo 2}
end for
```

Algoritmo 2 Procedimiento Scanline (línea)

```
punto ← encontrar el punto de intersección más cercano
color ← color de fondo
for all fuente de luz do
  color ← color + illumination directa
end for
Devolver(color)
```

Una de sus principales ventajas es la rapidez, de hecho, es el método que menos tiempo consume, además, trabaja bien con texturas y simula cualquier tipo de sombreado. Como contrapartida, no permite simular las reflexiones y refracciones de la luz de una manera realista, aunque pueden ser simuladas utilizando técnicas adicionales.

3.3.4.2. Raytracing

Algoritmo 3 Bucle general del Raytracing.

```
for all pixel de la imagen do
  rayo ← trazar un rayo desde la cámara al pixel
  color ← Raytracing (rayo) {Algoritmo 4}
end for
```

Algoritmo 4 Procedimiento Raytracing (rayo)

```
punto ← encontrar el punto de intersección más cercano
color ← color de fondo
for all fuente de luz do
  rayo_sombra ← trazar un rayo desde (punto) hasta la fuente de luz
  if el rayo no choca con ningún objeto then
    color ← color + iluminación directa
  if el material tiene propiedad de reflexión then
    color ← color + Raytracing (rayo reflejado)
  end if
  if el material tiene propiedad de refraccion then
    color ← color + Raytracing (rayo transmitido)
  end if
else
  color ← negro
end for
```

```
end if  
end for  
Devolver(color)
```

El método de raytracing o trazado de rayos fue propuesto por Whitted y proporciona un medio sencillo y recursivo de calcular superficies con reflejos y transparencia. La idea, tal y como ya se ha comentado, consiste en trazar rayos desde el observador a las fuentes de luz. En realidad, son las fuentes de luz las que emiten fotones que rebotan en la escena y llegan a los ojos del observador. Sin embargo, sólo una pequeñísima fracción de los fotones llegan a su destino, por lo que el cálculo en esta forma directa resulta demasiado costosa.

Los rayos que se emiten a la escena son evaluados respecto de su visibilidad trazando nuevos rayos desde los puntos de intersección (rayos de sombra). Una descripción básica del trazado de rayos se muestra en los Algoritmos 3 y 4. El raytracing permite simular reflexiones especulares y refracciones pero no está pensado para simular iluminaciones indirectas o sombras difusas.

3.3.4.3. Pathtracing

Supone una extensión al algoritmo de trazado de rayos y fue formulado por Kajiya en 1986. El pathtracing permite calcular la solución completa de la iluminación global. El mecanismo en el que se basa consiste en lanzar rayos para calcular todos los posibles caminos de donde pueda venir la luz.

El muestreo se realiza mediante la integración de Monte Carlo, que consigue crear rayos uniformemente por todos los posibles caminos.

Cuando un rayo choca contra una superficie difusa, en lugar de llamar al procedimiento recursivo con los múltiples rayos que rebotarían en la superficie, se selecciona uno de forma aleatoria.

La utilización de la irradiance cache evita el cálculo de la iluminación global en cada punto del modelo ya que se calculan algunos pixeles y los valores intermedios se interpolan.

3.3.4.4. Photon Mapping

Propuesto por Jensen en 1996, presenta como novedad que desacopla la representación de la iluminación de la geometría. Esta técnica consiste en dos pasos:

1. Construcción de la estructura del mapa de fotones desde las fuentes de luz al modelo. Para ello, se lanza un gran número de fotones desde las fuentes de luz. Cada uno de estos fotones contiene una fracción de la energía total de la fuente de la que proviene.
2. Se lleva a cabo el proceso de render mediante la utilización de la información contenida en el mapa de fotones. La recuperación de la información del mapa de fotones tiene una complejidad de $O(\log n)$ en el caso medio y $O(n)$ en el peor de los casos.

3.3.4.5. Pathtracing Bidireccional

Extensión del pathtracing introducida por Lafortune y Willems que se caracteriza porque también se calculan los caminos desde las fuentes de luz. Aprovecha la idea de que hay ciertos caminos que son más fáciles de trazar desde las fuentes

de luz, como por ejemplo el caso de las caústicas. Este método calcula dos caminos, uno que parte desde el punto de observación y otro que parte desde cada fuente. La información obtenida por ambos medios se combina para obtener los valores de iluminación finales.

Aunque requiere menos muestras que el pathtracing sencillo, el cálculo tarda más debido a que el coste de cada muestra es mayor y hay que añadir el tiempo de combinar los caminos. Efectos como las caústicas se calculan perfectamente.

3.6 Definición de variables

A continuación en la tabla **4.- Definición de Variables** se presenta la descripción teórica y operativa del conjunto de variables de estudio consideradas como relevantes que podrían ser causas potenciales del problema.

Variable Independiente	Código	Descripción Teórica	Descripción Operativa
Motor de renderizado	X1	Proceso que consiste en tomar un modelo virtual en tres dimensiones, incluir las cámaras objetivo, y mediante una configuración previa de iluminación y materiales de la escena, la computadora realiza los cálculos matemáticos necesarios para generar luz en los materiales, refracciones y transparencias con la finalidad de generar una	Son los factores que determina el tiempo en que tarda en renderizar una imagen o modelado 3D.

		imagen real, en donde el motor de renderizado está en función de la velocidad del procesamiento del equipo de cómputo. (ALEGSA, 2011)	
Cantidad de fotogramas	X2	Cantidad de fotogramas o cuadros de una imagen particular dentro de una sucesión de imágenes que componen una animación, con la finalidad de conseguir que el sistema visual humano vea movimiento. (PIERRE, 1996)	Es el número de fotogramas por segundo que se necesitan para crear movimiento.
Técnicas de Renderizado	X3	Proceso algorítmico para la visualización de imágenes 3D, por el cual una imagen descrita en un formato gráfico vectorial (modelo basado en el eje de coordenadas X,Y) se convierte en un conjunto de píxeles o puntos para ser desplegados en un medio de salida digital, como una pantalla de computadora. (Garola, 2009)	Método para generar una imagen o animación desde un modelo o imagen 2D a un modelo tridimensional, basado en un eje de coordenadas respecto al desplazamiento que hay que seguir respecto cada eje para llegar al punto, (X,Y,Z).

Tabla 6.- Definición de Variables

4. HIPÓTESIS

4.1 Hipótesis General

Para la disminución del tiempo de renderizado en modelos tridimensionales intervienen factores como el Motor de Renderizado, Cantidad de Fotogramas y Técnicas de Renderizado.

4.2 Hipótesis de Trabajo

De acuerdo con las variables independientes (X) seleccionadas y a su respectivo análisis desde la parte teórica, se presenta a continuación las hipótesis generadas de su relación con la variable dependiente **Y= Tiempo de Renderizado**.

HIPÓTESIS	DESCRIPCIÓN
H1	El tiempo de renderizado (Y) disminuye conforme se utilice un distinto Motor de renderizado (X1).
H2	El tiempo de renderizado (Y) disminuye conforme la cantidad de fotogramas (X2) disminuye.
H3	El Tiempo de renderizado (Y) disminuye conforme se utilice una distinta técnica de renderizado (X3).

Tabla 7- Descripción de Variables

5. DISEÑO DE LA INVESTIGACIÓN Y TRABAJO EMPÍRICO

5.1 Factores y nivel de experimentación

Para realizar las pruebas se tomo un total de 8 experimentos y se realizarán 5 replicas de cada uno, con la finalidad de obtener la variación del tiempo en que tarda en renderizar un modelo tridimensional o recorrido virtual, en relación a los parámetros establecidos, cada experimento está definido con características propias como son, **Motor de render, Técnica de render y Número de fotogramas** y se realizarán las distintas combinaciones posibles para realizar los experimentos correspondientes.

Cabe señalar que los criterios que se utilizaron están en relación al logro de los objetivos de estudio en este reporte, en este caso, dichos objetivos son la disminución del tiempo de renderizado en la creación de modelos tridimensionales o recorridos virtuales.

Prueba 1	Prueba 2	Prueba 3
Motor de render: MentalRay	Motor de render: MentalRay	Motor de render: V-Ray
Técnica: Scanline	Técnica: Scanline	Técnica: Scanline
Fotogramas: 12	Fotogramas: 24	Fotogramas: 12

Prueba 4	Prueba 5	Prueba 6
Motor de render: MentalRay	Motor de render: V-Ray	Motor de render: V-Ray
Técnica: PhotonMapping	Técnica: Scanline	Técnica: PhotonMapping
Fotogramas: 12	Fotogramas: 24	Fotogramas: 12

Prueba 7	Prueba 8
Motor de render: MentalRay	Motor de render: V-Ray
Técnica: PhotonMapping	Técnica: PhotonMapping
Fotogramas: 24	Fotogramas: 24

Tabla 8.- Factores establecidos para realizar el experimento

5.2 Diseño Experimental

Con el fin de sacar conclusiones sobre el comportamiento del sistema en un entorno real, se ejecutaron varios conjuntos de pruebas o test. Estos conjuntos de pruebas consisten en realizar una serie de proyectos de experimento, en donde todas las pruebas tienen distintos parámetros de calidad (como son el motor de render, técnica de renderizado y número de fotogramas).

Para hacer que los datos que se obtuvieran pudieran ser comparables, se optó por que las computadoras que se utilizaron para realizar las pruebas tuvieran la misma configuración, es decir, que todas las computadoras tuvieran similares características.

Para realizar los experimentos se utilizó un modelo tridimensional con una duración final de 1 minuto 56 segundos, el cual está formado por texturas, luz, reflexiones, refracciones y movimiento de objetos; para generar la animación se generaron dos distintos motores de render, **MentalRay y V-Ray**, se utilizaron dos distintas técnicas de renderizado **Scanline y PhotonMapping** y se realizaron pruebas con distintas cantidades de fotogramas 12 cuadros por segundo y 24 cuadros por segundo, cabe señalar que estos experimentos se llevaron a cabo para estudiar los efectos producidos por dos o más factores en el proceso de renderizado de modelos tridimensionales, en general los diseños factoriales son

los más eficientes para este tipo de experimentos. Por diseño factorial se entiende aquel en el que se investigan todas las posibles combinaciones de los niveles de los factores en cada ensayo completo o réplica del experimento.

El efecto de un factor se define como el cambio en la respuesta producida por un cambio en el nivel del factor. Con frecuencia, éste se conoce como efecto principal porque se refiere a los factores de interés primordial del experimento. El experimento que se realizó es 2^k completo, a continuación se presentan los datos obtenidos:

OrdenEst	Orden Corrida	Punto Central	Bloques	Motor de Render	Fotogramas	Técnica	Tiempo
2	1	1	1	Mentalray	12	Scanline	20.18
8	2	1	1	Mentalray	24	Photon	25.244
6	3	1	1	Mentalray	12	Photon	16.858
4	4	1	1	Mentalray	24	Scanline	28.452
3	5	1	1	Vray	24	Scanline	19.35
7	6	1	1	Vray	24	Photon	22.524
1	7	1	1	Vray	12	Scanline	30.538
5	8	1	1	Vray	12	Photon	26.318

Tabla 9.- Datos Obtenidos del experimento.

5.3 Resultados

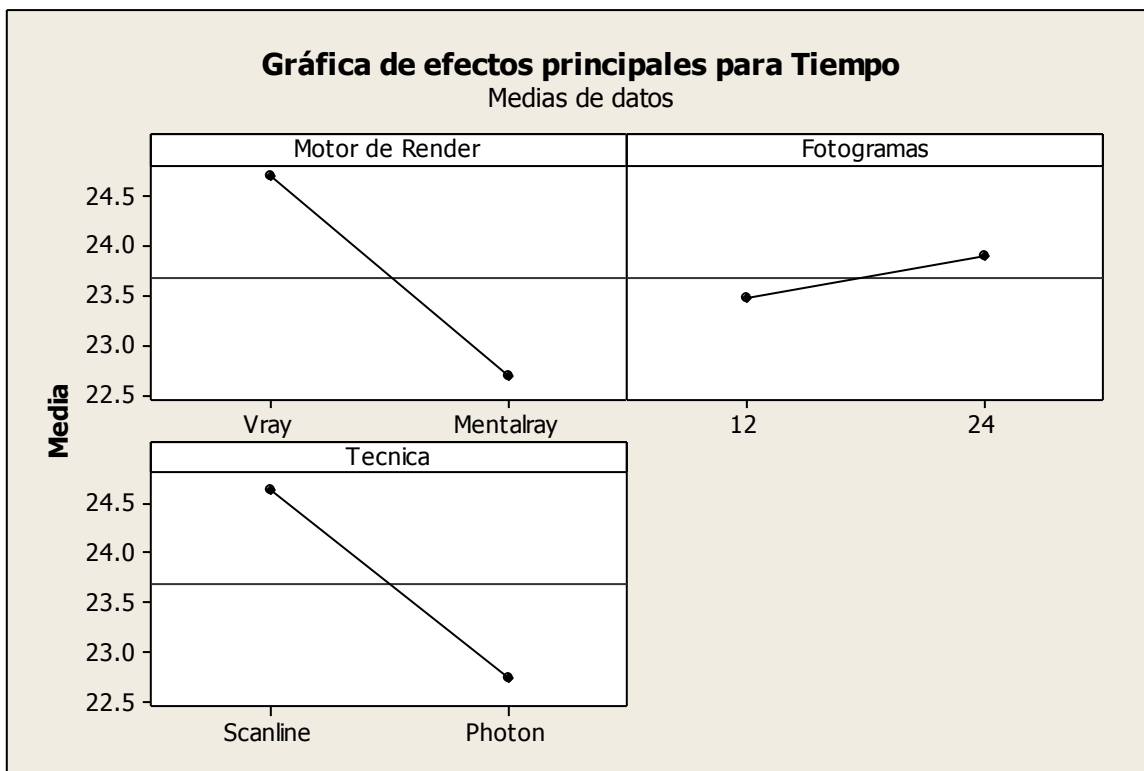


Gráfico 4.- Efectos principales de tiempo.

En la gráfica de Efectos principales para determinar el tiempo de renderizado, nos indica que se debe de utilizar una combinación de motor de render MentalRay, 12 fotogramas por segundo y utilizar la técnica de renderizado PhotonMapping, ya que esta combinación nos ayuda a reducir los tiempos de renderizado, sin ser la más óptima.

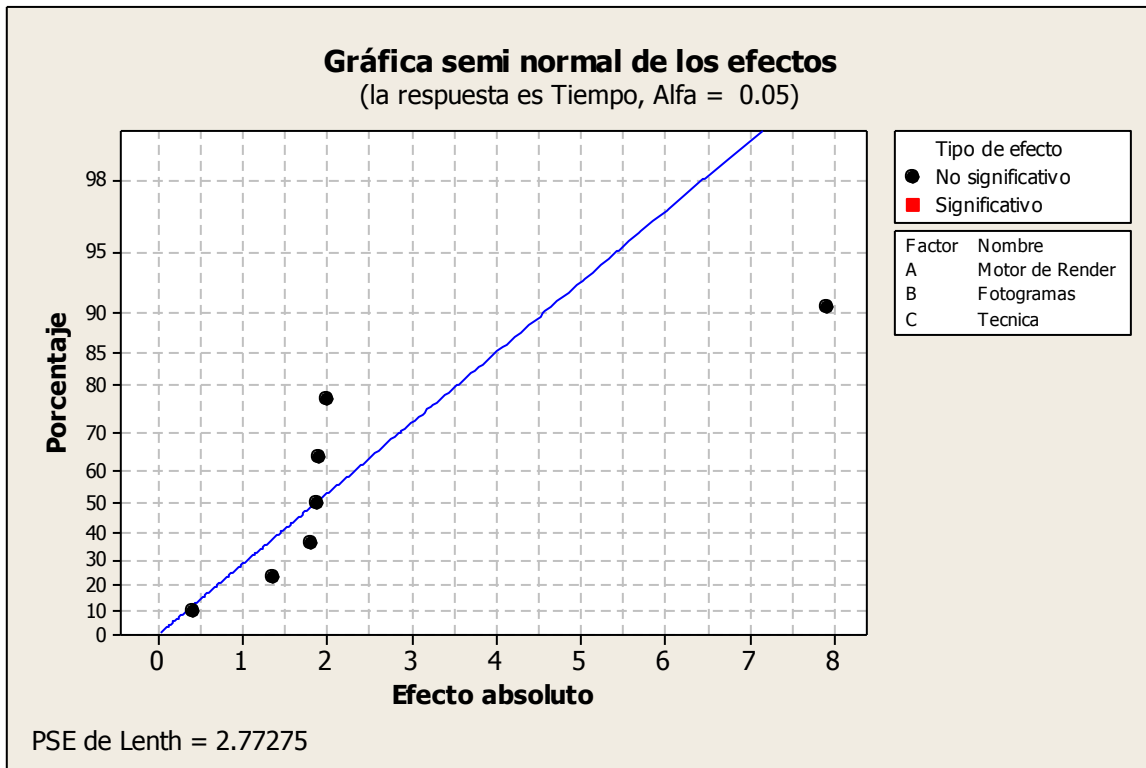


Gráfico 5.- Semi normal de los efectos

Esta gráfica nos muestra que los tiempos de renderizado no son significativos para poder determinar cuál es la mejor combinación para disminuir el tiempo de renderizado.

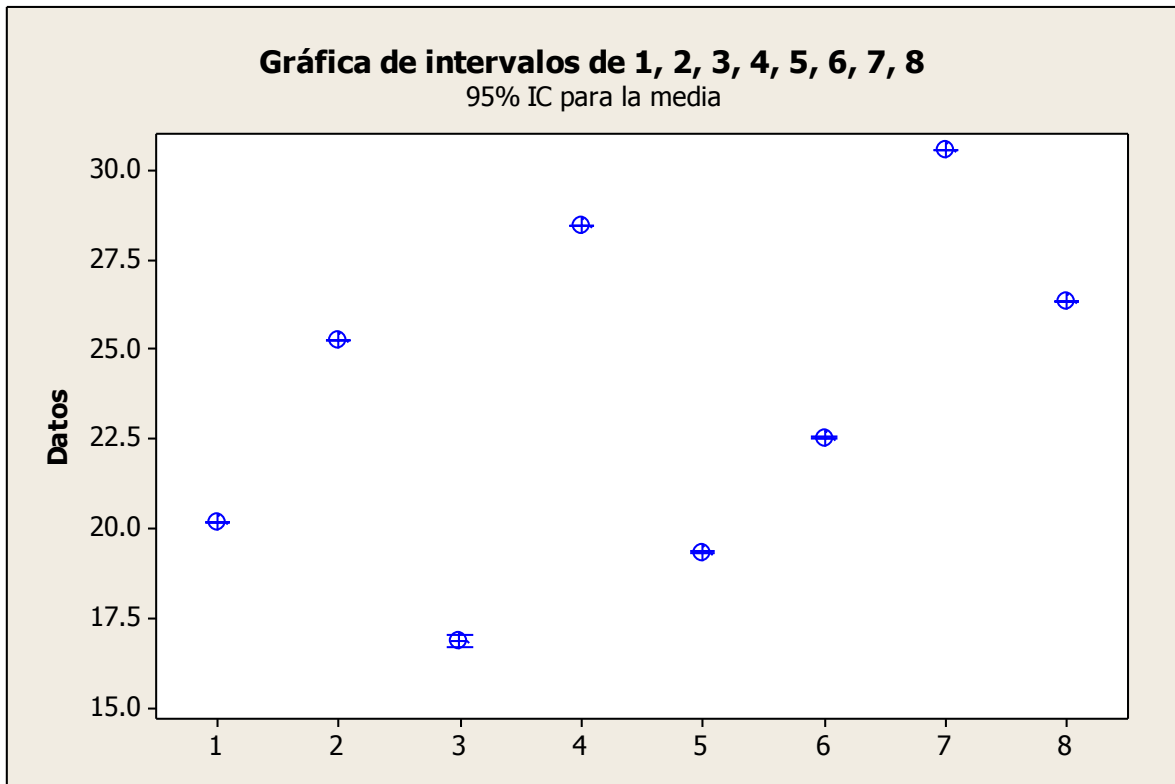


Gráfico 6.- Intervalos de experimentos.

En esta gráfica podemos observar que si existe una diferencia entre los distintos experimentos realizados, por lo tanto se puede decir que la mejor opción para disminuir los tiempos de renderizado es la número 3, pero si analizamos las estadísticas no existe una combinación significativa para determinar cuál es la mejor opción de disminuir los tiempos de renderizado.

5.5 Análisis de datos y resultados.

Para poder determinar los factores que nos permitan disminuir el tiempo en que tarda en renderizar una imagen o modelo tridimensional, contamos con tres variables independientes estas son: Motor de render, cantidad de fotogramas y técnica de renderizado, una vez realizados los experimentos y obteniendo los resultados ver tabla de datos, podemos determinar que efectivamente estos

factores nos ayudan a disminuir el tiempo de este proceso, sabemos que también intervienen otros factores que en esta ocasión no forman parte del estudio.

Una vez realizados los experimentos y obtenido los resultados podemos determinar cuál de las combinaciones establecidas para realizar los experimentos es la que más se adapta para reducir el tiempo de renderizado; más sin embargo al observar los datos de la tabla podemos notar que los datos arrojados no son los suficientemente significativos para poder determinar lo anterior.

En base a esto, se contempla el realizar en un futuro otro experimento tomando en cuenta otras variables que nos ayuden a determinar la disminución del tiempo de renderizado, así como involucrar otros factores o parámetros del modelo tridimensional para poder determinar algunos otros criterios.

Prueba	Test 1	Test 2	Test 3	Test 4	Test 5
1	20.19	20.18	20.18	20.17	20.18
2	25.23	25.23	25.25	25.25	25.26
3	17.09	17.03	17.03	16.58	16.56
4	28.45	28.46	28.45	28.45	28.45
5	19.38	19.36	19.36	19.34	19.31
6	22.58	22.5	22.51	22.51	22.52
7	30.56	30.57	30.52	30.52	30.52
8	26.34	26.32	26.31	26.31	26.31

Tabla 10.- Tabla de resultados del tiempo de renderizado

5.6 Contrastación de hipótesis

HIPÓTESIS	DESCRIPCIÓN	
H1	El tiempo de renderizado (Y) disminuye conforme se utilice un	Aceptada

	distinto Motor de renderizado (X1).	
H2	El tiempo de renderizado (Y) disminuye conforme la cantidad de fotogramas (X2) disminuye.	Aceptada
H3	El Tiempo de renderizado (Y) disminuye conforme se utilice una distinta técnica de renderizado (X3).	Aceptada

Tabla 11.- Tabla Contrastación de hipótesis.

6. PROPUESTA DE INTERVENCIÓN

La propuesta de solución tentativa para solucionar el problema identificado después de realizar un análisis conceptual con base a los resultados arrojados en el desarrollo del proyecto de investigación, es la creación de un clúster de renderizado el cual va permita mejorar los tiempos de renderizado atacando directamente las variables que fueron consideradas como significativas y que impactan y se relacionan directamente con el problema identificado, las cuales fueron Motor de renderizado (X1), Cantidad de fotogramas (X2) y Técnica de renderizado (X3).

Con la creación del clúster de renderizado se pretende contar con una plataforma tecnológica abierta que sea flexible y escalable que nos permita disminuir los tiempos de renderizado y con esto poder entregar en tiempo y forma los servicios tecnológicos ofertados y que a su vez sirva para lograr la integración de diversos servicios dentro de la misma plataforma.

A continuación se muestra la investigación realizada de los diferentes tipos de clúster que se encuentran en el mercado.

6.1 Estado de la cuestión

6.1.1. BackBurner renderizado por lotes utilizando 3D's MAX

Es muy conocido el hecho de que, en 3ds MAX, se puede hacer renders en red, con más de una computadora; fenómeno más conocido como "Net render". Algunos de los beneficios que trae este tipo de rendereo es que se puede trabajar con muchas computadoras rendereando una sola escena o muchos computadores rendereando un solo frame, esto resulta bastante útil cuando se tiene que hacer pruebas en renders que se demoran más de 5 minutos, Backburner divide el proceso de render en tres partes:

Manager: Es el que maneja al resto de las computadoras. Desde él se asignan tareas al resto de los computadores que están en modo server.

Server: Se debe inicializar en cada uno de las computadoras denominadas "obreros", es decir las computadoras que van a renderear. Estos se comunican con el manager y esperan tareas.

Monitor: Es una ventana desde la cual se puede ver y controlar todos los procesos. Desde esta ventana se pueden cancelar las tareas o cambiar prioridades o propiedades. Lo ideal es que este encendida solo en el pc que hace de manager. Lo primero que se debe hacer es encender el **manager** en la computadora que va a controlar todo. La primera vez aparecerá una ventana de configuración, donde generalmente la configuración por defecto funciona a la perfección. Si se quiere (no es necesario pero siempre es bueno visualizar lo que se hace), se puede encender el **monitor** en la misma computadora que tiene la función de manager, para poder ver lo que está en lista de espera y a que computadoras les está enviando trabajos. (Peivem, 2007)

Este proceso de renderizado funciona de la siguiente manera:

1. Todas las computadoras deben estar en red y deben tener **3d studio MAX** (la misma versión para todos) y los **plugins** necesarios instalados.

2. Una de las computadoras debe funcionar como **manager**, es decir, es el que le manda trabajos al resto de las computadoras a los cuales se les, llama **servers**.
3. En la computadora denominada manager se abre 3d studio MAX, se setea el render y se manda a renderear.
4. En ese momento, el render que se envió queda en una lista de espera para ser renderizado apenas el manager se contacte con cualquiera de los servers.
5. Después de eso, solo queda esperar el resultado final.

6.1.2. Optimización utilizando computación distribuida.

Si dividimos el problema en otros más pequeños (y cada uno de ellos es resuelto en un procesador diferente), el tiempo necesario para resolver el problema completo debería disminuir. Para obtener una solución óptima al problema todas las computadoras que forman el sistema deberán mantenerse ocupados. Por tanto, es necesario elegir una buena estrategia de distribución de tareas para conseguir este objetivo.

De acuerdo a la forma en la que se ha realizado la división de tareas, podemos distinguir sistemas de granularidad fina, si una imagen se ha dividido en fragmentos más pequeños y éstos han sido enviados a distintos procesadores para que sean ejecutados de forma independiente, o bien, sistemas de granularidad gruesa (en el caso de las animaciones 3D) si cada fotograma de la animación sin fragmentar es enviado a una unidad de procesamiento. **Dr. Queue** es una herramienta de código abierto diseñada para distribuir fotogramas a través de una granja de computadores interconectados en una red; en concreto, este software multi-plataforma trabaja en un nivel de división de granularidad gruesa. (DrQueue, 2011)

Doctor Queue es un entorno de procesamiento paralelo que comenzó orientado a la renderización pero que ha evolucionado para proporcionar distribución, monitorización y gestión de tareas a través de una red de nodos de procesamiento. **Las tareas son distribuidas entre las computadoras y procesadas de manera paralela.** La configuración de los trabajos se realiza mediante scripts y también ofrece una API Python que permite extender la funcionalidad. Dr. Queue puede utilizarse en plataformas Windows, MacOS X, Linux, Irix y FreeBSD. Soporta plataformas mixtas en las que incluso pueden coexistir arquitecturas de 32 y 64 bits. (Carabias, 2008)

6.2 Modelo de Propuesta

Cuando se realiza un modelado 3D o un recorrido virtual, es necesario una vez que se construye el modelo, llevar a cabo un proceso que nos permita construir cada fotograma de la animación o película en 3D, por lo que es necesario que el programa empleado para modelar construya una foto de la cámara virtual que está enfocada al escenario. Dicho proceso puede tardar dos o tres minutos en baja calidad, por cada segundo de película animada necesitamos al menos 25 cuadros y si se toma en cuenta que un cortometraje puede durar varios minutos, entonces una sola PC tardaría semanas o meses durante todo el día construyendo los cuadros; para darle una solución tenemos los clúster o granjas de render. (Driemeyer, 2008).

La granja de render es un conjunto de computadoras interconectadas a través de un sistema operativo y un programa control, las cuales realizan una tarea en conjunto, en este caso el calcular la escena de un modelado 3D y crear las imágenes o renders de esta.

Es necesario el empleo de una red con características específicas a gigabit, esto se debe a la gran complejidad de cálculos que necesita realizar una computadora y distribuirla entre las demás computadoras, todo ello se traduce en tiempo de

espera para obtener una sola imagen, la cual a la hora de requerir de una animación se puede transformar en muchas horas de renderizado, días e incluso hasta semanas. De aquí la necesidad de utilizar el cómputo paralelo. En la actualidad la mayoría de los motores de renderizado (Scanline, PhotonMapping, Mental Ray, Vray, Brasil, Maxwell, etc.) son complejos sus cálculos y son necesarios en la interpretación del modelo 2D de una imagen 3D, utilizan todo el recurso disponible en dicha labor. De aquí la importancia de contar con equipos de alto rendimiento y capacidad.

La utilización de herramientas comerciales puede ayudar a la construcción de un clúster de render sin mayores problemas, desafortunadamente el costo tan alto que tienen las licencias hacen que muchas empresas busquen otras opciones.

Existe un gran número de herramientas que permiten la construcción de un clúster de renderizado, de las cuales resaltan, como se observa en la tabla 12.

Herramienta	Características
3D Studio Max – Backburner	Herramienta comercial desarrollada por Autodesk, puede utilizarse con VRay como herramienta alterna.
Blender – Ray-tracking	Herramienta de software libre ampliamente utilizada para proyectos complejos, compatible con 3D Max Studio, su render permite la utilización una gran variedad de efectos.
Maya - Maya Satellite	Herramienta ampliamente utilizada en granjas de render propuesta por Autodesk, puede utilizar diferentes motores de render como MentalRay o VRay.

LightWave – ScreamerNet	Herramienta comercial que permite el render en red en sistemas Mac OS X, sobre-sale su velocidad y capacidad.
Simple Open Mosix	Conjunto de herramientas de software libre para construir clúster de render basados en Blender y el sistema Linux.

Tabla 12. Herramientas para render en red más utilizadas.

Como se puede observar una de las plataformas más utilizadas para crear clúster es el software ofrecido por la empresa Autodesk, con 3D Max Studio, Maya y su motor V-Ray, son las herramientas más utilizadas en un clúster de red, desafortunadamente el costo de las licencias por nodo de red es muy alto y difícil de pagar para la mayoría de las Instituciones educativas en México y algunas organizaciones; la licencia de estas herramientas por nodo tiene un costo aproximado de US\$3,900.

La utilización de herramientas de software libre, es una gran alternativa para los diseñadores y desarrolladores de recorridos en 3D, puesto que los costos de licencias pueden ser invertidos en la infraestructura del clúster y así mejorar la eficiencia en el proceso de render de un proyecto.

La alternativa que se propone con este proyecto se basa en la experiencia que se tiene como grupo de trabajo al implementar un conjunto de herramientas de software libre, que permiten mejorar y obtener resultados de proyectos de render de gran tamaño. Sin embargo es necesario probar con otra gama de herramientas emergentes y que permitirán mejorar el proceso de paralelización.

La propuesta de este proyecto es ofrecer una solución basada en software libre y hardware de bajo costo, de tal forma que la integración de una granja de render para las empresas de la región sea una posibilidad y la solución para grandes proyectos de animación generados en la institución educativa.

En nuestro entorno de ejecución, el sistema está compuesto por 5 estaciones de trabajo localizadas en el mismo lugar. Estos computadores son PCs con características iguales. Los requerimientos mínimos que debe tener cada computadora para pertenecer al sistema son una memoria RAM de 1 GB y tener una conexión de al menos 100Mbps/s (todas las computadoras están conectadas a la red utilizando switches de 100Mbps/s). En la siguiente figura se muestran estos requisitos.

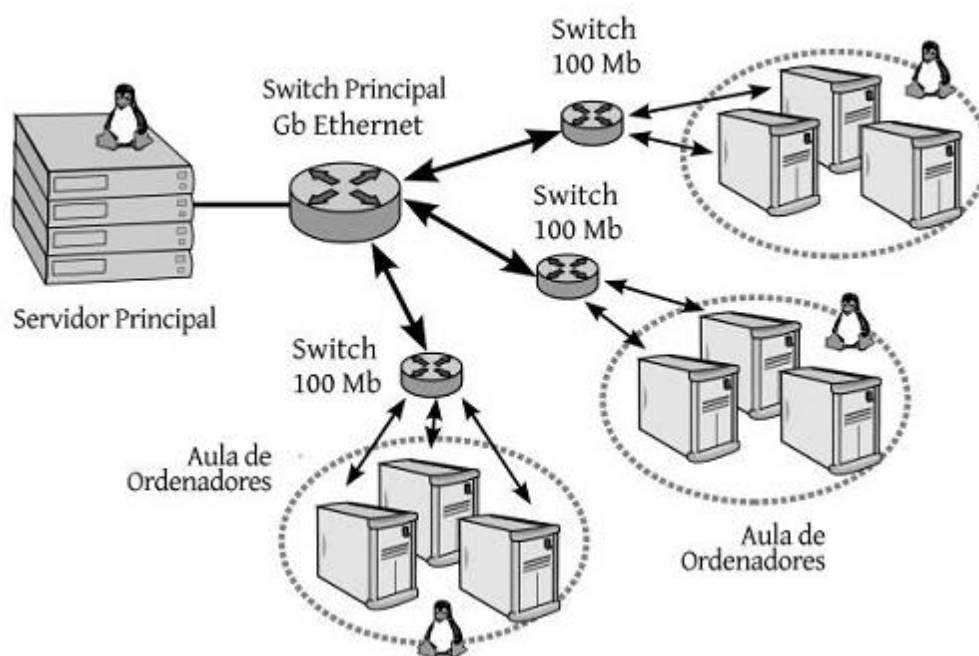


Imagen 8.- Requisitos para implementar el clúster de renderizado.

Para generar el clúster de renderizado se pretende utilizar Open Mosix como la herramienta base para construir el clúster sobre sistemas Linux con licencia GNU. Este utiliza la estrategia de trabajar como una máquina multiprocesador y permite el balanceo de carga entre todos los nodos que forman el clúster: migra procesos, independientemente de en qué nodo se han originado, al nodo con menos carga

de trabajo. Su mayor ventaja es que las aplicaciones que ejecuta no deben estar programadas para Open Mosix ya que trabaja con aplicaciones sin paralelizar; permite al usuario utilizarlo sin conocer su funcionamiento, es transparente. Tiene una limitante: sólo migra procesos que no usen memoria compartida, por lo que no migra procesos multihilo.

6.2.1. Software Open Mosix

Open Mosix es una herramienta diseñada para realizar balanceo de cargas en un clúster de forma totalmente transparente a tal grado de que los nodos del clúster se comportan como si fuera solamente una maquina y de esta manera incrementar el aprovechamiento de cada uno de los nodos.

El núcleo de la herramienta Open Mosix está formado por un conjunto de algoritmos diseñados para responder dinámicamente a las variaciones de carga entre los nodos que forman el clúster, migrando los procesos de un nodo a otro, con el fin de mejorar el desempeño.

Los usuarios pueden correr aplicaciones paralelas inicializando múltiples procesos en un nodo, entonces la tarea de Open Mosix es asignar estos procesos a los nodos con mejores recursos disponibles (procesador más rápido, más memoria RAM, mayor tamaño en disco duro, etc). Los algoritmos de Open Mosix están diseñados para monitorear constantemente los nodos, para asignar y reasignar procesos, esto es fundamentalmente importante porque da un uso eficiente de todos los recursos con los que cuenta cada nodo.

La propiedad más notable de la ejecución e aplicaciones en Mosix son las políticas de distribución de procesos y la flexibilidad de configuración de los nodos del clúster. Estas aplicaciones pueden ser ejecutadas creando muchos procesos de tal forma que parezca ante el usuario como si fuera solamente un nodo el que ejecuta todos los procesos.

6.2.2. Características de Open Mosix

Las características de Open Mosix que se muestran a continuación están implementadas a nivel del Kernel del sistema operativo de Linux.

- **Algoritmo de dispersión de información probabilística**, mediante estos algoritmos se tiene conocimiento suficiente de los recursos con que cuenta cada nodo. Además todos los nodos envían en cada determinado tiempo información acerca de sus recursos disponibles en ese instante, para que de esta forma el nodo principal les pueda asignar o reasignar procesos, al mismo tiempo cada nodo mantiene un buffer con información enviada a éste.
- **Migración de procesos preventivos**, esto significa que un usuario puede migrar procesos de forma manual a un nodo disponible ya que la otra forma mediante la asignación automática.
- **Balanceo de carga dinámica**, continuamente se intenta balancero y reducir la carga entre pares de nodos, es decir, cada vez que un nodo tiene un exceso de carga siempre se busca otro nodo que esté ociosos para asignarle trabajo, este mismo algoritmo es asignado por cada uno de los nodos del clúster Open Mosix.
- **Acomodo de memoria**, es un algoritmo que se activa cuando un nodo no tiene la memoria suficiente para ejecutar los procesos que se le han asignado. En estos casos se activa también el algoritmo de balanceo de carga dinámica para asignar los procesos a otros nodos.
- **Comunicación eficiente de Kernel**, esta característica fue especialmente diseñada para reducir la sobrecarga de comunicación entre los diferentes núcleos del clúster, con esta característica la asignación de intercomunicación de procesos se hace más eficiente.

6.2.3. Componentes necesarios para construir el clúster son:

1. Parche para Open Mosix sobre el kernel Linux
2. Instalación y configuración de MFS; sistema de archivos compartido
3. Configuración del demonio omdiscd para localizar los nodos en el clúster
4. Instalación y configuraron las herramientas de administración para el usuario desde línea de comandos Open Mosix-User y Open MosixView
5. Obtención de pruebas de stress para verificar el correcto funcionamiento del clúster

Después de verificar el funcionamiento del clúster, ahora se instala y configura el Blender y su alternativa de render POV-Ray; una de las ventajas es que Blender se ejecuta como un único proceso y Open Mosix puede paralelizarlo en una forma eficiente; se ejecutan varios programas Blender en el clúster y esto permite aumentar la rapidez del renderizado de un proyecto. El proceso de generación de un proyecto de render sigue los pasos siguientes:

1. Se obtienen los archivos .blend de una escena o proyecto; adicionalmente es posible con Blender cargar un proyecto de otras herramientas como 3D Max Studio o Maya (.3ds) y convertirlos a este formato.
2. Se divide el proyecto entre los nodos del clúster utilizando un conjunto de scripts disponibles en Open Mosix.
3. Se generan los archivos del render y se reúnen en un único archivo utilizando la herramienta mencoder; la cual puede generar archivos .avi y .mpg
4. Si se desea utilizar un render para texturas y posiciones de cámara se utiliza el KpovModeler y otras bibliotecas de POV-Ray para generar escenarios de recorridos virtuales.

6.3 Inversión y periodo de recuperación

Se estima que para la puesta en marcha del proyecto se haría una inversión inicial de **\$24,977.00** contando con los requerimientos de hardware y software para la creación del clúster de renderizado, el material necesario para la construcción del clúster y sus costos se muestra en la siguiente tabla:

Material	Costo Unitario	Costo Total
5 Computadoras Acer Procesador Intel Pentium (1.20GHz), Memoria de 1GB DDR2 D.D. de 250GB DVD±R/RW DL Tarjeta de Red 10/100 Mbps Monitor 15".	\$ 5,069.00	\$ 23,345.00
30 mts. cable de red UTP categoría 5e	\$ 10.60 mt.	\$ 318.00
15 Conectores RJ45	\$ 1.00	\$ 15.00
1 Switch 3Com Baseline Switch de 16 Puertos 10/100Mbps	\$ 1,299.00	\$ 1,299.00
Software libre para la Elaboración modelos 3D	Sin costo	\$ 0.00
Software libre S.O. Open Mosix	Sin costo	\$ 0.00
Mano de Obra	Gratuita	\$ 0.00
Total		\$ 24,977.00

Tabla 13.- Costo de implementación del clúster de renderizado

Referente a la mano de obra para la implementación y mantenimiento del sistema se hará cargo la persona encargada del laboratorio de multimedia con el apoyo de estudiantes de estancia estadia y de alumnos interesados en el proyecto.

La capacitación que se requiere tanto para alumnos como profesores la realizarán las personas involucradas en el proyecto, es decir el encargado de laboratorio y los profesores encargados del proyecto sin ningún costo adicional, puesto que son parte de sus funciones dentro de la Universidad.

Si consideramos el costo de implementación del clúster contra el costo de las penalizaciones por la no entrega a tiempo del producto final que equivale a un 20% del precio convenido sobre la elaboración del modelado tridimensional, podemos ver claramente, que si se invirtiera en la puesta en marcha del clúster el costo de recuperación sería de forma inmediata.

Proyecto	Costo total del Proyecto	Costo de Penalización	Diferencia
1	\$105,000.00	\$ 21,000.00	\$ 84,000.00
2	\$67,500.00	\$13, 500.00	\$ 54,000.00
3	\$ 195,000.00	\$ 39,000.00	\$ 156,000.00
Total		\$ 73,500.00	

Tabla 14.- Costo total de penalizaciones.

7. CONCLUSIONES

Los requisitos computacionales que son necesarios para el renderizado de calidad fotorrealistas son enormes y, por tanto, obtener resultados en un tiempo razonable y en una sola computadora es prácticamente imposible (incluso existen más dificultades en el caso de las animaciones). Muchas propuestas, basadas en diferentes tecnologías, han sido expuestas en este artículo. El clúster basado en el sistema Open Mosix presenta algunas características interesantes: tales como buen rendimiento en el renderizado de animaciones, el sistema divide cada fotograma de la animación en diferentes nodos del clúster, el procesamiento de nodos es usado durante tiempos ociosos (por ejemplo, durante la noche). Una de las principales ventajas de este sistema distribuido es la escalabilidad o el poder incrementar el número de nodos para obtener un mayor proceso de cómputo. De esta forma podemos determinar en base a las hipótesis generadas en el desarrollo del proyecto que:

1. El tiempo de renderizado (Y) disminuye conforme se utilice un distinto Motor de renderizado (X1). Como conclusión podemos afirmar que efectivamente dependiendo del motor de render que se utilice para el proceso de renderizado, este proceso disminuirá sin sacrificar nitidez, definición de colores y definición a la imagen o modelo renderizado.
2. El tiempo de renderizado (Y) disminuye conforme la cantidad de fotogramas (X2) disminuye. En esta hipótesis podemos decir que entre menos fotogramas se utilicen efectivamente reduce significativamente el tiempo de renderizado de un modelo tridimensional, lo que conlleva a que en la elaboración de un modelo animado al momento de reproducirse se vea afectado el movimiento que se quiere representar; por lo tanto al realizar una animación se requiere de un número constante de fotogramas 25 cuadros por segundo, para lograr el efecto de movimiento.

3. El Tiempo de renderizado (Y) disminuye conforme se utilice una distinta técnica de renderizado (X3). A pesar de la aparición de nuevas técnicas y algoritmos, el costo computacional de este proceso es elevado ya que necesita gran cantidad de tiempo para ser realizado, especialmente cuando la escena es compleja o bien cuando es necesario obtener imágenes fotorrealistas, por lo tanto existen técnicas (las expuestas en este documento) que nos permiten reducir el tiempo de renderizado dependiendo de las características del modelo tridimensional tales como geometría de cada objeto, materiales, texturas, fuentes de luz y cámaras virtuales, elementos que intervienen en el tiempo de renderizado.

8. RECOMENDACIONES

1. Realizar investigaciones sobre la creación de una plataforma que le permita al usuario optimizar el proceso de renderizado bajo las siguientes condiciones:
 - a) Que la plataforma tenga la capacidad de sugerir configuraciones de parámetros de render que maximicen la calidad de los resultados obtenidos y minimicen el tiempo de ejecución de los algoritmos de render empleados, en base a un modelo tridimensional.
 - b) Que tenga la capacidad de ofrecer estimaciones precisas del tiempo de render requerido y la calidad del resultado producido por una instancia del proceso de render configurada según el usuario sobre una escena determinada.
2. Otra línea de investigación es la posibilidad de generar un sistema que trate de dar asistencia al usuario en el proceso de render mediante técnicas de análisis de datos basadas en Data Mining “*Conjunto de técnicas de análisis de datos encaminadas a obtener información de una Base de Datos*” (Florida University of Central, 2011) o generación de contenidos en base al conocimiento, pues esto implicaría disminuir el tiempo de procesamiento de los sistemas. Por ello, es importante generar contenidos de alta calidad que permitan al sistema ofrecer sugerencias y estimaciones de la forma más precisa posible sobre configuraciones de render diferentes para una escena determinada en base a parámetros o criterios del modelo tridimensional solicitados por el cliente.
3. Dado que sólo son cinco nodos los que conforman el clúster en la actualidad, un trabajo futuro es aumentar el número de nodos para tener así mayor procesamiento de computo y resolver el mismo problema de

renderizado de modelos tridimensionales en un tiempo más pequeño, inclusive el clúster podría usarse para resolver otro tipo de aplicaciones tales como, formación de atmósferas, simulación de vuelos de aviones, simulaciones de procesos matemáticos, procesamiento digital de imágenes, etc.

4. Otra de las líneas de investigación que está tomando bastante importancia consiste en la utilización de granjas de GPU (Unidad de Procesamiento Gráfico.) para realizar tareas de render. Las ventajas de estas últimas con respecto a los habituales clúster de computadoras radica en que es posible conseguir un muy buen rendimiento a un menor costo.
5. Otra recomendación sería utilizar distintos motores de render tales como: Blender 3D, Toxic, Brazil, etc; con la finalidad de diferenciar mediante las técnicas que implementan, por las características propias de renderizado y por si son libre, gratuitos o comerciales, el tiempo que tardan en realizar el proceso de renderizado ya sea de forma independiente es decir en un solo nodo o mediante la utilización del clúster.
6. Implementar un sistema GRID como una herramienta e-cliente, para que el cliente pueda acceder remotamente a servicios de render a través de internet, en donde el usuario una vez registrado, puede enviar trabajos de render al sistema que serán procesados por un clúster que pertenece a la empresa en cuestión. La única diferencia reside en que el servicio es accesible a través de Internet, pero quien realiza el trabajo sigue siendo un clúster de computadores.

BIBLIOGRAFÍA

- ABC. (2007). Recuperado el 10 de Julio de 2011, de <http://www.definicionabc.com/general/fotograma.php>
- ADOBE. (2011). Recuperado el 05 de julio de 2011, de http://help.adobe.com/es_ES/Photoshop/10.0/help.html?content=WSfd1234e1c4b69f30ea53e41001031ab64-7943.html
- ALEGSA. (2011). Recuperado el 30 de Julio de 2011, de <http://www.alegsa.com.ar/Dic/renderizacion.php>
- Alegsa. (2011). *Alegsa*. Recuperado el 10 de Julio de 2011, de <http://www.alegsa.com.ar/Dic/animacion.php>
- ALEGSA. (2011). *Definición de Renderizado*. Recuperado el 30 de Junio de 2011, de <http://www.alegsa.com.ar/Dic/renderizacion.php>
- ALEGSA.com.arg. (2010). *Diccionario de Informática*. Recuperado el 14 de julio de 2011, de <http://www.alegsa.com.ar/Dic/renderizacion.php>
- Carabias, J. (2008). *Paraleliza con Doctor Queue*. Recuperado el 30 de Julio de 2011, de <http://tecnobetas.com/paraleliza-con-doctor-queue/>
- Castell, C. (2006). *3D Studio Max, Manual Práctico*. México, D.F.: RaMa.
- Centro de Radioastronomía y Astrofísica*. (2011). Recuperado el 10 de Mayo de 2010, de <http://www.crya.unam.mx/web/>
- CESAE. (23 de Mayo de 2010). <http://www.cesae.es>. Recuperado el 23 de Mayo de 2010, de <http://www.cesae.es>:
http://www.gestionempresarial.info/VerItemProducto.asp?Id_Prod_Serv=34&Id_Sec=13
- Coordinación General de Universidades Tecnológicas*. (29 de Marzo de 2011). Recuperado el 10 de Mayo de 2011, de <http://cgut.sep.gob.mx/>
- Debattista, K., Longhurst, P., Chalmers, A., and Troscianko, T. . (2005). Visual attention for e_cienthigh-_delity graphics. *In Spring Conference on Computer Graphics (SCCG 2005)*, (págs. 162-168).
- Diana L. REYES, Alfonso BARBOSA, César J. BUSTACARA. (s.f.). *Ingeniería de Sistemas, Pontificia Universidad Javeriana, Bogotá D. C., Colombia*. Recuperado el 06 de julio de 2011, de [http://www.iiisci.org/journal/CV\\$/risci/pdfs/GC741SS.pdf](http://www.iiisci.org/journal/CV$/risci/pdfs/GC741SS.pdf)
- Driemeyer. (2008). *Rendering with Mental Ray*.

- DrQueue. (2011). Recuperado el 30 de Julio de 2011, de <http://www.drqueue.org/cwebsite/>
- Electrónico, C. A. (2010). *Animación Tridimensional*. Morelia.
- Fetter, W. (2005). computer graphics. En W. Fetter, "*computer graphics" for his human factors cockpit drawings*.
- Finder, A. (2010). *What does NTSC stand for*. Recuperado el 30 de Julio de 2011, de <http://www.acronymfinder.com/NTSC.html>
- FIPA. (2002). Recuperado el 03 de julio de 2011, de <http://www.esi.uclm.es/www/David.Vallejo/docs/papers/cedi2007.pdf>
- Florida University of Central. (2011). Recuperado el 30 de Julio de 2011, de <http://dms.stat.ucf.edu/>
- flytech. (2007). *Granjas de Render*. Recuperado el 11 de Mayo de 2011, de http://www.flytech.es/productos/granjas_de_render
- Garola, J. S. (2009). *Técnicas de renderizado*. Madrid, España: E-Prints.
- Gonzalez, F. (2009). *Recorridos Virtuales*. México, D.F.
- González, M. C. (2009). *Fundamentos de Imágenes en Blender 3D*. Madrid, España: Marcombo.
- Hachisuka, T. (2005). *High-Quality Global Illumination Rendering using Rasterization*. .
- Holzner, S. (2006). *Java 2*. E.U.: Anaya Multimedia.
- I. Buck, T. Foley, D. Horn, J. Sugerman, K.Fatahalian, M. Houston, P. Hanrahan. (2004). Brook for GPUs: Proceedings of SIGGRAPH. *Stream Computing on Graphics Hardware*.
- I. Foster, C. Kesselman, S. Tuecke. (s.f.). *The Anatomy of the Grid: Enabling Scalable Virtual Organizations. International Journal of Supercomputing Applications*. Recuperado el 06 de 07 de 2011, de <http://www.ati.es/novatica/2007/190/Nv190-41.pdf>
- Isopixel. (2010). *Resolución, tamaño de imagen, tamaño de archivo*. Recuperado el 05 de Julio de 2011, de <http://isopixel.net/archivo/2003/11/resolucion-tamano-de-imagen-y-tamano-de-archivo/>
- James, W. (1980). *Psicología Mirada Integral*. Recuperado el 13 de Junio de 2010, de sitio Web de Psicología Mirada Integral: <http://www.psicologiamiradaintegral.com/modules.php?name=Content&pa=showpage&pid=78>
- Kenneth C y Jane P. Laudon. (s.f.). *Infraestructura Tecnológica*. Recuperado el 01 de Julio de 2011, de <http://sistemadecomputo.tripod.com/index.html>
- Kotler, P. (2006). *Dirección de Marketing*. Mexico: Pearson.

Kuhlo, M. (2010). *Representación arquitectónica con 3ds Max y V-Ray: visualización fotorrealista*. Focal Press.

López, J. (29 de Enero de 2010). *Los motores de renderizado de los navegadores WEB*. Recuperado el 16 de julio de 2011, de <http://www.circulodemaquetadores.com/motores-de-renderizado-navegadores>

MEDIAactive. (2009). *El gran libro de 3ds Max 2009*. México, D.F.: Alfaomega.

Morelia, U. T. (s.f.). *Universidad Tecnológica de Morelia*. Recuperado el 10 de Mayo de 2011, de <http://www.utmorelia.edu.mx/>

Narodowski, M. (28 de Febrero de 2007). *Opinion: El Clarin, Calidad Educativa no implica satisfacción del cliente*. Recuperado el 23 de Mayo de 2010, de <http://www.clarin.com/diario/2007/02/28/opinion/o-02501.htm>

Noticias3D. (2010). *Motor de Render*. Recuperado el 03 de Julio de 2011, de <http://www.noticias3d.com/articulo.asp?idarticulo=156&pag=1>

NVIDIA. (2011). *NVIDIA Gelato*. Recuperado el 16 de julio de 2011, de <http://www.nvidia.es/page/gelato.html>

Peivem. (Abril de 2007). Recuperado el 30 de Julio de 2011, de <http://www.peivem.com/?post=24>

Pele-Sol Engineered Solutions. (2010). *What is Business Process Automation?: Pele-Sol Engineered Solutions*. Recuperado el 20 de Julio de 2010, de sitio Web Pele-Sol Engineered Solutions: http://www.pele-sol.com/pele_factsheet_new.pdf

PIERRE, S. (1996). *Elementos para una teoría del fotograma*. Valencia, España: Fernando Torres.

R. Ra jagopalan, D. Goswami, S.P. Mudur. (2005). *Functionality Distribution for Parallel Rendering*.

Serrano, V. H. (2011). *Introducción al modelado tridimensional en tiempo real*. Recuperado el 02 de julio de 2011, de <http://www.ixtli.unam.mx/pdf/notas-intro3d.pdf>

Tanaka, K. (1998). *An introduction to fuzzy logic for practical applications*. Recuperado el 03 de Julio de 2011, de <http://www.esi.uclm.es/www/David.Vallejo/docs/papers/cedi2007.pdf>

Techterms. (2009). Recuperado el 16 de julio de 2011, de <http://www.techterms.com/definition/plugin>

video2brain. (2011). Recuperado el 30 de Julio de 2011, de <http://www.video2brain.com/mx/>

Vila, C. (Julio de 2011). *Eterea Studios*. Recuperado el 07 de julio de 2011, de http://www.eteraestudios.com/training_img/intro_3d/intro_3d.htm

Wikipedia. (2010). *Frame*. Recuperado el 01 de julio de 2011, de <http://es.wikipedia.org/wiki/Frame>

Wikipedia. (12 de Julio de 2011). *Imagen Fotorrealista*. Recuperado el 30 de Julio de 2011, de <http://es.wikipedia.org/wiki/Fotorrealismo>