Universidad de las Ciencias Informáticas
Facultad V

Título: Métodos Realistas de Iluminación Para Juegos 3D

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

Autor(es): Yirka Céspedes Boch

Yoander Cabrera Díaz

Tutor: Lic. Eduardo Lago Aguilar

Co-tutor: Ing. Igr Alexander Fernández Sauco

Asesor: MSc. Pedro Carlos Pérez Martinto

Ciudad de La Habana, Julio 2007

DECLARACIÓN DE AUTORÍA

Declaramos ser autores de la presente tesis y reconocemos a la Universidad de las Ciencias Informáticas los derechos patrimoniales de la misma, con carácter exclusivo.

Para que así conste firmo la presente a los ____ días del mes de _____ del año _____.

_____	_____	_____
Firma del Autor	Firma del Autor	Firma del Tutor

AGRADECIMIENTOS

A Dios. A mi abuelita que está en el cielo, gracias a ella es que he llegado hasta aquí. A mi mamita por ser tan comprensiva, por haberme brindado tanta confianza y ayudarme en todo lo que he necesitado.

A mi tutor Eduardo Lago por su sabio y valioso asesoramiento en la realización de este trabajo y por el apoyo que fraternalmente me ha brindado.

A mi esposo que es el mejor hombre que he conocido en mi vida. A toda mi familia, en especial a mis tías(os) María de los Ángeles, María Eugenia, María Dolores, Gerardo y Joseito, también a mi prima Yanetsi que es para mí como una hermana. A mis suegros por haberme apoyado como una hija más en los estudios desde que estoy en el Pre-Universitario, al igual que a toda la familia de mi esposo por todo el amor que me han brindado.

A nuestro Comandante en Jefe Fidel Castro y a la Revolución, que me han enseñado de qué parte está el deber y el valor infinito de la solidaridad humana. A mis profesores desde la primaria hasta la actualidad, especialmente a Zoraida.

A todos mis compañeros de estudios universitarios en especial a Lenna, Yailín, Dalay, Lidibet, Diana, Lisbety, Yulien, Yalina etc. A los que ahora son mis compañeros de trabajo en la Facultad Regional de Ciego de Ávila y que ha sido lo mejor que me ha podido suceder en estos últimos meses de la carrera, a Naryana, Pepe, Yadainy, Mirialys, Dunia y Lianet.

De Yirka

A mis padres por haberme dado siempre su apoyo, comprensión y guiado por el buen camino del estudio. A toda mi familia, en especial a mis abuelitos, que de una forma u otra han contribuido en mi educación como persona y han incentivado en mí el deseo de ser alguien útil en la vida para de esta manera ayudar a la Revolución, desde el lugar que me corresponda. A mis profesores desde que comencé a leer y a escribir porque sin ellos nunca hubiese podido realizar

este gran sueño de ser Ingeniero en Ciencias Informáticas, hasta los que hoy me están formando como profesional.

A mi esposa que a la vez es mi compañera de tesis, por haber compartido tantos años de estudios y estar junto a mí en los buenos y malos momentos de la carrera.

A mis compañeros de la Universidad que se han convertido en una gran familia para mi en estos cinco años de estudios en especial a Lenna, Dalay, Lidibet, Lisbety entre otros.

A mi tutor por apoyarme en todo lo que ha hecho falta para que este trabajo quede con la calidad requerida.

A nuestro Comandante en Jefe por ser el faro que ilumina nuestro país, gracias a él y a la Revolución he podido estudiar esta gran carrera.

De Yoander

DEDICATORIA

A mi tutor Eduardo Lago por su apoyo y preocupación en la realización de este trabajo. A mi abuelita que es lo que más amo en el mundo. A mi mamá porque sin ella nunca hubiese podido estudiar tan lejos gracias a su apoyo y comprensión. A toda mi familia.

A mis suegros porque han sido mis segundos padres y me considero una hija más para ellos, por todo su amor y por apoyarme en todos los momentos buenos y malos de mi vida. A todos mis profesores en especial a Zoraida.

A Fidel Castro por ser el único padre que he tenido, porque siempre he confiado en él más que en nadie y a él y a esta gran Revolución me debo.

De Yirka

A mi tutor Eduardo Lago por siempre estar cuando lo he necesitado; por su gran ayuda a la realización de este trabajo para que quede con la calidad requerida. A mis padres porque sin su apoyo, amor y comprensión alcanzar esta meta hubiese sido más difícil. A toda mi familia, en especial a mis abuelitos que tanto han colaborado en mi educación y amor por el estudio. A todos mis profesores, desde la primaria hasta la Universidad.

A Fidel, a la Revolución y a la Universidad de las Ciencias Informáticas.

De Yoander

"…lo fundamental es hacer algo nuevo cada día y luego perfeccionar lo que se ha hecho el día anterior…"

Ché

RESUMEN

Debido a la carencia de sensación de realismo en las escenas de los video-juegos que desarrolla el Grupo de Proyectos de Juegos Virtuales (GPJV) de la Facultad 5 (F5), así como la calidad en su visualización, se ha hecho un estudio de las técnicas modernas de iluminación local/global y sombreados poligonales utilizados en la actualidad para la creación de juegos 3D, con el fin de lograr escenas tridimensionales realistas con un elevado número de polígonos en los video-juegos que se desarrollan.

Dado a la gran complejidad del tema investigado, el mismo se resume en caracterizar, expresar matemáticamente e implementar el modelo de iluminación local (Phong), y el sombreado poligonal (Gouraud Shading) planteados para el uso particular de los video-juegos en etapa de tiempo real; utilizando el lenguaje de shaders OpenGL Shading Language (GLSL) mediante la herramienta RenderMokey; proponiendo a C++ como lenguaje de programación para visualizar los shaders en combinación con la biblioteca open source Games3D (G3D) en el entorno de Visual Studio .NET 2003.

Para etapa de pre-procesamiento se han caracterizado y formulado matemáticamente el método global (Radiosidad) y el sombreado poligonal (Phong Shading). Proponiéndose su futura implementación en Visual C++.Net 2003 empleando la librería gráfica G3D y en GLSL respectivamente, para el desarrollo de una aplicación que genere imágenes fotorrealistas.

El trabajo tiene como resultado obtener algoritmos de iluminación factibles en consecuencia con la potencia de procesamiento de la institución debido a la importancia de la luz en la creación de video-juegos ya que esta promueve la tridimensionalidad de los objetos y logra cierta sensación de continuidad visual en las escenas.

PALABRAS CLAVES

Shaders, Iluminación, Juegos, Tiempo Real, Pre-procesamiento, Pipeline.

ÍNDICE

AGRADECIMIENTOS ..... **I**
DEDICATORIA ..... **III**
RESUMEN ..... **V**
INTRODUCCIÓN **1**
1 MODELOS DE ILUMINACIÓN LOCAL/GLOBAL Y SOMBREADOS POLIGONALES. FUNDAMENTACIÓN DE LAS TECNOLOGÍAS **4**
1.1 Introducción 4
1.2 La Luz 4
1.3 Fenómeno Básico de Iluminación 5
1.4 Modelos de Iluminación Local 8
1.4.1 Componente Ambiental *9*
1.4.2 Componente Difusa *10*
1.4.3 Componente Especular. Modelo de Phong *14*
1.4.4 Modelo de Iluminación de Blinn *15*
1.4.5 Comparación entre los Modelos Phong y Blinn *17*
1.5 Modelos de Iluminación Global 17
1.5.1 Trazado de Rayos Recursivo *18*
1.5.2 Radiosidad *19*
1.5.3 Comparación entre los Métodos de Rendering *23*
1.6 Sombreado de Polígonos 24
1.6.1 Sombreado Constante o Flat *25*
1.6.2 Sombreado de Gouraud *26*
1.6.3 Sombreado de Phong *28*
1.6.4 Comparación entre el Sombreado de Gouraud y el Sombreado de Phong *29*
1.7 Fundamentación de las Tecnologías 30
1.7.1 RenderMonkey *30*
1.7.2 Librería Gráfica OpenGL y Lenguajes de Programación de Shaders *31*
1.7.3 Lenguaje de Programación Visual C++. Net *33*
1.7.4 Biblioteca Gráfica Multiplataforma G3D *33*
1.8 Consideraciones Finales 34

2 SOLUCIÓN PROPUESTA Y ANÁLISIS FINANCIERO. 34
2.1 Introducción 34
2.2 Plataforma Recomendada 34
2.2.1 Hardware 34
2.2.2 Software 34
2.3 Pipeline Gráfica Programable 35
2.4 Pre-Procesamiento 37
2.5 Tiempo Real 42
2.6 Importancia del Análisis Financiero 51
2.7 Planificación 52
2.7.1 Cálculo del esfuerzo, tiempo de desarrollo, cantidad de hombres y costo 54
2.7.2 Beneficios Intangibles 59
2.8 Consideraciones Finales 59
CONCLUSIONES 60
RECOMENDACIONES 61
REFERENCIAS BIBLIOGRÁFICAS 64
APÉNDICES 64
GLOSARIO 66

ÍNDICE DE FIGURAS

FIGURA 1.1: PERCEPCIÓN DE LA LUZ POR EL OJO HUMANO. 6
FIGURA 1.2: PLANO DE PROYECCIÓN. 6
FIGURA 1.3: INTERACCIÓN ENTRE LUZ Y MATERIAL. 7
FIGURA 1.4: VARIABLES VECTORIALES DEL MODELO DE ILUMINACIÓN. 9
FIGURA 1.5: SUPERFICIE RUGOSA. 11
FIGURA 1.6: ÁNGULO DE INCIDENCIA DE LA LUZ. 11
FIGURA 1.7: REFLEXIÓN ESPECULAR EN UN PUNTO DE UNA SUPERFICIE. 14
FIGURA 1.8: TRAZADO DE RAYOS RECURSIVO. 19
FIGURA 1.9: REGIÓN DEL PLANO QUE CONTIENE LA SUPERFICIE. 21
FIGURA 1.10: ESCENA SIN CÁLCULO DE RADIOSIDAD. 23
FIGURA 1.11: ESCENA CON CÁLCULO DE RADIOSIDAD. 23
FIGURA 1.12: MODELO DE SOMBREADO CONSTANTE. 25
FIGURA 1.13: MODELO DE SOMBREADO DE GOURAUD. 27
FIGURA 1.14: INTERPOLACIÓN DE NORMALES. 28
FIGURA 1.15: MODELO DE SOMBREADO DE PHONG. 29
FIGURA 2.1: PIPELINE GRÁFICA. 36
FIGURA 2.2: ESCENA DESCOMPUESTA EN PATCHES. 37
FIGURA 2.3 SUPERFICIE DIVIDIDA EN PATCHES. 40
FIGURA 2.4: RESULTADO DE LA RADIOSIDAD EN UNA ESCENA. 40
FIGURA 2.5: NORMAL DE LOS POLÍGONOS ADYACENTES. 41
FIGURA 2.6: INTERPOLACIÓN LINEAL DE NORMALES. 42
FIGURA 2.7: APLICACIÓN DE LAS COMPONENTES DE LUZ. 43
FIGURA 2.8: NORMAL DE LOS POLÍGONOS ADYACENTES. 44
FIGURA 2.9: INTERPOLACIÓN DE INTENSIDADES. 45

ÍNDICE DE TABLAS

TABLA 1.1: SOMBREADOS FLAT, GOURAUD Y PHONG PARA UNA ESFERA DE 990 POLÍGONOS Y 497 VÉRTICES. 30
TABLA 2.1: CASOS DE PRUEBA. 51
TABLA 2.2: SALIDAS EXTERNAS. 52
TABLA 2.3: ARCHIVOS LÓGICOS INTERNOS. 53
TABLA 2.4: PUNTOS DE FUNCIÓN DESAJUSTADOS. 53
TABLA 2.5: INSTRUCCIONES FUENTES. 54
TABLA 2.6: MULTIPLICADORES DE ESFUERZO. 56
TABLA 2.7: FACTORES DE ESCALA. 56
TABLA 2.8: COSTO DEL PROYECTO. 59

INTRODUCCIÓN

Cuando se habla de iluminación en Informática Gráfica, se hace referencia al proceso que permite averiguar la intensidad luminosa de cada uno de los puntos de una superficie, basados en la posición, orientación, características de la superficie y fuentes de luz incidentes. Para obtener esta intensidad luminosa se utilizan los denominados modelos de iluminación.

El realismo en las imágenes tridimensionales depende de la correcta simulación de los efectos de iluminación y sombreado. Se utilizan los modelos de iluminación para calcular las intensidades y colores de cada píxel de las imágenes, mientras que las técnicas de sombreado permiten reducir la cantidad de píxels para los que se debe computar el modelo de iluminación.

El modelo de iluminación representa una simulación de los fenómenos que, modificando o filtrando la distribución de energía de la luz incidente, dan lugar a los colores en las imágenes.

El modelado de los colores y efectos de iluminación que se observa en los objetos es un proceso complicado que implica principios físicos y psicológicos. Estos modelos de iluminación se derivan de las leyes físicas que describen las intensidades de luz de las superficies, aunque debido a la complejidad de estos modelos y al gran número de cálculos que implican, se suelen aplicar una serie de simplificaciones carentes de una base firme teórica pero que, en la práctica, dan resultados aceptables.

El proceso de iluminación de escenas aplicado por el Grupo Proyectos de Juegos Virtuales (GPJV) es realizado actualmente mediante lightmaps que genera el Q3radiant en etapa de pre-procesamiento y la aplicación de luces para tiempo real es creada en OpenGL ejecutándose sobre software, no es muy efectivo a la hora de crear grandes aplicaciones que demanden velocidad y calidad en el procesamiento de imágenes en tiempo real, al igual que las escenas que se cargan en pre-procesamiento no quedan con la calidad de realismo visual suficiente, ni se realizan en tiempos aceptables.

A raíz de lo anterior, el **problema científico** se centra en ¿Cómo mejorar la calidad de visualización en las escenas de los video-juegos para ofrecer mayor sensación de realismo, velocidad y nivel de inmersión del jugador mediante la modelación computacional de algoritmos de iluminación que se ejecuten a nivel de hardware?

Del problema científico anterior, se puede definir que el **objeto de estudio** corresponde a: modelos de iluminación empleados en los video-juegos.

Como **objetivo general** se propone desarrollar métodos de iluminación para el logro de una mejor simulación de luces en los video-juegos que desarrolla el GPJV.

Objetivos específicos:

- Implementar un algoritmo de iluminación local y un sombreado poligonal para etapa de tiempo real.
- Modelar matemáticamente un método global de iluminación y un sombreado poligonal para etapa de pre-procesamiento.

El **campo de acción** de la investigación se define como los algoritmos de iluminación para el GPJV que desarrolla el perfil de Entornos Virtuales (EV) en la UCI.

Como **idea a defender** los algoritmos seleccionados para el proceso de iluminación de video-juegos del GPJV posibilitarán entregar mayor realismo y rapidez en el procesamiento gráfico de las escenas.

Tareas Investigativas:

- Caracterizar los modelos de iluminación local, global y sombreados poligonales.
- Seleccionar el modelo de iluminación local, global y sombreados poligonales cuyas características respondan a las necesidades del GPJV.

- Modelar matemáticamente el modelo de iluminación local y el sombreado poligonal seleccionado.
- Identificar técnicas para el mejoramiento de la eficiencia del método de iluminación global seleccionado.
- Seleccionar las tecnologías y lenguajes de programación.

El trabajo ha sido organizado de la siguiente manera:

Capítulo 1: Se reflejarán algunos conceptos importantes relacionados con la iluminación local/global, así como los diferentes sombreados poligonales utilizados actualmente en el desarrollo de video-juegos, explicándose de manera científica en que consisten los mismos a fin de obtener una solución acorde a las necesidades planteadas; además se justificarán las tecnologías.

Capítulo 2: Serán introducidos un conjunto de términos técnicos que faciliten la comprensión del proceso de iluminación, donde se brinda la oportunidad de introducir los shaders para sustituir etapas de procesamiento de primitivas. Una vez que han sido propuestos los modelos de iluminación y sombreados poligonales a utilizar para lograr una correcta simulación de luces en los video-juegos, se procederá en el capítulo, a la descripción del modelo matemático de los métodos seleccionados e implementación del modelo local y sombreado poligonal para etapa de tiempo real en el lenguaje de shaders propuesto.

Se definirá la plataforma de hardware y software. Se realizará el análisis financiero de los algoritmos programados, mediante el modelo de COCOMO II y serán mostrados algunos casos de prueba.

Capítulo

1

1 MODELOS DE ILUMINACIÓN LOCAL/GLOBAL Y SOMBREADOS POLIGONALES. FUNDAMENTACIÓN DE LAS TECNOLOGÍAS.

1.1 Introducción

Se reflejarán algunos conceptos importantes relacionados con la iluminación local/global, así como los diferentes sombreados poligonales utilizados actualmente en el desarrollo de video-juegos, explicándose de manera científica en que consisten los mismos a fin de obtener una solución acorde a las necesidades planteadas; además se justificarán las tecnologías.

1.2 La Luz

El entorno está rodeado de fenómenos ópticos. Cuando la luz incide sobre una superficie muy lisa esta se refleja especularmente, los espejos en los hogares son buenos ejemplos de reflexión especular. Cuando la superficie se torna rugosa, la luz se refleja no solamente con una componente especular sino que también surge luz reflejada en forma difusa [1].

Fue Newton quien mostró que la luz blanca es una superposición de ondas con diferentes frecuencias, pudiéndose descomponer esta de forma artificial mediante un prisma [1].

Una de las propiedades de la luz es la interferencia: al hacer incidir sobre una pantalla dos haces de luz habrá regiones de la pantalla en donde las ondas que arriban se suman constructivamente creando una intensidad mayor que la que poseen las ondas incidentes, habrán regiones de la pantalla en las cuales las ondas se suman destructivamente, pudiéndose hasta cancelar su efecto [1].

Es a través de la luz que se puede ver la rugosidad de superficies que en primera instancia parecerían lisas, es la luz la que trae la imagen de diminutas bacterias en la punta de una aguja, la que hace ver distantes objetos que forman parte del Universo, aún aquellos ubicados a miles de millones de años luz. No basta conocer solo el origen de esa luz, cuáles son sus fuentes, es necesario tomar en consideración cómo se modifica hasta llegar a nosotros [1].

1.3 Fenómeno Básico de Iluminación

Una escena carece de realismo sin una iluminación correcta, la misma se consigue mediante la ubicación de varias fuentes de luz que interactúan sobre los objetos teniendo en cuenta las propiedades de los materiales de que estén constituidos. La luz es importante en la creación de video-juegos porque promueve la sensación de tridimensionalidad de los objetos. La forma en que el objeto refleja la luz da una idea del material del que está formado [2].

Desde una perspectiva física, la superficie de un objeto puede emitir luz de manera natural (focos de luz, bombillas, etc.) y/o reflejar luz de otras superficies que la iluminan (paredes, espejos, agua, etc.). El color que se ve en un punto de una superficie está determinado por las múltiples interacciones entre las fuentes de luz y superficies reflectoras. Particularmente, los objetos que no emiten su propia luz, reciben tres tipos de luz diferentes: *luz ambiental, luz difusa y luz especular*, que serán llamadas a partir de ahora simplemente componente ambiental, difusa y especular [2].

O sea, iluminar básicamente una escena consiste en:

1. Modelar las fuentes de luz en una escena: ubicación, orientación, coeficientes de atenuación, etc [2].

2. Construir un modelo de reflexión que trate con las interacciones entre materiales y fuentes de luz [2].

Para comprender el proceso de iluminación, se puede comenzar siguiendo los rayos de luz desde un punto fuente, donde el observador ve solamente la luz que emite la fuente y que llega a los ojos; probablemente a lo largo de complejos caminos y múltiples interacciones con objetos en la escena [2].

Si un rayo de luz entra al ojo directamente de la fuente, se verá el color de la fuente. Si un rayo de luz rebota en una superficie que es visible al observador, el color visto se basará en la interacción entre la fuente y el material de la superficie y se verá el color de la luz reflejado de la superficie a los ojos [2]. Véase la figura 1.1

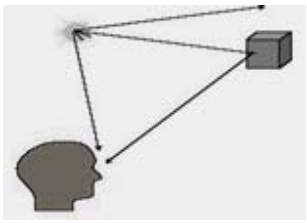


Figura 1.1: Percepción de la luz por el ojo humano.

En términos de Gráficos por Computadora (*GxC*), se reemplaza el observador por el plano de proyección. Véase la figura 1.2

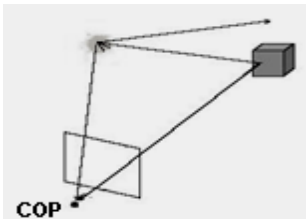


Figura 1.2: Plano de proyección.

El viewport, se mapea a la pantalla. El plano de proyección y su mapeo a la pantalla significa un número particular de píxels de despliegue. El color de la fuente de luz y las superficies determinan el color de uno o más píxels en el frame buffer. Se debe considerar solo aquellos

rayos que dejan las fuentes y llegan al ojo del observador, el centro de proyección, después de pasar por el viewport. Notar que la mayoría de los rayos que dejan la fuente no contribuyen a la imagen [2].

La naturaleza de interacción entre los rayos y las superficies determina si un objeto aparece rojo o rosado, claro u oscuro, reluciente o no. Cuando la luz da en una superficie, parte se absorbe, y parte se refleja [2].

1. Si la superficie es opaca, reflexión y absorción significará toda la luz que dé en la superficie [2].

2. Si la superficie es translucida, parte de la luz será transmitida a través del material y podrá luego interactuar con otros objetos [2].

El sombreado de los objetos también depende de la orientación de las superficies, caracterizado por el vector normal a cada punto. Las interacciones entre luz y materiales se pueden clasificar en tres grupos [2]. Véase la figura 1.3

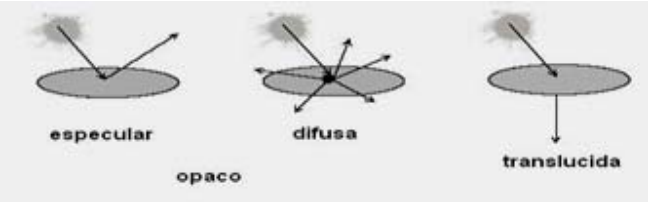


Figura 1.3: Interacción entre luz y material.

1. Las superficies especulares (espejos) se ven relumbrantes porque la mayoría de la luz reflejada ocurre en un rango de ángulos cercanos al ángulo de reflexión. Los espejos son superficies especulares perfectas. La luz del rayo de luz entrante puede absorberse parcialmente, pero toda la luz reflejada aparece en un solo ángulo, obedeciendo la regla que el ángulo de incidencia es igual al ángulo de reflexión [2].

2. Las superficies difusas se caracterizan por reflejar la luz en todas las direcciones. Paredes pintadas con mate son reflectores difusos. Superficies difusas perfectas dispersan luz de manera igual en todas las direcciones y tienen la misma apariencia a todos los observadores [2].

3. Las superficies translucidas permiten que parte de la luz penetre la superficie y emerja de otra ubicación del objeto. El proceso de refracción caracteriza el vidrio y el agua. Cierta luz incidente puede también reflejarse en la superficie [2].

1.4 Modelos de Iluminación Local

Los modelos de iluminación, también llamados de alumbrado se utilizan para calcular el color o intensidad de la luz que se percibe desde un punto determinado en la superficie de un objeto [3].

Son métodos simplificados que se basan en las propiedades ópticas de las superficies, las condiciones de la luz ambiente o de fondo, y las características de las fuentes de luz. Las propiedades ópticas de las superficies se especifican mediante parámetros que permiten controlar la cantidad de absorción y reflexión de la luz incidente [4].

Las fuentes de luz, se consideran como fuentes de luz puntuales que divergen en forma radial desde la posición de la fuente. Este tipo de modelo no tiene en cuenta las interreflexiones de luz entre los objetos de la escena. En estos algoritmos la iluminación global se modela mediante un término de iluminación ambiente que se considera constante para todos los puntos de los objetos [4].

El modelo de iluminación debe contemplar algunos de los comportamientos físicos más notorios de las superficies reales en interacción con la luz visible. Tradicionalmente se han identificado tres componentes distintas en el resultado final de la iluminación sobre un cuerpo dado: la ambiental, difusa y especular [5].

La figura siguiente muestra las variables vectoriales principales que dan origen a dichas componentes. *L (vector de incidencia de la luz), N (vector normal a la superficie), R (vector de reflexión) y V (vector del observador)* [5]. Véase la figura 1.4

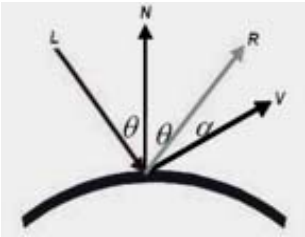


Figura 1.4: Variables vectoriales del modelo de iluminación.

1.4.1 Componente Ambiental

Una superficie que no está expuesta de manera directa a una fuente de luz será visible, aún si los objetos circundantes están iluminados, se trata de un nivel general de brillantez para toda la escena. La componente ambiental describe la luminosidad del entorno de una escena, es decir, la cantidad de luz que incide sobre los objetos desde todas las direcciones de manera prácticamente uniforme. De este modo, las superficies de todos los objetos de una escena recibirán la misma cantidad de luz ambiental, y reflejarán tanta luz como el material del que estén constituidos lo permita [3].

La luz ambiente no presenta características de espacio ni de dirección. La cantidad de luz incidente para cada objeto es constante para todas las superficies y en todas las direcciones. Esto último es modelado mediante una constante de reflexión ambiental $K_a \in [0..1]$, dependiente del material asociado a la superficie a iluminar, y que por lo tanto permite modificar la intensidad de luz ambiente para cada superficie en cada primario. Así, la intensidad ambiental resultante I_{amb} estará dada por la expresión:

$$I_{amb} = I_a K_a$$

donde I_a es la intensidad de luz ambiental en cada primario [3].

1.4.2 Componente Difusa

La reflexión de la luz ambiente es una aproximación de los efectos de iluminación difusa general. Las reflexiones difusas son constantes en cada superficie de una escena e independientes de la dirección de vista. La cantidad fraccional de luz incidente que se refleja de manera difusa se puede establecer para cada superficie con el parámetro K_d, coeficiente de reflexión difuso. A este parámetro se le asigna un valor constante de *0* a *1*, de acuerdo con las propiedades de reflexión que se quiera dar a la superficie. Un valor próximo a *1* indicará una superficie muy reflectante, mientras que para simular una superficie que absorba la mayor parte de la luz incidente, se establecerá un valor cercano a *0* [3].

Esta componente simula el resultado de aplicar una fuente de iluminación puntual de cierta intensidad, que es reflejada de manera uniforme (difusa) por la superficie del objeto. Pero la intensidad de la luz reflejada depende de qué tan grande sea la componente del vector normal de la superficie *N* en la dirección de la fuente de luz puntual, es decir, de qué tan parecidos sean el vector normal a la superficie del objeto en un punto determinado y el vector que representa la ubicación de la fuente de luz *L* en el espacio de coordenadas de la escena [5].

Esto hace que los polígonos de un objeto que se encuentren expuestos más directamente a la fuente de luz se vean más brillantes que los que reciben la iluminación de una manera más sesgada [5].

Un reflector difuso perfecto esparce la luz que refleja de manera igual en todas las direcciones, viéndose igual para todos los observadores. Sin embargo, la cantidad de luz reflejada depende del material, dado que parte de la luz es absorbida, de la posición de la fuente de luz relativa a la superficie. Reflexiones difusas son caracterizadas por superficies rugosas [3]. Véase la figura 1.5 (corte transversal).

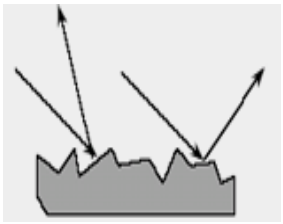


Figura 1.5: Superficie rugosa.

Los rayos de luz que inciden en la superficie son ángulos levemente diferentes. Superficies difusas perfectas son tan rugosas que no hay un ángulo preferido de reflexión. Tales superficies son a veces conocidas como superficies Lambercianas, pudiéndose modelar matemáticamente por la Ley de Lambert la cual plantea que la componente difusa de la luz reflejada por una superficie es proporcional al coseno del ángulo de incidencia [2].

La ecuación para la componente difusa se puede expresar como:

$$I_{dif} = I_p K_d \cos\alpha$$

donde I_p es la intensidad de la fuente de luz puntual y $\alpha \in [0^\circ..90^\circ]$ ángulo de incidencia de la luz, se observa que la componente difusa, depende del ángulo de incidencia entre N y L . Véase la figura 1.6

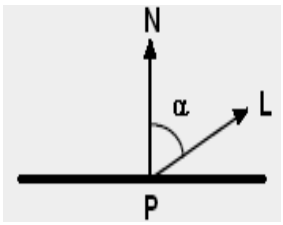


Figura 1.6: Ángulo de incidencia de la luz.

Si N es el vector normal unitario para una superficie, como se puede apreciar en la figura anterior, y L el vector de dirección unitario para la fuente de luz, entonces $\cos\alpha = N \cdot L$, y la ecuación anterior se puede expresar como:

$$I_{dif} = I_p K_d (N \cdot L)$$

Para obtener un ambiente más realista se añade una fuente de luz ambiental a la ecuación de la componente difusa, a fin de obtener una expresión difusa total:

$$I_{dif} = I_a K_a + I_p K_d (N \cdot L)$$

A medida que el ángulo entre los vectores *N* y *L* se hace más pequeño, su producto punto se acerca a uno, la componente difusa se hace más grande, y el efecto de la iluminación puntual más notorio, pués las intensidades de los polígonos que conforman un objeto varían más drásticamente según el cambio de su vector normal [6].

1.4.2.1 Atenuación de la Fuente Luminosa

Considerando el modelo anterior con atenuación del foco hay que tener en cuenta la distancia del foco al punto iluminado d_L.

donde f_{aat} es la inversa al cuadrado de la distancia que viaja desde la fuente de luz a la superficie.

$$f_{aat} = \frac{1}{{d_L}^2}$$

Se utiliza una función de atenuación con la distancia al foco. (Polinomio de segundo grado cuyos coeficientes se ajustan empíricamente).

$$f_{aat} = \min\left[\left(\frac{1}{c_1 + c_2 \cdot d_L + c_3 \cdot {d^2}_L}\right), 1\right]$$

Donde c_1, c_2, c_3 son coeficientes de atenuación constante, lineal y cuadrático respectivamente dependientes de la fuente, d_L es la distancia a la fuente puntual y f_{aat} la función de atenuación ϵ *[0..1]* [6].

La componente difusa con atenuación del foco quedaría:

$$I_{dif} = I_a K_a + f_{aat} I_p K_d (N \cdot L)$$

1.4.2.2 Luces y Superficies Coloreadas

Hasta ahora todo era monocromático. El color difuso de un objeto para la componente frecuencial *λ* se define como $O_{d\lambda}$ para cada componente, como se tienen tres colores básicos *R, G, y B* se aplica la ecuación a cada color [6].

Si se aplica a la ecuación de iluminación el siguiente término, se obtendría:

$$I_{dif\lambda} = I_{a\lambda} K_a O_{d\lambda} + f_{aat} I_{p\lambda} K_d O_{d\lambda} (N \cdot L)$$

1.4.2.3 Atenuación Atmosférica

Para simular la atenuación atmosférica entre el objeto y el observador se suele proporcionar indicaciones de profundidad.

Suponiendo que el observador se encuentra en proyección ortográfica se considerará la distancia como el valor absoluto de *z*.

Se pueden conseguir efectos de atmósfera tomando una intensidad de fondo I_f e interpolando con la $I_{difλ}$ obtenida.

Si $|z| < d_A$ la atmósfera siempre ofrece la misma atenuación mínima S_A.

Si $d_A < |z| < d_B$ la atenuación lineal con la distancia quedaría: $S = S_B + (((|z| - d_B)(S_A - S_B)) / (d_A - d_B))$.

Si $|z| > d_B$ la atenuación máxima constante sería S_B.

La componente difusa con atenuación del observador finalmente quedaría:

$$I' = S \cdot I_{dif\lambda} + (1 - S) \cdot I_f$$

donde I' representa la intensidad del punto observado después de aplicar el efecto atmosférico, $I_{difλ}$ indica la intensidad luminosa del punto calculada según el paso anterior, I_f indica la intensidad luminosa de fondo debido al efecto atmosférico, S coeficiente de mezcla de la iluminación atmosférica y de la iluminación recibida del objeto que depende de la distancia desde el observador al punto observado [6].

1.4.3 Componente Especular. Modelo de Phong

El modelo de iluminación de Phong trata de simular el comportamiento de ciertos materiales, cuya superficie regular hace que la luz sea reflejada de manera uniforme (especular), produciendo un brillo notorio en las partes de su superficie que reflejan la luz en una dirección cercana al vector en la dirección del observador. Este efecto es más apreciable en los objetos metálicos [5].

Cuando se observa en una superficie brillante iluminada, como metal pulido, se puede apreciar un punto de luz o una mancha brillante en algunas direcciones de vista. Este fenómeno se conoce como *reflexión especular*. En la siguiente figura se puede observar un esquema que representa el fenómeno [3]. Véase la figura 1.7

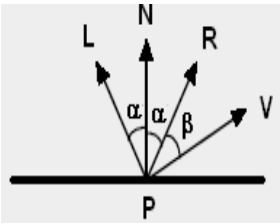


Figura 1.7: Reflexión especular en un punto de una superficie.

En la figura se tiene una superficie y su vector normal unitario N en el punto a iluminar P; L es el vector unitario dirigido hacia la fuente de luz; R el vector unitario reflejado; V el vector unitario que apunta al observador. El ángulo β es el formado por los vectores V y R. Para un reflector ideal (un espejo perfecto) la luz incidente se refleja solo en la dirección de reflexión especular R. En este caso solo se vería la luz reflejada cuando coincidiesen los vectores V y R ($\beta=0$) [3].

Los objetos que no son reflectores ideales presentan reflexiones en un rango finito alrededor del vector *R*. Dicho rango será más amplio cuanto menor sea la brillantez de la superficie [3].

Un modelo empírico para calcular este rango de reflexión, desarrollado por Phong, que recibe el nombre de *modelo de reflexión especular de Phong*, establece la intensidad de la reflexión especular proporcional a $cos^n\beta$. Se pueden asignar valores para el ángulo β en el rango de *0°* a *90°* de modo que el $cos\beta \in [0..1]$ [3].

El valor que se asigna al parámetro o exponente de reflexión especular *n (factor de glossiness)*, determina el tipo de superficie que se quiere visualizar. Así una superficie muy brillante se modela con un valor alto para *n*, y los valores más bajos *(hasta 1)* se emplean para superficies más opacas. Para un reflector perfecto, *n* es infinita; para una superficie rugosa se asignaría a *n* un valor cercano a *1* [3].

La ecuación de reflexión especular dada originalmente por Phong se puede expresar como:

$$I_{esp\,\lambda} = f_{aat} I_{p\,\lambda} k_s O_{s\lambda} \cos^n \beta$$

donde k_s es el coeficiente o factor de reflexión especular, tomando valores entre *0* y *1* para indicar la cantidad de brillo que tiene una superficie; I_p es la intensidad o color de la fuente de luz puntual y $O_{s\lambda}$ color especular de un objeto para la componente frecuencial λ. Si se hace uso del cálculo vectorial, y suponiendo que tanto *R* como *V* son vectores unitarios, se tiene que cos β = $R \cdot V$, se puede calcular el vector de reflexión *R* en términos de los vectores *N* y *L* mediante la expresión $R=2N(N \cdot L)-L$, la ecuación anterior quedaría [3]:

$$I_{esp\,\lambda} = f_{aat} I_{p\,\lambda} k_s O_{s\lambda} (R \cdot V)^n$$

1.4.4 Modelo de Iluminación de Blinn

El modelo de iluminación original de Phong puede resultar un poco simple al considerar un coeficiente de reflexión especular k_s constante para cada tipo de superficie. Esto puede resultar

más o menos cierto para muchos tipos de materiales, pero si se quiere simular otros, como dieléctricos (cristal, cuarzo,...) y conductores (cobre, plata, níquel) de forma más realista, habrá que utilizar modelos de iluminación más complejos basados en las propiedades físicas de dichos materiales [3].

La intensidad de la reflexión especular depende de las propiedades del material de la superficie, así como de otros factores, como la polarización y color de la luz incidente. Se puede modelar de manera aproximada variaciones de intensidad especular al utilizar un coeficiente de reflexión especular, $W(\alpha)$, para cada superficie donde α *(es el ángulo de incidencia de la luz)* [3].

En general, $W(\alpha)$, tiende a aumentar conforme se incrementa el ángulo de incidencia. Para $\alpha = 90°$, $W(\alpha) = 1$, se refleja toda la luz incidente, los materiales transparentes como el cristal, solo presentan reflexiones especulares perceptibles conforme α se aproxima a *90°*. La variante de la intensidad especular con el ángulo de incidencia se describe en la Ley de Reflexión de Fresnel demostrada por Torrance y Sparrow [3].

Blinn, además del término de reflexión de Fresnel, F_λ, para dieléctricos y conductores, incorpora un factor de atenuación geométrica, *G*. El modelo físico original de Torrance - Sparrow asume que las superficies están formadas en realidad por una colección de caras planas microscópicas, cada una siendo un reflector perfecto. La geometría y distribución de estas microfacetas y la dirección de la luz determinan la intensidad y dirección de la reflexión especular [3].

Por último Blinn emplea el vector equidistante *H*, entre *R* y *V*, para calcular el intervalo de reflexiones especulares, *m* el número de luces de la escena. En general, el uso del vector *H* requiere menor cantidad de operaciones que el vector de reflexión *R*, y los resultados son análogos. Así pues, se sustituye el producto escalar $R \cdot V$ del modelo de iluminación de Phong por el producto escalar $N \cdot H$. Teniendo todo esto en cuenta el modelo de Blinn se puede resumir mediante la siguiente expresión [3]:

$$I_\lambda = I_{a\lambda} K_a O_{d\lambda} + \sum_{i=1}^{m} f_{att} I_{p\lambda} [K_d O_{d\lambda} (N \cdot L) + O_{s\lambda} G K_s F_\lambda (N \cdot H)^n]$$

1.4.5 Comparación entre los Modelos Phong y Blinn

- Blinn actúa de forma muy parecida a Phong, con la única diferencia de que las superficies con Blinn poseen resaltes especulares más suaves en el lado posterior del material. El resultado es que el material parece más plano, con poco lustre.

- Blinn posee mayor coste computacional ya que utiliza el término de reflexión de Fresnel e incorpora un factor de atenuación geométrica.

- Phong consiste principalmente en parámetros de color, de resalte especular y de transparencia, al igual que Blinn. Sin embargo los resaltes especulares de la luz posterior son más anchos y más intensos que los de Blinn. En realidad el resalte especular de la luz posterior de un material Phong es más alargado que el de Blinn. Con frecuencia el resultado es un resalte posterior demasiado brillante.

- Phong funciona muy bien con los materiales plásticos y de manera general con todos. Como sus controles son muy sencillos, es fácil establecer sus parámetros y obtener rápidamente materiales bastante aceptables, además es recomendado para la implementación de luces dinámicas por la poca complejidad de los cálculos.

1.5 Modelos de Iluminación Global

Calculan la iluminación o color en un punto dependiendo de la luz directamente emitida por las fuentes y dependiendo también de la luz que alcanza el punto después de la reflexión y transmisión a través de otras superficies. Tradicionalmente se han utilizado dos tipos de algoritmos para generar imágenes teniendo en cuenta la contribución de la iluminación global. Uno de ellos es el conocido algoritmo de Trazado de Rayos, que intercala la determinación de superficies visibles con el cálculo de sombras, reflexiones y refracciones [4].

El otro algoritmo utilizado en iluminación global es el de Radiosidad que separa el cálculo de la iluminación de la determinación de superficies visibles. Este tipo de algoritmo modela todas las interacciones del entorno con las fuentes de luz independientemente de la vista, y posteriormente se calcula una o más imágenes desde los puntos de vistas deseados, utilizando métodos convencionales de obtención de superficies visibles y de sombreados [4].

1.5.1 Trazado de Rayos Recursivo

Consiste en trazar rayos a través de los caminos de reflexión y refracción entre los objetos de la escena. Cada rayo se traza desde el observador, pasando a través de un píxel de la pantalla, hasta encontrar el primer objeto de la escena. Si el rayo no corta ningún objeto *D*, entonces se pinta el píxel de un color de fondo [6].

Cuando intercepta un objeto *A*, se plantea un modelo de iluminación local (por ejemplo, Phong) para determinar el color, pero se computa además un rayo hacia la fuente de luz *E* para garantizar que otros objetos de la escena no estén haciendo sombra sobre el punto de la escena que se está iluminando. Si los objetos de la escena tienen, además, un coeficiente de reflexión especular o de refracción, entonces es necesario computar un factor proveniente de los rayos idealmente reflejados y refractados [6].

La intensidad de cada uno de dichos rayos es computada recursivamente, para lo cual se envían nuevos rayos que experimentan el mismo procedimiento *B, C*, lo cual configura un árbol de rayos, cada uno de los cuales ejerce una influencia en el color del píxel [6]. Véase la figura 1.8

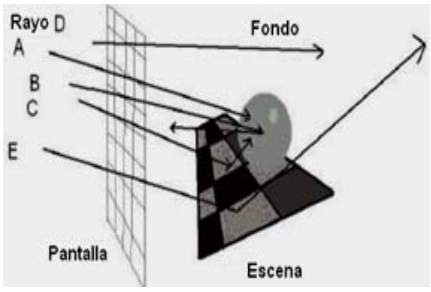


Figura 1.8: Trazado de Rayos Recursivo.

La ecuación para el Trazado de Rayos Recursivo adicionándole el factor reflejado y el refractado quedaría:

$$I_\lambda = I_{a\lambda} K_a O_{d\lambda} + \sum_{i=1}^{m} S_t \, f_{att} \, I_{p\lambda} \left[K_d O_{d\lambda} (N \cdot L) + K_s O_{s\lambda} (N \cdot H)^m \right] + K_s I_{r\lambda} + K_t I_{t\lambda}$$

donde I_r es la intensidad del rayo reflejado, I_t la intensidad del rayo transmitido refractado, se calculan recursivamente y se multiplican por el inverso de la distancia, I_p intensidad de la fuente de luz puntual, I_a intensidad de luz ambiente, $K_a \in [0..1]$ coeficiente de reflexión ambiente, $K_d \in [0..1]$ coeficiente de reflexión difusa, $K_s \in [0..1]$ coeficiente de reflexión especular, $K_t \in [0..1]$ coeficiente de transmisión, $O_{d\lambda}$ color difuso del objeto para la componente frecuencial λ, $O_{s\lambda}$ color especular del objeto para la componente frecuencial λ y f_{aat} factor de atenuación de la fuente de luz. Si la función es continua en K_t, mide el grado de "ocultación" del rayo (sombra) [6].

Ventajas: Modela correctamente el transporte especular y transparente [11].

Desventajas: Susceptibilidad a problemas de precisión numérica, los rayos generados pueden intersecar los objetos de donde salen y los rayos de luz *L* no se refractan en su trayectoria hacia la luz, además de demandar un gran coste computacional [6].

1.5.2 Radiosidad

La Radiosidad es un método de rendering para implementar iluminación global, que utiliza únicamente las componentes difusas de las superficies dentro de una escena. La idea principal

de esta técnica es buscar el equilibrio de la energía emitida por los objetos emisores de luz y la energía que es absorbida por los objetos dentro del ambiente. Es importante que todos los objetos se deban especificar por superficies descompuestas en *patches* [12].

Estos patches no tienen porqué ser necesariamente poligonales pero su tamaño debe ser lo bastante pequeño como para que el cálculo de la Radiosidad sea una buena aproximación a la realidad. La interacción entre todos los patches genera un sistema de ecuaciones muy grande, que hace muy lento e inmanejable la generación de la solución. Este método expresa las transferencias entre superficies por medio de un factor de forma, se puede decir que comprende las siguientes fases [12]:

1. Computar los factores de forma: determinando los polígonos visibles (o proporciones de ellos) desde cada otro polígono, usando algún artefacto auxiliar [12].

2. Resolver la ecuación de la matriz de Radiosidad usando algún método numérico. Como la matriz es diagonal dominante converge rápidamente. Se hace para cada banda de color por separado [12].

3. Mostrar los resultados: se determinan las superficies ocultas y se interpolan los valores de Radiosidad. Para ello es conveniente emplear los sombreados Gouraud o Phong [12].

Es necesario hallar los factores de forma para saber qué porcentaje de energía hay que enviar desde un patche a los demás. La forma directa de efectuar este cálculo es utilizar la integral espacial que define el factor de forma. Desafortunadamente, esta integral solamente puede resolverse de forma analítica para ciertos casos, en concreto cuando los dos patches son circulares o polígonos, o cuando uno de ellos se considera infinitesimal y el otro es un polígono [12].

Sin embargo, los factores de forma deben tener además en cuenta la posibilidad de oclusión o sombra producida por otros objetos. Esta posibilidad resulta un grave inconveniente para los

métodos analíticos, porque habría que introducir en la integral una nueva función H_{ij} que representará la visibilidad entre dos patches diferenciales *i* y *j*, lo que produciría una función total muy complicada y difícil de integrar analíticamente [12].

La otra posibilidad es efectuar la integración de forma numérica o aproximada. En algunos casos se aproxima uno de los dos patches como un punto infinitesimal, lo que facilita mucho el cálculo, para lo cual se puede emplear el Método de Montecarlos el cual permite calcular imágenes muy realistas con un coste aceptable. Este método es usado en cómputos de iluminación global que producen imágenes fotorrealistas 3D, para aplicaciones en los video-juegos [12].

Los métodos de Montecarlos son algoritmos que se utilizan extensamente para simular sistemas complejos mediante la resolución de ecuaciones por métodos de aproximación aleatoria, y utilizados para resolver ecuaciones integrales complejas, aplicables a cualquier tipo de problema.

Por ejemplo, si se tuviese que calcular el área de una superficie de forma extraña se podría escoger una serie de puntos (x_i, y_i) al azar en una región del plano que contiene a la superficie. Si la posición de los puntos ha sido generada con una distribución aleatoria uniforme, el porcentaje de puntos que caen dentro del área buscada respecto al número total de puntos dará el tamaño relativo del área [12]. Véase la figura 1.9

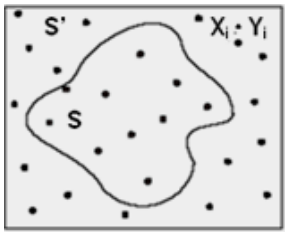


Figura 1.9: Región del plano que contiene la superficie.

Para calcular el factor de forma entre dos patches se tiene que lanzar rayos desde uno de ellos con una orientación al azar. Midiendo el porcentaje de rayos que atraviesan el otro patche se

sabrá lo grande que resulta visto desde el primero. Este sistema tiene la ventaja adicional de que incluye la influencia de la visibilidad, ya que si el rayo atraviesa un objeto intermedio, entonces no debe ser contado [12].

Ventajas del método de Radiosidad:

Guarda las propiedades de la luz en una red sobre la geometría del modelo. Permite calcular la interacción difusa entre objetos. Al considerar solamente las componentes difusas *(uniformes)* de las funciones de reflectancia de los materiales de la escena, la luz recibida y transmitida por los objetos no depende de la posición del observador en la escena. Por tanto, toda la iluminación de la escena es independiente de dónde se coloque el observador o cámara [13].

Esto va a suponer una gran ventaja ya que, aunque el proceso de calcular la iluminación por Radiosidad es muy costoso, sólo será necesario realizarlo una vez, siempre que no cambie la posición de los objetos, sus características materiales o las fuentes de luz de la escena [13].

Esta característica, junto con el hecho de suponer materiales difusores, lo hace muy adecuado para la visualización de escenas, donde no suele haber movimiento de objetos. Las imágenes realizadas con Radiosidad exhiben un tratamiento adecuado para problemas geométricos enormemente complejos, como las sombras suaves estos resultados son notablemente verosímiles, debido a la calidez y armonía que exhiben.

Otra de las ventajas que aporta el método de Radiosidad es que las fuentes de luz se definen utilizando patches extensos de superficie y se consideran a todos los efectos como un objeto más de la escena. Por tanto, pueden modelarse con la forma que se desee y resultan visibles como cualquier otro objeto.

Figura 1.10: Escena sin cálculo de Radiosidad.

Figura 1.11: Escena con cálculo de Radiosidad.

Desventajas del método de Radiosidad:

Por otra parte, el hecho de considerar distribuciones de reflexión totalmente uniformes constituye una importante limitación de este método. Al no considerar la componente especular no se podrá calcular la imagen reflejada de un objeto sobre otro, ni siquiera de forma aproximada. No permite simular efectos ópticos [4].

1.5.3 Comparación entre los Métodos de Rendering

Velocidad: El tiempo de ejecución del Trazado de Rayos puede acelerarse con algunas técnicas. Los tiempos de Radiosidad son más lentos, dado que el cómputo de los factores de forma y la subdivisión adaptativa involucra algoritmos de complejidad potencialmente exponencial, aunque puede ser acelerado por algunas técnicas [9].

Resultados: El método de rendering de Radiosidad es físicamente correcto, y por lo tanto las imágenes son las más verosímiles realizadas hasta la fecha. La falta de iluminación puntual y reflexión especular es una limitación. El Trazado de Rayos, por su parte, produce resultados que

impresionan pero no son siempre creíbles. El modelo de iluminación es incorrecto porque considera separadamente la componente local y la recursiva [9].

Algoritmos: Los algoritmos de Trazado de Rayos son sin duda los más sencillos y adaptados a una metodología de desarrollo. La Radiosidad necesita recurrir a algoritmos para computar el factor de forma y subdividir adaptativamente la escena, los cuales son enormemente complejos y problemáticos [9].

Primitivas geométricas: En el Trazado de Rayos es indispensable contar con objetos cuya intersección con rayos sea fácilmente computable. Poligonizar objetos puede tener un costo prohibitivo porque incrementa enormemente la cantidad de tests de intersección. Radiosidad debe trabajar con poligonizaciones, por lo que en principio tampoco está limitado en la geometría de los objetos [9].

1.6 Sombreado de Polígonos

Un modelo de sombreado, emplea los cálculos de intensidad hallados por un modelo de iluminación para determinar la intensidad de la luz en todas las posiciones de píxel que se proyectan en las diversas superficies de una escena. Se puede efectuar la visualización realista de la escena al aplicar el modelo de iluminación en todos los puntos visibles de las superficies, o mediante la interpolación de las intensidades a lo largo de las superficies, con base a un conjunto reducido de cálculos del modelo de iluminación [3].

Se puede asegurar que ninguna superficie real posee una tonalidad constante, debido a que las fuentes de luz ubicadas en una región limitada del espacio inciden con ángulos e intensidades diferentes sobre los puntos pertenecientes a una superficie dada. Esto se debe a la variación de su distancia respecto a la fuente de luz puntual [5].

Los métodos de sombreados poligonales proporcionan un efecto de suavidad y continuidad a las superficies aproximadas por mallas poligonales. Para modelar la variación de intensidad existen

varias estrategias, siendo más utilizadas las siguientes: Gouraud, que interpola las componentes frecuenciales de color; Phong que interpola el vector normal a la superficie del objeto [5].

1.6.1 Sombreado Constante o Flat

Es el más simple de los modelos de sombreado, debido a que calcula una sola intensidad para cada polígono. Todos los puntos de la superficie del polígono se muestran con el mismo valor de intensidad. El sombreado constante puede ser útil para visualizar con gran rapidez la apariencia general de una superficie curva. El punto donde se aplica el modelo de iluminación es el centro geométrico del polígono. En la siguiente figura se puede observar una esfera utilizando sombreado plano [3]. Véase la figura 1.12

Figura 1.12: Modelo de Sombreado Constante.

Ventajas: Es muy rápido ya que solamente se calcula la iluminación una vez por polígono. Podría usarse para bocetos en baja calidad. Da resultados bastante buenos para figuras con caras planas (cubos, poliedros,...).

Desventajas: Solo se puede poner un color por cada cara así que siempre tendrá un aspecto cuadriculado. Si el objeto es una aproximación poliédrica de un objeto curvo, se visualizan las aristas entre polígonos. No pueden representarse brillos interiores a los polígonos, pues todos sus puntos tienen intensidad constante. Para obtener una buena representación de objetos aproximados, son necesarios muchos polígonos de pequeños tamaño [7].

1.6.2 Sombreado de Gouraud

Es un método usado en los gráficos por computadora para simular los diferentes efectos de luz y color en las superficies de los objetos. En la práctica Gouraud se usa para lograr la iluminación lisa en las superficies de los polígonos [3].

Utiliza un esquema de interpolación bilineal de la intensidad o color a lo largo de una superficie. De esta forma, se consigue una transición suave en las aristas de los polígonos, y por tanto se eliminan las discontinuidades de la intensidad que se pueden apreciar en el sombreado constante. Los resultados son bastante realistas y suficientes para muchas aplicaciones que requieren gran velocidad de procesado, como sería el caso de un motor 3D de visualización en tiempo real de algunos juegos de la actualidad [3].

La idea consiste en calcular el vector normal de cada vértice como el promedio o suma vectorial normalizada de los vectores normales de los polígonos que contienen a dicho vértice [9], posteriormente se aplica el modelo de iluminación únicamente, en los vértices del polígono para obtener la intensidad de la luz en esos puntos, asignándole el color resultante. Algunos sistemas, a la hora de calcular la luz en los vértices, eliminan la *componente especular* del modelo de intensidad, debido a que, con Gouraud, la forma e intensidad de los brillos especulares dependen mucho de las características geométricas de la red poligonal, puesto que genera resultados imprevistos, por otro lado, para que la intensidad calculada dependa exclusivamente de la normal en los vértices, se suele suponer que las fuentes y el observador se encuentran en el infinito [5].

Para conseguir la intensidad o color en las aristas, se interpolan linealmente las intensidades de las componentes de color a lo largo de las aristas de cada polígono, asignando a cada punto el color resultante de la interpolación lineal de los dos vértices que generan la arista a la que pertenece [5].

Por último, tomando las intensidades en las aristas como valores iniciales, y siguiendo las líneas scan, se calculan las intensidades de los puntos interiores de los polígonos, también mediante

interpolación lineal. En la siguiente figura se puede observar una esfera utilizando este modelo de sombreado [5]. Véase la figura 1.13

Figura 1.13: Modelo de Sombreado de Gouraud.

Es muy importante destacar que todas estas operaciones que se realizan durante la conversión scan del polígono, pueden efectuarse con técnicas aritméticas que posibilitan su implementación por hardware. Tal es así que en la actualidad las PC poseen tarjetas gráficas que reciben directamente los puntos e iluminaciones de los polígonos, y efectúan la conversión scan y sombreado de Gouraud a miles de polígonos por segundo [5].

Ventajas: Este sombreado es eficiente comparado con otros métodos, razón por la que es extensamente utilizado por los juegos de vídeo y otras aplicaciones en las que el desempeño es más importante que el resultado final.

Es rápido, disminuye la visión de las aristas y con esto mejora la visualización de las aproximaciones poliédricas de objetos curvos, soluciona los problemas de facetado y disminuye el cálculo drásticamente. Es un método provisto por las tarjetas gráficas [9].

Desventajas: No consigue eliminar totalmente las aristas (efecto de Mach), no representa aceptablemente los brillos especulares, por lo que se suele utilizar sólo para reflexión difusa, no hay interpolación en 3D lo que conduce a perturbaciones en animación y posibles errores en superficies corrugadas [3].

1.6.3 Sombreado de Phong

Consiste en que además de que el objeto tenga sombra (como el de Gouraud) proyecta su sombra sobre los demás objetos de la escena.

Es un método de interpolación en gráficos 3D (que no debe confundirse con el modelo de iluminación de Phong), se utiliza usando la interpolación de las normales a la superficie de cada polígono, el cual aumenta la calidad de sombreado de la imagen suavizando las aristas de los polígonos [10].

Este algoritmo de sombreado es una mejora del de Gouraud. Al igual que en el anterior, se calcula el vector normal de cada vértice, posteriormente interpola el vector normal a la superficie sobre los puntos pertenecientes a las aristas de cada polígono, lo cual se consigue interpolando linealmente las componentes *x, y, z* a partir de las normales de los vértices, luego interpola el vector normal para cada punto a lo largo de las líneas de barrido, utilizando las normales de los extremos de dichas líneas, ubicadas sobre las aristas, simultáneamente al paso anterior, para cada punto en las líneas de barrido se aplica el modelo de iluminación [5]. Véase la figura 1.14

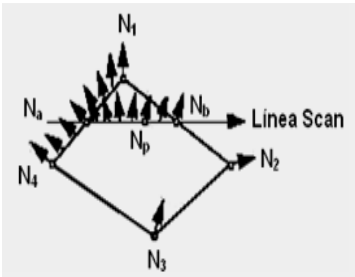


Figura 1.14: Interpolación de normales.

Ventajas: Los resultados son muy realistas, las bandas de Mach se reducen hasta el punto que resultan inapreciables y los objetos presentan brillos especulares [6]. Además, se obtienen otros beneficios importantes derivados del hecho de aplicar el modelo de iluminación en todos los puntos de la superficie, como son: la proyección de sombras y el aumento de la calidad de sombreado de la imagen suavizando las aristas de los polígonos, aplicación de texturas de

superficies, visualización de los conos de luz producidos por luces dirigidas cónicas [3]. Véase la figura 1.15

Figura 1.15: Modelo de Sombreado de Phong.

Desventajas: El método de interpolación de Phong es más lento que el de Gouraud, posee un gran coste computacional (el proceso necesita mucho más cálculos), se interpolan tres componentes en vez de una, para cada punto es necesario aplicar el modelo de iluminación, lo cual puede llegar a ser un problema en algunas aplicaciones gráficas, p.e. en la producción de secuencias animadas. Este sistema se suele utilizar al efectuar el trazado de rayos de objetos descritos por primitivas poligonales, o cuando se desea aplicar un modelo de sombreado local que no requiere tiempo real.

1.6.4 Comparación entre el Sombreado de Gouraud y el Sombreado de Phong

- El método de Phong genera una iluminación de las superficies más real que el de Gouraud. Las siluetas poligonales se reducen mucho con respecto a Gouraud, aunque no llegan a desaparecer [8] .

- El efecto de Mach, en general, es más pequeño que en Gouraud. Sin embargo, se ha demostrado que en algunos casos, como por ejemplo en las esferas, Phong produce peores efectos de Mach que el método de Gouraud [8].

- En el método de Phong, al aplicar el modelo de intensidad a cada píxel del polígono, no da problemas al trabajar con la componente especular. En otras palabras, este método de

interpolación depende mucho menos de las características geométricas de la malla poligonal que el de Gouraud [8].

- Por contra, el método de Phong requiere muchos más cálculos que el de Gouraud, ya que además de interpolar vectores, ha de calcular la ecuación de intensidad en cada píxel; con Gouraud solo se aplica el modelo de intensidad en los vértices, y además interpola valores escalares (intensidades).
- Los resultados que se obtiene con Phong son más exactos que con Gouraud.

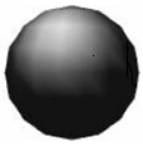
Sombreados Poligonales	**Flat**	**Gouraud**	**Phong**
Resultados			

Tabla 1.1: Sombreados Flat, Gouraud y Phong para una esfera de 990 polígonos y 497 vértices.

1.7 Fundamentación de las Tecnologías

1.7.1 RenderMonkey

Los shaders en tiempo real están en el corazón de todos los nuevos efectos visuales y continuarán siendo la fundación de una experiencia gráfica asombrosa para el futuro. Con la introducción de los lenguajes de sombreado de alto nivel de Directx y OpenGL, la complejidad de los shaders en tiempo real ha aumentado en gran medida, la rápida evolución de la capacidad del hardware gráfico ha causado una explosión en la cantidad de contenido del shader necesaria para los proyectos en tiempo real.

RenderMonkey es una herramienta creada con el objetivo de implementar shaders en un ambiente flexible coherente y de gran alcance para los programadores. Permite a cualquier persona interesada en crear shaders sumergirse en este proceso debido a su fácil manipulación.

Posibilita eliminar errores de shader con salida interactiva de los resultados intermedios computados por el shader. Esto puede ser particularmente útil en ambientes de enseñanza donde es fundamental visualizar un resultado intermedio a entender, ha sido elegido por varias universidades para enseñar tecnología del shader en su plan de estudio de gráficos por su elevado nivel de soporte de los modelos de shaders previstos con Directx 9.0 (HLSL) y para OpenGL (GLSL).

Contiene un redactor completamente equipado del shader. La ventana integrada del recopilador y de la inspección previa del shader proporciona la regeneración visual inmediata para el efecto, bajo desarrollo asegurándose de que los errores en la programación están identificados tempranamente en el proceso de desarrollo. Incorpora un número de características para ayudar a eliminar errores y a la optimización del shader.

1.7.2 Librería Gráfica OpenGL y Lenguajes de Programación de Shaders

OpenGL es la biblioteca gráfica 3D por excelencia, puede utilizarse en Linux, Unix, Mac OS, Windows e incluso en móviles. Desarrollada originalmente por Silicon Graphics Incorporated (SGI), es una biblioteca de gráficos que ofrece al programador una API que ha crecido a la par del hardware. Al igual que DirectX que tiene High Level Shader Lenguaje (HLSL), OpenGL tiene el GL Shading Language (GLSL), ambos son utilizados para programar sobre el Graphics Processing Unit (GPU) [19].

Una GPU es un procesador dedicado exclusivamente al procesamiento de gráficos, para aligerar la carga de trabajo del procesador central en aplicaciones como los videojuegos. Implementa ciertas operaciones gráficas llamadas primitivas optimizadas para el procesamiento gráfico. Una de las primitivas más comunes para el procesamiento gráfico en 3D es el antialiasing, que

suaviza los bordes de las figuras para darles un aspecto más realista. Actualmente disponen de gran cantidad de primitivas, buscando mayor realismo en los efectos [22].

Además cuenta con un enorme prestigio entre la comunidad de jugadores y expertos en informática por su excelente rendimiento 3D y la extraordinaria calidad de reproducción de video High Definition (HD) que ofrece. Esas mismas ventajas han convertido a las GPU NVIDIA en la opción preferida de tantos jugadores y programadores gráficos.

HLSL no es más que una capa adicional que reside conceptualmente sobre la API existente. Los shaders facilitan realmente la tarea de programación gráfica, ofrecen un aumento en la velocidad de proceso gráfico o flexibilidad en la programación. El lenguaje de programación HLSL de Microsoft para la GPU en DirectX 9.0 trabaja solamente en Windows y consigue altas marcas de calidad [14].

HLSL es el lenguaje de programación de alto nivel, que aunque puede crear animaciones ultra realistas, hasta efectos visuales sorprendentes, sin tener que preocuparse por el tipo específico de hardware, está basado en DirectX 3D y esta librería a pesar de ser muy popular por lo programadores gráficos no es multiplataforma [14].

Por el contrario, la técnica adoptada en OpenGL consiste en incrustar el compilador para el lenguaje de shaders GLSL en el mismo driver OpenGL. Esta técnica facilita a los desarrolladores de aplicaciones que van hacer uso de lenguajes de alto nivel al estilo C la implementación de los shaders directamente en el subsistema gráfico [21].

Por supuesto y al contrario que en los shaders HLSL que todo funciona con arquitectura PC, un shader OpenGL será compatible con todas las plataformas y sistemas que sean compatibles con OpenGL [21]. Permite redefinir el comportamiento por defecto del procesador de vértices (vertex processor) y de fragmentos (fragment processor) presente en los sistemas OpenGL modernos. Aunque esta librería no posee tanta popularidad como DirectX, ha sido la preferida de

programadores gráficos como Carmack, con la que ha desarrollado los motores del Quake y del Doom [19].

1.7.3 Lenguaje de Programación Visual C++. Net

Proporciona a los programadores un lenguaje orientado a objetos de probada eficacia para generar aplicaciones de alto rendimiento. Gracias a plantillas avanzadas, acceso a plataformas de bajo nivel y un compilador que optimiza las compilaciones, ofrece funcionalidad para generar componentes sólidos, utilizando el lenguaje de programación C++. Aporta un nivel de productividad muy superior a C++, sin comprometer la flexibilidad, el rendimiento o el control.

Es el lenguaje de sistemas más conocido del mundo, altamente compatible, soporta las librerías gráficas Glide, OpenGL, DirectX y G3D que son las utilizadas por excelencia en la industria de los video-juegos, debido a las ventajas que este posee, hace uso del paradigma de la programación orientada a objetos [15].

1.7.4 Biblioteca Gráfica Multiplataforma G3D

Librería open source, programada en C++, útil para la creación de video-juegos, es la base de un código robusto y de alto nivel común a la mayoría de proyectos 3D. Es un proyecto que intenta disminuir la complejidad matemática, ayudando en la medida de lo posible al programador. Muestra una capa sobre OpenGL que simplifica la complejidad en la escritura del código. No es un motor de juegos en sí mismo, sino un paquete de piezas necesarias para crear un motor 3D o proyecto de gráficos [16].

La librería tiene dos partes: G3D para las matemáticas y GLG3D para interactuar con la aceleración de hardware por OpenGL. Está orientado a usuarios familiarizados con C++, DirectX u OpenGL. Se utiliza en juegos, versiones parciales, papeles de investigación, simuladores militares, y cursos de la universidad. Proporciona un sistema de rutinas y del campo común de las estructuras, están necesitados en casi todos los programas de gráficos. Permite fácil

migración hacia cualquier sistema operativo y una libre distribución. Es de fácil maniobrabilidad, y enriquecido con colores atractivos [16].

Recomienda para el empleo de shaders su implementación en lenguaje GLSL, debido a que posee clases que apoyan su maniobrabilidad [16].

1.8 Consideraciones Finales

- En el desarrollo del trabajo se ha descrito el fenómeno básico de iluminación.
- Han sido caracterizados algunos modelos de iluminación local/global así como distintos tipos de sombreados poligonales.
- Se analizaron las ventajas y desventajas de cada uno por lo que se propone para etapa de pre-procesamiento el método de rendering Radiosidad, con el sombreado de polígonos Phong Shading y para etapa de tiempo real el modelo local de iluminación Phong con el sombreado Gouraud Shading.
- Se dio cumplimiento a la fundamentación de las diferentes tecnologías ha utilizar considerándose como lenguaje de programación de shaders GLSL, por su elevada compatibilidad con la libraría gráfica OpenGL, como herramienta para la creación de shaders se seleccionó RenderMonkey, y para futura continuación del trabajo se propone a C++ como lenguaje de programación para visualizar los shaders en combinación con la biblioteca G3D en el entorno de Visual Studio .NET 2003.

Capítulo
2

2 SOLUCIÓN PROPUESTA Y ANÁLISIS FINANCIERO.

2.1 Introducción

Serán introducidos un conjunto de términos técnicos que faciliten la comprensión del proceso de iluminación, donde se brinda la oportunidad de introducir los shaders para sustituir etapas de procesamiento de primitivas. Una vez que han sido propuestos los modelos de iluminación y sombreados poligonales a utilizar para lograr una correcta simulación de luces en los video-juegos, se procederá en el capítulo, a la descripción del modelo matemático de los métodos seleccionados e implementación del modelo local y sombreado poligonal para etapa de tiempo real en el lenguaje de shaders propuesto.

Se definirá la plataforma de hardware y software. Se realizará el análisis financiero de los algoritmos programados, mediante el modelo de COCOMO II y serán mostrados algunos casos de prueba.

2.2 Plataforma Recomendada

2.2.1 Hardware

Procesador Intel P4+ 2.0Ghz, 256Mb RAM, con Video Card NVIDIA 128Mg, Quadro FX 500, AGP 8X recomendada.

2.2.2 Software

Estos algoritmos deben correr sobre sistemas operativos Windows XP, Windows 2003, versiones superiores de Windows, Linux o Mac OS. Serán implementados sobre tecnología de shaders.

2.3 Pipeline Gráfica Programable

El proceso computacional que lleva desde la descripción de los objetos en la escena hasta la imagen final, se produce mediante una serie de fases consecutivas y conectadas entre sí de forma tal que la salida de datos de una fase constituye la entrada de la siguiente, dicho sistema se denomina pipeline gráfica.

El contexto gráfico del pipeline incluye variables cuyo valor puede ser cambiado por el programa a lo largo del proceso. Entre ellas hay opciones que controlan la activación o desactivación de ciertas funciones, o controlan el modo en que algunas operaciones son realizadas (por ejemplo, variantes en el método de iluminación) y datos que son utilizados por la pipeline, como las matrices de proyección y transformación, posición del observador, color actual, material actual, fuentes de luz, texturas, entre otros.

Se recuerda que uno de los objetivos de los sistemas de visualización en tiempo real es descomponer el proceso de visualización en fases simples que tengan un coste mínimo y puedan ser implementadas por hardware. Según cuál sea el equipo disponible, parte de la pipeline será ejecutada por el hardware específico y parte tendrá que ejecutarse por la CPU del sistema. Esta idea permite que una misma aplicación pueda funcionar en equipos con diferente tipo de aceleración gráfica.

En cualquier caso, la parte ejecutada en la CPU comenzará siempre por el principio de la pipeline, y a partir de cierto punto los datos serán transferidos al hardware gráfico que continuará el proceso hasta el final. La idea fundamental del procesado en tiempo real es que todos los objetos deben ser descompuestos en polígonos. Estos polígonos serán descompuestos a su vez en triángulos.

Cada triángulo será proyectado sobre la ventana bidimensional y rellenado con los colores adecuados para reflejar los efectos de la iluminación, texturas, entre otros. Una vez que se han generado los triángulos, en la pipeline existen dos partes claramente diferenciadas: una primera

etapa operaciones realizadas sobre cada uno de los vértices, y después de que estos se proyecten sobre la ventana, entonces comienza una segunda fase de cálculos realizados para cada píxel cubierto por los triángulos.

El siguiente esquema sería el proceso del pipeline que comienza por la base de datos de la escena [17]. Véase la figura 2.1.

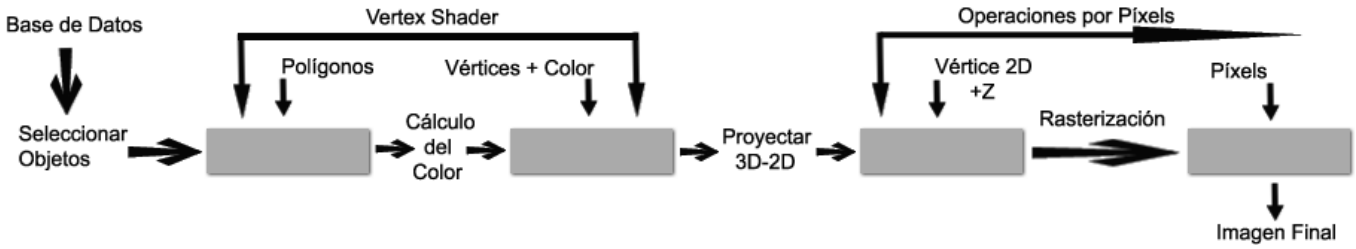


Figura 2.1: Pipeline gráfica.

Las fases principales son:

- Seleccionar los objetos que deben dibujarse en cada fotograma.

1- Convertir esos objetos a primitivas poligonales.

2- Producir a partir de ellas listas de vértices.

- Calcular el color para cada vértice según un método de sombreado local.
- Proyección 3D-2D de cada vértice, que además nos proporciona su profundidad.

1- Recorte del triángulo si alguna parte de él cae fuera de la ventana.

- Rellenado de los triángulos (rasterización). Para ello se realizan varias fases de procesamiento por píxel.

1- Comprobar si cada nuevo píxel es visible o no (comprobación de profundidad).

2- Interpolación lineal del color para el nuevo píxel (Sombreado de Gouraud).

3- Si existe una textura definida o transparencia, efectuar la modificación de color correspondiente [17].

2.4 Pre-Procesamiento

La capacidad de procesamiento del hardware gráfico ha tenido un desarrollo creciente, y aunque son difíciles de alcanzar resultados cien por ciento óptimos, el trabajo presenta una solución inicial que propone el desarrollo de un modelo de iluminación global en *pre-procesamiento,* para lograr mayor fidelidad en la representación de las escenas, a través de la Radiosidad, y así el costo computacional que presenta este tipo de algoritmo no sería un problema ya que el cálculo de intensidad de luces estáticas se realizaría solo una vez, gracias a la cualidad de este método que plantea, que de no existir cambios en la posición geométrica de los objetos o de las texturas no hay necesidad de recalcular los valores de intensidad.

Se supondrá una escena formada por objetos cuya superficie ha sido descompuesta en patches y se quiere calcular la Radiosidad que emite cada uno de esos patches como resultado del intercambio de energía radiante que se ha dado entre ellos hasta llegar a un estado de equilibrio [12]. Véase la figura 2.2.

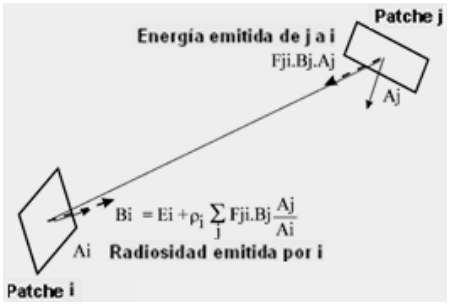


Figura 2.2: Escena descompuesta en patches.

Una vez alcanzado el equilibrio, la energía que emite un patche *i* se debe, por un lado, a la que emite por sí mismo E_i y, por otra parte, también refleja una cierta proporción *(dada por el coeficiente de reflectancia difuso)* de la que llega hasta él, desde otros patches de la escena B_{pe}:

$$B_i = E_i + \rho_i B_{pe}$$

La energía que proviene de otros patches será una suma, cada uno de cuyos términos será proporcional a la Radiosidad que emite el patche *j.* Es decir, una fracción, que se llamará F_{ij}, de la energía emitida por un patche *j* que llegará al patche *i*:

$$B_{pe} = \sum_{j=1}^{n} B_{viene\_de\_j} = \sum_{j=1}^{n} F_{ij} B_j$$

Combinando esta ecuación con la anterior resulta la ecuación global de la Radiosidad para un patche *i* [12]:

$$B_i = E_i + \rho_i \cdot \sum_{j=1}^{n} F_{ij} B_j$$

donde E_i es la energía debida a su propia emisión para cada patche *i (emisividad)*, ρ_i es el coeficiente de reflexión difuso para un patche *i*, B_j es la Radiosidad para la superficie *j*, F_{ij} es el factor de forma o proporción de la energía emitida por un patche *j* que llega a un patche *i*, y *n* es el número de polígonos en que se discretiza la escena [12].

Hay que hacer notar que en el caso de superficies planas o convexas los patches no se pueden ver por lo que los términos F_{ij} se hacen cero mientras que para superficies cóncavas esto no se da. Como ejemplo, para tres patches se podría calcular la Radiosidad como:

$$B_1 = E_1 + \rho_1 \left(B_1 F_{11} + B_2 F_{12} B_3 F_{13} \right)$$

Reordenando términos:

$$B_1(1-\rho_1 F_{11})+B_2(-\rho_2 F_{12})+B_3(-\rho_3 F_{13})=E_1$$

Estas ecuaciones se pueden resolver usando métodos numéricos iterativos de inversión de matrices, ya que el número de ecuaciones e incógnitas es tan elevado que la solución directa no es posible, lo que da una idea de cómo plantear la ecuación en su forma matricial [20].

$$\begin{bmatrix} 1-\rho_1 F_{11} & -\rho_1 F_{12} & \cdots & -\rho_1 F_{1n} \\ -\rho_z F_{z1} & 1-\rho_z F_{zz} & \cdots & -\rho_z F_{zn} \\ \vdots & \vdots & & \vdots \\ -\rho_n F_{n1} & -\rho_n F_{n2} & \cdots & -\rho_n F_{nn} \end{bmatrix} \begin{bmatrix} B_1 \\ B_z \\ \vdots \\ B_n \end{bmatrix} = \begin{bmatrix} E_1 \\ E_z \\ \vdots \\ E_n \end{bmatrix}$$

Esta ecuación dará los términos de Radiosidad que se pueden convertir a valores de intensidad de luz al dividir entre π, ya que $B = I\pi$ [20].

Como en el trabajo el propósito esencial es lograr la futura creación de escenas a color dotadas de realismo y sensación de continuidad visual, habría que calcular la Radiosidad para las componentes individuales de luz, rojo, verde y azul *(RGB)* ya que los coeficientes E_i y ρ_i son dependientes de la longitud de onda, los factores de forma son función únicamente de la geometría.

A la vista de la ecuación matricial anterior, para poder calcular la Radiosidad se deberá calcular los factores de forma F_{ij}.

Una vía inmediata de computar los factores de forma en aras de lograr mayor velocidad en el cálculo, es la aplicación del método estocástico de Montecarlos para ofrecer un mejoramiento en la eficiencia del método de iluminación global propuesto.

Estimación del número de líneas que cruzan un patche. Véase la figura 2.3.

$$E(N_i)=N\frac{2A_i}{A_t}$$

donde *N* es el número de líneas que cruza un patche.

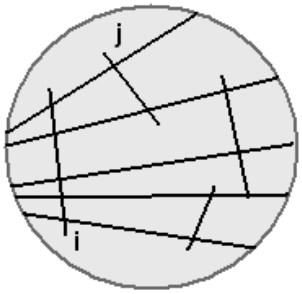


Figura 2.3 Superficie dividida en patches.

Computación del factor de forma:

$$F_{ji} = \frac{N_{ji}}{N_i}$$

Ahora se le puede dar solución a la matriz de radidosidad [12].

Representación gráfica de cómo se vería una escena aplicando la propuesta del trabajo. Véase la figura 2.4.

Figura 2.4: Resultado de la Radiosidad en una escena.

Una vez calculados los valores de Radiosidad para cada una de las superficies se utilizará el algoritmo de sombreado Phong, el cual se desarrollará de la siguiente forma:

1. Calcular el vector normal de cada vértice como el promedio de los vectores normales de los polígonos que contienen a dicho vértice [7]. Véase la figura 2.5.

$$N_v = \frac{\sum_{k=1}^{n} N_k}{\left|\sum_{k=1}^{n} N_k\right|}$$

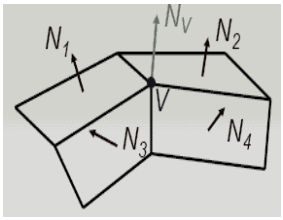


Figura 2.5: Normal de los polígonos adyacentes.

2. Interpolar el vector normal a la superficie sobre los puntos pertenecientes a las aristas de cada polígono [8]. Véase la figura 2.6.

$$N_a = N_1 \frac{y_{s,n} - y_2}{y_1 - y_2} + N_2 \frac{y_1 - y_{s,n}}{y_1 - y_2},\ N_b = N_1 \frac{y_{s,n} - y_3}{y_1 - y_3} + N_3 \frac{y_1 - y_{s,n}}{y_1 - y_3}$$

3. Interpolar el vector normal para cada punto a lo largo de las líneas de barrido [8]. Véase la figura 2.6.

$$N_{s,n} = N_a \frac{x_b - x_{s,n}}{x_b - x_a} + N_b \frac{x_{s,n} - x_a}{x_b - x_a}$$

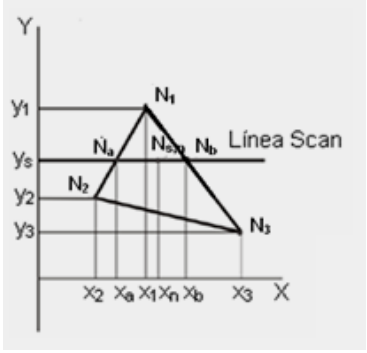


Figura 2.6: Interpolación lineal de normales.

De esta forma los polígonos quedarán dotados de luz y color [8].

2.5 Tiempo Real

Una vez que en pre-procesamiento han sido aplicadas las luces, dotando las escenas de realismo se procede al cálculo de la iluminación en tiempo real, en esta etapa se hará uso de un modelo de iluminación menos potente ya que lograr un costo de cómputo aceptable se convierte en un factor de primera necesidad, aunque se haya propuesto un lenguaje de shader para su implementación que trabaje directamente sobre la tarjeta gráfica de la máquina.

Como propuesta de modelo de iluminación local o ecuación de iluminación se propone a Phong con el objetivo de hallar los valores de intensidad.

Combinándose los cálculos de la componente especular con los cálculos de la componente difusa y ambiental para obtener una expresión lineal de la ecuación de iluminación de Phong, se obtiene:

$$I_\lambda = I_{a\lambda} K_a O_{d\lambda} + f_{aat} I_{p\lambda} [K_d O_{d\lambda} (N \cdot L) + K_s O_{s\lambda} (R \cdot V)^n]$$

Para optimizar el cálculo de la componente especular se sustituye el producto escalar *(R·V)* por *(N·H)*, puesto que el vector *H* se encuentra entre los vectores *R* y *V*, lo que disminuye el tiempo de cálculo de este modelo.

donde $H = \frac{L+V}{|L+V|}$

El modelo de iluminación de Phong con la incorporación del producto escalar *N·H*:

$$I_\lambda = I_{a\lambda} K_a O_{d\lambda} + f_{aat} I_{p\lambda} [K_d O_{d\lambda} (N \cdot L) + K_s O_{s\lambda} (N \cdot H)^n]$$

Se pueden observar mejorías en la escena a medida que se han incorporado componentes al modelo de iluminación de Phong [7]. Véase la figura 2.7.

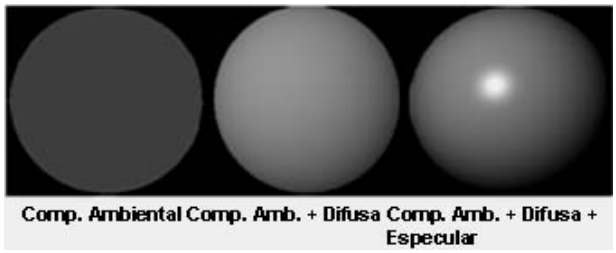


Figura 2.7: Aplicación de las componentes de luz.

Una vez que se han calculado los valores de intensidad a través de Phong se procede a calcular el sombreado de Gouraud.

Pasos para el cálculo del sombreado:

1. Calcular el vector normal de cada vértice como el promedio de los vectores normales de los polígonos que contienen a dicho vértice [7]. Véase la figura 2.8.

$$N_v = \frac{\sum_{k=1}^{n} N_k}{\sum_{k=1}^{n} |N_k|}$$

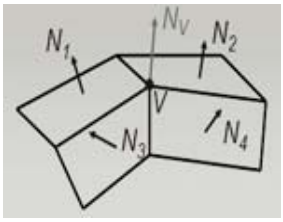


Figura 2.8: Normal de los polígonos adyacentes.

2. Aplicar el modelo de iluminación a cada vértice, asignándole el color resultante [8].

3. Interpolar linealmente las intensidades de las componentes de color a lo largo de las aristas de cada polígono, asignando a cada punto el color resultante de la interpolación lineal de los dos vértices que generan la arista a la que pertenece [8]. Véase la figura 2.9.

$$I_a = I_1 \frac{y_{s,n} - y_2}{y_1 - y_2} + I_2 \frac{y_1 - y_{s,n}}{y_1 - y_2} \qquad I_b = I_1 \frac{y_{s,n} - y_3}{y_1 - y_3} + I_3 \frac{y_1 - y_{s,n}}{y_1 - y_3}$$

4. Calcular las intensidades de los puntos interiores, a partir de los valores calculados en las aristas. La interpolación se efectúa siguiendo la línea scan, y por tanto, las intensidades utilizadas en la interpolación son las que se hayan en los puntos de intersección de las aristas con la línea scan (I_a e I_b). Véase la figura 2.9.

La ecuación de interpolación para calcular la intensidad en los puntos interiores es similar a las anteriores. La intensidad $I_{s,n}$ en un punto de la superficie viene dada por [8]: Véase la figura 2.9.

$$I_{s,n} = I_a \frac{x_b - x_{s,n}}{x_b - x_a} + I_b \frac{x_{s,n} - x_a}{x_b - x_a}$$

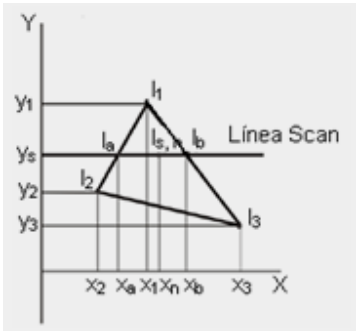


Figura 2.9: Interpolación de intensidades.

Representación en código shader del cálculo de la iluminación para tiempo real:

Vertex Program

```
/**

@file VertexShader.Vert

@brief Real-time surface shading

@This code calculates the illumination using Phong’s reflection equation

for vertexes.

@author   Yoander Cabrera Díaz, ycabrerad@estudiantes.uci.cu

@date 2007/05/26

Copyright(C) Virtual Games Group (VGG), support.vgg@gmail.com

University of Informatics Sciences

Havana, Cuba

*/
```

```
varying vec4 ambientIntensity; // ambient intensity

varying vec4 diffuseIntensity; // diffuse intensity

varying vec4 specularIntensity;// specular intensity

varying vec2 Texcoord;         // texture coordinates

varying float lightIntensityP; // light intensity

varying float fatten;          // attenuation factor

void main()

{

// receive texture coordinates

Texcoord = gl_MultiTexCoord0.xy;

// surface normal direction

vec3 normal = gl_NormalMatrix * gl_Normal;

// polygons vertex position

vec3 vertexPosition = gl_ModelViewMatrix * gl_Vertex;

// light source direction

vec3 lightDirection = gl_LightSource[0].position.xyz –

vertexPosition.xyz;

// viewer direction
```

```
vec3 viewDirection =  - vertexPosition.xyz;

// normalize vector N

vec3 normalN = normalize ( normal );

// normalize vector L

vec3 lightDirectionL = normalize( lightDirection );

// normalize vector V

vec3 viewDirectionV = normalize( viewDirection );

// ambient intensity calculation

ambientIntensity = gl_LightSource[0].ambient * gl_FrontMaterial.ambient;

// light intensity calculation

lightIntensityP  = dot(lightDirectionL, normalN);

// distance between light source and surface

float dist = length (lightDirectionL);

// attenuation factor calculation

fatten = min [(1.0 / gl_LightSource[0].constantAttenuation +
gl_LightSource[0].linearAttenuation * dist +
gl_LightSource[0].quadraticAttenuation * pow(dist,2)),1.0];

// scalar product between N and L

float nDotl = max(0.0,dot( normalN, lightDirectionL ));
```

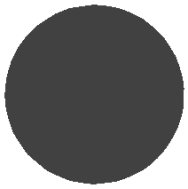
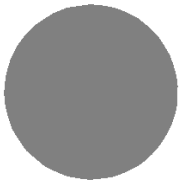
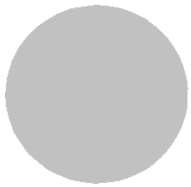
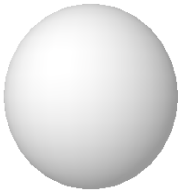
```
// diffuse intensity calculation

diffuseIntensity = gl_LightSource[0].diffuse * gl_FrontMaterial.diffuse
* nDotl;

// vector H calculation

vec3 vectorH = normalize(( lightDirectionL + viewDirectionV ) / abs(
lightDirectionL + viewDirectionV ));

// scalar product between N and H

float nDoth = max(0.0,dot( normalN, vectorH ));

// specular intensity calculation

specularIntensity = gl_LightSource[0].specular *
gl_FrontMaterial.specular * pow( nDoth, gl_FrontMaterial.shininess);

// vertex transformation especification

gl_Position = ftransform();
```

Fragment Program

```
/**

@file Iluminacion.Frag

@brief Real-time surface shading

@This code calculates the illumination utilizing Phong’s reflection

equation.
```

```
@author   Yirka Cespedes Boch, yecespedes@estudiantes.uci.cu

@date 2007/06/02

Copyright(C) Virtual Games Group (VGG), support.vgg@gmail.com

University of Informatics Sciences

Havana, Cuba

*/

uniform sampler2D texture;      // variable used to access a 2D texture

varying vec2 Texcoord;          // texture coordinates

varying float lightIntensityP; // light intensity

varying vec4 ambientIntensity;  // ambient intensity

varying vec4 diffuseIntensity;  // diffuse intensity

varying vec4 specularIntensity; // specular intensity

varying float fatten;           // attenuation factor

void main( void )

{

vec4 myTexture = texture2D (texture, Texcoord);

//specifies the fragment color which will be used for the

//subsequent fixed functionality pipeline
```

```
gl_FragColor = (ambientIntensity * myTexture) + fatten * lightIntensityP
* ((diffuseIntensity * myTexture) + specularIntensity);

}
```

Parámetros	Resultado 1	Resultado 2	Resultado 3
Coeficiente de reflexión ambiental.	 $k_a = 0.25$	 $k_a = 0.50$	 $k_a = 0.75$
Observaciones	Con la aplicación del k_a sobre los materiales no se observa tridimensionalidad en los objetos, debido a que la cantidad de luz incidente para cada objeto es constante para todas las superficies y en todas las direcciones.		
Resultado anterior, más el coeficiente de reflexión difusa.	 $K_d = 0.40$	 $K_d = 0.40$	 $K_d = 0.40$
Observaciones	Con la aplicación de un k_d constante sobre los resultados obtenidos se puede observar mejorías en el efecto de tridimensionalidad de los objetos; dado que		

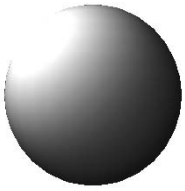
	esta componente simula el resultado de aplicar una fuente de iluminación puntual de cierta intensidad, que es reflejada de manera uniforme (difusa) por la superficie del objeto.		
Resultado anterior, más coeficiente de reflexión especular y exponente de reflexión especular.	 $K_s = 1.0$ $n = 3$	 $K_s = 1.0$ $n = 10$	 $K_s = 1.0$ $n = 100$
Observaciones	Con la aplicación de un k_s constante y n sobre los resultados obtenidos, se puede observar mayor calidad en la visualización y tridimensionalidad de los objetos, para reflectores perfectos a medida que los valores del exponente de reflexión especular aumentan la luz se ve más puntual, aumentando así el número de cálculos.		

Tabla 2.1: Casos de prueba.

2.6 Importancia del Análisis Financiero

El análisis del costo de un proyecto es imprescindible a la hora de acometer una tarea, es la forma que se tiene de saber si la realización del mismo es factible o no, por lo que es necesario estimar el esfuerzo humano, el tiempo de desarrollo que se requiere para la ejecución del mismo y también su costo. Estas estimaciones pueden realizarse a través del método de puntos de función del modelo de COCOMO II.

Antes de pasar al desarrollo del análisis financiero se cree oportuno hacer referencia a una cita:

"Para llevar a cabo un buen proyecto de desarrollo de software, se debe comprender el ámbito del trabajo a realizar, los recursos requeridos, las tareas a ejecutar, las referencias a tener en cuenta, el esfuerzo a emplear y la agenda a seguir" [18].

2.7 Planificación

Partiendo del análisis de la alta complejidad de los algoritmos a implementar, principalmente asociados a la representación y cálculo del balance de las líneas de producción y considerando lo planteado por Pressman, es necesario hacer los cálculos de estimación correspondientes utilizando una extensión de la métrica de los Puntos de Función que sea adaptada perfectamente a este tipo de problema: el método de Puntos de Característica.

En la estimación del trabajo se consideró un modo de desarrollo semilibre donde se tuvo en cuenta el cálculo mediante los puntos de función desajustados.

Salidas Externas

Nombre de la Salida Externa	**Cantidad de Ficheros**	**Cantidad de Elementos de Datos**	**Clasificación(Simple, Media y Compleja)**
Aplicación de luz en tiempo real	2	1	S

Tabla 2.2: Salidas externas.

Archivos lógicos internos

Nombre del Fichero Interno	Cantidad de Records	Cantidad de Elementos de Datos	Clasificación(Simple, Media y Compleja)
Vertex Program	1	1	S
Fragment Program	1	1	S

Tabla 2.3: Archivos lógicos internos.

Puntos de función desajustados

Elementos	Simples	X Peso	Medios	X Peso	Complejos	X Peso	Subtotal de Puntos de Función
Salidas externas	1	4	0	5	0	7	4
Archivos lógicos internos	2	3	0	4	0	6	6
Total	3						10

Tabla 2.4: Puntos de función desajustados.

Cantidad de instrucciones fuentes

Cocomo II, plantea que su cálculo se basa en la cantidad de instrucciones fuentes por punto de función desajustado que genera el lenguaje de programación empleado. Estos datos se conocen

a partir de estudios estadísticos realizados a cada lenguaje, el índice de instrucciones fuentes por cada punto de función de GLSL es de # 29 (se seleccionó el índice de instrucciones fuentes por cada punto de función de C++, porque la sintaxis de ambos lenguajes es muy semejante.)

Como es observable el cálculo de la cantidad de instrucciones fuentes del algoritmo representa el 100%, debido a que se empleó un solo lenguaje de programación.

Características	**Valor**
Puntos de función desajustados	10
Lenguaje	GLSL
Instrucciones fuentes por puntos de función	50
Instrucciones fuentes	500

Tabla 2.5: Instrucciones fuentes.

Total de instrucciones fuentes (miles) MF: 0.500 MF

2.7.1 Cálculo del esfuerzo, tiempo de desarrollo, cantidad de hombres y costo

Multiplicadores de esfuerzo: Modelo diseño temprano

Calculo de:	Significado	Valor	Justificación
RCPX	Fiabilidad del Producto y Complejidad	1	La información suministrada para el desarrollo de los shaders debe ser confiable. La complejidad de los algoritmos está determinada por el modelo matemático. Para el desarrollo del algoritmo se requiere de documentación asociada al ámbito del problema.
RUSE	Reutilización Requerida	1	Se trata de mantener la reusabilidad a través del proyecto, previendo que el código pueda ser utilizado en el futuro con facilidad.
PDIF	Dificultad de la Plataforma	1	La plataforma es estable. Los algoritmos de iluminación tienen restricciones en cuanto a tiempo de ejecución y limitación de memoria.
PREX	Experiencia Personal	0.87	Se considera que la experiencia que poseen los desarrolladores en el lenguaje y herramienta de desarrollo es media, pues el equipo de desarrollo no tiene experiencia en algoritmos con estas características.
FCIL	Facilidades	1.10	Se deben emplear herramientas básicas.
SCED	Planificación Temporal	1	La planificación establecida por el equipo de desarrollo no debe extenderse por lo que debe cumplirse en el

			plazo establecido.
PERS	Capacidad Personal	0.83	La rotación del personal es nula.

Tabla 2.6: Multiplicadores de esfuerzo.

Factores de escala

Factores	Significado	Valor	Justificación
PREC	Precedencia	3.72	Existe una adecuada comprensión organizacional y de objetivos del proyecto, no se posee experiencia en la realización de algoritmos de este tipo.
FLEX	Flexibilidad de desarrollo	3.04	El algoritmo debe cumplir considerablemente con los requisitos preestablecidos.
TEAM	Cohesión del equipo	2.19	Existe buena comunicación y alta cooperación entre los miembros del equipo.
PMAT	Madurez del proceso	7.80	Se encuentra en el nivel 1 según el Modelo de Madurez de Capacidad (CMM).

Tabla 2.7: Factores de escala.

Multiplicadores de esfuerzo

$$EM = \prod_{i=1}^{7} Emi = RCPX * RUSE * PDIF * PREX * FCIL * SCED * PERS = 0.72$$

Factores de escala

$$SF = \sum_{i=1}^{5} SFi = PREC + FLEX + TEAM + PMAT = 17.39$$

A continuación se especifican otros cálculos necesarios para completar el análisis de los costos.

Valores calibrados

$A = 4.94; B = 0.91; C = 3.67; D = 0.24$

$E = B + 0.01 * SF$ $\qquad$ $F = D + 0.2 * (E - B)$

$E = 0.91 + 0.01 * 17.39$ $\qquad$ $F = 0.24 + 0.2 *(1.1058 - 0.91)$

$E = 1.1058$ $\qquad$ $F = 0.27916$

Esfuerzo

$PM = A * (MF)^{E} * EM$

$PM = 4.94 * (0.5)^{1.10} * 0.72$

$PM \approx 1.7$ Hombres / Mes

Cálculo del tiempo

$TDEV = C + PM^F$

$TDEV = 1.67 + (1.7)^{0.28}$

$TDEV = 3 \text{ meses}$

Cálculo de la cantidad de hombres

$CH = PM / TDEV$

$CH = 1.7/3$

$CH \approx 1$

Como el sistema es desarrollado por 2 hombres disminuye el tiempo de desarrollo.

$TDEV= 1.5 \text{ meses}$

$CH = 2$

Costo del proyecto

Para realizar este cálculo es necesario conocer el costo por hombres al mes (CHM). Se asume como salario promedio mensual (SPM) $349, que es el salario básico de un adiestrado.

$Salario\ Promedio = \$349.00$

$CHM = 2 * Salario\ Promedio$

$CHM = 2 * \$349.00$

$CHM = \$ 698$

Luego se procede a realizar el cálculo del costo del proyecto:

$Costo = CHM * PM$

```
Costo = $698 * 1.7

Costo = $ 1186.6
```

Para el desarrollo del trabajo se incurre en un costo total de $ 1186.6.

Cálculo de:	**Valor**
Esfuerzo.	0.8 hombres/mes
Tiempo de desarrollo.	1.5 meses
Cantidad de hombres necesarios.	2
Salario promedio mensual.	$349.00
Costo del proyecto.	$1186.6

Tabla 2.8: Costo del proyecto.

2.7.2 Beneficios Intangibles

Los beneficios obtenidos con el desarrollo de estos algoritmos son fundamentalmente intangibles, ya que les facilita a los futuros desarrolladores de esta rama reutilizar el código a la hora de llevar a cabo el desarrollo de video-juegos que demanden escenas con gran realismo, además que serán capaces de investigar en un menor plazo todo lo referente a esta parte, lo que implica un ahorro del tiempo que se invierte en esta tarea.

2.8 Consideraciones Finales

- Se ha definido la plataforma de hardware y software.

- Se han adquirido conocimientos acerca de términos técnicos que proporcionan la comprensión del proceso de iluminación, lo cual facilita la búsqueda del realismo visual en imágenes.

- Se ha desarrollado el modelo matemático del método de Radiosidad y sombreado de Phong a utilizar en pre-procesamiento al igual que el modelo de iluminación de Phong y sombreado de Gouraud para tiempo real.

- Se han implementado en el lenguaje de shader GLSL dichos métodos excepto Radiosidad y el sombreado de Phong, los cuales quedaron planteados matemáticamente para el desarrollo de una futura aplicación en Visual C++.Net en combinación con la biblioteca G3D.

- Se han desarrollado casos de prueba y se ha realizado el análisis financiero de los algoritmos implementados.

CONCLUSIONES

- Se ha realizado un análisis de los modelos de iluminación local/global y sombreados poligonales utilizados en los videos-juegos actuales, el resultado de esta investigación demuestra que es factible la aplicación del método global de iluminación Radiosidad en combinación con el sombreado poligonal Phong para etapa de pre-procesamiento en los juegos y aplicaciones 3D del perfil de EV de la F5. Para tiempo real el modelo de iluminación Phong y el sombreado poligonal Gouraud muestran superioridad desde el punto de vista computacional.

- De los métodos antes mencionados se realizó la modelación matemática y además se implementó en GLSL el modelo de iluminación de Phong y sombreado de Gouraud.

- Se propuso a C++ como lenguaje de programación para visualizar los shaders en combinación con la biblioteca G3D en el entorno de Visual Studio .NET 2003.

- Se expusieron los beneficios de los métodos, lo cual brindó elementos para concluir que los mismos facilitan a los futuros programadores de esta rama reutilizar el código a la hora de llevar a cabo el desarrollo de video-juegos que demanden escenas con gran realismo, además que sean capaces de investigar en un menor plazo todo lo referente al tema.

- Por tanto, se han cumplido los objetivos del trabajo dando lugar a futuras investigaciones que amplíen el perfil y las aplicaciones del mismo.

RECOMENDACIONES

Finalizado el trabajo de diploma donde se analizaron los métodos de iluminación local/global y sombreados poligonales a emplear para darle mayor sensación de realismo a los video-juegos. Se recomienda que:

- El GPJV de la F5 continúe trabajando en esta línea, específicamente en la implementación del método de rendering Radiosidad en Visual C++.Net 2003 empleando la librería gráfica G3D, así como en la implementación del sombreado poligonal Phong en el lenguaje de shaders GLSL a fin de crear una aplicación que sea capaz de renderizar imágenes fotorrealistas.

- Se haga uso de los métodos propuestos para solucionar el problema de la simulación realista de los video-juegos; introduciendo estos temas de iluminación y efectos especiales haciendo uso de los shaders en los programas de clases del 2do perfil y la asignatura de GxC.

- Se utilice este material como referencia académica para la enseñanza de dichas asignaturas.

- Se genere un hábito de uso de los shaders en los grupos de proyectos del perfil de EV.

- Se continúe investigando respecto al tema de la iluminación en la rama de los video-juegos y realidad virtual mediante futuros trabajos de diplomas, publicaciones, proyectos, entre otros.

REFERENCIAS BIBLIOGRÁFICAS

[1] Castro, W.E.V. [consultado en: noviembre 2006] "La luz, su Propagación, Distribución y Aplicaciones" (Agosto 2001). Disponible en: http://www.conicit.go.cr/recursos/documentos/la_luz.html

[2] Weizenfeld, A. [consultado en: diciembre 2006]. "Iluminación y Sombreado" (2006). Disponible en: http://cannes.itam.mx/Alfredo/Espaniol/Cursos/Grafica/Sombreado.pdf

[3] Alcaraz, J.L.A. [consultado en: diciembre 2006]. "Diseño de un Visualizador Poligonal" (Diciembre de 1996). Disponible en: http://ditec.um.es/~jlaragon/proyectoFC.pdf

[4] López, M.J.V., J.L. Crespo, and R.V. Hernando. [consultado en: noviembre 2006] "Iluminación Realista para Escenas con Vegetación". Disponible en: http://www.sig.upv.es/websig/investigacion/docs/IIluminacion.pdf

[5] Cuevas, H.F.T.y.A.R. [consultado en: noviembre 2006]. "Sistema para la Reconstrucción Gráfica Tridimensional de Objetos Físicos en Computador"(2000). Disponible en: http://www.dfmf.uned.es/~hfranco/ProyFinal.pdf

[6] Belmonte., A.M., [consultado en: diciembre 2006]. "Modelos de Iluminación". Disponible en: http://dis.um.es/grupos/sig/08BI/iluminacion.pdf

[7] [consultado en: octubre 2006]. "Visualización de Polígonos: Sombreados" (2003). Departamento de Ciencia de la Computación e Inteligencia Artificial. Universidad de Alicante. Disponible en: http://www.dccia.ua.es/dccia/inf/asignaturas/GC/Teoria/Tema207b.%20Sombreado%20de%20polígonos.pdf

[8] Ramos., R.,. [consultado en: diciembre 2006]. "Visualización Standard: obtención de la imagen discreta" (2006). Disponible en: http://di002.edv.uniovi.es/~rr/Tema10.pdf

[9] Dielrieux., C., [consultado en: noviembre 2006]. "Introducción a la Computación Gráfica". Disponible en: http://www.memi.umss.edu.bo/mscinfo/cursos/graficos/apunte.pdf

[10] [consultado en: enero 2007]. "Phong Shading". Disponible en: http://en.wikipedia.org/wiki/Phong_shading

[11] [consultado en: diciembre 2006]. "Ray-Tracing". Consultado en: http://ima.udg.es/~gonzalo/teaching/tranpas/Ray-tracing.pdf

[12] [consultado en: noviembre 2006]. "Modelos de Iluminación Global". Disponible en: http://informatica.uv.es/iiguia/AIG/web_teoria/tema4.pdf

[13] [consultado en: enero 2007]. Artículo “Radiosidad”. Disponible en: http://www.erco.com/en_index.htm?http://www.erco.com/guidev2/guide2/simulation95/radiosity_2686/es/es_radiosity_intro_1.htm

[14] Santos., J., S., [consultado en: enero 2007]. Artículo “Introducción a Shaders” (2006). Disponible en http://adva.com.ar/foro/viewtopic.php?t=2099

[15] [consultado en: febrero 2007]. Microsoft Visual Studio.Net, artículo “Características de Visualización C++.Net”. Disponible en: http://www.microsoft.com/latam/visualc/producto/caracteristicas.asp

[16] Cromo. [consultado en: enero 2007]. Linux Juegos, “Graphics3D 6.07“(2005). Disponible en: http://www.linuxjuegos.com/2005/10/07/graphics3d-607/

[17] [consultado en: enero 2007]. Tema 3: "Sistemas de Visualización en Tiempo Real: Visualización por Hardware”. Disponible en: http://www.informatica.uv.es/iiguia/AIG/web_teoria/tema3.pdf

[18] Pressman., R. [consultado en: diciembre 2006] “Ingeniería de Software: Un enfoque práctico”. McGraw-Hill, México.” (2002). Disponible en: http://www.idg.es/pcworld/index.asp?link=estructura/i_articulo_centroArticulo.asp&IdArticulo=173220

[19] [consultado en: febrero 2007]."Tecnologías gráficas en los juegos" (2006) Reportaje: OpenGL. Disponible en: http://www.meristation.com/v3/des_articulo.php?pic=HRD&id=cw44cf819b159be&idj=&idp=&tipo=art&c=1&pos=2

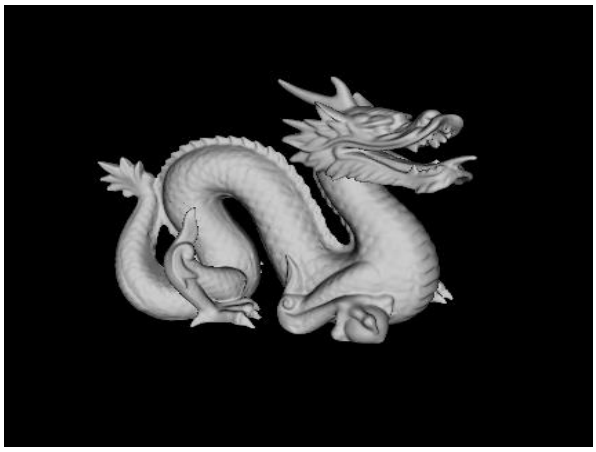
[20] Alonso,.R,.D.[consultado en: enero 2007]. "Radiosidad: Gráficos y Visualización 3D" (2003-2004) Disponible en: http://blackspiral.org/docs/radiosidad/transparencias.pdf

[21] [consultado en: marzo 2007]. "Nueva Tecnología Gráfica". Disponible en: http://www.azken.com/download/Wildcat-Realizm-PO.pdf

[22] [consultado en: mayo 2007]. "Graphics Processing Unit". Disponible en http://es.wikipedia.org/wiki/Graphics_Processing_Unit

APÉNDICES

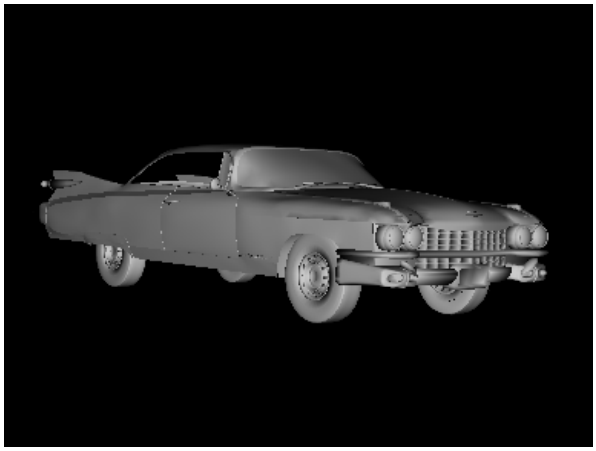
Aplicación de la ecuación de iluminación de Phong conjuntamente con el sombreado de Gouraud propuesto como solución de luces dinámicas en tiempo real sobre los siguientes modelos internacionales.



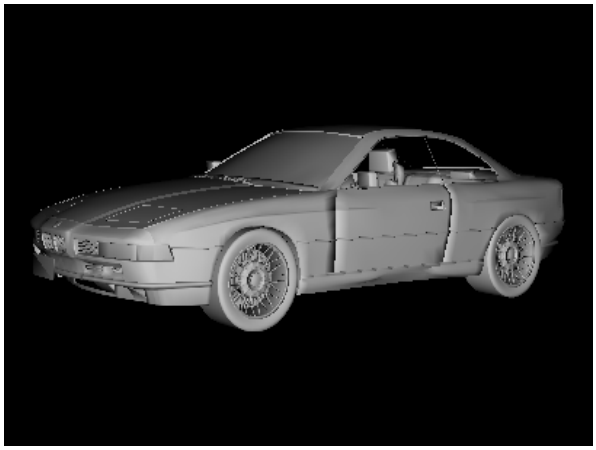
Apéndice 1: Modelo del dragón

Apéndice 2: Modelo del conejo

Aplicación de la ecuación de iluminación de Phong conjuntamente con el sombreado de Gouraud propuesto como solución de luces dinámicas en tiempo real sobre los modelos creados por el GPJV de la F5 para el juego “Rápido y Curioso”.



Apéndice 3: Modelo del auto Chevy



Apéndice 4: Modelo del auto Peugeot

GLOSARIO

1. **Antialiasing:** Empleado para darle suavidad a las curvas del los objetos.
2. **DirectX:** Colección de APIs creadas para facilitar tareas relacionadas con la programación de juegos en la plataforma Microsoft Windows.
3. **Efecto de Mach:** El brillo percibido por el ojo humano no rebasa los límites de intensidad constante.
4. **Engine:** En el contexto de gráficos por computadoras se conoce como motor ó módulo principal gráfico.
5. **Escena:** Cualquier espacio gráfico tridimensional generado por un ordenador.
6. **Estocástico:** Concepto matemático que sirve para caracterizar y estudiar todo tipo de fenómenos aleatorios que evolucionan con el tiempo.
7. **Factor de glossiness:** Factor de pulimiento.
8. **Fragment shaders:** Es un programa que opera en el Fragment Processor y es el encargado de generar los Fragments en lugar del MultiTexturing Fragment Fixed Pipeline el cual es una máquina de estados finitos.
9. **Frame buffer:** Memoria de pantalla.
10. **G3D:** Librería open source orientada a objetos, programadas en C++ y de alto nivel usada para la programación de juegos 3D.
11. **GLSL:** Lenguaje de programación de shaders GL Shading Language.
12. **GPU:** Procesador dedicado exclusivamente al procesamiento de gráficos, para aligerar la carga de trabajo del procesador central en aplicaciones como los videojuegos.

13. **HLSL:** Lenguaje de programación High Level Shader Lenguaje de Microsoft para la GPU en DirectX.

14. **Líneas Scan:** Líneas de barrido empleadas para la interpolación de intensidades y normales.

15. **Luz ambiental:** Luz proveniente del medio ambiente, todas las superficies la reciben independientemente de su ubicación y orientación.

16. **Luz difusa:** Luz reflejada en todas las direcciones.

17. **Luz especular**: Luz reflejada en pocas direcciones que representa el brillo de los materiales.

18. **Objeto:** En el contexto del presente trabajo se refiere a cualquier entidad geométrica contenida en una escena con fuentes luminosas.

19. **OpenGL:** Biblioteca gráfica 3D desarrollada por Silicon Graphics Incorporated.

20. **Patches:** Pequeñas porciones de superficies en las que se divide una escena.

21. **Pipeline:** Canal de proceso y comunicación en paralelo.

22. **Píxel:** Único punto en una imagen gráfica.

23. **Shader:** Fragmento de código escrito en un lenguaje gráfico específico que se ejecuta en una plataforma programable a nivel de hardware, para procesar tanto píxeles (píxel shader) como vértices (vertex shader) y sustituyen etapas del pipeline gráfico.

24. **Subsistema gráfico:** Hardware de una tarjeta gráfica.

25. **Vertex shaders:** Es un programa que se ejecuta sobre cada uno de los vértices a renderizar.

26. **Viewport:** Porción de la ventana que establece una correspondencia entre las coordenadas espaciales transformadas sobre el plano de proyección y las coordenadas en pantalla.