Universidad de las Ciencias Informáticas

UCI

Facultad 5 Realidad Virtual

Efecto de Campo de Profundidad

para Entornos de Realidad Virtual

Trabajo de Diploma para optar por el título de

Ingeniero en Ciencias Informáticas

Autores: Janet Bormey Martín

Yariel Dueñas Hernández

Tutor: Ing. Jaime González Campistruz

Ciudad de la Habana

Julio 2008

OPINIÓN DEL TUTOR DEL TRABAJO DE DIPLOMA.

Título: Efecto Campo de Profundidad para entornos de Realidad Virtual.

Autores: Janet Bormey Martín y Yariel Dueñas Hernández.

El tutor del presente Trabajo de Diploma considera que durante su ejecución los estudiantes mostraron las cualidades que a continuación se detallan.

Por todo lo anteriormente expresado considero que los estudiantes están aptos para ejercer como Ingeniero en Ciencias Informáticas; y propongo que se le otorgue al Trabajo de Diploma la calificación de:

_____

Ing. Jaime González Campistruz

Firma _____

Fecha

DECLARACIÓN DE AUTORÍA

Declaramos que somos los únicos autores de este trabajo y autorizamos a la facultad 5 de la Universidad de las Ciencias Informáticas a hacer uso del mismo en su beneficio.

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

Janet Bormey Martín

_____

Yariel Dueñas Hernández

_____

DATOS DE CONTACTO

Nombre: Jaime González Campistruz

Edad: 25

Ciudadanía: cubano

Institución: Universidad de las Ciencias Informáticas

Título: Ingeniero en Ciencias Informáticas

Categoría Docente: Profesor Instructor

E-mail: jgonzalezc@uci.cu

Graduado en la Universidad de las Ciencias Informáticas como Ingeniero Informático, con 3 años de experiencia en proyecto de Realidad Virtual.

"Internet debe ser un medio de comunicación entre los pueblos que contribuya a la paz mundial y que el principal objetivo de la alta tecnología es mejorar el nivel de vida de las personas."

Larry Ellison

AGRADECIMIENTOS

…a todas las personas que han tenido que ver con la realización del mismo, en especial a nuestro tutor y a la Universidad de las Ciencias Informáticas por darnos esta maravillosa oportunidad.

Yariel

… Quiero agradecerle en primer lugar a una persona que de no ser por él esta tesis no se hubiese podido realizar, a mu tutor Jaime.

… A mis padres Amparo y Miguel por su apoyo incondicional y darme aliento cada vez que lo necesité.

…A mis hermanos Ady y Luis por estar siempre presentes en mi vida.

… A mis abuelos Odavia y Manolo y a mi tío Jose por estar siempre pendiente de mis estudios y por ayudarme cada vez que lo necesité.

…A mi novio por estar presente en los momentos más difíciles de mi carrera y darme siempre el apoyo que necesité.

…A una persona que a pesar de estar lejos siempre me a ayudado mucho, a mi abuela Zenaida.

…A todos mis amigos y en especial a el Cori, Noel, Julio y Marlon por haberme ayudado cuando más necesite de ellos y por demostrarme que en la vida todo se puede, solo hay que tener disposición para hacerlo.

… A todos aquellos profesores que de una forma u otra me enseñaron y me ayudaron a llegar a ser una ingeniera.

…En fin a todos aquellos que han estado conmigo y me han apoyado.

Janet

DEDICATORIA

……este trabajo de diploma esta dedicado a mi compañera de tesis que ha estado todo el tiempo a mi lado en la confección de la tesis.

…a mi tutor Jaime por su eterna colaboración en la realización de este trabajo.

…a dos amigas que quiero como hermanas Susej y Yurian, a Raciel que es como el hermano que nunca tuve, a todos los amigos que he sembrado en estos años aquí en la Universidad.

…a mi familia, a mis abuelos paternos y mi bisabuela que "EPD" en especial a mis primos Milaysis, Yoan, Yosmil que es lo mas bonito que me a pasado, a mis padres que siempre me han alentado en todo momento a no rendirme jamás ante las dificultades y guiarme por el buen camino, a mis abuelos que los quiero con la vida, a mis tíos los más grandes del mundo, a mi hermosa Facultad 5, mi dedicada y paciente decana Mayra, su secretaria Ania, el profesor Millet, Andrea, a mis compañeros de estos 5 años de esfuerzo y continuo trabajo, a mi compañera de tesis que me ha soportado todo este tiempo, el piquete de la descarga el Leo, marquito, Marlene, Yausell, Fácil, Candyman, Parroquia, Mayeta, Jorge, Yaself(gordito), Rijkaard, Gera, Hippie, Rainer, Pupi.

…a las chicas de aula, los mangones, a las cangrejas, al carro de la carne, la majaza, a las negras del piquete de do carmo, a los muchachos de equipo de atletismo y fútbol.

…..a el lechón, piti, la rata.

Se me van a quedar algunos nombres pero en fin a todos los que de una forma u otra han estado involucrados en este trayecto para lograr lo que soy hoy, esto es para ellos que siempre los recordaré por el resto de los días.

Yariel

Quisiera dedicarle esta tesis a dos personas que son la luz de mis ojos, dos personas que siempre están presentes en mi vida. En primer lugar dedicarle esta tesis a mi Abuela Odavia por estar siempre su dedicación, su apoyo, cariño y confianza. También quiero dedicarle esta tesis a una personita que no por el hecho de ser una niña tiene menos importancia o menos valor para mí, a mi Hermanita Ady como cariñosamente le llamo.

Janet

RESUMEN

La Realidad Virtual ha llegado adquirir un gran auge no solo en la Informática, sino también en la vida diaria de las personas. Son muchas las aplicaciones de la Realidad Virtual, estas se utilizan en disímiles campos, así como: la medicina, educación, cine, entre otras. El objetivo principal de la Realidad Virtual es lograr un mejor realismo en las escenas donde esta se aplica. Para esto se utilizan efectos que son los encargados de que el entorno virtual que se está simulando logre el mayor realismo posible para poder lograr engañar al ojo humano y que el usuario se sienta como si estuviese en la vida real.

En este documento quedará reflejado un estudio de uno de los efectos que se utilizan para lograr ese realismo. Este efecto se viene desarrollando desde hace algún tiempo, y en la actualidad es muy imprescindible para hacer aplicaciones de entornos virtuales en tiempo real. Al efecto que se está haciendo referencia es: Campo de Profundidad. El estudio estará enfocado preferentemente hacia las características y aplicaciones que tiene la profundidad de campo en las aplicaciones de entornos virtuales.

Como principal objetivo se hace referencia a los distintos algoritmos que se pueden utilizar para crear este efecto en entornos virtuales. Son varios los algoritmos que existen para crear este efecto; en este trabajo de diplomado solo se explicarán 5 de ellos.

PALABRAS CLAVE:

Realidad Virtual, efecto Campo de Profundidad, Entorno Virtual.

ÍNDICE

INTRODUCCIÓN

Hoy en día, dentro del campo de la Informática, evoluciona rápidamente la rama de la Realidad Virtual (RV). Rama mediante la cuál se establece una nueva forma de relación entre el uso de las coordenadas de espacio y del tiempo, con el objetivo de superar las barreras espaciotemporales y configurar un entorno en el que la información y la comunicación se nos muestran accesibles desde perspectivas hasta ahora desconocidas al menos en cuanto a su volumen y posibilidades.

Los simuladores y video-juegos son ejemplos concretos de esta evolución. Estos hacen uso de los efectos especiales con el objetivo de aumentar la calidad visual y el nivel de realismo en los productos, de manera que el usuario final que lo utilice se sienta engañado por la simulación y logre que este sea absorbido por esta. Por ejemplo, los simuladores de conducción, aviación, quirúrgicos, militares o espaciales.

Dentro de los efectos especiales se encuentra el denominado efecto Campo de Profundidad, efecto que es muy usado actualmente y que no es más que el trabajo en los campos de profundidad del entorno virtual, y de aplicar su uso, la visualización del entorno se torna más realista en cuanto a la simulación de la sensación de profundidad.

En nuestra Universidad, existe una línea de desarrollo dedicada a la Realidad Virtual y a la elaboración de simuladores de diferentes tipos. Simuladores que necesitan entornos virtuales con un realismo aceptable, de manera que los productos tengan la calidad suficiente como para poder emerger en el mercado mundial y tener una mayor demanda. La simulación de estos efectos aumenta considerablemente lo anterior planteado, y para ello es necesaria la completa simulación de una variedad de efectos que sustenten su realismo y que no han sido implementados todavía en la herramienta de visualización que se desarrolla, ni tampoco se ha documentado sobre el tema, de qué es el efecto Campo de Profundidad.

A raíz de lo anterior expuesto, se propone como **problema científico**: ¿Cómo mejorar la visualización de entornos virtuales en los proyectos productivos de la Facultad 5 utilizando el efecto Campo de Profundidad? El **objeto de estudio** de este trabajo es el proceso de creación de Campo de

Profundidad para sistemas de Realidad Virtual, y se tiene como **objetivo general** realizar una comparación entre los métodos que se utilizan para crear el efecto Campo de Profundidad.

En el transcurso del trabajo se le dará respuesta a todos los objetivos trazados y eso será posible ya que el trabajo tiene un campo de acción en el cual enmarcarse. Este **campo de acción** es el siguiente: Efecto Campo de Profundidad para sistemas de Realidad Virtual.

Para dar cumplimiento a los objetivos trazados se tienen en cuenta varias **tareas de investigación**:

1. Estudiar los métodos y algoritmos para la creación del efecto Campo de Profundidad.
2. Describir algunos de los métodos y algoritmos existentes para crear el efecto Campo de Profundidad.
3. Comparar los métodos y algoritmos que se utilizan para crear el efecto Campo de Profundidad.
4. Proponer un algoritmo para la implementación del efecto Campo de Profundidad.

El **resultado esperado** con este trabajo es obtener una amplia investigación en la cual quede plasmada un método y el algoritmo que lo implementa, de manera que los proyectos productivos de la facultad interesados en este efecto puedan basar su desarrollo a partir de este.

Los **métodos teóricos** que se utilizarán en el trabajo son:

Analítico-Sintético: Este método es utilizado aquí ya que se realizará una profunda investigación de todo aquello que ya existe sobre el tema. Aquí se analizarán documentos, teorías y características sobre Campo de Profundidad para así extraer de todo lo estudiado los elementos más importantes y que más se relacionan con lo planteado en los objetivos.

Análisis Histórico-Lógico: Este método es utilizado pues se realiza un estudio de la trayectoria histórica real del efecto, así como su evolución y desarrollo.

Este documento cuenta con 3 capítulos en los cuales se les da respuesta a cada uno de los objetivos trazados. En el capítulo 1 se aprecia la fundamentación teórica, donde se podrán observar algunos conceptos sobre Realidad Virtual así como sus aplicaciones. La Profundidad de Campo es un efecto especial que se utiliza para darle más realismo a las aplicaciones donde se utilicen entornos virtuales,

por esto se puede observar una pequeña historia de los efectos especiales, es decir, como es que fueron surgiendo poco a poco los efectos especiales. Finalmente, en este capítulo ya se habla sobre el efecto Campo de Profundidad analizando primeramente conceptos como antialiasing, blurring, modelos de cámara, shader, conceptos que son fundamentales al crear este efecto y que deben ser conocidos para un mejor entendimiento de lo que se está hablando.

Para crear el efecto Campo de Profundidad se utilizan varios algoritmos, que utilizan su propia metodología. Por esto en el capítulo 2 se describen 5 de los métodos que se utilizan a nivel global para crear este efecto. Un ejemplo de estos algoritmos se tiene el Círculo de Confusión, este es el más utilizado hoy en día.

Estos primeros capítulos son una descripción y caracterización de algunos aspectos importantes a tener en cuenta, pero eso no es suficiente, es importante conocer también como es que se puede crear este efecto. Para esto en el capítulo 3 se propone una solución técnica que servirá de guía en un futuro cuando se determine aplicar este efecto a las aplicaciones virtuales de la facultad. En el mismo se propone un algoritmo para la implementación de este y se muestra un modelo de dominio para dar una mejor idea de como crear este efecto.

CAPÍTULO 1: FUNDAMENTACIÓN TEÓRICA

Introducción

En el mundo del gráfico por computadora juega un papel muy importante el tratamiento de las imágenes virtuales y la aplicación de diversos efectos especiales en aplicaciones del campo de la industria fotográfica, la industria cinematográfica y principalmente en el campo de la Realidad Virtual, debido a que esto trae un considerable aumento en la realidad del producto, que es un factor determinante a la hora de ver la calidad del producto, además de que aumenta también la capacidad de engañar los ojos humanos en las simulación, aspecto que es fundamental en el campo de la Realidad Virtual.

Por ello, la Realidad Virtual tiene mundialmente una importancia vital, desde hace mucho tiempo, ha entrado en un marco de herramientas para hacer, las cuales contribuyen diariamente en diferentes ramas de la sociedad mundial y de ahí su necesidad. Debido a esto se necesita de efectos especiales que le den un mayor realismo a los entornos virtuales. En este capítulo se describirá uno de esos efectos, el efecto Campo de Profundidad. Además antes de adentrarse en lo que es Profundidad de Campo se verán las principales características y aplicaciones de la Realidad Virtual.

1.1- Conceptos sobre Realidad Virtual

Son muchos los conceptos que se han dado sobre la Realidad Virtual, muchos estudiosos del tema han dado sus propias definiciones pero todas están enmarcadas en el mismo objetivo. Por ejemplo:

- "La realidad virtual es como una combinación de la potencia de una computadora sofisticada de alta velocidad, con imágenes, sonidos y otros efectos" [1].

- "La Realidad Virtual es un entorno en tres dimensiones sintetizado por computadora en el que varios participantes acoplados de forma adecuada pueden atraer y manipular elementos físicos simulados en el entorno y, de alguna manera, relacionarse con las presentaciones de otras personas pasadas, presentes o ficticias o con criaturas inventadas"[1].

- "La Realidad Virtual suele asociarse a casi todo lo que tiene que ver con imágenes en tres dimensiones generadas por ordenadores y con la interacción de los usuarios con este ambiente gráfico" [1].

Sin embargo, a pesar de que todas estas definiciones son válidas, no muestran totalmente toda la potencia, todo el jugo que se puede extraer de esta no tan nueva tecnología o forma de trabajar, por lo que la definición más sencilla y la más general es: "La Realidad Virtual es aquella forma de trabajo donde el hombre puede interactuar totalmente con la computadora, generando ésta espacios virtuales (ambientes virtuales) donde el humano puede desempeñar sus labores y donde el humano se comunica con la computadora a través de efectores o dispositivos de interacción"[1].

Como se ve en los conceptos presentados anteriormente la Realidad Virtual tiene muchas características importantes que son determinantes en el uso de esta. Debido a esto, la virtualidad es usada en diferentes sectores y aplicaciones de nuestra contemporaneidad. En este sentido es de suma importancia detenerse y observar algunas de las aplicaciones de la Realidad Virtual.

1.1.1- Aplicaciones de la Realidad Virtual

A pesar de sus límites, el nivel actual de desarrollo tecnológico es suficiente para crear aplicaciones adecuadas para solucionar de una manera eficaz un cierto número de problemas en diversos campos de la actividad humana. Mediante la Realidad Virtual se han creado aplicaciones que pueden significar un importante ahorro de tiempo, dinero y un aumento de la eficacia del trabajo, esto trae consigo que las inversiones efectuadas sean rentabilizadas con mayor facilidad y rapidez.

La Realidad Virtual es utilizada en muchas circunstancias para darle una visión más real y una sensación de realismo a lo que el usuario esta observando. Por ello se emplea en muchos sectores: en el sector de las investigaciones científicas, la medicina, la arquitectura, el diseño industrial, las telecomunicaciones, la ingeniería y la publicidad, entre otros que no son menos importante para el desarrollo mundial.

Ejemplo de ello se tiene:

Ejército: Los simuladores de vuelo para el entrenamiento de los pilotos. Estos sistemas incorporan interfaces para simular la situación dentro del avión así como las distintas maniobras. También en los simuladores para paracaidistas. (Ver figura 1).

Medicina: Existen simuladores que permiten al médico experimentar la sensación de estar ante operaciones complicadas, esto es de gran utilidad ya que permite darle mayor habilidad y destreza a los médicos antes de enfrentarse a una situación real (CathSim).

Educación: El campo de la educación es uno de los más beneficiados de los avances en la Realidad Virtual. Actualmente existen experiencias de distintos grupos de investigación para la creación de material educativo. La compañía Perceptual Computing Group ha implantado un entorno virtual colectivo e interactivo con varios submundos por los cuales pueden navegar y aprender los niños.

Marketing y Comercio electrónico: Cada día, más empresas publicitarias empiezan a ofrecer la posibilidad de realizar visitas virtuales a sus clientes para que observen los productos, navegar virtualmente por los pasillos de un centro comercial, por las habitaciones de un hotel u observar detalladamente vehículos desde distintas perspectivas.

Arquitectura: Una de las primeras aplicaciones en este campo ha sido la creación de modelos virtuales de futuros edificios y la navegación virtual por ellos, tanto para su venta como para su análisis. Tanto es así que diversas compañías comercializan herramientas específicas de diseño tridimensional, de manera que el arquitecto puede experimentar en la etapa de diseño con distintos modelos para decidirse por el más adecuado. También con fines publicitarios u ociosos pueden encontrarse recreaciones virtuales de monumentos históricos o incluso ciudades enteras. Un ejemplo de esto último podemos disfrutarlo en la web virtual de Girona, donde el usuario puede recorrer cada calle de la ciudad e incluso conversar con otros usuarios virtuales que se encuentren dando un paseo virtual. Ejemplo Ottawa, New York, Bilbao.

En la Exploración Planetaria: Dentro de los esfuerzos de investigación, el más notable es el Sistema Virtual de Exploración Planetaria desarrollado por la NASA, dónde se desarrollan y se prueban los conceptos, métodos y estrategias de interacción, útiles para el diseño de estaciones de exploración planetaria.

Ocio y Entretenimiento: Este es el campo más habitual y conocido por todos, pues se puede disfrutar de video-juegos en 3D en nuestras propias casas, en nuestros ordenadores y video-consolas. Ejemplo de estos son: Crysis, Need for Speed, MVP, World of Warcraft. (Ver figura 2)

Precisamente para estas aplicaciones y en especial para la última mencionada, se crean diversos efectos especiales desarrollados en computadora, con el fin de lograr el alto porcentaje de realismo que requieren estas aplicaciones. Estos son de verdadera importancia ya que ayudan conseguir que el usuario se sienta satisfecho al usar los distintos simuladores, videojuegos, entre otros.

1.2- Los efectos especiales en la Realidad Virtual

1.2.1- Historia de los efectos especiales

Desde los inicios del cine se han utilizado los efectos especiales para producir sensaciones en el espectador, motivadas por escenas difíciles o imposibles de conseguir con actores o decorados reales. Son muchos los ejemplos que se podrían mostrar sobre los efectos especiales, a continuación se muestran algunos ejemplos de esto.

1895: Es rodada la película “The Arrival of a Train”, por los hermanos Lumiere en el que un tren parecía que salía de la pantalla para arrollar a los asustados espectadores.

1906: En la película “The Teddy Bears” utilizan la técnica de animación foto a foto en una secuencia donde un oso de peluche cobra vida, de alrededor de un minuto, tardo 56 horas en animarse.

1925: En la película “Ben Hur”, es usada la técnica de las miniaturas colgadas.

Años 40 y 50: La época dorada de los efectos especiales, brilló con luz propia Ray Harryhausen (”Mighty Joe Young”, 1949, “It Came from Beneath the Sea”, 1955...), siendo memorables las escenas de acción de la película “Jason y los Argonautas”.

Década de los 50: Para intentar recuperar audiencia, los productores crean películas cargadas de efectos e introducen nuevos formatos, como Cinemascope, Todd-AO, VistaVision, y 3D.

1968: Se estrena “2001, Una Odisea Espacial”. En dicha película, se utiliza una forma temprana de control de movimiento (motion control), en la que un ordenador controla el movimiento y los parámetros de la cámara.

Década de los 80: Los gráficos por computadora empiezan a cobrar importancia en la producción de las películas. Por ejemplo, el Efecto Génesis de “Star Trek: La Ira de Khan”, 1982, que mostraba una tecnología capaz de dar vida instantáneamente a un planeta muerto, es la primera secuencia totalmente generada por computador que aparece en una película.

1995: “Toy Story”, primera película totalmente generada por computador.

2000: Se utilizan en la película “Dinosaurio” paisajes reales como fondo para mostrar una aventura protagonizada por dinosaurios completamente generados por computador.

1.2.2- Estado Actual

Como se planteó anteriormente, los efectos especiales son utilizados en diversas aplicaciones en el mundo de la Informática. Naturalmente en la actualidad es un factor importante que no puede faltar. En la actualidad, para lograr una mejora del hardware gráfico, con el objetivo de mejorar el rendimiento de la CPU y dar opción de implementar una mayor gama de efectos especiales, se creó la arquitectura superescalar de la nueva serie de GPUs GeForce 6, con su canal de gráficos de 32 bits nativos y sus avances en la generación de imágenes, mejora los valores de velocidad y precisión en una amplia gama de operaciones y efectos visuales. Por primera vez en el mercado gráfico, numerosas operaciones se vuelven factibles en aplicaciones de tiempo real y juegos (filtro de texturas, efectos de alta serie GeForce 6: Imágenes más nítidas y detalladas, rango dinámico, Campo de Profundidad, desenfoque y filtrado anisótropo 16×) para llevar el realismo del cine a la PC. En la actualidad, la Realidad Virtual se plasma en una multiplicidad de sistemas, el más conocido es el que ha desarrollado la empresa norteamericana VPL Research (Visual Programming Language), con la que la NASA trabaja en estrecha colaboración en el desarrollo de sus propias aplicaciones.

En los video-juegos también se utilizan mucho los efectos especiales, se tiene, por ejemplo, el juego Crysis, el cual es, hoy en día, uno de los mejores y de los más realistas debido a la gran cantidad de efectos especiales que tiene implementados.

Uno de los principales efectos que se tiene en cuenta a lo hora de lograr un mejor realismo es el efecto Campo de Profundidad. En este sentido es importante conocer qué es la profundidad de campo y algunos de sus aspectos más importantes, así como sus aplicaciones.

1.3- Efecto Campo de Profundidad

En el mundo real, en la fotografía o la cinematografía, las propiedades físicas de la cámara pueden manchar algunas partes de la escena mientras se mantenga la agudeza en estas áreas. Existen, algunas veces, imperfecciones que distorsionan la imagen original y los detalles de la escena, también se usan herramientas que proporcionen valiosas pistas y guíen la atención a las partes importantes de la escena.

Así, como en la fotografía, desde los inicios que empezaron a crearse los video-juegos, el hombre siempre ha querido que estos sean lo más real y cercano posible a la vida humana, que estos actúen tal y como son las personas. Para poder acercarnos más a esa realidad se utiliza el efecto Campo de Profundidad el cuál le aumenta el sensacionalismo y la realidad a los videojuegos, este efecto trae consigo que todos los objetos en un rango de vista determinado y a cierta distancia se van a ver con la claridad suficiente y si su distancia aumenta, la imagen se hace más engorrosa, así como sucede con los seres humanos en la vida real. (Ver Figura 3 y 4).

1.3.1- Conceptos Generales

Se define como Profundidad de Campo: el rango de distancias reproducidas donde la imagen es aceptablemente nítida comparada con el plano más nítido de la misma [2].

La Profundidad de Campo, no es una zona en la que se está enfocada perfectamente, sino, la zona donde el foco es lo suficientemente cercano al plano nítido como para ser aceptable. Esta no dicta tampoco cuán borrosos estarán los planos alejados del plano nítido, una confusión común [2].
El efecto Campo de Profundidad es una parte integral de la visión humana, en la que el lente del ojo humano se acomoda a las distintas distancias de observación. Cuando un objeto está fuera de ese foco de observación se ve borroso. El grado de desenfoque depende de la potencia del lente, el diámetro y la distancia del objeto.

Se han desarrollado algunos algoritmos para crear este efecto, sin embargo, la mayoría de estos algoritmos son demasiado costosos para utilizarlos en aplicaciones en tiempo real. En general todos estos algoritmos se clasifican en dos grupos: métodos de filtrado de post.proceso y de rendering de múltiples pasadas.

Para realizar el efecto Profundidad de Campo se requiere dominar algunos conceptos generales como: antialiasing, modelos de cámara, blurring, shader; por ello se ofrece una panorámica de ellas.

1.3.1.1- Antialiasing

Dentro de los gráficos por computadora es fácil reconocer el aliasing como aquellos pequeños saltos que se producen en la continuidad de las imágenes.

Se denomina "alisación" (*antialiasing*) a los diversos recursos que se utilizan para ocultar o corregir *alias* o artefactos, un término de teoría de la imagen que se refiere a las defectos que aparecen cuando se quiere simular un objeto continuo que presenta cambios notables de aspecto (o, más exactamente, una función de variable continua con bruscos cambios de intensidad) por medio de muestras discontinuas. Esto se hace, en 3D Studio, al igual que en otros programas, tanto desde el Editor de Materiales, por medio de parámetros específicos de cada mapa, como desde los Controles de Rendering [7].

Estos se producen tanto en líneas rectas como en curvas y el motivo de las mismas surge de la imposibilidad de las pantallas de representar una imagen ideal en forma perfecta. Las imágenes ideales están definidas por funciones f(x, y) donde x e y son números reales que representan las coordenadas en la misma imagen. Evidentemente, esta representación contará con un nivel de detalle infinito, alcanzando obtener de manera exacta el color de cada punto en la imagen. Ahora bien, los equipos tienen limitaciones y no permiten mostrar más que una cantidad limitada de puntos, que en una pantalla difícilmente se supere los pocos millones de puntos. El simple hecho de “recortar” el nivel de definición produce efectos indeseables en el resultado final que son, por ejemplo, los ya mencionados serruchos o escaleras en la continuidad de las líneas, muy fácilmente detectables en los primeros planos de los juegos en primera persona, tanto en las armas como en las manos. También se genera lo que se conoce como $patrones\ de\ Moiré$[1].

El antialiasing es el método con el cuál se disminuyen estos defectos, buscando lograr formas más parejas y uniformes dentro de la imagen. Contrariamente a lo que muchos piensan, el antialiasing no funciona únicamente sobre líneas rectas a pesar de que el defecto suele ser más factible sobre estas, persiste sobre las curvas y allí también trabaja.

1.3.1.2- Difuminación (Blurring)

Cuando una cámara crea una imagen, aquella imagen no representa un instante de tiempo, si no que representa la escena para el período del tiempo. La imagen de una escena debe representar una integración de todas las posiciones de aquellos objetos, así como el punto de vista de la cámara. En una imagen determinada, cualquier objeto que se mueve en lo que concierne a la cámara parecerá velado o untado a lo largo de la dirección de movimiento relativo. Esto puede ocurrir sobre un objeto que se mueve o sobre un fondo estático si la cámara se mueve. Como el efecto es causado según el movimiento relativo entre la cámara, los objetos y la escena, el aspecto borroso de movimiento puede ser evitado por la cámara para rastrear aquellos objetos de movimiento. En este caso, los objetos aparecerán más agudos, y el fondo más velado [8].

Asimismo, en la animación de ordenador en tiempo real cada marco muestra un caso perfecto en el tiempo con el aspecto borroso igual cero. Esto es porque en un video-juego con un valor de marco de 25-30 marcos por segundo parecerán sombreados, mientras el movimiento natural filmado en la misma tarifa de marco aparece continuo. Muchos video-juegos de generación destacan el aspecto borroso de movimiento, sobre todo juegos de simulación de vehículo. En la animación de ordenador pre-dada, como películas CGI, el aspecto borroso de movimiento realista puede ser dibujado porque el render tiene más tiempo para dibujar cada marco. El Anti-aliasing temporal produce marcos como un compuesto de muchos instantes.

La selección de una operación de blur de 2D para aplicarse a las capas está basada en las siguientes consideraciones: el algoritmo debe generalizar suavemente el $blur^2$ producido por la profundidad realista del campo; la propiedad relevante de la Profundidad de Campo se relaciona con la oclusión parcial a lo largo de los bordes de un objeto desenfocado. Como el lente de una cámara tiene una abertura finita, los bordes de un objeto velado se hacen semitransparentes; se debe decidir como la oclusión parcial debería comportarse en la presencia de campos de blur arbitrarios; no se puede utilizar en cualquier cámara físicamente realizable o modelo de cámara.

Por ejemplo, una simple escena que consta de dos capas, un primer plano y un objeto de fondo: se puede hacer el blur a la capa del primer plano de acuerdo al blur del campo, manteniendo al mismo tiempo la oclusión parcial estimable. Primero, se explora una variante de convolución espacialmente ingenua (en cada capa). Ciertamente, esto da resultado a imágenes que son más o menos borrosas,

de conformidad con el blur del terreno, y, de hecho, se obtiene una transparencia parcial a lo largo de los bordes. Sin embargo, la transparencia también se produjo en el interior de las regiones borrosas aisladas a los objetos del primer plano. (Ver figura 5).

1.3.1.3- Modelos de Cámaras

Las imágenes generadas por computadora con métodos convencionales se ven demasiado nítidas y carecen de los efectos y artilugios de las imágenes captadas con cámaras verdaderas. Sin dichos efectos, es difícil engañar al ojo humano, como para creer que fueron imágenes capturadas por una cámara de verdad. La calidad de las cámaras se torna aún más importante cuando las imágenes generadas por computadora se han de combinar con otras producidas mediante cámaras reales. La discrepancia visual surge en gran medida a partir de la diferencia que existe entre las cámaras tradicionales y aquellos modelos de cámaras normalmente utilizadas para imágenes computarizadas. Las imágenes computarizadas generalmente hacen uso de modo implícito del modelo de cámara de agujero mientras que las cámaras reales utilizan lentes de dimensiones finitas.

Modelo de cámara de agujero

Una cámara de agujero es una cámara sin lente. [3] En este tipo de cámara se trazan infinitos rayos de luz de un objeto que pasan por un pequeño agujero. De todos los rayos emitidos solo uno es capaz de pasar a través de él y los demás son desechados. Ese rayo que logra pasar a través del agujero proyecta la imagen de un punto en un plano. (Ver figura 6).

En este modelo de cámara mientras más pequeño sea el agujero más nítida será la imagen proyectada en el plano. Un empleo común de una cámara de agujero es la captura del movimiento de luz del sol a lo largo de un período de tiempo.

El modelo de cámara de agujero describe la relación matemática entre las coordenadas de un punto 3D y su proyección en el plano, donde la abertura de la cámara es descrita como un punto y como se dijo anteriormente no se usa ningún lente para enfocar la luz. El modelo no incluye, por ejemplo, las distorsiones geométricas o difuminación de las imprecisiones causadas por los lentes y aberturas de

tamaño finito. Así como tampoco se tiene en cuenta las coordenadas de la imagen. Esto significa que el modelo de cámara de agujero solo puede ser usado para dar una aproximación de una escena en 3D a una imagen en 2D. Su validez depende de la calidad de la cámara y, en general, disminuye desde el centro de la imagen a los bordes con el objetivo de aumentar los efectos de distorsión.

Lentes de dimensiones finitas

Todos los lentes tienen dimensión finita y los rayos vienen desde diferentes direcciones. Un lente con longitud de foco f y una imagen sostenida de un determinado objeto es producido en el plano para deslucir desde los lentes por v, cuando el objeto está a una distancia u del lente. La ecuación 1 representa como se adelgaza el lente:

$$\frac{1}{u} + \frac{1}{v} = \frac{1}{f}$$

Ecuación 1

La distancia desde el plano de la imagen al objeto en el foco puede ser expresada como se ve en la ecuación 2:

$$Z_{focus} = u + v$$

Ecuación 2

Múltiples rayos se esparcen desde un determinado punto en un objeto que pasará a través de un lente, moldeando el cono de la luz. Si el objeto esta en el foco, todos los rayos convergerían en un punto simple o en el plano de la imagen. Sin embargo, si un determinado punto en un objeto de la escena no se encuentra cerca de la distancia focal, los rayos de la luz interceptarían la imagen del plano en un área conformada como una sección del cono. Típicamente, la sección de un cono está dada aproximadamente por un círculo llamado círculo de confusión.

En diámetro del círculo de confusión b depende de la distancia del foco al plano y de la obertura del lente en el escenario. Para la distancia de un foco conocido y los parámetros del lente, el diámetro del círculo de confusión se calcula como se muestra en la ecuación 3:

$$b = \frac{D * f(Z_{focus} - Z)}{Z_{focus}(Z - f)}$$

Ecuación 3

Donde:

D: Diámetro del círculo de confusión

f: Longitud del foco

$Z_{focus:}$ Distancia del plano al Objeto

Z: Obertura del lente

Para realizar estos cálculos es necesario conocer el diámetro del lente. La ecuación 4 representa la forma de realizar este cálculo:

$$D = \frac{f}{a}$$

Ecuación 4

Donde:

f: Longitud del foco

a: Altura del objeto con respecto al eje z

Cualquier círculo de confusión es más grande que cualquier punto de un ojo humano que pueda resolver a la contribución de la imagen que se puede ver como Profundidad de Campo. (Ver figura 7).

1.3.1.4- Shader

Lo expresado anteriormente son 3 conceptos fundamentales a tener en cuenta a la hora de crear el efecto Campo de Profundidad en tiempo real, pero estos no son los únicos. Para la programación es necesario utilizar la GPU. Una GPU es un procesador dedicado exclusivamente al procesamiento de gráficos, para aligerar la carga de trabajo del procesador central en aplicaciones como los video-juegos o aplicaciones 3D interactivas. De esta forma, mientras gran parte de lo relacionado con los gráficos se procesa en la GPU, la CPU puede dedicarse a otro tipo de cálculos (como la inteligencia artificial o los cálculos mecánicos en el caso de los video-juegos).

Una GPU 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. Adicionalmente, existen primitivas para dibujar rectángulos, triángulos, círculos y arcos. Las GPU actualmente disponen de gran cantidad de primitivas, buscando mayor realismo en los efectos. Además, la GPU está altamente segmentada, lo que indica que posee gran cantidad de unidades funcionales. Estas unidades funcionales se pueden dividir principalmente en dos: aquéllas que procesan vértices, y aquéllas que procesan píxeles. Por tanto, se establecen el vertex y el píxel como las principales unidades que maneja la GPU. El píxel shader y el vertex shader son dos etapas en el $pipeline\ gráfico$[3] de render del shader. Por lo tanto es fundamental detenerse a conocer que es el shader antes de estudiar el píxel shader y el vertex shader.

Un Shader, es un conjunto de instrucciones gráficas destinadas para el acelerador gráfico, estas instrucciones dan el aspecto final de un objeto. Los Shader determinan materiales, efectos, color, luz, sombra, entre otras [4].

Los lenguajes existentes para el uso de shader son:

- HLSL usado con $DirectX$[4] y propiedad de $Microsoft$[5].
- GLSL usado con $OpenGL$[6] y libre.

- CG propiedad de la empresa $NVidia^{7}$.

Usualmente mientras más avanzado sea el Shader, la cantidad de objetos, texturas, efectos ambientales (Sol, Nubes 3D, Humos, Fuegos Realistas, Aguas, Iluminación) serán mayores con formas, colores y texturas más realistas.

Como se explicó anteriormente en el Pipeline gráfico de dibujado, se diferencian dos tipos de Shader, que son: el pixel shader y el vertex shader. Estos dos conceptos son muy utilizados en la creación de los efectos especiales en los entornos virtuales y en este caso en el efecto Campo de Profundidad, por lo que es importante definir que son el píxel shader y el vertex shader. Para esto se procede a definir cada uno de ellos.

Vertex shader

El vertex shader es una herramienta capaz de trabajar con la estructura de vértices de los modelos tridimensionales y con ello realizar operaciones matemáticas modificando estas variables y así definiendo colores, texturas e incidencia de la luz. Esto da libertad a los programadores para realizar diferentes efectos desde la deformación de un objeto hasta la recreación de las olas del mar [5].

En caso de representaciones gráficas de pelo, se basaría en los vértices de la maya dando un efecto más realista al resultado, lo que conlleva una rápida ejecución de la imagen puesto que se utiliza el hardware específico, en este caso el de las tarjetas gráficas.

Lo que en realidad pretende esta herramienta es adicionar a una maya de polígonos elementos que se alojan en los vértices de dichos polígonos o simplemente modificarlos. Incluido en $Direct3D^{8}$ y OpenGL, el vertex shader puede reproducir diferentes efectos realistas.

Píxel shader

El píxel shader no interviene en el proceso de la modelación del "esqueleto" de la escena, sino que forma parte de la segunda etapa: la rasterización [6]. Allí es donde se aplican las texturas y se tratan los $píxel$[14] que forman parte de ella. Básicamente, un píxel shader especifica el color de un píxel. Este tratamiento individual de los píxeles, permite que se realicen cálculos principalmente relacionados con la iluminación del elemento que forma parte de la escena, y en tiempo real. Teniendo la posibilidad de iluminar cada píxel por separado. De esta forma es como se logró crear los fabulosos efectos que se pueden apreciar en Doom 3, Far Cry y Half Life 2, por mencionar sólo los más conocidos. La particularidad de los píxel shader es que, a diferencia de los vertex shader, requieren de un soporte de hardware compatible. En otras palabras, un juego programado para hacer uso de píxel shader requiere de una tarjeta de video con capacidad para manipularlos.

Luego de conocer los principales conceptos que se emplean cuando se crea el efecto Campo de Profundidad es importante también conocer cuál es la teoría físico real que sustenta este efecto. En el siguiente epígrafe se describe la teoría físico real.

1.3.2- Teoría Física Real

Para crear el efecto Campo de Profundidad se necesita de un lente caracterizado por la longitud focal y la obertura. La longitud focal es la distancia que existe entre el lente y los rayos paralelos que se trazan hasta un punto. Debido a esto surge una abertura que no es más que el diámetro del lente cuando recibe la luz. (Ver figura 8). Siguientemente de esto, se tiene la relación de los puntos que se utilizan para calcular la longitud focal, donde, d es la obertura del lente, u_0 se encuentra en el foco del plano de la imagen donde también se encuentra v_0, c es el diámetro que se forma cuando la imagen se encuentra entre v_0 o v_n y u_n o u_f, la imagen se encuentra en este caso en v_0, La relación que existe entre estos valores se observa en la ecuación 5:

$$\frac{v_n - v_0}{v_n} = \frac{c}{d} = \frac{v_0 - v_f}{v_f}$$

Ecuación 5

Para calcular c se tiene la ecuación 6:

$$c = d * \left| \frac{v_p - v_0}{v_p} \right|$$

Ecuación 6

Donde v_p y v_0 son cualquier punto del espacio.

1.3.3- Métodos Off-Line y Online

Para crear el efecto Campo de Profundidad existen métodos Off-Line y métodos Online. Los métodos Online son aquellos donde el estudio de hardware y los sistemas de software son sujetos a una coacción en tiempo real, plazos (fechas límites) operacionales del acontecimiento a la respuesta de sistema. Por el contrario, los métodos Off-Line son para el cual no hay ningún plazo (fecha límite), incluso si la respuesta rápida o el alto rendimiento son deseados o aún preferidos. En el caso de los métodos Online existen distintos algoritmos que pueden ser utilizados para crear este efecto. A continuación se mencionaran 5 de estos algoritmos y posteriormente en el capítulo 2 se describirá cada uno de ellos. Estos son:

- Círculo de Confusión[6]
- Efecto Blur[9]
- Blurring con filtro Gaussiono separable[6]
- Over-complete Discrete Wavelet Transform[11]
- Rendereo por punto[10]

Como se explicó anteriormente, en las fotografías se hace uso de los métodos Off-Line para crear el efecto Campo de Profundidad y esto depende de tres factores importantes:

- Apertura del Diafragma.
- Distancia Focal.

- Distancia de la cámara al motivo enfocado.

Apertura del diafragma

El diafragma es el encargado (junto la velocidad de disparo) de regular el flujo de luz que penetra en la cámara. Además, incide directamente en la Profundidad de Campo. Cuanto más cerrado este el diafragma (mucha luz, *f* mayor) la Profundidad de Campo aumenta.

Distancia focal

Para regular la cantidad de luz que incide sobre el sensor, este se expone a la luz que penetra por la óptica del objetivo mediante un sistema de apertura, que normalmente varía entre 1/2000 seg y 4 seg, otras hasta 15 seg. Esta velocidad interviene en que en la nitidez de la imagen del movimiento de los elementos fotografiados influya el movimiento del fotógrafo al tomar la imagen. Para trabajos con escasa luz, se deberá de asegurar la estabilidad de la máquina (mediante trípode y disparador), y limitar los movimientos de los motivos a fotografiar.

Distancia de la cámara al motivo enfocado

Es la distancia que existe entre el centro del lente y el sensor CCD. Estas medias suelen variar en las cámaras digitales con zoom óptico entre 6 mm y 24 mm. Esto equivale en la fotografía química de 35 mm, al angular (W) de 30 mm y el teleobjetivo (T) de 115 mm.

La distancia focal, en un inmediato estudio, acerca o aleja los motivos. Con el angular (W) ensancha el campo, las cosas parecen más lejanas y entran más elementos en la fotografía. Con el tele-objetivo (T), acerca la imagen.

Pero una segunda consecuencia de la distancia focal se encuentra en que el tele-objetivo aplana la imagen, acercando o amontonando los elementos que en la realidad se encuentran en segundo plano con los de primer plano.

Después de haber abarcado los conceptos fundamentales sobre profundidad de campo es bueno conocer en que se utiliza este efecto.

1.3.4- Aplicaciones

El efecto Campo de Profundidad tiene muchas aplicaciones en el mundo actual en que se desarrolla la especie humana, es un factor importante pues se utiliza para darle una mejor visualización al ojo humano, ya que a través de este se crea una visión más realista de lo que el ojo es capaz de observar. Existen muchos ejemplos donde se ha aplicado este efecto; los simuladores necesitan que sean lo más reales posibles para que el cliente se sienta satisfecho y contento con el producto que tiene en sus manos. Este efecto tiene un papel muy importante en la realización de video-juegos y películas de animados. Es muy utilizada en el sector militar como por el ejemplo, el simulador de tiro que se realizó en $SIMPRO^{9}$, para que lo puedan utilizar nuestros soldados cuando están pasando el servicio militar.

CAPÍTULO 2: MÉTODOS DE SIMULACIÓN DE CAMPO DE PROFUNDIDAD

Introducción

Actualmente existe más de un algoritmo para el uso del efecto Campo de Profundidad en tiempo real. Algoritmos que presentan sus propias características, diferenciándose en un conjunto desventajas y ventajas que en este capitulo se analizarán.

Cabe destacar que estos algoritmos son muy costosos para utilizarlos en aplicaciones en tiempo real, pero no por esto dejan de ser imprescindible y su utilización es muy importante en la creación de entornos virtuales. Por esto es muy importante conocer sobre estos algoritmos y en los epígrafes de este capítulo se podrá apreciar una descripción de cinco de estos, los cuales se pueden utilizar para crear el efecto Campo de Profundidad.

2.1- Blurring con filtro Gaussiano separable

El algoritmo blurring con $filtro\ Gaussiano$[10] separable fue desarrollado por Rokita donde hace uso de un filtro Gaussiano para la realización de este. Este algoritmo fundamentalmente se aplica en el post-procesamiento de la imagen con fases de cambio durante la simulación mediante el círculo de confusión del lente de la cámara física. Lo primero que se hace es hallar la resolución del buffer, se determina la profundidad para cada píxel del buffer. Luego se reduce la imagen ½ en el eje x y ½ en el eje y. Acto seguido de esto se enturbia la imagen en dos pasos separados, primero para el eje x y luego para el eje y. Por último se mezclan la imagen original y la imagen post procesada. Es necesario conocer mejor que se realiza en cada paso y para esto se da una explicación más abierta y detallada de cómo se realiza este método. Vale destacar que este algoritmo cuenta con cinco pasos fundamentales.

2.1.1- Primer Paso: Renderizado de Escena

Este primer paso consiste en darle la máxima resolución a la escena fuera del $buffer$[11] calculando el color y la profundidad para cada píxel. Esto determinará la cantidad de píxeles que serán enturbiados en el post-procesamiento. Cuando se renderiza la escena hay que hacerlo teniendo en cuenta el píxel shader y el vertex shader en los cuales se calculan determinados valores que son necesarios.

Renderizando escena del vertex shader

Aquí se calcula la profundidad y el plano focal del vertex shader. Primero se determina la distancia de cada vértice en el plano focal obteniéndose un rango de distancia en escala de 0…1 del plano focal en el interpolador de textura.

Renderizando escena del píxel shader

En el píxel shader se renderiza la escena como se desee. El canal alpha recibe el valor del blurring expresado como la distancia del plano focal.

2.1.2- Segundo Paso: Downsampling

En este segundo paso se toma una imagen de máxima resolución, se renderiza ¼ del tamaño de la imagen original, mientras que el muestreo de la imagen original y la salida es para los buffer más pequeños fuera de pantalla. El canal alpha de la imagen a la que se le hizo el $downsampling$[12] recibe valores calculados del efecto borroso como la distancia escalada del plano focal de cada píxel. Esta información se utilizará durante el post-procesamiento para controlar la cantidad de efecto borroso que será aplicado a la imagen a la que se le hizo el downsampling, así como una mezcla de la imagen borrosa de la escena y la renderización original para simular el efecto Campo de Profundidad. En este paso también se realizan varias operaciones.

Downsampling para el vertex shader

En este paso se transforman los vértices y se propagan las coordenadas de textura entrante para el píxel shader. Hay que tener en cuenta, que el modelo entrante debe ser un $quad$[13] de pantalla alineada de dimensiones de 1/4 de tamaño de la imagen original.

Downsampling para el píxel shader

En el píxel shader para pasar el downsampling se muestra la escena original renderizada usando coordenadas de textura del quad de pantalla alineada más pequeño y se almacenan los resultados en un target renderizado fuera de pantalla.

Post-procesamiento

Uno de los filtros más frecuentemente utilizados para la realización de suavizado de una imagen es el filtro Gaussiano. La ecuación 6 muestra como se aplica este filtro:

$$F = \frac{\sum_{i=1}^{n} \sum_{j=1}^{n} P_{ij}\, C_{ij}}{S}$$

Ecuación 7

Donde:

F: Es el valor filtrado del $target\ píxel$[14].

P: Es un píxel en la rejilla 2D.

C: Es un coeficiente en la matriz gaussiana de 2D.

n: Son las dimensiones vertical / horizontal de la matriz.

S: Es la suma de todos los valores en la matriz gaussiana.

Una vez que un núcleo adecuado se ha calculado, el suavizado Gaussiano se puede realizar utilizando los métodos estándar de convolución. La convolución puede, de hecho, ser realizada con bastante rapidez ya que la ecuación Gaussiana isotrópica 2D es separable en componentes "**x**" y "**y**". De este modo, la convolución 2D puede ser llevada a cabo: primero se convoluciona con una Gaussiana 1D en la dirección **"x"**, y, entonces se convoluciona con otra Gaussiana 1D en la dirección **"y"**. Esto permite aplicar un filtro de mayor tamaño a la imagen de entrada en dos pasos sucesivos de filtros 1D. Se realizará esta operación renderizando en un buffer temporal y muestreando una línea (o una columna, por eje "**y**" filtrado) de $texel$[15] por cada uno de los pases.

El tamaño del buffer del downsampled determina el tamaño de texels utilizados para controlar los puntos de muestreo para los filter taps del filtro Gaussian. Esto puede ser pre-calculado como una constante para el shader antes de tiempo.

2.1.3- Tercer Paso: Filtro Gaussiano separable para el eje x

En este paso lo primero que se hace es hacerle el filtro Gaussiano a la imagen disminuida en el eje x. La muestra de centro y las taps interiores del filtro son hechas con las coordenadas de textura interpoladas que son calculados en el vertex shader. Para calcular las compensaciones para las siete primeras muestras se usa la coordenada de textura de entrada y las compensaciones de taps pre-calculadas basadas en la resolución de la imagen.

El código del píxel shader saca las coordenadas de textura para las muestras externas basadas en deltas pre-calculados de la posición de la muestra de centro.

Todas las muestras son basadas en los umbrales pre-definidos y los valores borrosos. Esto causa una suma ponderada de 25 texels de la imagen de la fuente, que es bastante grande para permitir crear un efecto convincente que enturbia para simular la Profundidad de Campo sin violar el número máximo de instrucciones para 2.0 píxel shader.

2.1.4- Cuarto Paso: Filtro Gaussian separable para el eje y

En este paso se realiza lo mismo que en el anterior lo que a lo largo del eje y. La imagen de entrada que se toma es la imagen que ya se le realizó el enturbiado a lo largo del eje x.

Vertex Shader para el filtro Gaussian separable en el eje y

En el vertex shader se vuelve a calcular las compensaciones de muestras de textura para usarlas en el píxel shader para el muestreo de la imagen prevelada. Aquí el vertex shader usa el mismo acercamiento que para el eje x lo que con valores de compensación diferentes.

Píxel Shader para el filtro Gaussian separable en el eje y

De modo similar al tratamiento de la imagen en el paso anterior, se prueban las 7 primeras muestras a lo largo del eje y que usa compensaciones de textura interpoladas. Entonces se calcula después 6 coordenadas de compensación y se prueba la imagen que usa la textura dependiente. Finalmente se combinan todas las muestras y se da un valor de salida fuera del buffer.

2.1.5- Quinto Paso: Composición Final

En el paso final se crea una imagen compuesta obtenida en la renderización de la escena con la imagen obtenida en el Gaussiano que usa la distancia de la información focal plana que es almacenada en el canal alfa de la imagen original. Todas las texturas son probadas usando coordenadas de textura interpoladas. En el vertex shader, simplemente se transforman los vértices y se propaga la coordenada de textura al píxel shader y en este se recupera la profundidad y la distancia almacenada en el canal alfa de la imagen.

2.2- Rendereo por punto

Otro de los métodos que se utilizan para la realización del efecto Campo de Profundidad es el rendereo por punto. Este algoritmo fue desarrollado por primera vez en el 2001 por Zwicker y en el 2002 fue mejorado por Gross. Es usado tanto para superficies opacas como para superficies transparentes. Se basa en definir una colección de puntos. Cada punto tiene una posición, la normal, radio, color difuso, un color especular y un brillo especular. Todos los puntos son almacenados en una serie de vértices para que un gran número de estos sea enviado a la tarjeta gráfica en una sola operación. Para esto se emplea un vertex shader para procesar cada punto durante la interpretación. Para calcular el color final se hace mediante per-vertex shading. La ecuación 8 representa como se calcula el color final:

$$c = K_{e+} K_a A + K_d \sum_{i=0}^{n} l_i (N * L_i) + K_3 \sum_{i=0}^{n} l_i (N * H_i)^3$$

Ecuación 8

Donde:

K_e : Término de emisión para cada punto.

K_a: Término ambiental para cada punto.

A: Color ambiental.

K_d: Difuso.

l_i: Color de luz.

N: Normal del punto.

L_i: Vector normalizado del punto.

K_3: Valor especular del punto.

H_i: Vector normalizado que se encuentra entre el L_i y el vector del punto.

Aquí también se calcula el vertex shader para darle espacio a la pantalla en cada punto donde r es el radio, d la distancia, w es la anchura de la pantalla y fov es el campo visual horizontal. La fórmula para calcular el vertex shader se observa en la ecuación 9:

$$size = \frac{r * w}{2 * d * \tan\left(\frac{fov}{2}\right)}$$

Ecuación 9

Hasta el momento se ha usado un modelo de cámara de agujeros pero para realizar la Profundidad de Campo se necesita una cámara que tenga lente de dimensiones finitas. En este modelo se especifica la obertura(n) y la longitud focal (f). El cálculo de la longitud focal se realiza mediante la ecuación 10:

$$F = \frac{\cos\left(\frac{fov}{2}\right)}{\sin\left(\frac{fov}{2}\right)}$$

Ecuación 10

Ahora se procede a calcular el círculo de confusión para una distancia P del plano focal y una longitud focal F. Este círculo es usado para mostrar puntos desenfocados. El radio del círculo de confusión *C* para un punto a una distancia *U* del lente se determina como se muestra en la ecuación 11:

$$C = \frac{1}{2}|V_u - V_p|\frac{F}{nV_u}$$

Ecuación 11

Donde V_u y V_p se calculan como se muestra en las ecuaciones 12 y 13 respectivamente

$$V_u = \frac{FU}{U - F}, U > F$$

Ecuación 12

$$V_p = \frac{FP}{P - F}, P > F$$

Ecuación 13

Finalmente se determina el círculo de confusión (Ver Ecuación 14):

$$C_r = C \frac{w}{R - L} \frac{Z_{near}}{V_p}$$

Ecuación 14

Donde w es la anchura de la pantalla, R y L son los valores a la derecha e izquierda del $frustum$[16], Z_{near} se refiere al plano recortado. Este radio es calculado para cada punto en el vertex shader además de que se calcula un valor alpha para lograr los efectos correctos de enturbiado. Los valores de alpha oscilan entre 0 y 1, esto quiere decir que un valor alpha de 0 es totalmente transparente y un valor 1 de alpha quiere decir que es totalmente opaco. Por ejemplo, en una imagen se le da valores de alpha 1 a los puntos que están enfocados y 0 a los puntos que están fuera del foco.

En la práctica, una cota inferior es puesta sobre el valor alfa para cada punto con el fin de prevenir las partes de la escena que tienden a desaparecer completamente cuando los puntos son sumamente desenfocados.

Con el fin de producir una correcta imagen borrosa los puntos deben ser ordenados según la profundidad comenzando por los puntos más lejanos. Estos puntos son dados por una función común (Ver ecuación 15):

$$c = \propto_1 c_1 + (1 - \propto_1) c_2$$

Ecuación 15

Hay que tener muy en cuenta que la suma de los colores sea 1. Hasta el momento se almacenan los puntos en una estructura jerárquica de subdivisión llamada octree. Un octree es una estructura de datos de árbol donde cada nodo tiene 8 nodos hijos. La notación que se utiliza es UNE para definir el eje que ocupa el Y positivo (superior), Z negativo (norte) y X positivo (este) con respecto al nodo padre. Los puntos de datos se almacenan en la hoja nodos como $Vertex\ Arrays$[17]. Hay que tener en cuenta que el nodo raíz es el nodo más grande y abarca a todos los demás nodos hijos. Este paso puede llegar a ser muy costoso pero es realizado cuando la escena es cargada. Una vez construido el octree la escena no se podría cambiar ya que sería estática.

2.3- Over-complete Discrete Wavelet Transform

Un método interesante fue descrito a principios de 1880 por Joseph Fourier, probó que se podía superponer los senos y los cosenos para representar otras funciones. Wavelet es un método matemático que tiene determinados requisitos y es utilizado en la representación de datos u otros tipos de funciones. Este método puede ser utilizado en distintas aplicaciones. Dentro de la Realidad Virtual se puede aplicar al crear el efecto Campo de Profundidad. Este algoritmo es llamado Discrete Wavelet Transform.

Este algoritmo crea una sola imagen donde todos los objetos de la imagen son el foco. Es usado en muchas aplicaciones de representación, pero es en particular útil en la microscopia de campaña brillante, donde las lentillas de aumento altas usadas causan una profundidad muy estrecha del campo. Por lo tanto, una imagen tomada en cualquier posición focal sólo tendrá un subconjunto de los objetos en el foco.

Típicamente este método es como una pila de imágenes, esto quiere decir que se tendrá en la pila la imagen en varias posiciones del foco s(x, y, z). Aquí se selecciona la mayor parte de la imagen y se crea una nueva imagen p(x, y). Ya esta imagen vendría siendo conveniente para el análisis visual o automatizado como un sustituto de la pila en 3D.

Con lo planteado anteriormente se puede describir y decir que este algoritmo es usado para ofrecer resultados prometedores comparados con otros métodos basados en la métrica focal más tradicional. En su forma más simple, se realizan una transformación de Wavelet en cada imagen en la pila plana focal y luego se seleccionan el coeficiente de wavelet de magnitud más grande en cada posición espacial para cada escala, j. (Ver ecuación 16)

$$WT : s(x, y; z) \rightarrow \left\{c_j(n, m; z)\right\}_j$$

$$g_j(n, m) = c_j(n, m; arg_z \max |c_j(n, m; z)|)$$

Ecuación 16

Mientras este acercamiento simple es razonablemente eficaz, es común hacer cumplir alguna forma de la coacción espacial en los coeficientes seleccionados y realizar el post-procesamiento en la imagen para reducir los efectos de toque y los artefactos en color falsos.

Como se puede apreciar este algoritmo se utiliza para el procesamiento de aplicaciones tales como la compresión de datos y cómputos rápidos. Sin embargo es más usado en la variante de cambio, es decir para el movimiento. Por esto no es óptimo para realizar Profundidad de Campo ya que puede llegar a producir cambios significativos en las imágenes.

Esta variante para crear el efecto Campo de Profundidad necesita de la consistencia espacial y la consistencia subcinta. La consistencia espacial implica que la selección de coeficiente debería estar basada en una vecindad local alrededor de cada coeficiente, mientras la consistencia subcinta implica que la selección de coeficiente debería estar basada en las tres subcintas en cada escala, j. La consistencia espacial es puesta en práctica calculando la discrepancia o la máxima sobre un 3x3 o un 5x5.

El método lo único que hace es almacenar los valores de coeficientes máximos. Esto evita la necesidad de cargar el plano focal en la memoria y los múltiples pasos que implica las comprobaciones de consistencia. Hay que destacar que este método es capaz de coger una imagen en el plano seleccionado por cada coeficiente lo cuál puede ser usado en la visualización topológica.

2.4- Círculo de Confusión

Este algoritmo es una extensión del método de post-procesamiento propuesto por Potmesil y Chakravarty. El Círculo de Confusión (CoC) es la distribución circular de luz de un punto de la imagen en un plano. Este se basa en un modelo de cámara de lente. En la figura 9 el lente se centra en un punto B en el lado izquierdo de la figura. Por lo tanto, la imagen aparece como un punto fuerte en el plano. Por otra parte, se tiene un punto A y un punto C que se encuentran fuera del foco. Las imágenes de los puntos A y C se encuentran delante y detrás de la imagen del punto B respectivamente. En este caso la imagen del plano es la pantalla.

Dado un punto X en el espacio, el diámetro del círculo de confusión de X se puede calcular como se muestra en la Figura 10. Donde *a* es la altura del objeto X desde el eje z, *d* la distancia del objeto, *s* la distancia de la pantalla y *f* la longitud focal del lente. Si X se encuentra fuera del foco, la imagen que se forma en el plano seria un círculo, que vendría siendo el círculo de confusión de X. La altura de esta X será x' y x''.

Por lo tanto el diámetro del círculo de confusión de X se calcula como se muestra en la ecuación 17:

$$
\begin{aligned}
CoC\ diameter &= |x'' - x'| \\
&= \left| a\left(\frac{s}{f} - 1\right) - a\left(\frac{s}{d}\right) \right| \\
&= \left| \left[s\left(\frac{1}{f} - \frac{1}{d}\right) - 1 \right] \right|
\end{aligned}
$$

Ecuación 17

En los gráficos que se muestran en las figuras 11, 12, 13, y 14 se observa la relación que existe entre el diámetro del círculo de confusión y las cuatro variables (*a, s, f, d*).

Puesto que una imagen perfectamente nítida sólo se produce cuando el diámetro del círculo de confusión es igual a cero, se tiene que: *d = s * f / (s - f),* donde el valor de «*d*» es el centro de la cámara y aparece un punto fuerte sólo si se encuentra a una distancia *d* de la cámara. Sin embargo, puesto que cada píxel en una pantalla no tiene un tamaño de cero, un punto fuerte aparece en la pantalla, siempre y cuando su diámetro sea mayor o igual al tamaño de un píxel. Así cuando se produce una distancia *d* de un punto que produce un círculo de confusión del tamaño de un píxel se refiere a la Profundidad de Campo con respecto a un determinado tamaño de píxel. Si un punto tiene su diámetro de círculo de confusión más grande que el tamaño de un píxel, aporta su luz a cada píxel en el interior del círculo de confusión.

Para la implementación de este algoritmo se puede utilizar diferentes API de programación gráfica que existen. En este caso se dará una breve explicación de cómo se realiza la implementación de este algoritmo haciendo uso del DirectX y de OpenGL.

2.4.1- Implementación del Círculo de Confusión con DirectX

Fundamentalmente cuenta con dos pasos. Se observará que en le primer paso se formara la escena, sacando el color, así como la información necesaria para desenfocar la imagen. Luego, como segundo paso se filtra la imagen obtenida en el primer paso con una variable de tamaño del núcleo del filtro para simular el círculo de confusión. Un factor de desenfoque calculado en el primer paso controla el tamaño del filtro del núcleo utilizado en la segunda pasada. Se adoptan medidas especiales para eliminar la pérdida de color de los objetos enfocados en fondos que han sido desenfocados.

2.4.1.1- Primer Paso: Rendereo de escena

La escena entera es dada por la profundidad y el factor borroso, que es usado para describir la cantidad de píxel que debería ser enturbiado, además de la escena que da el color. Esto puede ser logrado dándole a la escena los múltiples buffers en solo un tiempo. DirectX 9 tiene un rasgo útil llamado Dar Objetos Múltiples (MRT) que permite la salida simultánea de shader en múltiples buffers. La utilización de este rasgo nos da la capacidad a la salida de todos los canales de datos (el color de escena, la profundidad y el factor borroso) en nuestro primer paso. Una de las restricciones MRT contra algunos hardwares, son los requerimientos dado cualquier superficie, donde deben de tener el mismo bit de profundidad, permitiendo el empleo de formatos superficiales diferentes.

Dirigido según esta exigencia se puede escoger el formato de D3DFMT_A8R8G8B8 para la salida de color de escena y la textura de dos canales y para la profundidad y el factor borroso se escoge el formato D3DFMT_G16R16. Como se muestra en la figura 15, ambos formatos son de 32 bit por píxel, y proporciona bastante espacio para la información necesaria y con la precisión deseada.

2.4.1.2- Segundo paso: Post-Procesamiento

Durante el post-procesamiento los resultados obtenidos anteriormente son procesados y luego se enturbia la imagen en color basándose en el factor borroso calculado en el primer paso. Para enturbiar la imagen se usa un filtro clasificado que representa el círculo de confusión. Luego de haber realizado

el enturbiamiento de la imagen se realiza la filtración de imagen que consiste en dibujar en pantalla un cuadrilátero alineado.

2.4.1.2.1- Post-Procesamiento de Píxel Shader

En el post-procesamiento existe un grano con filtro el cual tiene 13 muestras, una muestra de centro y otras 12 externas. (Ver figura 16). La muestra del centro es alineado por el píxel filtrado, mientras que las externas son las muestras de píxeles cercanos. El filtro usa la prueba estocástica y las muestras externas son alineadas en el filtro según una distribución de Poisson.

El tamaño del filtro se calcula por píxel del factor borroso de la muestra del centro y el valor máximo aceptable del tamaño del círculo de confusión. En la figura 17 se muestra una relación entre el grano borroso y el grano con filtro.

El píxel shader calcula las posiciones de muestras basadas en 2D almacenadas en el $filter\ taps$[18] y el tamaño de círculo de confusión. Una vez que se calcula las posiciones de las muestras se saca un promedio de muestras para determinar el color. Si el valor esta cerca de cero es porque todas las muestras son del mismo píxel, no se procede a enturbiar la imagen y el filtro comienza a probar los píxeles vecinos hasta lograr enturbiar la imagen. Todas las imágenes son probadas con la filtración de D3DTEXF_LINEAR. La utilización de este filtro no es muy exacta en los bordes de los objetos, donde la profundidad podría cambiar bruscamente, sin embargo esto produce mejor calidad en las imágenes.

Uno de los principales problemas asociados a los métodos post-filtración es el escape de los colores agudos del objeto en los fondos borrosos. Esto ocurre porque el filtro para el fondo borroso prueba el color agudo del objeto en las cercanías debido al tamaño grande del filtro. Para solucionar este problema, se desechan las muestras externas que pueden contribuir al escape según los criterios siguientes: si la muestra externa está en el foco y está delante de la muestra de centro borroso, esto no debe contribuir al color.

2.4.1.2.2- Sombreador de Vértices

El sombreador del vértice para el rendereo de la escena es un sombreador de vértice normal con una pequeña adición, sobrepone la profundidad de la escena en el espacio de la cámara. Este valor de profundidad es posteriormente usado en el sombreador de píxeles para calcular.

2.4.2- Implementación del Círculo de Confusión con OpenGL

En OpenGL al igual que en DirectX utiliza un modelo de cámara de agujero. No hay ningún lente y el tamaño de la abertura es igual a cero. Cada punto del objeto tiene exactamente un camino que va a través del agujero y luego se proyecta en el plano de visualización (Ver Figura 18). Como resultado, la imagen es perfectamente nítida en todo el plano de visualización.

Para aplicar el efecto Campo de Profundidad es necesario un nuevo modelo de cámara de lente que apoye el tamaño de abertura y la longitud focal del lente. En este nuevo modelo de cámara se define una apertura que permite la entrada de la luz (Ver Figura 19). Es bueno destacar que con una mayor abertura se obtendrá un mayor círculo de confusión. Para determinar el círculo de confusión se determina una variable *a* que representa la abertura. Este modelo de cámara es utilizado para calcular el tamaño del círculo de confusión de cada píxel. Es importante saber que al utilizar este modelo de cámara la posición y el tamaño del objeto no cambian.

Luego de saber como es el trabajo con el modelo de cámara utilizado en OpenGL es bueno conocer como se implementa este algoritmo. El mismo consta de tres pasos fundamentales.

En primer lugar, lee la información de los buffers de cuadro generado por el modelo ordinario de OpenGL, que incluye el color y el valor de profundidad de cada píxel. En segundo lugar, se aplica la ecuación del círculo de confusión con los parámetros dado el modelo de cámara para cada píxel. Luego el color de cada píxel se distribuye a los píxeles vecinos en el marco de su diámetro del círculo de confusión calculado y los datos se acumulan en el buffer. En el tercer paso el buffer original es sustituido por un nuevo buffer.

Agregar el efecto Campo de Profundidad en un programa de OpenGL no requiere de un cambio en los códigos. Para la implementación de este efecto con OpenGL solo se añaden pasos adicionales en el momento de visualización. Con más precisión, esto procesa la imagen que es resultado del modelo de OpenGL en tres pasos mencionados antes, y sustituye la imagen por uno filtrado. A continuación se muestran los pasos de una función de visualización OpenGL que implementa el efecto Campo de Profundidad.

- Se usa rutinas de OpenGL para el rendereo de la escena en el $frame\ buffer$[19].
- Se cambia la proyección ortogonal a 2D.
- Se lee el color del buffer local *fb1* y la profundidad del buffer local *db* usando la función *glReadPixels ()* de OpenGL.
- Se convierten los valores normalizados de profundidad en z a los valores del objeto espacial en z por cada pixel en el *db.* Este paso es fundamental ya que el cálculo del círculo de confusión se basa en el objeto espacial. El valor de profundidad de un píxel debería ser la distancia de unidad entre el píxel y el lente.
- Inicializar el buffer secundario *fb2.* La nueva imagen está formada por la acumulación de luz en este nuevo buffer.
- Se determina el núcleo del filtro: En el se realiza el cálculo del círculo de confusión y se aplica la distribución de cada píxel.
- Sustituir la nueva trama por la del buffer *fb2.* Esto se realiza a través de la función *glDrawPixels ()*.

2.5- Efecto blur

La simulación visual de blur se introdujo a principios de que se introdujera en el campo de la Informática la Computación Gráfica para mejorar el aspecto foto realístico de imágenes sintéticas. Potmesil y Chakravarty [1981] fueron los primeros en proponer la simulación de un lente óptico para simular Profundidad de Campo mediante el efecto del blur. Este efecto enturbia los objetos que se encuentra delante o detrás del foco. Este foco es asociado con una distancia focal.

2.5.1- Uso de un modelo de cámara de lente

Aquí se procede a determinar el diámetro de círculo de confusión (Ver Ecuación 18):

$$DC_0C_d = \left|\frac{D * f * (f_d - z)}{f_d * (z - f)}\right|$$

Ecuación 18

Donde:

D: Diámetro del lente

f: Longitud focal del lente

f_d: Distancia focal

z: Profundidad del punto

2.5.2- Cálculo automático de la distancia focal

Una manera óptima de determinar en tiempo real la distancia focal es mediante el uso de un sistema de seguimiento. Sin embargo los dispositivos que se utilizan para esto son bastante caros, complejos y no están disponibles en el mercado. Debido a esto se propone determinar la zona de concentración que consiste en un cuadrado situado en el centro de la pantalla que es donde preferentemente el usuario deberá enfocarse.

La profundidad de cada píxel de la zona de concentración se calcula utilizando un buffer auxiliar. Aquí se propone utilizar una ponderación semántica de los píxel. Para hacer esto se puede añadir una descripción inicial de los objetos virtuales en un determinado peso semántico, además, una ponderación espacial modifica ligeramente el peso del píxel central. También se puede lograr mediante una función Gaussiana que da un peso de WGmax al centro y WGmin a las fronteras de la zona.

Todo esto son distintas maneras de determinar la distancia focal, pero este cálculo se realiza a través de la siguiente ecuación 19:

$$f_d = \frac{\sum_{p \in area} WS_{(p)} * WG\big(d2AC(p)\big) * depth(p)}{\sum_{p \in area} WS_{(p)} * WG\big(d2AC(p)\big)}$$

Ecuación 19

Donde:

$WS_{(p)}$: Peso semántico de píxel p

WG(x): Peso del espacio gaussiano de la distancia x

$d2AC(p)$: La distancia de p para el centro de la zona de concentración

2.5.3- Simulación del fenómeno de alojamiento

Para determinar el enfoque en la simulación del fenómeno de alojamiento se utiliza un filtro temporal para el cálculo de la distancia focal final, mediante el filtrado $lowpass^{20}$ dado por la ecuación 20:

$$\overline{f_d}(n) = \left(f_d(n) + \frac{\tau}{Te}\overline{f_d}(n-1)\right)\frac{1}{1+\frac{\tau}{Te}}$$

Ecuación 20

Donde:

T_e: Período de muestreo

$\overline{f_d}\,(n)$: Filtrado focal de distancia n

$f_d(n)$: Distancia focal dada por el autofoco

El valor de τ se determina como se muestra en la ecuación 21:

$$\tau = (\pi * f_c)/2$$

Ecuación 21

Donde:

f_c: Frecuencia de corte

2.5.4- Escena de blur periférico

El blur periférico simula el hecho de la disminución de la nitidez de los objetos percibidos visualmente cuando estos se encuentran en la periferia del campo de visión del ojo humano. Este efecto desdibuja progresivamente los píxeles que están situados a cierta distancia del centro de la zona de concentración. El principal objetivo de este efecto es incitar al usuario a que mire al centro de la pantalla. El cálculo del blur periférico está dado por la ecuación 22:

$$DC_0C_p = \left(\sqrt{\frac{1}{z * p}} - 1\right)^n$$

Ecuación 22

Donde:

z: Dirección de la cámara

p: Dirección normalizada del píxel en la cámara

2.5.5- Cálculo final del blur visual

Una vez que es calculado el valor periférico y el blur es obtenido, se procede a calcular la cantidad total de blur para cada píxel, es decir, el diámetro final de su círculo de confusión DCoCf. (Ver Ecuación 23):

$$DC_0C_f = D_{max} * \min(1, DC_0C_d + DC_0C_p)$$

Ecuación 23

Donde:

D_{max}: Cantidad máxima de blur

DC_0C_d: Diámetro normalizado del círculo de confusión

DC_0C_p: Diámetro normalizado del blur periférico

El blur se puede aplicar a la imagen mediante la mezcla de cada píxel del color con los colores de los píxeles que se encuentran dentro de su círculo de confusión, de acuerdo con una distribución de Poisson. Para lograr el color se usa la técnica de recolección.

CAPÍTULO 3: PROPUESTA DE LA SOLUCIÓN TÉCNICA

Introducción

En el presente capítulo se propone la solución técnica más óptima para crear el efecto Campo de Profundidad. Como se podrá apreciar se muestra por mediación de una tabla una comparación de los distintos algoritmos para la realización de este efecto, en cuanto a: calidad visual, complejidad y soporte técnico que necesita cada uno de estos, no se tubo en cuenta el costo de los algoritmos explicados en el capítulo anterior porque todos son verdaderamente costosos y la buena calidad de el efecto depende del hardware que presente el ordenador con que se trabaje.

3.1- Comparación de los algoritmos

Antes de hacer una descripción de la solución técnica que se propone, es necesario realizar una comparación de estos algoritmos atendiendo a una serie de características que son fundamentales en el momento de crear el efecto Campo de Profundidad. En la tabla 3.1 se puede observar una comparación donde se podrá ver cual de estos algoritmos sería el más factible a utilizar en nuestro centro.

Algoritmo	Complejidad	Calidad Visual	Factibilidad
Blurring con filtro Gaussiono separable	Lineal	Buena	Factible
Rendereo por punto	Lineal	Buena	Factible
Círculo de Confusión	Lineal	Muy buena	Factible
Efecto Blur	Lineal	Bastante buena	Factible
Over-complete Discrete Wavelet Transform	Lineal	Buena	Factible

Tabla 3.1: Comparando algoritmos

Como se explicó anteriormente en la tabla 3.1 se observa una comparación de los 5 algoritmos estudiados. Estos algoritmos básicamente tienen una complejidad lineal(O(n)), pues la complejidad de estos algoritmos depende de las pasadas de render necesarias para crear este efecto.

Debido a que este trabajo esta basado en la mejora de la visualización de las aplicaciones que se desarrollen con entornos virtuales, los métodos también se compararon con respecto a la calidad visual, obteniendo como resultado que estos métodos le aportan una mayor calidad visual y realismo a los entornos virtuales.

Todos los métodos brindan una buena calidad visual, pero con el método Círculo de Confusión se obtiene una mejor calidad visual, por esto es el más utilizado para aplicar el efecto Campo de Profundidad.

Por esto se propone que se realice la implementación del algoritmo Círculo de Confusión. Muchos de los algoritmos descritos, en algún momento de su desarrollo calculan el diámetro del círculo de

confusión para posteriormente realizar procedimientos característicos de cada uno de ellos. En el siguiente epígrafe se describe como crear este efecto haciendo uso del shader.

3.2- Propuesta de algoritmos para crear efecto Campo de Profundidad

Como se explica en el capítulo 2 este algoritmo consta de dos pasos. Primeramente, con los datos de la posición, la normal y las coordenadas de textura de la imagen, se analiza cada vértice y se obtiene la posición, el color y dos valores importantes que son la profundidad y una nueva coordenada de textura. En el siguiente algoritmo se observa como se realiza esto:

Algoritmo 1: Determinar el color, la profundidad, las coordenadas de textura y la posición de cada vértice.	
Entrada:	El color, la normal y las coordenadas de cada vértice en la imagen.
Requerimientos:	Cargar la escena e inicializar shader.
Salida:	El color, la profundidad, las coordenadas de textura y la posición de cada vértice.
1: Se realiza la transformación del vértice en la escena o.vPos <- multiplicar(posición del vértice, matriz del mundo); 2: Se obtiene la posición de la cámara en la pantalla vPosWV <- multiplicar(posición del vértice, matriz vista modelo); 3: Determinar la profundidad del cada vértice o.fDepth <- posición de la cámara en el eje z; 4: Cáculo del vector director de la luz vLightDir <- normaliza(posición de la luz – Posición del vértice en la escena); 5: Normalizar la normal y determinar el color vNorm <- normaliza (vector normal); o.vColor <- producto escalar(vector normalizado, vector director de la luz) * material difuso + material ambiental; 6: Se guarda la coordenada de textura o.vTexCoord <- coordenada de textura del vértice;	

Tabla 3.2: Primer algoritmo para crear Campo de Profundidad haciendo uso del algoritmo Círculo de Confusión.

Con los valores obtenidos se procede a determinar para cada píxel de la pantalla el círculo de confusión, el blur y obtener el color y el valor de la profundidad en escala de 0 a 1. Estos dos valores son almacenados en dos buffer distintos debido a que se trabaja con una textura 2D.

Algoritmo 2: Determinar el color y la profundidad para cada pixel.	
Entrada:	El color, la profundidad, las coordenadas de textura y la posición de cada vértice.
Requerimientos:	Obtener el valor del vértice
Salida:	Profundidad y color de cada pixel.
1: Calculo del color con la textura en 2D o.vColor = color del vértice * textura en 2D(muestra de textura, coordenadas de textura del vértice); 2: Se calcula el circulo de confusión pixCoC = valor absoluto(distancia del lente * longitud focal * (Zfocus – profundidad del vértice) / (Zfocus * (profundidad del vértice - longitud focal))); 3: Se calcula el blur en escala de 0 - 1 blur = saturate(circulo de confusión * escala / valor máximo del circulo de confusión); 4: Se obtiene el valor de la profundidad en escala de 0 - 1 o.vDoF = float4(profundidad del vértice / rango de la escena, blur, 0, 0);	

Tabla 3.3: Segundo algoritmo para crear Campo de Profundidad haciendo uso del algoritmo Círculo de Confusión.

Después de explicar como se realiza la primera pasada de render que tiene el algoritmo se procede a realizar la segunda pasada. En esta es donde se determina el color final de cada objeto según su posición en la pantalla. Para determinar este valor se calcula el círculo de confusión pero primero se obtiene una nueva coordenada de textura y la posición de cada vértice. Para esto se hace uso de los valores de posición y las coordenadas de texturas obtenidas en el paso 1.

Algoritmo 3: Determinar nuevos valores de coordenadas de textura y posición de los vértices.	
Entrada:	Posición y coordenadas de texturas obtenidas en el algoritmo 1.
Requerimientos:	Obtener los valores en el primer paso.
Salida:	Coordenada de textura y posición de cada vértice.
1: Obtener una nueva posición de los vértices. o.vPos <- posición obtenida en el paso 1 * viewportScale + viewportBias; 2: Obtener nuevos valores de coordenadas de textura. o. vTexCoord <- coordenadas de texturas obtenida en el paso 1	

Tabla 3.4: Tercer algoritmo para crear Campo de Profundidad haciendo uso del algoritmo Círculo de Confusión.

Para obtener el valor final de la profundidad se procede a crear un arreglo de 12 elementos que guarda los valores de filtrado. Para esto se toma un filtro que tiene 13 muestras, una muestra de centro y las otras 12 son muestras externas. Para determinar el color final se calcula el color de cada muestra y se determina el promedio de todas las muestras del filtro. Estos valores son hallados teniendo la coordenada de textura obtenida en el paso anterior, además de que se realiza esta operación para cada piel de la escena.

Algoritmo 4: Determinar el color final.	
Entrada:	Coordenadas de texturas obtenidas en el algoritmo 3.
Requerimientos:	Realizar todos los pasos anteriores.
Salida:	Color final.

1: Se determina la suma de los colores de cada pixel.
colorSum <- textura en 2D(color de muestra de la escena, textura de coordenada obtenida en el algoritmo 3);
2: Se determina la profundidad del centro de la muestra
centerDepthBlur <- textura en 2D(profundidad de la muestra, textura de coordenada obtenida en el algoritmo 3);
3: Calculo del tamaño del círculo de confusión
sizeCoC <- profundidad de la muestra del centro en el eje y * el valor máximo del circulo de confusión;
4: for(desde i=0 hasta arreglo creado de 12 elementos)
5: Cálculo de las coordenadas de la muestra
tapCoord <- textura de coordenada obtenida en el algoritmo 3 + valor que esta guardado en la posición del arreglo * tamaño del círculo de confusión;
6: Cálculo del color del filter tap de la muestra
tapColor <- textura en 2D(color de la muestra de la escena, coordenadas de la muestra);
7: Cálculo de la profundidad de la muestra
tapDepthBlur <- textura en 2D (profundidad de la muestra, coordenadas de la muestra);
8: Cálculo de la contribución basado en la profundidad
tapContribution <- (profundidad de la muestra en el eje x > profundidad del centro de la muestra en el eje x)?
1.0f : tapDepthBlur.y;
9: Suma de los valores de los colores:

```
   colorSum += color del filter tap de la muestra * contribución;
10: Cálculo de la contribución total
   totalContribution += contribución;
11: end for
12: Cálculo del color final
   finalColor <- suma de los valores de los colores / contribución total;
```

Tabla 3.5: Algoritmo final para crear Campo de Profundidad haciendo uso del algoritmo Círculo de Confusión.

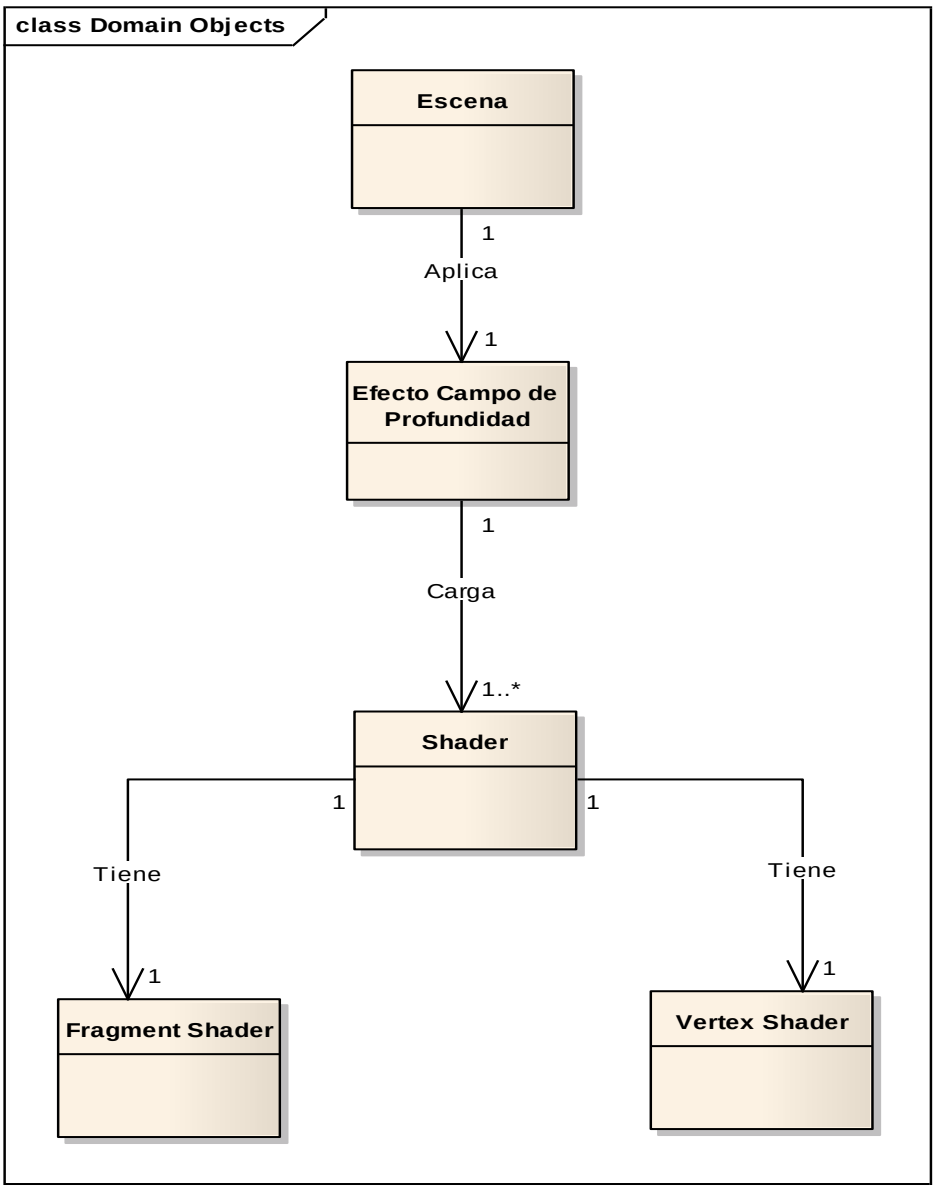
De esta manera se obtiene el color final para determinar la Profundidad de Campo en entornos virtuales haciendo uso de shader. Para un mejor entendimiento de cómo crear este efecto, hay que concentrarse y tener bien en cuenta algunos aspectos que son fundamentales. En el siguiente epígrafe se hablará acerca de los lenguajes de programación y de shader que se pueden utilizar.

3.3- Descripción

Para dar la solución se puede utilizar cualquier lenguaje de programación como: C#, java, C++ entre otros. Se propone que se haga uso del C++, debido al alto nivel que tiene y su capacidad de mantener su portabilidad hacia otras plataformas. Además, es muy importante no olvidar la necesidad de tener una tarjeta gráfica que sea capaz de soportar las características que tendrá la aplicación a la que se le realice este efecto. Por ejemplo, pudiera ser una tarjeta gráfica ATI RADEON XPRESS 200M Series 256Mb de Video. Como se ha hecho referencia en capítulos anteriores, para la solución técnica es necesaria la utilización de shader. Para programar el shader existen varios lenguajes entre los que se pueden encontrar: HLSL, GLSL y CG, para esto se propone que se implemente el efecto haciendo uso de CG, debido a que es un lenguaje de gráficos de alto nivel, resulta muy atractivo, pues permite desarrollar efectos gráficos, así como impactantes aplicaciones de larga duración. Además de que el uso de CG aumenta la productividad de los programadores y reduce el tiempo de desarrollo de juegos de mayor complejidad gráfica. Para el desarrollo de la implementación de este efecto se pueden utilizar cualquiera de las API existente, DirectX u OpenGL. Para este caso en particular se propone que se trabajan con OpenGL. Con el uso de esta API la aplicación que desarrolle este efecto podrá usarla en diversas plataformas, además para software libre.

Para mejor entendimiento de cómo desarrollar la solución para crear el efecto Campo de Profundidad, se presenta en el siguiente epígrafe un modelo de dominio y un glosario de términos donde se dará una breve explicación de lo que representa cada concepto incluido en el modelo de dominio, el cual servirá de ayuda para los programadores.

3.3- Modelo Dominio



3.1.1- Elementos del Dominio

Shader: Interpreta y manipula los ficheros con el código para su posterior uso.

Efecto Campo de Profundidad: Efecto que simula una cámara que enfoca un objeto en una escena 3D mientras los demás objetos de esa escena quedan desenfocados. Se le aplica a los objetos para que su presencia en la escena sea lo más parecido a la vida real.

Escena: Imagen en 3D a la que se le aplica el efecto Campo de Profundidad.

Fragment Shader: Programa de sombreado, normalmente ejecutado en la unidad de procesamiento gráfico. Sirve para la manipulación de cada uno de los pixeles en la escena individualmente, especialmente sobre las imágenes y así lograr mayor objetividad y realismo.

Vertex Shader: Herramienta que trabaja con la estructura de vértices de los modelos tridimensionales. Es una función de proceso gráfico, es decir, añade efectos especiales a los objetos de una escena 3D. Se encarga de las operaciones de vértices, es decir, iluminación, texturas y transformaciones como se planteó anteriormente. Aquí también se trabaja individualmente con cada vértice.

CONCLUSIONES

Para la realización de esta tesis se trazaron varios objetivos a los cuales se le fue dando respuesta con el desarrollo del mismo. Primeramente fue necesario alimentar nuestros estudios relacionados con los algoritmos, tendencias y tecnologías que existen a nivel mundial para crear el efecto Campo de Profundidad muy tierno todavía la información en nuestra Universidad acerca de este efecto. Con la realización del mismo se pudieron definir algunos conceptos fundamentales que se deben de tener presentes al implementar estos algoritmos.

También se analizaron los avances que existen en correspondencia con el hardware y las tarjetas gráficas que es las que se manipulan para lograr este efecto, muy importante en el mundo actual a la hora de hacer aplicaciones de Realidad Virtual. Además se pudo realizar una descripción de 5 de estos algoritmos y llegar a conclusiones muy importantes. Se propone la utilización de uno de estos algoritmo que consideramos que es una solución factible para poder aplicar este efecto en entornos virtuales y proyectos que se realicen en nuestra Facultad y la Universidad.

Dándole respuesta a estos objetivos se puede decir que se refutó el objetivo general de esta tesis, que era hacer una comparación de los algoritmos existentes a nivel global para crear el efecto Campo de Profundidad. Para darle cumplimiento final a este objetivo general se realizó una tabla comparativa de estos algoritmos donde se tiene en cuenta algunos aspectos importantes.

RECOMENDACIONES

Por la importancia que tiene este efecto en el mundo de la Realidad Virtual se recomienda:

- Continuar en el estudio más a fondo de estos algoritmos para crear este efecto que tanto realismo le da a las escenas.

- Realizar la implementación de dicho efecto.

- Luego de realizar la implementación, es recomendable que se aplique a los proyectos productivos de la facultad y de la Universidad que trabajen con entornos virtuales.

BIBLIOGRAFÍA

[1]. **Arevalo, Carlos Fernando.** *Desarrollo de un seudolenguaje para construccion de mundos virtuales basado en VRLM y montaje de un laboratorio de Realidad Virtual basado en dispositivos de vision 3D.* 2005.

[2].Wikipedia.*Wikipedia.*[Online] [Cited: 1 30, 2088.] http://es.wikipedia.org/wiki/Profundidad_de_campo.

[3]. Wikipedia . *Wikipedia.* [Online] [Cited: 11 15, 2007.] http://en.wikipedia.org/wiki/Pinhole_camera.

[4]. Wikipedia. *Wikipedia.* [Online] [Cited: 4 2, 2008.] http://es.wikipedia.org/wiki/Shader.

[5]. Wikipedia. *Wikipedia.* [Online] [Cited: 4 2, 2008.] http://es.wikipedia.org/wiki/Vertex_Shaders.

[6]. **Guennadi Riguer, Natalya Tatarchuk, John Isidoro.** *Real-Time Depth of Field Simulation.*

[7]. **Richard Cant, Nathan Chiad, David Al-Dabass.** *New Anti-Aliasing and Depth of Field Techniques for games.* Nottingham : s.n.

[8]. **Michael Kass, Aaron Lefohn, John Owens.** *Interactive Depth of Field Using Simulated Di®usion on a GPU.*

[9]. **S´ebastien Hillaire, Anatole L´ecuyer, R´emi Cozot, G´ery Casiez.** *Depth-of-Field Blur Effects for First-Person Navigation in Virtual Environments.*

[10]. **Bradley, Derek.** *Real-Time Depth of Field Rendering.* Vancouver : s.n.

[11]. **Andrew P. Bradley, Pascal C. Bamford.** *A One-pass Extended Depth of Field Algorithm Based on the Over-complete Discrete Wavelet Transform.* St Lucia : s.n.

[12]. **Francisco Abad, Emilio Camahort, Roberto Vivó.** *Antecedentes y fundamentos de la integración de objetos sintéticos en escenas reales.* Valencia : s.n., 2002.

[13]. **Todd Jerome Kosloff, Brian A. Barsky.** *An Algorithm for Rendering Generalized Depth of Field Effects Based on Simulated Heat Diffusion.* Berkeley : s.n., 2007.

[14]. **Division, Computer Science.** *Computer Science Division.* 2004.

[15]. **Placeres, Frank Puig.** *Empleo eficiente del hadrware grafico en la iluminacion de entornos virtuales.* Ciudad de La Habana : s.n., 2006.

[16]. **Yu, Tin-Tin.** *Depth of Flied implementation with OpenGL.* Houghton : s.n.

ANEXOS

Figura 1: Imagen donde se puede apreciar la utilización de equipos que emplean realidad virtual en software para el ejército.

Figura 2: Escena del juego Crysis.

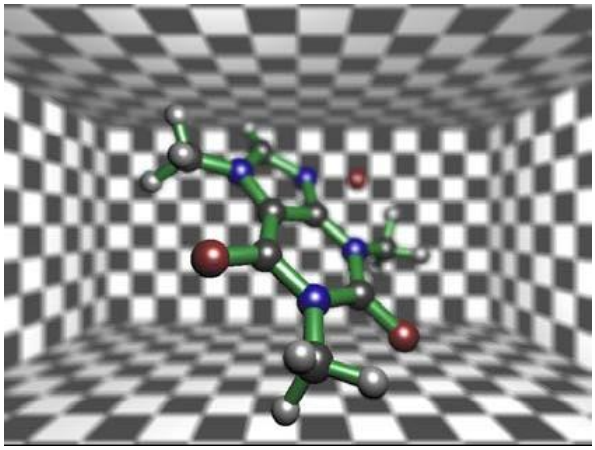


Figura 3: Imagen aplicando el efecto Campo de Profundidad.

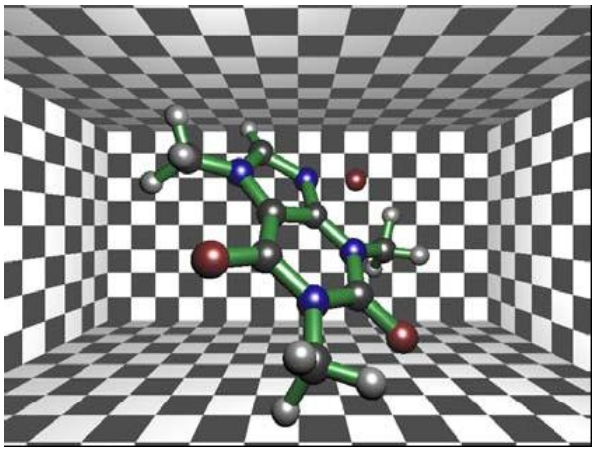


Figura 4: Imagen sin aplicar efecto de Campo de Profundidad.

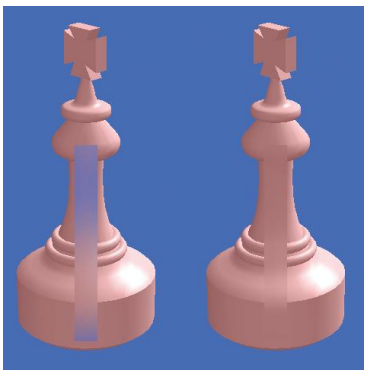


Figura 5: El objeto izquierdo muestra un ejemplo de un agujero.

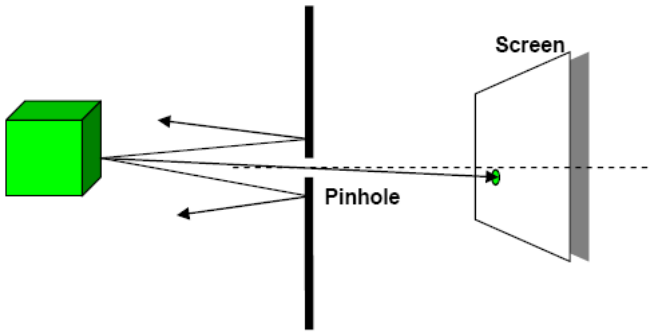


Figura 6: Modelo de cámara de agujero

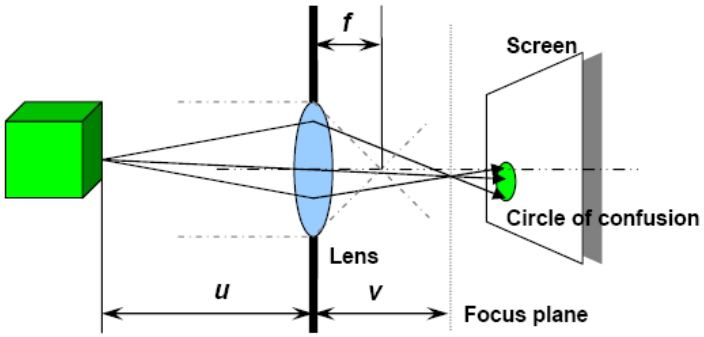


Figura 7: Modelo de cámara de tipo lente de dimensiones finitas

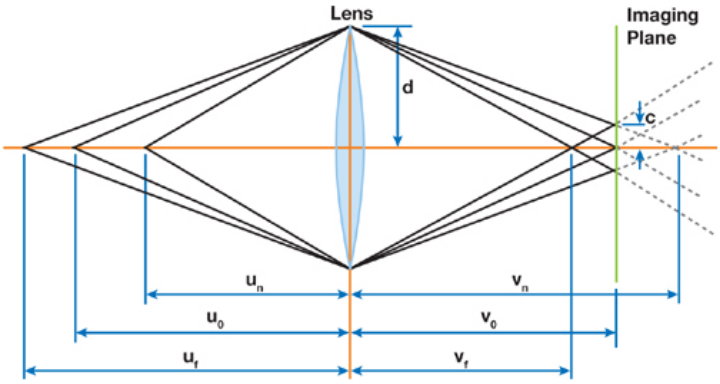


Figura 8: Ejemplo donde se muestra la geometría de un lente.

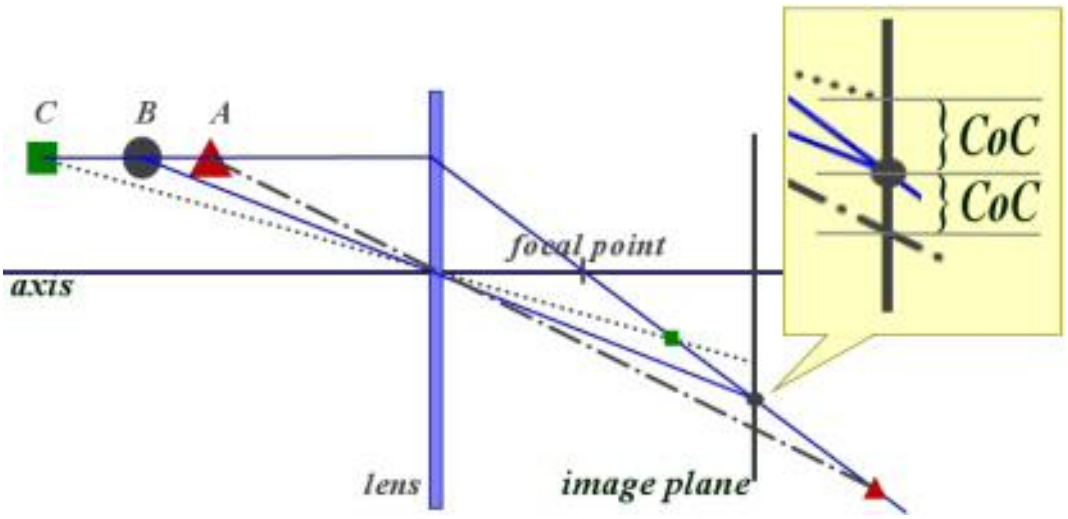


Figura 9: Formación del círculo de confusión.

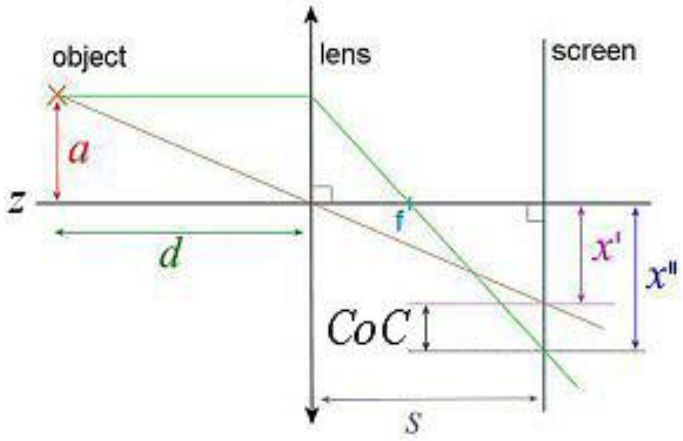


Figura 10: Calculando el círculo de confusión

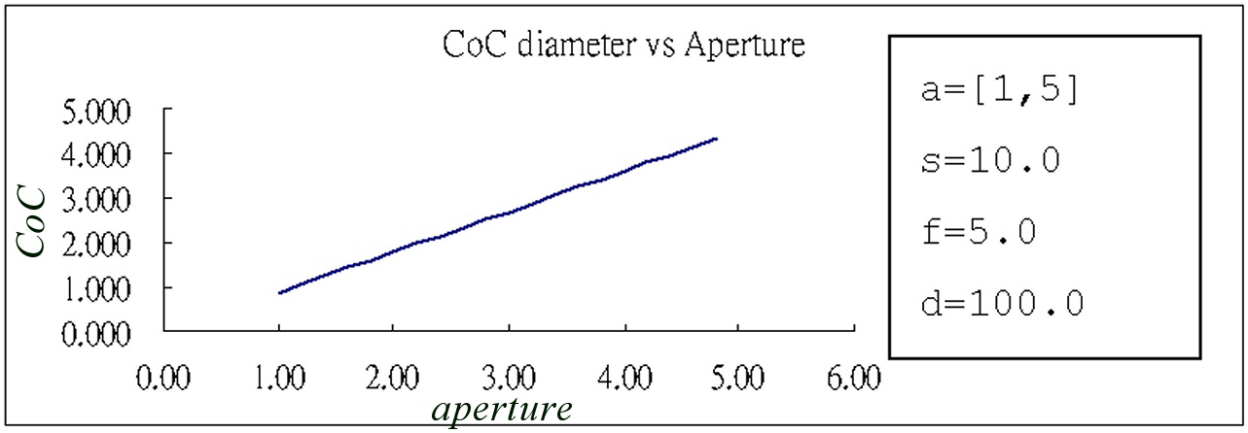


Figura 11: Relación lineal entre la diámetro del círculo de confusión y la distancia del objeto *a* al eje z, cuando una $a \in [0, 4]$.

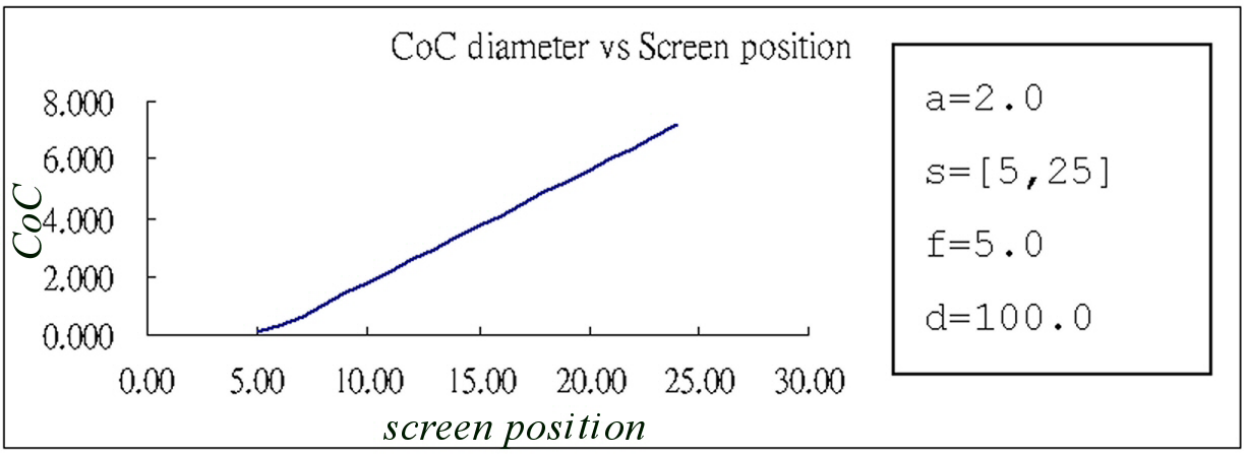


Figura 12: Relación lineal entre el diámetro del círculo de confusión y la posición *s* en la pantalla done $s \in [5, 25]$.

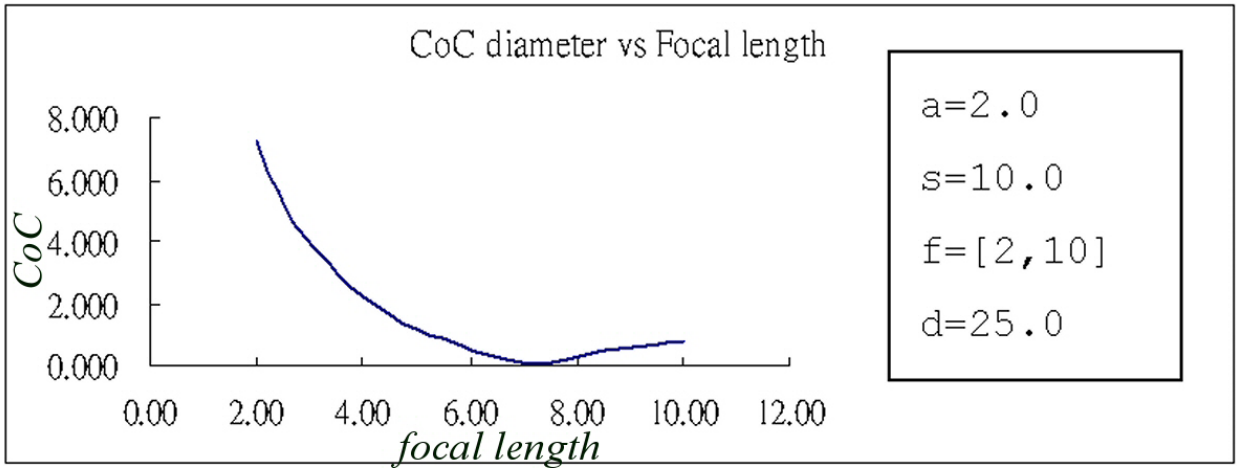


Figura 13: Muestra la no relación lineal que existe entre la diámetro del círculo de confusión y la longitud focal *f* donde $f \in [2, 10]$.

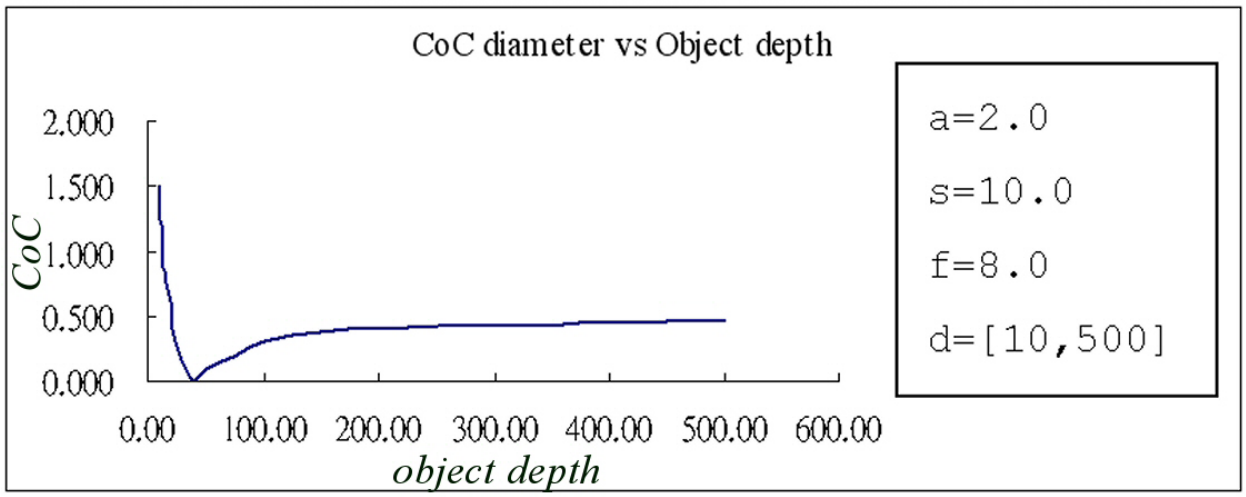
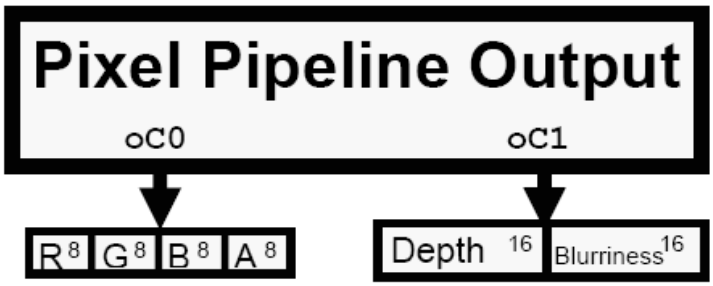
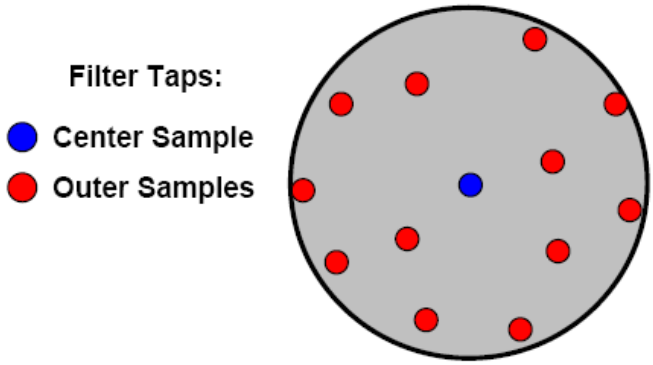


Figura 14: Muestra la no relación lineal que existe entre el diámetro del círculo de confusión y la distancia *d* del objeto donde $d \in [10, 500]$.



Figuar 15: Rendereo de escena del píxel shader



Figuar 16: Grano con filtro utilizado en el pos-procesamiento del píxel shader

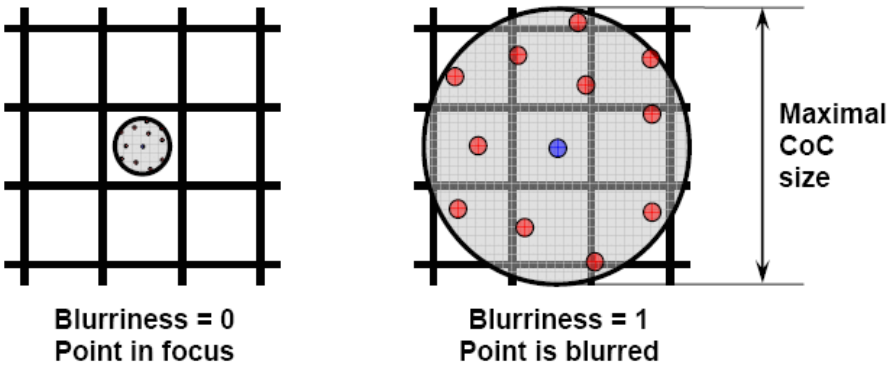


Figura 17: muestra una relación entre el grano borroso y el grano con filtro.

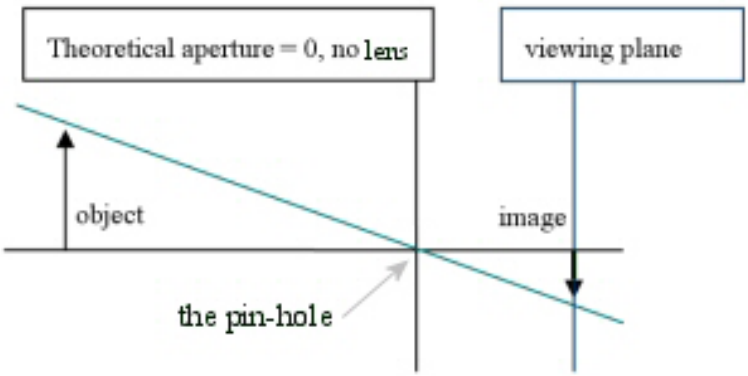


Figura 18: Modelo de cámara en OpenGL

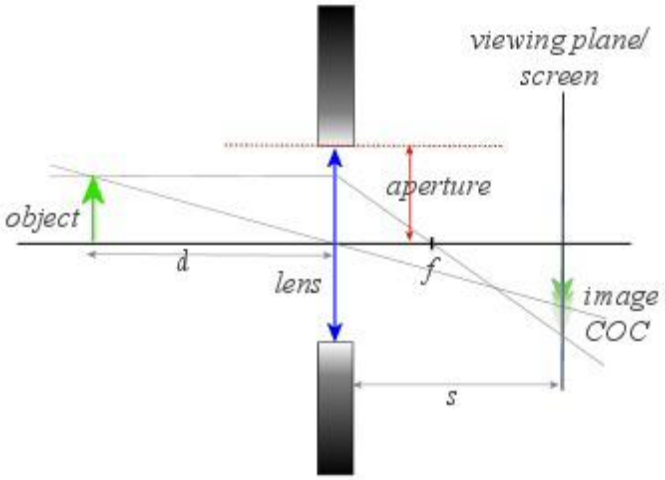


Figura 19: Modelo de cámara

Figura 20: Imagen del juego Crysis donde se puede apreciar el efecto Campo de Profundidad

Figura 21: Imagen de un demo de un juego de ajedrez al que se le aplico el efecto Campo de Profundidad.

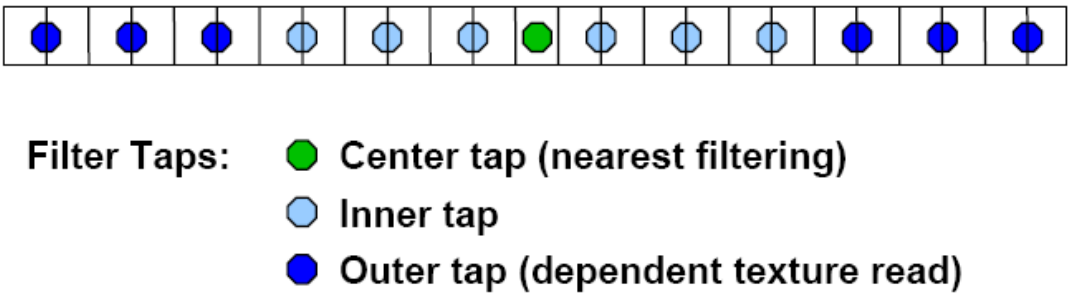


Figura 23: Muestras de n texturas.

Figura: 24: Ejemplo del juego Crysis

Figura 25: Ejemplo de Profundidad de Campo en el juego Need for Speed

GLOSARIO

***Patrones de Moiré*[1]:** Los patrones de Moiré son aquellos que se producen al superponer dos grillas distintas de líneas con un ángulo entre ellas o que tienen una separación distinta entre si. El ojo humano termina percibiendo esto como zonas difusas con franjas oscuras y claras. El nombre de Moiré viene dado por una tela textil fabricada con seda la cual produce el efecto óptico mencionado.

***Blur*[2]:** Técnica para enturbiar la imagen.

***Pipeline gráfico*[3]:** Conjunto de procedimientos para lograr obtener una aplicación gráfica.

***DirectX*[4]:** Es una colección de APIs creadas y recreadas para facilitar las complejas tareas relacionadas con multimedia, especialmente programación de juegos y vídeo en la plataforma Microsoft Windows

***Microsoft*[5]:** Es una empresa multinacional estadounidense, fundada en 1975 por Bill Gates y Paul Allen. Dedicada al sector de la informática, Microsoft desarrolla, fabrica licencia y produce software para equipos electrónicos. Siendo sus productos más usados el Sistema operativo Microsoft Windows y la suite Microsoft Office.

***OpenGL*[6]:** Es una especificación estándar que define una API multilenguaje y multiplataforma para escribir aplicaciones que produzcan gráficos 2D y 3D.

***NVidia*[7]:** NVIDIA Corporation es un fabricante estadounidense de procesadores gráficos (GPUs), chipsets, tarjeta gráficas y dispositivos para consolas (PlayStation 3).

***Direct3D*[8]:** utilizado para el procesado y la programación de gráficos en tres dimensiones (una de las características más usadas de DirectX).

***SIMPRO*[9]:** "Centro de Investigación y Desarrollo #2", empresa que ha impulsado el desarrollo de los Sistemas de Realidad Virtual.

***Filtro Gaussiano*[10]:** Filtro que se utiliza para obtener el efecto Campo de Profundidad.

***Buffer*[11]:** En informática, un buffer es una ubicación de la memoria en una computadora o en un instrumento digital reservada para el almacenamiento temporal de información digital, mientras que está esperando ser procesada.

***Downsampling*[12]** : Segundo paso que se realiza cuando se va a crear el efecto Campo de Profundidad haciendo uso del algoritmo Blurring con filtro Gaussiano Separable.

***Quad*[13]:** Cuadrado a pantalla completa que representa ¼ de la imagen original.

***Target pixel*[14]:** Objeto del píxel**.**

***Texels*[15]:** Es la unidad mínima de una textura aplicada a una superficie, usada en gráficos por computador.

***Frustum*[16]:** Es el volumen de visualización de una cámara en un punto determinado, el cual está generado a partir de una matriz de proyección en perspectiva.

***Vertex arrays*[17]:** Matriz de vértices.

***Filter taps*[18]:** Arreglo donde se van a guardar los valores de filtrado.

***Frame buffer*[19]:** Se le llama framebuffer a una categoría de dispositivos gráficos, los cuales basan su funcionamiento en representar cada uno de los píxeles de la pantalla como localidades de memoria en RAM.

***Lowpass*[20]:** Paso bajo**.**

GPU: Es un acrónimo utilizado para abreviar Graphics Processing Unit, que significa "Unidad de Procesado de Gráficos".

C++: Es un lenguaje de programación, diseñado a mediados de los años 1980, por Bjarne Stroustrup, como extensión del lenguaje de programación C.

CG: Son gráficos computarizados o realizados por ordenador, son aquellos que utilizan ordenadores tanto para generar imágenes visuales sintéticamente como para integrar o cambiar la información visual y espacial obtenida del mundo real.

1D: Primera dimensión

2D: Dos dimensiones.

3D: Tres dimensiones.

API: Application Programmer's Interface, (interfaces para programadores de aplicaciones).

GLSL: Opengl Shading Language. Lenguaje de programación de alto nivel para realizar aplicaciones gráficas y lograr mayor eficiencia y realismo.

HLSL: *High Level Shading Language*. Lenguaje de programación de alto nivel para realizar aplicaciones gráficas y lograr mayor eficiencia y realismo.

Realidad Virtual: Simulación de un medio ambiente real o imaginario que se puede experimentar visualmente en tres dimensiones.

Shader: Término que abarca las diferentes imágenes y parámetros que se pueden asignar a la superficie de un objeto geométrico en 3D.

Píxel shader: Encargado de realizar todas las operaciones a nivel de píxel.

Vertex shader: Encargados de transformar todos los vértices de la escena.

Tarjeta gráfica: Encargada de procesar los datos provenientes de la CPU y transformarlos en información comprensible y representable en un dispositivo de salida, como un monitor o televisor**.**

CPU: Unidad central de procesamiento (Central Processing Unit) o, simplemente, es el componente en una computadora digital que interpreta las instrucciones y procesa los datos contenidos en los programas de computadora.

CGI: Computer-generated imagery (Imagen generada por computadora (CGI)), es la aplicación del campo de gráficos realizados por computadora (CG, o más expresamente, gráficos 3D por computadora) para la creación, entre muchas otras cosas, de efectos especiales.

INDICE DE ECUACIONES