Universidad de las Ciencias Informáticas

”Facultad 5”

Título: “Aplicación de Shaders de Relieve a objetos 3D en entornos de Realidad Aumentada.”

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

Autor(es): Maydelis Romero Naranjo.

Julio Antonio Galván Zaldívar.

Tutor(es): Lic. Lidiexy Alonso Hernández

Co-tutor: Ing. Jaime González Campistruz

Ciudad De La Habana, Julio 2008

"Todo lo que una persona puede imaginar, otros pueden hacerlo realidad."

Julio Verne.

DECLARACIÓN DE AUTORÍA

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

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

Maydelis Romero Naranjo	Lidiexy Hernández Alonso	Julio A. Galván Zaldivar
_____	_____	_____
Firma del Autor	Firma del Tutor	Firma del Autor

DATOS DE CONTACTO

Tutor

Lic. Lidiexy Hernández Alonso (lidiexy@uci.cu)

Graduado en Ciencias de la Computación en la Universidad Central Marta Abreu, Villa Clara.

Profesor Instructor Facultad 5 con cuatro años de graduado.

1 año de experiencia en el tema de la Realidad Aumentada.

Co-tutor

Ing. Jaime González Campistruz (jgonzalezc@uci.cu)

Graduado de Ingeniería en Ciencias Informáticas en la UCI

Jefe del Proyecto de Entrenamientos Aduana.

5 años de experiencia en el tema de la Realidad Virtual.

AGRADECIMIENTOS

A la Revolución y al Comandante Fidel Castro por habernos dado la oportunidad de formar parte de este proyecto.

A la facultad por habernos demostrado que solo los Campeones creen en si mismo aunque nadie más crea en ellos.

A todos los que de una forma u otra fueron parte de nuestras vidas en la universidad.

Maydelis y Julio.

A mi madre por ser luz y guía durante todos estos años en este camino tan empedrado que es la vida.

A mi padre por ser mi sostén, mi soporte, mi bastón cuando he tropezado.

A mis hermanos por ser una de las cosas que más quiero en el mundo.

A mis abuelos por estar siempre a mi lado brindándome apoyo, cariño, satisfacciones y siempre ofreciéndome consejos útiles para la vida.

A mi gran familia por ayudarme y estar presente durante estos cinco años.

A mis amigos (Yanet, Yeilin, Rita, William, Alejandro, Julio, Macia, Gelson , Duniel), que más que amigos son mis hermanos, a ellos por soportar mis buenos y malos días, por darme confianza, por criticar y valorar mis acciones, por estar siempre a mi lado, más en los malos que en los buenos momentos, por ser otra de las cosas que mas quiero.

A mi tutor y co-tutor por apoyarnos y encaminarnos.

A todas aquellas personas que con su granito de arena posibilitaron que se hiciera realidad este sueño.

Maydelis.

A mis padres y a mi hermana por confiar en mí, apoyarme y guiarme en los momentos más difíciles de mi vida.

A mi familia en Holguín por haber confiado en mí.

A toda mi familia que vive en La Habana por haberme ayudado todos estos años.

A mis compañeros de los diferentes grupos por los que pase.

A la Comunidad del Anillo (William, Ernesto, Elieser, Alejandro, Gelson, Duniel, Luis Alberto, Amaury, Dairai, Mailyn, Yeilin, Maydelis, Dayami) por ayudarme y apoyarme en todo momento.

A mi tutor (Lidiexy) y mi co-tutor (Jaime) por haberme ayudado sin importar hora y lugar.

A mis amigos de siempre (Manuel Macías, Tamara Martínez, Alejandro Notario, Pedro Jiménez, Adonis Muñoz, Alexander Quesada, Pedro Salas) por ser simplemente "mis amigos".

A Gerandi por haberlo molestado tanto.

A todos los que de una forma u otra formaron parte de mi vida en la universidad.

Julio Antonio

DEDICATORIA

A todas las personas que nos ayudaron a ser mejores cada día.

A todas las personas que crean en lo imposible.

Maydelis y Julio.

A mi madre, A mi padre A mis hermanos.

A mi familia, A mis amigos.

Maydelis

A mi abuelo (Manuel Galván), que en paz descanse,

por enseñarme a levantarme ante las adversidades,

por toda la confianza que depositó en mí,

por apoyarme y ayudarme a cumplir mí meta.

Julio Antonio.

RESUMEN

Dada la insuficiencia de fotorealismo que existe en las escenas de Realidad Aumentada se ha hecho un estudio de las técnicas de gráficos computacionales 3D utilizadas para la creación de Shaders de Relieve con el fin de lograr escenas con un nivel de credulidad que el usuario las pueda confundir con la realidad.

Debido a la diversidad de las técnicas de gráficos computacionales 3D que se pueden utilizar, se hizo un análisis de sus características, ventajas y desventajas que estas presentan, las herramientas a utilizar y los lenguajes de programación para implementar las mismas.

Un Shader es un conjunto de instrucciones gráficas capaces de brindar efectos a los objetos representados. (Shader). Un Shader de Relieve ofrece una mejor credulidad a los objetos 3D, logrando así una semejanza con el mundo real.

Con este trabajo se pretende realizar una investigación con el objetivo de profundizar en el desarrollo de los Shaders de Relieve, utilizando dentro de las técnicas de gráficos computacionales 3D, la más óptima, ya que estas constituyen el eslabón fundamental para ofrecer el realismo necesario a los objetos 3D que actúan en los entornos de Realidad Aumentada.

PALABRAS CLAVES

Fotorealismo, Realidad Aumentada, Realidad Virtual, Shader

TABLA DE CONTENIDO

LISTADO DE IMÁGENES

LISTADO DE TABLAS

INTRODUCCIÓN

Los constantes cambios surgidos en las tecnologías se encuentran muy avanzados exponiéndonos a una serie de transformaciones, las que el hombre ha tenido que subdividir en pequeñas ramas para su mejor comprensión, estudio y utilización. Entre estas existe una que combina los datos virtuales con el ambiente real observado por el usuario, o sea, la Realidad Aumentada (RA).

Esta nueva tecnología está comenzando su inserción en muchos campos tanto científicos como sociales, teniendo varias aplicaciones fundamentalmente en las simulaciones de entornos reales con objetos 3D, encontrando en su desarrollo diferentes problemas, entre los cuales, se podría catalogar como el mayor problema, la insuficiencia de fotorealismo en las escenas de RA.

En nuestra universidad, desde hace algún tiempo se está trabajando en la inserción de la RA teniendo como base el trabajo con la Realidad Virtual (RV) desempeñado en la facultad 5 conjunto a colaboradores de otras instituciones del país. Se han obtenido resultados en proyectos de RV como el Simulador de Conducción de Auto, Simulador Quirúrgico, entre otros, desarrollando sistemas informáticos que permiten una simulación cercana a la "realidad", pero carecen del suficiente realismo necesario para la inmersión de los usuarios en los entornos generados por computadoras. Esto último se resuelve con el uso de un conjunto de instrucciones capaces de ser ejecutadas por el procesador gráfico y en especial la conformación de relieves que acerca en lo posible al mayor reto que tiene la tecnología de la RA como parte de la RV. La inclusión de la RA en la solución a problemáticas actuales puede constituir uno de los mayores aportes dentro de la informática, ayudando así a nuestro país en el desarrollo de la Industria del Software Cubano.

La interactividad de los gráficos por computadoras con el medio natural es el mayor reto que existe actualmente en el campo de la informática donde se han desarrollado disímiles herramientas para ello, siendo estas antecedidas por el gráfico computacional con una mínima

interactividad con los usuarios, dándole paso a una serie de herramientas que por sus dimensiones y posibilidades no les eran muy factibles al sistema ni a los usuarios, como por ejemplo el Proyecto Mago de Oz, imposibilitando también la movilidad, la toma de decisiones, entre otros aspectos, pero aun continúan las deficiencias en este sentido. (Neoteo, 2007)

Con estos cambios y logros que se han alcanzado no es suficiente para sentar las bases; la idea fundamental es hacer estos entornos tan parecidos a la realidad que puedan confundirse y donde el usuario sea el principal protagonista. Para servirse de las particularidades que trae consigo la RA, sería factible que se usara sin la carencia de realismo para las escenas en entornos 3D.

Existen pocos software que den solución a la falta de realismo en las escenas de RA, aunque esto no quiere decir que no se esté trabajando en ellos, pues existen Shaders de Iluminación y de Sombras que ofrecen un ambiente más real y creíble en las investigaciones actuales, pero aún así no logran la ecuanimidad que se desea, por lo que este trabajo pretende investigar acerca de los Shaders de Relieve para brindar a los entornos de RA una mejor visualización e interacción.

Para mantenernos ajustados a los cambios efectuados en las nuevas tecnologías nuestro país ha decidido insertarse en el campo de la RA y para ello es necesario manejarla acorde a las necesidades y condiciones de este. Un mal que ataña a la RA en la actualidad es la carencia de realismo por medio de lo antes expuesto, de ahí la necesidad de la investigación acerca de diferentes Shaders, y principalmente de un Shader de Relieve ofreciéndole una mejor credulidad a los objetos 3D y logrando así crear una semejanza con el mundo real y por ende una mejor inmersión de los usuarios en este entorno.

Para ofrecer mayor realismo a las escenas que conforman el entorno de RA se pretende investigar acerca de los Shaders de Relieve, donde los efectos visuales conforman el objeto de estudio y el trabajo con relieve, el campo de acción.

Expuesto el problema presentado anteriormente este trabajo se traza como objetivo general y fundamental investigar sobre los Shaders de Relieve y su aplicación en objetos 3D ubicados

dentro de las escenas que conforman un entorno de RA, para que estas escenas se conviertan lo más real posible donde el usuario pueda interactuar cómodamente. Se ha considerado que, con la aplicación de un Shader de Relieve a los objetos 3D que interactúan en las escenas que participan en el entorno de RA, permita ofrecer a los objetos 3D mayor realismo y credulidad contribuyendo a que estas escenas sean más reales y obtengan mayor aceptación por parte de los usuarios.

Para el cumplimiento del objetivo de este trabajo se han planteado algunas tareas investigativas:

- Analizar los problemas en la RA.
- Resumir las potencialidades de los lenguajes de programación para la creación de Shader.
- Estudiar las técnicas de gráficos computacionales 3D para la aplicación de los Shaders de Relieve.
- Investigar las aplicaciones para trabajar los Shaders.
- Conocer las potencialidades de la tarjeta gráfica a utilizar.

La creación de la RA ha posibilitado la inserción de esta en diferentes dominios, desarrollando aportes prácticos, los cuales sirven de guía para el desarrollo de los Sistemas de RA.

El mayor reto de la RA es lograr combinar objetos del mundo real con objetos virtuales dentro de un entorno aumentado, que creen la ilusión para el usuario de que los objetos virtuales o generados por computadoras se encuentran de manera coherente y exacta en el entorno real que se visualiza. En fin que las coordenadas registradas del objeto virtual se encuentren en correspondencia con su posición en el entorno virtual que a su vez tiene relación directa con el mundo real. Los objetos virtuales deben actuar recíprocamente con el entorno real en la escena de una manera natural y coherente. La meta es crear un sistema donde el usuario no pueda

percibir la diferencia entre el mundo real y el aumento virtual. Al usuario de este sistema le parecería que está trabajando en un solo ambiente real. (Alonso Hernández, y otros, 2007)

Existen varios proyectos en los que se ha ido trabajando, entre los cuales se encuentra un sistema en el área de la medicina, que crea la fusión y el registro correcto de los datos de un estudio imaginario de pre-operatorio hacia la cabeza del paciente. El teatro que opera reforzaría la actuación del cirujano y posiblemente eliminaría lo que necesite para cualquier otro adorno de la calibración durante el procedimiento. (Vallino, 1998)

Uno de estos ejemplos lo podemos observar en la siguiente figura.

Ilustración 1: Ejemplo del dominio médico en la RA. (Vallino, 1998)

Otro de los trabajos, pero este en el dominio de la Arquitectura, es el que se considera el guión de pintado de la ciudad que se quiere visualizar donde el paisaje parecerá como cuando una nueva pasarela es construida. Ellos irían al sitio propuesto y (poniéndose un dispositivo de despliegue de realidad aumentado) ven el área con el nuevo puente unido en su vista del paisaje.

Si se necesitaran los ajustes, ellos podrían realizarse directamente en el modelo visual del nuevo puente. (Vallino, 1998)

La siguiente figura muestra como se pone de manifiesto este ejemplo.

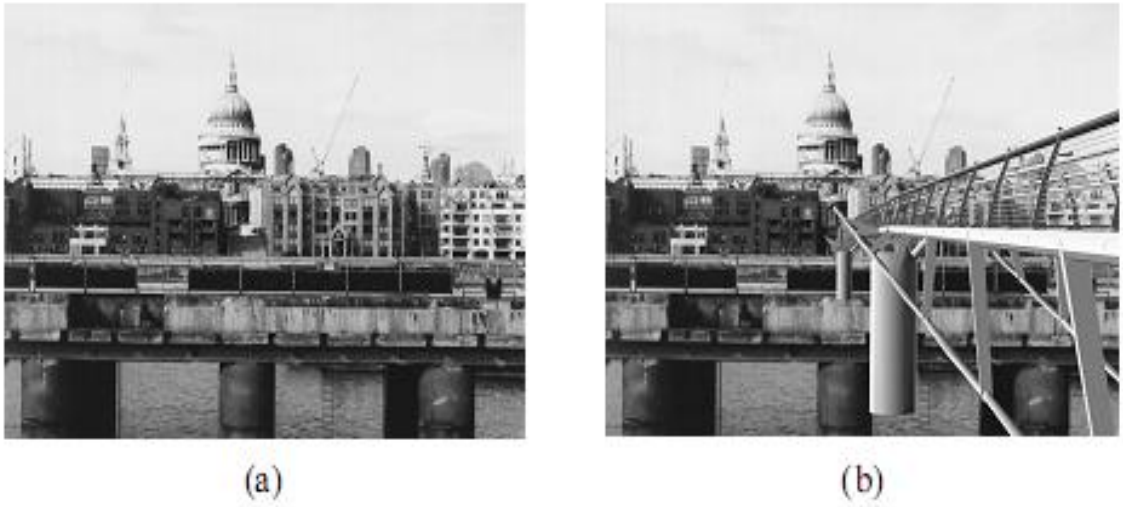

Ilustración 2. Puesta en práctica de los entornos de RA. (Vallino, 1998)

(a) Muestra de un entorno de RA sin objeto virtual.

(b) Muestra de un entorno de RA con objeto virtual.

La investigación acerca de la aplicación de un Shader de Relieve a los objetos 3D que mejore el realismo y la coherencia con las escenas que conforman el entorno de RA será el resultado ha obtener en este trabajo posibilitando que los usuarios se encuentren más motivados y atraídos por estos entornos y así poder utilizar los mismos en otros dominios donde las simulaciones ejerzan una gran ayuda a la organización, distribución y práctica del trabajo a realizar.

CAPÍTULO 1: FUNDAMENTACIÓN TEÓRICA

Con el transcurso del tiempo, los trabajos en los sistemas de Realidad Aumentada se han ido perfeccionando siguiendo la necesidad de buscar una realidad fidedigna entre el mundo real y el virtual, se ha hecho necesario en el mundo desarrollar las herramientas, tecnologías, software y hardware, manteniendo así las mejoras en las prestaciones y logrando la inmersión de los usuarios en estos entornos.

1.1 Realidad Virtual

La Realidad Virtual es una técnica fotográfica de 360 grados, la cual permite moverse hacia arriba o hacia abajo y realizar acercamientos o alejamientos, que posibilita sintetizar un mundo tridimensional ficticio, creándose una ilusión de realidad.; es una representación mediante medios electrónicos; la diferencia es que en la RV tú tienes el control absoluto de los movimientos y produce la sensación, en el usuario, de estar en una situación real en la que podemos interactuar con el entorno. La RV es considerada en muchos aspectos como la interfaz definitiva entre los seres humanos y el ordenador. (Activamente)

La Realidad Virtual puede ser:

- Inmersiva: ambiente tridimensional creado por computadora el cual se manipula a través de cascos, guantes u otros dispositivos que capturan la posición y rotación de diferentes partes del cuerpo humano, presenta un costo alto y poca familiaridad del usuario con la interfaz. (Activamente)

- No Inmersiva: utiliza medios como el que actualmente nos ofrece Internet en el cual podemos interactuar a tiempo real con diferentes personas en espacios y ambientes que en realidad no existen sin la necesidad de dispositivos adicionales a la computadora, presenta bajo costo y facilidad de uso. (Activamente)

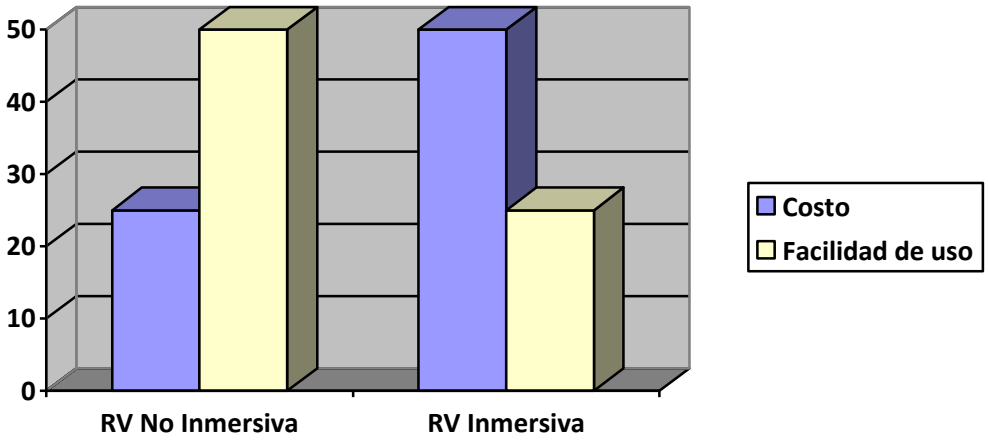

Ilustración 3. Comparación entre RV No Inmersiva y RV Inmersiva.

Básicamente la RV consiste en simular todas las posibles percepciones de una persona, como los gráficos para la vista, sonido, tacto e incluso sensaciones de aceleración o movimiento. Todas estas sensaciones diferentes deben ser presentadas al usuario de forma que se sienta inmerso en el universo generado por el ordenador, hasta el punto de dejar de percibir la realidad y ser engañado, creerse transportado (al otro lado de la pantalla) como si se tratase de un universo nuevo.

1.1.1 Historia de la Realidad Virtual

Todo lo relacionado con la RV comenzó a finales de los 70´s como material para una clase de aviación en el departamento de defensa de los Estados Unidos, para hacer simulaciones de vuelo, practicando y no arriesgando vidas. (Medicin)

Después de esto en 1982 Scott Fisher fue considerado uno de los "Padres Fundadores" de la RV y en 1985 él creo el VISIOCASO más avanzado en la Nasa Ames Center. Por todas partes empiezan a surgir equipos de desarrollo trabajando en lo que era la tecnología de la RV, y se empiezan a ver los primeros resultados comerciales. (Medicin)

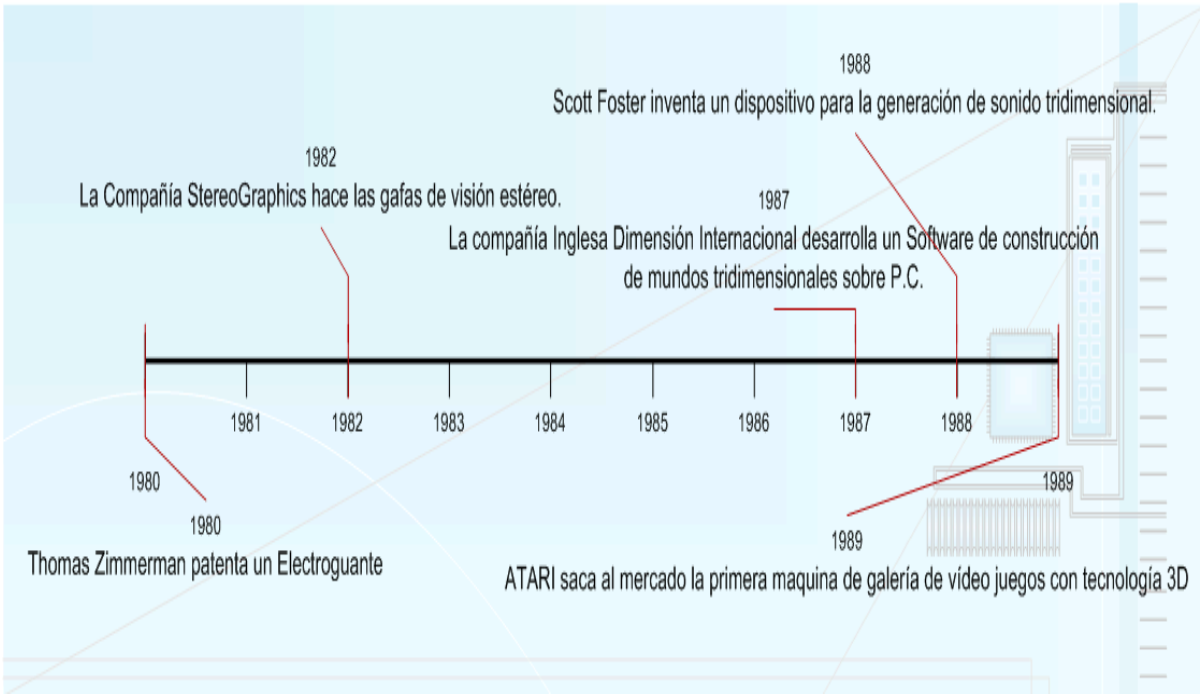

Ilustración 4: Sucesión Cronológica de Avances en la RV.

A partir de aquí entra de lleno a la carrera comercial, los sistemas de RV comienzan a popularizarse y muchos productos empiezan a invadir el mercado, en forma paralela se crea un cierto movimiento cultural conocido como el Cyberpunk. La estética y la temática del Cyberpunk han llegado en los últimos años a la televisión y al cine, quizás los mejores ejemplos son "El hombre del jardín" y "Blade Runer". (Medicin)

Los primeros albores de la RV pueden remontarse, según algunos autores, a distintas épocas, pero uno de los precedentes más claros es la industria del cine. Desde siempre la cinematografía ha intentado crear formatos de imagen y sonido que hiciesen creer al espectador que se encontraba formando parte de la escena. De este intento han surgido tecnologías como el Cinemascope o el más moderno Omnimax, así como sistemas de sonido del tipo del Dolby Surround. (Medicin)

A comienzos de los 70 se empezó a investigar como hacer más fácil el entendimiento hombre - computadora, para mejorar el rendimiento y obtener toda la potencia de estas máquinas, ya que mientras la capacidad y velocidad de los ordenadores aumentaba vertiginosamente las

habilidades para comunicarse con ellos permanecía limitada por interfaces inadecuadas. (Medicin)

También por esta época se comenzaron a apreciar las grandes ventajas de entrenar a pilotos de aviación en simuladores, en lugar de emplear auténticos aviones: menores costos, reducción de tiempo y mejora del aprendizaje, además del consiguiente y obvio nivel de seguridad que impone la práctica virtual. (Medicin)

1.1.2 Mecanismos básicos de la Realidad Virtual

Existen cinco mecanismos habitualmente empleados en las aplicaciones de la realidad virtual, siendo: (Difementes)

- Gráficos tridimensionales (3D): Simulaciones computarizadas.
- Técnicas de estereoscopia: Esta técnica permite al usuario no solo percibir las claves de la profundidad, sino además ver la imagen en relieve. Esto se debe a que la imagen que percibe cada ojo es algo distinto, lo que le permite al cerebro comparar las dos imágenes y deducir a partir de las diferencias relativas.
- Simulación de comportamiento: La simulación en el mundo virtual no posee pre-calculada la evolución, esta se va calculando en tiempo real.
- Facilidades de navegación: Es el dispositivo de control, que te permite indicar lo que quieres en la navegación, esto se realiza a través de un joystick o de las teclas de control del computador o también se puede cuando mueves la cabeza, en ese momento el sistema detecta el hecho y desplaza la imagen de la pantalla.
- Técnicas de inmersión: Consisten en aislarte de los estímulos del mundo real, al quedar privado de sensaciones procedentes de este; pierdes la referencia con la cual puedes comparar las sensaciones que el mundo virtual produce.

1.2 Realidad Aumentada

La línea entre el mundo real y el mundo virtual es cada vez más delgada. La información de los sistemas digitales es cada vez, más parte de las actividades diarias de las personas.

La RA es un tipo de ambiente virtual en el cual el usuario no se sumerge completamente en un mundo virtual sino en una mezcla de éste con el mundo real. Para el usuario aparecen los objetos virtuales y reales coexistiendo en el mismo espacio.

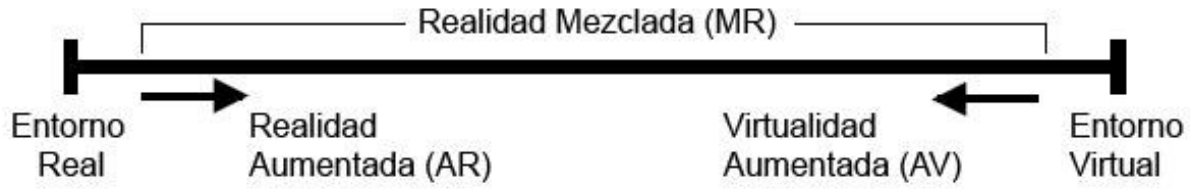

Ilustración 5. Conexión entre Realidad – Virtualidad de Milgram. (Vallino, 1998)

La RA permite presentar información digital en el mundo real por medio de dispositivos de representación especiales. Una aplicación de RA se separa de los sistemas tradicionales, pues la interacción con el usuario y el entorno es más alta y a mayor velocidad. Desarrollar sistemas de RA representa retos en cuanto a métodos de procesamiento y representación de información, no consiste simplemente en sobreponer geométricamente un mundo sobre el otro. (Revista Universidad EAFIT, 2005)

El salto de la información digital al mundo requiere de metodologías apropiadas para hacer que los sistemas conciban lo que está sucediendo en el ambiente y puedan reaccionar ante los eventos que allí ocurren. Las aplicaciones de RA actuales se ven limitadas por el poco conocimiento que tienen del mundo. Su dependencia en los métodos tradicionales de desarrollo de sistemas de gráficos tridimensionales y de RV ha restringido su acceso a la riqueza de información del entorno. Los modelos de datos heredados de tales aplicaciones

fueron efectivos dentro de su alcance, pero son limitados para los requerimientos de la RA. (Revista Universidad EAFIT, 2005)

Una aplicación de RA requiere de un modelo de datos apropiado para manejar la información del mundo real, que sea flexible, extensible y con la capacidad para representar información del mundo real y virtual. Modelos más apropiados impactarán positivamente el tiempo de desarrollo y harán los sistemas de RA más efectivos, prestando un mayor beneficio al usuario. Investigaciones en esta área han permitido establecer las características principales del contexto y ha provisto soluciones cercanas, más no completas, de representación del mundo. (Revista Universidad EAFIT, 2005)

1.2.1 Características de un sistema de Realidad Aumentada

Un sistema de RA cuenta con las siguientes características: (Revista Universidad EAFIT, 2005)

- La información digital es combinada con la realidad.
- La combinación de lo real y lo virtual se hacen en tiempo real.
- La información aumentada se localiza o registra en el espacio.

1.2.2 Funcionamiento de un sistema de Realidad Aumentada

En cuanto a su funcionamiento, las aplicaciones de RA tienen tres subsistemas fundamentales: (Revista Universidad EAFIT, 2005)

Ilustración 6: Subsistemas Fundamentales.

Visualización: Se logra con el uso de dispositivos de visualización similares a los de RV. Algunos de estos dispositivos son cascos y gafas. Éstos se componen por pantallas de cristal líquido funcionando como si fueran lentes trasparentes para que pueda observarse el mundo con la adición de los objetos virtuales. (Revista Universidad EAFIT, 2005)

Ubicación de objetos virtuales en el mundo real: Consiste en lograr que los objetos virtuales puedan "registrarse" con el mundo real, de tal forma que cuando el usuario se mueva los objetos parezcan conservar su posición. (Revista Universidad EAFIT, 2005)

Interacción: Consiste en métodos para manipular o modificar tales objetos. (Revista Universidad EAFIT, 2005)

El principal objetivo en el diseño de sistemas de RA es que éstos lleguen a ser tan portables, ligeros, pequeños y robustos como sea posible y que permitan al usuario explorar cualquier entorno no preparado, ya sea interior o exterior, sin ninguna restricción.

1.3 Shader

Gracias al avance en el hardware gráfico, el rendimiento de las aplicaciones de visualización se ha podido incrementar notablemente. Permitiendo representar escenas virtuales de mayor tamaño y complejidad. Sin embargo, aunque el aumento en el rendimiento ha posibilitado ejecutar las secuencias de visualización en un tiempo cada vez menor, se puede argumentar

que al mismo tiempo estos sistemas no se han desarrollado en cuanto a tecnologías de visualización.

La limitante fundamental del hardware de aceleración gráfica es que su estructura no se puede cambiar. Cuando se crean estos hardwares, los ingenieros prefijan un conjunto de instrucciones y algoritmos en el chip de video. Cada uno de estos algoritmos es acelerado por el hardware gráfico. Pero no se brinda la posibilidad de ejecutar en estos chips otras instrucciones que no sean las codificadas en el momento de la creación del hardware. A esta estructura estática se le denomina sistema de funciones fijas.

En los últimos 5 años se ha elaborado una alternativa al sistema de funciones fijas para aumentar la flexibilidad gráfica del hardware de video. En dos momentos claves del Pipeline gráfico se ha permitido introducir códigos que permitan ejecutar algoritmos no diseñados inicialmente en el hardware. A estos códigos se le llaman Shader y según su funcionamiento y lugar de ejecución se dividen en *Vertex Shader* y *Píxel Shader.*

1.3.1 Pixel y Vertex Shader

Un gran cambio se dio a partir de la incorporación de los *Píxel Shaders* y *Vertex Shaders*. Esto permitió a los programadores una mayor libertad a la hora de diseñar gráficos en tres dimensiones, ya que puede tratarse a cada píxel y cada vértice por separado. De esta manera, los efectos especiales y de iluminación pueden crearse mucho más detalladamente, sucediendo lo mismo con la geometría de los objetos. (Barbosa, 2006)

Un *Vertex Shader* es una función que recibe como parámetro un vértice. Sólo trabaja con un vértice a la vez, y no puede eliminarlo, sólo transformarlo. Para ello, modifica propiedades del mismo para que repercutan en la geometría del objeto al que pertenece y son los encargados de transformar todos los vértices de la escena. En ellos se ejecutan las transformaciones de espacio objeto a espacio de mundo, de cámara, y finalmente se obtiene la posición en la pantalla. Además, en ellos se incluyen todas las demás operaciones a nivel de vértices como son los cálculos procedurales de coordenadas de texturas, iluminación per-vertex, entre otras. (Barbosa, 2006)

Con esto se puede lograr ciertos efectos específicos, como los que tienen que ver con la deformación en tiempo real de un elemento; por ejemplo, el movimiento de una ola. Donde toma una gran importancia en el tratamiento de las superficies curvas, y su avance se vio reflejado en los videojuegos más avanzados de la actualidad. Particularmente, en el diseño de los personajes y sus expresiones corporales. (Barbosa, 2006)

La figura 1 muestra como trabaja el Pipeline gráfico donde la interfaz de DirectX u OpenGL pasa los vertex al driver, la GPU recibe estos datos y/o comandos y luego el *vertex shader* transforma los vértices recibidos y realiza otras operaciones a nivel de vértices, el interpolador (Rasterizador) recibe la posición en la pantalla de cada uno de los vértices y genera los triángulos correspondientes. Al dibujar estos triángulos se calcula la posición de cada uno de los píxeles que conforman el triángulo. (Barbosa, 2006)

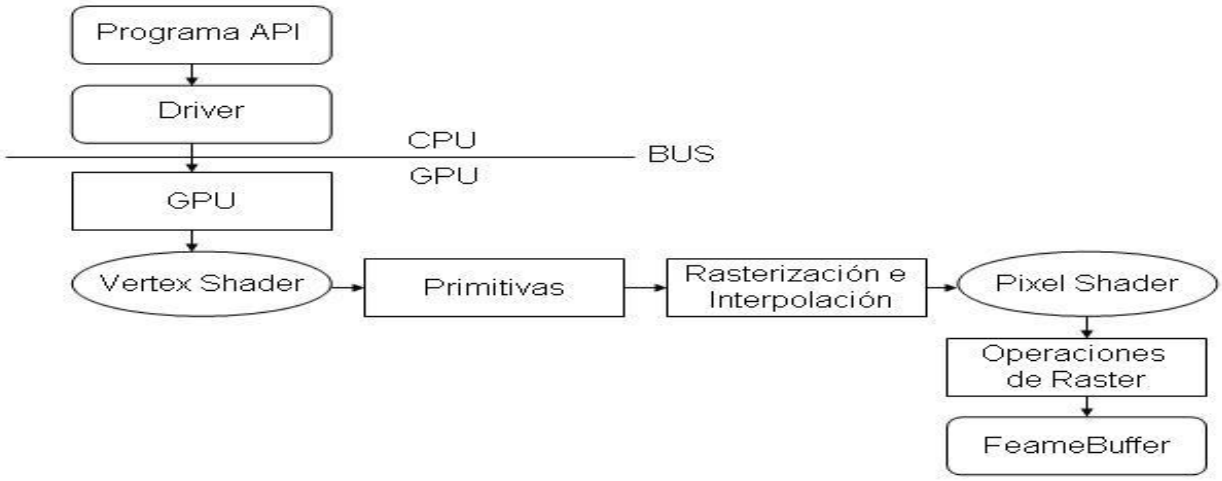

Ilustración 7: Pipeline Gráfico Conceptual usando Shaders. (Valdés & Torres, 2007)

Cada uno de estos píxeles se introduce en el *Píxel Shader* para que este calcule el color del píxel en la pantalla. De esta forma en el *Píxel Shader* se realizan todas las operaciones a nivel de píxel como es el cálculo de la iluminación per-píxel, mapeo de texturas, uso de los mapas de normales para la determinación de la normal del píxel, etc. (Barbosa, 2006)

En cambio, un *Píxel Shader* no interviene en el proceso de la definición del "esqueleto" de la escena (Wireframe), sino, forma parte de la segunda etapa: la rasterización (Rendering). Allí es donde se aplican las texturas y se tratan los pixeles que forman parte de ellas. Básicamente, un *Píxel Shader* especifica el color de un píxel. Este tratamiento individual de los pixeles permite que se realicen cálculos principalmente relacionados con la iluminación del elemento del cual forman parte en la escena, y en tiempo real. Teniendo la posibilidad de iluminar cada píxel por separado es como se lograron crear los fabulosos efectos de este estilo que se pueden apreciar en los juegos como Doom 3, Far Cry y Half Life 2, por mencionar sólo los más conocidos. (Barbosa, 2006).

Ilustración 8:.Juego Doom 3.

Ilustración 9:.Juego Far Cry.

Ilustración 10: Juego Half Life 2.

(Google)

La particularidad de los *Píxel Shaders* es que, a diferencia de los *Vertex Shaders*, requieren de un soporte de hardware compatible. En otras palabras, un juego programado para hacer uso de *Píxel Shaders* requiere de una tarjeta de video con capacidad para manipularlos.

De esta forma, el uso de los shader permite introducir nuevos algoritmos en el hardware de video. Dando la posibilidad de realizar avances en la visualización virtual sin la necesidad de esperar a que estos algoritmos sean codificados en el chip de video para obtener aceleración por hardware.

1.4 Técnicas de Gráficos Computacionales 3D

En distintas situaciones el hombre ha intentado representar la realidad desde un punto de vista artístico, matemático (curvas y superficies, ecuaciones implícitas o paramétricas) o industrial. El ordenador es un potente instrumento para que esta representación se pueda efectuar a través de capacidades computacionales o capacidades gráficas.

En este contexto la matemática y la informática trabajan juntas para conseguir buenas representaciones de curvas, adecuados modelos en tres dimensiones, algoritmos rápidos y que ocupen poca memoria para su resolución.

Para un buen desempeño de estos Shader de Relieve se expuso anteriormente la necesidad de trabajar con las funciones (*píxel y vertex shader*) las cuales nos ofrecen una forma mas detallada de trabajar los efectos especiales y de iluminación logrando que los objetos parezcan mas reales, pero para obtener este resultado no solo se puede limitar al trabajo con estas funciones sino también se deben utilizar técnicas capaces de darle a estos objetos virtuales la mayor realidad posible los cuales se puedan confundir con los objetos reales.

Las técnicas de gráficos computacionales son de vital importancia a la hora de desarrollar estos shader porque son capaces de darle al objeto mayor realismo y credulidad. Estas técnicas trabajan directamente con los pixel y vertex de la textura de los objetos siendo capaces de deformarla así como su geometría.

Dentro de estas técnicas se encuentran cinco que son fundamentales para el desarrollo de shader de relieve, las cuales se mencionan a continuación:

- Bump Mapping: Es una técnica de gráficos computacionales 3D que consiste en dar un aspecto rugoso a las superficies de los objetos. Esto puede ser claramente aplicado cuando se desea dar un efecto de relieve en el objeto. La técnica del Bump Mapping se utiliza para agregar el detalle a una imagen sin aumentar el número de polígonos. Modifica las normales de las texturas dependiendo de si la zona es clara u obscura sin modificar en ningún momento la topología ni la geometría del objeto. (Bump)

- Normal Mapping: Es la aplicación de una técnica 3D que permite dar una iluminación y relieve mucho más detallado a la superficie de un objeto. Es una evolución del Bump Mapping y se aplica en videojuegos a los que les dota un mayor realismo, así como en películas de animación para agilizar los cálculos y reducir por tanto el número de polígonos con los que en un principio contaba los objetos. Esta técnica no aplica la información por pixel en tonos de grises el cual representaría la altura, sino en colores RGB dando más fidelidad al original que se quiere imitar y trabaja con los tres ejes X, Y, Z. (Normal)

- Parallax Mapping: Esta técnica es el siguiente paso después del Normal Mapping, pues tiene en cuenta el ángulo con el que se mira a la superficie e introduce variaciones en la iluminación. Sigue siendo un truco, pues la geometría de la superficie se mantiene intacta, pero el efecto está mejor conseguido. (Welsh, 2004)

- Displacement Mapping: Esta es el último grito en la búsqueda de superficies con relieves cada vez más creíbles. Así, en lugar de utilizar las trampas del Parallax o Normal Mapping, esta técnica modifica realmente la superficie, por lo que el detalle es real, no una simple ilusión. Esto conlleva que la sombra que proyecta el objeto o superficie con Displacement Mapping tenga en cuenta la rugosidad de la misma. (Guinot, 2006)

- Relief Mapping: Esta técnica es muy similar a la de Parallax Mapping en el sentido de que intenta simularse con una técnica de deformación de la imagen .Esta produce puntos de vista de superficies 3-D y escenas para aumentar con texturas por Texel de profundidad. (Policarpo, Oliveira, & Comba)

1.5 Unidad de Procesado de Gráficos (GPU)

Una de las herramientas a utilizar es la GPU (Graphics Processing Unit) la cual no es más que una "Unidad de Procesado de Gráficos" dedicado exclusivamente al procesamiento de gráficos, para aligerar la carga de trabajo del procesador central en aplicaciones como los videojuegos y/o aplicaciones 3D interactivas. Las GPU actualmente disponen de gran cantidad de primitivas, buscando mayor realismo en los efectos. (100cia)

Las modernas GPU son descendientes de los chips gráficos monolíticos de finales de la década de 1970 y 1980. Estos chips tenían soporte BitBLT limitado en la forma de sprites y usualmente no tenían soporte para dibujo de figuras. (100cia)

Algunos GPU podían ejecutar varias operaciones en una lista de "display" y podían usar DMA para reducir la carga en el procesador anfitrión. Hacia finales de los 80 y principios de los 90, microprocesadores de propósito general de alta velocidad fueron muy populares para implementar los GPUs más avanzados. Muchas tarjetas gráficas para PCs y Estaciones de Trabajo usaban Procesadores Digitales de Señales (DSP por sus siglas en inglés) tales como la serie TMS340 de Texas Instruments, para implementar funciones de dibujo rápidas y muchas impresoras laser contenían un procesador de barrido de imágenes "PostScript" (un caso especial de GPU) corriendo en un procesador RISC como el AMD 29000. (100cia)

Conforme la tecnología de proceso de semiconductores fue mejorando, eventualmente fue posible mover las funciones de dibujo y las BitBLT a la misma placa y posteriormente al mismo chip a manera de un controlador de buffer de "marcos"(frames), tal como VGA. Estos aceleradores gráficos de 2D "reducidos" no eran tan flexibles como los basados en microprocesadores, pero eran mucho más fáciles de hacer y vender. La Commodore AMIGA fue la primera computadora de producción en masa que incluía una unidad blitter y el sistema gráfico IBM 8514 fue una de las primeras tarjetas de video para PC en implementar primitivas 2D en hardware. (100cia)

1.5.1 Diferencias y especializaciones entre GPU y CPU

Si bien en un computador genérico no es posible reemplazar la CPU por una GPU, hoy en día las GPU son muy potentes y pueden incluso superar la frecuencia de reloj de una CPU antigua (más de 500MHz). Pero la potencia de las GPU y su dramático ritmo de desarrollo reciente se deben a dos factores diferentes. El primer factor es la alta especialización de las GPU, ya que al estar pensadas para desarrollar una sola tarea, es posible dedicar más silicio en su diseño para llevar a cabo esa tarea más eficientemente. Por ejemplo, las GPU actuales están optimizadas para cálculo con valores en coma flotante, predominantes en los gráficos 3D. (100cia)

Por otro lado, muchas aplicaciones gráficas conllevan un alto grado de paralelismo inherente, al ser sus unidades fundamentales de cálculo (vértices y píxeles) completamente independientes. Por tanto, es una buena estrategia usar la fuerza bruta en las GPU para completar más cálculos en el mismo tiempo. Los modelos actuales de este suelen tener una media docena de procesadores de vértices (que ejecutan *Vertex Shaders*), y hasta dos o tres veces más procesadores de fragmentos o píxeles (que ejecutan Fragment Shaders (*Pixel Shader*)). De este modo, una frecuencia de reloj de unos 500-600MHz (el estándar hoy en día en las GPU de más potencia), muy baja en comparación con lo ofrecido por las CPU (3.8-4 GHz en los modelos más potentes (pero no lo más eficientes), se traduce en una potencia de cálculo mucho mayor gracias a su arquitectura en paralelo. (100cia)

Una de las mayores diferencias con la CPU estriba en su arquitectura. A diferencia del procesador central, que tiene una arquitectura Eckert-Mauchly, la GPU se basa en el Modelo Circulante. Éste modelo facilita el procesamiento en paralelo, y la gran segmentación que posee la GPU para sus tareas. (100cia)

1.5.2 Arquitectura de la GPU

Una 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 vértice y el píxel como las principales unidades que maneja la GPU. (100cia)

Adicionalmente, y no con menos importancia, se encuentra la memoria. Esta se destaca por su rapidez y juega un papel relevante a la hora de almacenar los resultados intermedios de las operaciones y las texturas que se utilicen. (100cia)

Inicialmente, a la GPU le llega la información de la CPU en forma de vértices. El primer tratamiento que reciben estos vértices se realiza en el *Vertex Shader*. Aquí se realizan transformaciones como la rotación o el movimiento de las figuras. Tras esto, se define la parte de estos vértices que se va a ver (*clipping*), y los vértices se transforman en píxeles mediante el proceso de rasterización. Estas etapas no poseen una carga relevante para la GPU. (100cia)

Donde sí se encuentra el principal cuello de botella del chip gráfico es en el siguiente paso: el *Pixel Shader*. Aquí se realizan las transformaciones referentes a los píxeles, tales como la aplicación de texturas. Cuando se ha realizado todo esto, y antes de almacenar los píxeles en la caché, se aplican algunos efectos como el *antialiasing*, *blending* y el efecto niebla. (100cia)

Otras unidades funcionales llamadas ROP toman la información guardada en la caché y preparan los píxeles para su visualización. También pueden encargarse de aplicar algunos efectos. Tras esto, se almacena la salida en el *frame buffer*. Ahora hay dos opciones: o tomar directamente estos píxeles para su representación en un monitor digital, o generar una señal analógica a partir de ellos, para monitores analógicos. Si es este último caso, han de pasar por un DAC, (*Digital-Analog Converter)*, para ser finalmente mostrados en pantalla. (100cia)

1.5.3 Programación de la GPU

Al inicio, la programación de la GPU se realizaba con llamadas a servicios de interrupción de la BIOS. Tras esto, la programación de la unidad de procesado se empezó a hacer en el lenguaje ensamblador específico a cada modelo. Posteriormente, se situó un nivel más entre el hardware y el software, diseñando las API (Application Program Interface), que

proporcionaban un lenguaje más homogéneo para los modelos existentes en el mercado. El primer API usado ampliamente fue estándar abierto OpenGL (Open Graphics Language), tras el cuál Microsoft desarrolló DirectX. (100cia)

Tras el desarrollo de APIs, se decidió crear un lenguaje más natural y cercano al programador, es decir, desarrollar un lenguaje de alto nivel para gráficos. Por ello, de OpenGL y DirectX surgieron estas propuestas. El lenguaje estándar de alto nivel, asociado a la biblioteca OpenGL es el "OpenGL Shading Language", GLSL, implementado en principio por todos los fabricantes. La empresa californiana NVidia creó un lenguaje propietario llamado Cg (del inglés, "C for graphics"), con mejores resultados que GLSL en las pruebas de eficiencia. En colaboración con NVidia, Microsoft desarrolló su "High Level Shading Language", HLSL, prácticamente idéntico a Cg, pero con ciertas incompatibilidades menores. (100cia)

1.6 Tarjeta gráfica

Una **tarjeta gráfica**, **tarjeta de vídeo**, **tarjeta aceleradora de gráficos** o **adaptador de pantalla**, es una tarjeta de expansión para una computadora personal, 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.

Se denota con el mismo término tanto a las habituales tarjetas dedicadas y separadas como a las GPU integradas en la placa base (aunque estas ofrecen prestaciones inferiores).

1.6.1 Historia de las tarjetas gráficas

La historia de las tarjetas gráficas comienza a finales de los años 1960, cuando se pasa de usar impresoras como elemento de visualización a utilizar monitores. Las encargadas de crear aquellas primeras imágenes fueron las tarjetas de vídeo. (García, 2004)

La primera tarjeta gráfica, que se lanzó con los primeros IBM PC, fue desarrollada por IBM en 1981. La MDA (*Monochrome Display Adapter*) trabajaba en modo texto y era capaz de representar 25 líneas de 80 caracteres en pantalla. Contaba con una memoria de vídeo de

4KB, por lo que sólo podía trabajar con una página de memoria. Se usaba con monitores monocromo, de tonalidad normalmente verde. (García, 2004)

Desde que comenzó la "revolución 3D" en el ámbito de los juegos de computadora, allá por mediados de la década de los 90', la tendencia de la tecnología aplicada a este rubro ha sido trasladar el trabajo de procesamiento de gráficos tridimensionales, desde la CPU hacia la tarjeta de video. (García, 2004)

A partir de ahí se sucedieron diversas controladoras para gráficos, resumidas en la tabla adjunta. (García, 2004)

	Año	**Modo Texto**	**Modo Gráficos**	**Colores**	**Memoria**
MDA	1981	80*25	-	1	4 KB
CGA	1981	80*25	640*200	4	16 KB
HGC	1982	80*25	720*348	1	64 KB
EGA	1984	80*25	640*350	16	256 KB
IBM	1987	80*25	1024*768	256	-
MCGA	1987	80*25	320*200	256	-
VGA	1987	720*400	640*480	256	256 KB
SVGA	1989	80*25	1024*768	256	2 MB
XGA	1990	80*25	1024*768	65K	1 MB

Tabla 1: Resumen de las diferentes tarjetas gráficas. (García, 2004)

VGA tuvo una aceptación masiva, lo que llevó a compañías como ATI, Cirrus Logic y S3 Graphics, a trabajar sobre dicha tarjeta para mejorar la resolución y el número de colores. Así nació el estándar SVGA (*Super VGA*). Con dicho estándar se alcanzaron los 2 MB de memoria de vídeo, así como resoluciones de 1024 x 768 puntos a 256 colores. (García, 2004)

La evolución de las tarjetas gráficas dio un giro importante en 1995 con la aparición de las primeras tarjetas 2D/3D, fabricadas por Matrox, Creative, S3 y ATI, entre otros. Dichas tarjetas cumplían el estándar SVGA, pero incorporaban funciones 3D. En 1997, 3dfx lanzó el chip gráfico **Voodoo**, con una gran potencia de cálculo, así como nuevos efectos 3D (*Mip Mapping*, Z-Buffering, Antialiasing...). A partir de ese punto, se suceden una serie de lanzamientos de tarjetas gráficas como **Voodoo2** de 3dfx, **TNT** y **TNT2** de NVIDIA. La potencia alcanzada por dichas tarjetas fue tal que el puerto PCI donde se conectaban se quedó corto. Intel desarrolló el puerto AGP (*Accelerated Graphics Port*) que solucionaría los cuellos de botella que empezaban a aparecer entre el procesador y la tarjeta. Desde 1999 hasta 2002, NVIDIA dominó el mercado de las tarjetas gráficas (absorbiendo incluso a 3dfx) con su gama GeForce. En ese período, las mejoras se orientaron hacia el campo de los algoritmos 3D y la velocidad de los procesadores gráficos. Sin embargo, las memorias también necesitaban mejorar su velocidad, por lo que se incorporaron las memorias DDR a las tarjetas gráficas. Las capacidades de memoria de vídeo en la época pasan de los 32 MB de GeForce, hasta los 64 y 128 MB de GeForce 4. (García, 2004)

En 2006, NVIDIA y ATI se repartían el liderazgo del mercado con sus series de chips gráficos GeForce y Radeon, respectivamente. (García, 2004)

CAPÍTULO 2: TÉCNICA DE GRÁFICOS COMPUTACIONALES 3D.

En este capítulo se realizará el estudio de algunas técnicas de gráficos computacionales que se consideran las más importantes por el realismo que son capaces de ofrecer a las escenas virtuales. A continuación se desarrollará una breve explicación de cada una de ellas, las cuales fueron mencionadas en el capítulo anterior y se abordará con mayores detalles las que se utilizarán en el desarrollo de la investigación, incluyendo sus métodos matemáticos.

2.1 Bump Mapping.

Una de las técnicas a las que se hace referencia anteriormente es Bump Mapping, que consiste en dar un aspecto rugoso a las superficies de los objetos. Esta técnica modifica las normales de la superficie a modificar, sin modificar su geometría. Lógicamente, las normales originales de la superficie serán perpendiculares a la misma. El Bump Mapping se encarga de eliminar esa perpendicularidad y modificar estas normales para lograr el efecto deseado, todo ello sin modificar en ningún momento la topología ni la geometría del objeto. El resultado es bastante enriquecedor y detallado, y pueden lograrse grandes parecidos a elementos naturales. (Bump) En la Fig.11 se muestra el empleo de la técnica Bump Mapping.

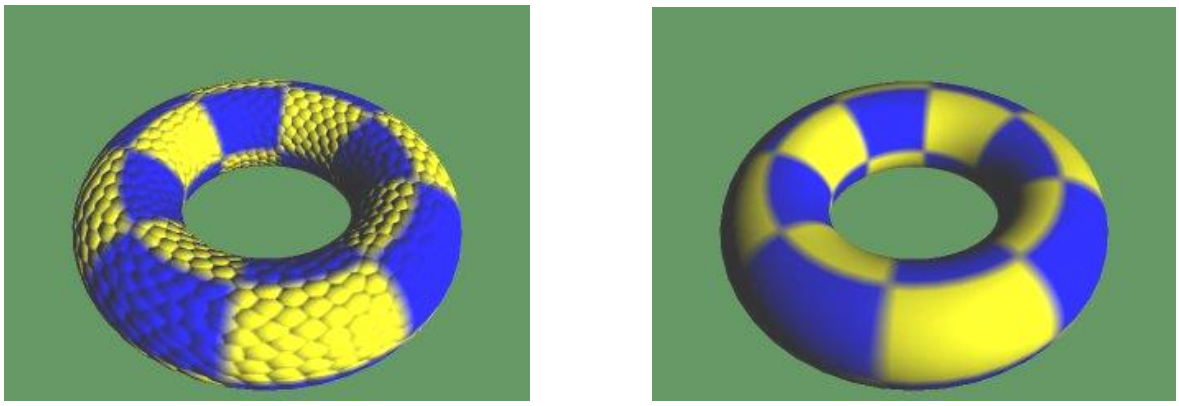

Ilustración 11. Comparación utilizando la técnica de Bump Mapping (Bump)

El Bump Mapping se utiliza en el cálculo de la pendiente de cada píxel y en la determinación de la ruta de altura. Esto puede ser tan poco como dos líneas de código para el cálculo de la X y Y que determinan el gradiente en cada píxel o puede ser más complejo. El resultado final es el cálculo de la pendiente, por lo que la complejidad del código tal vez no importe. (Dreijer)

- Primeramente se hacen los cálculos al vector del punto alineado al eje de U y V del Bump Mapping ajustándolo correctamente a la normal.

- Luego es obtenida la transformación de la normal de la luz de U y V de la sombra en la textura del píxel comparando el vector de dirección del punto en el mapa con la normal de la luz de X, Y, Z de los puntos calculados, cambiando de la luz del Per-pixel a la rasterización de esta.

Si el punto está más expuesto a la luz, entonces es más brillante, frente a los puntos más alejados de la luz realmente obtenemos rápidamente más oscuridad.

Esta técnica utiliza como método matemático el método de Blinn, desarrollado en 1978 por James Blinn, el cuál se encarga de simular arrugas en una superficie sin necesidad de modificar geométricamente el modelo. Para esto se utiliza una imagen en escala de grises en la que los colores claros significaban protuberancias y los oscuros huecos.

Un factor importante para la creación del Bump Mapping es el cálculo del vector de intensidad de la luz, a continuación se presenta la Fig.12 donde se ve la interacción de esta con la normal del objeto. Teniendo en cuenta el planteamiento anterior se plantea el objetivo de eliminar la perpendicularidad de los objetos.

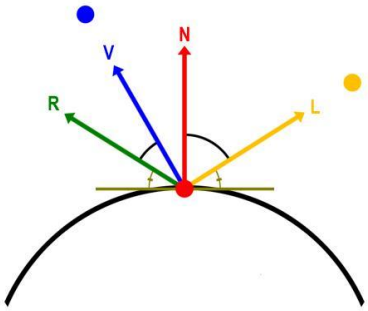

Ilustración 12. Interacción de los vectores normal, luz, vista y reflexión en un pixel. (Dreijer)

N: Normal de la superficie en el pixel actual.

L: Vector desde el punto en la superficie a la fuente de luz (el vector luz).

V: Vector vista.

R: Vector reflexión.

En resumen el proceso para obtener Bump Mapping se realiza de la siguiente forma: (Dreijer)

1. Calcular una matriz inversa constituida por Tangente, Binormal y Normal (TBN) para todos los triángulos.
2. Calcular el vector luz y transformarlo desde el espacio objeto en el espacio tangente.
3. Leer el vector normal desde el mapa normal y descomponer el rango que va desde [-1, 1] en lugar de [0, 1].
4. Calcular al final el color difuso tomando el producto escalar entre la luz y el vector normal y multiplicando con el color de la luz y el color de la superficie del material.
5. Repetir este procedimiento para todos los pixeles en una determinada superficie.

2.2 Normal Mapping.

En gráficos 3D de la computadora Normal Mapping es una aplicación de la técnica conocida como Bump Mapping. Esta técnica reemplaza la normal completamente. Al igual que Bump Mapping, se utiliza para añadir detalles al sombreado sin utilizar más polígonos. Pero cuando un Bump Mapping suele ser calculado sobre la base de un solo canal (interpretarse en escala de grises) la imagen, la fuente de las normales en Normal Mapping suele ser una imagen multicanal (X, Y, Z canales) derivados de un conjunto de las versiones más detalladas de los objetos. Los valores de cada canal por lo general representan a la X, Y, Z coordenadas de la normal en el punto correspondiente a ese Texel. (Normal)

Normal Mapping se encuentra normalmente en dos variedades: objeto-espacio y espacio-tangente. Se diferencian en sistemas de coordenadas en el que las normales se miden y se almacenan. Uno de los usos más interesantes de esta técnica es en gran medida mejorar la apariencia de un objeto que tiene un modelo de explotación que carece de polígonos (bajo poli).

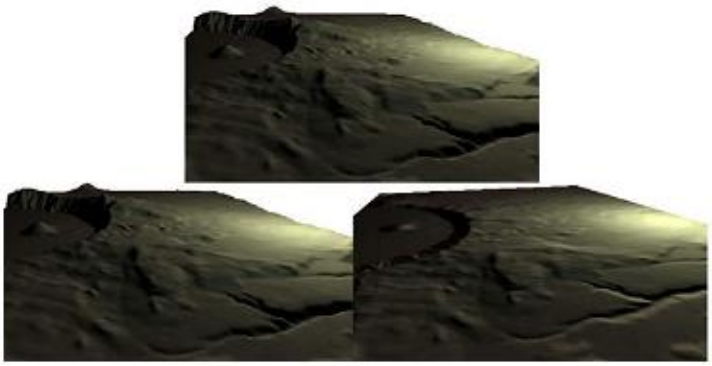

Ilustración 13: Renders de la alta resolución del terreno (Arriba), uno de baja resolución con Normal Mapping aplicado (a la izquierda abajo) y un plano con la ruta normal aplicada (parte inferior derecha). (Jesús Gumbau)

Un método matemático utilizado por el Normal Mapping es el espacio tangente, para la generación de Normal Mapping por este método deben ser realizadas las siguientes tareas:

- El vértice shader recibirá como parámetro de entrada el vector espacio tangente conformado por la normal, la tangente y la binormal (TBN) como vértice atributo y como parámetro de salida obtendrá estos valores con interpolación lineal para cada fragmento. (Normal)

- El *pixel shader* genera el mapa de normales, de modo que se pasan las coordenadas de las normales a las componentes RGB del color resultante tras ejecutar el *pixel shader*. Para ello hay que pasar los valores de la normal, transformados al espacio tangente, al rango de valores aceptado por el plano RGB, que es [0,1]. (Normal)

A continuación se mostrará un ejemplo del pseudocódigo programado donde se refleja la entrada de valore por vertex y *pixel Shader*.

```
result.pos.x = in.texcoord.u;
result.pos.y = in.texcoord.v;
result.pos.z = 0;
result.normal = in.normal;
result.binormal = in.binormal;
result.tangent = in.tangent;
```

```
normal= in.normal.tangentSpace()
normal = normal.range(0,1);
result.color.r = normal.x;
result.color.g = normal.y;
result.color.b = normal.z;
```

Algotimo1 (izquierda): *Vertex Shader* para la generación de Normal Map y Algotimo2 (derecha): *Pixel Shader* para la generación de Normal Map. (Normal)

Como se había planteado en el capítulo anterior el *vertex Shader* trabaja transformando las coordenadas de lo vectores y el *pixel Shader* los colores, de ahí que el algoritmo 1 trabaje con el vector TBN (tangente, Binormal, Normal) donde las coordenadas (X, Y, Z) son las del objeto y las coordenadas (U, V) las de la textura; el algoritmo 2 trabaja con la componente RGB (Rojo, Verde, Azul).

2.2.1 Normal Mapping en los videojuegos

Normal Mapping extendió el uso de la propiedad comercial en videojuegos a partir de finales del 2003, y seguido de los juegos de código abierto en los últimos años. La popularidad de renderizado de Normal Mapping en tiempo real se debe a su buena calidad y a los requisitos de procesamiento frente a otros métodos de producir efectos similares. Gran parte de esta eficiencia se hace posible por la utilización de una técnica que disminuye selectivamente el detalle de la ruta normal de una determinada textura nombrada mipmapping. Esta plantea que las superficies más distantes requieren de menor complejidad para ser simuladas. (Normal)

Ilustración 14: Ejemplo de la utilización de Normal Mapping en el videojuego Dewy de Aventura. (Google)

Destacar que Normal Mapping es una ampliación del Bump Mapping, esta técnica se trabaja con mayor efectividad ya que logra darle a los objetos mayor realidad con la misma cantidad de polígonos que el Bump Mapping.

2.3 Parallax Mapping.

Parallax Mapping (también llamado offset Mapping o mapas de desplazamiento virtual) es una mejora de las técnicas de gráficos computacionales Bump Mapping o Normal Mapping que se aplica a las técnicas de mapeo de texturas 3D en aplicaciones como juegos de vídeo. Para el usuario final, esto significa que las texturas que representan muros de piedras, serán más evidentes y, por tanto, se observará una mayor profundidad de realismo en los entornos de simulación. (Welsh, 2004)

Esta técnica es un proceso de un solo paso que no tiene en cuenta la oclusión. Con posterioridad se han hecho mejoras al algoritmo iterativo que incorporan enfoques para permitir la oclusión y una silueta exacta de renderizado. (Welsh, 2004)

Ilustración 15: Ejemplo de la técnica Parallax Mapping. (Welsh, 2004)

Parallax Mapping es una técnica correcta de aproximación de la aparición de superficies irregulares, modificando la textura a coordinar para cada píxel. Parallax se puede simular bien mediante la construcción de un modelo geométrico de una superficie, pero para extraer todos los polígonos resulta computacionalmente caro.

Esta técnica es ejecutada por el desplazamiento de las coordenadas de textura en un punto en el polígono proporcionado por una función del ángulo de visión en el espacio tangente (el ángulo relativo a la superficie normal) y el valor de la altura de ruta en ese momento. Por el criterio de la función del ángulo de visión, las coordenadas de texturas más pronunciadas son más desplazadas y así dan la ilusión de profundidad debido a los efectos de paralelaje. (Welsh, 2004)

2.3.1 Aplicación de Parallax Mapping

La aproximación de efectos 2D en Parallax Mapping es un poco costosa. Todo lo que hace es modificar las coordenadas de textura con el fin de deformar una imagen. Debido a su facilidad de aplicación, puede ser utilizado en combinación con otros tipos de técnicas Mapping. Las nuevas coordenadas de textura se pueden utilizar para regular índices de mapas de textura, para colorear una superficie Normal Mapping para Bump Mapping y reflexión de vectores computacionales. Las coordenadas de textura modificadas no son perfectas, aunque, se debe tener cuidado en la producción de mapas de altura y escala especificando los valores y prejuicios. Debido a la defectuosa hipótesis de que los valores de altura son los mismos de coordinar una textura a la siguiente, los mapas de altura tienden a causar problemas. Parallax Mapping es incapaz de detectar zonas de una superficie que ocluyen otras áreas, lo cual es común en una superficie con ángulos empinados. (Welsh, 2004)

2.4 Displacement Mapping.

El Displacement Mapping es una técnica alternativa de gráficos por computadoras, en contraste con Bump Mapping, Normal Mapping, y Parallax Mapping, utilizando un procedimiento de la textura o la altura de ruta para causar un efecto real en que la posición geométrica de los puntos en la superficie con textura son desplazados a lo largo de la superficie normal, de acuerdo con el valor de la textura en función de la evaluación de cada uno de los puntos en la superficie. Esta técnica es para superficies de un gran sentido de profundidad y detalle, que permita la libre

oclusión, la libre sombra y silueta; por otro lado, es la más costosa de esta clase de técnicas, debido a la cantidad adicional de geometría que utiliza. (Guinot, 2006)

Vertex Displacement Mapping permite hacer realidad, por ejemplo, muy realista, simulaciones de agua. (Gamasutra)

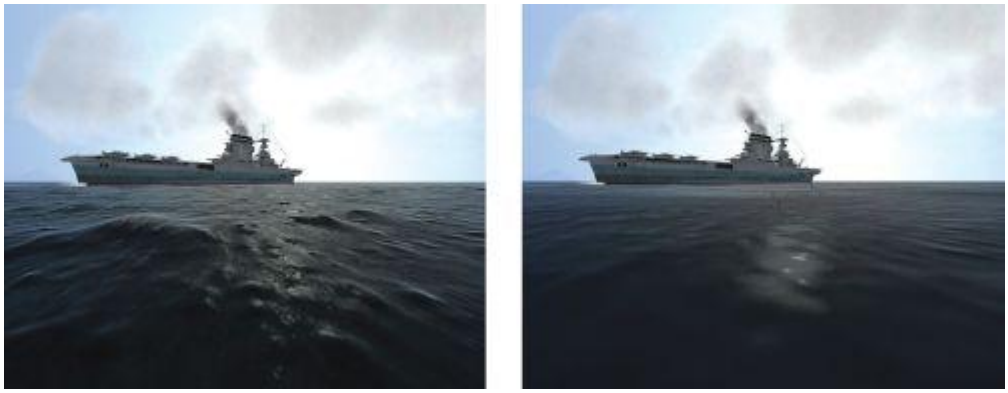

Ilustración 16. El beneficio de Displacement Mapping la superficie de agua renderizada (a la izquierda) con el Displacement Mapping, (a la derecha) sin Displacement Mapping. (Gamasutra)

Durante años, el Displacement Mapping fue una peculiaridad de gama alta como los sistemas de prestación PhotoRealistic RenderMan, mientras que en tiempo real de APIs, como OpenGL y DirectX, carecían de esta posibilidad. Una de las razones de esta ausencia es que la original aplicación de Displacement Mapping requiere un mosaico de adaptación de la superficie a fin de obtener micro polígonos cuyo tamaño coincide con el tamaño de un pixel en la pantalla. (Guinot, 2006)

Vertex Displacement Mapping puede utilizarse para crear efectos en fresco y permite aprovechar la potencia de los procesadores de vértices que se olvida frecuentemente en beneficio de los procesadores de pixel. Sin embargo, se debe tener en cuenta una serie de limitaciones como las capacidades de hardware para conseguir la textura de obtención del vértice (ATI / Radeon GPU no apoyan Vertex Displacement Mapping), el problema de las normales que se hacen mal, e incluso la imposibilidad de utilizar algunos algoritmos (o partes de su motor 3D) como sombra de

los volúmenes porque estos algoritmos deben actuar en los vértices almacenados en la memoria del sistema y no en los almacenados en la memoria gráfica GPU. (Guinot, 2006)

Ventajas

- El uso de Displacement Mapping reduce la cantidad de memoria requerida para un determinado nivel de detalle en las mallas de los objetos. (Guinot, 2006)
- La reducción de la malla es una versión mucho más simple (por lo general en torno a unos pocos cientos de vértices en lugar de decenas de miles) mediante operaciones como la animación, es más barata. Debido a esto, la gama de posibles operaciones se ha ampliado, animaciones más complejas son posibles y diferentes técnicas son utilizadas, tales como el volumen de preservación y tela de simulación. (Guinot, 2006)
- Otra ventaja es que la animación de algoritmos utilizados ya no están ligados a la GPU. (Guinot, 2006)
- Una ventaja más general es que el uso de Displacement Mapping se convierte en mallas (delicado 3D con entidades de compleja conectividad) en algunas entidades 2D. Objetos 2D (es decir, texturas e imágenes) han sido estudiados ampliamente, y hay una gran cantidad de técnicas existentes que pueden ser aplicadas a las mallas. (Guinot, 2006)
- El uso de hardware gráfico puede ser más acelerado. (Guinot, 2006)

Desventajas

- Displacement Mapping presenta algunas restricciones en las mallas pues no son aplicable en todo el mundo. (Guinot, 2006)
- Los árboles u otras plantas no son fáciles de representar utilizando mapas de desplazamiento, ya que no existe una buena parametrización en 2D para la utilización a lo

largo de sus superficies. Esto se debe a que una ruta de desplazamiento sólo puede tener un solo valor de altura. (Guinot, 2006)

- A primera vista, el soporte de hardware es muy delgado para la técnica Displacement Mapping ya que esta solo se puede visualizar en una PC que contenga una tarjeta gráfica NVidia. (Guinot, 2006)

2.5 Relief Mapping

En los gráficos computacionales Relief Mapping es una técnica alternativa de Parallax Mapping que es mucho más precisa, y puede apoyar el libre sombreado y Normal Mapping. Puede ser simplemente descrita como una corta distancia de trazado de rayo hecha en un *pixel shader*. (Policarpo, Oliveira, & Comba)

Relief Mapping es una técnica de mapeo de texturas que apoya la representación de detalles de superficies 3D. Esta técnica proporciona corregir puntos de vista de superficies 3D y escenas para aumentar con texturas por Texel de profundidad. Todavía no es común en los juegos de vídeo, pues se trata de una técnica bastante lenta debido a la necesidad de una gran cantidad de procesamiento de per-pixel. (Policarpo, Oliveira, & Comba)

Ilustración 17: Tetera presentando diferentes texturas de Relief Mapping utilizando iluminación per- pixel y libre sombreado. (Policarpo, Oliveira, & Comba)

Relief Mapping simula la apariencia de los detalles de la superficie geométrica por sombreado de fragmentos individuales de acuerdo a cierta profundidad y a la información de la superficie normal que está asignada en los modelos poligonales. Los mapas de profundidad y normal pueden ser almacenados como una sola textura RGBA (32-bit por Texel) llamada como textura de relief. (Policarpo, Oliveira, & Comba)

Con las características que se han mencionado anteriormente de las técnicas de gráficos computacionales descritas anteriormente se puede observar que estas se han ido perfeccionado con el paso del tiempo, siendo capaces de ofrecerle mayor realismo a los objetos y por tanto hay que tenerlas en cuenta para la realización de Shaders de Relieve constituyendo estas el escalón fundamental para la confección de los mismos.

Una de las técnicas que se tratará en el próximo capítulo es Parallax Mapping ya que esta se destaca por sus características y su aplicación en objetos sólidos, dando un relieve destacado, es decir, consigue un mejor esfuerzo. De esta forma se ha querido mostrar la importancia de la utilización de estas técnicas para ofrecerle mayor realismo a los objetos 3D en las escenas que conforman el entorno de RA.

CAPÍTULO 3: DESCRIPCIÓN DE LA PROPUESTA DE SOLUCIÓN.

En este capítulo se dará a conocer la propuesta de solución más óptima para la creación de un Shader de Relieve, teniendo en cuenta su necesidad y requisitos para su buen desarrollo, mostrando la técnica de gráficos computacionales más balanceada, así como sus características principales, ventajas y desventajas. También se hará una comparación entre todas las técnicas prescritas en el capítulo anterior teniendo en cuenta sus principales características.

3.1 Propuesta para la aplicación de un Shader de Relieve.

La propuesta que este trabajo presenta como solución, no es más que el reto principal que posee la futurista RA, el cual consiste en hacer pensar al usuario que está interactuando en un mundo completamente real, donde se mezclan objetos 3D con la verdadera realidad formando parte de esta, obteniéndose esto solamente con la acción de las técnicas y recursos expuestos anteriormente.

El relieve es la cúspide entre las formas que se utilizan para convertir un objeto 3D en algo lo más cercano posible a un objeto real, para esto se cuenta con la aplicación de un Shader de Relieve a objetos 3D, con el interés de ser utilizado en las escenas de entornos de RA, permitiendo ofrecer a los objetos 3D mayor realismo y credulidad contribuyendo a que estas escenas sean más reales y obtengan mayor aceptación por parte de los usuarios, ayudando así a la inmersión de estos en ese mundo y mejorando la interacción del hombre con la máquina.

Para ello se utiliza como soporte técnico las Tarjetas Gráficas NVIDIA (preferentemente de la familia de las GForce) o ATI (Radeon) por las características que presentan al ofrecer mayores y mejores prestaciones que las GPUs integradas en la placa base, además de ser las dos empresas con mejores resultados en el campo de los gráficos computacionales, permitiendo el trabajo con sus unidades funcionales como son el vertex y pixel (fragment) shader.

A estas tarjetas se le incorporan diversas técnicas de gráficos computacionales 3D como son Bump Mapping, Normal Mapping, Parallax Mapping, Relief Mapping y por último Displacement Mapping, expuestas en un orden ascendente en cuanto a su característica principal que constituye en darle un aspecto rugoso a los objetos 3D ofreciéndole al usuario una mayor credulidad en las escenas que participa.

Se emplea como lenguajes de programación Cg o CgFx, por su característica de simplemente, hacerle la vida más fácil al programador, intentando ser lo que fue el C para los programadores de los primeros tiempos de los ordenadores: un lenguaje de alto nivel, más cerca del lenguaje humano que del de la máquina, con el objetivo de simplificar al máximo la programación de los Shaders de las tarjetas DX8, no se ha dicho tarjetas de NVIDIA, sino DX8, por la sencilla razón de que el código creado va a funcionar en cualquier tarjeta compatible con la versión 8 de las librerías de Microsoft, siempre que el fabricante haga las modificaciones pertinentes en dicho código, algo sencillo por la naturaleza de código abierto que va a tener el Cg y su predecesor CgFx, el cual es una versión más moderna del Cg ganándose el alias de **las siglas del futuro del desarrollo de los juegos en 3D**.

3.2 Técnica Propuesta.

Toda esta amplia gama de propuestas expresadas anteriormente es necesario simplificarla y convertirla en un solo camino el cual finalice en la aplicación de un Shader de Relieve que resulte ser lo más óptimo y balanceado posible para su utilización en cualquiera de la ramas de trabajo que sea aplicable este entorno, con todos los requisitos y necesidades que se necesitan para su desarrollo, aplicación e implementación.

La solución que se quiere expresar en este trabajo es que el entorno de RA en el que se esté trabajando posea suficiente fotorealismo, para aplicar relieve a los objetos virtuales se hace necesario emplear un Shader que tenga el balance en cuanto a visualización y optimización

requerida, por tanto este Shader debe utilizar como técnica de gráficos computacionales el Parallax Mapping.

3.2.1 Principales características de la técnica propuesta.

Parallax Mapping (o Virtual Displacement Mapping) es el siguiente paso después del Normal Mapping, pues tiene en cuenta el ángulo con el que se observa a la superficie e introduce variaciones en la iluminación (Self Shadows). Sigue siendo un truco, pues la geometría de la superficie se mantiene intacta, pero el efecto está mejor conseguido. (Anónimo).

Parallax Mapping es un proceso de un solo paso que no tiene en cuenta la oclusión. Práctico para los juegos y parece más realista que Normal Mapping. La entrada de estructuras de datos es muy eficaz porque utiliza las técnicas de Normal y Bump Map como entrada.

3.2.2 Ventajas y Desventajas.

Ventajas:

- Al acercarnos a la textura desde distintos ángulos la sensación es mucho mejor que en los casos anteriores.
- Es soportada por las mejores tarjetas gráficas (NVIDIA/ATI).
- Alta velocidad de procesamiento.
- Buena capacidad de procesamiento.

Desventajas:

- Costo computacional Caro.
- No oclusión.
- Problemas con los mapas de altura.

3.3 Comparación entre las técnicas de gráficos computacionales.

Al realizar una comparación entre todas las técnicas de gráficos computacionales mencionadas anteriormente en este trabajo, tomando diferentes datos de interés para su utilización, implementación y resultado a obtener como son la visualización del resultado obtenido, el costo computacional, las tarjetas gráficas que las soportan con respecto a los dos tipos expuestos en este trabajo(ATI/NVIDIA) y la capacidad de procesamiento contra la velocidad de procesamiento se afirma que esta técnica de gráfico computacional es la más acorde y balanceada para ser utilizada obteniendo el resultado esperado por el usuario como se muestra en la siguiente tabla.

	Comparación Técnicas de Gráficos Computacionales				
	Bump Mapping	Normal Mapping	Parallax Mapping	Displacement Mapping	Relief Mapping
Visualización	2	3	4	5	4
Costo Computacional	Bajo Costo	Bajo Costo	Costosa	Mas Costosa	Costosa
Tarjetas que Soportan (NVIDIA/ATI)	Si/Si	Si/Si	Si/Si	Si/No	Si/Si
Capacidad Procesamiento	Poca	Poca	Normal	Alta	Alta
Velocidad Procesamiento	Rápida	Rápida	Rápida	Lenta	Lenta

Tabla 2: Comparación entre las Técnicas de Gráficos Computacionales.

En esta tabla se explica el porque de utilizar como solución propuesta la técnica de Parallax Mapping y no solo influye en la técnica propuesta sino que da las razones suficientes y necesarias para la utilización de la tarjeta grafica NVIDIA implicando también por consecuencia el nuevo lenguaje creado por su Corporación, que no es más que el Cg y su sucesor CgFx.

Para esto se le dio, a las diferentes características necesarias de este análisis, valores, analizando la optimización requerida para una mejor solución en la aplicación, implementación y desarrollo del shader de relieve a mostrar. Estas fueron delimitadas por las razones principales

que sujetan la elaboración de un software, las que se nombran seguidamente: el resultado a obtener (visualización), costo general de las operaciones necesarias computacionalmente a desarrollar (Costo Computacional), influencia en cuanto a hardware necesario para su realización y posterior utilización (Tarjetas que soportan (NVIDIA/ATI)) y por último y no menos significativa, el balance que se requiere en el código a presentar(Capacidad de Procesamiento y Velocidad de Procesamiento).

Al tratar de entender este análisis se puede ver que a la característica de visualización se le dio valores entre 2 y 5, en orden ascendente en cuanto a, como su nombre lo indica, lo que se muestra, la fidelidad de lo que el usuario puede ver como resultado de la aplicación de cada técnica de gráfico computacional que se ha investigado y que infiere en el relieve de los objetos virtuales a utilizar en los entornos de RA.

Pasando a la siguiente característica a evaluar, no es más que el Costo Computacional, se muestra, arrojados por las investigaciones pertinentes realizadas a las diferentes técnicas, resultados de forma que se puede evaluar en bajo costo, costosa y más costosa.

Otro de los descubrimientos realizados en la investigación son las técnicas que eran soportadas, que podrían ser utilizadas por una, por otra o simplemente por las dos tarjetas de las evaluadas en el análisis, siendo mostradas en la tercera característica a la cual se hace alusión.

Así también, en el desarrollo de las investigaciones, se encontró con dos de las características necesarias para realizar una buena evaluación como son: Capacidad de Procesamiento y Velocidad de Procesamiento, estas van de la mano en el tema del software y su rendimiento.

3.4 Creación del paquete contenedor (CgFx)

Para la implementación del shader de relieve se creó un paquete contenedor estándar que soporta la mayoría de las técnicas de representación de relieve a través de gráficos computacionales 3D, exceptuando el Displacement Mapping, está compuesto por un segmento

de código estándar *vertex shader* que se complementa con segmentos que implementan diferentes técnicas en el *fragment shader*.

El *vertex shader*, a través de la asignación de parámetros que le hace la aplicación, envía los valores necesarios a los diferentes *fragment shader* para su implementación, contando con diferentes cálculos necesarios para una u otra técnica, un ejemplo de esto se muestra a continuación en el *vertex shader* creado:

```
VertexOutput view_spaceVS(AppVertexData IN)
{
    VertexOutput OUT = (VertexOutput)0;
 //aislar la matriz WorldViewXf rotando solamente una parte:
    float3x3 modelViewRotXf;
    modelViewRotXf[0] = WorldViewXf[0].xyz;
    modelViewRotXf[1] = WorldViewXf[1].xyz;
    modelViewRotXf[2] = WorldViewXf[2].xyz;
    float4 Po = float4(IN.pos.xyz,1.0);
    OUT.hpos = mul(WvpXf,Po);
 //posición del vértice teniendo en cuenta el espacio (con el modelo de //transformaciones):
    OUT.vpos = mul(WorldViewXf,Po).xyz;
 //posición de luz teniendo en cuenta el espacio:
    float4 Lw = float4(Lamp0Pos.xyz,1);   // este punto en el espacio mundo      OUT.lightpos = mul(ViewXf,Lw);   // este punto teniendo en cuenta el espacio
```

```
//Vectores espacio tangente a la vista el espacio (con el modelo de //transformaciones):
    OUT.tangent = mul(modelViewRotXf,IN.tangent);
    OUT.binormal = mul(modelViewRotXf,IN.binormal);
    OUT.normal = mul(modelViewRotXf,IN.normal);
//Copia del color y coordenadas de la textura:
    OUT.color = IN.color;   // Actualmente, ignora todas las técnicas
    OUT.UV = TileCount * IN.txcoord.xy;
    return OUT;
}
```

Por otra parte, luego de obtener todas las coordenadas necesarias, son utilizadas por el fragment shader para su implementación dando lugar a la deformación de la textura como se muestra en el siguiente ejemplo de código de la técnica Parallax Mapping, insertado en el paquete junto a la implementación de las demás antes mencionadas:

```
float4 parallax_mapPS(VertexOutput IN) : COLOR
{
//direcciones de la luz y la vista:
    float3 Vn = normalize(IN.vpos);
    float3 Ln = normalize(IN.lightpos.xyz-IN.vpos);
    float2 uv = IN.UV;
//código parallax:
    float3x3 tbnXf = float3x3(IN.tangent,IN.binormal,IN.normal);
    float4 reliefTex = tex2D(ReliefSampler,uv);
    float height = reliefTex.w * 0.06 - 0.03;
```

```
    uv += height * mul(Vn,tbnXf).xy;
//mapa normales:
    float3 tNorm = reliefTex.xyz - float3(0.5,0.5,0.5);
//transformar tNorm para el espacio mundo:
    tNorm = normalize(tNorm.x*IN.tangent -
                      tNorm.y*IN.binormal +
                      tNorm.z*IN.normal);
    float3 texCol = tex2D(ColorSampler,uv).xyz;
//calcular los términos diffuse y specular:
    float att = saturate(dot(Ln,IN.normal));
    float diff = saturate(dot(Ln,tNorm));
    float spec = saturate(dot(normalize(Ln-Vn),tNorm));
    spec = pow(spec,PhongExp);
//computar el color final:
    float3 finalcolor = AmbiColor*texCol +
            att*(texCol*DiffColor.xyz*diff+SpecColor*spec);
    return float4(finalcolor.rgb,1.0);
}
```

En la siguiente figura se muestra el orden en que se comunica la aplicación, el *vertex* y *fragment shader,* terminando con la técnica y el paso que referencian la compilación del *vertex y el fragment shader*:

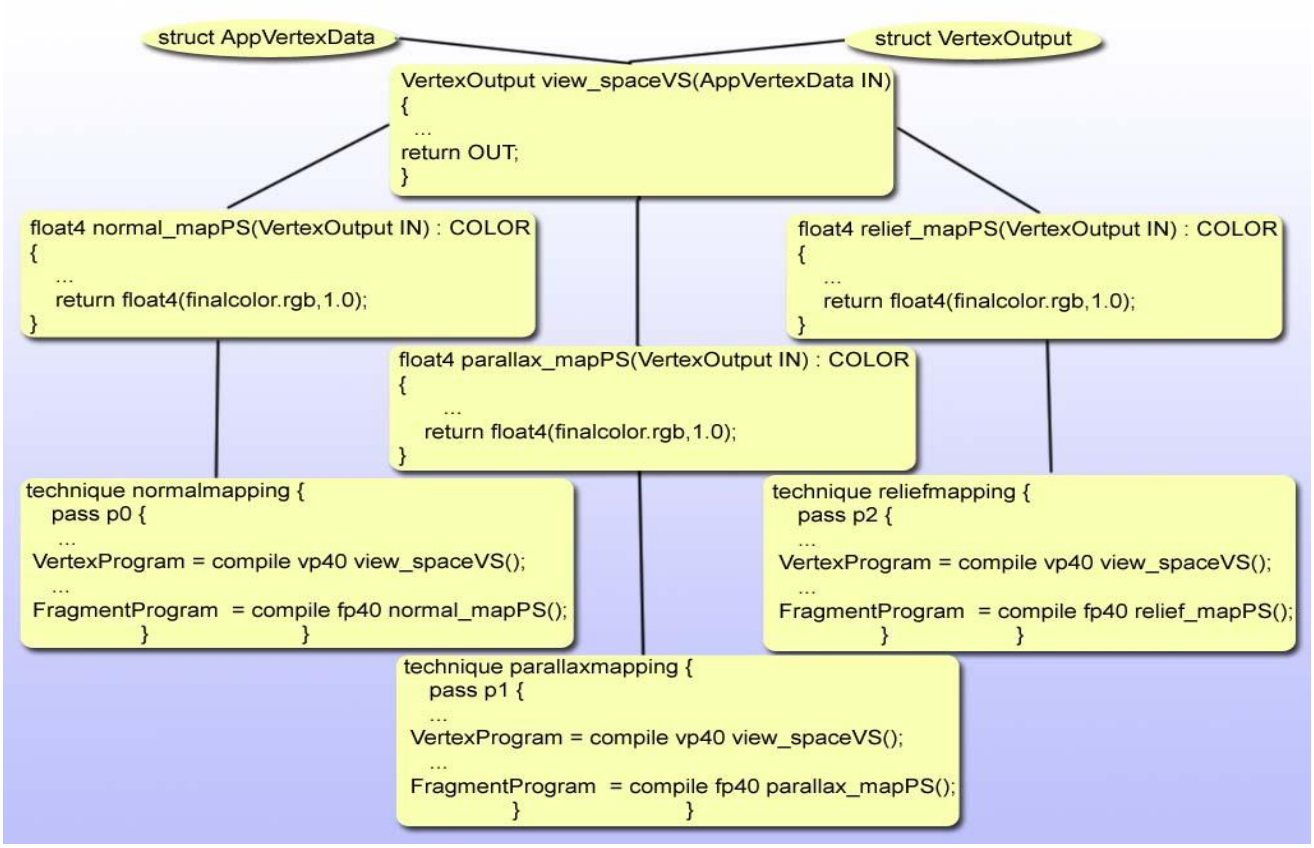

Ilustración 18: Modelo del orden de comunicación entre la aplicación, el vertex y el fragment shader.

Luego de analizar el código propuesto se puede llegar a la conclusión que la función que se implementa en el *vertex shader* puede ser utilizada por cualquier técnica de representación de relieve, mientras que el *fragment shader* va a ser modificado en dependencia de la técnica que se va a utilizar. Se debe señalar que el lenguaje de programación utilizado para la elaboración del *vertex y fragment shader* es el futurista CgFx, el cual permite el empaquetado de las funciones y como aspecto más importante, permite la reutilización de código y el agrupamiento de estas técnicas, las cuales pueden ser llamadas por separadas sin importar que se encuentren en el mismo paquete.

3.5 Herramientas a utilizar

Entre las herramientas que se disponen para la creación y visualización del Shader de Relieve encontramos, en orden de utilización:

3.5.1 FX Composer

El NVIDIA ® FX Composer software ™ es un paquete integrado de entorno de desarrollo el cual permite la creación de modernos Shaders con soporte para múltiples lenguajes. FX Composer permite a los desarrolladores crear Shaders en tiempo real de alto rendimiento con una optimización de una vista previa de funciones disponibles sólo a partir de NVIDIA. (NVIDIA, 2008)

Diseñado para crear Shaders, desarrollarlo y optimizarlo siendo más fácil para los programadores, FX Composer ofrece una intuitiva interfaz de usuario para personalizar por los artistas en una escena en particular. (NVIDIA, 2008)

FX Composer tiene una amplia gama de potentes capacidades, que van desde la edición de Shaders básica a la avanzada, ajustando el rendimiento. Lo que sigue es una lista de estas características. (NVIDIA, 2008)

Características

- Prototipos de sistema de partículas con varias plantillas comunes: fuego, humo, fuegos artificiales, y fuentes.
- Soporta la industria estándar de archivos con formatos 3d: COLLADA, OBJ, X, 3DS y FBX.
- Trabaja directamente con COLLADA FX, CgFX y Shader HLSL para crear varias técnicas y pases.

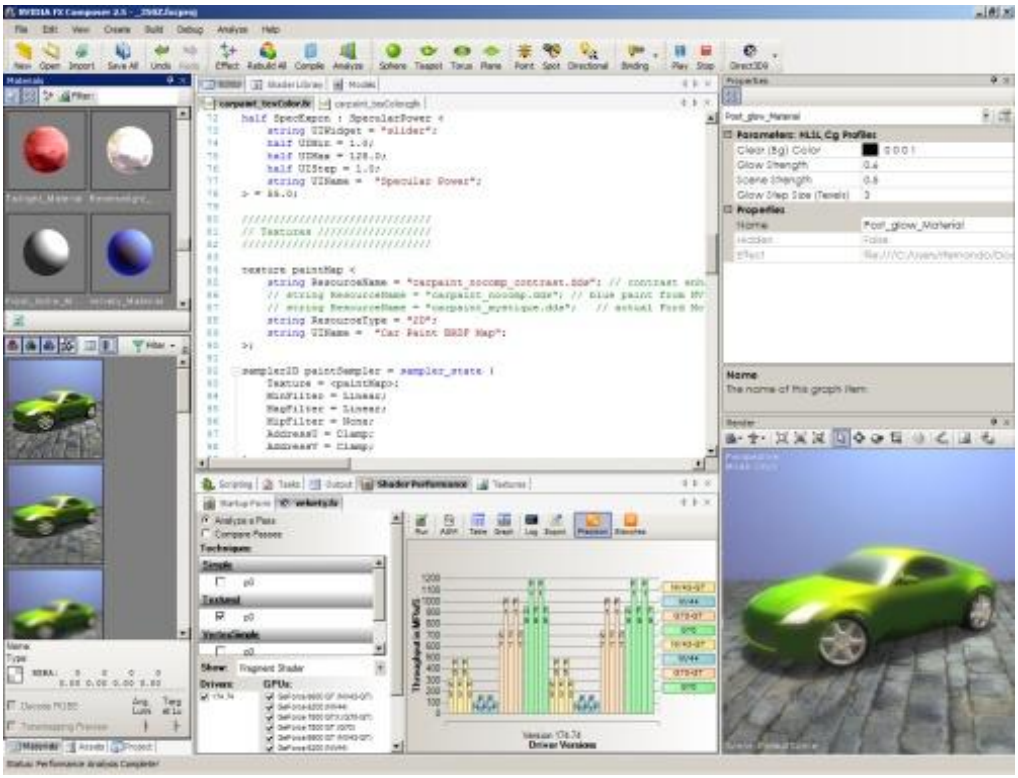

Ilustración 19: Interfaz Fx Composer. (NVIDIA, 2008)

3.5.2 3D Estudio Max

Autodesk 3D Studio Max es un programa de creación de gráficos y animación 3D desarrollado por Autodesk Media & Entertainment (Anteriormente conocidos como Discreet y Kinetix). **3D Max** es uno de los programas de animación 3D más utilizados. Dispone de una sólida capacidad de edición, una omnipresente arquitectura de plugins y una larga tradición en plataformas Microsoft Windows. 3D Max es utilizado en mayor medida por los desarrolladores de videojuegos, aunque también en el desarrollo de proyectos de animación como películas o anuncios de televisión, efectos especiales y en arquitectura. (Unav)

Este programa es uno de los mejores modeladores 3D masivo, más orientado a videojuegos, con el que se han hecho enteramente juegos como la saga Tomb Raider, la saga Splinter Cell, y la

mayoría de los juegos de Ubisoft. Incluido el magnífico Crysis y FarCry. Con él también se crearon películas enteras de animación como ToyStory, Los Increíbles, Buscando a los Robinson, y parte de FinalFantasy. (Unav)

La utilización de 3D Studio Max permite al usuario la fácil visualización y representación de los modelos, así como su exportación y salvado en otros formatos distintos del que utiliza el propio programa. (Unav)

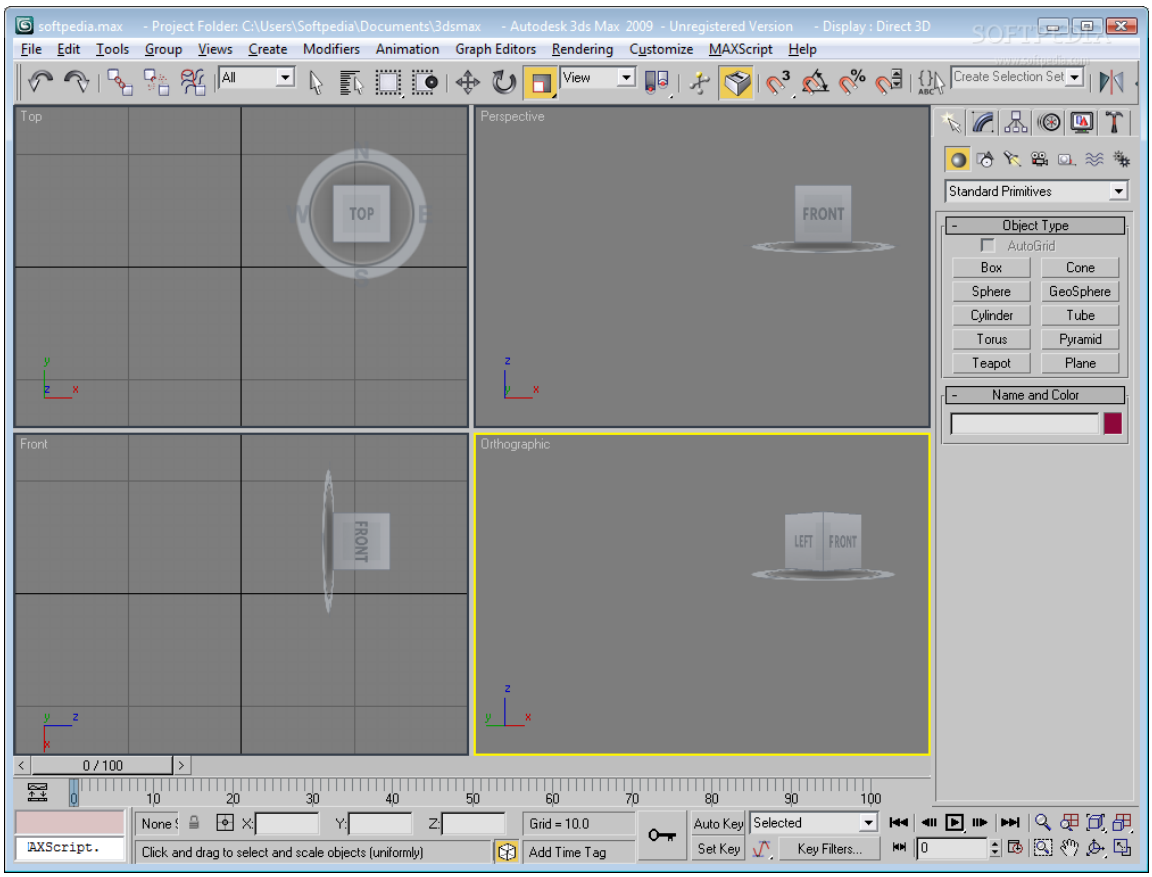

Ilustración 20: 3D Estudio Max. (Google)

3.5.3 Plugins

Un **plugin** es una aplicación informática que interactúa con otra aplicación para aportarle una función o utilidad específica, generalmente muy específica, como por ejemplo servir como driver (controlador) en una aplicación, para hacer así funcionar un dispositivo en otro programa.(Codepixel). Esta aplicación adicional es ejecutada por la aplicación principal. Los plugins típicos tienen la función de reproducir determinados formatos de gráficos, reproducir datos multimedia, codificar/decodificar emails, filtrar imágenes de programas gráficos, entre otras.

Actualmente son una forma de expandir los programas de forma modular, de manera que se puedan añadir nuevas funcionalidades sin afectar a las ya existentes ni complicar el desarrollo del programa principal. (Codepixel).

Los plugins que serán utilizados en la creación del Shader de Relieve, siendo este impregnado en los objetos 3D y visualizado en un entorno de Realidad Aumentada son:

3.5.3.1 Collada Max

Collada Max es un plugin importador y exportador para 3ds Max. El plugin te da la posibilidad de integrar Shaders complejos en su escena usando el Cg de alto nivel de sombreado de idiomas, entre ellos el estándar de NVIDIA Cg programas y CgFX efectos. (Khronos, 2005)

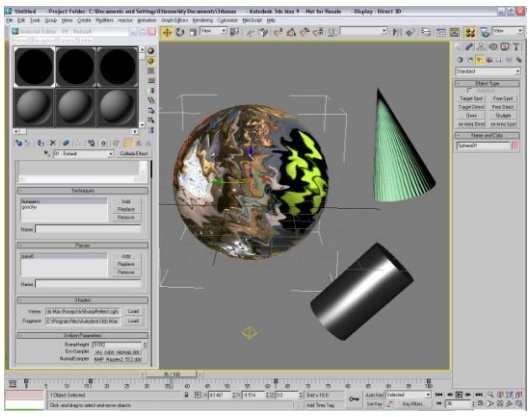

Ilustración 21: El ColladaEffect Plugins en acción. (Khronos, 2005)

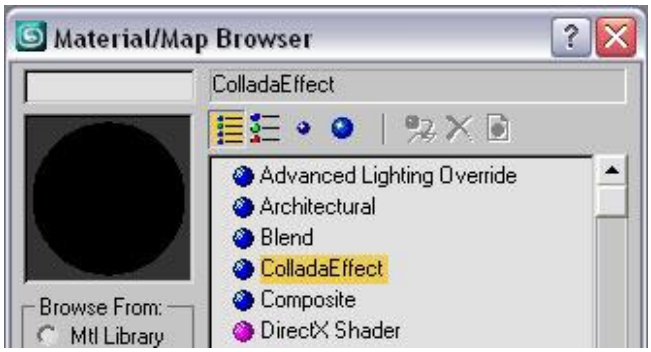

Ilustración 22: ColladaEffect en el navegador de materiales de 3D Estudio Max. (Khronos, 2005)

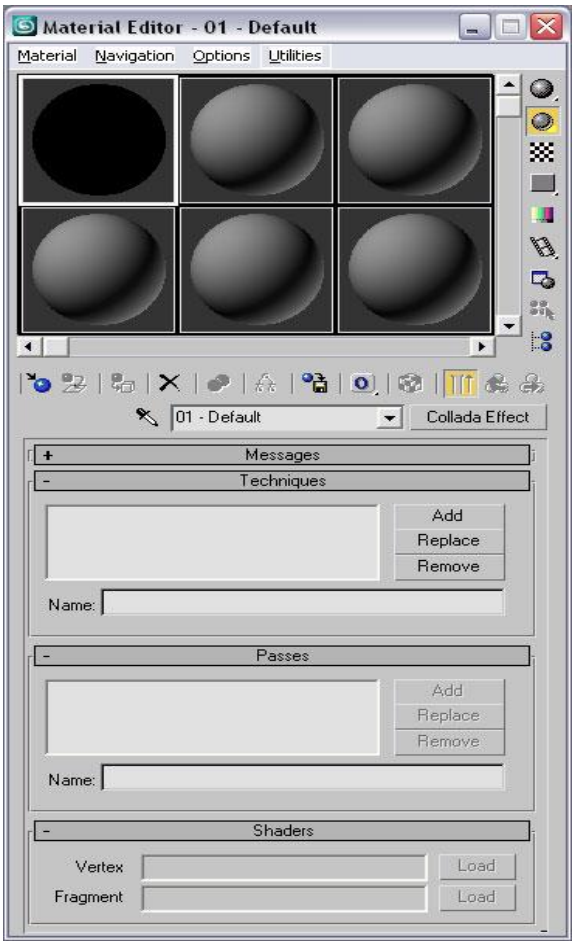

Ilustración 23: La interfaz de usuario ColladaEffect estática en el Material Editor. (Khronos, 2005)

3.5.3.2 oFusion

El oFusion es un plugin para 3D Max que permite desarrollar contenidos desde el editor, y trabajar de forma totalmente integrada entre artistas y desarrolladores. La verdad es que es muy completo, con integración de materiales y Shaders en el propio max, un viewport propio del motor gráfico Ogre 3D, y una librería que permite cargar directamente las escenas creadas con este plugin. (Codepixel). Incluye las herramientas que usted necesita para crear y una vista previa de contenido de alta calidad para ser utilizado con el motor Ogre. (Devmaster)

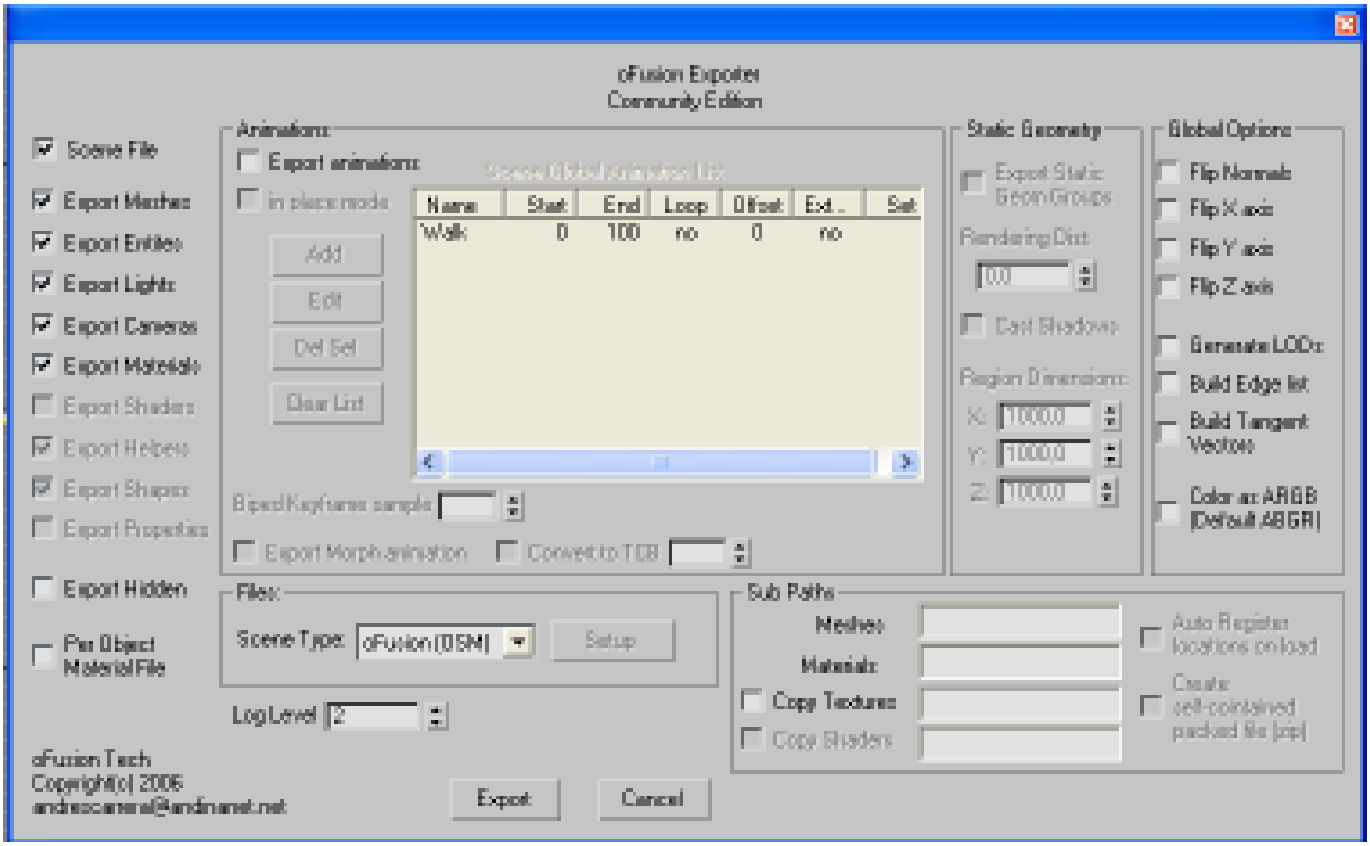

Ilustración 24: Exportador de oFusion para 3D Studio Max. (Devmaster)

3.5.3 Cg Toolkit

El galardonado Cg Toolkit proporciona un compilador para el lenguaje Cg 2,0, bibliotecas de tiempo de ejecución para su uso con ambos líderes gráficos API, bibliotecas de tiempo de ejecución para CgFX, ejemplo de aplicaciones, una amplia documentación y le permitirá incorporar impresionantes efectos interactivos en 3D. (Developer)

Los componentes incluyen:

• NVIDIA Cg Compiler Release 2,0.

• Cg / CgFX en tiempo de ejecución para las bibliotecas de OpenGL y Direct3D.

• Cg Manual de Usuario.

• Documentación para el idioma Cg con especificaciones, tiempo de ejecución API, Cg biblioteca estándar, CgFX estados, y los perfiles de Cg. (Developer)

• Cg ejemplos.

Cg 2,0 apoya una amplia gama de sistemas operativos de escritorio (Windows 2000, XP, Vista, Mac OS X Tiger y para Leopard, 32-bit y 64-bit Linux x86, x86 y Solaris) para que su Shader habilitado para las aplicaciones pueda ser usado, sin importar su elección de sistema operativo o gráficos API. (Developer)

Cg 2,0 introduce nuevas características y mejoras:

• Nuevos perfiles de DirectX 9 (hlslv y hlslf) traducir de HLSL a Cg.

• Documentación de actualizaciones, incluyendo librería estándar de Cg y estados estándar de CgFX.

• Mejorar la generación de código del compilador.

• Mejoras de rendimiento en tiempo de ejecución.

• Compatibilidad con Cg 1,5.

3.5.4 Mr. Planet

Mr.Planet es una aplicación de Realidad Aumentada con un diálogo sencillo con el usuario. El usuario es capaz de relacionar modelos 3D realizados con su editor preferido (AutoCad, 3D Studio,...) con una gran variedad de patrones de Realidad Aumentada. La herramienta permite modificar el escalado, rotar el modelo y trasladarlo en relación al patrón con un sencillo menú de opciones. Además estos modelos también pueden contener animaciones que podrán reproducirse desde la aplicación. (Planet, 2006)

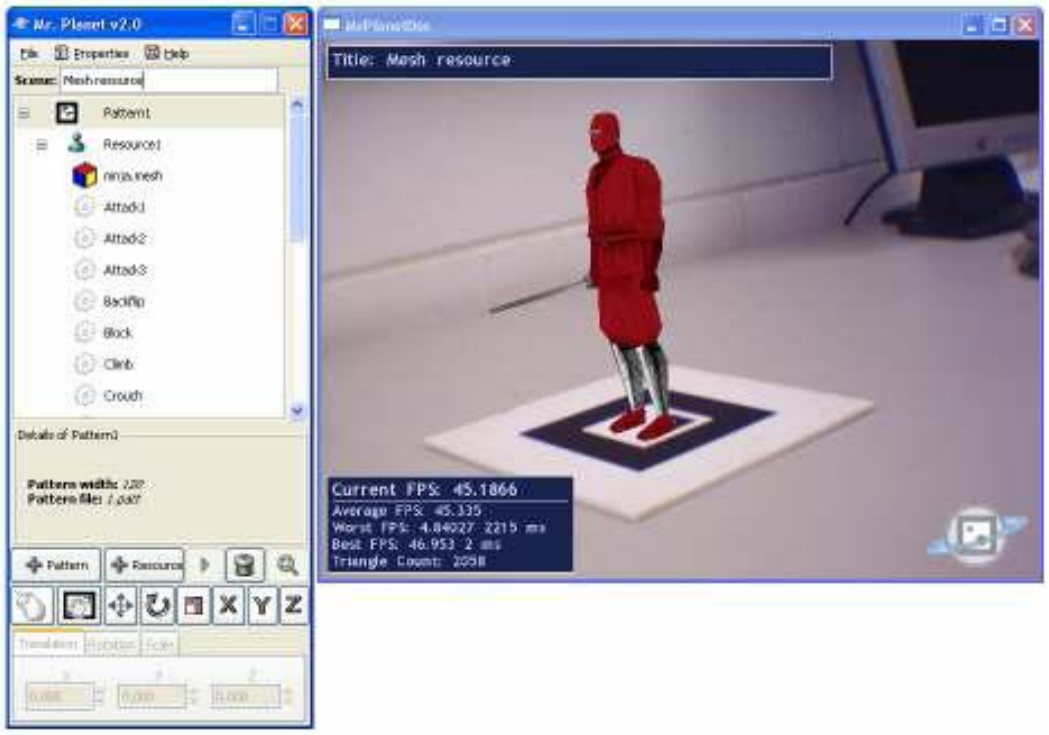

Ilustración 25: Escenario de un recurso de tipo Mesh. (NVIDIA, 2008)

A raíz de estas explicaciones acerca de la técnica y las herramientas óptimas para la creación de un Shader de Relieve, se ha expuesto la propuesta de solución para la resolución del problema científico, siendo este la falta de fotorealismo en los objetos 3D que conforman las escenas de Realidad Aumentada.

CONCLUSIONES

Para llevar a cabo esta investigación se analizó como desarrollar un Shader de Relieve para que sea aplicado a los objetos 3D que participan de los entornos de Realidad Aumentada.

Se analizaron los problemas existentes en la RA llegando a la conclusión, de que la solución para estos es el desarrollo de Shaders de relieve, ofreciendo mayor realismo a sus entornos.

Se resumieron las potencialidades de los lenguajes de programación para la creación de Shader, en este caso los lenguajes Cg. y CgFX., fundamentales para la programación con *píxel* y *vertex Shader*.

En el transcurso de esta investigación se estudiaron las técnicas de gráficos computacionales 3D a utilizar en el desarrollo de Shaders de Relieve, llegando a emplear la técnica Parallax Mapping debido al balance entre todas las características esenciales que nos brinda, siendo esta la más óptima de estas técnicas.

Para obtener un Shader de Relieve fue necesario investigar acerca de las herramientas más efectivas para su correcta visualización como el 3D Max, plugins como Collada y oFusion y el Mr. Planet.

También se estudiaron las potencialidades de las tarjetas gráficas que permiten visualizar los Shader de Relieve, en este caso las ATI y las NVidia.

Por tanto se ha cumplido el objetivo del trabajo dando lugar a futuras investigaciones y a la realización y desarrollo de estos Shaders, resolviendo con esto el principal problema que presentan los entornos de RA.

RECOMENDACIONES

Finalizado el trabajo de diploma donde se investigó acera de la aplicación de Shader de Relieve a los objetos 3D que participan en las escenas en un entorno de RA para ofrecer mayor realismo se recomienda:

- Se haga uso de la técnica propuesta para solucionar los problemas de realismo existentes en los entornos de RA.
- La Facultad 5 continúe trabajando en esta rama más detalladamente en el desarrollo e implementación de Shaders de Relieve para su uso en los diferentes simuladores existentes utilizando como lenguajes de programación los creados por la corporación NVidia los cuales son Cg y CgFX.
- Se continúe investigando respecto al tema de Shaders de Relieve para su aplicación tanto en juegos como en simuladores.
- Se haga utilización del paquete contenedor para la asignación de Shaders de Relieve a los objetos 3D modelados en el 3D Estudio Max.

BIBLIOGRAFÍA

1. Azuma, R., Baillot, Y., Behringer, R., Feiner, S., Julier, S., & MacIntyre, B. (2001). *Recent Advances in Augmented Reality.*

2. Coorporation, N. (2005). *User Guide Fx Composer.*

3. Dempski, K., & Viale, E. (2004). *Advanced Lighting and Materials with Shaders.*

4. Esteban, P., Restrepo, J., Trefftz, H., Jaramillo, J. E., & Alvarez, N. (2005). *La realidad aumentada: un espacio para la comprensión de conceptos del cálculo en varias variables.*

5. Gumbau, J., González, C., & Chover, M. *Fast GPU-based normal map generation for simplified models.*

6. Haller, M. (2003). *Shader Programming Cg, nVIDIA´s shader language.*

7. Haller, M. (2003). *Shader Programming CgFX, OpenGL 2.0.*

8. Kessenich, J. (2004). *Features of the OpenGL Shading Language.*

9. Kilgard, M. J. (2005). *A Follow-up Cg Runtime Tutorial for Readers of The Cg Tutorial*.*
10. Landis, M. *Normal Map Create.*

11. Liarokapis, F., Mourkoussis, N., Petridis, P., Rumsey, S., Lister, P. F., & White, M. (2002). *An Interactive Augmented Reality System for Engineering Education.*

12. NVidia, C. (2004). *Nvidia GPU Programing Guide.*

13. Pulido, E., Quirós, R., Fabregat, G., & Rubio, M. *UN SISTEMA PARA LA AUMENTACIÓN DE INFORMACIÓN EN OBJETOS APILADOS.*

14. Tatarchuk, N. (2005). *Practical Dynamic Parallax Occlusion Mapping.*

15. Vallino, J. R. (1998). *Interactive Augmented Reality.*

16. Welsh, T. (2004). *Parallax Mapping with Offset Limiting: A PerPixel Approximation of Uneven Surfaces.*

.

REFERENCIAS

1. **Alonso Hernández, Lic. Lidiexy y Durán Benejam, MSc. Mayra. 2007.** *Aplicación de la Realidad Aumentada en el proceso de formación del universitario.* 2007.

2. **Anónimo.** **h**ttp://www.meristation.com/v3/des_articulo.php?pic=HRD&id=cw44cf819b159be&idj=&idp=&tipo=art&c=1&pos=1. [En línea]

3. Activamente. (n.d.). http://www.activamente.com.mx/vrml/. Retrieved from http://www.activamente.com.mx/vrml/.

4. Arpa-solutions. (n.d.). http://www.arpa-solutions.net/arpa-realidad-aumentada.html. Retrieved from http://www.arpa-solutions.net/arpa-realidad-aumentada.html.

5. Barbosa, P. (2006). http://www.neoteo.com/pixel-shaders-y-vertex-shaders.neo.

6. Bump. (n.d.). *http://www.blacksmith-studios.dk*.

7. **Codepixel.** http://www.codepixel.com/content/view/4275/2/. [En línea]

8. **Developer.** http://developer.nvidia.com/object/cg_toolkit.html. [En línea]

9. **Devmaster.** http://www.devmaster.net/news/index.php?storyid=929. [En línea]

10. Difementes. (n.d.). *www.difementes.com/realidadvirtual/mecanismosbasicos.html*. Retrieved from www.difementes.com/realidadvirtual/mecanismosbasicos.html.

11. Dreijer, S. *Bump Mapping Using CG (3rd Edition).*

12. Gamasutra. *Gamasutra - Book Excerpt - Using Vertex Texture Displacement for Realistic Water Rendering.*

13. García, I. M. (2004). *Uso de tecnologias en el hardware gráfico en el apoyo al realismo en entornos virtuales arquitectónicos.*

14. Glassner, A. (1989). An Introduction to Ray Tracing. San Francisco: Morgan & Kaufmann.

15. Google. (n.d.). *www.google.com*. Retrieved from www.google.com.

16. **Guinot, Jerome.** *Vertex Displacement Mapping using GLSL.*

17. *Hand Interaction in Augmented Reality.* **McDonald, C. 2003.** 2003.

18. http://www.arpa-solutions.net. *http://www.arpa-solutions.net.* [En línea]

19. Jesús Gumbau, C. G. *Fast GPU-based normal map generation for simplified*.

20. Khronos, J. (2005). *COLLADA™ an open standard for the interactive entertainment industry.*

21. Medicin. (n.d.). *http://www.medicin.com.ar/realidad_virtual.asp*. Retrieved from http://www.medicin.com.ar/realidad_virtual.asp.

22. *Neoteo.* **Reggiani, Federico y Micchielli, Lucas. 2007.** 2007, Vol. 13.

23. Neoteo. (n.d.). http://www.neoteo.com/pixel-shaders-y-vertex-shaders.neo. Retrieved from http://www.neoteo.com/pixel-shaders-y-vertex-shaders.neo.

24. Normal. (n.d.). *http://amber.rc.arizona.edu/lw/docs/NMCManual.pdf* .

25. NVIDIA, C. (2008). *User's Guide FX Composer 2.5.*

26. Parallax. (n.d.). *GameDev_net - A Closer Look At Parallax Occlusion Mapping.htm*.

27. **Planet, Mr. 2006.** *Mr. Planet Sofware de Realidad Aumentada.* Universidad de Rovira y Virgilia : s.n., 2006.

28. Policarpo, F., Oliveira, M. M., & Comba, J. L. *Real-Time Relief Mapping on Arbitrary Polygonal Surfaces.*

29. Relief. (n.d.). *Relief Mapping\RTM.html*.

30. *Revista Universidad EAFIT.* **Toro, A. A. 2005.** 2005.

31. Shader. (n.d.). http://es.wikipedia.org/wiki/Shader. Retrieved from http://es.wikipedia.org/wiki/Shader.

32. **Unav.** http://www.unav.es/cti/manuales/3DStudioMax/indice.html. [En línea]

33. Valdés, Y. N., & Torres, L. M. (2007). *MÓDULO DE EFECTOS VISUALES PARA MOTORES DE REALIDAD VIRTUAL.*

34. **Vallino, James R. 1998.** *Interactive Augmented Reality.* University of Rochester,Rochester, New York : s.n., 1998.

35. Welsh, T. (2004). *Parallax Mapping with Offset Limiting: A PerPixel.*

36. 100cia. (n.d.). *http://www.100cia.com/recursos/enciclopedia/GPU*. Retrieved from http://www.100cia.com/recursos/enciclopedia/GPU.

ANEXOS

Anexo1. Realidad Virtual Inmersiva.

Anexo2. Realidad Virtual No Inmersiva

GLOSARIO

Bajo poli: Es un término utilizado por los creadores de videojuegos, se dice que tiene bajo poli a los personajes tridimensionales generados por computadoras que carecen de polígonos.

Bump, Normal, Parallax, Displacement y Relief Mapping: Son técnica de gráfico computacional 3D utilizada para crear Shaders de Relieve.

Fotorealismo: Se utiliza generalmente para dar a entender que un objeto o lugar tiene presencia de realidad, que es parecido al mundo real.

Iluminación Per-Vertex: Se emplea generalmente para referirse a un conjunto de métodos que se utilizan para calcular la iluminación en cada vertex de una imagen.

***Normal*:** Es un vector que presenta todo vértice en una figura geométrica, en ese caso un objeto 3D.

Paralelaje: Es la desviación angular de la posición aparente de un objeto, dependiendo del punto de vista elegido.

Pixel Shader: Son pequeños programas que se encargan del procesamiento de pixeles. (Neoteo).

Per-Pixel: Se emplea generalmente para referirse a un conjunto de métodos que se utilizan para calcular la iluminación en cada pixel de una imagen.

Pipeline: Unidad de cálculo especializada. Calculan polígonos, sus coordenadas y posición. Existen pipelines de geometría (3D) y de imagen (2D).

Polígono: En geometría es tradicionalmente una figura plana que está limitada por una ruta cerrada, compuesto por una secuencia finita de líneas.

Rasterización: Es el proceso por el cual una imagen descrita en un formato gráfico vectorial se convierte en un conjunto de pixeles o puntos para ser desplegados en un medio de salida digital, como una pantalla de computadora o una impresora electrónica.

Realidad Aumentada: Es una nueva rama de interfaces donde los elementos reales conviven con los elementos virtuales, que sirven para aportar información adicional a los primeros. Es una tecnología totalmente innovadora, ligada a la Realidad Virtual (RV), aunque diferente en varios aspectos. (Arpa-solutions).

Realidad Virtual*:* La realidad virtual es una representación de las cosas a través de medios electrónicos, que nos da la sensación de estar en una situación real en la que podemos interactuar con lo que nos rodea. (Activamente).

Shader: En informática gráfica, es el término usado para referirse a un conjunto de instrucciones capaces de ser ejecutadas por el procesador gráfico (GPU), en otras palabras, es un programa para la tarjeta gráfica. Su uso esta vinculado comúnmente a aplicar efectos a la representación, usar la tarjeta gráfica para cálculos pesados como por ejemplo la detección de colisiones y otras aplicaciones. (Shader).

Texel*:* (contracción del inglés *texture element*, o también *texture pixel*) es la unidad mínima de una textura aplicada a una superficie. (Glassner, 1989).

Vertex Shader*:* Pequeños programas que se encargan del procesamiento de vértices. (Neoteo).