UNIVERSIDAD NACIONAL
AUTÓNOMA DE MÉXICO

FACULTAD DE INGENIERÍA

PABELLÓN EN LÍNEA DE MÉXICO EN LA EXPOSICIÓN UNIVERSAL SHANGHÁI 2010.

T E S I S

QUE PARA OBTENER EL TÍTULO DE:
INGENIERO EN COMPUTACIÓN

P R E S E N T A:

JOSÉ FELIPE DE JESÚS CONTRERAS FLORES

DIRECTOR DE TESIS
ING. JOSÉ LARIOS DELGADO

CIUDAD UNIVERSITARIA, MÉXICO D.F., ENERO 2013

Agradecimientos.

A Dios por crear la materia y las leyes que la rigen y otorgar el libre albedrío, permitiendo así mi desarrollo físico y espiritual y por mostrarme su existencia.

A mis papás Fortino Contreras Martínez y María del Rosario Flores Carapia por la vida, enseñanzas, cuidados y momentos felices.

A mis hermanos Fortino y Santiago por su compañía, consejos y apoyo.

A Brenda Isabel Trejo Mendiola, mi novia y futura compañera de mi vida, por su amor, dulzura, apoyo y cariño.

A mis tías Tere y Pana por su cariño y apoyo.

A Arturo Villanueva, José Luis Martínez, Ángel Angulo, Irving Álvarez, Fernando San Juan, Luis Calzada, Jacobo Abarca, Rubén Elizalde y Gerardo Payán por su amistad, consejos, enseñanzas y anécdotas que podemos contar.

A los compañeros y amigos Facultad de Ingeniería y de DGSCA por sus enseñanzas y amistad.

Al Ing. José Francisco Salgado Rodríguez por su apoyo, enseñanza, consejos y ayuda durante la primera parte de esta tesis.

Al Ing. José Larios Delgado por su ayuda en la conclusión de esta tesis.

A los sinodales María del Carmen Ramos, Alberto Templos y Noé de Jesús Romero por su corrección y consejos.

A los profesores de toda mi trayectoria escolar, buenos y malos, por dicha formación académica y que con su actuar me enseñaron a discernir entre lo que se debe seguir y lo que no.

A la UNAM y a la Facultad de Ingeniería por mi formación como ingeniero.

A la sociedad mexicana quien pagó gran parte de mi formación académica.

Índice.

1. Objetivos.

- Utilizar las herramientas de la ingeniería para realizar un recorrido virtual en línea que muestre el pabellón de México en la Exposición Universal Shanghái 2010.

- Cumplir con los requerimientos técnicos y visuales del comité organizador de la Exposición universal en el rubro de pabellón virtual.

- Modificar el recorrido virtual para que pueda cargar otros modelos, siguiendo un formato especificado.

2. Introducción.

Una Exposición universal puede definirse como una especie de museo pedagógico y técnico que permite reconocer el progreso de la civilización en su conjunto. Son manifestaciones de carácter público en las que se exhiben productos industriales, comerciales o artísticos provenientes de países de todo el mundo[1]. Las exposiciones universales no tienen una periodicidad específica, sino que se convocan de acuerdo a las inquietudes globales sobre diversos temas, tales como los adelantos tecnológicos o la resolución de los problemas que enfrentan las sociedades contemporáneas.

Del 1º de mayo al 31 de octubre de 2010 se llevó a cabo la Exposición Universal Shanghái 2010, cuyo tema fue "Mejor ciudad, mejor vida", en la cual México participó con un pabellón. En dicho pabellón se expusieron varias piezas museográficas del acervo cultural mexicano, por ejemplo cuadros de Frida Kahlo, José Chávez Morado, un altar a la Virgen de los Dolores, esculturas mayas. En otras áreas del pabellón se instalaron recursos de multimedios para mostrar parte de la biodiversidad de nuestro país, grupos de personas de las diferentes zonas de México, venta de comidas típicas en el restaurante y artesanías en la tienda de recuerdos. En el exterior del pabellón se presentaron grupos de música folklórica, moderna, mariachis y bailes típicos para mostrar la cultura mexicana.

Los organizadores en China pidieron a los países participantes que se construyera un pabellón virtual en línea para extender dicha exposición a personas en todo el mundo que no pudieran asistir. El organismo encargado de organizar la construcción del pabellón físico y virtual en México fue la secretaría de turismo del gobierno federal a través de ProMéxico[2] quien encargó a la UNAM con apoyo de la Dirección General de Servicios de Cómputo Académico[3] la construcción del pabellón virtual.

El pabellón virtual debía ser lo más apegado al pabellón físico edificado en Shanghái para proveer así a los visitantes virtuales la experiencia más vívida posible de la visita al pabellón de México y darle la oportunidad de acceder a otro espacio que le permitiera abordar con mayor profundidad la riqueza visual y cultural de algunas ciudades mexicanas así como su problemática urbana.

El proyecto completo incluye una parte en dos dimensiones y una parte en tres dimensiones realizada con un motor de juegos[4] (virtools) tuvo como objetivo ponerlo en línea y como restricciones que fuera atractivo, que tardara poco tiempo en descargarse y que no necesitara una computadora muy potente para poder ejecutarse.

[1] http://www.expo2010mexico.com.mx/es/PM_historia.htm

[2] ProMéxico es el Organismo del Gobierno Federal encargado de coordinar las estrategias dirigidas al fortalecimiento de la participación de México en la economía internacional.

[3] Actualmente Dirección General de Cómputo y de Tecnologías de la Información y Comunicación.

[4] El proyecto completo puede verse en la página http://shanghai2010.unam.mx/ssize/welcome.html

La tesis está acotada a la parte tridimensional con el motor de juegos. Primero se realizaron los modelos tridimensionales por computadora utilizando planos, fotos y esquemas de la edificación y de las piezas a realizar. En segundo lugar se pusieron texturas a los modelos para darles un terminado más parecido a los objetos físicos. En tercer lugar se hizo una exportación del programa de modelado por computadora al motor de juegos. En cuarto lugar se realizó la programación en el motor de juegos y por último se buscó optimizar el recorrido virtual.

Para realizar el modelado por computadora se empleó el programa 3D studio max porque se contaba con conocimientos previos y licencia para su utilización. El motor de juegos empleado fue virtools debido a restricción de los organizadores en China.

3. Antecedentes.

Índice del capítulo.

En este capítulo se verán a grandes rasgos los temas principales que componen la tesis para dar una introducción a la parte teórica que la sustenta.

3.1 Exposiciones universales.

3.1.1 ¿Qué son las exposiciones universales?

Las exposiciones universales se distinguen de las ferias y exposiciones comerciales internacionales por la finalidad que las impulsa, así como por su periodicidad. Las exposiciones universales no tienen una periodicidad específica, sino que se convocan de acuerdo a las inquietudes globales sobre diversos temas, tales como los adelantos tecnológicos o la resolución de los problemas que enfrentan las sociedades contemporáneas.

Las exposiciones universales pueden definirse como una especie de museo pedagógico y técnico que permite reconocer el progreso de la civilización en su conjunto. Son manifestaciones de carácter público en las que se exhiben productos industriales, comerciales o artísticos.

La primera exposición universal tuvo lugar en Londres en 1851, dirigida por la Sociedad de Arte, con el fin de mostrarle al mundo el progreso que suponía la Revolución Industrial.

3.1.2 Participación mexicana en exposiciones universales.

México ha estado presente en 29 exposiciones universales alrededor del mundo, en las que ha participado de forma activa con la presentación de propuestas viables para la solución de los principales problemas de índole ambiental, urbana y social.

La primera participación oficial del país en una exposición universal tuvo lugar en Filadelfia 1876. La actuación de México obedeció a su necesidad de enunciarse como nación independiente. Del 1 de mayo al 31 de octubre de 2010 Shanghái fue la sede de la exposición universal más grande de la historia, cuyo tema fue “Mejor Ciudad, Mejor Vida”. México estuvo presente en la Expo con un pabellón que dio cabida a su propuesta para encontrar soluciones sustentables para los principales retos urbanos.

El pabellón mexicano albergó un restaurante de alta cocina nacional, una tienda de artesanías, un área dedicada a la promoción turística del país, e hizo énfasis especial en la promoción del comercio y la inversión extranjera a través de un Centro de Negocios.

3.1.3 Expo Universal Shanghái 2010.

Más de 190 países y casi 50 organizaciones internacionales se reunieron en la Expo Universal 2010 para mostrarle al mundo sus ciudades y los retos que éstas tendrán que resolver para fomentar entornos más sustentables.

La exposición universal mostró cómo crear sociedades “eco-amigables” (que tienen como prioridad el respeto al medio ambiente y el aprovechamiento de los recursos naturales) y propiciar el desarrollo sustentable de los seres humanos para así mejorar la calidad de vida en las ciudades.

3.2 Modelado 3D por computadora.

El término modelado 3D por computadora se refiere a trabajos de arte gráfico que son creados con ayuda de computadoras y programas especiales 3D.

Un modelo 3D difiere de uno 2D principalmente por la forma en que ha sido generado. Este tipo de modelos se originan mediante un proceso de cálculos matemáticos sobre entidades geométricas tridimensionales producidas en una computadora, y cuyo propósito es conseguir una proyección visual en dos dimensiones para ser mostrada en una pantalla o impresa en papel.

Figura 1. Objetos de vidrio elaborados por computadora.

En general, el arte de los modelos 3D es similar a la escultura o la fotografía, mientras que el arte de los modelos 2D es análogo a la pintura. En los programas de modelado por computadora esta distinción es a veces difusa: algunas aplicaciones 2D utilizan técnicas 3D

para alcanzar ciertos efectos como iluminación, mientras que algunas aplicaciones 3D primarias hacen uso de técnicas 2D.

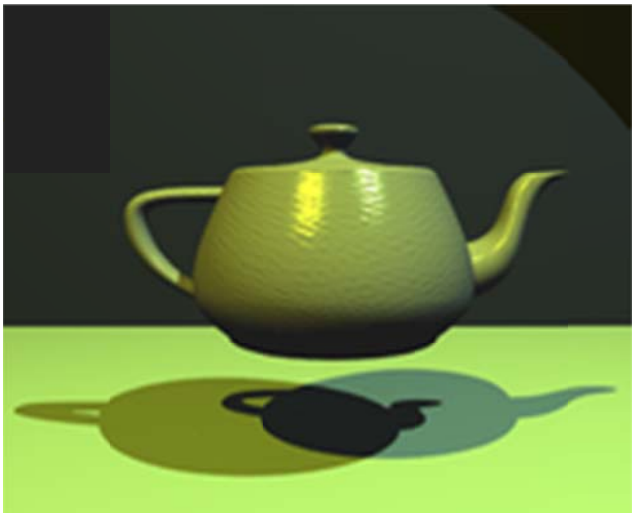

Figura 2. Tetera dibujada mediante gráficos 3D.

3.3 Ambientes virtuales.

"La realidad virtual es un medio compuesto por simulaciones de computadora interactivas que reaccionan a la posición y acciones del usuario y producen retroalimentación en uno o más sentidos, generando la sensación de estar inmerso o presente en una simulación"[5].

Un ambiente virtual es una simulación por computadora que proporciona información a uno o varios de nuestros sentidos: visión, sonido, tacto y gusto, con el propósito de que el usuario se sienta inmerso en un mundo que reacciona ante sus acciones. A diferencia de una película tridimensional, donde la información se integra en una secuencia de imágenes definidas de antemano y en la que el participante no puede intervenir; o de una aplicación multimedia, donde la interacción está limitada a seleccionar la secuencia en que se despliegan los objetos bidimensionales, un ambiente virtual es naturalmente tridimensional, dinámico y cambiante según los movimientos o peticiones del usuario, quien puede explorar y experimentar de acuerdo con las situaciones generadas como combinación de su interacción con el mundo virtual y la retroalimentación que éste, a su vez, le proporciona al participante[6].

La realidad virtual puede ser de dos tipos: inmersiva y no inmersiva. La realidad virtual no inmersiva o también llamada realidad virtual de mesa es aquélla que se crea cuando el participante explora diversos ambientes haciendo uso de los dispositivos de hardware

[5] "Understanding Virtual Reality", William R. Sherman.

[6] Ramos Nava, María del Carmen. Enter@te en línea año 2, número 24, noviembre de 2003. Dirección General de Servicios de Cómputo Académico, UNAM. http://www.enterate.unam.mx/Articulos/2003/noviembre/realivirt.htm

comunes: mouse, monitor, tarjeta de sonido y bocinas. Ejemplo de ésta son las aplicaciones que utilizan Internet y lenguajes como el VRML con objeto de interaccionar en tiempo real con diferentes personas en espacios y ambientes tridimensionales, sin la necesidad de dispositivos adicionales en la computadora.

Este tipo de aplicaciones son llamativas para los usuarios, pues no requieren de hardware especializado y, por lo tanto, son de fácil acceso; no obstante, tienen como inconveniente que los sentidos de los participantes se distraen por eventos ajenos a la simulación, como por ejemplo, salir del campo de visión de la simulación al mover la cabeza, o utilizar el mouse que no es un dispositivo de interacción natural, con lo que se percibe de manera inconsciente que el ambiente virtual no constituye una realidad.

Los métodos inmersivos buscan crear la sensación de encontrarse dentro de un ambiente específico; para lograrlo se generan simulaciones con la mayor calidad posible de despliegue, junto con formas naturales de interacción donde se utilizan sistemas sofisticados de alta calidad de despliegue con efecto de profundidad como cascos o proyectores de alta resolución, equipo de cómputo capaz de controlar la simulación, el despliegue, los dispositivos y la interacción a una velocidad adecuada con objeto de que el usuario tenga una respuesta rápida.

Los dispositivos de interacción regularmente se basan en sistemas de captura de los movimientos del usuario, de tal manera que se realizan de forma natural, sin tener que concentrarse en cambiar protocolos de interacción, como sucede al emplear el mouse o teclado.

4. Marco teórico.

Índice del capítulo.

En este capítulo se tratarán las bases teóricas que se utilizarán para el desarrollo del recorrido virtual. Se tratarán las definiciones y los procesos necesarios para realizarlo.

4.1 Computación gráfica.

La computación gráfica (CG) o gráficos por computadora es el campo de la informática visual, donde se utilizan computadoras tanto para generar imágenes visuales sintéticamente como integrar o cambiar la información visual y espacial probada del mundo real.

Este campo puede ser dividido en varias áreas: Visualización 3D en tiempo real (a menudo usado en juegos de vídeo), animación por computadora, captura de video, edición de efectos especiales (a menudo usado para películas y televisión), edición de imagen, y modelado (a menudo usado para ingeniería y objetivos médicos).

El primer avance en la CG fue la utilización del tubo de rayos catódicos. Hay dos tipos de gráficos 2d: vector y gráficos *raster*. El gráfico de vector almacena datos geométricos precisos, topología y estilo como posiciones de coordenada de puntos, las uniones entre puntos (para formar líneas o trayectos) y el color, el grosor y posible relleno de las formas. La mayor parte de los sistemas de vectores gráficos también pueden usar primitivas geométricas de forma estándar como círculos y rectángulos etc. En la mayor parte de casos una imagen de vectores tiene que ser convertida a una imagen de trama o *raster* para ser vista.

Los gráficos de tramas o *raster* (llamados comúnmente Mapa de bits) son una rejilla bidimensional uniforme de pixeles. Cada pixel tiene un valor específico como por ejemplo brillo, transparencia en color o una combinación de tales valores. Una imagen de trama tiene una resolución finita de un número específico de filas y columnas. Hoy se utiliza a menudo una combinación de trama y gráficos vectorizados en formatos de archivo compuestos.

4.2 Diseño asistido por computadora.

Uno de los mayores usos de los gráficos por computadora se encuentra en los procesos de diseño, particularmente en arquitectura e ingeniería, aunque ahora muchos productos se diseñan por computadora.

Generalmente, se conoce como CAD (*Computer Aided Design*, diseño asistido por computadora) o CADD (*Computer Aided Drafting and Design*). Estos métodos se emplean rutinariamente en el diseño de edificios, automóviles, aeronaves, barcos, naves espaciales, computadoras, telas, electrodomésticos y muchos otros productos.

En algunas aplicaciones de diseño, los objetos se visualizan primero en su modelo en malla de alambre (*wire frame*) mostrando su forma general y sus características internas. El modelo en malla de alambre permite a los diseñadores ver rápidamente los efectos de los

ajustes interactivos que se hacen en las formas sin esperar a que la superficie completa de los objetos esté completamente generada. Las figuras siguientes proporcionan ejemplos de modelos de malla de alambre en aplicaciones de diseño.

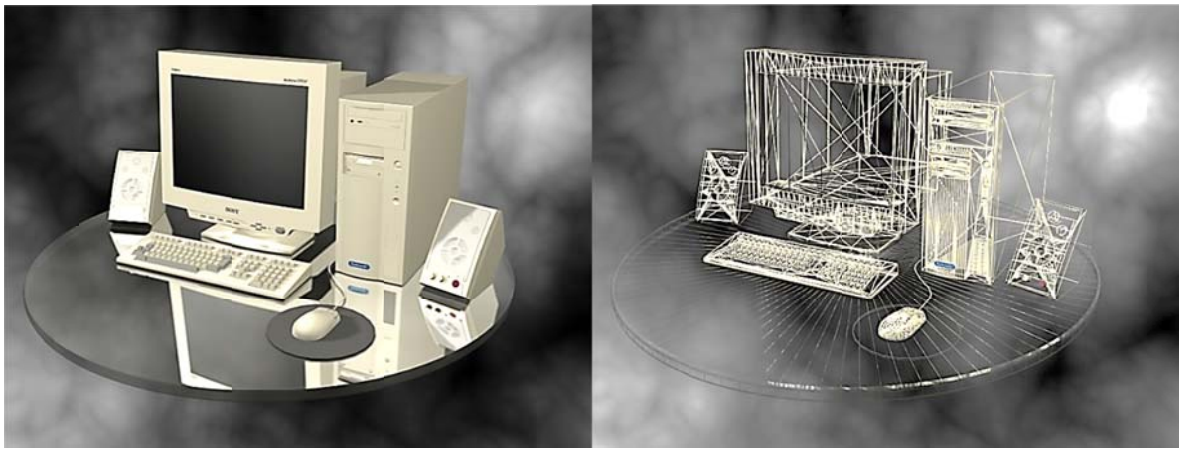

FIGURA 3. Modelo terminado y en malla de alambre de una computadora.

Animaciones en tiempo real del modelo de malla de alambre de las formas son muy útiles para comprobar rápidamente el funcionamiento de un vehículo o un sistema.

Dado que las imágenes en modelo de malla de alambre no muestran las superficies, los cálculos para cada segmento de animación pueden realizarse rápido para así producir movimientos suaves en la pantalla. También el modelo de malla de alambre permite ver su interior y observar los componentes internos durante el movimiento.

Cuando los diseños del objeto están completos o casi completos, se aplican condiciones reales de iluminación y de representación de superficies para producir visualizaciones que mostrarán la apariencia final del producto. Ejemplos de esto se proporcionan en la figura 4.

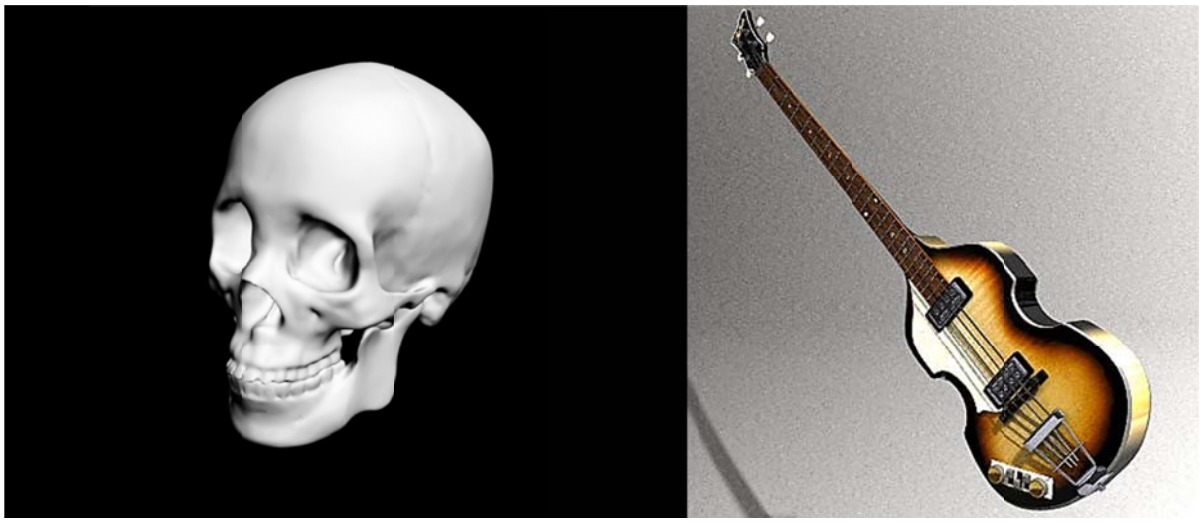

Figura 4. Cráneo y bajo Hofner con texturas e iluminación finales.

El proceso de fabricación también va unido a la descripción computarizada de los objetos diseñados, de modo que el proceso de fabricación del producto puede ser automatizado, utilizando métodos que son conocidos como CAM (*Computer - Aided Manufacturing*, fabricación asistida por computadora). El plano de una placa de circuito, por ejemplo, puede transformarse en la descripción individualizada de los procesos necesarios para construir el circuito electrónico.

4.3 Procesamiento de imágenes.

La modificación o interpretación de imágenes existentes como fotografías o provenientes de TV es conocida como procesamiento de imágenes. Aunque los métodos empleados en los gráficos por computadora y el procesado de imágenes se traslapan, las dos áreas están dedicadas a operaciones fundamentales diferentes. En los gráficos por computadora, la computadora se utiliza para crear una imagen. Por otra parte las técnicas de procesamiento de imágenes se utilizan para mejorar la calidad de un dibujo, analizar las imágenes o reconocer patrones visuales para aplicaciones robotizadas. Sin embargo, los métodos de procesamiento de imágenes se utilizan frecuentemente en los gráficos por computadora, y los métodos de los gráficos por computadora se aplican también en el procesamiento de imágenes.

Por lo general, antes de procesar una imagen o una fotografía, primero se almacena en un archivo de imagen. Entonces es cuando se pueden aplicar los métodos digitales de reorganización de las partes de la imagen, para resaltar separaciones de color, o mejorar la calidad del sombreado. En la figura 5 se da un ejemplo de los métodos de procesamiento de imagen para el realce de la calidad de una imagen.

Figura 5. Mejora de calidad de una foto de una placa de auto.

4.4 Gráficos 3d por computadora.

El término gráficos 3D por computadora (*3D computer graphics*) se refiere a trabajos gráficos que son creados con ayuda de computadoras y programas especiales 3D. En general, el término puede referirse también al proceso de crear dichos gráficos, o el campo de estudio de técnicas y tecnología relacionadas con los gráficos 3D.

Un gráfico 3D difiere de uno 2D principalmente por la forma en que ha sido generado. Este tipo de gráficos se originan mediante un proceso de cálculos matemáticos sobre entidades geométricas tridimensionales producidas en una computadora, y cuyo propósito es conseguir una proyección visual en dos dimensiones para ser mostrada en una pantalla o impresa en papel.

Los polígonos tridimensionales son la parte principal de los gráficos 3d realizados en computadora. Como consiguiente, la mayoría de los motores de gráficos de 3D están basados en el almacenaje de puntos (por medio de 3 coordenadas dimensionales X, Y, Z), líneas que conectan aquellos grupos de puntos, las caras son definidas por las líneas, y luego una secuencia de caras crean los polígonos tridimensionales. Adicional al manejo de los polígonos, los programas de gráficos 3D emplean herramientas de sombreado (*Shading*), texturizado (*texturing*) y rasterización[7](En referencia a mapas de bits).

4.5 Fases para la creación de elementos gráficos 3D.

4.5.1 Modelado.

La etapa de modelado consiste en ir dando forma a objetos individuales que luego serán usados en la escena[8]. Existen diversos tipos de geometría para modelar como NURBS[9] y modelado poligonal o Subdivisión de Superficies (*Subdivision Surfaces*). Además, aunque menos usado, existe otro tipo llamado "modelado basado en imágenes" o "*image based modeling*" (IBM); consiste en convertir una fotografía a 3D mediante el uso de diversas técnicas, de las cuales, la más conocida es la fotogrametría.

4.5.1.1 Coordenadas, vértices, aristas y polígonos.

Sistemas de coordenadas de referencia.

Para describir una escena, primero es necesario seleccionar un sistema de coordenadas cartesianas adecuado, denominado sistema de coordenadas de referencia del mundo, que puede ser bidimensional o tridimensional.

Después se describen los objetos de la escena proporcionando sus especificaciones geométricas en términos de la posición dentro de las coordenadas del mundo. Por ejemplo, definimos un segmento de línea recta proporcionando la posición de los dos puntos extremos, mientras que un polígono se especifica proporcionando el conjunto de posiciones de sus vértices. Estas coordenadas se almacenan en la descripción de la escena, junto con

[7]La *rasterización* es el proceso por el cual una imagen descrita en un formato gráfico vectorial se convierte en un conjunto de pixeles para ser desplegados en un medio de salida digital.
[8]Escena es el conjunto de objetos 3D, luces, texturas y animaciones, es similar a las escenas cinematográficas.
[9] NURBS (Non Uniform Rational B-splines) es un modelo matemático para generar curvas y superficies.

otras informaciones acerca de los objetos, como por ejemplo su color y su extensión de coordenadas, que son los valores x, y y z máximos y mínimos para cada objeto. Un conjunto de coordenadas de extensión también se denomina “recuadro de contorno” del objeto.

A continuación, los objetos se visualizan pasando la información de la escena a las rutinas de visualización que identifican las superficies visibles y asignan los objetos a sus correspondientes posiciones en el monitor de vídeo. El proceso de conversión de líneas almacena la información sobre la escena, como por ejemplo los valores de color, en las apropiadas ubicaciones dentro del búfer de imagen, y los objetos de la escena se muestran en el dispositivo de salida.

Coordenadas de pantalla.

Las ubicaciones sobre un monitor de vídeo se describen mediante “coordenadas de pantalla” que son números enteros y que se corresponden con las posiciones de píxel dentro del búfer de imagen. Los valores de las coordenadas de píxel proporcionan el “número de línea de exploración” (el valor A) y el “número de columna” (el valor v) dentro de una línea de exploración. Los procesos hardware, como el de refresco de pantalla, normalmente direccionan las posiciones de píxel con respecto al extremo superior izquierdo de la pantalla. Las líneas de exploración se identifican por tanto comenzando por 0, en la parte superior de la pantalla, y continuando hasta un cierto valor entero, A_{max}, en la parte inferior de la pantalla, mientras que las posiciones de píxel dentro de cada línea de exploración se numeran desde 0 a v_{max} , de izquierda a derecha.

Los algoritmos de líneas de exploración para las primitivas gráficas utilizan las descripciones de coordenadas que definen los objetos para determinar la ubicación de los píxeles que hay que mostrar. Por ejemplo, dadas las coordenadas de los extremos de un segmento de línea, un algoritmo de visualización debe calcular las posiciones para los píxeles comprendidos en la línea definida entre los dos puntos extremos. Puesto que una posición de píxel ocupa un área finita en la pantalla, es preciso tener en cuenta ese tamaño finito de los píxeles dentro de los algoritmos de implementación. Una vez identificadas las posiciones de los pixeles para un objeto, hay que almacenar los valores de color apropiados dentro del búfer de imagen.

Vértice.

Es el lugar geométrico donde concurren dos o más curvas, no tiene dimensión pero sí un lugar en el espacio.

Arista.

Es en geometría el segmento de recta donde intersecan dos planos. Por extensión también se conoce con este nombre al segmento común que tienen dos caras adyacentes de un poliedro, y que forman al estar en contacto.

Polígonos.

Un polígono es una figura geométrica formada por segmentos consecutivos no alineados, llamados lados.
Otro elemento útil, además de los puntos, los segmentos lineales y las curvas, para la descripción de los componentes de una imagen son las áreas rellenas con algún color homogéneo o patrón. Un elemento de imagen de este tipo se denomina normalmente área rellena. La mayoría de las veces, las áreas rellenas se utilizan para describir las superficies de los objetos sólidos, pero también resultan útiles en diversas aplicaciones.
Asimismo, las regiones rellenas suelen ser superficies, principalmente polígonos. Pero en términos generales hay muchos tipos de formas posibles para una región del dibujo que podamos querer rellenar con algún determinado color. La Figura 6 ilustra algunos ejemplos de áreas rellenas.

FIGURA 6. Áreas rellenas de color homogéneo especificadas con diversas fronteras. Una región circular rellena, un área rellena delimitada por una polilínea cerrada y un área rellena especificada mediante una frontera irregular curva.

Aunque pueden existir áreas rellenas de cualquier forma, las bibliotecas gráficas no suelen soportar la especificación de formas de relleno arbitrarias. La mayoría de las rutinas de biblioteca requieren que las áreas de relleno se especifiquen en forma de polígonos. Las rutinas gráficas pueden procesar más eficientemente los polígonos que otros tipos de áreas de relleno, porque las fronteras de los polígonos se describen mediante ecuaciones lineales. Además, la mayoría de las superficies curvas pueden aproximarse razonablemente bien mediante una serie de “parches” poligonales, de la misma forma que una línea curva puede aproximarse mediante un conjunto de segmentos lineales. Y cuando se aplican efectos de iluminación y procedimientos de sombreado de superficies, la superficie curva aproximada puede mostrarse con un grado de realismo bastante bueno. La aproximación de una superficie curva mediante caras poligonales se denomina en ocasiones *teselación* de la superficie, o ajuste de la superficie mediante una malla poligonal.
La visualización de tales tipos de figuras puede generarse muy rápidamente mediante vistas en malla de alambre, que sólo muestran las aristas de los polígonos con el fin de proporcionar una indicación general de la estructura de la superficie. Posteriormente, el

modelo en malla de alambre puede sombrearse para generar una imagen de una superficie material con aspecto natural. Los objetos descritos con un conjunto de parches de superficie poligonales se suelen denominar objetos gráficos estándar o simplemente objetos gráficos.

Áreas de relleno poligonales.

Desde el punto de vista matemático, un polígono se define como una figura plana especificada mediante un conjunto de tres o más puntos, denominados vértices, que se conectan en secuencia mediante segmentos lineales, denominados bordes, o aristas del polígono. Además es necesario que las aristas del polígono no tengan ningún punto en común aparte de los extremos. Así, por definición, un polígono debe tener todos sus vértices en un mismo plano y las aristas no pueden cruzarse. Como ejemplos de polígonos podemos citar los triángulos, los rectángulos, los octágonos y los decágonos. Algunas veces, cualquier figura plana con un contorno de tipo polilínea cerrada se denomina también polígono, y si además sus aristas no se cortan se le denomina polígono estándar o polígono simple.

Tablas de polígonos.

Normalmente, los objetos de una escena se describen como conjuntos de caras poligonales de superficie. La descripción de cada objeto incluye la información de coordenadas que especifica la geometría de las caras poligonales y otros parámetros de la superficie como el color, la transparencia, las propiedades de reflexión de la luz. A medida que se introduce la información correspondiente a cada polígono, los datos se colocan en tablas que se utilizarán en el subsiguiente procesamiento como visualización y manipulación de los objetos de la escena. Estas tablas de datos de los polígonos pueden organizarse en dos grupos: tablas geométricas y tablas de atributos. Las tablas de datos geométricos contienen coordenadas de los vértices y parámetros para identificar la orientación espacial de las superficies poligonales. La información de atributos de un objeto incluye parámetros que especifican el grado de transparencia del objeto y la reflectividad y características de textura de su superficie.

Los datos geométricos de los objetos de una escena se pueden ordenar cómodamente en tres listas; una tabla de vértices, una tabla de aristas y una tabla de caras de la superficie, "Los valores de coordenadas de cada vértice del objeto se almacenan en la tabla de vértices. La tabla de aristas contiene punteros que hacen referencia a la tabla de vértices y que permiten identificar los vértices de cada arista del polígono. Por su parte, la tabla de caras de la superficie contiene punteros que hacen referencia a la tabla de aristas, con el fin de identificar las aristas que definen cada polígono. Este esquema se ilustra en la Figura 7 para dos caras poligonales adyacentes de la superficie de un objeto.

Además, se pueden asignar a los objetos individuales y a sus caras poligonales componentes unos identificadores de objeto y de cara para poder efectuar las referencias más fácilmente.

Enumerar los datos geométricos en tres tablas, como en la Figura 7, permite hacer referencia cómodamente a los componentes individuales (vértices, aristas y caras de la superficie) de

cada objeto. Asimismo, el objeto puede visualizarse de manera eficiente utilizando los datos de la tabla de aristas para identificar los contornos de los polígonos.

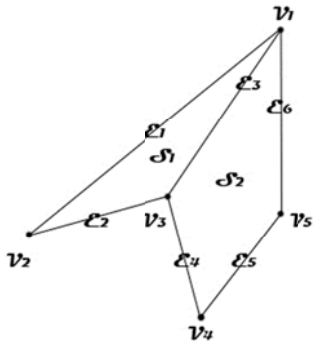

Tabla de vértices.	
V_1 :	x_1, y_1, z_1
V_2 :	x_2, y_2, z_2
V_3 :	x_3, y_3, z_3
V_4 :	x_4, y_4, z_4
V_5 :	x_5, y_5, z_5

Tabla de aristas.	
E_1 :	V_1, V_2
E_2 :	V_2, V_3
E_3 :	V_1, V_3
E_4 :	V_3, V_4
E_5 :	V_4, V_5
E_6 :	V_1, V_5

Tabla de caras de la superficie.	
S_1 :	E_1, E_2, E_3
S_2 :	E_3, E_4, E_5, E_6

FIGURA 7. Representación en tabla de los datos geométricos para dos caras poligonales adyacentes de una superficie, formadas por seis aristas y cinco vértices.

Podemos añadir información adicional a las tablas de datos de la Figura 7 para poder extraer más rápidamente la información. Por ejemplo, podríamos expandir la tabla de aristas para incluir punteros inversos que hagan referencia a la tabla de caras de la superficie, con el fin de poder identificar más rápidamente las aristas comunes existentes entre los polígonos. Esto resulta particularmente útil para los procedimientos de renderización que necesitan variar con suavidad el sombreado de la superficie al cruzar una arista desde un polígono a otro. De forma similar, la tabla de vértices podría expandirse para hacer referencia a las aristas correspondientes, con el fin de extraer más rápidamente la información.

Caras poligonales anteriores y posteriores.

Puesto que usualmente tratamos con caras poligonales que encierran un objeto interior, es necesario distinguir entre las dos caras de cada superficie. La cara de un polígono que apunta hacia el interior del objeto se denomina cara posterior, mientras que la cara visible es la cara anterior. La identificación de la posición de los puntos en el espacio con relación a las caras anterior y posterior de un polígono es una de las tareas básicas que deben llevarse a cabo en muchos algoritmos gráficos, como por ejemplo a la hora de determinar la visibilidad de los objetos. Todo polígono está contenido en un plano infinito que divide el espacio en dos regiones.

Todo punto que no se encuentre en el plano y que esté situado del lado de la cara anterior de un polígono se considerará que está delante (o fuera) del plano y, por tanto, fuera del objeto. Cualquier punto que esté del lado de la cara posterior del polígono se encontrará detrás (o dentro) del plano. Un plano que esté detrás (dentro) de todos los planos correspondientes a los polígonos de la superficie estará dentro del objeto.

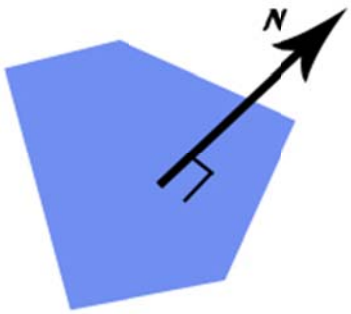

FIGURA 8. El vector normal N para un plano descrito por la ecuación Ax + By + Cz + D es perpendicular al plano y tiene como componentes cartesianas(A, B. C).

La orientación de la superficie de un polígono en el espacio puede describirse mediante el vector normal del plano que contiene dicho polígono, como se muestra en la Figura 8. Este vector normal a la superficie es perpendicular al plano y apunta en una dirección que va desde el interior del plano hacia el exterior, es decir, desde la cara trasera del polígono hacia la cara delantera.

4.5.1.2 Transformaciones geométricas.

Una posición tridimensional, expresada en coordenadas homogéneas, se representa como un vector columna de cuatro elementos. Así, cada operando de la transformación geométrica es una matriz de 4 por 4, que premultiplica un vector columna de coordenadas. Cualquier secuencia de transformaciones se representa como una matriz simple, formada por la concatenación de matrices para las transformaciones individuales de la secuencia. Cada matriz sucesiva en una secuencia de transformación se concatena a la izquierda de las matrices de transformación previas.

Traslación.

Una posición P = (x,y,z) en un espacio tridimensional, se traslada a la posición P' = (x',y',z') añadiendo las distancias de traslación x, y, z, a las coordenadas cartesianas de P.

Podemos expresar estas operaciones de traslación tridimensionales en matrices de la forma de la Ecuación 1. Las posiciones de coordenadas, P y P' se representan en coordenadas homogéneas con matrices columna de cuatro elementos y el operador de traslación T es una matriz de 4 por 4:

Un objeto se traslada en tres dimensiones, transformando cada una de las posiciones de coordenadas de definición para el objeto, y reconstruyendo después el objeto en la nueva localización. Para un objeto representado como un conjunto de superficies poligonales, trasladamos cada vértice de cada superficie (Figura 9) y volvemos a mostrar las caras del polígono en las posiciones trasladadas.

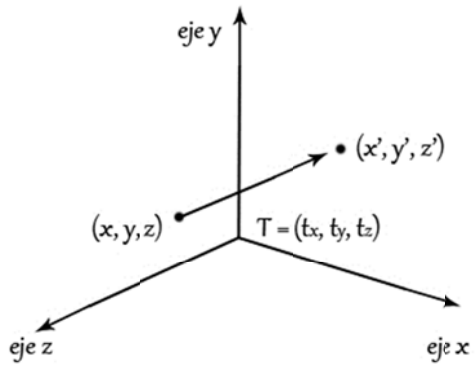

FIGURA 9. Movimiento de una posición de coordenadas con el vector de traslación $T = (t_x, t_y, t_z)$.

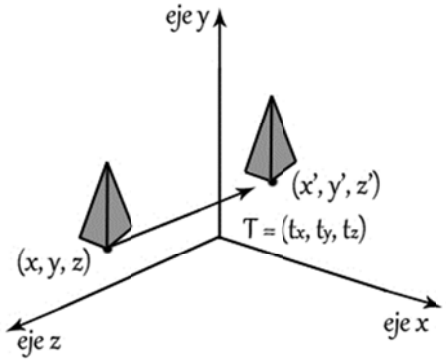

FIGURA 10. Cambio de la posición de un objeto tridimensional usando el vector de traslación T.

$$\begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & t_x \\ 0 & 1 & 0 & t_y \\ 0 & 0 & 1 & t_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} \quad \text{..... (1)}$$

ó **P ' = T · P** ..... *(2)*

La inversa de una matriz de traslación tridimensional se obtiene si negamos las distancias de traslación t_x, t_y, t_z. Esto produce una traslación en la dirección opuesta, y el producto de la matriz de traslación y su inversa es la matriz identidad.

Rotación.

Podemos rotar un objeto sobre cualquier eje en el espacio, pero la forma más fácil de llevar a cabo una rotación de ejes, es aquella que es paralela a los ejes de coordenadas cartesianos. También, podemos usar combinaciones de rotaciones de ejes de coordenadas (con las traslaciones apropiadas) para especificar una rotación sobre cualquier otra línea en el espacio.

Por convenio, los ángulos de rotación positivos producen rotaciones en el sentido contrario al de las agujas del reloj sobre un eje de coordenadas, asumiendo que estamos mirando en la dirección negativa a lo largo de dicho eje de coordenadas (Figura 11).

Rotaciones de ejes de coordenadas tridimensionales.

Para rotar un objeto tomando como eje de rotación al eje z las ecuaciones son las siguientes:

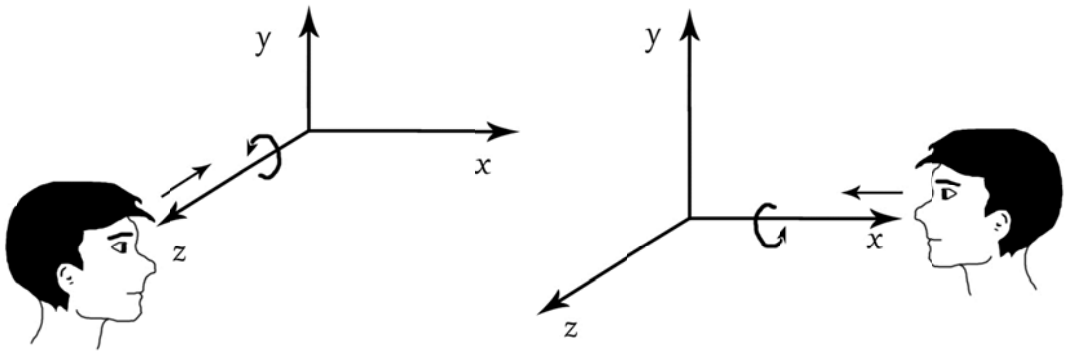

FIGURA 11. Las Rotaciones positivas alrededor de un eje de coordenadas se realizan en el sentido contrario a las agujas del reloj, cuando se está mirando a lo largo de la mitad positiva de los ejes con respecto al origen.

$$x' = x\cos\theta - y\sin\theta$$

$$y' = x\sin\theta + y\cos\theta \quad \text{..... (3)}$$

$$z' = z$$

El parámetro θ especifica el ángulo de rotación sobre el eje z, y los valores de la coordenada-z no se pueden cambiar con esta transformación. En la forma de coordenadas homogéneas, las ecuaciones para la rotación tridimensional del eje-z son:

$$\begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix} = \begin{bmatrix} \cos\theta & -\sin\theta & 0 & 0 \\ \sin\theta & \cos\theta & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} \quad \text{.....(4)}$$

y las podemos escribir de manera más compacta como:

$$P' = R_z(\theta) \cdot P \quad \text{.....(5)}$$

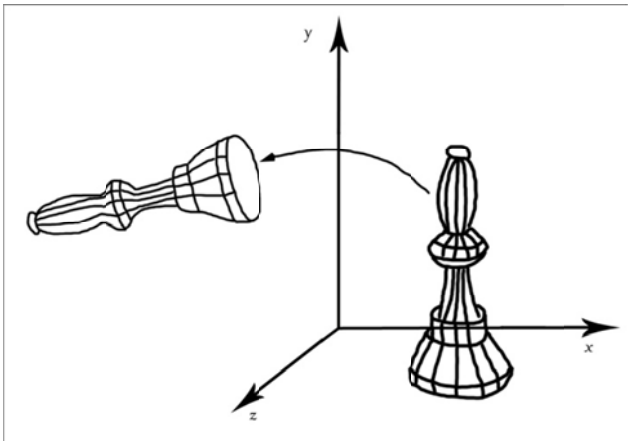

FIGURA 12. Ilustra la rotación de un objeto alrededor del eje z.

Las ecuaciones de transformación para rotaciones alrededor de los otros dos ejes de coordenadas pueden obtenerse con una permutación cíclica de los parámetros de coordenadas x, y y z en las Ecuaciones 3:

$$x \rightarrow y \rightarrow z \rightarrow x \quad \text{..... (6)}$$

Así, para obtener las transformaciones de rotación del eje-x y el eje-y, sustituimos cíclicamente x por y, y por z, y z por x, tal como se ilustra en la Figura 13.

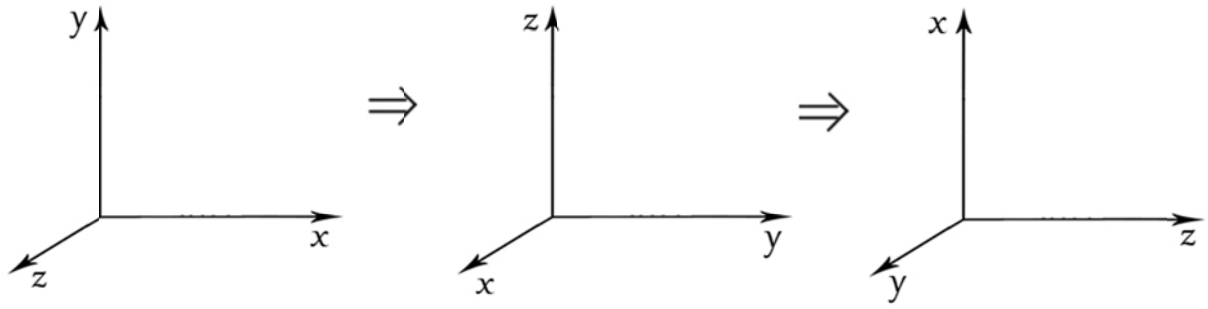

FIGURA 13. Permutación cíclica de los ejes de coordenadas cartesianas para producir los tres juegos de ecuaciones de rotación de ejes de coordenadas.

Sustituyendo las permutaciones de 6 por las Ecuaciones 3, obtenemos las ecuaciones para una rotación del eje x:

$$y' = y\cos\theta - z\sin\theta$$

$$z' = y\sin\theta + z\cos\theta \quad \text{..... (7)}$$

$$x' = x$$

Las cuales pueden ser escritas en coordenadas homogéneas como:

$$\begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos\theta & -\sin\theta & 0 \\ 0 & \sin\theta & \cos\theta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} . \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} \quad \text{..... (8)}$$

$$P' = R_x(\theta) \cdot P \quad \text{.....(9)}$$

Una permutación cíclica de coordenadas en las Ecuaciones 3 proporciona las ecuaciones de trasformación para una rotación del eje-y:

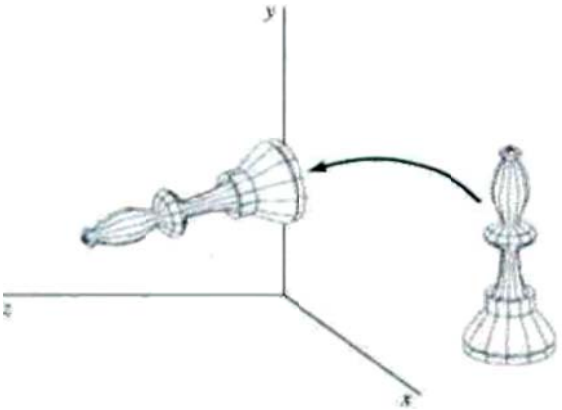

FIGURA 14. Rotación de un objeto alrededor del eje x.

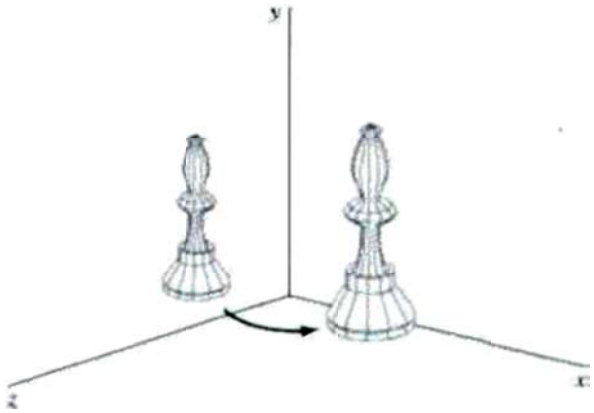

FIGURA 15. Rotación de un objeto alrededor del eje y.

$$z' = z\cos\theta - x\sin\theta$$

$$x' = z\sin\theta + x\cos\theta \quad \text{..... (10)}$$

$$y' = y$$

$$\begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix} = \begin{bmatrix} \cos\theta & 0 & \sin\theta & 0 \\ 0 & 1 & 0 & 0 \\ -\sin\theta & 0 & \cos\theta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} \quad \text{..... (11)}$$

$$P' = R_y(\theta) \cdot P \quad \text{.....(12)}$$

Una matriz de rotación tridimensional inversa se obtiene sustituyendo el ángulo θ por - θ. Los valores negativos para los ángulos de rotación generan rotaciones en el sentido de las agujas del reloj y la matriz identidad se obtiene multiplicando cualquier matriz de rotación por su inversa. Mientras sólo la función seno se vea afectada por el cambio de signo del ángulo de rotación, la matriz inversa puede obtenerse también intercambiando filas por columnas. Es decir, podemos calcular la inversa de cualquier matriz de rotación R formando su traspuesta $(\mathrm{R}^{-1} = \mathrm{R}^{\mathrm{T}})$.

Rotaciones tridimensionales generales.

Una matriz de rotación, para cualquier eje que no coincide con un eje de coordenadas, puede realizarse como una transformación compuesta incluyendo combinaciones de traslaciones y rotaciones de ejes de coordenadas. Primero, movemos el eje de rotación designado dentro de uno de los ejes de coordenadas, Después aplicamos la matriz de rotación apropiada para ese eje de coordenadas. El último paso en la secuencia de transformación es devolver el eje de rotación a su posición original.

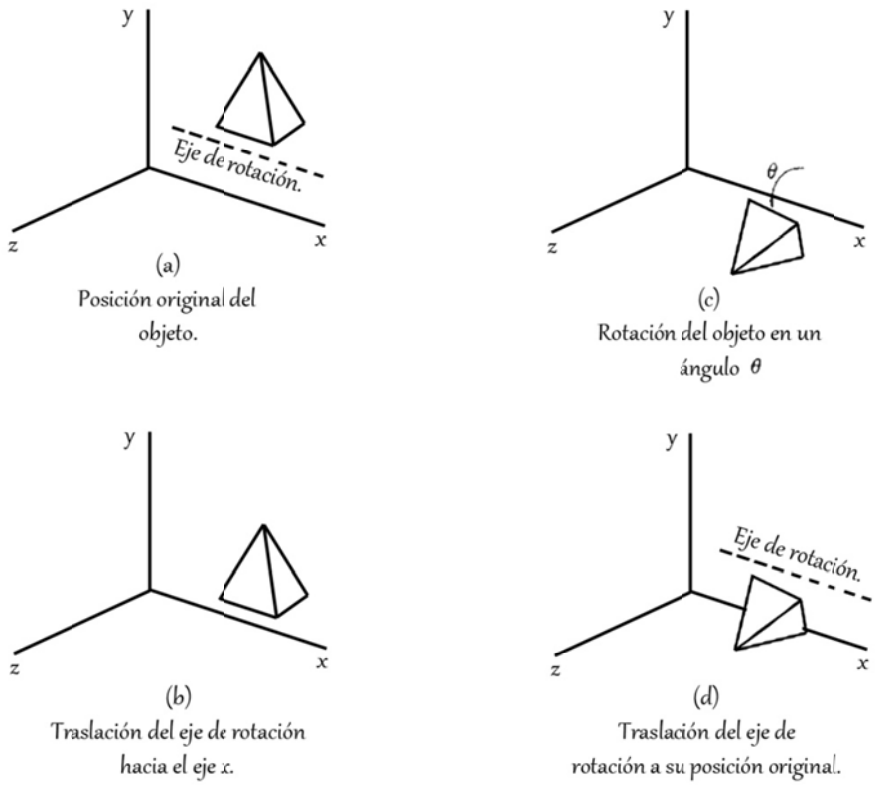

FIGURA 16. Secuencia de transformaciones para la rotación de un objeto sobre un eje que es paralelo al eje x.

En el caso especial de que un objeto vaya a ser rotado alrededor de un eje que es paralelo a uno de los ejes de coordenadas, conseguimos la rotación deseada con la siguiente secuencia de transformaciones.

(1) Traslado del objeto de tal forma que el eje de rotación coincida con el eje de coordenadas paralelo.

(2) Se realiza la rotación especificada sobre ese eje.

(3) Traslado del objeto de tal forma que el eje de rotación se mueva de nuevo a su posición original.

Los pasos de esta secuencia se ilustran en la Figura 16. Una posición de coordenadas P se transforma con la secuencia mostrada en esta figura como:

$$P' = T^{-1} \cdot R_x(\theta) \cdot T \cdot P \dots\dots\dots\dots\dots\dots\dots\dots\dots\dots (13)$$

donde la matriz de rotación compuesta para la transformación es:

$$R(\theta) = T^{-1} \cdot R_x(\theta) \cdot T \dots\dots\dots\dots\dots\dots\dots\dots\dots\dots (14)$$

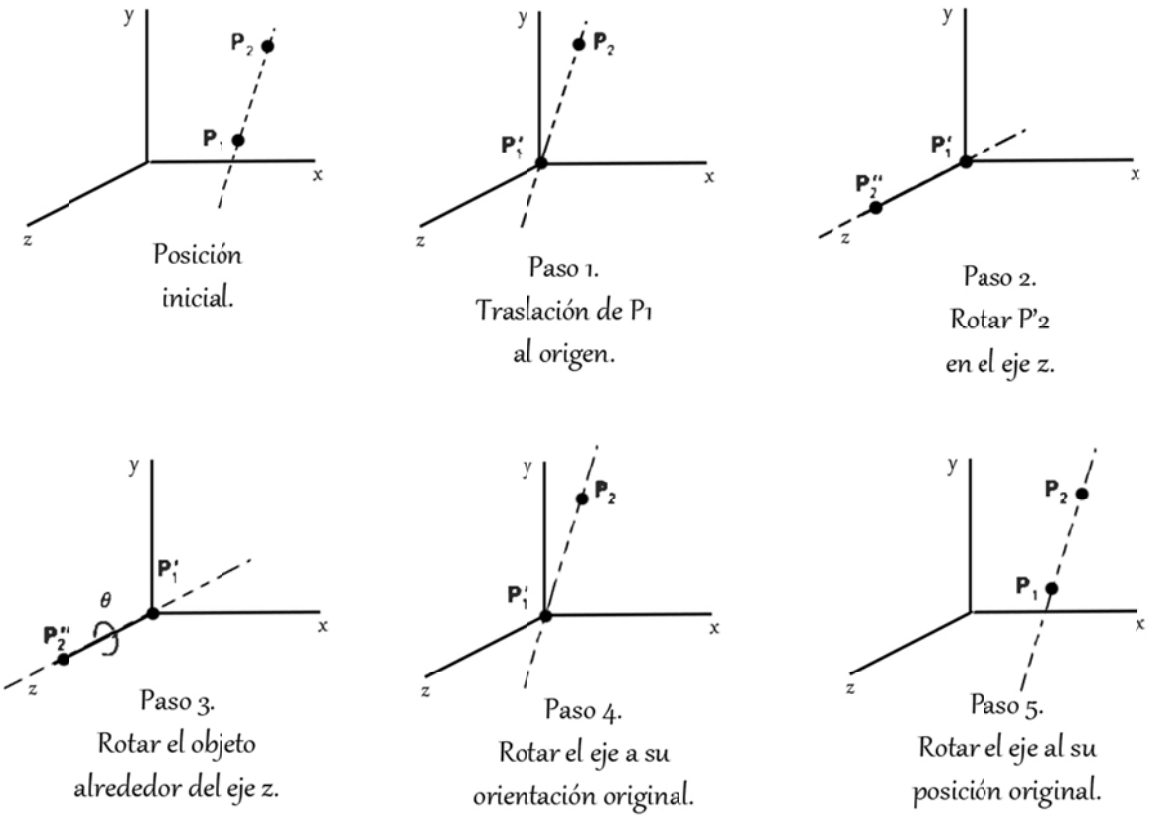

FIGURA 17. Cinco pasos de transformación para obtener una matriz compuesta para la rotación alrededor de un eje arbitrario, con el eje de rotación proyectado sobre el eje z.

Cuando un objeto va a ser rotado sobre un eje que no es paralelo a uno de los ejes de coordenadas, necesitamos desarrollar algunas transformaciones adicionales. En este caso, también necesitamos rotaciones para alinear el eje de rotación con un eje de coordenadas seleccionado y luego devolver el eje de rotación a su orientación original. Dando las especificaciones para la rotación de ejes y del ángulo de rotación, podemos llevar a cabo la rotación requerida en cinco pasos:

(1) Trasladar el objeto de tal forma que el eje de rotación pase a través del origen de coordenadas.

(2) Rotar el objeto de forma que el eje de rotación coincida con uno de los ejes de coordenadas.

(3) Realizar la rotación especificada sobre el eje de coordenadas seleccionado.

(4) Aplicar las rotaciones inversas para devolver al eje de rotación su orientación original.

(5) Aplicar la traslación inversa para devolver el eje de rotación a su posición espacial original.

Podemos transformar el eje de rotación dentro de cualquiera de los tres ejes de coordenadas. El eje-z es a menudo una elección conveniente. Después consideramos una secuencia de transformación usando la matriz de rotación del eje-z (Figura 17).

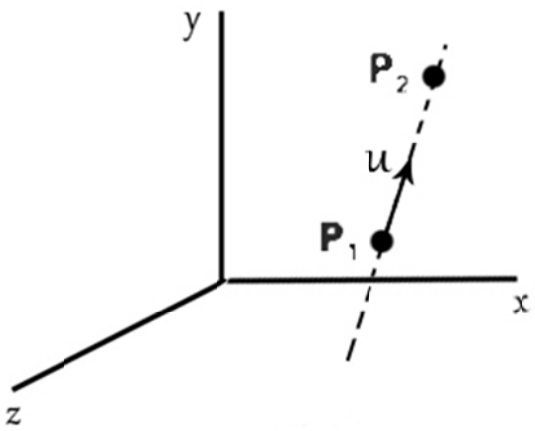

FIGURA 18. Un eje de rotación (línea de puntos) definidopor los puntos P_1 y P_2. La dirección para el vector unitario de eje, u, sedetermina especificando la dirección de rotación.

Un eje de rotación puede definirse con dos posiciones de coordenadas, como en la Figura 18, o con un punto coordenado y ángulos de dirección (o cosenos de dirección) entre el eje de rotación y dos de los ejes de coordenadas. Asumimos que el eje de rotación se define con dos puntos, como se ilustra, y que la dirección de rotación va a ser en el sentido contrario a las agujas del reloj cuando se mira a lo largo del eje de P_2 a P_1. Las componentes del vector del eje de rotación se calculan entonces del siguiente modo:

$$\boldsymbol{V = P_2 - P_1 = (x_2 - x_1, y_2 - y_1, z_2 - z_1)} \ldots\ldots\ldots (\mathbf{15})$$

Y el vector unitario del eje de rotación u es:

$$u = \frac{V}{|V|} = (a, b, c) \ldots\ldots\ldots\ldots (16)$$

donde las componentes a, b y c son los cosenos de dirección para la rotación del eje:

$$a = \frac{x_2 - x_1}{|V|}, \quad b = \frac{y_2 - y_1}{|V|}, \quad c = \frac{z_2 - z_1}{|V|} \ldots\ldots\ldots (17)$$

Si la rotación se va a realizar en sentido contrario (en el sentido de las agujas del reloj cuando se mira de P_2, a P_1) entonces deberíamos invertir el vector de eje V y el vector unitario u de tal manera que apuntasen en la dirección de P_2 a P_1.

El primer paso en la secuencia de rotación es establecer la matriz de traslación que recoloca el eje de rotación, de tal forma que pasa a través del origen de coordenadas. Mientras

queramos una rotación en sentido contrario a las agujas del reloj cuando miramos a lo largo del eje de P_2 a P_1 movemos el punto P_1 al origen. (Si la rotación ha sido especificada en la dirección opuesta, deberemos mover P_2, al origen).Esta matriz de traslación es:

$$T = \begin{bmatrix} 1 & 0 & 0 & -x_1 \\ 0 & 1 & 0 & -y_1 \\ 0 & 0 & 1 & -z_1 \\ 0 & 0 & 0 & 1 \end{bmatrix} \dots\dots\dots\dots (18)$$

A continuación hay que formular las transformaciones que colocarán el eje de rotación sobre el eje z.

Podemos usar las rotaciones del eje de coordenadas para llevar a cabo este alineamiento en dos pasos, y hay disponibles un cierto número de modos de desarrollar estos dos pasos. Para este ejemplo, primero giramos alrededor del eje x y después alrededor del eje y. La rotación del eje x obtiene el vector u dentro del plano xz, y la rotación del eje y cambia la dirección de u sobre el eje z. Estas dos rotaciones se ilustran en la Figura 19 para una posible orientación del vector u.

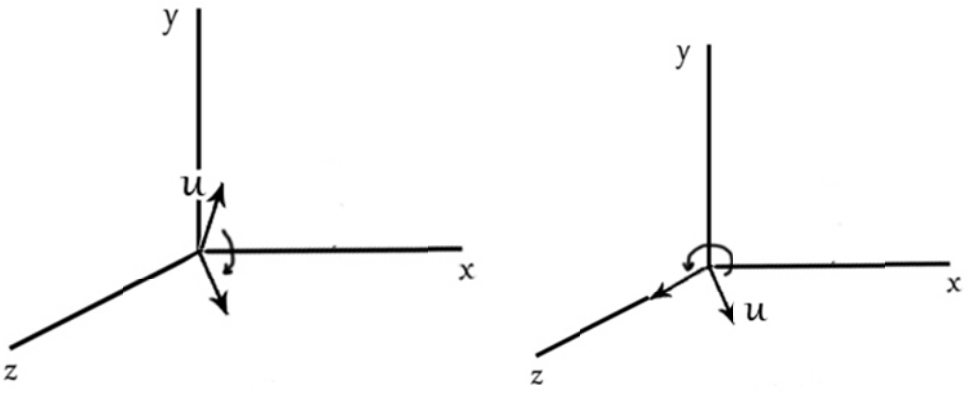

FIGURA 19. El vector unidad u se gira sobre el eje x paradejarlo en el plano xz (a), y después se gira alrededor del eje y para alinearlo con el eje z (b).

Establecemos la matriz de transformación para la rotación alrededor del eje x, determinando los valores para el seno y el coseno del ángulo de rotación necesario para obtener u dentro del plano yz. Este ángulo de rotación es el ángulo entre la proyección de u en el plano yz y el eje positivo de z (Figura 20).

Si representamos la proyección de u en el plano yz como el vector u' = (0, b, c) entonces el coseno del ángulo de rotación α puede determinarse a partir del producto escalar de u' y el vector unitario u, a lo largo del eje z:

$$\boldsymbol{\cos \alpha = \frac{u' \cdot u_z}{|u'||u_z|} = \frac{c}{d}} \dots\dots\dots (\mathbf{19})$$

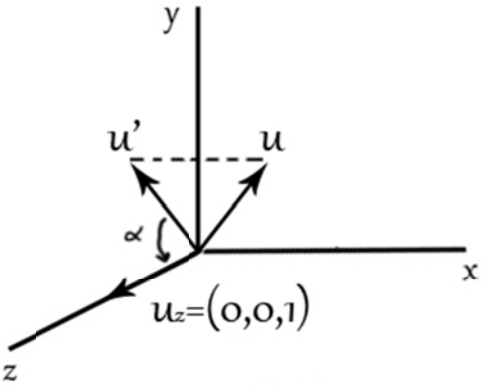

FIGURA 20. La rotación de u alrededor del eje x dentrodel plano xz se lleva a cabo rotando u' (que es la proyección de u en el plano yz) a través del ángulo α sobre el eje z.

Donde *d* es el módulo de u':

$$d = \sqrt{b^2 + c^2} \dots\dots\dots\dots (20)$$

De manera similar, podemos determinar el seno deα a partir del producto vectorial de u' y u. La forma independiente de las coordenadas de este producto vectorial es:

$$u' \times u_z = u_x |u'||u_z| \sin \propto \dots\dots\dots\dots (21)$$

y la forma cartesiana para el producto vectorial nos da:

$$u' \times u_z = u_x \cdot b \dots\dots\dots (22)$$

Igualando las Ecuaciones 21 y 22, y sabiendo que $|u_z| = 1$ y $|u'| = d$**,** tenemos:

$$d \sin \propto = b \quad \text{ó} \quad \sin \propto = \frac{b}{d} \dots\dots\dots\dots (23)$$

Ahora que hemos determinado los valores para $\cos \propto$y $\sin \propto$ en función de las componentes del vector **u**, podemos establecer los elementos de la matriz para la rotación de este vector sobre el eje ***x***y dentro del plano

$$R_x(\propto) = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & {}^{c}/_{d} & {}^{-b}/_{d} & 0 \\ 0 & {}^{b}/_{d} & {}^{c}/_{d} & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \dots\dots\dots\dots . (24)$$

El siguiente paso en la formulación de la secuencia de transformaciones es determinar la matriz que cambiará en sentido contrario a las agujas del reloj el vector unidad en el plano ***xz*** alrededor del eje y sobre el eje positivo ***z***. La Figura 21 muestra la orientación del vector unidad en el plano ***xz***, resultante de la rotación sobre el eje ***x***. Este vector, etiquetado como

u", tiene el valor ***a*** para su componente *x,* mientras que la rotación sobre el eje *x* deja la componente x invariable. Su componente *z* es ***d*** (el módulo de **u')** porque el vector **u'** ha sido rotado sobre el eje ***z***. Y la componente *y* de **u"** es 0, porque ahora se encuentra en el plano ***xz***. De nuevo podemos determinar el coseno del ángulo de rotación β a partir del producto escalar de los vectores unitarios **u"** y $\mathbf{u_z}$

$$\cos\beta = \frac{u'' \cdot u_z}{|u''||u_z|} = d \ldots\ldots\ldots\ldots (25)$$

Así, mientras $|u_z| = |u''| = 1$. Comparando la forma independiente de las coordenadas del producto vectorial:

$$u'' \times u_z = u_y|u''||u_z|\sin\beta \ldots\ldots\ldots\ldots (26)$$

con la forma cartesiana:

$$u'' \times u_z = u_y \cdot (-a) \ldots\ldots\ldots\ldots (27)$$

encontramos que,

$$\sin\beta = -a \ldots\ldots\ldots\ldots (28)$$

Por tanto, la matriz de transformación para la rotación de **u"** sobre el eje **y** es

$$R_y(\beta) = \begin{bmatrix} d & 0 & -a0 \\ 0 & 1 & 0\ 0 \\ a & 0 & d\ 0 \\ 0 & 0 & 0\ 1 \end{bmatrix} \ldots\ldots\ldots\ldots (29)$$

Con las transformaciones de matrices 18, 24 y 29, alineamos el eje de rotación con el eje positivo **z**. El ángulo de rotación especificado θpuede ahora aplicarse como una rotación alrededor del eje z:

$$R_z(\theta) = \begin{bmatrix} \cos\theta & -\sin\theta & 0 & 0 \\ \sin\theta & \cos\theta & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \ldots\ldots\ldots\ldots\ldots (30)$$

Para completar la rotación requerida sobre el eje dado, necesitamos transformar el eje de rotación de vuelta a su posición original. Esto se hace aplicando la inversa de las transformaciones 18, 24 y 29. La matriz de transformación para la rotación sobre un eje arbitrario puede entonces expresarse como la composición de estas siete transformaciones individuales:

$$R(\theta) = T^{-1} \cdot {R_x}^{-1}(\alpha) \cdot {R_y}^{-1}(\beta) \cdot R_z(\theta) \cdot R_y(\beta) \cdot R_x(\alpha) \cdot T \ldots\ldots\ldots\ldots\ldots (31)$$

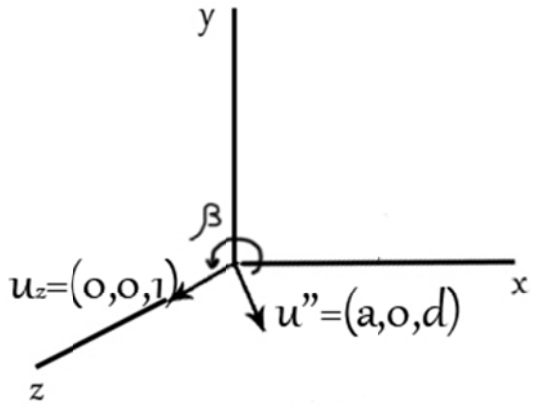

FIGURA 21. Rotación de un vector unidad u" (elvector u después de la rotación dentro del plano xz)sobre el eje y. Un ángulo de rotación positivo alineau" con el vector u.

Escala.

La expresión de la matriz para la transformación de cambio de escala tridimensional de una posición $P = (x, y, z)$relativa al origen de coordenadas puede representarse como:

$$\begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix} = \begin{bmatrix} s_x & 0 & 0 & 0 \\ 0 & s_y & 0 & 0 \\ 0 & 0 & s_z & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} \dots\dots\dots (32)$$

ó:

$$P' = S \cdot P \dots\dots\dots\dots\dots\dots (33)$$

donde a los parámetros de escala $\mathbf{s_x}$, $\mathbf{s_y}$ y $\mathbf{s_z}$ se les asignan cualesquiera valores positivos. Las expresiones explícitas para la transformación de cambio de escala respecto del origen son:

$$x' = x \cdot s_x\,, \quad y' = y \cdot s_y, \quad z' = z \cdot s_z \ \dots\dots\dots\dots\dots (34)$$

Cambiar la escala de un objeto con la transformación dada por las Ecuaciones 34 cambia la posición del objeto respecto del origen de coordenadas. Un valor del parámetro superior a 1 mueve un punto alejándolo del origen en la correspondiente dirección de coordenadas. De la misma manera, un valor de parámetro inferior a 1 mueve un punto acercándolo al origen en esa dirección de coordenadas. Además, si los parámetros de escala no son todos iguales, las dimensiones relativas del objeto transformado cambian. La forma original de un objeto se conserva realizando un "***cambio de escala uniforme". $s_x = s_y = s_z$.*** El resultado de aplicar un cambio de escala uniforme sobre un objeto con cada parámetro de escala igual a 2 se ilustra en la Figura 22.

Dado que algunos programas gráficos sólo ofrecen una rutina que realiza cambios de escala respecto al origen de coordenadas, podemos construir siempre una transformación de cambio de escala con respecto a cualquier ***posición fija*** seleccionada (x_f, y_f, z_f) usando la siguiente secuencia de transformaciones:

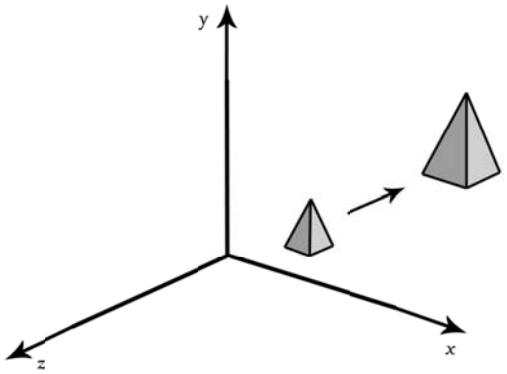

FIGURA 22. Duplicar el tamaño de un objeto con la transformación.

(1) Trasladar el punto fijo al origen.

(2) Aplicar la transformación de cambio de escala respeto al origen de coordenadas usando la Ecuación 34.

(3) Trasladar el punto fijo de vuelta a su posición original.

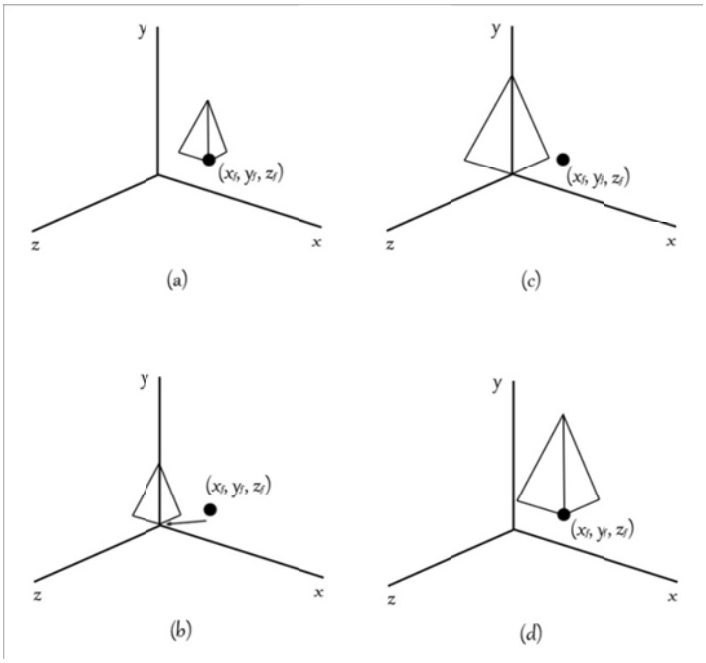

FIGURA 23.Secuencia de transformacionespara el cambio de escala de un objeto respecto a un punto fijo seleccionado usando la Ecuación 34.

Esta secuencia de transformaciones se muestra en la Figura 23. La representación de la matriz para un punto fijo de cambio de escala arbitrario puede expresarse como la concatenación de estas transformaciones de traslación - cambio de escala - traslación:

$$T(x_f, y_f, z_f) \cdot S(s_x, s_y, s_z) \cdot T(-x_f, -y_f, -z_f) = \begin{bmatrix} s_x & 0 & 0 & (1-s_x)x_f \\ 0 & s_y & 0 & (1-s_y)y_f \\ 0 & 0 & s_z & (1-s_z)z_f \\ 0 & 0 & 0 & 1 \end{bmatrix} \dots\dots\dots (35)$$

Podemos establecer procedimientos programados para la construcción de matrices de escalado tridimensional, usando tanto la secuencia traslación-escalado-traslación como la incorporación directa de las coordenadas del punto fijo.

4.5.1.3 Modelado con primitivas.

Para este método de modelado se crea una primitiva gráfica como un cubo, un cilindro, esfera, etc. y se extruden las caras que se necesiten hasta formar un modelo básico de la figura que se quiere realizar. Posteriormente se modifican los vértices de la figura hasta ajustarlos al modelo que quiere formarse.

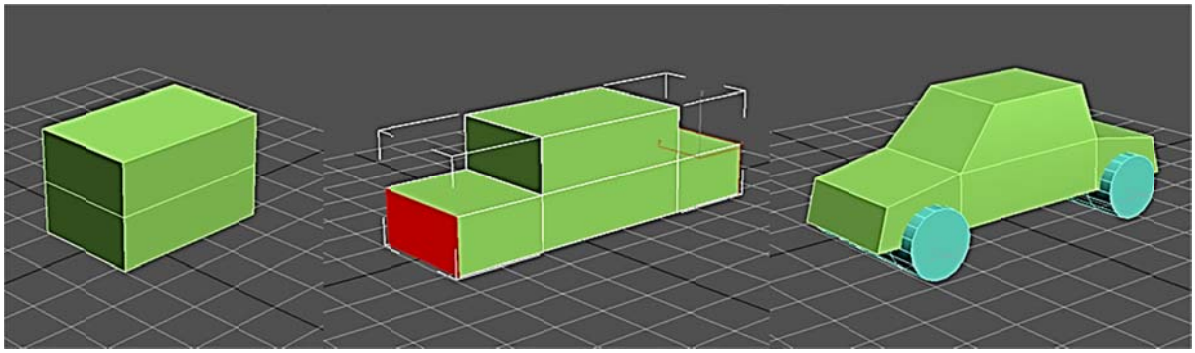

FIGURA 24. Ejemplo de modelado con primitivas.

4.5.1.4 Representaciones con splines.

En el dibujo de bocetos, un ***spline*** es una banda flexible que se utiliza para producir una curva suave que pasa por puntos concretos. Se distribuyen varios pesos pequeños a lo largo de la banda para mantenerla en suposición sobre la mesa de dibujo mientras se traza la curva. El término ***curva con spline*** en principio hacía referencia a una curva dibujada de este modo. Podemos describir matemáticamente tal curva con una función creada por tramos de polinomios cúbicos, cuya primera y segunda derivadas son continuas en las diferentes partes de la curva. En los gráficos por computadora, el término curva con ***spline*** ahora se refiere a cualquier curva compuesta por partes polinómicas que satisfacen condiciones de continuidad específicas en los límites de las mismas. Una superficie con ***splines*** se puede describir con dos conjuntos de curvas ortogonales con ***splines.*** Existen varias clases

diferentes de especificaciones de ***splines*** que se utilizan en aplicaciones de gráficos por computadora.

Cada especificación individual simplemente hace referencia a un tipo particular de polinomio con ciertas condiciones específicas en los límites.

Los ***splines*** se utilizan para diseñar formas de curvas y de superficies, para digitalizar dibujos y para especificar trayectorias de animación de objetos o la posición de la cámara en una escena. Entre las aplicaciones habituales de CAD con ***splines*** se incluyen el diseño de la carrocería de automóviles, las superficies de aviones o naves espaciales, los cascos de las embarcaciones y los electrodomésticos.

FIGURA 25. (Izquierda). Conjunto de seis puntos de control interpolados con secciones polinómicas continuas por tramos,(derecha). conjunto de seis puntos de control aproximados con secciones polinómicas continuas por tramos.

Curvas con splines de Bézier.

Este método de aproximación con ***spline*** fue desarrollado por el ingeniero francés Pierre Bézier para su uso en el diseño de las carrocerías de automóviles Renault. Los ***splines* de Bézier** disponen de unas propiedades que los hacen especialmente útiles y convenientes para el diseño de curvas y superficies, además, son fáciles de implementar.

Por lo general, una parte de una curva de Bézier se puede ajustar a cualquier número de puntos de control. El grado del polinomio de Bézier se determina con el número de puntos de control que hay que aproximar y con su posición relativa.

B-splines.

Esta clase de ***splines*** es la más profusamente utilizada y las funciones de ***splines*B** están disponibles habitualmente en los sistemas CAD y en muchos paquetes de programación de gráficos. Al igual que los ***splines*** de Bézier, los ***splines*B** se generan aproximando un

conjunto de puntos de control. Pero los ***splines*B** presentan dos ventajas frente a los ***splines*** de Bézier:

(1) el grado de un polinomio de un ***spline*B** se puede establecer de forma independiente al número de puntos de control, y

(2) los ***splines*B** permiten control local sobre la forma de un ***spline.***

La desventaja es que los ***splines*B** son más complejos que los ***splines*** de Bézier.

4.5.1.5 Nurbs.

Acrónimo inglés de la expresión *Non Uniform Rational* B-splines. Las **NURBS**, B-splines racionales no uniformes, son representaciones matemáticas de geometría en 3D capaces de describir cualquier forma con precisión, desde simples líneas en 2D, círculos, arcos o curvas, hasta los más complejos sólidos o superficies orgánicas[10] de forma libre en 3D. Gracias a su flexibilidad y precisión, se pueden utilizar modelos NURBS en cualquier proceso, desde la ilustración y animación hasta la fabricación de productos.

4.5.1.6 Sistema de partículas.

Un sistema de partículas es un grupo especializado de objetos que son modificados como una sola entidad. Al agrupar todas las partículas en un solo sistema, se pueden hacer modificaciones a todos los objetos con parámetros sencillos, además que no hace tan lento el sistema como ocurriría al hacerlos por separado.

4.5.1.7 Representación basada en Imagen –Image Based Rendering (IBR).

El modelado basado en imágenes se refiere al proceso de utilización de imágenes para la reconstrucción de modelos geométricos 3D[11]. Su objetivo es:

- Gran realismo.
- Reducción del tiempo de procesamiento.
- Simplificación de la tarea del modelado mediante el uso de imágenes como primitivas de modelado y renderizado.

Se presentan los siguientes tipos:

- IBR: Uso de imágenes en lugar de polígonos para el renderizado de primitivas.
- IBM: Uso de imágenes para guiar la reconstrucción de modelos geométricos 3D.

[10]Superficie orgánica se refiere a la superficie que no está compuesta por polígonos regulares y toma su nombre de las formas de la naturaleza que son más complejas.

[11]Modelado basado en imágenes. Mario Rodríguez Martín. http://www.slideshare.net/MarioRM/modelado-basado-en-imgenes

Clasificación:

IBR (*Image based rendering* – Representación basada en imágenes).

- IBR puro. Se capturan muestras del entorno mediante fotos o secuencias de video. No se requiere 3D y la velocidad de renderizado no depende de la complejidad de la escena.
  - ❖ Panoramas cilíndricos (*Cylindrical panoramas*). Proporciona una orientación horizontal desde un punto. Se realiza con cámaras panorámicas especiales. El espectador puede rotar, pero no moverse, por lo tanto, no sirve para crear entornos virtuales.

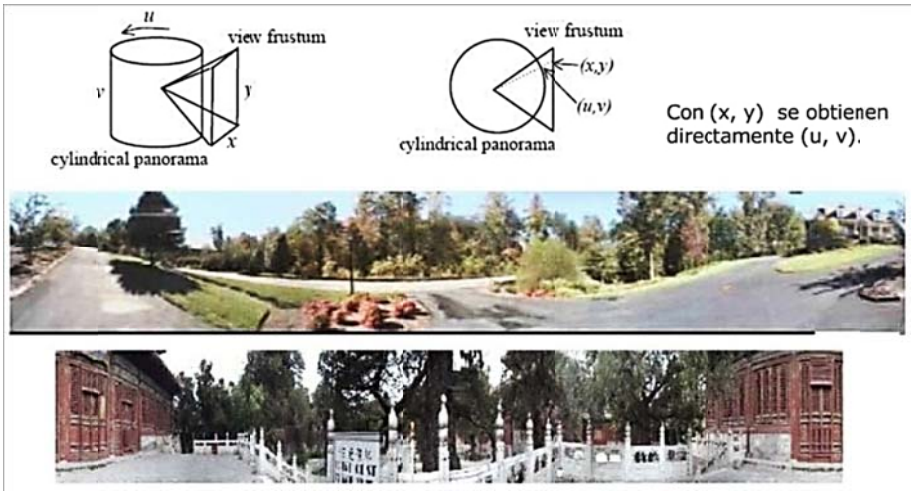

FIGURA 26. Ejemplo de un panorama cilíndrico.

- IBR híbrido.

  - ❖ Objetos basados en imágenes (Image-based objects). Se construye adquiriendo múltiples vistas del objeto que se registran y muestrean desde un centro de proyección hasta formar un paralepípedo.

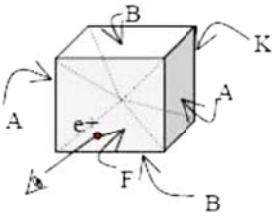

FIGURA 27. Ejemplo de modelado de objetos basados en imágenes.

El uso de paralepípedos permite que la representación se descomponga en regiones parametrizadas por planos, lo que facilita su implementación.

Suponiendo una esfera y un rayo que incide en ella y pasa por su centro, interseca en dos puntos siendo el más cercano el epipolo positivo y el más lejano el negativo. F indica el epipolo positivo y k el negativo.

IBM (*Image Based Modeling*). Dependiendo de la cantidad de información geométrica que se almacena para la reconstrucción se diferencia IBM puro e IBM híbrido. Al igual que el IBR, a partir de unas imágenes de entrada (fotografías) se obtiene una reconstrucción del modelo.

- IBM puro.
  - ❖ Método proyectivo (Proyective method). Estas técnicas se aprovechan de las propiedades proyectivas de la escena para reconstruir modelos geométricos a través de un conjunto de fotografías. Se suele utilizar para superficies planas por la dificultad para reconstruir formas arbitrarias.

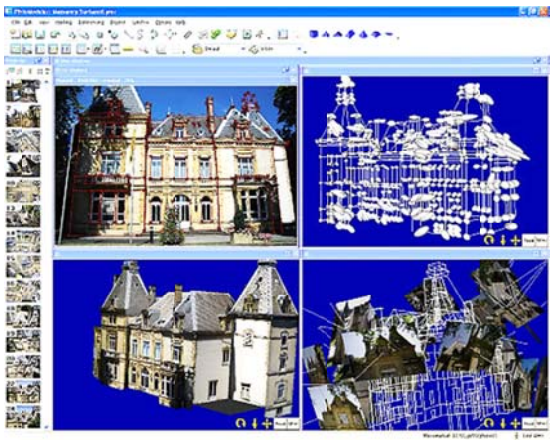

FIGURA 28. Ejemplo de método proyectivo.

- IBM híbrido.
  - ❖ Tour into the picture. Crea una animación a partir de una imagen 2D. Permite navegar por la escena. Para construir la escena se realizan los pasos siguientes:
    - ➢ Se separan los objetos que están en primer plano de los del fondo.
    - ➢ Se determinan los puntos que desaparecen y se fija la perspectiva.
    - ➢ Se modela el fondo a partir de 5 polígonos. El primero es un rectángulo y los siguientes están determinados por líneas radiales que parten desde los extremos del rectángulo. Las 5 caras forman un paralelepípedo que puede ser renderizado por mapeo de texturas.
    - ➢ Se introducen los objetos que estaban en primer plano, computando su posición respecto al fondo.

4.5.2 Iluminación.

Se utiliza un **modelo de iluminación,** también denominado **modelo de sombreado,** para calcular el color de cada posición iluminada en la superficie de un objeto. Un **método de representación superficial** utiliza los cálculos de color del modelo de iluminación para determinar los colores de los píxeles para todas las posiciones proyectadas de una escena. El modelo de iluminación puede aplicarse a cada posición de proyección, o bien puede llevarse a cabo la representación de la superficie interpolando los colores de las superficies a partir de un pequeño conjunto de cálculos relativos al modelo de iluminación.

FIGURA 29. Ejemplo de un modelo hecho con el método Tour into the picture.

Los modelos físicos de iluminación tienen en cuenta una serie de factores, como las propiedades de los materiales, las posiciones de los objetos en relación con las fuentes de iluminación y con otros objetos y las características de las fuentes luminosas. Los objetos pueden estar compuestos de materiales opacos, o bien pueden ser más o menos transparentes. Además, pueden tener superficies brillantes o mates, y exhibir diversos patrones de textura superficial. Pueden utilizarse fuentes luminosas, de formas, colores y posiciones variables para iluminar una escena.

Dados los parámetros de las propiedades ópticas de las superficies, dadas las posiciones relativas de las superficies dentro de una escena, dados el color y las posiciones de las fuentes luminosas, dadas las características de dichas fuentes y dadas la posición y la orientación del plano de visualización, se utilizan los modelos de iluminación para calcular la intensidad de la luz proyectada desde una posición concreta de la superficie en una dirección de visualización especificada.

Los modelos de iluminación en infografía[12] son a menudo aproximaciones de las leyes físicas que describen los efectos de iluminación de una superficie. Para reducir los cálculos, la mayoría de los programas utilizan modelos empíricos basados en cálculos de fotometría

[12]Infografía es la técnica de elaboración de imágenes por computadora.

simplificados. Otros modelos más precisos, como el algoritmo de radiosidad, calculan las intensidades luminosas considerando la propagación de la energía radiante entre las fuentes luminosas y las diversas superficies de una escena.

4.5.2.1 Fuentes luminosas.

Cualquier objeto que emita energía radiante es una **fuente luminosa** que contribuye a los efectos de iluminación que afectan a otros objetos de la escena. Podemos modelar fuentes luminosas con diversas formas y características. Por ejemplo, en algunas aplicaciones, puede que nos interese crear un objeto que sea a la vez una fuente luminosa y un reflector de luz. Por ejemplo, un farol de plástico que rodee a una bombilla emite luz, pero también los rayos luminosos procedentes de otras fuentes se reflejan en la superficie del farol.

También podríamos modelar el farol como una superficie semitransparente dispuesta en torno a una fuente luminosa, pero para algunos objetos, como por ejemplo un panel fluorescente de gran tamaño, puede que sea más conveniente describir la superficie simplemente como una combinación de un emisor y un reflector.

Las fuentes luminosas pueden definirse con diversas propiedades. Podemos definir su posición, el color de la luz emitida, la dirección de emisión y la forma de la fuente. Si la fuente es también una superficie reflectora de la luz, necesitaremos indicar sus propiedades de reflectividad. Además, podemos definir una fuente luminosa que emita diferentes colores en diferentes direcciones.

En la mayoría de las aplicaciones, y particularmente en los gráficos en tiempo real, se utiliza un modelo simple de fuentes luminosas para evitar complicar demasiado los cálculos.

4.5.2.2 Fuentes luminosas puntuales.

El modelo más simple para un objeto que emite energía radiante es la **fuente luminosa puntual** de un único color, el cual se especifica mediante las tres componentes RGB. Podemos definir una fuente puntual para una escena indicando su posición y el color de la luz emitida. Como se muestra en la figura 28, los rayos luminosos se generan según una serie de trayectorias radialmente divergentes a partir de esa única fuente puntual monocromática. Este modelo de fuente luminosa constituye una aproximación razonable para aquellas fuentes cuyas dimensiones sean pequeñas comparadas con el tamaño de los objetos de la escena. También podemos simular fuentes de mayor tamaño mediante emisores puntuales si dichas fuentes no están demasiado próximas a la escena. Utilizamos la posición de una fuente puntual dentro de un modelo de imagen para determinar qué objetos de la escena se ven iluminados por dicha fuente y para calcular la dirección de los rayos luminosos cuando éstos inciden sobre una posición seleccionada de la superficie del objeto.

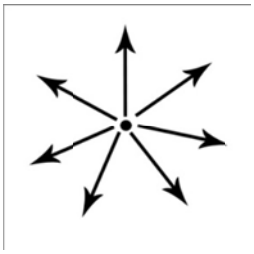

FIGURA 30. Trayectorias divergentes de los rayos a partir de una fuente luminosa puntual.

4.5.2.3 Fuentes luminosas infinitamente distantes.

Una fuente luminosa de gran tamaño, como por ejemplo el Sol, pero que esté muy lejos de una escena puede también aproximarse como un emisor puntual, aunque en este caso la variación que existe en sus efectos direccionales es muy pequeña. El trayecto del rayo luminoso que va desde una fuente distante hasta cualquier posición de la escena es prácticamente constante, como se ilustra en la Figura 31.

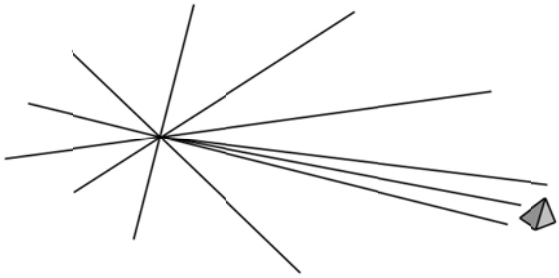

FIGURA 31. Los rayos luminosos procedentes de una fuente infinitamente distante iluminan a los objetos según una serie de trayectos prácticamente paralelos.

4.5.2.4 Modelos básicos de iluminación.

Los modelos más precisos de iluminación superficial calculan los resultados de las interacciones entre la energía radiante incidente y la composición material de un objeto. Para simplificar los cálculos de iluminación superficial, podemos utilizar representaciones aproximadas de los procesos físicos reales. El modelo empírico produce resultados razonablemente buenos y es el que se implementa en la mayoría de los sistemas gráficos.

Los objetos emisores de luz en un modelo básico de iluminación suelen estar limitados, generalmente, a fuentes puntuales. Sin embargo, muchos programas gráficos proporcionan funciones adicionales para incluir fuentes direccionales (como reflectores) y fuentes luminosas complejas.

4.5.2.5 Luz ambiente.

En nuestro modelo básico de iluminación, podemos incorporar la luz de fondo definiendo un nivel de brillo general para la escena. Esto produce una iluminación ambiente uniforme que

es igual para todos los objetos y que aproxima las reflexiones difusas globales correspondientes a las diversas superficies iluminadas.

Las reflexiones producidas por la iluminación mediante la luz ambiente son sólo una forma de reflexión difusa, y son independientes de la dirección de visualización y de la orientación espacial de las superficies. Sin embargo, la cantidad de luz ambiente incidente que se refleje dependerá de las propiedades ópticas de las superficies, que determinan qué parte de la energía incidente se refleja y qué parte se absorbe.

4.5.2.6 Reflexión difusa.

Podemos modelar la reflexión difusa de una superficie asumiendo que la luz ambiente es reflejada con igual intensidad en todas las direcciones, independientemente de la posición de visualización. Tales superficies se denominan **reflectores difusos ideales.** También se les denomina **reflectores lambertianos,** porque la energía luminosa radiante reflejada por cualquier punto de la superficie se calcula mediante la **ley del coseno de Lambert**[13]. (Figura 32).

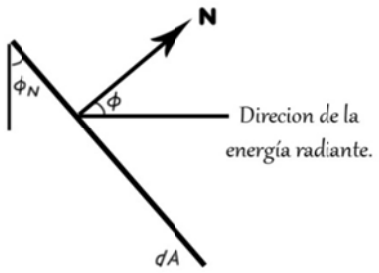

FIGURA 32. La energía radiante de un elemento de área superficial dA en la dirección ϕ_N relativa a la normal a la superficie es proporcional al cosϕ.

Para la reflexión lambertiana, la intensidad de la luz es la misma en todas las direcciones de visualización.

Suponiendo que haya que tratar a todas las superficies como un reflector difuso ideal (lambertiano), podemos definir un parámetro para cada superficie que determine la fracción de la luz incidente que hay que dispersar en forma de reflexiones difusas. Este parámetro se denomina **coeficiente de reflexión difusa** o **reflectividad difusa (k_d).** La reflexión difusa en todas las direcciones se da entonces una constante cuyo valor es igual a la intensidad de la luz incidente multiplicada por el coeficiente de reflexión difusa. Para una fuente luminosa monocromática, al parámetro k_d se le asigna un valor constante en el intervalo 0.0 a 1.0, de acuerdo con las propiedades de reflexión que queramos que la superficie tenga. Para una

[13]Esta ley establece que la cantidad de energía radiante procedente de cualquier pequeña área "de superficie" dA en una dirección ϕ_N relativa a la normal a la "superficie es proporcional a $\cos(\phi)$".

superficie altamente reflectante, asignaremos a k_d un valor próximo a 1.0. Si queremos simular una superficie que absorba la mayor parte de la luz incidente, asignaremos a la reflectividad un valor próximo a 0.0.

4.5.2.7 Superficies transparentes.

Podemos describir un objeto, como el cristal de una ventana, como ***transparente*** si podemos ver las cosas que están situadas detrás del objeto. De forma similar, si no podemos ver las cosas que están detrás del objeto, diremos que el objeto es ***opaco.*** Además, algunos objetos transparentes, como el cristal esmerilado y ciertos materiales plásticos, son translúcidos, de modo que la luz transmitida se difunde en todas direcciones. Los objetos visualizados a través de materiales translúcidos parecen borrosos y a menudo no se les puede identificar claramente.

Una superficie transparente, en general, produce luz tanto reflejada como transmitida. La luz transmitida a través de la superficie es el resultado de emisiones y reflexiones de los objetos y de las fuentes situadas detrás del objeto transparente.

4.5.2.8 Materiales translúcidos.

En la superficie de un objeto transparente puede producirse tanto transmisión difusa como especular[14]. Los efectos difusos tienen gran importancia cuando haya que modelar materiales translúcidos. La luz que pasa a través de un material translúcido se dispersa, de modo que los objetos situados en segundo plano se ven como imágenes borrosas. Podemos simular la transmisión difusa distribuyendo las contribuciones de intensidad delos objetos de segundo plano a lo largo de un área finita, o bien utilizar métodos de trazado de rayos para simularlos objetos translúcidos. Estas manipulaciones requieren mucho tiempo de procesamiento, por lo que los modelos básicos de iluminación sólo suelen calcular los efectos de transparencia especular.

4.5.2.9 Refracción de la luz.

Cuando un haz luminoso incide sobre una superficie transparente, parte del mismo se refleja y parte se transmite a través del material, en forma de luz refractada, como se muestra en la Figura 33.

La Figura 34 ilustra las modificaciones del trayecto debidas a la refracción para un rayo de luz que atraviesa una fina lámina de cristal. El efecto global de la refracción consiste en desplazar la luz incidente hasta una trayectoria paralela cuando el rayo emerge del material.

Incluirlos objetos de refracción en una escena puede producir imágenes muy realistas, pero la determinación de los trayectos de refracción y las intersecciones con los objetos requiere

[14]Especular: relativo o perteneciente a un espejo. En este caso se refiere a la luz reflejada por el material.

una cantidad de proceso considerable. La mayoría de los métodos del espacio de imagen basados en líneas de barrido modelan la transmisión de la luz mediante aproximaciones que reducen el tiempo de procesamiento.

4.5.2.10 Sombras.

Pueden utilizarse métodos de detección de la visibilidad para localizar regiones que no estén iluminadas por las fuentes de luz. Con la posición de visualización situada en la ubicación de una fuente luminosa, podemos determinar cuáles secciones de las superficies de una escena no son visibles. Estas serán las áreas de sombra.

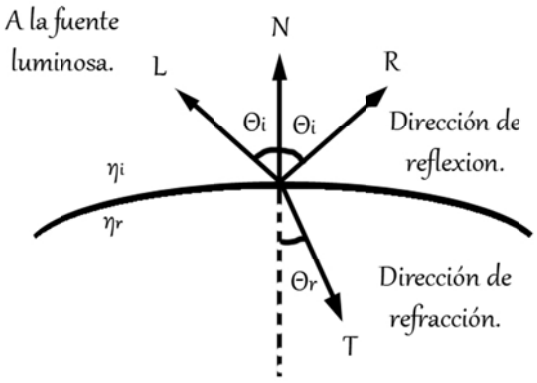

FIGURA 33. Dirección de reflexión R y dirección de refracción (transmisión) T para un rayo de luz que incide sobre una superflcie con un índice de refracción n .

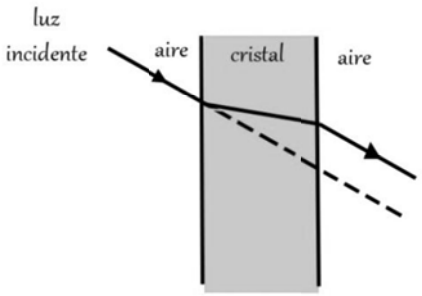

FIGURA 34. Refracción de la luz a través de un panel de cristal. El rayo refractado emergente describe una trayectoria que es paralela a la del rayo luminoso incidente(línea punteada).

La Figura 35 ilustra una serie de regiones de sombra sobre la cara de un carácter animado. En esta imagen, las regiones de sombra son secciones de la superficie que no son visibles desde la posición de la fuente luminosa que está situada encima dela figura. Así, la mano y el brazo levantados son iluminados, pero las secciones de la cara situadas detrás del brazo, según la línea de visión que proviene de la fuente luminosa, estarán en sombras.

Los patrones de sombra generados mediante un método de detección de superficies visibles son válidos para cualquier posición de visualización seleccionada, mientras no se varíen las posiciones de las fuentes luminosas. Las superficies que sean visibles desde la posición de visualización se sombrean de acuerdo con el modelo de iluminación, que puede combinarse con patrones de texturas.

Podemos mostrar las áreas de sombras únicamente con la intensidad de la luz ambiente, o podemos combinar la luz ambiente con una serie de texturas de superficie especificadas.

FIGURA 35. Patrones de sombra mapeados sobre la cara de Aki Ross, un carácter animado de la película Final Fantasy: The Spirits Within.

4.5.2.11 Métodos de trazado de rayos.

La ***proyección de rayos,*** es una técnica que se utiliza en geometría constructiva de sólidos para localizar las intersecciones con las superficies a lo largo de la trayectoria de un rayo trazado desde una posición de píxel.

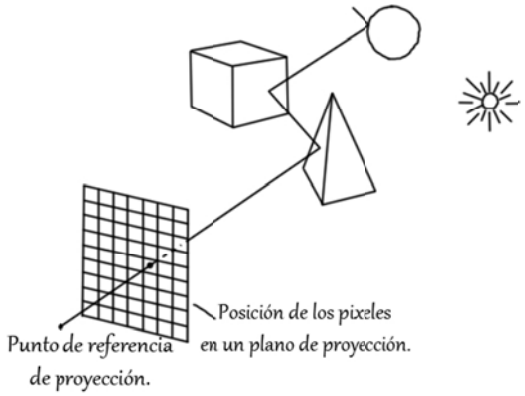

FIGURA 36. Trayectos múltiples de reflexión y transmisión para un rayo trazado desde elpunto de referencia de proyección a través de una posición de píxel y a través de una escena que contiene diversos objetos.

El trazado de rayos es la generalización del procedimiento básico de proyección de rayos. En lugar de limitarnos a localizarla superficie visible desde cada posición de píxel, lo que hacemos es continuar rebotando el rayo correspondiente al píxel a través de la escena, como se ilustra en la Figura 36, con el fin de recopilar las diversas contribuciones de intensidad. Esto proporciona una técnica simple y potente de representación para obtener efectos globales de reflexión y transmisión. Además, el algoritmo básico de trazado de rayos detecta las superficies visibles, identifica las áreas en sombra, permite representar efectos de transparencia, genera vistas de proyección en perspectiva y admite efectos de iluminación con múltiples fuentes luminosas. Las imágenes de escenas generadas mediante trazado de rayos pueden ser enormemente realistas, particularmente cuando la escena contiene objetos brillantes, pero los algoritmos de trazado de rayos requieren un tiempo de cálculo considerable. En la Figura 37 se muestra un ejemplo de los efectos globales de reflexión y transmisión que pueden conseguirse mediante las técnicas de trazado de rayos.

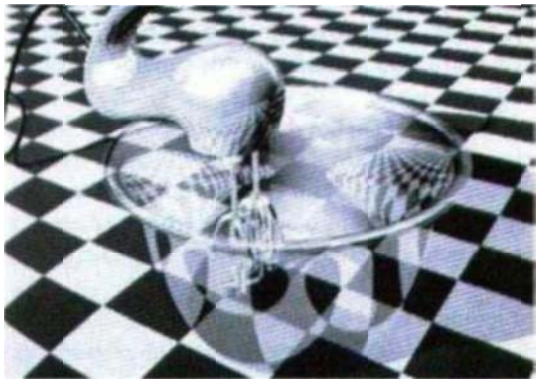

FIGURA 37. Una escena generada mediante trazado de rayos,donde se muestran los efectos globales de reflexión y transparencia.

4.5.2.12 Modelo de iluminación de radiosidad.

Aunque el modelo básico de iluminación produce resultados razonables para muchas aplicaciones, hay diversos efectos de iluminación que no quedan descritos de forma precisa mediante las aproximaciones simples de este modelo. Podemos modelar más adecuadamente los efectos de iluminación si tenemos en cuenta las leyes físicas que gobiernan las transferencias de energía radiante dentro de una escena iluminada, éste método para el cálculo de los valores de color de los pixeles se denomina generalmente modelo de radiosidad.

4.5.3 Texturizado (Texturing).

Las superficies poligonales (secuencia de caras) pueden contener datos correspondientes de más de un color, pero en el software más avanzado, pueden ser una superficie virtual para una imagen. Tal imagen es colocada en una cara, o la serie de caras y es llamada textura.

Las texturas añaden un nuevo grado de detalle en cuanto a cómo las caras y los polígonos que contendrán por último la forma en que serán sombreados y cómo la imagen es interpretada durante el sombreado.

Un método común para añadir detalles a un objeto consiste en mapear patrones sobre la descripción geométrica del objeto. El patrón de texturas puede definirse mediante una matriz de valores de color o mediante un procedimiento que modifique los colores del objeto. Este método para incorporar detalles de los objetos a una escena se denomina **mapeado de texturas** o **mapeo de patrones** y las texturas pueden definirse como patrones unidimensionales, bidimensionales o tridimensionales. Cualquier especificación de textura se denomina **espacio de textura,** que se referencia mediante **coordenadas de textura**.

4.5.3.1 Patrones de reducción de texturas (mipmaps).

En animación y otras aplicaciones, el tamaño de los objetos suelen cambiar a menudo. Para objetos mostrados con patrones de textura, necesitamos entonces aplicar los procedimientos de mapeado de texturas a las dimensiones modificadas del objeto. Cuando el tamaño de un objeto texturizado se reduce, el patrón de textura se aplica a una región más pequeña y esto puede hacer que aparezcan dispersiones en las texturas. Para evitar esto, podemos crear un conjunto de patrones de reducción de texturas que se deberán utilizar cuando el tamaño visualizado de los objetos se reduzca.

Normalmente, cada patrón de reducción tiene la mitad del tamaño del patrón anterior. Por ejemplo, si tenemos un patrón bidimensional 16 por 16, podemos definir cuatro patrones adicionales con los tamaños reducidos 8 por 8, 4 por 4, 2 por 2 y 1 por 1. Para cualquier vista de un objeto, podemos entonces aplicar el patrón de reducción apropiado con el fin de minimizar las distorsiones. Estos patrones de reducción se suelen denominar mapas MIP o mipmaps, donde el término mip es un acrónimo de la frase latina *multum in parva,* que podría traducirse como «mucho en un pequeño objeto».

4.5.3.2 Métodos de texturizado procedural.

Otro método para añadir un patrón de texturas a un objeto consiste en utilizar una definición procedimental para las variaciones de color que hay que aplicar. Esta técnica evita los cálculos de transformación implicados en el mapeado de patrones matriciales sobre las descripciones de los objetos. Además, el texturizado procedimental elimina los requisitos de almacenamiento necesarios cuando hay que aplicar muchos patrones de textura de gran tamaño a una escena.

Generamos una textura procedimental calculando variaciones para las propiedades o características de un objeto. Por ejemplo, las vetas de madera o del mármol pueden crearse para un objeto utilizando funciones armónicas (curvas sinusoidales) definidos en una región

del espacio tridimensional. Entonces, se superponen perturbaciones aleatorias a las variaciones armónicas con el fin de descomponer los patrones simétricos.

4.5.3.3 Mapeado de relieve. Bump mapping.

Aunque las matrices de texturas pueden utilizarse para añadir detalles de carácter fino a una superficie, usualmente no son efectivas para modelar la apariencia rugosa de las superficies de algunos objetos tales como naranjas, fresas o pasas. Un método mejor para modelar la rugosidad de las superficies consiste en aplicar una función de perturbación a la normal a la superficie y luego usar el vector normal perturbado en los cálculos realizados dentro del modelo de iluminación. Esta técnica se denomina mapeado de relieve ***(bump mapping).***

FIGURA 38. Representación del aspecto característico de las superficies rugosas mediante mapeado de relieve.

4.5.3.4 Mapas de luz. Lightmaps.

Esta técnica para usar texturas tiene ventajas para juegos en tiempo real como su espectacularidad, bajo costo computacional y relativa sencillez de implementación. Tiene algunas desventajas, como su bajo nivel de detalle de las sombras, o que no son recomendables en escenarios exteriores.

Los *lightmaps* (mapas de luz) consisten en añadir una segunda textura a todas y cada una de las caras existentes en una escena 3D. Esta textura contendría la luz que recibe cada cara. Los lightmaps son precalculados, ya que si por un lado una vez creados son muy rápidos de mostrar, el costo de su creación es elevado.

Por ejemplo, si tenemos modelada una habitación, con sus respectivas 4 paredes, suelo y techo, aunque todas las caras tengan la misma textura, necesitaremos 6 texturas diferentes para la iluminación. Esto supondrá una gran cantidad de texturas al realizarlo en una escena

compleja, y por tanto, un gran consumo de memoria RAM. La solución es crear Lightmaps de baja resolución.

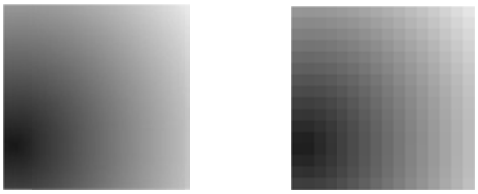

FIGURA 39. Lightmap Ideal y lightmap de baja resolución.

Un ejemplo de un lightmap es el siguiente:

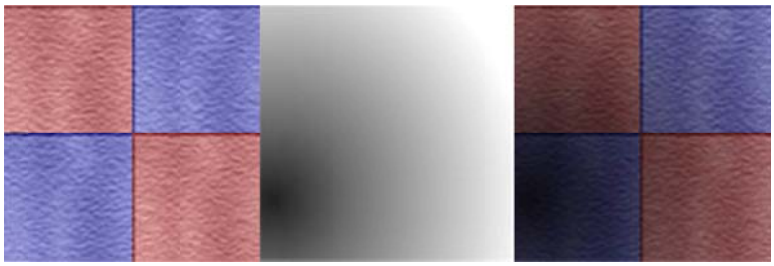

FIGURA 40. Ejemplo de un lightmap; textura inicial, lightmap y textura resultante.

4.5.3.5 Texturas horneadas (texture baking).

El método de hacer render a textura, "*render to texture*", o "*texture baking*", permite crear mapas de textura basados en la apariencia del objeto al ser "*rendereado"*. Las texturas son "horneadas" (*baked*) junto con el modelo, y de esta manera, todos los reflejos, sombras, luces, etc., quedan fusionados en un solo mapa de texturas, aplicado al objeto.

FIGURA 41. Ejemplo de una textura horneada.

Este es un proceso que se realiza para cada objeto en la escena, y permite optimizar los tiempos de render de manera considerable, pues no se necesita realizar cálculos adicionales – en el render final -, como los de iluminación o sombras.

Este proceso se utiliza, principalmente, en aplicaciones 3D en tiempo real, como los juegos, para reducir el tiempo de procesado. Sin embargo, es posible utilizarlo en cualquier tipo de proyecto. Este método es bastante útil para series animadas y fondos (*backgrounds*), teniendo cuidado de preparar y pensar bien la escena para obtener resultados convincentes.

También es posible utilizarlo con diferentes motores de render, y con iluminación global simulada o real.

Una vez terminada la escena y definida la textura y la iluminación definitivas, se debe realizar el proceso de "render a textura", para lo cual, a cada uno de los objetos que se deseen *"renderear"*, se le deberá aplicar un mapeado UVW[15].

Aunque la gran mayoría de los programas de desarrollo 3D cuentan con sistemas de render a textura, según el software que se esté utilizando, este proceso se podrá hacer automática o manualmente. El proceso general consta de exportar la textura "horneada", separada según el mapeado UVW, a un formato de imagen, y aplicársela al material de cada modelo, en el canal difuso (diffuse map). En caso de que el software realice esta tarea automáticamente, no habría por qué preocuparse de aplicar mapeados, sino solamente de exportar.

En estos momentos, si se hace un render, el material tendrá incluidos todos los reflejos, luces y sombras que debería recibir de la escena. Sin embargo, al mismo tiempo, el modelo se estará calculando con el resto de los objetos en la escena, pues todavía poseerá los mapas en los demás canales. Esto implica que todo tenga doble intensidad.

Para "limpiar" la escena, se deberán desactivar o eliminar todas las propiedades y mapas de los materiales que se le apliquen cada objeto. Si el render a textura se ha realizado para todos los elementos de la escena, se eliminan todas las luces. Si sólo se ha aplicado a algunos objetos, ellos deberán ser excluidos de la iluminación que reciban.

Para que cada objeto se vea exactamente igual que en el render final sin hornear, se deberá autoiluminar al 100% cada material. De esta manera, se mostrará la textura exactamente igual como se recibió.

Como las texturas resultantes de este proceso serán mapas de bits comunes, será posible editarlas, para incluirles algunos brillos, manchas, reflejos, o todo lo que sea necesario.

4.5.4 Animación.

4.5.4.1 Animación por computadora.

La animación por computadora es la técnica de crear imágenes en movimiento mediante el uso de computadoras. Para crear la ilusión del movimiento, una imagen se muestra en pantalla sustituyéndose rápidamente por una nueva imagen en un fotograma diferente. Esta técnica es idéntica a la manera en que se logra la ilusión de movimiento en las películas y en la televisión.

[15]Mapa de textura con coordenadas UVW correspondiente al objeto.

Para las animaciones 3D, los objetos se modelan en la computadora (modelado) y los caracteres 3D se unen a un esqueleto virtual (huesos). Para crear una cara en 3D se modelan el cuerpo, ojos, boca, etc. del personaje y posteriormente se animan con controladores de animación (por ejemplo *morpher*). Finalmente, se renderiza la animación.

Otro tipo de animación más realista es la captura del movimiento, que requiere que un actor vista un traje especial provisto de sensores, siendo sus movimientos capturados por una computadora y posteriormente incorporados en el personaje 3D.

Dos métodos básicos para la construcción de una secuencia animada son la **animación en tiempo real y** la **animación imagen a imagen.** En una animación por computadora en tiempo real, cada etapa de la secuenciase visualiza a medida que se la genera. Por ello, la animación debe generarse a una frecuencia que sea compatible con las restricciones de la frecuencia de refresco. Para una animación imagen a imagen, se genera de forma separada cada imagen de la secuencia y se almacena. Posteriormente, las imágenes pueden grabarse sobre una película o mostrarse de forma consecutiva en un monitor de vídeo. Las secuencias animadas simples se suelen producir en tiempo real, mientras que las animaciones más complejas se construyen más lentamente, imagen a imagen.

4.5.4.2 Diseño de secuencias de animación.

Un enfoque básico consiste en diseñar secuencias de animación mediante las siguientes etapas de desarrollo:

• Realización del guión.

• Definición de los objetos.

• Especificación de los fotogramas clave.

• Generación de los fotogramas intermedios.

El guión es un resumen de la acción en el que se define la secuencia de movimiento como el conjunto de sucesos básicos que deben tener lugar. Dependiendo del tipo de animación que haya que producir, el guión puede estar compuesto por un conjunto de burdos dibujos y una breve descripción de los movimientos, o puede ser simplemente una lista de las ideas básicas que describen la acción. Originalmente, ese conjunto de burdos dibujos que describen el guión se solía fijar en un panel de gran tamaño que se utilizaba para presentar una vista global del proyecto de animación. De aquí proviene el nombre inglés ***«storyboard».***

Para cada participante en la acción se proporciona una definición del objeto. Los objetos pueden definirse en términos de las formas básicas, como por ejemplo polígonos o ***splines*** superficiales. Además, suele proporcionarse una descripción de los movimientos que tengan que realizar cada personaje u objeto descrito en el guión.

Un fotograma clave es un dibujo detallado de la escena en un cierto momento de la secuencia de animación. Dentro de cada fotograma clave, cada objeto (o personaje) se posiciona de acuerdo con el tiempo correspondiente a dicho fotograma. Algunos fotogramas clave se eligen en las posiciones extremas de la acción, mientras que otros se espacian para que el intervalo de tiempo entre un fotograma clave y el siguiente no sea excesivo. Para los movimientos intrincados se especifican más fotogramas clave que para los movimientos simples o lentos. El desarrollo de los fotogramas clave suele, por regla general, ser responsabilidad de los animadores expertos, siendo normal que se asigne un animador distinto para cada personaje de la animación.

Los fotogramas intermedios *(in-betweens)* son los comprendidos entre los sucesivos fotogramas clave. El número total de imágenes o fotogramas, y por tanto el número total de fotogramas intermedios, necesarios para una animación vendrá determinado por el medio de visualización que se utilice. Las películas requieren24 imágenes por segundo, mientras que los terminales gráficos se refrescan con una tasa de 60 o más imágenes por segundo. Normalmente, los intervalos temporales de la secuencia se configuran de tal modo que haya entre tres y cinco fotogramas intermedios entre cada par de fotogramas clave sucesivos. Dependiendo de la velocidad especificada para la secuencia, será necesario definir más o menos fotogramas clave.

Una tarea importante dentro de la especificación de la animación es la ***descripción de la escena.*** Esto incluye el posicionamiento de los objetos y las fuentes luminosas, la definición de los parámetros fotométricos (intensidades de las fuentes luminosas y propiedades de iluminación de las superficies) y la configuración de los parámetros de la cámara (posición, orientación y características del objetivo). Otra función estándar en este tipo de lenguajes es la ***especificación de acciones,*** que implica la disposición de las trayectorias de movimiento correspondientes a los objetos y a la cámara.

4.5.4.3 Morfismo.

La modificación de la forma de un objeto, para que éste adopte una forma distinta, se denomina morfismo, palabra que proviene de «metamorfosis». Un animador puede modelar un morfismo haciendo que las formas de los polígonos efectúen una transición a lo largo de los fotogramas intermedios comprendidos entre un fotograma clave y el siguiente.

4.5.4.4 Cinemática y dinámica.

También podemos construir secuencias de animación utilizando descripciones ***cinemáticas*** o ***dinámicas.*** Con una descripción cinemática, especificamos la animación proporcionando los parámetros de movimiento (posición, velocidad y aceleración) sin referencia a las causas ni a los objetivos del movimiento. Para una velocidad constante (aceleración cero), designamos los movimientos de los cuerpos rígidos de una escena proporcionando una

posición inicial y un vector de velocidad para cada objeto. Si también especificamos las aceleraciones (tasa de cambio de la velocidad), podemos modelar arranques, paradas y trayectos de movimiento curvos.

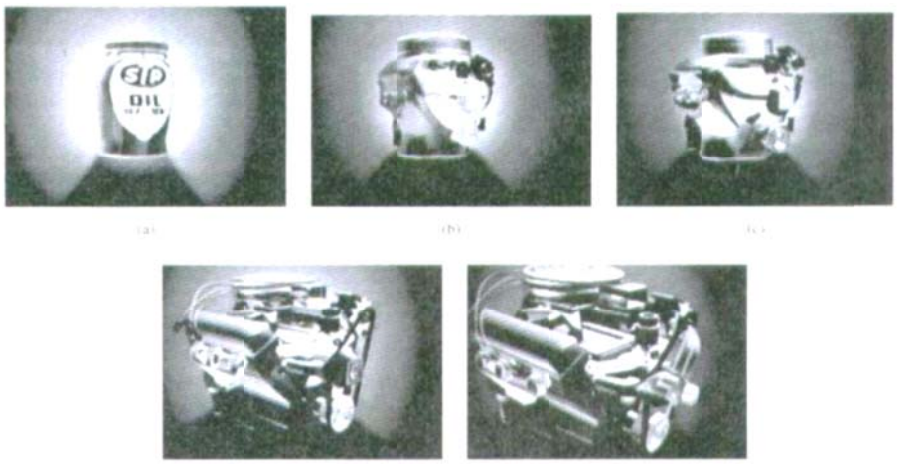

FIGURA 42. Transformación de una lata de aceite para automóviles STP en un motor de automóvil.

La especificación cinemática de un movimiento también puede proporcionarse describiendo la trayectoria del movimiento. Esto se suele hacer mediante curvas de ***spline.***

Un enfoque alternativo consiste en utilizar ***cinemática inversa.*** Con este método, especificamos las posiciones inicial y final de los objetos en instantes determinados y es el sistema el que se encarga de calcular los parámetros del movimiento.

Las descripciones dinámicas requieren la especificación de las fuerzas que producen las velocidades y aceleraciones. La descripción del comportamiento de los objetos en términos de la influencia de las fuerzas se suele denominar ***modelado físico***. Los movimientos de los objetos se obtienen a partir de las ecuaciones de las fuerzas que describen leyes físicas.

Entre las aplicaciones del modelado físico se incluyen los sistemas complejos de cuerpos rígidos y también otros sistemas no rígidos como las ropas y los materiales plásticos. Normalmente, se utilizan métodos numéricos para obtener los parámetros de movimiento incrementalmente a partir de las ecuaciones dinámicas, utilizando condiciones iniciales de valores de contorno.

4.5.4.5 Animación de figuras articuladas.

Una técnica básica para animar personajes humanos, animales, insectos y otras criaturas consiste en modelar las como figuras articuladas, que son estructuras jerárquicas compuestas de un conjunto de enlaces rígidos conectados mediante uniones rotatorias. En términos menos formales, esto quiere decir que modelamos los objetos animados como si fueran esqueletos simplificados, los cuales podemos envolver luego con superficies que representen la piel, el pelo, las plumas, las ropas u otros tipos de recubrimientos.

Los puntos de conexión de una figura articulada se sitúan en los hombros, las caderas, las rodillas y otras articulaciones del esqueleto, y esos puntos de unión siguen unas trayectorias de movimiento especificadas a medida que el cuerpo se traslada. Por ejemplo, cuando se especifica un movimiento para un objeto, el hombro se mueve automáticamente de una cierta forma y, a medida que el hombro se mueve, los brazos también lo hacen. Con estos sistemas, se definen diferentes tipos de movimientos, como por ejemplo andar, correr o saltar, y esos movimientos se asocian con movimientos concretos de las uniones y de los enlaces conectados a ellas.

4.5.4.6 Captura de movimiento (Motion capture).

Cuando la animación por computadora es realizada con esta técnica, un actor real realiza la escena como si fuera el personaje a ser animado. Su movimiento es grabado en una computadora utilizando cámaras de vídeo y marcadores, y ese movimiento es aplicado al personaje animado.

Esta "captura de movimientos" es apropiada en situaciones en las que se requiere un comportamiento realista, pero que las características de un personaje exceden lo que se puede realizar con maquillaje y vestuario convencional.

FIGURA 43. Ejemplos de captura de movimiento para una animación.

4.5.5 Render.

Se llama *render* al proceso final de generar la imagen 2D o animación a partir de la escena creada. Esto puede ser comparado a tomar una foto o en el caso de la animación, a filmar una escena de la vida real. Generalmente se buscan imágenes de calidad fotorrealista, y para este fin se han desarrollado muchos métodos especiales. Las técnicas van desde las más sencillas, como el *render* de malla de alambre (*wire frame rendering*), pasando por el *render* basado en polígonos, hasta las técnicas más modernas como el *Scanline Rendering*[16], el *Raytracing*[17], la *radiosidad* o el *Mapeado de fotones.*

[16]Es un algoritmo que realiza la interpretación de la imagen calculando pixel por pixel la imagen a mostrar, tomando en cuenta el orden de los polígonos en dirección de la perspectiva que debe ser mostrada.

El software de *render* puede simular efectos cinematográficos como el *lens flare*[18], la profundidad de campo, o el *motion blur* (desenfoque de movimiento).

El proceso de *render* necesita una gran capacidad de cálculo, pues requiere simular gran cantidad de procesos físicos complejos. La capacidad de cálculo se ha incrementado rápidamente a través de los años, permitiendo un grado superior de realismo en los *renders*. Estudios de cine que producen animaciones generadas por computadora hacen uso de lo que se conoce como granja de *render* para acelerar la producción de fotogramas.

FIGURA 44. Render de un modelo 3D de la actriz Song Hye Kyo.

4.5.6 Aplicaciones 3D.

Algunos programas para modelado 3D son los siguientes:

Nombre	Compañía	Vínculo
Maya	Autodesk	http://www.autodesk.com/maya
SOFTIMAGE|XSI	Autodesk	http://www.softimage.com
3DStudio MAX	Autodesk	http://www.autodesk.com/3dsmax
LightWave	Newtek	http://www.newtek.com/
Blender	Blender (OpenSource)	http://www.blender.org/
Cinema 4D	Maxon	http://www.maxon.net
Houdini	SideEffects	http://www.sidefx.com/
Rhinoceros	Rhino	http://www.rhino3d.com/
Pov-ray	Povray	http://www.povray.org/
Cheetah 3D	Cheetah 3D	http://www.cheetah3d.com/

[17]Es un algoritmo que realiza la interpretación de la imagen tomando en cuenta simulaciones de las trayectorias de la luz y su paso por las texturas de los objetos de la escena a interpretar.
[18]Efecto de refracción de la luz a través de un lente, se percibe un halo alrededor de una fuente luminosa.

4.6 Realidad virtual.

"La realidad virtual es un medio compuesto por simulaciones de computadora interactivas que reaccionan a la posición y acciones del usuario y producen retroalimentación en uno o más sentidos, generando la sensación de estar inmerso o presente en una simulación[19]".

Un ambiente virtual es una simulación por computadora que proporciona información a uno o varios de nuestros sentidos: visión, sonido, tacto y gusto, con el propósito de que el usuario se sienta inmerso en un mundo que reacciona ante sus acciones. A diferencia de una película tridimensional, donde la información se integra en una secuencia de imágenes definidas de antemano y en la que el participante no puede intervenir; o de una aplicación multimedia, donde la interacción está limitada a seleccionar la secuencia en que se despliegan los objetos bidimensionales, un ambiente virtual es naturalmente tridimensional, dinámico y cambiante según los movimientos o peticiones del usuario, quien puede explorar y experimentar de acuerdo con las situaciones generadas como combinación de su interacción con el mundo virtual y la retroalimentación que éste, a su vez, le proporciona al participante.

La realidad virtual puede ser de dos tipos: inmersiva y no inmersiva. La realidad virtual no inmersiva o también llamada realidad virtual de mesa es aquélla que se crea cuando el participante explora diversos ambientes haciendo uso de los dispositivos de hardware comunes: mouse, monitor, tarjeta de sonido y bocinas. Ejemplo de ésta son las aplicaciones que utilizan Internet y lenguajes como el VRML con objeto de interaccionar en tiempo real con diferentes personas en espacios y ambientes tridimensionales, sin la necesidad de dispositivos adicionales en la computadora.

Este tipo de aplicaciones son llamativas para los usuarios, pues no requieren de hardware especializado y, por lo tanto, son de fácil acceso; no obstante, tienen como inconveniente que los sentidos de los participantes se distraen por eventos ajenos a la simulación, como por ejemplo, salir del campo de visión de la simulación al mover la cabeza, o utilizar el mouse que no es un dispositivo de interacción natural, con lo que se percibe de manera inconsciente que el ambiente virtual no constituye una realidad.

4.6.1 Métodos inmersivos.

Buscan crear la sensación de encontrarse dentro de un ambiente específico; para lograrlo se generan simulaciones con la mayor calidad posible de despliegue, junto con formas naturales de interacción donde se utilizan sistemas sofisticados de alta calidad de despliegue con efecto de profundidad como cascos o proyectores de alta resolución, equipo de cómputo

[19] Ramos Nava, María del Carmen.Enter@te en línea año 2, número 24, noviembre de 2003. Dirección General de Servicios de Cómputo Académico, UNAM. http://www.enterate.unam.mx/Articulos/2003/noviembre/realivirt.htm

capaz de controlar la simulación, el despliegue, los dispositivos y la interacción a una velocidad adecuada con objeto de que el usuario tenga una respuesta rápida.

Los dispositivos de interacción regularmente se basan en sistemas de captura de los movimientos del usuario, de tal manera que se realizan de forma natural, sin tener que concentrarse en cambiar protocolos de interacción, como sucede al emplear el mouse o teclado. Por la potencialidad que estos sistemas ofrecen, el área donde se pueden utilizar es muy extensa.

4.6.2 Despliegue visual.

Las pistas visuales, quizás, son la parte más importante de retroalimentación en los sistemas de realidad virtual inmersiva; para obtener un sentido de realidad, regularmente, se producen vistas tridimensionales con visión estereoscópica.

Uno de los dispositivos de uso común para generar estereoscopía son los llamados cascos o despliegues binoculares montados en la cabeza (HMD), que incorporan pequeños monitores que se colocan uno frente a cada ojo. Cada monitor visualiza la perspectiva que el ojo correspondiente vería en un ambiente real.

Otras alternativas para lograr la sensación de inmersión son los sistemas de proyección en múltiples pantallas, como el sistema CAVE, desarrollado por la Universidad de Illinois en 1992, que muestra una serie de proyecciones generadas por computadora que envuelven al usuario en un espacio en forma de cuarto. El uso de lentes especiales permite ver al usuario en un tiempo, sólo una de las vistas, de tal forma que se engaña al cerebro al generar el efecto de profundidad de la escena.

4.6.3 Audio 3D.

Además de los efectos visuales, el mundo virtual incorpora un campo de audio tridimensional para reflejar las condiciones que representa. El campo de sonido tiene que reaccionar en paredes, fuentes múltiples de sonidos y ruido de fondo. Al igual que dos perspectivas visuales producen una imagen en 3D, dos perspectivas de audio producen un paisaje sonoro en 3D. Con bocinas estereofónicas fijas, los sonidos derecho e izquierdo se mezclan y ambos oídos lo reciben. Si se usan audífonos y se presentan las perspectivas acústicas correctas para cada oído, se puede preservar el aspecto espacial de los sonidos.

4.6.4 Localización y seguimiento.

Existen sistemas de localización y seguimiento para medir posiciones y orientaciones, de esta forma, la computadora determina el modo de visualizar el entorno virtual de manera que el participante perciba que se encuentra dentro de él. En un mundo virtual en el que observamos un objeto y nos asomamos por debajo de éste, el localizador debe percibir

cualquier cambio de posición para ajustar la visualización y poder apreciar la parte de abajo del objeto, como sucede en el mundo real.

También es importante la velocidad con la que el sistema muestrea la posición del usuario, así como el tiempo que le toma a la computadora regenerar la nueva imagen; pues cuando este proceso es lento, la vista envía al cerebro informaciones contradictorias acerca de la dirección donde apunta la cabeza, provocando mareo en el espectador.

4.6.5 Otros dispositivos de interacción.

Para comunicar a las computadoras que simulan el mundo virtual, se utilizan diversos dispositivos de entrada, uno de los más utilizados es el mouse tridimensional, extensión del mouse común, usado en los equipos de cómputo personales con un conjunto de botones que indican una cierta acción pero que, además, añaden la posibilidad de mover los objetos en los ejes de rotación cartesianos, con lo que se extiende su funcionalidad a espacios 3D.

Los guantes sensitivos funcionan como un localizador que percibe la posición y orientación de la mano; para ello, cuentan con algunos sensores de flexión que permiten medir la curvatura de los dedos. Cuando se mueve el guante en el espacio, el cursor en 3D del ambiente virtual reacciona y se dirige a la dirección indicada por el guante; así, se pueden tomar, arrastrar y soltar objetos virtuales, moverlos horizontal o verticalmente, en profundidad y rotarlos alrededor de los tres ejes coordenados.

4.6.6 Retroalimentación.

Reproducir una interfaz realista, significa desarrollar retroalimentación táctil y de fuerza que corresponda con los objetos del mundo real. La retroalimentación de la fuerza tiene que ver con la manera como los ambientes virtuales afectan al usuario, por ejemplo, las paredes deberían de parar a alguien, en vez de permitirle pasar por ella; mientras que una pelota debería golpear al usuario si choca con ella. La retroalimentación táctil es la manera como un objeto virtual se siente. La temperatura, la forma, la firmeza y la textura son algunas de las piezas de información obtenidas mediante el tacto. Hoy en día, no existe una sola interfaz que simule todas estas interacciones juntas, pero, hay dispositivos que simulan una u otra, como sucede con las cirugías que pueden seguir la forma de los órganos con los que se esté interaccionando, de tal manera que los estudiantes de medicina pueden entrenarse mediante cirugías virtuales. Existen, también, sistemas que por medio de vibración permiten sentir la forma de un objeto virtual.

4.6.7 Cómputo de alto rendimiento.

Los elementos anteriores, deben ser coordinados y/o generados por medio de un sistema de cómputo suficientemente poderoso para sincronizar cada una de las partes en tiempo real.

La finalidad es que el usuario perciba el ambiente virtual de forma similar al de la vida real, en ese sentido, la computadora o computadoras del sistema deberán ser capaces de desplegar imágenes de los espacios tridimensionales a una velocidad de por lo menos 30 cuadros por segundo. Por otro lado, para sincronizar los movimientos del usuario con los dispositivos, éstos deben ser muestreados con rapidez y exactitud, de tal manera que se produzca una respuesta rápida y confiable, de acuerdo con la interacción realizada.

4.6.8 CAVE.

CAVE es el acrónimo de "Cave Automatic Virtual Environment" que en español algo como "Entorno Virtual Automático tipo Cueva".

Se trata de un entorno de realidad virtual inmersiva que consiste en un cubo en el que el usuario está en el interior del mismo, y son proyectadas en cada una de las paredes del cubo imágenes desde el exterior. Existen CAVEs en los que se proyecta las imágenes en 3, 4, 5 o 6 caras del cubo.

Generalmente el usuario lleva una lentes estereoscópicos que sincronizados con las proyecciones le permiten ver en 3D las imágenes, e incluso contemplar objetos flotando dentro del cubo que puede observar desde todos los ángulos.

El primer CAVE fue creado por el equipo EVL (Electronic Visualization Laboratory) de la Universidad de Illinois (Chicago-USA), y fue presentado por primera vez en 1992 en la conferencia anual SIGGRAPH[20].

FIGURA 45. Ejemplo de una CAVE.

[20]ACM SIGGRAPH. Association for Computing Machinery's Special Interest Group on Computer Graphics and Interactive Techniques.

4.7 Ingeniería de programación.

4.7.1 Conceptos de la ingeniería de programación.

4.7.1.1 Software.

"El software no son sólo programas, sino todos los documentos asociados y la configuración de datos que se necesitan para hacer que estos programas operen de manera correcta. Por lo general, un sistema de software consiste en diversos programas independientes, archivos de configuración que se utilizan para ejecutar estos programas, un sistema de documentación que describe la estructura del sistema, la documentación para el usuario que explica cómo utilizar el sistema y sitios web que permitan a los usuarios descargar la información de productos recientes[21]".

Existen dos tipos de productos de software:

1. ***Productos genéricos.*** Son sistemas aislados producidos por una organización de desarrollo y que se venden al mercado abierto a cualquier cliente que le sea posible comprarlos. Ejemplos de este tipo de producto son el software para PCs tales como bases de datos, procesadores de texto, paquetes de dibujo y herramientas de gestión de proyectos.

2. ***Productos personalizados (o hechos a medida).*** Son sistemas requeridos por un cliente en particular. Un contratista de software desarrolla el software especialmente para ese cliente. Ejemplos de este tipo de software son los sistemas de control para instrumentos electrónicos, sistemas desarrollados para llevar a cabo procesos de negocios específicos y sistemas de control del tráfico aéreo.

Una diferencia importante entre estos diferentes tipos de software es que, en los productos genéricos, la organización que desarrolla el software controla su especificación. La especificación de los productos personalizados, por lo general, es desarrollada y controlada por la organización que compra el software. Los desarrolladores de software deben trabajar con esa especificación.

4.7.1.2 Ingeniería del software.

La ingeniería del software es una disciplina de la ingeniería que comprende todos los aspectos de la producción de software desde las etapas iniciales de la especificación del sistema, hasta el mantenimiento de éste después de que se utiliza.

[21] Ingeniería (bibliografía del libro)

4.7.1.3 Métodos de la ingeniería del software.

Un método de ingeniería del software es un enfoque estructurado para el desarrollo de software cuyo propósito es facilitar la producción de software de alta calidad de una forma costeable. Métodos como Análisis Estructurado (DeMarco, 1978) y JSD (Jackson, 1983) fueron los primeros desarrollados en los años 70. Estos métodos intentaron identificarlos componentes funcionales básicos de un sistema, de tal forma que los métodos orientados a funciones aún se utilizan ampliamente. En los años 80 y 90, estos métodos orientados a funciones fueron complementados por métodos orientados a objetos, como los propuestos por Booch (1994) y Rumbaugh (Rumbaugh *et al,* 1991). Estos diferentes enfoques se han integrado en un solo enfoque unificado basado en el Lenguaje de Modelado Unificado (UML) (Booch *et al,* 1999; Rumbaugh *et al,* 1999; Rumbaugh *et al.,* 1999).

No existe un método ideal, y métodos diferentes tienen distintas áreas donde son aplicables. Por ejemplo, los métodos orientados a objetos a menudo son apropiados para sistemas interactivos, pero no para sistemas con requerimientos rigurosos de tiempo real.

Todos los métodos se basan en la idea de modelos gráficos de desarrollo de un sistema y en el uso de estos modelos como un sistema de especificación o diseño. Los métodos incluyen varios componentes diferentes.

4.7.1.4 CASE.

CASE (Ingeniería del Software Asistida por Computadora) comprende un amplio abanico de diferentes tipos de programas que se utilizan para ayudar a las actividades del proceso del software, como el análisis de requerimientos, el modelado de sistemas, la depuración y las pruebas.

Las herramientas CASE también incluyen un generador de código que automáticamente genera código fuente a partir del modelo del sistema y de algunas guías de procesos para los ingenieros de software.

4.7.1.5 Atributos de un buen software.

Así como los servicios que proveen, los productos de software tienen un cierto número de atributos asociados que reflejan la calidad de ese software. Estos atributos no están directamente asociados con lo que el software hace. Más bien, reflejan su comportamiento durante su ejecución y en la estructura y organización del programa fuente y en la documentación asociada.

Ejemplos de estos atributos (algunas veces llamados atributos no funcionales) son el tiempo de respuesta del software a una pregunta del usuario y la comprensión del programa fuente.

El conjunto específico de atributos que se espera de un sistema de software depende de su aplicación. Por lo tanto, un sistema bancario debe ser seguro, un juego interactivo debe tener capacidad de respuesta, un interruptor de un sistema telefónico debe ser fiable, etcétera. Esto se generaliza en el conjunto de atributos que se muestra en la Figura 46, el cual tiene las características esenciales de un sistema de software bien diseñado.

Mantenibilidad.	El software debe escribirse de tal forma que pueda evolucionar para cumplir las necesidades de cambio de los clientes. Éste es un atributo crítico debido a que el cambio en el software es una consecuencia inevitable de un cambio en el entorno de negocios.
Confiabilidad.	La confiabilidad del software tiene un gran número de características, incluyendo la fiabilidad, protección y seguridad. El software confiable no debe causar daños físicos o económicos en el caso de una falla del sistema.
Eficiencia.	El software no debe hacer que se malgasten los recursos del sistema, como la memoria y los ciclos de procesamiento. Por lo tanto, la eficiencia incluye tiempos de respuesta y de procesamiento, utilización de la memoria, etc.
Usabilidad.	El software debe ser fácil de utilizar, sin esfuerzo adicional, por el usuario para quien está diseñado. Esto significa que debe tener una interfaz de usuario apropiada y una documentación adecuada.

FIGURA 46. Atributos esenciales de un buen software.

4.7.2 Ingeniería de sistemas.

4.7.2.1 Definición de requerimientos del sistema.

Las definiciones de requerimientos del sistema especifican qué es lo que el sistema debe hacer (sus funciones) y sus porpiedades esenciales y deseables. Crear definiciones de requerimientos del sistema requiereconsultar con los clientes del sistema y con los usuarios finales. Esta fase de definiciónde requerimientos usualmente se concentra en la derivación de tres tipos de requerimientos:

1. ***Requerimientos funcionales abstractos.*** Las funciones básicas que el sistema debe proporcionarse definen en un nivel abstracto. Una especificación más detallada de requerimientos funcionales tiene lugar en el nivel de subsistemas.

2. ***Propiedades del sistema.*** Son propiedades emergentes no funcionales del sistema, tales como la disponibilidad, el rendimiento y la seguridad.Estas propiedades no funcionales del sistema afectan a los requerimientos detodos los subsistemas.

3. ***Características que no debe mostrar el sistema.*** Algunas veces es tan importante especificarlo que el sistema no debe hacer como especificar lo que debe hacer.

Una parte importante de la fase de definición de requerimientos es establecer un conjuntocompleto de objetivos que el sistema debe cumplir. Éstos no necesariamente deben expresarseforzosamente en términos de la funcionalidad del sistema, pero deben definir por qué seconstruye el sistema para un entorno particular.

3.7.2.2 Diseño del sistema.

El diseño del sistema se centra en proporcionar la funcionalidad del sistema a través de sus diferentes componentes. Las actividades que se realizan en este proceso son:

1. ***Dividir requerimientos.*** Analizar los requerimientos y organizarlos en grupos afines.

2. ***Identificar subsistemas.*** Debe identificar los diferentes subsistemas que pueden, individual o colectivamente, cumplir los requerimientos. Los grupos de requerimientos están normalmente relacionados con los subsistemas, de tal forma que esta actividad y la de división de requerimientos se pueden fusionar.

3. ***Asignar requerimientos a los subsistemas.***

4. ***Especificar la funcionalidad de los subsistemas.***

5. ***Definir las interfaces del subsistema.*** Una vez que estas interfaces se han acordado, es posible desarrollar estos subsistemas en paralelo.

4.7.3 Procesos del software.

Un proceso del software es un conjunto de actividades que conducen a la creación de un producto software. Estas actividades pueden consistir en el desarrollo de software desde cero en un lenguaje de programación estándar como Java o C. Sin embargo, cada vez más, se desarrolla nuevo software ampliando y modificando los sistemas existentes y configurando e integrando software comercial o componentes del sistema.

Los procesos del software son complejos y, como todos los procesos intelectuales y creativos, dependen de las personas que toman decisiones y juicios. Debido a la necesidad

de juzgar y crear, los intentos para automatizar estos procesos han tenido un éxito limitado. Las herramientas de ingeniería del software asistida por computadora (CASE) pueden ayudar a algunas actividades del proceso.

Una razón por la cual la eficacia de las herramientas CASE está limitada se halla en la inmensa diversidad de procesos del software. No existe un proceso ideal, y muchas organizaciones han desarrollado su propio enfoque para el desarrollo del software. Los procesos han evolucionado para explotar las capacidades de las personas de una organización, así como las características específicas de los sistemas que se están desarrollando. Para algunos sistemas, como los sistemas críticos, se requiere un proceso de desarrollo muy estructurado. Para sistemas de negocio, con requerimientos rápidamente cambiantes, un proceso flexible y ágil probablemente sea más efectivo.

Aunque existen muchos procesos diferentes de software, algunas actividades fundamentales son comunes para todos ellos:

1. ***Especificación del software.*** Se debe definir la funcionalidad del software y las restricciones en su operación.

2. ***Diseño e implementación del software.*** Se debe producir software que cumpla su especificación.

3. ***Validación del software.*** Se debe validar el software para asegurar que hace lo que el cliente desea.

4. ***Evolución del software.*** El software debe evolucionar para cubrir las necesidades cambiantes del cliente.

4.7.3.1 Modelos del proceso del software (paradigmas de procesos).

Un modelo del proceso del software es una representación abstracta de un proceso del software. Cada modelo de proceso representa un proceso desde una perspectiva particular, y así proporciona sólo información parcial sobre ese proceso.

Estos modelos generales son abstracciones de los procesos que se pueden utilizar para explicar diferentes enfoques para el desarrollo de software. Puede pensarse en ellos como marcos de trabajo del proceso que pueden ser extendidos y adaptados para crear procesos más específicos de ingeniería del software.

1. ***El modelo en cascada.*** Considera las actividades fundamentales del proceso de especificación, desarrollo, validación y evolución, y los representa como fases separadas del proceso, tales como la especificación de requerimientos, el diseño del software, la implementación, las pruebas, etcétera.

2. ***Desarrollo evolutivo.*** Este enfoque entrelaza las actividades de especificación, desarrollo y validación. Un sistema inicial se desarrolla rápidamente a partir de especificaciones abstractas. Este se refina basándose en las peticiones del cliente para producir un sistema que satisfaga sus necesidades.

3. ***Ingeniería del software basada en componentes.*** Este enfoque se basa en la existencia de un número significativo de componentes reutilizables. El proceso de desarrollo del sistema se enfoca en integrar estos componentes en el sistema más que en desarrollarlos desde cero.

Estos tres modelos de procesos genéricos se utilizan ampliamente en la práctica actual dela ingeniería del software. No se excluyen mutuamente y a menudo se utilizan juntos, especialmente para el desarrollo de sistemas grandes. Los subsistemas dentro de un sistema más grande pueden ser desarrollados utilizando enfoques diferentes.

a) El modelo en cascada.

Debido a la cascada de una fase a otra, dicho modelo se conoce como modelo en cascada o como ciclo de vida del software. Las principales etapas de este modelo se transforman en actividades fundamentales de desarrollo:

1. ***Análisis y definición de requerimientos.*** Los servicios, restricciones y metas del sistema se definen a partir de las consultas con los usuarios. Entonces, se definen en detalle y sirven como una especificación del sistema.

2. ***Diseño del sistema y del software.*** El proceso de diseño del sistema divide los requerimientos en sistemas hardware o software. Establece una arquitectura completa del sistema. El diseño del software identifica y describe las abstracciones fundamentales del sistema software y sus relaciones.

3. ***Implementación y prueba de unidades.*** Durante esta etapa, el diseño del software se lleva a cabo como un conjunto o unidades de programas. La prueba de unidades implica verificar que cada una cumpla su especificación.

4. ***Integración y prueba del sistema.*** Los programas o las unidades individuales de programas se integran y prueban como un sistema completo para asegurar que se cumplan los requerimientos del software. Después de las pruebas, el sistema software se entrega al cliente.

5. ***Funcionamiento y mantenimiento.*** Por lo general (aunque no necesariamente), ésta es la fase más larga del ciclo de vida. El sistema se instala y se pone en funcionamiento práctico. El mantenimiento implica corregir errores no descubiertos en las etapas anteriores del ciclo de vida, mejorar la implementación de las unidades del sistema y resaltarlos servicios del sistema una vez que se descubren nuevos requerimientos.

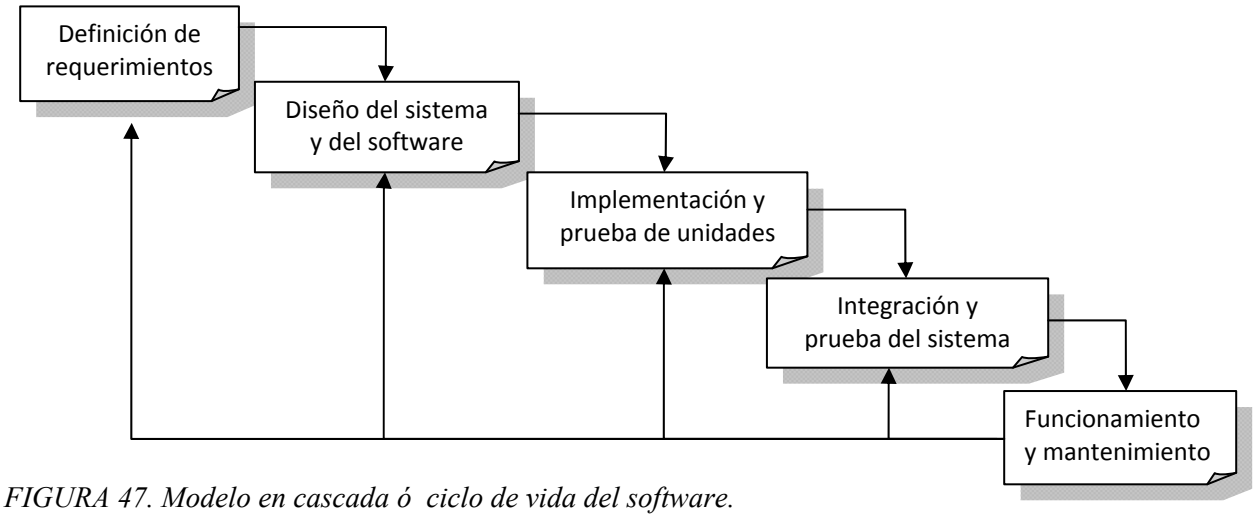

FIGURA 47. Modelo en cascada ó ciclo de vida del software.

La siguiente fase no debe empezar hasta que la fase previa haya finalizado. En la práctica, estas etapas se superponen y proporcionan información a las otras. Durante el diseño se identifican los problemas con los requerimientos; durante el diseño del código se encuentran problemas, y así sucesivamente. El proceso del software no es un modelo lineal simple, sino que implica una serie de iteraciones de las actividades de desarrollo. Debido a los costos de producción y aprobación de documentos, las iteraciones son costosa se implican rehacer el trabajo.

Durante la fase final del ciclo de vida (funcionamiento y mantenimiento), el software se pone en funcionamiento. Se descubren errores y omisiones en los requerimientos originales del software. Los errores de programación y de diseño emergen y se identifica la necesidad de una nueva funcionalidad. Por tanto, el sistema debe evolucionar para mantenerse útil. Hacer estos cambios (mantenimiento del software) puede implicar repetir etapas previas del proceso.

Las ventajas del modelo en cascada son que la documentación se produce en cada fase y que éste cuadra con otros modelos del proceso de ingeniería. Su principal problema es su inflexibilidad al dividir el proyecto en distintas etapas.

b) Desarrollo evolutivo.

El desarrollo evolutivo se basa en la idea de desarrollar una implementación inicial, exponiéndola a los comentarios del usuario y refinándola a través de las diferentes versiones hasta que se desarrolla un sistema adecuado (Figura 48). Las actividades de especificación, desarrollo y validación se entrelazan en vez de separarse, con una rápida retroalimentación entre éstas.

Existen dos tipos de desarrollo evolutivo:

1. ***Desarrollo exploratorio,*** donde el objetivo del proceso es trabajar con el cliente para explorar sus requerimientos y entregar un sistema final. El desarrollo empieza con las partes del sistema que se comprenden mejor. El sistema evoluciona agregando nuevos atributos propuestos por el cliente.

2. ***Prototipos desechables,*** donde el objetivo del proceso de desarrollo evolutivo es comprenderlos requerimientos del cliente y entonces desarrollar una definición mejorada de los requerimientos para el sistema. El prototipo se centra en experimentar con los requerimientos del cliente que no se comprenden del todo.

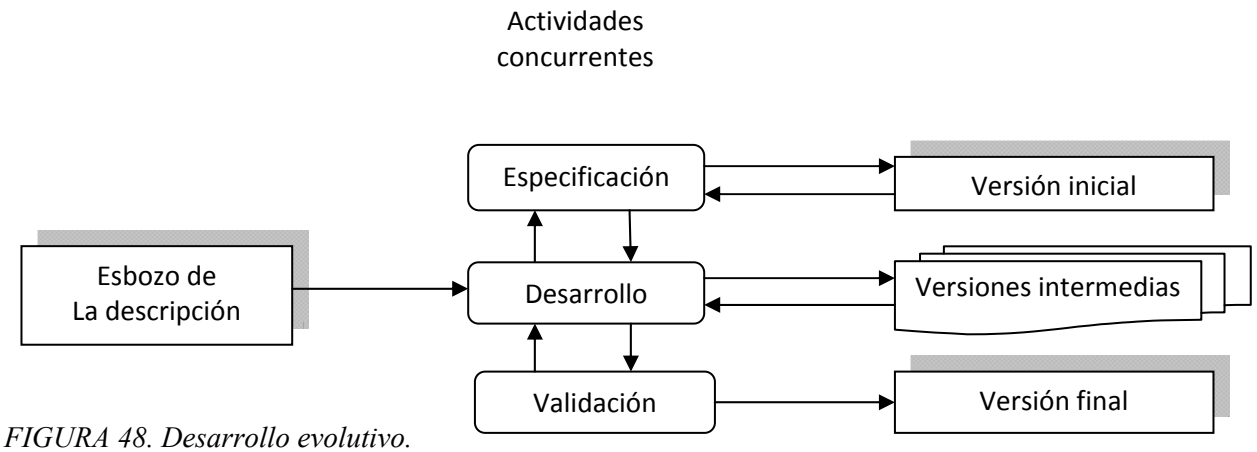

FIGURA 48. Desarrollo evolutivo.

En la producción de sistemas, un enfoque evolutivo para el desarrollo de software suele ser más efectivo que el enfoque en cascada, ya que satisface las necesidades inmediatas de los clientes. La ventaja de un proceso del software que se basa en un enfoque evolutivo es que la especificación se puede desarrollar de forma creciente. Tan pronto como los usuarios desarrollen un mejor entendimiento de su problema, éste se puede reflejar en el sistema software.

Sin embargo, desde una perspectiva de ingeniería y de gestión, el enfoque evolutivo tiene dos problemas:

1. ***El proceso no es visible.*** Los administradores tienen que hacer entregas regulares para medir el progreso. Si los sistemas se desarrollan rápidamente, no es rentable producir documentos que reflejen cada versión del sistema.

2. ***A menudo los sistemas tienen una estructura deficiente.*** Los cambios continuos tienden a corromper la estructura del software. Incorporar cambios en él se convierte cada vez más en una tarea difícil y costosa.

Para sistemas pequeños y de tamaño medio (hasta 500.000 líneas de código), el enfoque evolutivo de desarrollo es el mejor. Los problemas del desarrollo evolutivo se hacen

particularmente agudos para sistemas grandes y complejos con un periodo de vida largo, donde diferentes equipos desarrollan distintas partes del sistema. Es difícil establecer una arquitectura del sistema estable usando este enfoque, el cual hace difícil integrar las contribuciones de los equipos.

Para sistemas grandes, se recomienda un proceso mixto que incorpore las mejores características del modelo en cascada y del desarrollo evolutivo. Esto puede implicar desarrollar un prototipo desechable utilizando un enfoque evolutivo para resolver incertidumbres en la especificación del sistema. Puede entonces reimplementarse utilizando un enfoque más estructurado.

Las partes del sistema bien comprendidas se pueden especificar y desarrollar utilizando un proceso basado en el modelo en cascada. Las otras partes del sistema, como la interfaz de usuario, que son difíciles de especificar por adelantado, se deben desarrollar siempre utilizando un enfoque de programación exploratoria.

c) Ingeniería del software basada en componentes.

En la mayoría de los proyectos de software existe algo de reutilización de software. Por lo general, esto sucede informalmente cuando las personas que trabajan en el proyecto conocen diseños o código similares al requerido. Los buscan, los modifican según lo creen necesario y los incorporan en el sistema. En el enfoque evolutivo la reutilización es a menudo indispensable para el desarrollo rápido de sistemas.

Esta reutilización informal es independiente del proceso de desarrollo que se utilice. Sin embargo, existe un enfoque de desarrollo de software denominado ingeniería del software basada en componentes (CBSE) que se basa en la reutilización.

Este enfoque basado en la reutilización se compone de una gran base de componentes software reutilizables y de algunos marcos de trabajo de integración para éstos. Algunas veces estos componentes son sistemas por sí mismos que se pueden utilizar para proporcionar una funcionalidad específica, como dar formato al texto o efectuar cálculos numéricos. En la Figura 49 se muestra el modelo del proceso genérico para la CBSE.

Aunque la etapa de especificación de requerimientos y la de validación son comparables con otros procesos, las etapas intermedias en el proceso orientado a la reutilización son diferentes.

Estas etapas son:

1. ***Análisis de componentes.*** Dada la especificación de requerimientos, se buscan los componentes para implementar esta especificación. Por lo general, no existe una

concordancia exacta y los componentes que se utilizan sólo proporcionan parte de la funcionalidad requerida.

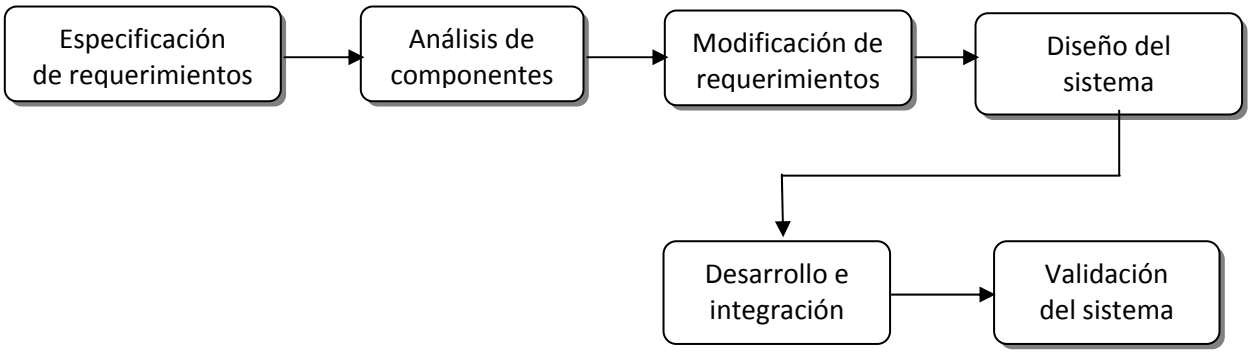

FIGURA49. Ingeniería del software basada en componentes.

2. ***Modificación de requerimientos.*** En esta etapa, los requerimientos se analizan utilizando información acerca de los componentes que se han descubierto. Entonces, estos componentes se modifican para reflejar los componentes disponibles. Si las modificaciones no son posibles, la actividad de análisis de componentes se puede llevar a cabo nuevamente para buscar soluciones alternativas.

3. ***Diseño del sistema con reutilización.*** En esta fase se diseña o se reutiliza un marco de trabajo para el sistema. Los diseñadores tienen en cuenta los componentes que se reutilizan y organizan el marco de trabajo para que los satisfaga. Si los componentes reutilizables no están disponibles, se puede tener que diseñar nuevo software.

4. ***Desarrollo e integración.*** Para crear el sistema, el software que no se puede adquirir externamente se desarrolla, y los componentes y los sistemas COTS[22] se integran. En este modelo, la integración de sistemas es parte del proceso de desarrollo, más que una actividad separada.

La ingeniería del software basada en componentes tiene la ventaja obvia de reducir la cantidad de software a desarrollarse y así reduce los costos y los riesgos. Por lo general, también permite una entrega más rápida del software. Sin embargo, los compromisos en los requerimientos son inevitables, y esto puede dar lugar a un sistema que no cumpla las necesidades reales de los usuarios. Más aún: si las nuevas versiones de los componentes reutilizables no están bajo el control de la organización que los utiliza, se pierde parte del control sobre la evolución del sistema.

[22]COTS: Commercial Off The Shelf, componentes comerciales.

4.7.3.2 Iteración de procesos.

El proceso del software no es un proceso único; más bien, las actividades del proceso se repiten regularmente conforme el sistema se rehace en respuesta a peticiones de cambios.

A continuación se describen dos modelos de procesos que han sido diseñados explícitamente para apoyar la iteración de procesos:

1. ***Entrega incremental.*** La especificación, el diseño y la implementación del software se dividen en una serie de incrementos, los cuales se desarrollan por turnos.

2. ***Desarrollo en espiral.*** El desarrollo del sistema gira en espiral hacia fuera, empezando con un esbozo inicial y terminando con el desarrollo final del mismo.

La esencia de los procesos iterativos es que la especificación se desarrolla junto con el software. Sin embargo, esto crea conflictos con el modelo de obtención de muchas organizaciones donde la especificación completa del sistema es parte del contrato de desarrollo del mismo.

a) Entrega incremental.

El modelo de desarrollo en cascada requiere que los clientes de un sistema cumplan un conjunto de requerimientos antes de que se inicie el diseño y que el diseñador cumpla estrategias particulares de diseño antes de la implementación. Los cambios de requerimientos implican rehacer el trabajo de captura de éstos, de diseño e implementación. Sin embargo, la separación en el diseño y la implementación deben dar lugar a sistemas bien documentados susceptibles de cambio. En contraste, un enfoque de desarrollo evolutivo permite que los requerimientos y las decisiones de diseño se retrasen, pero también origina un software que puede estar débilmente estructurado y difícil de comprender y mantener.

La entrega incremental (Figura 50) es un enfoque intermedio que combina las ventajas de estos modelos. En un proceso de desarrollo incremental. Los clientes identifican, a grandes rasgos, los servicios que proporcionará el sistema. Identifican qué servicios son más importantes y cuáles menos. Entonces, se definen varios incrementos en donde cada uno proporciona un subconjunto de la funcionalidad del sistema. La asignación de servicios a los incrementos depende de la prioridad del servicio con los servicios de prioridad más alta entregados primero.

Una vez que los incrementos del sistema se han identificado, los requerimientos para los servicios que se van a entregar en el primer incremento se definen en detalle, y éste se desarrolla. Durante el desarrollo, se puede llevar a cabo un análisis adicional de requerimientos para los requerimientos posteriores, pero no se aceptan cambios en los requerimientos para el incremento actual.

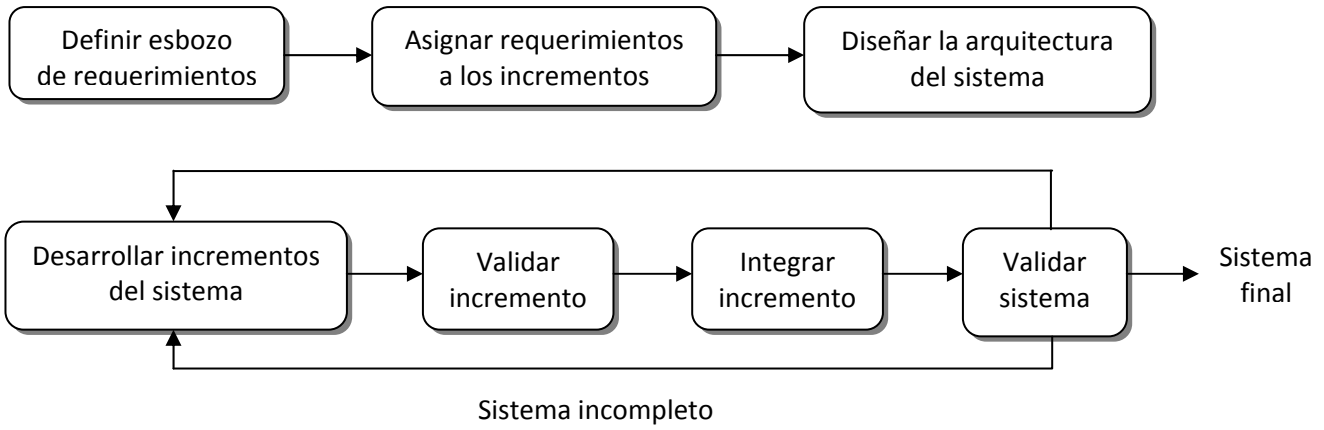

FIGURA 50. Entrega incremental.

Una vez que un incremento se completa y entrega, los clientes pueden ponerlo en servicio. Esto significa que tienen una entrega temprana de parte de la funcionalidad del sistema. Pueden experimentar con el sistema, lo cual les ayuda a clarificar sus requerimientos para los incrementos posteriores y para las últimas versiones del incremento actual. Tan pronto como se completan los nuevos incrementos, se integran en los existentes de tal forma que la funcionalidad del sistema mejora con cada incremento entregado. Los servicios comunes se pueden implementar al inicio del proceso o de forma incremental tan pronto como sean requeridos por un incremento.

Este proceso de desarrollo incremental tiene varias ventajas:

1. Los clientes no tienen que esperar hasta que el sistema completo se entregue para sacar provecho de él. El primer incremento satisface los requerimientos más críticos de tal forma que pueden utilizar el software inmediatamente.

2. Los clientes pueden utilizar los incrementos iniciales como prototipos y obtener experiencia sobre los requerimientos de los incrementos posteriores del sistema.

3. Existe un bajo riesgo de un fallo total del proyecto. Aunque se pueden encontrar problemas en algunos incrementos, lo normal es que el sistema se entregue de forma satisfactoria al cliente.

4. Puesto que los servicios de más alta prioridad se entregan primero, y los incrementos posteriores se integran en ellos, es inevitable que los servicios más importantes del sistema sean a los que se les hagan más pruebas. Esto significa que es menos probable que los clientes encuentren fallos de funcionamiento del software en las partes más importantes del sistema.

b) Desarrollo en espiral.

Más que representar el proceso del software como una secuencia de actividades con retrospectiva de una actividad a otra, se representa como una espiral. Cada ciclo en la espiral representa una fase del proceso del software. Así, el ciclo más interno podría referirse a la viabilidad del sistema, el siguiente ciclo a la definición de requerimientos, el siguiente ciclo al diseño del sistema, y así sucesivamente.

Cada ciclo de la espiral se divide en cuatro sectores:

1. ***Definición de objetivos.*** Para esta fase del proyecto se definen los objetivos específicos. Se identifican las restricciones del proceso y el producto, y se traza un plan detallado de gestión. Se identifican los riesgos del proyecto. Dependiendo de estos riesgos, se planean estrategias alternativas.

2. ***Evaluación y reducción de riesgos.*** Se lleva a cabo un análisis detallado para cada uno de los riesgos del proyecto identificados. Se definen los pasos para reducir dichos riesgo. Por ejemplo, si existe el riesgo de tener requerimientos inapropiados, se puede desarrollar un prototipo del sistema.

3. ***Desarrollo y validación.*** Después de la evaluación de riesgos, se elige un modelo para el desarrollo del sistema. Por ejemplo, si los riesgos en la interfaz de usuario son dominantes, un modelo de desarrollo apropiado podría ser la construcción de prototipos evolutivos. Si los riesgos de seguridad son la principal consideración, un desarrollo basado en transformaciones formales podría ser el más apropiado, y así sucesivamente. El modelo en cascada puede ser el más apropiado para el desarrollo si el mayor riesgo identificado es la integración de los subsistemas.

4. ***Planificación.*** El proyecto se revisa y se toma la decisión de si se debe continuar con un ciclo posterior de la espiral. Si se decide continuar, se desarrollan los planes para la siguiente fase del proyecto.

La diferencia principal entre el modelo en espiral y los otros modelos del proceso del software es la consideración explícita del riesgo en el modelo en espiral. Informalmente, el riesgo significa sencillamente algo que puede ir mal. Por ejemplo, si la intención es utilizar un nuevo lenguaje de programación, un riesgo es que los compiladores disponibles sean poco fiables o que no produzcan código objeto suficientemente eficiente. Los riesgos originan problemas en el proyecto, como los de confección de agendas y excesos en los costos; por lo tanto, la disminución de riesgos es una actividad muy importante en la gestión del proyecto.

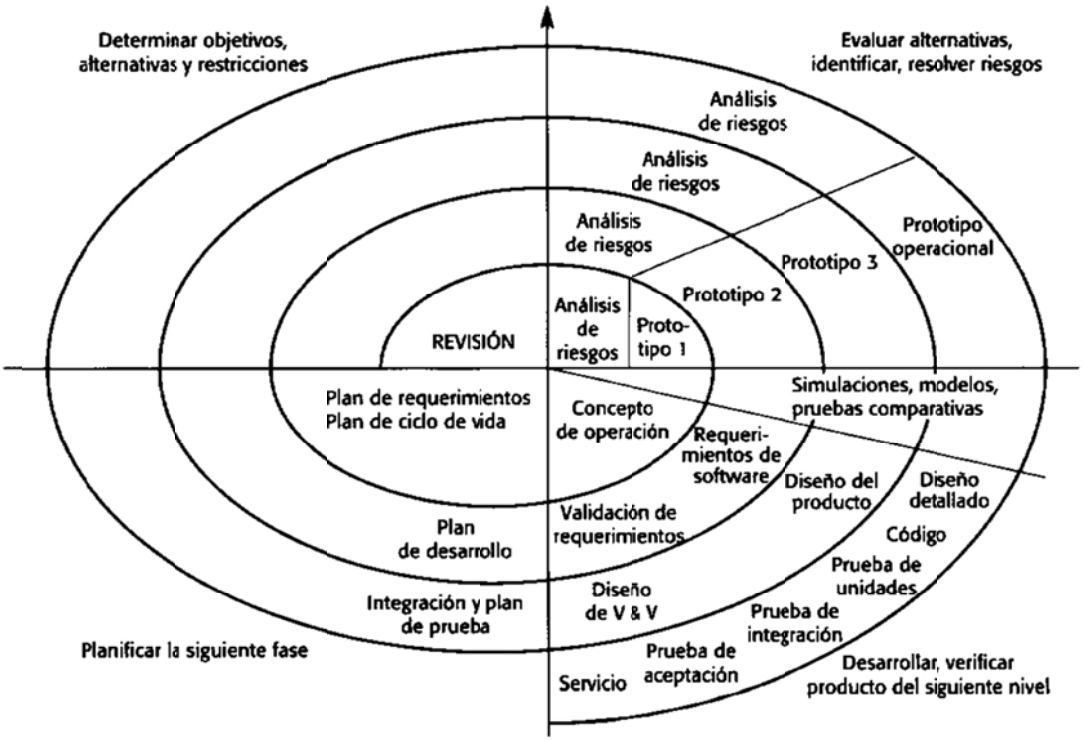

FIGURA 51. Modelo en espiral de Boehm para el proceso de software.

4.7.3.3 Pruebas finales.

La prueba de aceptación algunas veces se denomina prueba alfa. Los sistemas personalizados se desarrollan para un único cliente. El proceso de prueba alfa continúa hasta que el desarrollador del sistema y el cliente acuerdan que el sistema que se va a entregar es una implementación aceptable de los requerimientos del sistema.

Cuando un sistema se va a comercializar como un producto de software, a menudo se utiliza un proceso de prueba denominado prueba beta. Ésta comprende la entrega de un sistema a un número potencial de clientes que acuerdan utilizarlo, los cuales informan de los problemas a los desarrolladores del sistema. Esto expone el producto a un uso real y detecta los errores no identificados por los constructores del sistema. Después de esta retroalimentación, el sistema se modifica y se entrega ya sea para una prueba beta adicional o para la venta.

5. Análisis y diseño.

Índice de capítulo.

Este capítulo tratará sobre los requerimientos del sistema y los pasos a seguir para su elaboración. Los requerimientos se presentarán en forma de lista y estarán acompañados de una descripción. Posteriormente se realizará el diseño del sistema y los pasos que deberán seguirse.

El proyecto de esta tesis está basado en un recorrido virtual encargado por ProMéxico[23] a DGSCA[24] para cumplir con una de las condiciones para participar en la Exposición Universal Shanghái 2010. Por esta razón los principales requerimientos fueron hechos por el comité organizador en China y por ProMéxico. Con esto se cumplirá el segundo objetivo de esta tesis. Para cumplir con el tercer objetivo se añadirán otros requerimientos planteados por el Ing. José Larios Delgado y estará dentro de la evolución del recorrido virtual solicitado por ProMéxico. El proyecto fue realizado por un equipo del cual formé parte y la tesis está acotada a una parte de la programación del recorrido virtual, la sección que me asignaron para desarrollar.

Debido a que este tipo de proyectos era nuevo para el equipo que lo desarrolló y para minimizar el riesgo de falla o dejar insatisfecho al cliente se optó por utilizar un modelo de desarrollo evolutivo, utilizando prototipos desechables porque "el objetivo del proceso de desarrollo evolutivo es comprender los requerimientos del cliente para desarrollar una definición mejorada de los requerimientos para el sistema y el prototipo se centra en experimentar con los requerimientos del cliente que no se comprenden del todo[25]".

La principal dificultad al realizar este recorrido se debió a que el grupo de personas de ProMéxico que se encargaron de solicitar y revisar el proyecto no sabían exactamente el resultado final que deseaban, continuamente pidieron cambios y a la tardanza en la entrega de la información que se les solicitó.

El modelo 3D del pabellón estuvo sujeto a muchos cambios y mezcla de versiones de los objetos realizadas por los miembros del equipo del cual fui parte, resultando difícil decir cuál modelo fue realizado por cada uno, por esta razón pienso que el crédito de la creación del modelo pertenece al equipo[26] y solicito que no se tome esta parte para evaluación mía, solamente la parte de la programación del recorrido virtual[27]. Deseo que esta tesis sirva también como fuente de consulta y añadiré las principales técnicas en la elaboración de los modelos del pabellón.

[23] ProMéxico es el Organismo del Gobierno Federal encargado de coordinar las estrategias dirigidas al fortalecimiento de la participación de México en la economía internacional.
[24] Actualmente Dirección General de Cómputo y de Tecnologías de la Información y Comunicación de la UNAM.
[25] Ingeniería del software, Sommerville, Ian. 7ª Ed. Edit. Addison Wesley.
[26] El equipo que realizó los modelos estuvo formado por Emmanuel Rojón González, Daniel Lorenzo, Emilio Quiroz Galván, José Felipe de Jesús Contreras Flores y Víctor Hugo Franco.
[27] Sólo las funciones del recorrido que programé están contenidas en esta tesis, la versión que se entregó a ProMéxico contiene, además, otras que fueron programadas por César Ordoñez Rodríguez, integrante del equipo de desarrollo.

Para la evolución del recorrido utilicé un modelo del edificio A de la facultad de Ingeniería y modifiqué la programación del recorrido del pabellón para satisfacer las nuevas especificaciones. El desarrollo de esta parte es enteramente mío.

Los prototipos fueron entregados para revisión a los directores de proyecto de DGSCA, a los funcionarios de ProMéxico y después al Ing. José Francisco Salgado Rodríguez[28] y al Ing. José Larios Delgado[29]. Cada revisión produjo correcciones o incrementos en los requerimientos o en los elementos del recorrido.

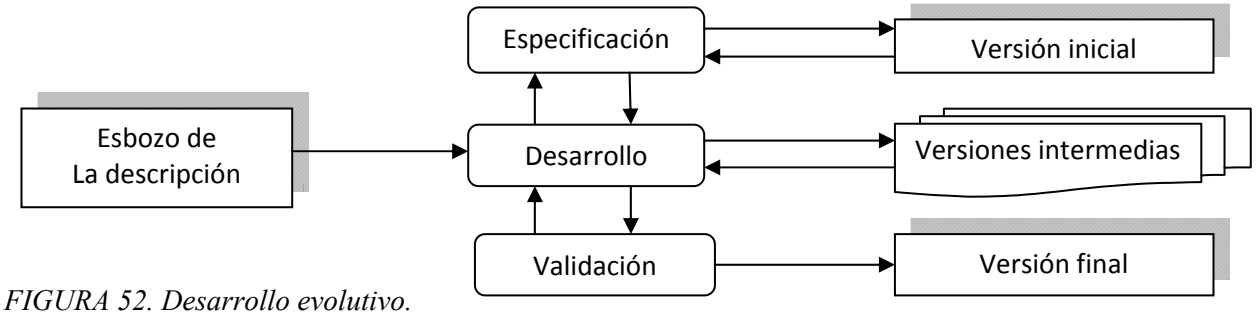

FIGURA 52. Desarrollo evolutivo.

5.1 Definición de requerimientos del sistema.

- ✓ Debe ser una representación del pabellón mexicano construido en Shanghái, China para la Expo universal 2010. El proyecto requiere hacer un modelo tridimensional lo más fiel posible al pabellón construido en Shanghái, China. Se incluirán las áreas de su interior, las piezas museográficas y el área exterior.

- ✓ Debe ser accesible para cualquier persona que cuente con internet. Para llegar a una gran cantidad de personas en todo el mundo se deberá poner en un servidor de internet y estar disponible en cualquier momento. Los usuarios no deberán comprar software adicional para poder verlo y si se necesita adquirir alguno debe ser de fácil acceso y gratis. El programa que servirá para la interacción será Virtools.

- ✓ Debe cargar rápido. La carga del programa debe ser rápida para evitar que el usuario pierda el interés y deje de verlo por esta causa.

- ✓ Debe tardar poco en iniciar movimiento o interacción con el usuario. El usuario no debe esperar mucho tiempo para comenzar a interactuar con el programa, esto provoca que se pierda el interés y se deje de verlo.

[28] Primer director de esta tesis.
[29] Segundo director de esta tesis.

- ✓ Debe ser navegable a gusto del usuario. Para dar una sensación de estar presente en el pabellón físico se debe permitir que el usuario lo recorra a su gusto.

- ✓ Debe mostrar información acerca de algunos elementos del pabellón. Se debe mostrar información sobre algunos elementos como las piezas museográficas para ampliar información.

- ✓ Controles simples. Los controles deben ser simples e intuitivos.

- ✓ Debe ser capaz de mostrar información social, científica, histórica y cultural del país.

- ✓ Debe centrarse en el tema de la Expo “Mejor ciudad, mejor vida” y reflejar los contenidos de la exposición en el pabellón físico[30].

- ✓ El contenido y la forma del recorrido debe basarse en la nacionalidad, edad, nivel educativo y hábitos de los usuarios que lo usarán. Debe ser de fácil comprensión y uso.

- ✓ Debe ser interesante, innovador y capaz de enseñar sobre los temas en él expuestos.

- ✓ Los archivos que utilice deben tener un tamaño y calidad adecuados para asegurar la transferencia de datos a través de internet.

- ✓ No se deben mostrar marcas comerciales.

- ✓ Requerimientos para poder ejecutarlo: CPU Pentium IV 800 MHz o superior, 1 MB en memoria RAM, adaptador de video integrado con soporte para DirectX8.

- ✓ Ancho de banda: 256 Kbps o superior.

- ✓ Resolución de la ventana: 800x600 pixeles.

- ✓ Navegador de internet: Internet explorer 6.0 o superior, Mozilla Firefox 2.0 o superior.

- ✓ Debe realizarse en 3D via virtools 4.1.0.64 o superior y utilizar el plug-in 3D via player 5.0.0.10 superior.

[30] Guide for development and construction of the experiencing pavilion. Bureau of Shanghai world expo coordination. 3a ed.

- ✓ Tipos de archivos permitidos: jpg, png (imágenes), mp3, wma (audio), wmv (video), html, vmo (publicación).

- ✓ No se deben usar Building Blocks[31] creados con el SDK[32] de virtools.

- ✓ Se debe modificar la programación del recorrido para que se puedan cargar otros modelos[33].

- ✓ Debe mantener las principales características del recorrido del pabellón (navegación de la cámara, colisiones, rapidez en la carga, mostrar fichas informativas).

5.2 Análisis y diseño del software.

Para la elaboración del programa se utilizará un modelo de desarrollo evolutivo con publicación de prototipos. Cada prototipo pasará por una etapa de especificación, de desarrollo y de validación hasta llegar a la versión final. En este subtema se dará una explicación sobre los análisis y diseños hechos a través de los diferentes prototipos dentro de la etapa de especificación.

El programa para realizar los modelos tridimensionales fue 3D Studio Max porque ya existía conocimiento previo de este programa por parte del equipo desarrollador y porque existía un exportador de los modelos hacia virtools. Otros programas de modelado que se analizaron fueron blender y maya pero por el nivel de conocimiento sobre 3D Studio max se eligió éste. El motor de juegos ocupado fue virtools porque así lo exigieron los requerimientos del cliente. Para la edición de imágenes se analizaron los programas photoshop y gimp y se optó por usar photoshop por el conocimiento que ya tenía el equipo de desarrollo en el manejo de este programa.

Análisis y diseño general para la elaboración de los prototipos.

El modelo del pabellón consta de una construcción arquitectónica de tipo galerón con trece áreas (área exterior, techo, entrada, salida, promoción turística, tienda, restaurante, ciudad y población, ciudad y patrimonio, ciudades vibrantes, ciudad y naturaleza y centro de negocios). Primero se elaborará el pabellón con las paredes, techos y pisos y los objetos que irán en cada área se agregarán en posteriores prototipos.

Lo siguiente será agregar texturas a los modelos y poner los objetos que servirán como colisión. Los objetos de colisión deberán tener una geometría sencilla y carecerán de texturas pues permanecerán ocultos.

[31] Funciones de la biblioteca de virtools.
[32] Herramienta para la programación de funciones de virtools.
[33] A partir de este punto, los requerimientos serán para cumplir con el tercer objetivo de la tesis.

El modelo se exportará a virtools y se elaborarán las funciones necesarias para que una cámara pueda navegar por el pabellón y que responda correctamente a la detección de colisiones con las paredes, objetos y pisos. La cámara se posicionará a una altura que resulte parecida a la realidad.

La interacción sólo será con algunos objetos porque no todos poseen información atractiva (por ejemplo las paredes, pisos o ventanas). Los objetos se elegirán por su contenido histórico, cultural o que necesiten ampliar información y para ubicarlos tendrán un modelo en forma de mano con el dedo índice apuntando hacia el objeto (a manera de puntero de mouse). Al dar un click sobre el objeto desplegará una ficha con información. Otra interacción será cuando la cámara se acerque a una puerta de entrada o salida y ésta responda abriéndose para permitir el paso.

Ahora se mostrarán el análisis y diseño para los modelos y funciones del recorrido.

Modelado.

Debido a la naturaleza de la información proporcionada por ProMéxico y a las características físicas de los objetos el modelado se realizó con dos técnicas principales.

La primera tuvo como información planos arquitectónicos con medidas y descripción de materiales, por ejemplo las paredes, los techos, puertas y mobiliario del restaurante.

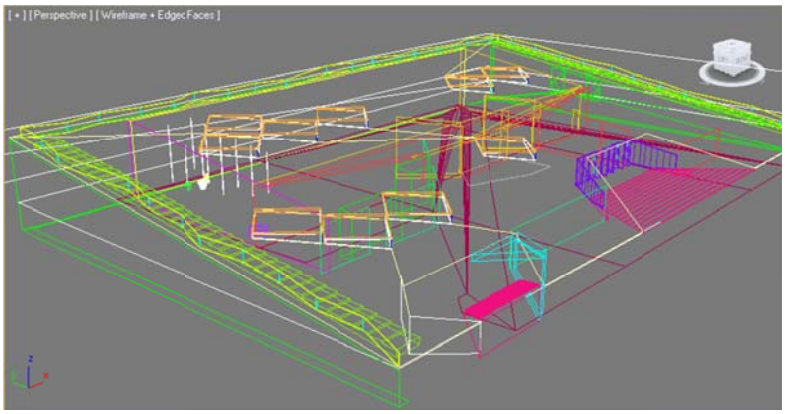

FIGURA 53. Ejemplo de modelado con primitivas gráficas en el pabellón. Paredes y escaleras principales del pabellón.

Para estos objetos se utilizaron las medidas de los planos y se construyeron los objetos a partir primitivas gráficas. Algunos planos fueron enviados en un archivo de AutoCAD y 3D studio max permitió usar las vistas laterales, planta y frontal como plantillas para el levantamiento del modelo.

La segunda técnica tuvo como información recibida imágenes de los modelos, descripciones y medidas generales (largo, ancho y profundidad). Para estos objetos se utilizará el modelado basado en imagen.

Para modelar estos objetos las imágenes se debieron modificar para obtener una vista ortogonal y utilizar elementos aledaños como referencia de dimensiones.

Debido a que algunas de las pruebas arrojaron el dato de que el número de polígonos es directamente proporcional al tamaño del archivo se adoptó la política de que los objetos menos importantes o lejanos a la cámara tendrían menos polígonos. Otra forma para reducir el número de polígonos de algunos objetos fue cambiar los detalles en geometría por texturas que los simularan. Los polígonos que no se verán serán borrados para reducir el número de éstos. Por ejemplo, el polígono de un pedestal que se encuentre pegado al piso no se observará y puede borrarse.

FIGURA 54. Ejemplo de objeto hecho con modelado basado en imagen. Esculturas de Paloma Torres.

Iluminación y texturizado.

Cuando los modelos estén terminados se procederá a agregar el material que requieran. Se respetará el material que los organizadores hayan puesto en las especificaciones del objeto cuando la información así lo presente o el más parecido si la información recibida fuese una imagen. Las texturas para las piezas museográficas deberán ser idénticas debido al valor cultural contenido en ellas.

Se preferirán las texturas procedurales y si se requiere se crearán las texturas que no tenga el programa de modelado.

Para algunos objetos se deberá utilizar la técnica del mapeo de textura con coordenadas uvw y aplicarles como textura una imagen de la pieza real como en el caso de las piezas museográficas.

Posteriormente se integrarán las luces necesarias para que el modelo tenga una iluminación agradable y que los detalles de los modelos puedan apreciarse correctamente.

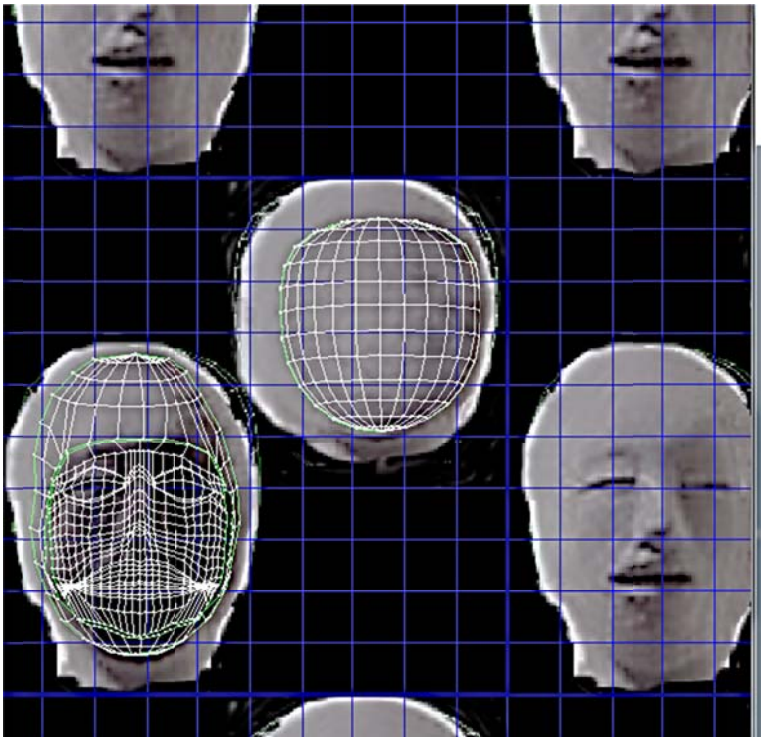

FIGURA 55. Ejemplo de la técnica de mapeo de textura para el modelo de la escultura "La aburrida"

Exportación al motor de juegos.

Las pruebas realizadas mostraron que el exportador hacia virtools no soporta las texturas procedimentales por lo tanto se deberá utilizar la técnica de horneado de texturas, esto permitirá que el objeto mantenga las características de la textura original y el efecto de iluminación al tiempo que solo se necesitará una luz en virtools.

Otros datos obtenidos de las pruebas mostraron que mientras más texturas se tengan el tamaño del archivo se incrementa, para solucionar esto se deberá programar una rutina que cargue las texturas para cada objeto después de haber cargado el recorrido sin texturas. Con esto se acelerará la carga inicial.

Otras pruebas realizadas mostraron que los objetos cuyas texturas usan transparencia el canal alfa no puede cargar la textura necesaria, para solucionar esto únicamente estos objetos serán exportados con textura.

Las texturas se almacenarán en una carpeta dedicada solamente a éstas y estará al mismo nivel que el archivo html que ejecute el recorrido. El formato de las texturas será jpg con un tratamiento de optimización en tamaño para internet para que ocupe menos espacio y sea

más rápida la carga. Esto agregará la facilidad de modificar las texturas sin la necesidad de modificar ni volver a publicar el recorrido.

Navegación.

Se incluirá una cámara para que el usuario pueda hacer el recorrido. La cámara contará con una altura lo más parecida a las dimensiones de una persona adulta. Datos de las pruebas hechas indicaron que la parte exterior necesita una altura mayor y que presenta problemas en el interior, por esto se tendrá que incluir una rutina que cambie de altura al entrar al pabellón.

La cámara tendrá ocho movimientos, cuatro desplazamientos y cuatro giros. Pruebas hechas a varias personas, entre quienes se encontraban jugadores de videojuegos de computadora, de consolas, adultos, niños, personas con uso cotidiano de la computadora y usuarios esporádicos dieron por resultado que la forma más intuitiva fue relacionar los movimientos de la cámara con teclas de la siguiente manera:

Flecha arriba: desplazamiento hacia adelante.
Flecha abajo: desplazamiento hacia atrás.
Flecha derecha: giro a la derecha.
Flecha izquierda: giro a la izquierda.
Tecla w: giro hacia arriba.
Tecla s: giro hacia abajo.
Tecla a: desplazamiento hacia la derecha.
Tecla d: desplazamiento hacia la izquierda.
Mouse: interacción con los objetos.

FIGURA 56. Teclas para dirigir el movimiento de la cámara.

Colisiones.

La cámara tendrá colisiones con las paredes y objetos de la escena para que no los atraviese y que el movimiento parezca más natural. Las pruebas mostraron que en la parte externa la cámara atravesaba objetos y en la parte interna no podía pasar por ciertas partes,

para solucionarlo se añadirá una función que aumente el radio de colisión en el exterior y lo reduzca en el interior.

Carga de escenas.

Resultados de las pruebas mostraron que mientras más objetos estén en el campo de visión de la cámara más tardará en realizar el render y se necesitará más capacidad de procesamiento. Esto ocurre a pesar de que algunos objetos estén detrás de otros y parezca que no se muestran. Para solucionar lo anterior se utilizará una rutina para ocultar los objetos que salgan de la vista de la cámara, para esto se dividirá el pabellón en regiones o escenas que serán cargadas al acercarse la cámara.

Interacción y muestra de información.

Cuando un objeto pueda desplegar información tendrá un modelo de una mano que lo señale y al dar *click* en él con el *mouse* mostrará una ficha con información relacionada al objeto o a un tema afín. Cuando la cámara se acerque a una puerta de entrada o salida se ejecutará una animación de éstas para abrirse y permitir el paso.

FIGURA 57. Ejemplo de una ficha informativa del modelo de un cuadro de Frida Kahlo.

Publicar.

Cuando se tenga todo el prototipo del recorrido listo, y después de realizar una prueba en el modo de desarrollo, se exportará, resultando un archivo vmo[34] que contendrá el recorrido y un archivo html que contendrá el visualizador del archivo vmo[35] y funciones de verificación del tipo de navegador de internet que se utilice, si está instalado el programa visualizador y la

[34] Tipo de archivo ejecutable de virtools.
[35] El visualizador es el 3D via player.

versión del visualizador instalada. Si no se encuentra instalado el visualizador o si existiera alguna versión más nueva mandará el aviso y pondrá un vínculo para descargarlo gratis desde la página de Dassault systèmes[36].

Las pruebas realizadas mostraron que el tiempo de carga del recorrido se incrementa con el tamaño y que éste se incrementa a mayor número de polígonos y texturas embebidas en el archivo vmo. Para solucionar esto se necesitó crear la rutina de carga de texturas externas permitiendo que la carga fuera más rápido pero resultó que el pabellón carga con los objetos en tonos grises. De este análisis se desprende la política siguiente para el orden de carga de texturas:

a) Los objetos que estén más cercanos a la cámara deberán cargar primero sus texturas.
b) Los objetos que se noten más deberán cargar sus texturas primero[37].
c) Los objetos que se encuentren en la ruta más probable del usuario deberán cargarse primero.

Análisis y diseño para modificar el recorrido para que pueda cargar cualquier modelo.

Después de realizar algunas pruebas determiné que la forma más sencilla para configurar el recorrido y que pudiera cargar modelos sin restringir demasiado su formato sería usar archivos de texto. Los archivos de texto deberán contener datos en forma de lista para que el recorrido los utilice.

Modelado, iluminación y texturizado.

Los modelos deberán tener la menor cantidad de polígonos posibles y deberán tener texturas horneadas. Debido a que no se tendrá control sobre el tipo de modelos que se utilizarán se recomienda que el conjunto no exceda de 100 000 polígonos.

Exportación al motor de juegos.

Cada objeto se exportará sin texturas, sólo con el nombre del archivo de textura. Los objetos que utilicen el canal alfa se exportarán con texturas.

Los objetos estarán en archivos nmo en una carpeta y las texturas en formato jpg en otra carpeta. Los objetos de colisión se exportarán aparte y sin texturas.

Se añadirá una rutina para cargar los objetos, una para las texturas y una para los objetos de colisión que también los ocultará después de cargarlos. Se necesitará un archivo de texto con los nombres de los archivos nmo que contendrán los objetos, uno con la lista de texturas y uno con la lista nombres de los objetos de colisión.

[36] Compañía propietaria de virtools. http://www.3ds.com/
[37] Como en el caso del piso o del cielo.

Navegación.

El recorrido solamente contará con una cámara y una luz. Cuando se cargue algún modelo se utilizará un archivo con datos para configurar la ubicación inicial de la cámara y su altura. El programa ubicará la cámara dentro de un sistema cartesiano con origen en el centro del primer objeto cargado. A partir de este origen se referirán las coordenadas que ubicarán la cámara. Este archivo también tendrá la altura a la cual se colocará la cámara. Estos dos parámetros se podrán cambiar en el archivo de texto. La cámara tendrá los mismos ocho movimientos que el recorrido del pabellón.

Flecha arriba: desplazamiento hacia adelante.
Flecha abajo: desplazamiento hacia atrás.
Flecha derecha: giro a la derecha.
Flecha izquierda: giro a la izquierda.
Tecla w: giro hacia arriba.
Tecla s: giro hacia abajo.
Tecla a: desplazamiento hacia la derecha.
Tecla d: desplazamiento hacia la izquierda.
Mouse: interacción con los objetos.

Colisiones.

La cámara tendrá colisiones con las paredes y objetos y se mantendrá a altura constante con respecto a los pisos. Los objetos de colisión y pisos deberán estar en una lista en un archivo de texto para declararse. Esta misma lista se utilizará para ocultarlos. También el radio de colisión entre la cámara y los objetos podrá cambiarse para que pueda pasar por puertas o pasillos.

Interacción y muestra de información.

Debido a que no se controlarán las características de los modelos la carga de escenas no será incluida en esta modificación. Tampoco la animación de puertas se incluye en esta modificación.

Para el despliegue de fichas de información se utilizará un archivo de texto con los objetos que contendrán esta característica. Esta lista se utilizará para asignarles una “manita”[38] señalándolos. Debido a que las fichas informativas en el pabellón son imágenes y que el número de éstas no será controlado en la nueva modificación se hizo un cambio para presentarlas. Al dar click un objeto con información se abrirá una página html donde se encontrará la información deseada. Esto dará más libertad para el manejo de estas fichas.

[38] Modelo en forma de puntero de mouse tipo mano con el índice señalando.

Publicar.

Cuando esté listo este prototipo se realizará una prueba en modo de desarrollo, posteriormente se exportará y se hará otra prueba.

6. Desarrollo y pruebas

Índice de capítulo.

En este capítulo se mostrará el proceso de creación de este recorrido virtual siguiendo la línea que se planteó en el capítulo 5.

FIGURA 58. Imágenes del pabellón real y virtual.

El desarrollo se realizó siguiendo un modelo de desarrollo evolutivo con prototipos desechables explicado en el capítulo anterior (capítulo 5), con los procesos: modelado, iluminación, texturizado, exportación, movimiento de la cámara, declaración de colisiones, carga dinámica de texturas, interacción, optimización y publicación.

Las pruebas se realizaron cuando alguna función o proceso estuvieran terminados, por ejemplo se probó al terminar la carga de texturas o el control de avance de la cámara. Otras pruebas se realizaron al concluir alguno de los prototipos.

A continuación se detallarán cada uno de los procesos seguidos en cada prototipo. Dado que en etapas de desarrollo similares se usaron técnicas similares la explicación se agrupará en una sola.

6.1 Desarrollo de los modelos 3D.

Para el modelado y texturizado de objetos se utilizó el programa 3D Studio max versión 2010.

6.1.1 Modelado.

Modelos a partir de planos arquitectónicos. Modelado a partir de primitivas gráficas.

Para realizar el levantamiento del pabellón se utilizaron planos arquitectónicos que los arquitectos que construyeron el pabellón físico proporcionaron.

Dichos planos se utilizaron como plantilla, teniendo las vistas laterales, de planta y frontal, así como las medidas en ellos marcadas. Por ejemplo para las paredes se creó un prisma rectangular con las medidas marcadas en los planos proporcionados. Para el caso de las paredes laterales se modificaron los vértices de una de las aristas superiores para reducir su altura y que terminara con una pendiente, pues el modelo tiene techo “de un agua”. Otros objetos fueron hechos con cilindros como el caso de los sillones siguiendo la misma técnica.

FIGURA 59. Imagen del modelo terminado y fotografía de una escultura de Paloma Torres con una persona como referencia para las dimensiones.

Modelos a partir de fotografías. Modelado basado en imágenes.

Para elaborar otros objetos sólo se contó con fotografías como en el caso de las piezas museográficas y objetos de la tienda. Para realizarlos, las fotografías fueron editadas con Photoshop para lograr una proyección ortogonal y con ella usarla como plantilla para construir el objeto a partir de primitivas gráficas.

Con la escultura de “la aburrida” se utilizó un prisma para el cuerpo y un cubo para la cabeza. Se fue dividiendo la malla del objeto y se extrudieron los polígonos necesarios hasta conseguir la forma de la escultura. Se manipularon los vértices para hacerlos coincidir con la imagen base y por último se juntó la cabeza con el cuerpo. Algunos objetos no necesitaron construirse con primitivas separadas.

El proceso para hacer las esculturas de Paloma Torres tuvo una variante. Como esta escultura estaba en proceso de creación no se tenían las dimensiones para poder realizar el modelo, para resolver este problema a las imágenes se trazaron líneas para obtener una altura aproximada utilizando la altura conocida de una persona que aparecía junto a la escultura.

La prueba para este proceso (modelado) fue verificar el número de polígonos y reducirlos hasta que la figura tuviera un terminado aceptable, es decir encontrar una relación favorable entre número de polígonos y curvas suaves en la superficie del objeto.

Para cada objeto se realizó una prueba de número de polígonos con una herramienta de conteo. Si el objeto presentaba demasiados polígonos se procedió a reducirlos juntando vértices y aristas. El criterio para decidir si el número de polígonos era alto dependió del objeto, el detalle que debía presentar y la importancia del objeto dentro del pabellón.

Los polígonos que no debían verse, como los que quedaban cerca del piso, se eliminaron.

Algunos objetos se realizaron con más de una primitiva y al final se juntaron para formar un solo objeto, eliminando caras que quedaban entre una y otra y que no se mostrarían.

La herramienta de conteo que se usó fue el *polygon counter* que tiene el programa 3DS max.

El conteo final de polígonos del modelo del pabellón muestra que contiene 172,657 polígonos. Conviene señalar el contraste de números de polígonos entre objetos con diferente importancia para su detalle y forma. El número de polígonos de uno de los sillones es de 211 mientras que el de la escultura “La aburrida” es de 1445.

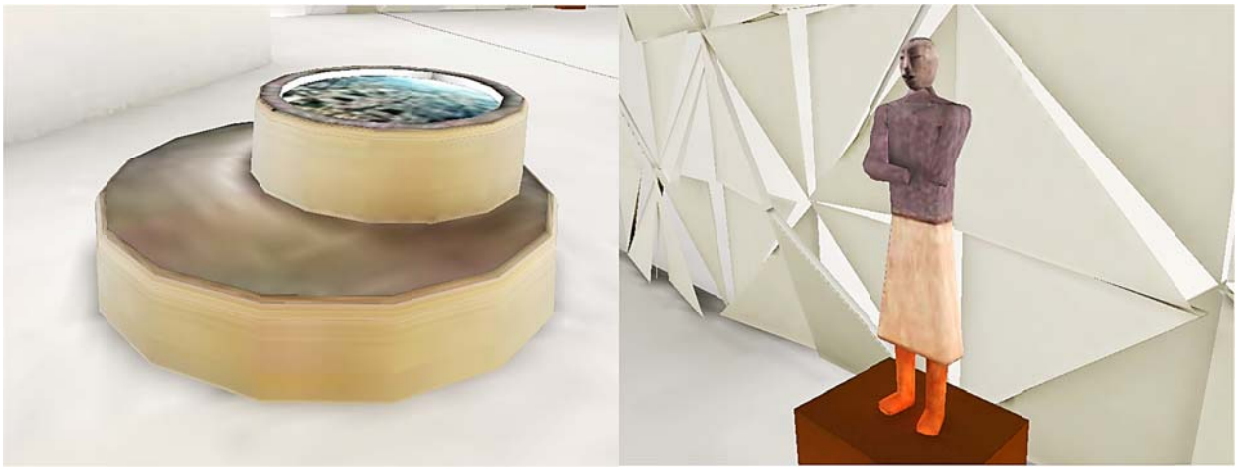

FIGURA 60. Objetos con número de polígonos contrastante.

6.1.2 Objetos para colisión.

Los objetos para colisión se realizaron con prismas envolventes de las zonas u objetos que la cámara no debía traspasar. Estos objetos tienen muy pocos polígonos y no cuentan con textura ya que serán ocultos a la vista. La geometría es un prisma con pocos lados como se muestra en la siguiente figura.

6.1.3 Iluminación.

Para la iluminación se utilizó una luz principal que simulara el sol y se agregaron otras para corregir zonas obscuras y lograr una iluminación natural en la escena.

Para los efectos de luz se utilizaron una luz direccional de tipo “sol” (*sunlight*) para la iluminación general y que genera las sombras, 22 luces direccionales tipo reflector (*spot*) dirigidas hacia las piezas museográficas y 21 luces puntuales tipo omnidireccionales para corregir las zonas obscuras que quedaran.

FIGURA 61. Los objetos de colisión se muestran en azul en la imagen derecha, son envolventes de los objetos que no debe atravesar la cámara.La escultura del “Cactus” (últimas dos imágenes) tiene 778 polígonos mientras que su objeto de colisión tiene 6.

La luz de tipo sol es un sistema de luz direccional que simula la luz del sol en cualquier punto geográfico de la tierra, día del año y hora del día.

Los efectos de sombras se generaron con los parámetros de la luz tipo sol para dar mayor profundidad. Estos efectos de luz y sombras se exportarán posteriormente juntos con las texturas mediante el proceso de “texturas cocinadas”.

El recorrido contiene cuatro luces puntuales omnidireccionales que dan una iluminación general.

6.1.4 Texturas.

Las texturas que se utilizaron dependieron del objeto al que se aplicaran, algunos tuvieron texturas procedimentales como el caso del piso, techo y paredes, otras fueron hechas con patrones repetidos como el piso de pasto en el exterior y algunos requirieron mapas usando

coordenadas UV como la escultura “la aburrida”. A los objetos que requerían relieves se les agregó una imagen[39] para reducir el número de polígonos y que el efecto deseado.

Después de la iluminación se empleó el horneado de texturas exportando todas las texturas a imágenes y utilizando un mapeo de textura UV. Esto se hizo para tener un efecto óptico de luces, sombras y relieves sin necesidad de que el motor de render hiciera estos cálculos y también para optimizar la carga del recorrido.

Se utilizaron varios tipos de texturas para los objetos del pabellón, cada tipo respondió a las necesidades visuales de los objetos en particular. Todas las texturas empleadas en este recorrido virtual son imágenes mapeadas con las coordenadas de los objetos. Las imágenes contienen los efectos de luz y sombras realizados con 3DS max cuando se hizo el modelado. Los materiales empleados antes de realizar el horneado de texturas fueron con repetición de patrones como el pasto del techo del pabellón, el piso exterior; mapeo de texturas como en la aburrida, el cactus, las máscaras, los tótem; materiales con múltiples submateriales como algunas paredes, el Friso El osario; materiales procedurales como en la base de la aburrida o lis pisos interiores; materiales translúcidos como en los vidrios y pantallas de las infosalas. Algunos materiales tuvieron incluido un mapeado de relieve, por ejemplo las piezas museográficas, los objetos de la tienda, algunos muros del restaurante.

6.1.5 Exportación de modelos.

Se utilizó un exportador para exportar los objetos de 3D studio max al motor de juegos (virtools). En la exportación cada objeto fue exportado con su malla, material y nombre de archivo de textura, ya que ésta última se agregaría de forma independiente. Sólo los objetos que necesitaron transparencia (canal alfa) fueron exportados con todo y textura pues las pruebas mostraron que no se pueden agregar los mapas de transparencia de forma externa. Las luces no fueron exportadas.

6.2 Desarrollo de la programación.

El motor de juegos empleado fue 3D vía Virtools versión 5[40].

Virtools es un motor de juegos que permite programar las funciones necesarias (*scripts*) uniendo otras funciones (*Building Blocks*) de su biblioteca, con líneas para activarlas entre ellas. De igual manera los parámetros a las funciones se pueden pasar mediante líneas, lo cual facilita la programación. Se empleó este motor de juegos debido a que pertenece a uno de los requerimientos del proyecto.

[39] Bumping map.
[40] http://www.3ds.com/products/3dvia/3dvia-virtools/

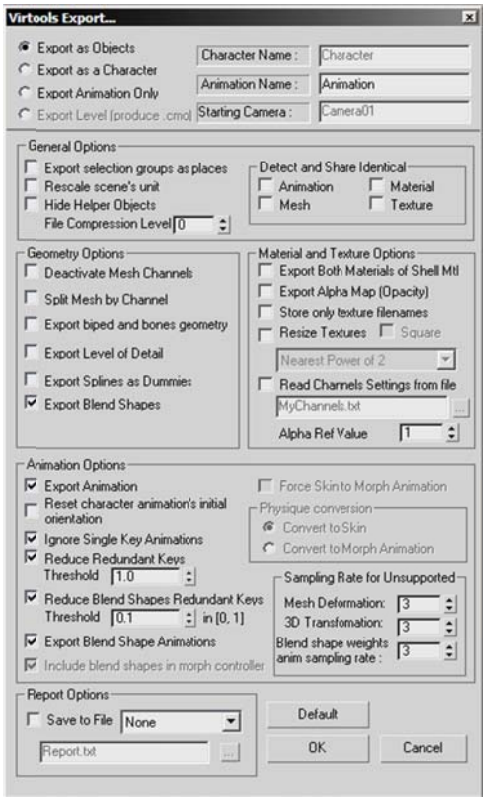

FIGURA 62. Ventana de exportación de modelos de 3DS max a virtools.

6.2.1 Programación de la cámara para la navegación.

Primero establecí las condiciones iniciales de ubicación y orientación para la cámara. La cámara tiene asociado dos *dummies*[41] uno para controlar el desplazamiento y otro para controlar la dirección. Estos elementos están anidados en jerarquía (dummy de traslación > dummy de rotación > cámara) y en conjunto forman el sistema de navegación.

La primera función fue para la translación. Cuando se presionan las teclas “flecha arriba”, “flecha abajo”, A o D se mueve hacia adelante, atrás, izquierda y derecha respectivamente. Un valor específico, que dará la magnitud del movimiento, se multiplicará por un vector unitario que dará la dirección del movimiento según la tecla presionada. Cada segundo se realizará esta multiplicación mientras la tecla se mantenga presionada. Después la ubicación del *dummy* pasará a la cámara. Se utiliza un sistema de referencia local con el origen en el centro del *dummy*.

La segunda función se realizó para la rotación de la cámara. Esto ocurre cuando se presionan las teclas “flecha derecha”, “flecha izquierda”, W y S, girando hacia la derecha, izquierda, arriba y abajo respectivamente. Se multiplica un valor cada segundo por un vector que dará el eje sobre el cuál girará el *dummy* y después se pasará la orientación a la cámara.

[41] Elementos adimensionales que ayudan a la programación sin afectar las características visuales.

En el giro vertical, si pasa de 90˚ hacia arriba o abajo la cámara regresará a un valor inmediato anterior para evitar un giro completo.

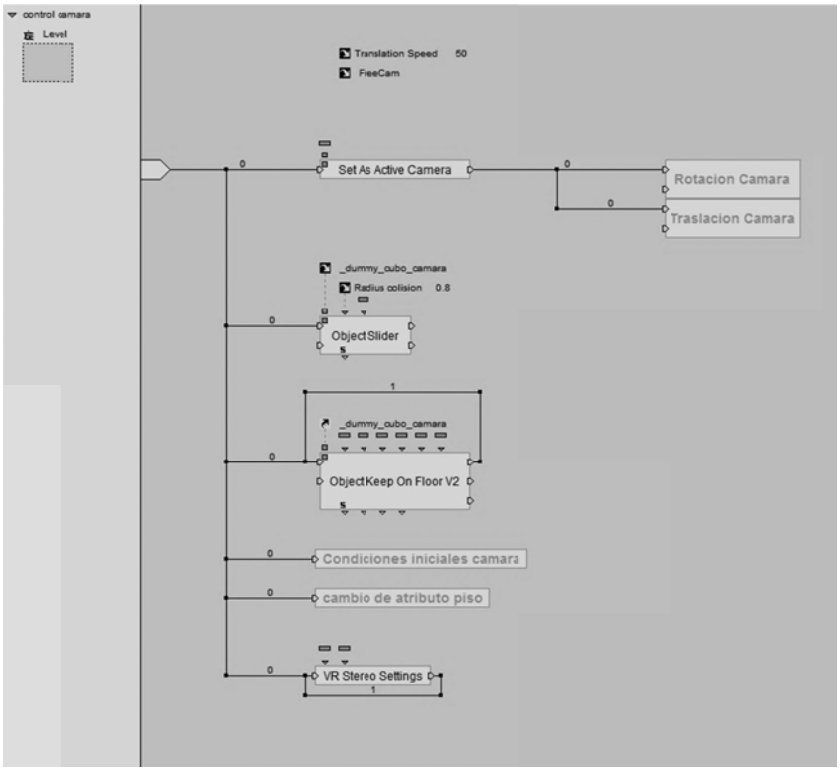

FIGURA 63. Programación de la cámara.

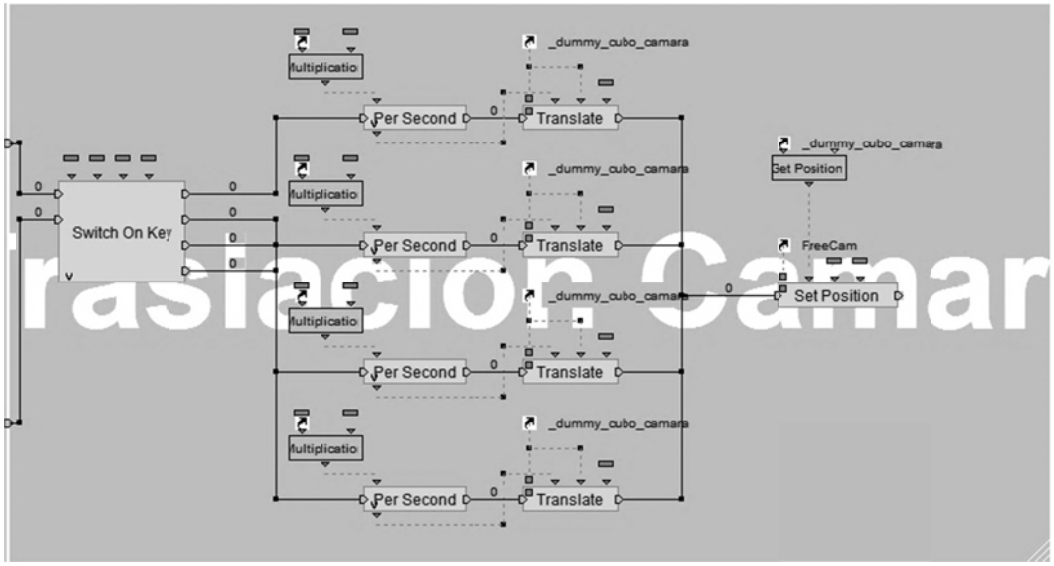

FIGURA 64. Programación de la función de traslación para la cámara.

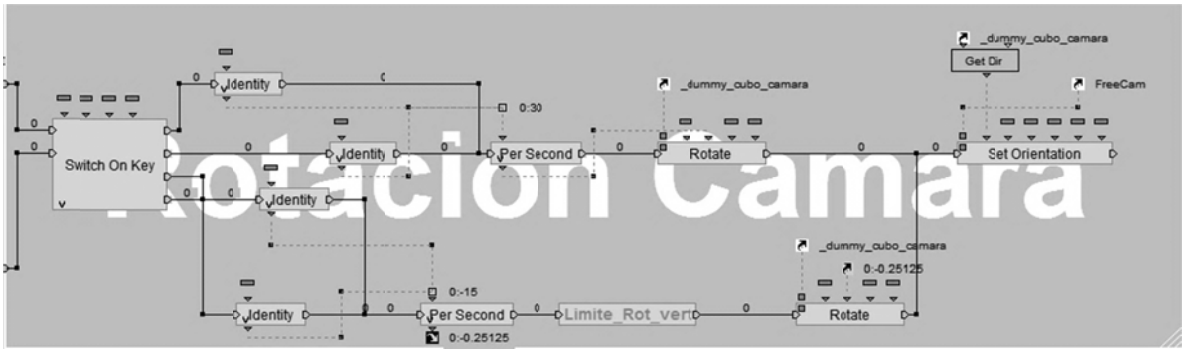

FIGURA 65. Programación de la función para la rotación de la cámara.

6.2.2 Declaración de colisiones.

Para las colisiones, los objetos colisión que fueron hechos con pocos polígonos se agregaron a un grupo y se ocultaron para que no pasaran por el *render*. Se utilizó la función "*object slider*" que permite definir un radio en el cuál un objeto no podrá traspasar otro. Estos objetos son el *dummy* de la cámara y los objetos de colisión.

Para que la cámara permaneciera sobre un objeto de colisión que sirviera de piso se utilizó la función "*object keep on floor V2*" la cual permite que el *dummy* de la cámara permanezca a una altura constante de un objeto con atributo "*floor".* El pabellón cuenta con dos pisos y al utilizar esta función había una confusión entre ellos y posicionaba la cámara en el más cercano, impidiendo en algunos momentos recorrer el pabellón por completo. Para solucionar esto se programó una función que detectara la altura de la cámara en un sistema de referencia global y al disminuir o aumentar de un nivel intermedio se agrega o quita el atributo "*floor"* al piso inferior permitiendo que se recorran tanto el piso superior como inferior. El nivel que permite hacer el cambio se alcanza en las escaleras de entrada y salida.

Adicionalmente se programó una función que cambia el radio de colisión dependiendo del piso en el que se encuentre la cámara, esto solucionó dos problemas: el primero que en el piso inferior existen puertas que necesitan de un radio de colisión pequeño para poder pasar y el segundo que en el exterior se necesita un radio de colisión mayor para que la cámara no atraviese objetos.

6.2.3 Carga dinámica de texturas.

Al exportar el recorrido con texturas se observó que aumentaba el tamaño en memoria del archivo, esto provocaba que tardara más tiempo en descargarse y podía provocar que el observador no quisiera seguir esperando. Para solucionar esto se quitaron las texturas dentro de *virtools* y se programó una rutina para cargarlas después de que empezara el

recorrido. Esto permite que el observador tarde menos en descargar el recorrido y que pueda comenzar a utilizarlo antes que termine de cargar[42].

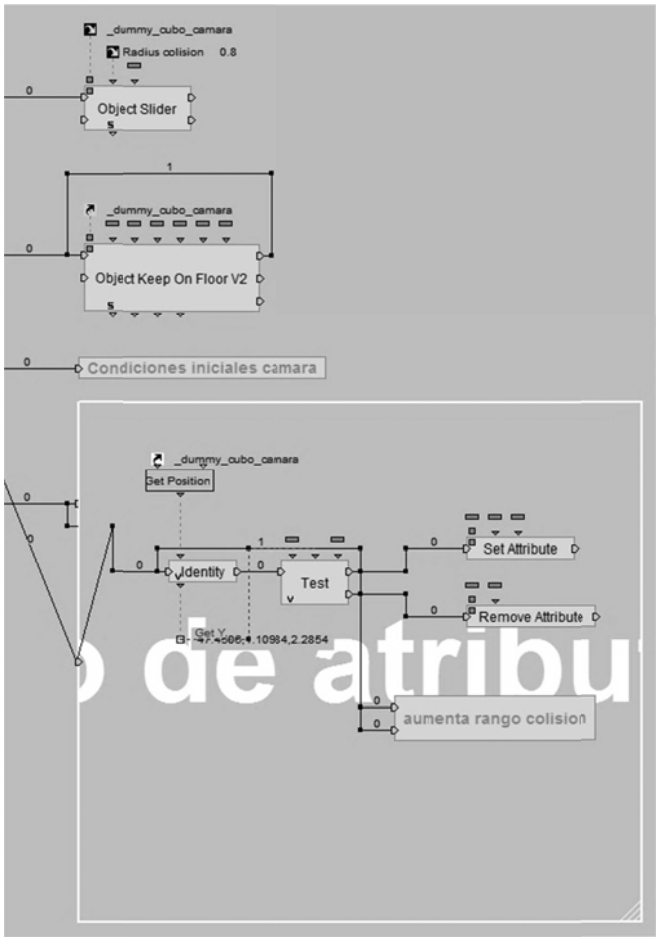

FIGURA 66. Programación de la función para declarar pisos.

La función requiere de una lista (*array*) con los nombres de las texturas para ser descargadas, una variable con el total de texturas que deban ser descargadas y la dirección de la carpeta que contiene los archivos de imágenes con texturas. La lista permite disponer del orden de carga de las texturas; las primeras texturas que se cargan son las de los objetos que aparecen primero ante la cámara.

La función recorrerá cada uno de los renglones de la lista y con el nombre construirá la dirección de donde debe descargar el archivo de textura, si no encuentra la textura pasará al siguiente renglón de la lista, si lo encuentra cargará la textura en el objeto correspondiente.

Adicionalmente tiene una función que muestra una barra que se va llenando mientras se van descargando las texturas y muestra un porcentaje de texturas descargadas.

[42] En promedio 45 segundos con una velocidad de 200 Kbps.

FIGURA 67. Lista con los nombres de las texturas.

6.2.4 Interacción.

Apertura de puertas.

Las puertas tienen una detección de aproximación con la cámara que lanza una animación que abre o cierra la puerta, permitiendo el paso. La detección se hace cuando entra o sale de un umbral y ejecuta una animación en un sentido o en otro, que es cuando abre o cierra la puerta.

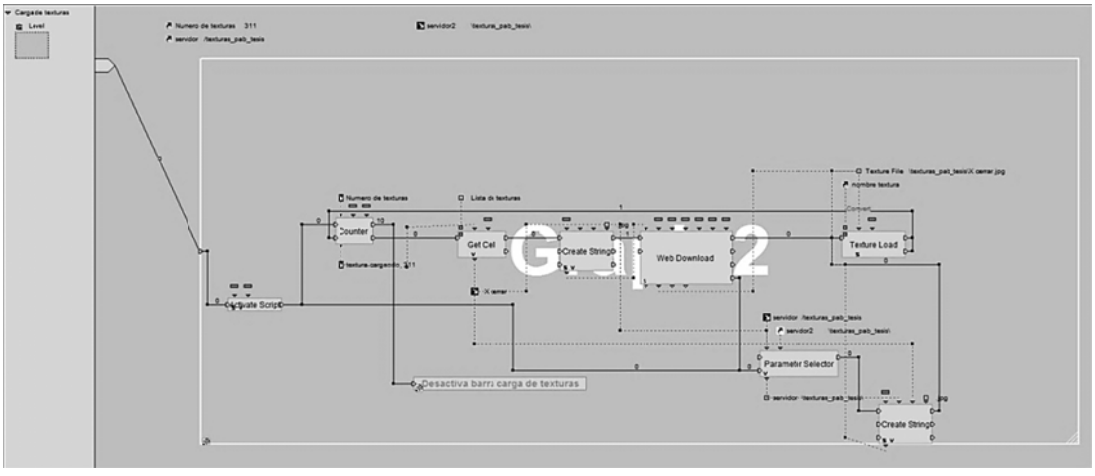

FIGURA 68. Función que carga las texturas.

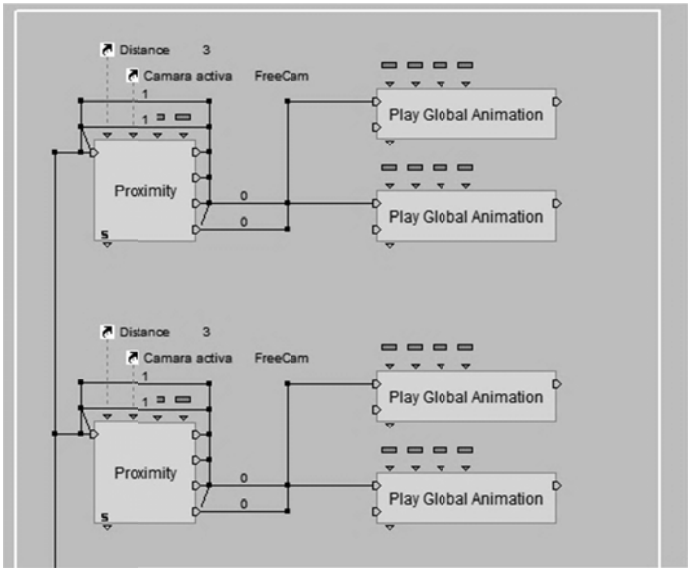

FIGURA 69. Función para abrir y cerrar las puertas.

Muestra de fichas.

Algunos objetos tienen una función que permite que al dar "*click*" con el *mouse* muestre una ficha con información acerca de él. Los objetos que tienen dicha función están señalados con una mano que gira.

La función detecta si se hace "*click*" sobre el objeto, luego detecta si está dentro del grupo de objetos con información y asigna una imagen con la información a la textura de un plano (*frame 2D*) y luego muestra el *frame 2D*. Posteriormente espera un *click* del *mouse* en una imagen de una X para cerrar la imagen.

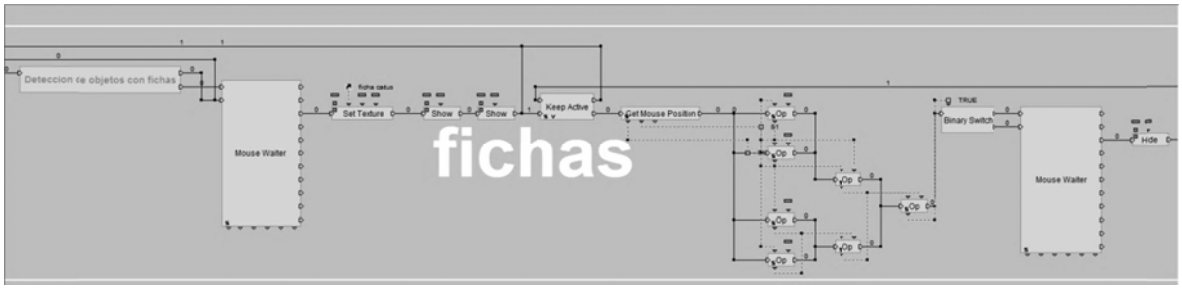

FIGURA 70. Función que muestra la apertura de fichas con información de algunos objetos.

6.2.5 Optimización.

Para reducir el trabajo del procesador al realizar el *render* se utilizaron dos métodos, unos fue hacer una carga por escenas y el otro un algoritmo de nivel de detalle (LOD, *Level Of Detail*) para lograrlo.

Cambio de escenas.

Se dividió el pabellón en cinco zonas y los objetos contenidos en ellas o que se alcanzaran a ver se asignaron a igual número de escenas. Estas escenas permiten realizar el *render* a los objetos contenidos en ellas junto con las funciones programadas.

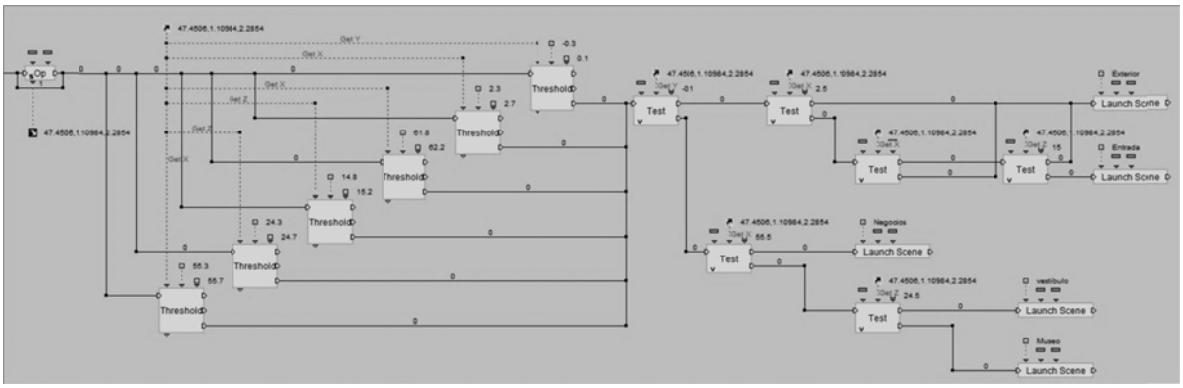

FIGURA 71. Función para activar y desactivar las diferentes escenas.

Para lanzar cada una de las escenas se toma la ubicación de la cámara y si pasa de los límites de las zonas se lanza la escena correspondiente.

Nivel de detalle geometría.

Para el nivel de detalle se utilizó el algoritmo de malla progresiva que va mostrando un porcentaje de polígonos de cada objeto de acuerdo a una proporción del tamaño de éste con el tamaño de la pantalla. Mientras más espacio ocupe el objeto en la pantalla un mayor número de sus polígonos se muestran.

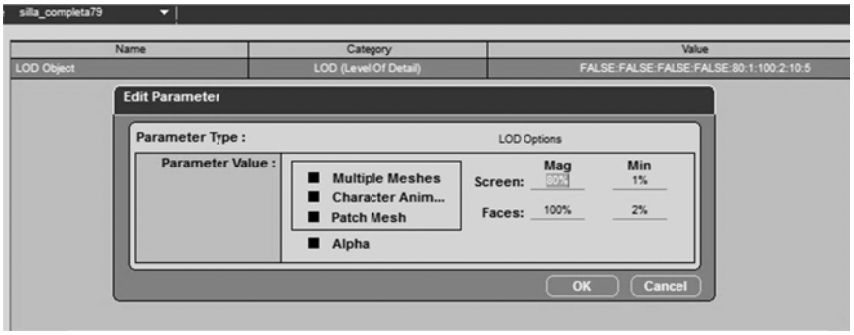

FIGURA 72. Asignación de parámetros para el nivel de detalle de los objetos.

6.2.6 Publicación.

El motor de juegos virtools genera una archivo html y uno vmo cuando se requiere hacer una presentación distribuible y publicable. El archivo vmo es el que contiene todos los datos del recorrido (en el caso de esta tesis) como son los modelos, las texturas, luces, cámaras y

programación. El archivo html funciona como esqueleto para ejecutar el recorrido a través de un *player*[43] gratuito de virtools.

El archivo html comienza con tres funciones, la primera detecta el navegador de internet que se está utilizando, la segunda revisa si se tiene instalado el *player*, si no lo está muestra un enlace para descargarlo y si lo está revisa que sea la versión está actualizada, si no es la más actual manda un aviso. La tercera función ejecuta el *player*.

Además de las funciones anteriores contiene el color de fondo y tamaño de la ventana, en pixeles, del recorrido y el nombre del archivo que contiene los datos del recorrido (vmo).

El código html es el siguiente:

```
<html>
<head>
<script src="http://3dlifeplayer.dl.3dvia.com/player/install/DetectBrowser.js"></script>
<script
src="http://3dlifeplayer.dl.3dvia.com/player/install/3DLifePlayer_last_version.js"></script>
<script src="http://3dlifeplayer.dl.3dvia.com/player/install/3DLifePlayer.js"></script>
<title>pabellon38</title>
</head>

<body bgcolor="#000000" text="#00FF00" link="#FFFFFF" vlink="#C0C0C0">

<div align="center">
<script language="JavaScript">
Generate3DLifePlayerHtmlTag("pabellon_38_escenas_scripts.vmo",800,600,"Virtools");
</script>
</div>

</body>
</html>
```

6.3 Plantilla para navegar en otro modelo.

Para aumentar la funcionalidad de esta tesis el Ing. José Larios Delgado propuso adaptar el recorrido virtual por el pabellón de México a un sistema que pudiera utilizar cualquier modelo para recorrerlo.

El resultado se detallará a continuación sirviendo también como “manual para el desarrollador” para adecuar los modelos al recorrido.

[43] 3D via player. http://www.3dvia.com/products/3dvia-player/

6.3.1 Archivos del modelo a utilizar.

Los archivos para el recorrido deben estar en una carpeta pudiendo contener más de un recorrido si así se desea, siempre y cuando mantengan la estructura siguiente.

Archivo “recorrido.html”
Archivo “recorrido.vmo”
Archivo “manita.nmo”
Archivo “carpetas_de_configuracion.txt”
Archivo “pasos.mp3”
Archivo [sonido_ambiente].mp3
Carpeta [archivos_de_configuracion]
    Archivo “archivos_de_configuracion.txt”
    Archivo [condiciones_iniciales].txt
    Archivo [lista_archivos].txt
    Archivo [lista_colisiones].txt
    Archivo [lista_fichas].txt
    Archivo [lista_pisos].txt
    Archivo [lista_texturas].txt
    Archivo [posición_manos].txt
    Archivo [sonido_ambiental].txt
Carpeta [fichas]
    Archivo [nombre del objeto 1].html
    :
    :
    Archivo [nombre del objeto n].html
Carpeta [modelosNMO]
    Archivo [modelo 01].nmo
    :
    :
    Archivo [modelo n].nmo
Carpeta [texturas]
    Archivo [nombre del modelo 01].jpg
    :
    :
    Archivo [nombre del modelo n].jpg

Los nombres que están entre comillas (“ ”) no deben cambiarse, los que están entre corchetes ([ ]) sí pueden cambiarse.

Los archivos de configuración están en formato de texto para que sea más fácil cambiarlos para cada modelo que se quiera cargar.

6.3.2 Carga del modelo.

El programa pedirá el nombre del archivo de texto que contiene la configuración de archivos y carpetas del modelo que deberá cargar y lo buscará en la misma carpeta donde se encuentra el archivo vmo del recorrido (por ejemplo *ingenieria_conf.txt)*. Los nombres de las carpetas se añadirán a una lista dentro del programa.

Los nombres de las carpetas deben tener este orden para que los datos sean utilizados correctamente:

1º A*rchivos de configuración*. En esta carpeta se deben escribir los nombres de los archivos con los datos necesarios para configurar el recorrido.

2º *fichas*. En esta carpeta se pondrán los archivos html que funcionarán como fichas que ampliarán la explicación sobre algún objeto específico del recorrido. El nombre de la carpeta puede cambiar pero se sugiere que mantenga en su nombre la palabra <fichas> para identificarla (por ejemplo fichas_recorrido1, fichas_recorrido2, etc.).

3º *modelos NMO*. Esta carpeta contendrá los archivos .nmo con los modelos que serán cargados por el recorrido, incluyendo los objetos de colisión y los archivos que incluyan transparencia en su textura. El nombre de la carpeta puede cambiar pero se sugiere que mantenga en su nombre la palabra <modelos> para identificarla (por ejemplo modelos_recorrido1, modelos_recorrido2, etc.).

4º *texturas*. Esta carpeta contendrá los archivos de imagen .jpg para las texturas de los modelos que no necesitan transparencia. El nombre de la carpeta puede cambiar pero se sugiere que mantenga en su nombre la palabra <texturas> para identificarla (por ejemplo texturas_recorrido1, texturas_recorrido2, etc.).

El archivo *archivos_de_configuracion.txt* no debe cambiar de nombre. Contendrá los nombres de los archivos de datos:

\condiciones_iniciales.txt

\lista_archivos.txt

\lista_colisiones.txt

\lista_fichas.txt

\lista_pisos.txt

\lista_texturas.txt

\sonido_ambiental.txt

\posicion_manos.txt

Estos nombres deben estar precedidos por una diagonal inversa (\).

El archivo ***condiciones_iniciales.txt*** contiene un vector con las coordenadas que tendrá la cámara al iniciar el recorrido, el radio de colisión entre la cámara y los objetos de colisión y la altura de la cámara con respecto al piso.

El archivo ***lista_archivos.txt*** contiene los nombres de los archivos con los objetos del modelo y los objetos de colisión. Estos archivos tienen una extensión .nmo

El archivo ***lista_colisiones.txt*** contiene los nombres de los objetos que son utilizados como colisiones en el recorrido.

El archivo ***lista_fichas.txt*** contiene los nombres de los objetos que tienen una ficha con más información.

El archivo ***lista_pisos.txt*** contiene los nombres de los objetos que sirven como pisos. Estos objetos se ocultarán y se declararán como pisos dentro del recorrido.

El archivo ***lista_texturas.txt*** contiene los nombres de todos los objetos, estos nombres corresponden a los nombres los archivos de imagen que se usan para sus texturas. Los archivos de texturas deben estar en formato jpg.

El archivo ***posición_manos.txt*** contiene las coordenadas, en un sistema relativo, para ubicar las manos que señalan a los objetos que tienen una ficha. El sistema es relativo al centro del objeto.

El archivo ***sonido_ambiental.txt*** contiene el nombre del archivo de audio .mp3 que se utilizará como sonido ambiental.

6.3.3 Carga de las colisiones.

Con los datos del archivo *lista_colisiones.txt* se crea una lista y cada elemento será una archivo .nmo que el programa buscará, después de cargarse cada objeto se agregará a un grupo donde estarán declaradas todas las colisiones. Después se ocultarán. Se recomienda que estos objetos tengan una forma sencilla con pocos polígonos.

El proceso se repetirá para los pisos, con la diferencia que se les añadirá el atributo “piso” para que sean así reconocidos por el recorrido. Los objetos que sirven de piso están en el archivo *lista_pisos.txt*

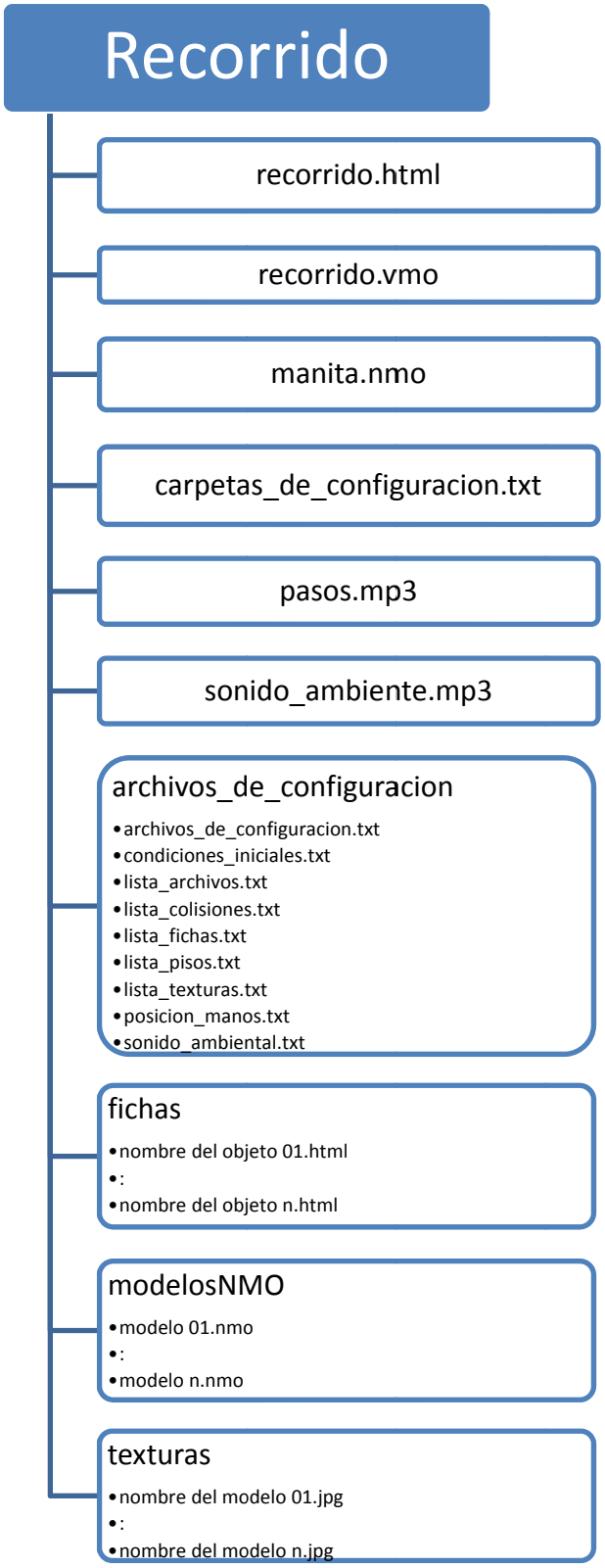

FIGURA 73. Esquema de organización para los archivos y carpetas del recorrido.

6.3.4 Carga de texturas.

Con los datos del archivo *lista_texturas.txt* se cargarán los archivos .jpg como texturas en los objetos previamente cargados. Los objetos que necesiten texturas con transparencia deberán exportarse con texturas incluidas.

6.3.5 Interacción.

Una vez que se han cargado todos los elementos del recorrido se activarán los controles de la cámara. Para avanzar y retroceder se utilizarán las flechas arriba y abajo, para girar a la derecha e izquierda las flechas derecha e izquierda. Adicionalmente se podrán utilizar las teclas A y D para desplazarse hacia la derecha e izquierda y las teclas W y S para girar hacia arriba y abajo. Al dar click sobre los objetos que estén señalados por una "mano" se desplegará una ventana con una ficha que contendrá información sobre dicho objeto.

Pruebas.

Después de terminar cada una de las funciones descritas anteriormente se hizo una prueba de funcionamiento para identificar y corregir los errores que se presentaran.

Al terminar cada prototipo se realizaron pruebas con el equipo de desarrollo, con los directores de proyecto y de la UNIDI[44] de la DGSCA[45]. Después de estas pruebas se presentaron prototipos a ProMéxico para su aprobación.

Durante las pruebas se vieron fallas que fueron corregidas para los prototipos posteriores. Las principales fallas fueron un tamaño grande del archivo que derivaba en mayor tiempo de carga del recorrido o en saturación del sistema donde se observara. Estas fallas se corrigieron al reducir el tamaño del archivo reduciendo polígonos de los objetos, reducción de los tamaños de texturas y carga externa, uso de herramientas de optimización como el nivel de detalle para la malla de los objetos.

Cuando se aprobó el último prototipo se abrió a una prueba alfa con un grupo de familiares, amigos y compañeros del equipo de desarrollo arrojando resultados cualitativos y de desempeño.

Los resultados cualitativos se refirieron a la parte visual y facilidad de uso de los controles. Para los controles las personas con menos familiaridad con computadoras se les hizo más fácil usar las teclas de navegación, en contraste con las personas que han tenido más contacto con videojuegos se les hizo atractivo el uso adicional de las teclas A,W,S,D.

[44] Unidad de Desarrollo e Investigación.
[45] Dirección General Servicios de Cómputo Académico de la UNAM.

	Tipo 1	Tipo 2	Tipo 3	Tipo 4
Procesador	Celeron 2.6 GHz	Pentium IV 2.0 GHz	Doble núcleo 2.9 GHz	4 núcleos 2.8 GHz
RAM	256 MB	512 MB	1 GB	2GB
Tarjeta de video	No	Integrada	Sí	Sí
Velocidad de conexión	512 Kbps	512 Kbps	512 Mbps	512Mbps
Comentario	Se traba	Se traba al inicio	Sin problemas	Sin problemas

FIGURA 74. Tabla comparativa de desempeño del recorrido en diferentes equipos.

Los resultados de desempeño se refieren al tipo de equipo de cómputo que se utilizaron las personas de la prueba alfa. El desempeño se incrementó de izquierda a derecha en la siguiente tabla. Los equipos similares se agrupan en un solo tipo.

6.3.6 Caso práctico: Facultad de Ingeniería de la UNAM.

Ejemplo del proceso que realiza el recorrido virtual al cargar un modelo del edificio A de la Facultad de Ingeniería de la UNAM.

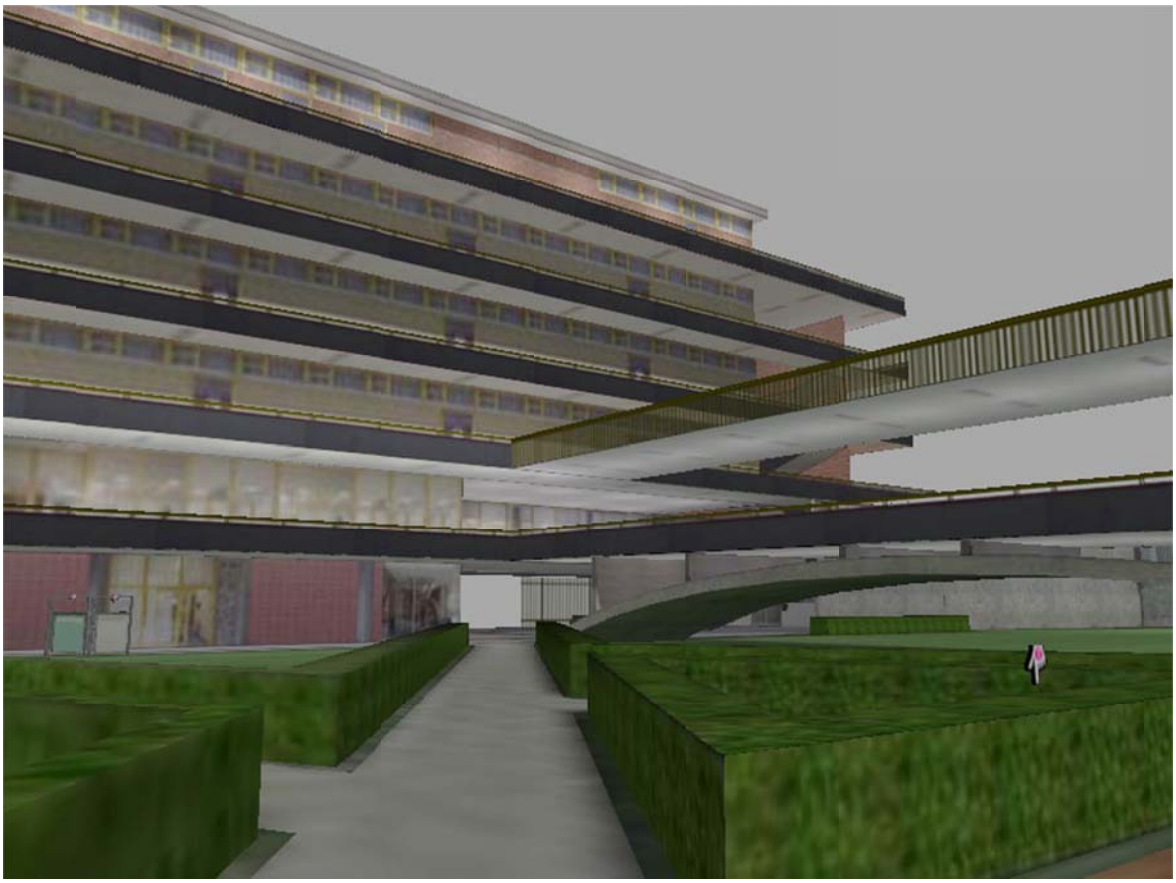

FIGURA 75. Imagen del recorrido virtual del modelo de un edificio de la Facultad de Ingeniería de la UNAM.

El programa ejecuta el *script* "elección de modelo" y solicita que se escriba el nombre del archivo de configuración (en formato de texto plano .txt) del recorrido que se quiera ejecutar. Se pueden tener más de un recorrido teniendo cada uno sus propios archivos y carpetas de configuración y datos. Enseguida busca el archivo *ingenieria_conf.txt*[46], que debe estar en la misma carpeta en la cual se encuentra el programa del recorrido.

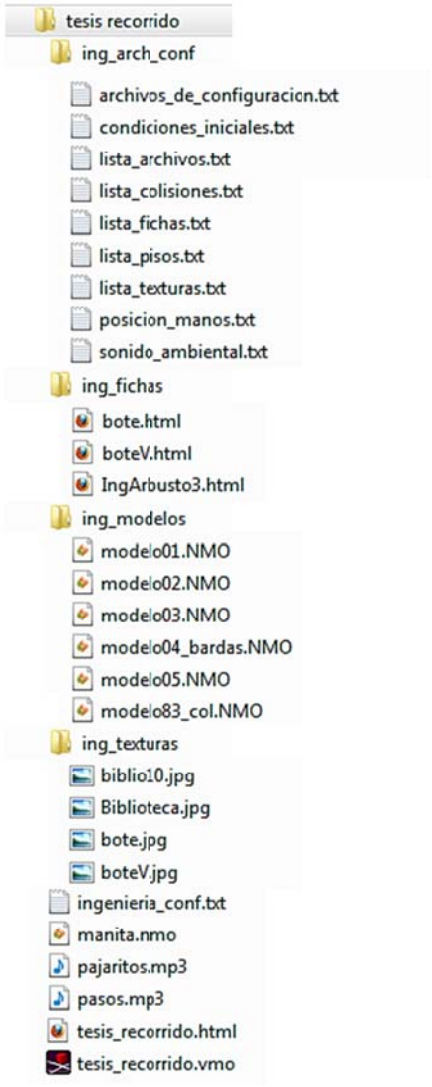

FIGURA 76. Ejemplo de la organización de archivos y carpetas del modelo del edifico A de la Facultad de Ingeniería de la UNAM para utilizarlo en este recorrido.

Si hay algún error en el nombre o no existe el archivo repetirá la petición y búsqueda hasta que sea exitosa. Cuando encuentre el archivo guardará su contenido en una lista llamada lista_carpetas.

[46] Nombre que debe ingresarse para el recorrido con este modelo.

En el archivo ingenieria_conf.txt contiene los nombres de las carpetas con los datos de configuración para el recorrido. Las carpetas son las siguientes:

- ing_arch_conf
- ing_fichas
- ing_modelos
- ing_texturas

Por último manda ejecutar el *script* siguiente, que se llama “condiciones iniciales”.

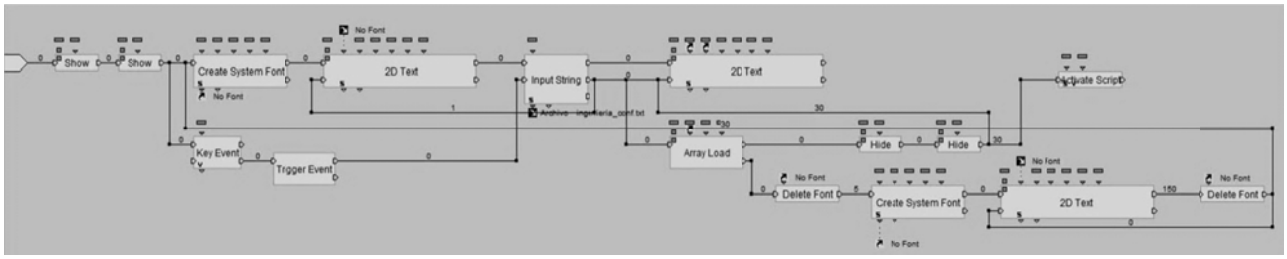

FIGURA 77. Script elección de modelo

El siguiente *script* “condiciones iniciales” comienza desactivando el *script* “elección de modelo”. Luego busca en la lista de “carpetas de configuración” la carpeta “ing_arch_conf” para buscar en ella el archivo “archivos_de_configuracion.txt”

La carpeta “carpetas de configuración” tiene los siguientes archivos:

- archivos_de_configuracion.txt
- condiciones_iniciales.txt
- lista_archivos.txt
- lista_colisiones.txt
- lista_fichas.txt
- lista_pisos.txt
- lista_texturas.txt
- posicion_manos.txt
- sonido_ambiental.txt

El contenido del archivo”archivos_de_configuracion.txt” contiene los nombres de los archivos con los datos necesarios para la configuración del recorrido y son colocados en la lista “archivos_configuracion”.

Para finalizar activa el *script* “carga objetos”.

El *script* “carga objetos” busca el archivo “lista_archivos.txt” dentro de la carpeta “ing_arch_conf” y su contenido lo almacena en la lista “lista_archivos”. En este archivo se encuentran los nombres de los archivos que contienen los modelos del recorrido y los va

cargando desde los archivos con extensión .nmo que se encuentran en la carpeta “ing_modelos”.

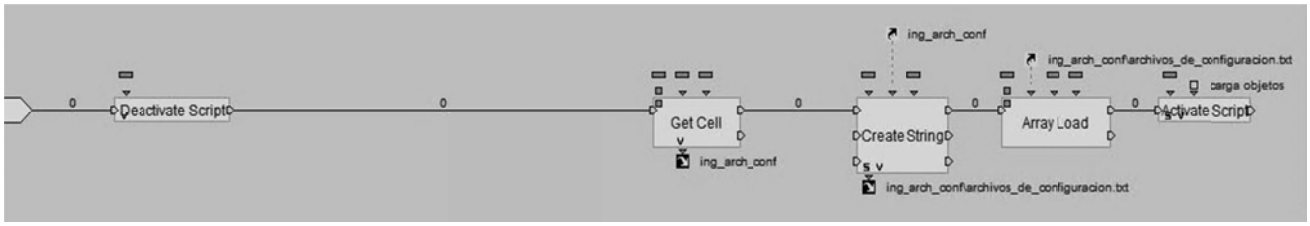

FIGURA 78. Script condiciones iniciales.

Posteriormente se desactiva el *script* “condiciones iniciales” y se activa “carga de texturas”.

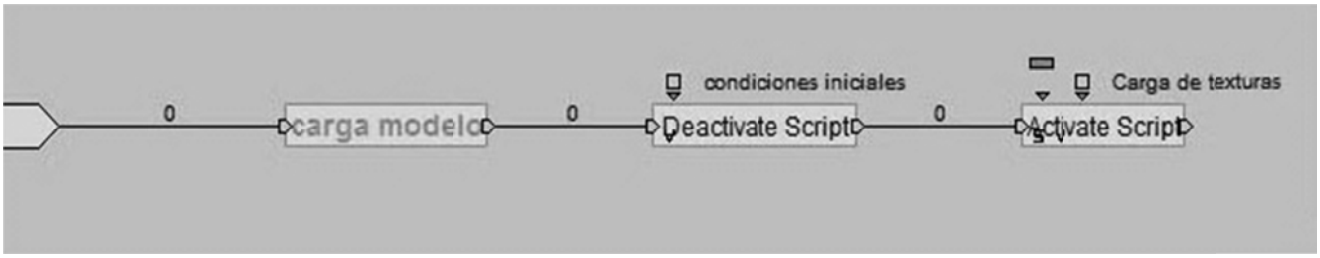

FIGURA 79. Script carga objetos.

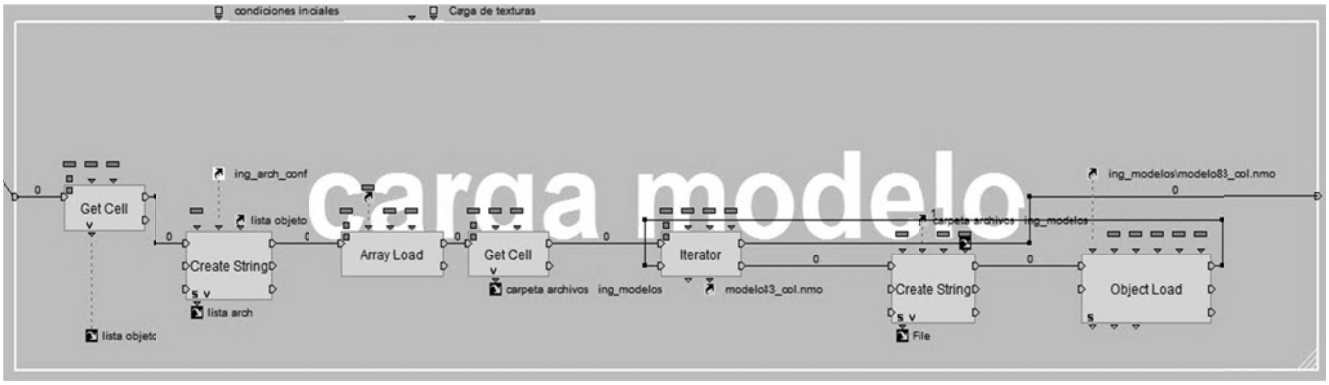

FIGURA 80. Detalle de la función carga modelo del script carga objetos.

El *script* “carga de texturas” pone en una lista los nombres de los archivos de textura que se encuentran en el archivo “lista_texturas.txt” y después carga cada textura y lo asigna al objeto al cual pertenece. Los archivos de textura deben tener el mismo nombre que el objeto para que pueda ser asignado correctamente.

Después ponen en una lista los datos para la ubicación de la cámara que se encuentran en el archivo “archivos_de_configuracion.txt” y sitúa la cámara a la distancia que se haya establecido. Las coordenadas utilizan un sistema local de coordenadas con origen en el centro del primer objeto que se haya cargado.

Al final desactiva el *script* “carga objetos” y activa “_colisiones carga”.

FIGURA 81. Script carga de texturas.

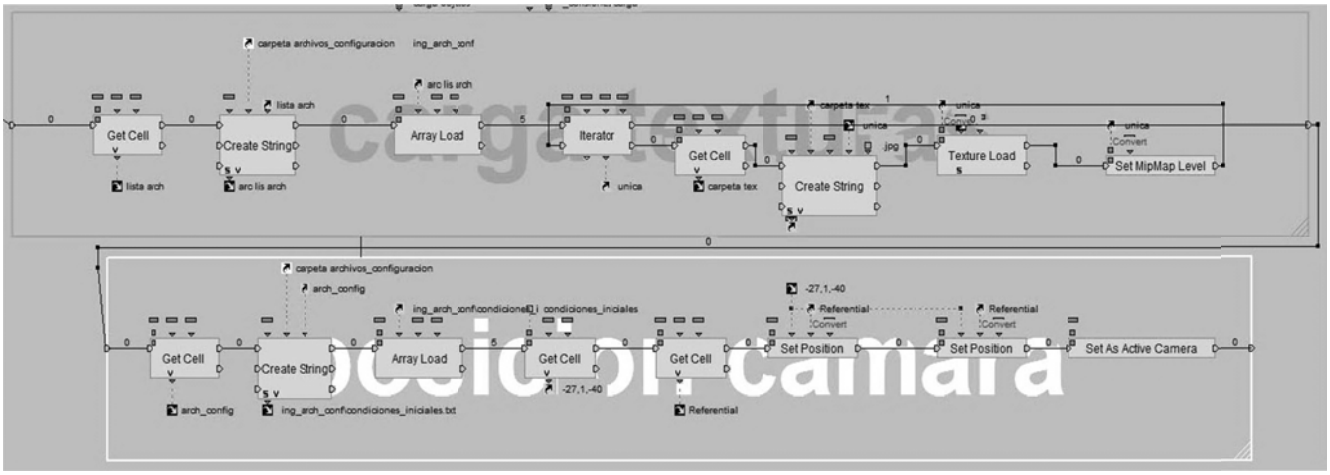

FIGURA 82.Detalle de las funciones carga texturas y posición cámara del script carga de texturas.

FIGURA 83.Script carga colisiones.

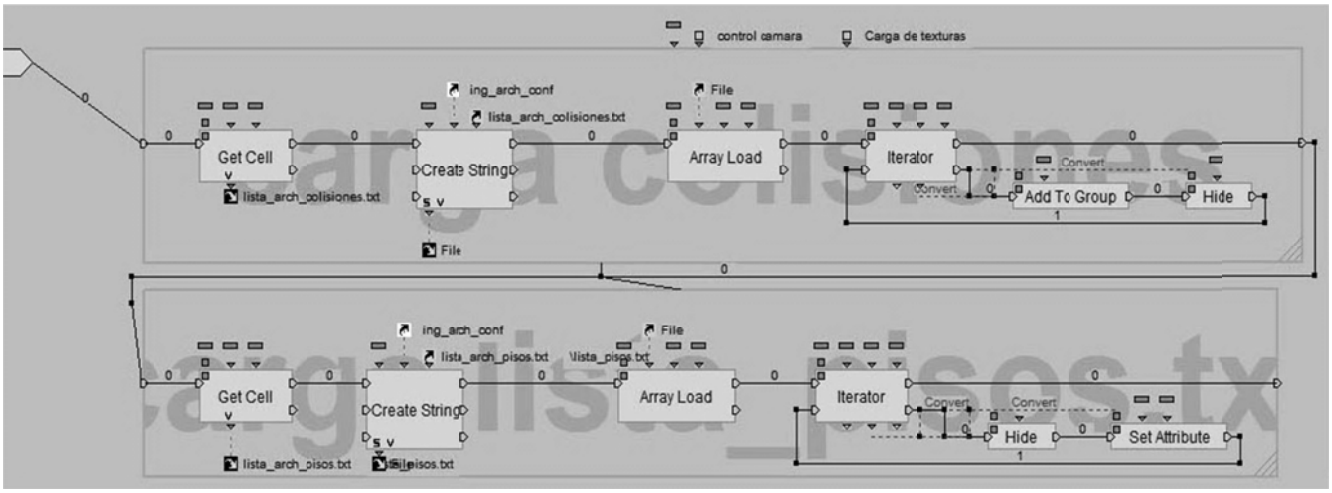

FIGURA 84. Detalle de las funciones carga colisiones y carga pisos del script carga colisiones.

El *script* “carga colisiones” pone en una lista los nombres de los archivos que contienen los objetos de colisión que se encuentran en el archivo “lista_colisiones.txt”, con estos datos carga los objetos que se encuentran en la carpeta “ing_modelos”, los agrega al grupo “_colisiones” y los oculta.

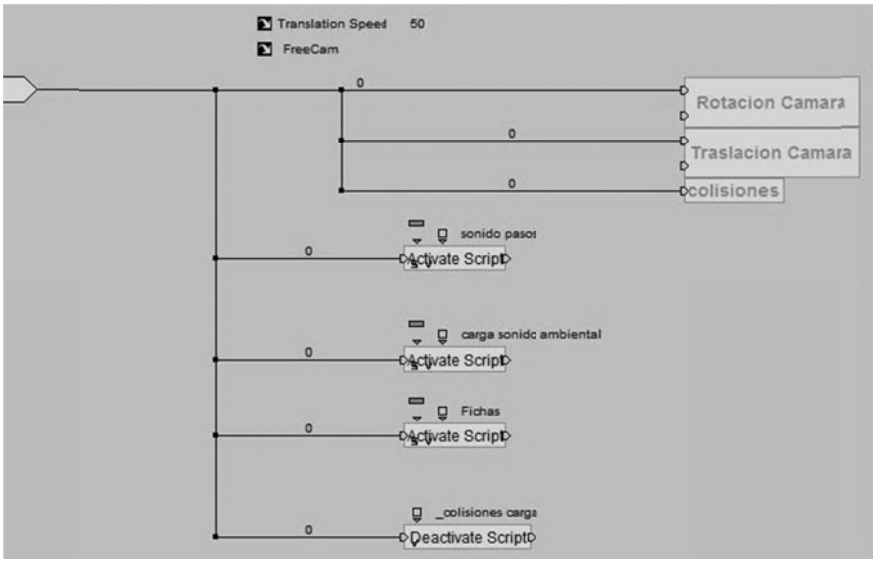

FIGURA 85.Script control cámara.

Después pone en una lista los nombres de los archivos que contienen los objetos que servirán de pisos, los carga en el recorrido, les asigna el atributo “*floor”* y los esconde.

Para terminar activa el *script* “control cámara” y desactiva “carga de texturas”.

El *script* “control cámara” tiene tres funciones principales, “rotación cámara”, “traslación cámara” y “colisiones”. Además de realizar esta funciones activa los *script* “sonido pasos”, “carga sonido ambiental” y “fichas” y desactiva “_colisiones carga”.

En la función “rotación cámara” se espera a que las teclas “flecha derecha”, “flecha izquierda”, “w” o “s” se presionen para que haya una rotación de la cámara hacia la derecha, izquierda, arriba o abajo respectivamente. Cuando la rotación se hace hacia arriba o abajo se tiene un límite para que no rebase los 90 o -90 grados.

En la función “traslación cámara” se espera a que las teclas “flecha arriba”, “flecha abajo”, “a” o “d” sean presionadas para que la cámara se mueva hacia adelante, atrás, a la derecha o a la izquierda respectivamente. Mientras se mantengan presionadas estas teclas cada segundo se multiplica un valor a un vector unitario con estas direcciones. El sistema de referencia que usan estos vectores es local con centro en el dummy de traslación de la cámara.

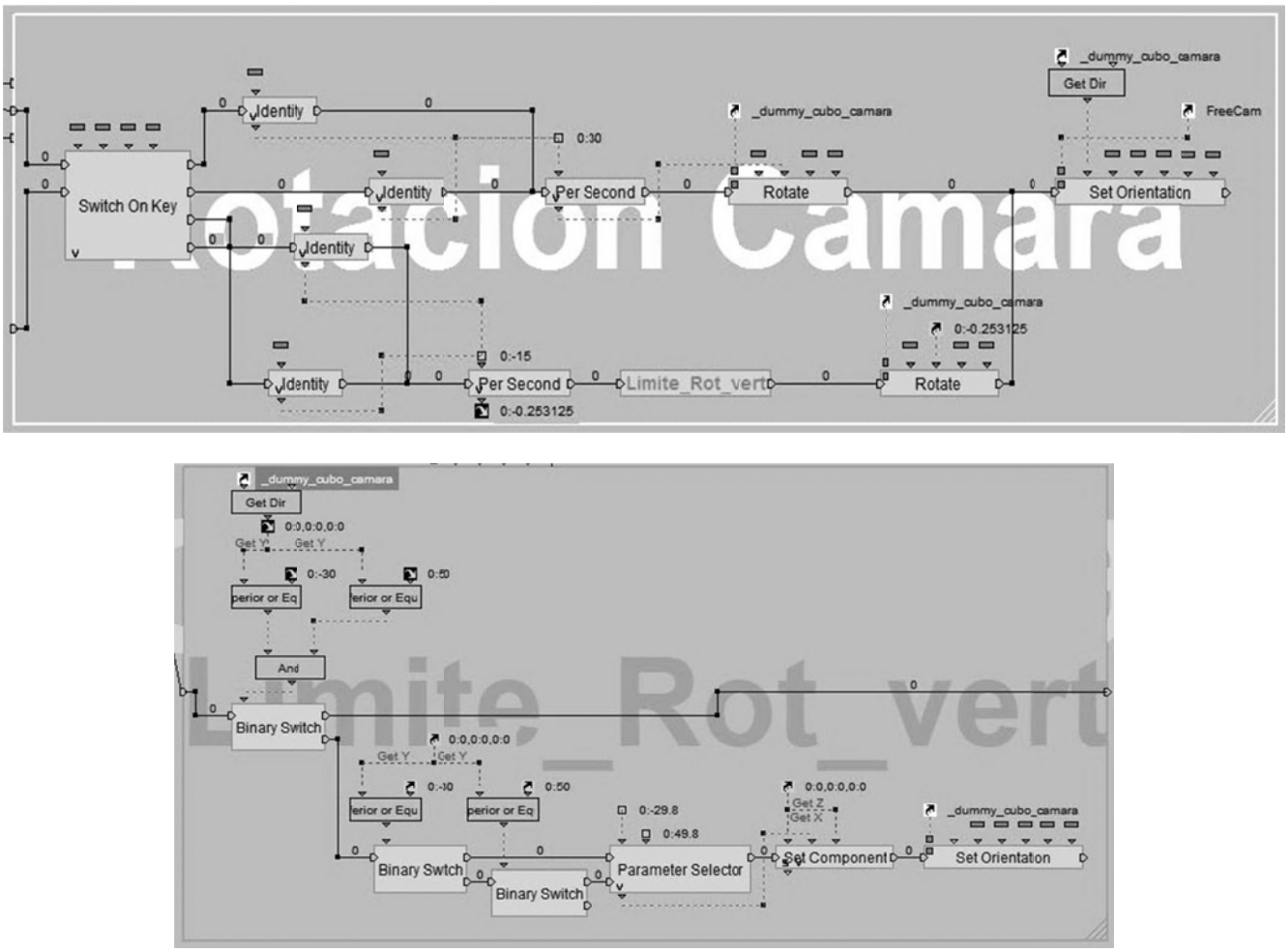

FIGURA 86.Detalle de la función Rotación cámara del script control cámara.

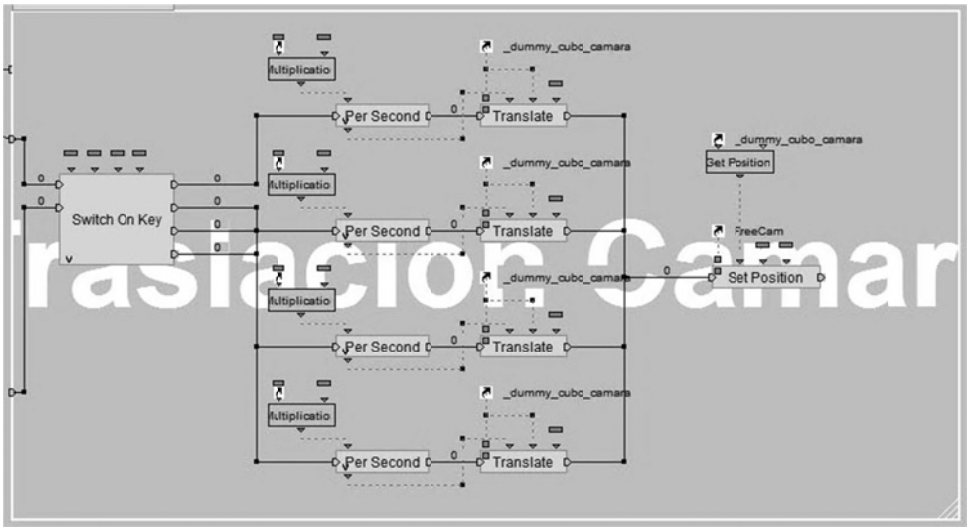

FIGURA 87.Detalle de la función Traslación cámara del script control cámara.

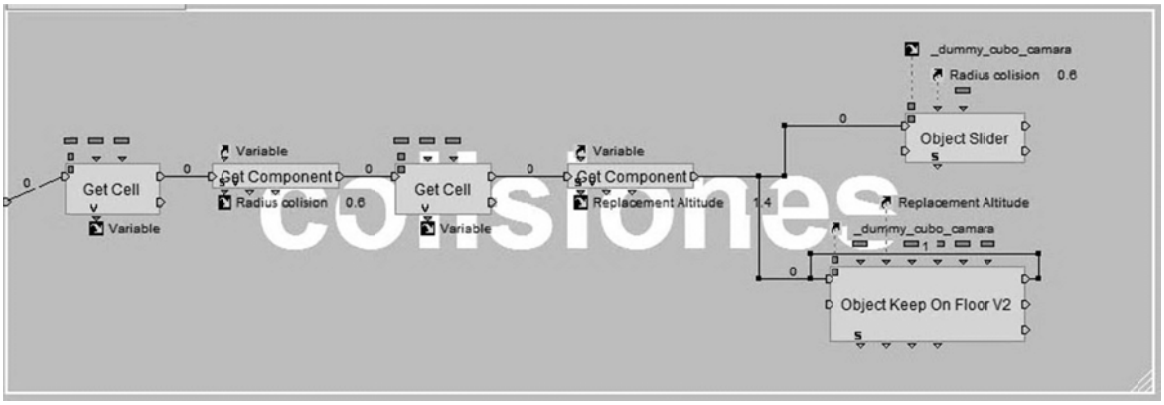

FIGURA 88. Detalle de la función colisiones del script control cámara.

En el *script* "fichas" se espera a que se apriete el botón izquierdo del *mouse*, cuando ocurre se verifica cuál objeto se encuentra en ese momento entre el puntero del mouse y la cámara y se busca un archivo html con el nombre de este objeto en la carpeta "fichas", si el objeto tiene una ficha asociada se abre una ventana del navegador de internet con datos sobre este objeto, si no hay ficha no ocurre cambio alguno.

El *script* "carga sonido ambiental" carga primero el archivo "sonido_ambiental.txt" que contiene el nombre del archivo de sonido "pajaritos.mp3" que será cargado en el recorrido. Por último se activa el *script* "sonido ambiental". Se puede asignar otro archivo .mp3 como sonido ambiental.

FIGURA 89. Script fichas.

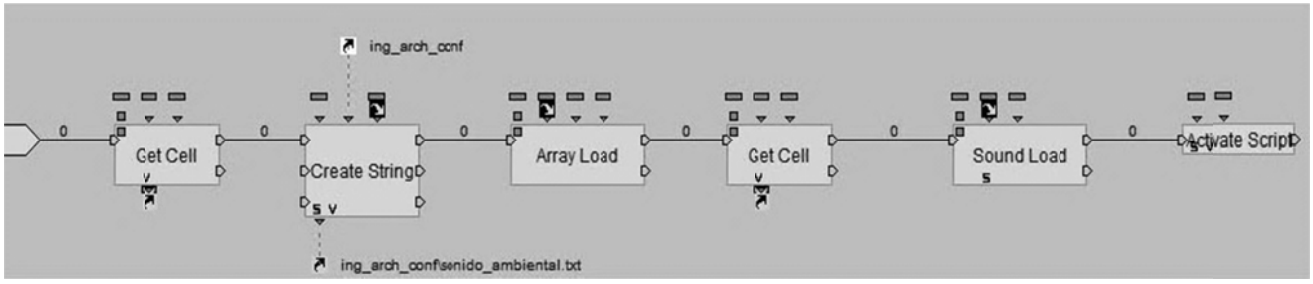

FIGURA 90. Script carga sonido ambiental.

El *script* “sonido ambiental” reproduce el archivo de audio que servirá como fondo para el recorrido. Después desactivará el *script* “carga sonido ambiental”.

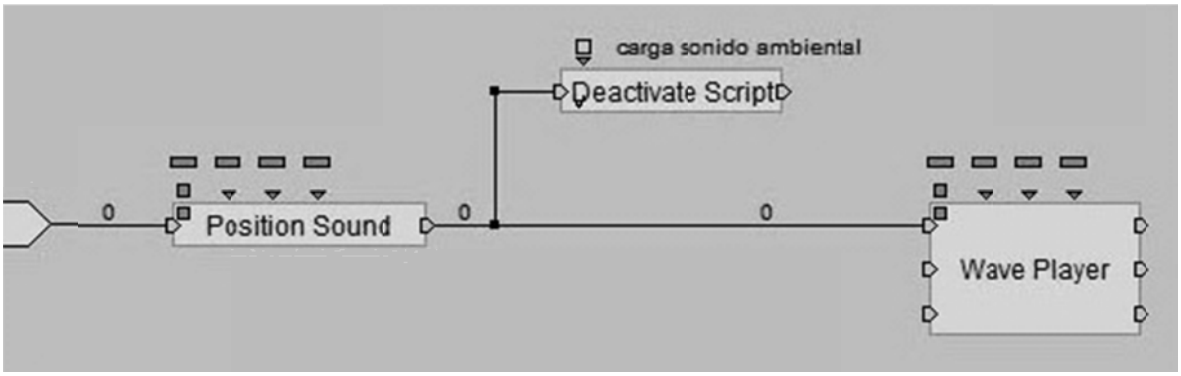

FIGURA 91. Script sonido ambiental.

El *script* “sonido pasos” reproduce con un ciclo continuo una grabación de pasos mientras las teclas “flecha arriba”, “flecha abajo”, “A” o “D” se encuentran presionadas, cuando dejan de presionarse se detiene la reproducción. Después se activa el *script* “pon manitas”.

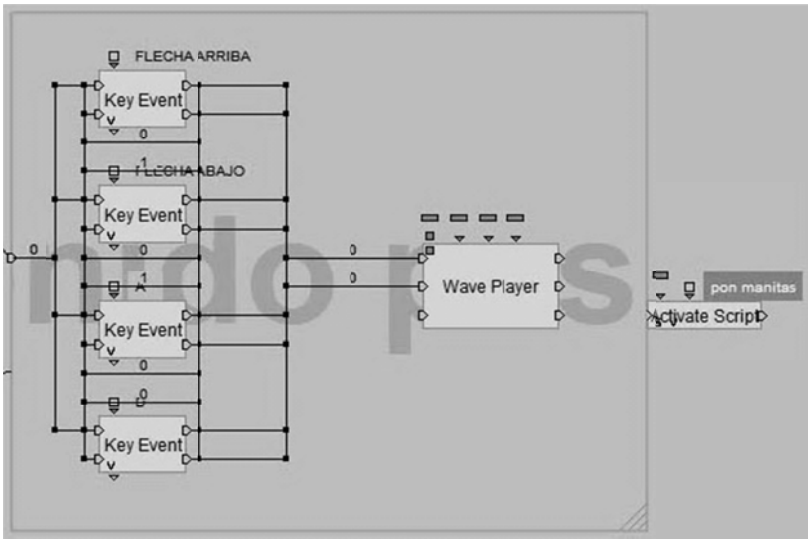

FIGURA 92. Script sonido pasos.

El *script* “pon manitas” carga los nombres de los objetos que tienen una ficha informativa asociada. Por cada uno de estos objetos se cargará una “manita”, es decir, un objeto en forma y con la imagen de una mano apuntando con el dedo índice que señalará el objeto de interés. También se cargará un vector con la posición, relativa al centro del objeto, en la cual se colocará la “manita”. Los nombres de los objetos y los vectores se encuentran en el archivo “posición_manos.txt”.

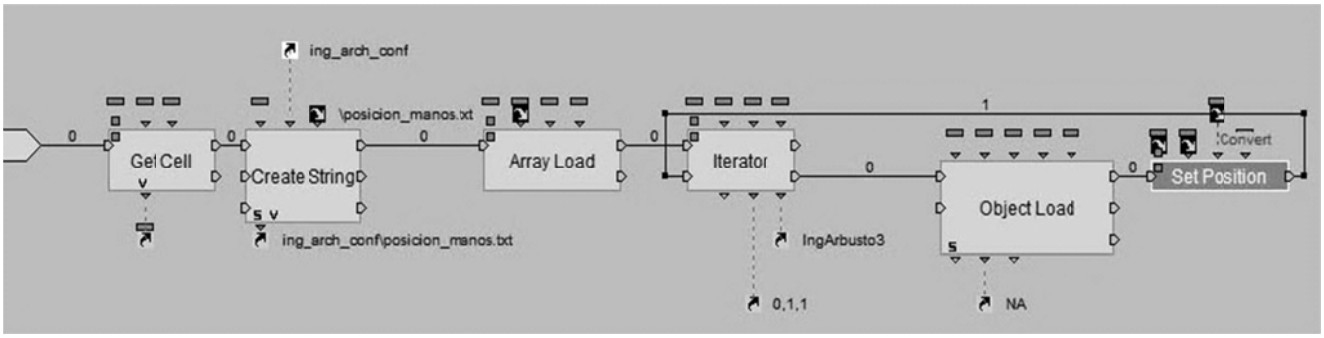

FIGURA 93. Script pon manitas.

Pruebas.

Se utilizaron las misma pruebas que para el recorrido del pabellón y otras pruebas para este modelo fueron revisar el número de polígonos, tamaño de los archivos de las texturas y de exportación de los objetos. Los resultados de estas pruebas son los siguientes.

Se tienen 260 objetos con 10 504 cara en total y un total de 5.93 MB para las texturas que utiliza. La programación de cada script fue probada con este modelo corriegiendo las fallas se presentaron antes de continuar con el siguiente. El tiempo de carga fue de 30 segundos.

El pabellón tuvo 967 objetos con 250 638 caras y utiliza 14.4 MB para el total de las texturas. El tiempo de carga fue de 50 segundos.

7. Conclusiones.

La Ingeniería utiliza los conocimientos de matemáticas y física para desarrollar métodos y productos que resuelvan necesidades de la sociedad. Hay dos grandes vertientes de la Ingeniería: la investigación y el desarrollo de sistemas. Esta tesis siguió el camino de desarrollo para solucionar un problema específico utilizando herramientas computacionales ya existentes.

El primer problema a solucionar fue la creación de un recorrido virtual que reprodujera el pabellón de México en la Expo Universal 2010 realizada en Shanghái, China.

El resultado final fue dicho recorrido virtual y un método para realizar recorridos virtuales con características y funciones similares. La funcionalidad principal consiste en recorrer el pabellón con una vista en primera persona y mostrar información al dar click en algunos objetos.

Las herramientas utilizadas fueron un editor de objetos tridimensionales 3DS max[47] y un motor de juegos 3D via virtools[48]. 3DS max se utilizó por tener un conocimiento previo del programa lo cual redujo el tiempo de realización de los objetos; virtools se utilizó debido a que era una especificación del proyecto, al conocimiento previo, a las herramientas que contiene, por la facilidad de programación y porque el visualizador requerido[49] se puede adquirir gratuitamente.

Primero se construyeron los modelos tridimensionales usando el programa 3DS max. Para crearlos se utilizaron las medidas de los planos arquitectónicos del pabellón proporcionados por los constructores del pabellón de México en China y se hizo el levantamiento a partir de primitivas gráficas (cubos, cilindros, esferas). Para ajustar dichas primitivas a las proporciones y formas originales se modificó la malla del objeto manipulando sus vértices y aristas.

Otra técnica para hacer los objetos que se utilizó fue el modelado basado en imágenes, ya que no se contaba con diagramas o planos; éste fue el caso de las piezas museográficas. Primero se construyeron primitivas gráficas y se fueron modificando dividiendo sus aristas, extrudiendo algunas caras y manipulando sus vértices y aristas. Se revisó el número de polígonos de cada objeto y dependiendo de su importancia para el recorrido se redujeron algunos de ellos, también se borraron caras que no aparecerían a la vista de la cámara.

Teniendo los modelos hechos se procedió a poner una iluminación a la escena, primero general y luego luces específicas donde hicieran falta.

Se utilizó un motor de render de trazado de rayos para obtener un efecto de iluminación, sombras, reflejos y texturas más llamativas.

[47] La versión utilizada fue 3D studio max 2010 de autodesk.
[48] La versión utilizada fue Virtools 5 de Dasault systèmes.
[49] 3D via player

Después de la iluminación se agregaron las texturas correspondientes a cada objeto. En algunos se utilizaron texturas procedurales, en otras se usó repetición de patrones y en otras una imagen específica con manipulación de sus coordenadas UV.

Después de tener todas las texturas se procedió a realizar el "horneado de texturas" para que al importarlo en el motor de juego tuviese menos problemas de compatibilidad. Con el horneado de texturas se utilizaron menos luces y efectos de texturas en el motor de juegos, reduciendo el cálculo de estas características.

Posteriormente procedí a importar los modelos tridimensionales al motor de juegos y poner tres luces generales para que iluminaran los objetos, los efectos de luz y sombra están incluidos como imágenes en las texturas de cada uno. Se agregó una cámara y se ocultaron los objetos de colisión.

A continuación se realizó la programación del recorrido, comencé con la programación de los controles de la cámara, su colisión, luego siguieron las funciones (*scripts*) para la carga de texturas (las texturas se cargan después del recorrido porque así el observador no requiere esperar a que se cargue completamente para navegar en él), la carga de escenas (para reducir los elementos a mostrar) y la interacción con algunos objetos.

Con esta tesis se propone un método a seguir para realizar un recorrido virtual y se tomó como ejemplo específico el Pabellón de México en la Expo Universal Shanghái 2010. Se propuso que este trabajo fuera atractivo visualmente y que sirviera de estímulo para la enseñanza. El método admite su aplicación a una gran variedad de temas haciéndolo flexible para todos los campos de educación.

Calculo el tiempo de vida de este recorrido en cuatro años tal como está porque al ser su tema principal una exposición universal su atractivo caduca pronto. Después de caducar el tema principal su atractivo principal serán la navegación, la parte visual y la parte técnica.

Derivado del desarrollo de esta tesis y como propuesta del Ingeniero José Larios Delgado, se realizaron modificaciones a la programación del recorrido para que pudiera cargarse un modelo de forma externa, es decir que no fuera necesario que el modelo estuviera contenido en el archivo que ejecuta el recorrido; esto se realizó para hacer más flexible y amplio el uso del recorrido.

El resultado es un archivo html que llama a un archivo vmo[50] que a su vez llama varios archivos nmo[51] que contienen los modelos, esto aunado a la carga dinámica de texturas y funciones que ya existían en el recorrido del pabellón. El recorrido también busca en archivos de texto los datos para configurar la carga de los modelos. Opté por usar archivos de texto por su facilidad para modificarlos.

[50] Tipo de archivo del motor de juegos.
[51] Tipo de archivos de datos y modelos para el motor de juegos.

La segunda versión del recorrido utiliza un modelo del edificio A de la Facultad de Ingeniería de la UNAM para mostrar su funcionamiento.

Calculo el tiempo de vida de este recorrido en diez años sin realizar modificaciones porque permite el uso de cualquier modelo que se realice con el formato del motor de juegos (nmo). La reducción de su vida será porque deje de existir exportadores de programas de modelado hacia el motor de juegos. Este pronóstico se hizo porque Dassault systèmes actualmente vende un programa de modelado y dejó de hacer el exportador para programas de otras compañías como 3D studio max, sin embargo es posible programar un exportador y algunas personas que lo programan lo pusieron a disposición de quien lo necesitara. Otro factor que reduce el tiempo de vida es la evolución de herramientas para realizar recorridos virtuales.

Para realizar este recorrido tuve que programar funciones que no se encuentran en la biblioteca de virtools, estas funciones se pueden reutilizar en virtools o para crear funciones similares en otros motores de juego. A mi parecer las funciones más útiles para alguien que quiera reutilizarlas en un recorrido con estas características son la carga externa de objetos, carga externa de texturas, carga de sonidos ambientales y visualización de fichas informativas porque reducen el tiempo de carga del recorrido y dan más libertad para modificar los modelos que se mostrarán sin cambiar la programación del recorrido.

Para aumentar el tiempo de vida de este navegador se proponen realizar las siguientes mejoras y hacerlo más atractivo.

- Incluir una rutina que permita crear las listas de los archivos de configuración mediante una ventana que gestione las carpetas y archivos, parecido al explorador de carpetas de Windows.
- Incluir un campo donde se puedan introducir los datos para la ubicación de la cámara, su altura relativa al piso, radio de colisión, posición relativa de las “manitas informativas” y otros datos parecidos a éstos sin necesidades de reiniciar el recorrido cada vez que se modifiquen dichos datos.
- Crear una función para asignar valores necesarios para utilizar un algoritmo de optimización del tipo Nivel de Detalle para la geometría de los objetos.
- Crear una función para optimizar el manejo de texturas usando *Mipmaps*.
- Crear una función que permita la apertura y el cierre de las puertas de acceso que tenga el modelo.
- Crear una función que permita el uso de diferentes materiales en los pisos. Para que la cámara reaccione de diferente manera si el piso tiene un atributo “pasto”, “hielo”, “asfalto”, etc.
- Crear una función que permita la carga y uso de sonidos puntuales.
- Incluir una rotación sobre su eje de las “manitas informativas”.

- Crear una interfaz más atractiva.
- Crear una función que permita dividir y cargar el modelo por áreas para reducir la carga de procesamiento al ocultar los objetos que no se observan en la cámara.
- Incluir una visualización estereoscópica pasiva con anáglifos.
- Crear una función de captura de imagen de la cámara que pueda almacenarse en un archivo de imagen como recuerdo de la visita.

Un impedimento para completar este recorrido se presentó hacia la parte final, la licencia del motor de juegos virtools caducó y no fue renovada. Continué terminando el recorrido con una licencia de evaluación pero cuando puse los archivos finales en el servidor la página fue bloqueada. Por esta razón se presentará el recorrido en un sistema local.

8. Manuales.

Manual técnico.

En este manual se darán los pasos a seguir para configurar los datos para cargar un modelo en este recorrido. Se tomará el modelo del edificio A de la Facultad de Ingeniería de la UNAM para ilustrar la configuración.

Los archivos y carpetas de un modelo se deben poner dentro de la carpeta donde están los archivos del recorrido de como en la siguiente figura.

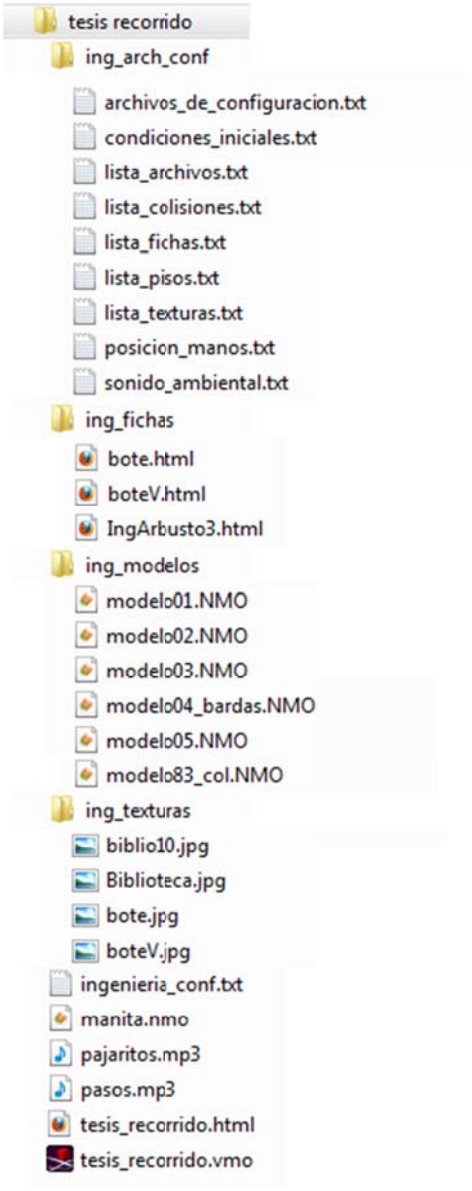

FIGURA 94. Orden de las carpetas y archivos para un modelo.

Se debe tener un archivo de texto con los nombres de las carpetas que contendrán los archivos de configuración, los archivos con las fichas informativas, los modelos y las texturas.

Se sugiere que este archivo tenga el nombre del modelo a cargar seguido de la palabra configuración abreviada, por ejemplo *ingeniería_config.txt*, para tener un mejor orden. Asimismo se sugiere que los nombres de estas carpetas tengan una parte del nombre del modelo para identificarlas más fácilmente.

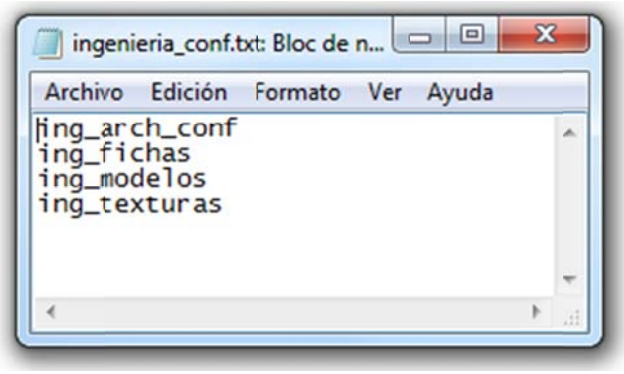

FIGURA 95. Contenido del archivo ingeniería_conf.txt

La carpeta de archivos de configuración contendrá los archivos *archivos_de_configuracion.txt, condiciones_iniciales.txt, lista_archivos.txt, lista colisiones.txt, lista_fichas.txt, lista_pisos.txt, lista_texturas.txt, posición_manos.txt y sonido_ambiental.txt* con los datos necesarios para cargar y configurar el recorrido.

El archivo *archivos_de_configuracion.txt* contiene los nombres de los archivos con datos para configurar el recorrido. Estos nombres deben estar precedidos por una diagonal inversa.

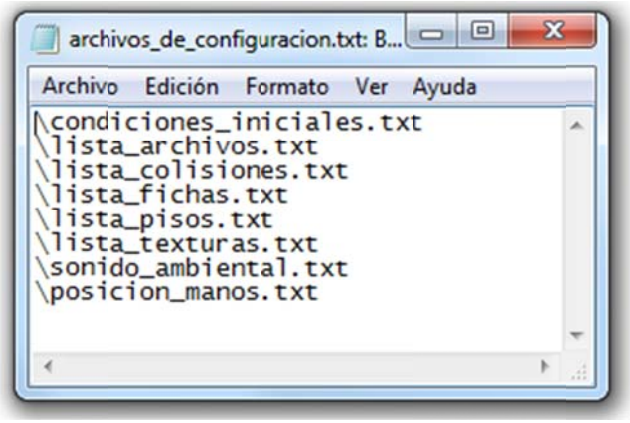

FIGURA 96. Contenido del archivo archivos_de_configuracion.txt

El archivo condiciones_iniciales.txt contiene un vector con las coordenadas para ubicar la cámara, el radio de colisión y la altura de la cámara con respecto al piso. Se sugiere comenzar con un vector 0,0,0 y modificar una por una las coordenadas para ubicar la cámara en el lugar donde se desee. Para el radio de colisión se sugiere hacer una prueba al pasar por un pasillo estrecho del modelo y modificar el valor si no permite el acceso; para la altura

de la cámara se sugiere hacer una prueba estando ente un piso y un techo y modificarla si brinca de un piso a otro.

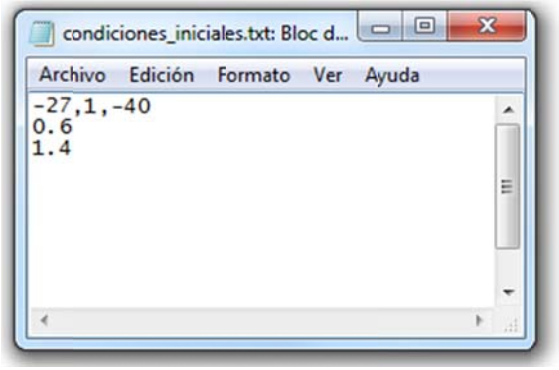

FIGURA 97. Ejemplo de valores para la configuración de la cámara.

El archivo lista_archivos.txt contiene los nombres de los archivos con los modelos que deben cargarse. Estos archivos deben tener extensión nmo procedente del exportador de virtools.

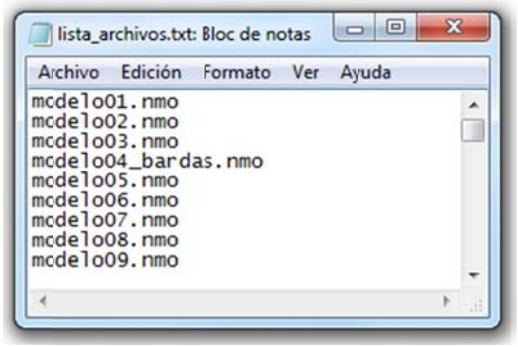

FIGURA 98. Archivo con los nombres de los archivos con los modelos que se cargarán.

El archivo lista_colisiones.txt contiene los nombres de los objetos que servirán para colisión con la cámara. Se sugiere que comiencen con un guión bajo para diferenciarlos del resto de objetos.

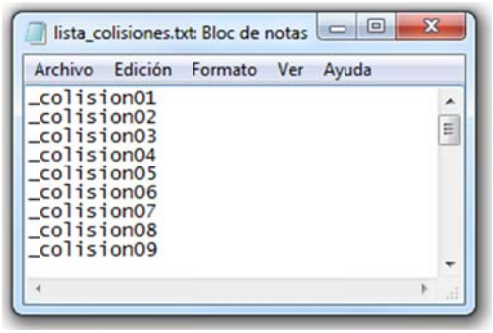

FIGURA 99. Nombres de los objetos de colisión.

El archivo lista_fichas.txt contiene los nombres de los objetos que tienen una ficha asociada.

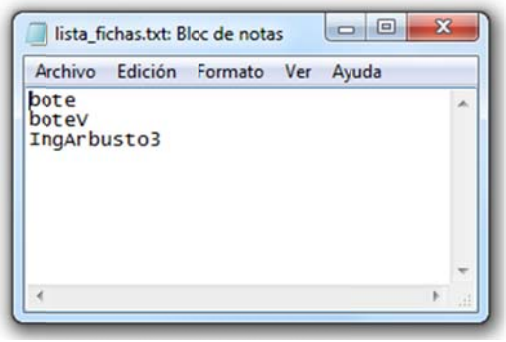

FIGURA 100. Contenido del archivo lista_fichas.txt

El archivo lista_pisos.txt contiene los nombres de los objetos que sirven como piso, se sugiere que comiencen con un guión bajo y que incluyan en su nombre la palabra piso para diferenciarlos de los demás objetos.

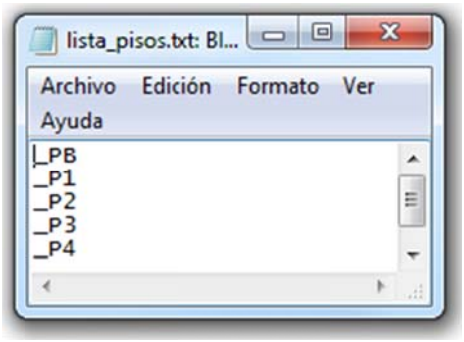

FIGURA 101. Contenido del archivo lista_pisos.txt

El archivo lista_texturas.txt contiene los nombres de los objetos y debe ser el mismo nombre el de su textura para que sean asignadas correctamente.

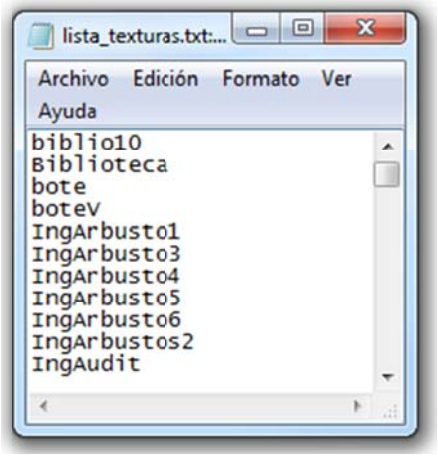

FIGURA 102. Contenido del archivo lista_texturas.txt

El archivo posición_manos.txt contiene una columna con vectores de tres componentes para ubicar la manita que señales que tiene una ficha asociada y una columna con el nombre del objeto que tiene la ficha y manita asociadas. Se sugiere iniciar el vector con los valores 0,0,0 y modificar cada componente por separado hasta ubicar la manita en el lugar deseado.

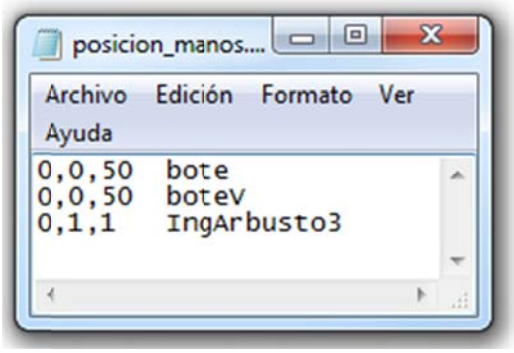

FIGURA 103. Contenido del archivo posicion_manos.txt

El archivo sonido_ambiental.txt contiene el nombre del archivo de audio para emplearlo como sonido ambiental.

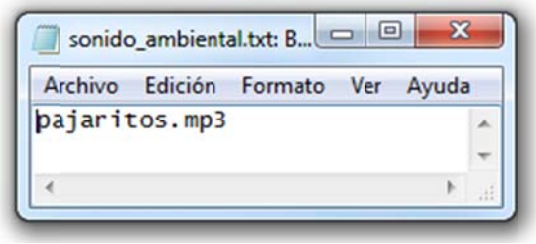

FIGURA 104. Contenido del archivo sonido_ambiental.txt

La carpeta _fichas[52] contiene los archivos html de las fichas que se mostrarán. Estos archivos deben tener el mismo nombre que el objeto que las contiene.

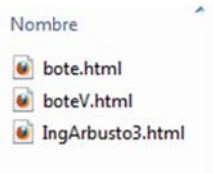

FIGURA 105. Contenido de la carpeta ing_fichas.

La carpeta ing_modelos contiene los archivos nmo con los modelos que serán cargados en el recorrido. Estos archivos son creados con el exportador de virtools y los modelos, a

[52] En este ejemplo la carpeta se llama ing_fichas

excepción de los que necesitan mapas de transparencia[53], fueron exportados con texturas referenciadas. En esta carpeta se incluyen los archivos con los objetos para colisión y pisos.

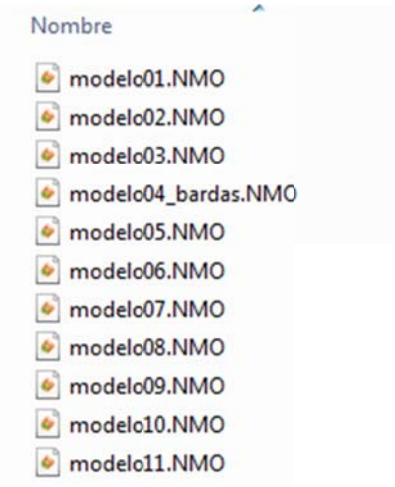

FIGURA 106. Contenido de la carpeta ing_modelos.

En la carpeta ing_texturas se encuentran los archivos de imagen[54] que sirven para las texturas. Deben tener el mismo nombre que el del objeto al cual pertenecen.

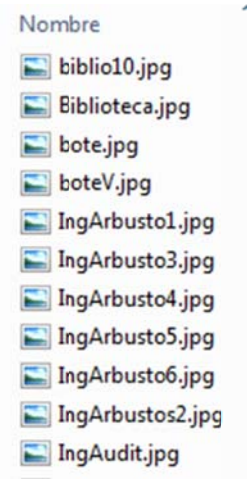

FIGURA 107. Contenido de la carpeta ing_texturas.

[53] Los modelos con mapas de transparencia se exportan con texturas embebidas.
[54] Los archivos de imagen para texturas deben estar en formato jpg.

Manual de usuario.

En este manual se mostrarán las instrucciones para utilizar el recorrido.

Primero se debe ejecutar el archivo tesis_recorrido.html el cual abrirá una ventana con el navegador de internet predeterminado y buscará si se tiene instalado el visualizador, si no está instalado o si existe una versión más actual mostrará un vínculo para descargarlo e instalarlo gratuitamente. Si el visualizador se encuentra instalado y actualizado comenzará el recorrido. Se mostrará una pantalla gris y se pedirá que se introduzca el nombre del archivo de configuración del modelo que quiera cargarse. Si el archivo se encuentra se iniciará el recorrido y mostrará el modelo cargado.

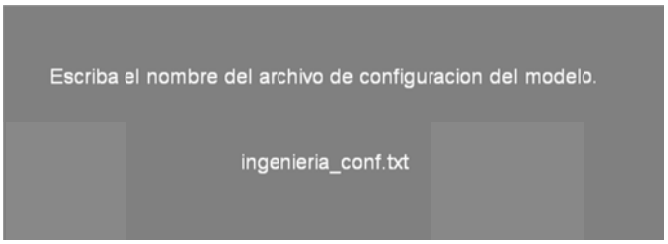

FIGURA 108. Pantalla de inicio del recorrido.

Cuando aparezca el modelo se podrá comenzar a navegar por él utilizando las teclas de navegación (flechas) para avanzar, retroceder, girar a la izquierda o derecha o las teclas A, D, W o S para desplazarse a la izquierda, derecha, girar arriba o abajo respectivamente.

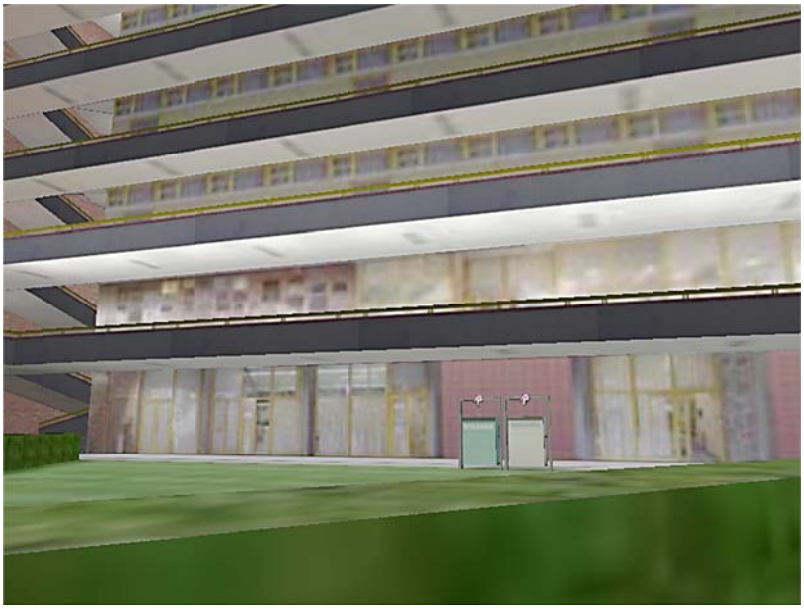

FIGURA 109. Modelo cargado en el recorrido.

Al dar click a los objetos que tengan una “manita” se abrirá una ficha con información referente al objeto o a algún tema relacionado con él.

FIGURA 110. Teclas para el movimiento de la cámara.

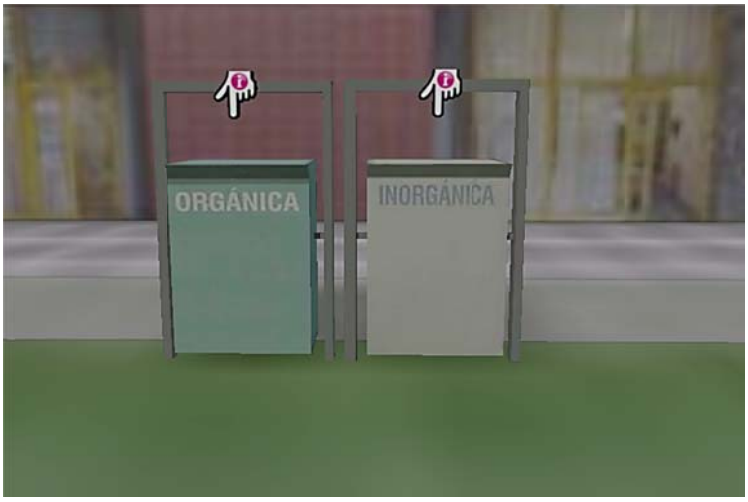

FIGURA 111. Objetos con “manitas”.

9. Glosario.

Arista. En geometría es el segmento de recta donde intersecan dos planos. Por extensión también se conoce con este nombre al segmento común que tienen dos caras adyacentes de un poliedro, y que forman al estar en contacto.

Búfer de imagen. Es una memoria auxiliar en donde se almacena momentáneamente la información que pasará a la memoria de video.

Componentes de color RGB. Son los valores de nivel de color para los colores primarios rojo (Red), verde (Green) y azul (Blue).

Dummy. Objeto que no reconoce el motor de render sirve para ayudar a la programación o a la animación. Ayuda en animaciones o movimientos complejos.

Epipolo. Son los puntos de intersección de una recta con una esfera cuando la primera pasa por el centro de la segunda.

Escena. Es el conjunto de objetos 3D, luces, texturas y animaciones, es similar a las escenas cinematográficas.

Extrudir. Dar forma a una masa metálica, plástica, etc., haciéndola salir por una abertura especialmente dispuesta. En el ámbito del modelado por computadora ocurre cuando un polígono o conjunto de ellos cambia de posición siguiendo la trayectoria de su normal.

Fotograma. Imagen estática parte de una secuencia que en conjunto forman una animación o video.

Gráfico *raster*.Es una imagen digitalizada formada por una rejilla bidimensional uniforme de pixeles es llamado comúnmente Mapa de bits.

Infografía. Infografía es la técnica de elaboración de imágenes por computadora.

Malla de alambre. En el ámbito del modelado por computadora es la representación de objetos a través de sus aristas y vértices.

Modelo 3D. Figura tridimensional realizada con programas de modelado por computadora.

Motor de juegos. Es un programa que incluye las herramientas de programación necesarias para realizar un videojuego o un recorrido virtual. Algunas de estas herramientas son el motor de render, funciones para colisiones, control de cámaras, de periféricos, programación, etc.

NURBS. (Non Uniform Rational B-splines) es un modelo matemático para generar curvas y superficies.

Ortogonal. Que tiene sus ángulos rectos.

Polígono. Es una figura geométrica formada por segmentos consecutivos no alineados, llamados lados.

Realidad virtual. La realidad virtual es un medio compuesto por simulaciones de computadora interactivas que reaccionan a la posición y acciones del usuario y producen retroalimentación en uno o más sentidos, generando la sensación de estar inmerso o presente en una simulación

Render. Proceso por el cual los datos de geometría, textura, iluminación y efectos de cámara se traducen en una imagen digital.

Software. "El software no son sólo programas, sino todos los documentos asociados y la configuración de datos que se necesitan para hacer que estos programas operen de manera correcta. Por lo general, un sistema de software consiste en diversos programas independientes, archivos de configuración que se utilizan para ejecutar estos programas, un sistema de documentación que describe la estructura del sistema, la documentación para el usuario que explica cómo utilizar el sistema y sitios web que permitan a los usuarios descargar la información de productos recientes[55]".

Spline. Es una banda flexible que se utiliza para producir una curva suave que pasa por puntos concretos. El término curva con *spline* se refiere a cualquier curva compuesta por partes polinómicas que satisfacen condiciones de continuidad específicas en los límites de las mismas.

Superficie orgánica. Superficie orgánica se refiere a la superficie que no está compuesta por polígonos regulares y toma su nombre de las formas de la naturaleza que son más complejas

Vértice. Es el lugar geométrico donde concurren dos o más curvas, no tiene dimensión pero sí un lugar en el espacio.

[55] Ingeniería del software, Sommerville, Ian. 7ª Ed. Edit. Addison Wesley.

10. Bibliografía.

- Apuntes de álgebra lineal, Solar González Eduardo, Speziale de Guzmán Leda. 3ª Ed. Limusa, México 2001.

- Ingeniería del software, Sommerville, Ian. 7ª Ed. Edit. Addison Wesley, España 2005.

- Método de producción de contenido gráfico para tiempo real. Franco Serrano, Víctor Hugo. Depto. De Realidad Virtual DGSCA UNAM, México 2009.

- Gráficos por computadora con OpenGL, Hearn Donald. 3ª Ed. Prentice Hall, España 2006.

- 3D computer graphics. Buss Samuel. 1ª ed. Cambridge University Press, EE.UU. 2003.

- Computer graphics c version. Hearn, Donald. 2a ed, EE.UU. 2005.

- Técnicas y dispositivos de realidad virtual, dispositivos de entrada/salida. Toharia, Pablo. Universidad Rey Juan Carlos, España 2004.

- Introducción a la generación de imágenes estéreo. García, Marcos. Universidad Rey Juan Carlos, España 2008.

- Guide for development and construction of the experiencing pavilion. Boureau of Shanghai world expo coordination. 3a ed. China, 2009.

- Exportación de modelos para OSG, VRML y virtools. Franco Serrano, Víctor Hugo. Depto. de realidad virtual DGSCA, UNAM, México 2009.

- Enter@te en línea año 2, número 24, noviembre de 2003.Ramos Nava, María del Carmen. Dirección General de Servicios de Cómputo Académico, UNAM.
  http://www.enterate.unam.mx/Articulos/2003/noviembre/realivirt.htm

- Dispositivos hápticos. García, Marcos. Universidad Rey Juan Carlos, 2008.
  http://dac.escet.urjc.es/rvmaster/rvmaster/asignaturas/TemaV.Hapticos.pdf

- Modelado basado en imágenes. Mario Rodríguez Martín, 2009.
  http://www.slideshare.net/MarioRM/modelado-basado-en-imgenes

- Avendaño Martínez, José Carlos, Salgado Rodríguez, José Francisco, Valiente Gómez, Santiago Igor (asesor), "Reconstrucción de sitios arqueológicos: Teotihuacán", Facultad de Ingeniería, UNAM, México, 2004.

- Martínez Flores, Uriel Vladimir, Sanginés Huitrón, Manuel, Salgado Rodríguez, José Francisco (asesor). "Reconstrucción virtual de la zona arqueológica del Templo Mayor", Facultad de Ingeniería, UNAM, México, 2007.

- Delgadillo López, José Noé, González Rodríguez, Carlos Javier, Jacobo Romero, Miguel Ángel, Salgado Rodríguez, José Francisco (asesor). "Paseo virtual por la zona arqueológica de Xochicalco", Facultad de Ingeniería, UNAM, México, 2008.

- Figueroa Franco, José Luis, Garrido Lazcano, Genaro Andrés (asesor). "Paseo virtual por el palacio de Bellas Artes", Facultad de Ingeniería, UNAM, México, 2011.

- Vega Munguía, Rosy Yalim, Vega Munguía, Elio. "Metodología para generar un proyecto interactivo tridimensional: Paseo virtual por la zona arqueológica Tenochtitlán en 3D". FES Aragón, UNAM, México, 2009.

- Procesamiento y análisis digital de imágenes mediante dispositivos lógicos programables. Mendoza Manzano, Manuel Alejandro. Universidad Tecnológica de la mixteca. México 2009.

11. Índice de figuras.

Créditos de las figuras.

Figura 1. Gilles Tran. http://www.oyonale.com/modeles.php?lang=en&page=40

Figura 2. Render de una “Utah teapot”. Finlay McWalter.

Figuras 3 y 4. http://www.turbosquid.com
Figuras 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 30, 31, 32, 33, 34, 36, 37, 38 y 42. Gráficos por computadora con OpenGL, Hearn Donald. 3ª Ed. Prentice Hall.

Figuras 26, 27, 28, 29. Modelado basado en imágenes. Mario Rodríguez Martín. http://www.slideshare.net

Figura 35. Square picture.

Figuras 39 y 40. http://macedoniamagazine.frodrig.com

Figura 41. http://www.optimizacion3d.info

Figura 43. http://www.cybercollege.com

Figura 44. Max Edwin Wahyudi. http://www.3dm3.com

Figuras 45, 62. Dassault systèmes.

Figuras 46, 47, 48, 49, 50, 51, 52. Ingeniería del software, Sommerville, Ian. 7ª Ed. Edit. Addison Wesley.

Figuras 53, 55 Modelo del pabellón.

Figuras 54, 59. Eder Navarro.

Figuras 56 y 110. http://www.geniusnet.com

Figuras 57, 58, 60, 61, 63, 64, 65, 66, 67, 68, 69, 70, 71, 72, 73, 74, 75, 76, 77, 78, 79, 80, 81, 82, 83, 84, 85, 86, 87, 88, 89, 90, 91, 92, 93, 94, 95, 96, 97, 98, 99, 100, 101, 102, 103, 104, 105, 106, 107, 108, 109 y 111. José Felipe de Jesús Contreras Flores.