UNIVERSIDAD NACIONAL AUTONOMA DE MEXICO

FACULTAD DE ESTUDIOS SUPERIORES

ARAGON

CREACION DE ESPACIOS FUNCIONALES CON REALIDAD VIRTUAL

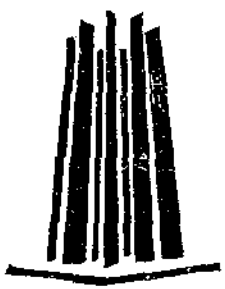
T E S I S

QUE PARA OBTENER EL TITULO DE:

INGENIERO EN COMPUTACION

P R E S E N T A :

ALFONSO ARTEAGA JORGE



DIRECTOR DE TESIS:
BIOL. JOSE LUIS VILLARREAL BENITEZ

MEXICO, D. F. 2005

Autorizo a la Dirección General de Bibliotecas de la UNAM a difundir en formato electrónico e impreso el contenido de mi trabajo recepcional.

NOMBRE: ALFONSO ORTEGA JORGE

FECHA: 4-JULIO-2005

FIRMA:

Agradecimientos:

A LA UNIVERSIDAD NÁCIONAL AUTÓNOMA DE MÉXICO:

Por haberme forjado en sus aulas e instalaciones como estudiante, deportista y ahora académico.

A MI DIRECTOR DE TESIS Biol. José Luis Villarreal Benítez:

Te agradezco enormemente la gran paciencia que me tuviste, los materiales proporcionados, las facilidades otorgadas para ingresar y utilizar el equipo del Departamento de Visualización, tus regaños, consejos y pláticas para la realización de este proyecto.... Gracias de verdad José Luis.

AL LA GENTE DEL DEPARTAMENTO DE VISUALIZACIÓN:

Lizbeth Heras Lara, Ricardo Mercado Mendoza, Juan Carlos Espino Velásquez, Omar Hernández Carapia, Yessica Ávila Sánchez que colaboraron conmigo siempre incondicionalmente... gracias.

A MI FAMILIA:

MI PADRE Lic. Alfonso Arteaga Tovar:

Gracias por confiar en mi, tu sabes que eres mi motivación y ejemplo a seguir. No lo hubiera logrado sin tus regaños y consejos... que más te puedo decir... TE DEDICO este logro y espero darte más satisfacciones.

Mamá, Rosaura, Isis, Brenda, Cristian, Deniss, Melissa ...por estar ahí siempre... ¡si se puede¡.

Brenda Hernández gracias por enderezar el rumbo y estar ahí, espero ser ejemplo para ti.

Francisco Lugo Díaz, Omar Lugo Díaz, Juan Carlos Guzmán, Roberto Monreal... gracias por ser mis amigos y apoyarme siempre.

Creación de espacios funcionales con realidad virtual.

Introducción.

Hoy en día las empresas dedicadas al ramo inmobiliario necesitan de nuevas herramientas que les ayuden a hacer más atractivos sus productos debido a la gran competencia del mercado y a que cada vez son más las empresas de este tipo. El presente trabajo pretende ofrecer una herramienta que resuelva estos problemas. Desde hace varios años se han explotado recursos como: comerciales en radio y televisión, anuncios publicitarios de todo tipo, exposición de productos, entre otros. Pero se puede observar que no todos los recursos brindan los mismo resultados, algunos tienen éxito y otros no, pero si volteamos a ver hacía la tecnología nos percataremos de que existe un medio más que crece a cada momento y a nivel mundial como nueva alternativa, hablo de: Internet.

En otro contexto también se observa que en los últimos años, se está confirmando otra área tecnológica muy importante en el cómputo gracias a sus avances y logros que ya está al alcance de usuarios comunes: la Realidad Virtual (RV) basada en Internet. Se ha demostrado que la Realidad Virtual permite el acceso público a los diferentes avances y aplicaciones en áreas como la enseñanza, la simulación, el comercio, la ingeniería, la industria, la física, el cine, la televisión, juegos, entre muchos otros y que dentro de las alternativas que existen para interactuar con la Realidad Virtual, la que se realiza por ordenador es la más accesible a los usuarios, ya que, elementos de hardware virtual como dataglove's (guantes de datos), cascos estereoscópicos, visores 3D y otros elementos de alto costo se pueden sustituir por una pantalla, un ratón y conexión a Internet, lo cual, representa hoy en día algo usual en cualquier equipo de cómputo.

De esta manera el presente trabajo pretende ser una herramienta que ayude a la creación de espacios funcionales, para lo cual, fue necesario buscar y experimentar entre la amplia gama de posibilidades con que se pudo realizar este trabajo, lo que no significa que todas eran las óptimas, hasta encontrar las herramientas idóneas del 3D en cuanto a Realidad Virtual para Internet se refiere (véase "Objetivos del proyecto"). Una vez que se encontraron dichos elementos se crearon los espacios siguiendo toda una metodología y se les dio la funcionalidad.

Se ha observado que empresas que cuentan con recursos de cómputo necesarios se han involucrado con ésta tecnología, creando sus sitios o páginas Web para ofertar sus productos. Aquí es donde se presenta la oportunidad de brindar herramientas que trabajan sobre Internet, que son interactivas en 3D y que representan una solución muy atractiva para cualquier consumidor. Esta herramienta es un espacio funcional desarrollado utilizando como medio el lenguaje de descripción de escenas VRML del que se hablará más adelante.

Este proyecto no sólo podrá ayudar a empresas inmobiliarias o de diseño, también a arquitectos, diseñadores y personas afines particulares que pretendan tener un prediseño de la organización de sus espacios. Para demostrar que la realidad virtual y en particular el lenguaje VRML (Virtual Reality Modeling Language) tiene una amplia gama de aplicaciones se desarrolló una aplicación, que consiste en un apartamento funcional, que representa una herramienta para tener un prediseño de la decoración y ubicación del mobiliario que lo conformará, ésto será una herramienta que ayudará a diseñar interactivamente. Se entiende por "funcional" aquella aplicación que permite la libre manipulación de los objetos que conforman el espacio.

El presente escrito es producto de la aplicación desarrollada como complemento de este trabajo y está debidamente fundamentado con sus respectivas direcciones de Internet y bibliografías para ampliar el panorama del entorno que puede cubrir un lenguaje como VRML. Todo en su conjunto conforma espacios 3D en los que un usuario es libre de agregar, eliminar, modificar y manipular el entorno interactuando con las diferentes opciones.

Se pretende que esta aplicación sea una aportación a la ingeniería y a la sociedad en el ámbito del diseño de espacios funcionales, esperando sean útiles y atractivos en un uso particular o masivo en Internet. El objetivo general de este trabajo es entonces: desarrollar herramientas para la creación de espacios funcionales utilizando la realidad virtual como medio para lograr este fin.

I. GENERACIÓN DE AMBIENTES VIRTUALES.

I.1. Historia del VRML.

En los años sesenta cuando comenzaba la revolución tecnológica, surgieron sistemas que podían ejecutar varios procesos en forma simultánea los cuales fueron llamados "sistemas multitareas". Estos sistemas fueron los que sentaron las bases para el nacimiento de la Realidad Virtual. Antes de su existencia, nadie había pensado en gráficas, interacción e idear formas para poner a trabajar a nuestros cinco sentidos al estar frente a una computadora. Fue un investigador del Instituto Tecnológico de Massachusetts llamado Ivan Sutherland, a quien le nació la inquietud de tomar a los sistemas multitareas como base y crear una aplicación en la que cualquier usuario pudiera tener interacción con la computadora. Creó entonces un programa que permitía realizar dibujos y diseños libremente con ayuda de un lápiz óptico, una barra de colores, iconos y ventanas [Mark Pesce, 1996].

Aunque hoy en día suene irrelevante, en su época fue algo nuevo y diferente a lo existente, Ivan Sutherland continuó trabajando este tipo de sistemas hasta lograr la invención de implementos como anteojos visores en 3D, procesadores de gráficas tridimensionales, entre otros, que sentaron las bases de un nuevo concepto tecnológico llamado: Realidad Virtual.

Una vez conocido este concepto, fue el Departamento de Defensa norteamericano el que inmediatamente comenzó a trabajar en el desarrollo de simuladores basados en los trabajos de Sutherland. En esta década, la RV aún no estaba al alcance de los usuarios comunes ni comerciales debido a que los implementos virtuales eran muy costosos y se necesitaban equipos de rápido procesamiento, por esto, la RV tuvo auge hasta principios de los ochenta, cuando comenzó la llamada "guerra de los procesadores" [Mark Pesce, 1996].

En la NASA también se conjuntaron las ideas de investigadores en un proyecto que pretendía crear interfaces gráficas para estaciones de trabajo. Para ello se creó un desplegador portátil con lentes de visión 3D compuesto por una pantalla de cristal líquido y los lentes, mientras que otros investigadores crearon un guante que detectaba y simulaba los movimientos de la mano; al cual llamaron "Data Glove". Con estos dos elementos, un usuario podía ver un objeto en 3D y simular tocarlo, a esto se le conoce hasta hoy como "inmersión" y fue un ingrediente que a nadie se le había ocurrido tomar en cuenta para crear interfaces [Mark Pesce, 1996]. Se estudiaba hacer más ameno y amigable a Internet con interfaces gráficas, ya que se proporcionaría interactividad con los sitios, se decía que una buena interfaz por sencilla que pareciera, representaría al Web en forma más natural.

Figura I.1.1. Utilización del DataGlove (Guante de datos) y un visor para poder interactuar en un mundo virtual.

Una vez analizado el nacimiento y objetivos de la RV, debemos retomar al VRML como un lenguaje que ha contribuido para hacer implantaciones 3D en Internet. La palabra clave seguía siendo "sensibilidad", ya que por primera vez se concluyó que un usuario se entendería mejor con una computadora si ésta representaba la información de manera sensible. El hecho era representar las cosas, la naturaleza, los lugares, las obras de arte de manera más parecida a como los humanos lo percibimos con nuestros cinco sentidos, se necesitaba de mucha inventiva y fue el desarrollador Mark Pesce quien después de observar que el Web permitía - por la estructura en que estaba hecho - modificaciones locales en cualquier máquina, porque fue construido en un código fuente fácil de copiar y compilar en una computadora, ideó un visualizador 3D básico en el que se pudiera navegar.

Con ayuda del investigador Anthony Parisi, un experto desarrollador quien entre sus logros más notables está el haber ayudado a la creación de Lotus 123. Trabajando conjuntamente, Pesce y Parisi dieron origen a dos tipos de software virtual, primero uno llamado *Reality Lab* y posteriormente uno llamado *Labyrinth*. Para 1994, Tim Berners-Lee creador del WWW, el HTTP y el HTML, se encargó de dar difusión a sus proyectos invitándolos a mostrarlos en la primera conferencia internacional sobre Internet, a celebrarse en la sede del CERN (Centro Europeo para el estudio de Partículas Físicas), en Ginebra Suiza [Mark Pesce, 1996].

En la citada conferencia David Ragget uno de los principales impulsores del sistema HTML junto con Tim Berners-Lee presentaron un excelente trabajo conceptualizado como "Personas de intereses afines", este trabajo era un sitio de Internet con sesiones donde se podían intercomunicar persona con intereses semejantes y charlar sobre algún tema; era algo parecido a lo que hoy llamamos "news group"o chat. David Ragget decidió que el acrónimo que mejor representaba a esta aplicación era: VRML. Mark Pesce y Anthony Parisí presentaron algo que en este 2004 podríamos ver como algo común: una banana cibernética, cibernética porque se formaba con información digital en tres dimensiones (algo poco visto en ese entonces) y que representaba un enlace a un sitio que trataba sobre el espacio cibernético al hacer clic sobre ella.

El evento fue un éxito y Netscape, Silicon Graphics (ahora SGI), WIRED, entre otros, apoyaron esta tecnología y la comunidad del Web lanzó un llamado en el que se invitó a todos los Internautas a colaborar en el desarrollo de VRML, nació así *Labyrinth*. Para seguir con el proyecto se hicieron varios análisis después de los cuales se concluyó que lo más apropiado, sería adaptar un lenguaje existente, que ya haya sido diseñado por expertos en gráficas y después de una intensa búsqueda entre varios candidatos, el consejo encargado del proyecto se inclinó hacia las cualidades de *Open Inventor* cuya biblioteca de programación fue elaborada por Silicon Graphics y utilizada para crear prototipos rápidos de aplicaciones en tercera dimensión [Mark Pesce, 1996]. *Open Inventor* contaba con un lenguaje de descripción de escenas, un formato con los elementos suficientes para crear aplicaciones de calidad comercial y con muchos años de respaldo e inversión. A mediados de 1994 se dió a conocer al público VRML 1.0. Para lograr que este lenguaje pudiera ser entendible en diferentes máquinas, se necesitaba algo similar al HTML pero en ese tiempo pensar en solucionar este problema en poco tiempo era una alternativa totalmente nula.

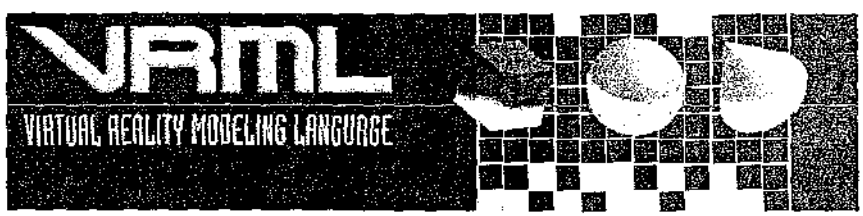


Figura I.1.2. Logo de la versión 1.0 de VRML.

Aún con este problema, los usuarios de VRML explotaron las ventajas del Web que aún no era interactivo. Para finales de 1994 se realizó la segunda conferencia mundial sobre el Web, Parisi y Gavin Bell presentaron una nueva herramienta, con especificaciones de dibujo para el VRML 1.0. VRML trabajaba ya sobre las bases de *Open Inventor*, Netscape Communications y personal del CERN se interesaron mucho más ya que sabían que tenían equipos que trabajaban bajo esta plataforma. Esto dió origen a un analizador sintáctico de VRML llamado QLiv, que fue la pauta para originar un visualizador de VRML para Internet [Marck Pesce, 1996].

En 1995 Silicon Graphics colaboró en la realización del visualizador y la empresa TGS (Template Graphics Software) realizó la conversión entre plataformas, dando origen a *Web Space*, primer visualizador formal para VRML. Para esta operación se tuvo un patrocinio de varios millones de dólares aportados por Silicon Graphics, Ford, Netscape y Digital, entre otras, realmente una inversión inteligente, ya que estas empresas sabían que con esta tecnología cualquier cosa se vendería sola.

La primera publicación de la existencia formal del VRML y del primer visualizador se dió en abril de 1995, aquella fecha fue llamada el "Día Cero" debido a que la comunidad estadounidense entraría en contacto por primera vez con éste lenguaje. En la tercera conferencia mundial sobre el Web celebrada en Alemania, el VRML se mencionó como un elemento desarrollado en forma y no como prototipo, se aclararon dudas y se explicó como cualquier usuario o empresa podría implantarlo.

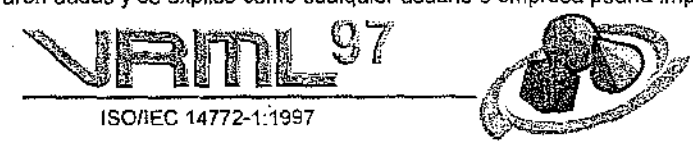


Figura I.1.3. Logo de VRML 97, la mejor versión que ha tenido.

Después de estas colaboraciones, en el primer mes fueron creados al menos 50 sitios VRML y se establecieron un mínimo de 100 sitios en la red. Desde su nacimiento en febrero de 1994, hasta su aceptación por la comunidad Web en abril de 1995, el VRML ha estado en constante preparación y ascenso por el bien de la interactividad.

A VRML se le ha denominado "La Realidad Virtual en Internet", debido a que con su potencial podemos interactuar en tiempo real en espacios y ambientes que no existen en el mundo real sin la necesidad de dispositivos adicionales a la computadora. En Internet surgió VRML, como un estándar para la creación de objetos en 3D que permite combinarlos en escenas y mundos virtuales. Se utiliza para representar simulaciones interactivas, que incorporan animaciones, contenidos multimedia y la participación de múltiples usuarios en tiempo real.

Cabe mencionar que en 1994, se creó el VRML mailing list [5] el cual aún existe, donde se hace un llamado abierto para que todo el público pueda hacer propuestas para una especificación formal de 3D en el WWW. Algunas de sus características son que los productos creados con VRML permiten al usuario interactuar con ellos y observarlos desde diferentes ángulos, integrando imágenes y posiblemente audio. VRML, disminuye el tiempo de transmisión.

En la conferencia Siggraph de 1996, la propuesta Moving Worlds (Mundos en Movimiento) de Silicon Graphics fue ratificada por el VAG (VRML Architecture Group) como la especificación oficial VRML 2.0. Esta nueva versión es mucho más sofisticada que su predecesora y en la cual destacan los siguientes aspectos:

- Posibilidad de especificar comportamientos para los objetos, ya sea usando el propio lenguaje VRML o mediante scripts en lenguajes externos (Java script, Java, Visual Basic, etc.), los cuales no están limitados por la especificación.

- Posibilidad de interacción con el usuario mediante la definición de una serie de sensores de posición, de contacto, de colisión, etc. La información registrada por estos sensores es enviada a los diferentes objetos que componen el mundo virtual y en función de los valores recibidos, cada objeto virtual actuará en consecuencia.

- Finalmente, este lenguaje de descripción de escenas tridimensionales ha sido ampliado significativamente, posibilitando efectos de fondo, sonidos tridimensionales, niebla, etc.

La última versión (VRML 2.0) aún vigente, fué utilizada en el presente trabajo y cualquier persona que quiera convertirse en diseñador de mundos VRML encontrará gran cantidad de información y muy completa (diferente a los manuales comunes de este lenguaje que se encuentran en internet), en un repositorio [6], creado por gente especializada en el tema y que han sido partícipes de su desarrollo, el cual entre otras herramientas cuenta con un manual en línea con todas las especificaciones de los nodos que maneja VRML desde su versión de 1997 hasta la fecha.

En resumen tenemos la siguiente cronología:

- 1994 La primera propuesta pública sobre la posibilidad de crear un lenguaje que permitiese recrear mundos virtuales, surge en la conferencia sobre WWW que se celebró en Ginebra en mayo de la mano de Mark Pesce y Anthony Parisi.

- Ante el amplio apoyo recibido en dicha conferencia y junto a la ayuda de la revista Wired, se estableció una lista de correo con el objetivo de conseguir una primera especificación del lenguaje en cinco meses, de forma que se pudiese presentar en la segunda conferencia Web en octubre de ese mismo año.

- Tras una serie de debates en los que se discutieron diferentes propuestas, la alternativa presentada por Silicon Graphics fue la que consiguió un mayor número de votos, convirtiéndose de este modo en la base del nuevo estándar. Esta propuesta consistía en utilizar como punto de partida el lenguaje en el que estaba basado el programa *Inventor*, producto de dicha compañía.

- En octubre de 1994 se presenta la especificación de VRML 1.0.

- Después de la conferencia de octubre, se fundó el VRML Architecture Group (Grupo para la Arquitectura de VRML), también conocido como VAG. Este grupo tenía la misión de ayudar en la clarificación e implantación de la especificación inicial de este nuevo lenguaje. Con posterioridad, este organismo ha sido sustituido por el Consorcio VRML, entre cuyos miembros se encuentran Netscape, Microsoft, IBM, Silicon Graphics, entre otros.

- 1995, Silicon Graphics pone a disposición del dominio público QvLib, el cual se convirtió en el primer parser (intérprete) VRML capaz de traducir el texto de una escena virtual a un formato entendible por un navegador. Posteriormente apareció WebSpace, el primer navegador capaz de leer e interpretar todo el estándar VRML.

- 1997, Con la colaboración de distintos desarrolladores, sgi crea a través del Web, la versión VRML 2.0
- Enero-1999 -- Se termina la Versión 2.19 de VRML con mejoras en EAI y algunas actualizaciones de la documentación.
- Junio-1999 -- La Versión 2.20 alpha1 esta lista, aunque todavía tiene algunos errores al trabajar. Se tiene soporte a EXTERNPROTO, ayuda mejorada en Javascript y otros cambios.
- Octubre-1999 -- La Versión 2.21 alpha2 esta lista. Se trabajó en puntos de vista animados y se incluyeron notas de instalación.
- Enero-2000 -- Surge la Versión 2.22 que cuenta con una ayuda mucho mejor en la textura y corrección de errores, además de mejores instrucciones de instalación en Linux RedHat.
- Mayo-2000 -- Se lanza la Versión 2.23, son depurados los mensajes de error y carga.
- Julio-2000 -- La Versión 2.25 snapshot1 tiene mejoras en la codificación de OpenGL.
- Septiembre-2000 -- La Versión 2.26 tiene muchos cambios como la corrección de los errores que presentaban los nodos pixelTexture y texture. También se trabaja en los problemas con Java e interacción de Netscape.
- Marzo-2001 -- Se lanza la Versión 2.28. Se corrigieron errores de velocidad y representación utilizando archivos wrl en las pruebas.
- Agosto-2001 -- Se prepara X3D un editor mejorado para gráficos 3D compatible con VRML en su Versión 2.30.
- Enero-2002 -- Versión 2.30 de X3D pre-lanzamiento en sitios web.
- Mayo-2002 -- Se tiene la Versión 2.32 en la que se trabaja la mejora de los nodos NormalInterpolator, Fog, normales lisas sobre extrusiones y de la estructura de OSX.
- Julio-2002 -- En la Versión 2.33 se crea código fuente y formato para los sistemas de RedHat 7.2 y 7.3.
- Agosto-2002 -- Se lanza versión previa del editor X3D y los comentarios sobre este suceso son de bienvenida.

Figura 1.1.4. Logo distintivo del nuevo editor de gráficos 3D y VRML.

Presente y futuro de VRML.

El lenguaje VRML ha pasado por diferentes etapas desde su creación en 1994, como lo son el haber sido un lenguaje experimental ambicioso en su primera versión VRML 1.0, aunque para esta primera etapa aún no lograba muchas cosas, la gente relacionada con el leguaje sabia que podían hacerlo crecer de manera considerable. Ya para 1996 el lenguaje habia evolucionado de manera que sus componentes eran la mayoría de lo que se conoce como especificaciones de VRML '97 y de la misma manera llamándolas la versión 2.0 aún vigente.

El presente de VRML es el producto de la evolución de este poderoso lenguaje para Internet. Es el consorcio Web3D quien trabaja en esta actualización denominada "X3D", para ello laboran en el desarrollo sus dos grupos: "Grupo de trabajo browser" y "Grupo de trabajo X3D". X3D es una nueva especificación que debe reunir las necesidades de toda la comunidad 3D los cuales son:

- Compatibilidad con el existente contenido de VRML, browsers y herramientas.
- Mecanismo de extensión que permita introducir nuevas características, vista rápida de avances y adopción formal de esas extensiones.
- Perfil completo VRML para soportar contenidos existentes.
- Soporte para otras codificaciones incluida XML para una firme integración con las tecnologías y herramientas Web.

La nueva especificación X3D maneja perfiles (colección de componentes), dos ejemplos de estos son, el pequeño núcleo para soportar una simple animación no interactiva y el perfil base compatible con VRML que soporta mundos totalmente interactivos. X3D permite extender componentes individuales o modificarlos agregando nuevos "niveles", también permite agregar nuevos componentes para introducir nuevas características como "streaming", es decir, que los avances que tenga X3D pueden moverse rápidamente porque el desarrollo en un área no retarda la especificación en conjunto.

X3D el presente de VRML.

El extensible 3D es llamado así porque puede usarse en una pequeña y eficiente animación 3D o para soportar lo último en extensiones streaming o de renderizado. Soporta codificaciones múltiples y API's, para que pueda integrarse fácilmente con browsers Web a través de XML o con otras aplicaciones. Puede también ser soportado fácilmente por herramientas de creación, browsers propietarios y otras aplicaciones 3D. Reemplaza a VRML pero al mismo tiempo proporciona compatibilidad con los contenidos y visualizadores existentes de VRML. El contenido actual de VRML podrá ser usado sin modificación en cualquier browser X3D.

La especificación completa estará disponible en www.web3d.org. Un ejemplo del funcionamiento de X3D se puede encontrar en el sitio http://www.openworlds.com X3D como tal, es VRML 97 en componentes, con un mecanismo para agregar nuevos componentes y extenderse más allá de la funcionalidad del VRML 97. Tenemos así que para convertir un archivo de VRML en uno X3D solo se necesita agregar en el encabezado del archivo el comentario #X3D profile : base.

Ventajas de X3D

Cualquier usuario o empresa que tenga X3D se beneficiará con ventajas como:

- El contenido de X3D es modular y reutilizable, ahorrando tiempo de desarrollo y dinero.
- X3D influye en el contenido y herramientas de VRML.
- Programas que ya exportan VRML como 3Dmax son compatibles con X3D.
- X3D soporta las codificaciones XML alternativas para la integración con otras tecnologías Web.

XML característica técnica de X3D

XML se adopto como una sintaxis para resolver varios problemas reales:

La rehospeabilidad, a pesar de que la sintaxis de VRML es similar a la de Open Inventor, en la cual esta basada, la sintaxis dominante de uso mundial es XML ya que el "marcado" ha demostrado ser la mejor solución a los problemas de los largos ciclos de vida de archivos de datos y rehosting.

Páginas de integración, las páginas XML basadas en integración van directamente al problema de mantener un sistema más simple para que más personas puedan desarrollarlas en cuanto a contenido e implementación.

Integración con la próxima generación web, ya que los miembros del Consorcio Web están pendientes de las nuevas versiones de los navegadores Netscape y Explorer que integran funciones XML y se debe estar integrado.

El Grupo de Trabajo X3D trabaja en una DTD (Definiciones de Tipo de Documento) para que X3D tenga una a una las correspondencias con los nodos y campos VRML. Los autores usan estas etiquetas definidas y por lo pronto no necesitaran desarrollar su propio DTD. Trabajan en construir traductores para convertir archivos VRML en archivos X3D para que cualquiera de las herramientas de modelado VRML puedan seguir usándose.

En que esta hecho X3D

X3D no está escrito en Java como muchos creen, X3D es una especificación gráfica de escenas 3D. Esta especificación puede ejecutarse por cualquiera de los varios lenguajes, incluyendo a C, C++, Java, etc. Existen muestras de implementaciones abiertamente disponibles que permitiran escribir en Java usando Java3D y en C++ usando Contact de Blaxxun, otras implementaciones pueden incluir Shout (Java puro) y DraW (Farenheit Scene Graphic), lo cual. No significa que X3D necesite solo ser escrito en Java o C++.

Aseveraciones sobre VRML Y X3D

Se ha rumorado que habrá nodos de VRML que se eliminarán en X3D, pero este hecho es malinterpretado ya que en realidad probablemente estos nodos no estarán en el núcleo pero serán implementados en una extensión como la extensión VRML 97. La razón es que Box, Sphere, Cone ElevationGrid, etc. No deberían incluirse en el núcleo porque pueden derivarse fácilmente de primitivos (bajo nivel) como el IndexedFaceSet. Este hecho simplifica el núcleo haciéndolo más pequeño, más fácil de implementar y mantener.

La aseveración de que X3D es una sustitución de VRML afectuosamente llamada VRML Lite es falsa ya que la idea es solo reducir la funcionalidad no esencial en los browsers actuales, por lo tanto, X3D no sustituye funcionalidad, la idea total del núcleo DE X3D es que puede extenderse para demostrar funcionalidad extensa.

En cuanto a compatibilidad, X3D podrá reconocer archivos VRML, pero de forma contraria un visualizador de VRML no puede sencillamente interpretar un archivo creado en X3D, debido a que X3D usará XML como sintaxis para su formato de archivo. Como un visualizador de VRML no tiene un parser XML, no puede trabajar con contenido X3D. Se deberá usar un convertidor , así como actualmente se convierten los archivos de VRML 1.0 a archivos VRML 97.

Algunos sitios relacionados:

http://www.w3.org/XML
http://www.xml.com
http://www.oasis-open.org/cover/xml.html
http://xml.about.com

Alternativas

Es sabida la existencia de aplicaciones alternativas para realizar desarrollos 3D, entre los más conocidos se encuentra Cult 3D, Anark, CoolFusion, entre otros.

Cult3D permite a usuarios crear objetos interactivos disponibles hoy en día en internet. Los objetos creados con Cult 3D tienen gran transparencia, reflectividad, sistemas de partícula, sombreo pong, movimiento y sonido, demostrando el objeto como realmente es hasta en los detalles más pequeños. Los objetos de Cult3D responden al clic del ratón, permitiendo a los usuarios hacer zoom y rotación del objeto además de interacción con los componentes, creando una experiencia multisensorial. Cult3D puede hacer algo tan complejo como un ayudante digital personal (PDA), o tan único como la cabeza humana.

Los objetos de Cult3D crean una comercialización dinámica en el Web, presentaciones únicas del producto y experiencias interactivas.

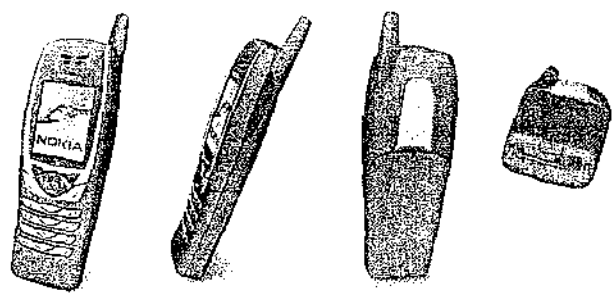


Figura I.1.5. Movimientos interactivos utilizando el visualizador Cult 3D.

Anark es una aplicación que proporciona incomparables soluciones interactivas en la visualización de un producto que utiliza contenido 3D a través del diseño, la ingeniería, entrenamiento y funciones de productos complejos. Analizando las aplicaciones que genera se puede deducir que su aplicación va más enfocada a uso empresarial que a usuarios comunes. Anark ofrece usos interactivos altamente realistas y la visualización del producto en forma dinámica haciendo el proceso de ventas por Internet un proceso más realista del producto antes de que lo compren.

Independientemente de lo que aplicaciones de este tipo pueden hacer, VRML se ha convertido en un superconjunto de lenguajes integrados lo cual permite a un desarrollador, implementar o modificar un objeto 3D a fondo, es decir, los parámetros son código abierto, Cult 3D no funciona de esta manera, ni Maya ni Anark ya que son estructuras predefinidas en las que solo se puede hacer lo que permite la aplicación.

Figura 1.1.6. Logo de Anark.

Algunos sitios donde se pueden ver alternativas de aplicaciones 3D se muestran continuación, dejando claro que para visualizar objetos 3D de estas paginas se debe descargar su respectivo cliente o visualizador.

- www.3d-test.com/3dsites
- www.cult3d.com
- www.anark.com

I.2. Dispositivos de Realidad Virtual.

En el mundo de la captura de movimiento encontramos a empresas dedicadas a construir cámaras y aditamentos como sensores para capturar los movimientos de una persona para posteriormente procesarlos por computadora. Las cámaras usadas con este fin tienen una resolución de hasta1.3 Megapixeles y una velocidad de captura mayor a los 1000 frames por segundo.

Motion Capture

Un sistema de este tipo captura los movimientos completos del cuerpo y rasgos faciales con gran exactitud y después los exporta a un software en el que se trabaja la animación. Una de las empresas más conocidas en la construcción de estos sistemas es "Vicon" [28].

Figura I.2.1. La captura de movimiento como se conoce a una serie de dispositivos y técnicas que permiten capturar las coordenadas en tres dimensiones y durante una secuencia de tiempo, de un conjunto de puntos marcados en un personaje real ó físico (generalmente un actor), para transferírselos a un personaje virtual. Estos dispositivos son ópticos, electromecánicos, infrarrojos y requieren de diversas técnicas y software para lograr la tarea. *Fotos: Biól. José Luis Villarreal Benítez.*

Brazos Digitalizadores

Actualmente se dispone de muchos dispositivos para capturar las geometrías y las texturas de objetos reales para incorporarlos a los mundos virtuales. En la figura I.2.2 se pueden apreciar dispositivos de brazos digitalizadores 3D, que inclusive se acoplan con cascos o lentes estereoscópicos para capturar o interactuar, ahora con objetos virtuales.

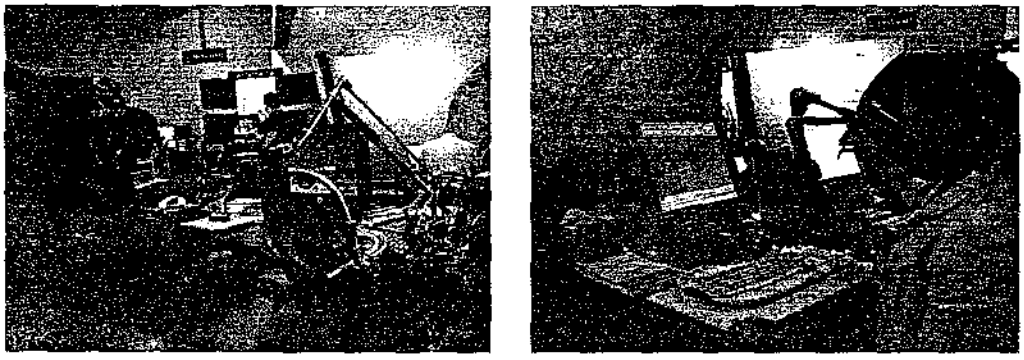


Figura I.2.2. Brazos digitalizadores 3D, creados específicamente para tomar los movimientos voluntarios de un usuario y emularlos digitalmente en una pantalla de computadora. *Fotos: Biól. José Luis Villarreal Benítez.*

Omnipantallas

Por otra parte, las pantallas parabólicas son sistemas de inmersión personales que permiten explorar mundos virtuales calculados en computadoras PC, usando un proyector con un lente especial y las pantallas cóncavas semiesféricas.

Figura I.2.3. Pantallas cóncavas en las cuales se puede observar un mayor campo de visión e inmersión de un mundo creado virtualmente por computadora. *Fotos: Biól. José Luis Villarreal Benítez.*

VideoWall

Las tarjetas gráficas de última generación, permiten controlar el despliegue de hasta cuatro monitores; permitiendo ampliar el ancho de banda visual. En la fotografía se aprecia el despliegue de un mundo virtual ampliado con tres monitores.

Figura I.2.4. Despliegue ampliado con tres monitores gracias a una tarjeta gráfica que realiza la escala y multiplicación de una imagen hacia tres destinatarios. *Fotos: Biól. José Luis Villarreal Benítez.*

Visionarium

Las pantallas curvas de proyección (generalmente 160°), permiten la sensación de inmersión tridimensional para grupos amplios de personas. En la imagen se ve la visualización de un fluido en un "Visionarium".

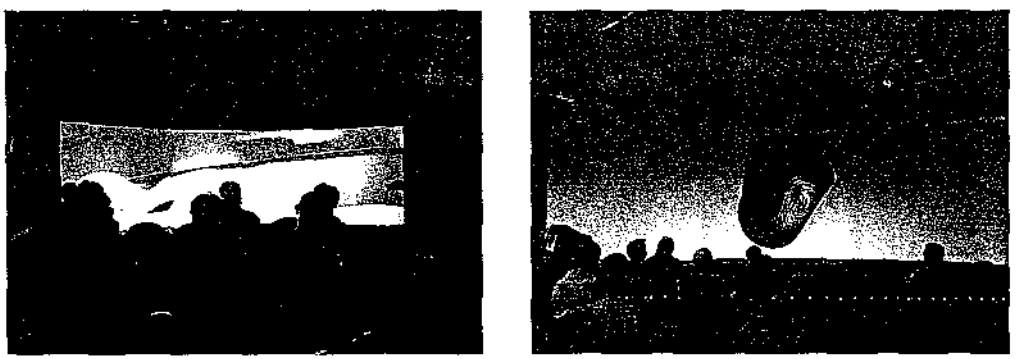


Figura I.2.5. Ejemplo de grandes pantallas curvas que al igual que las semiesféricas fueron creadas con este aspecto para dar mayor realismo debido al ángulo con que el ojo humano percibe las imágenes. *Fotos: Biól. José Luis Villarreal Benítez.*

InmersaDesk

Los escritorios con despliegue por proyección son una herramienta muy útil para el trabajo en grupos pequeños de personas. Estos sistemas van desde simples monitores de retroproyección montados en una mesa, hasta sistemas estereoscópicos con sensores para detectar la posición de la cabeza para corregir los despliegues, ajustándolos a la perspectiva del espectador. Algunos cuentan con monitores interactivos (touch screen) y mouses láseres o 3D (6 grados de libertad).

Figura I.2.6. Estos escritorios fueron creados para simular escritorios virtuales de trabajo con características como: despliegue por proyección, pantalla plana ajustable a la vista del espectador, detectores de posición, alta resolución entre otros. *Fotos: Biól. José Luis Villarreal Benítez.*

Reality Center

La presentación de nuevos diseños y de recorridos virtuales de proyectos se consigue con mucho realismo en los Centros de Realidad Virtual o Reality Center (como se les ha proporcionado en el sector). Estos Centros de Realidad Virtual son sistemas de despliegue amplios (generalmente por retrospección) montados en paneles de exhibición, algunos incluyen estereoscopia a través de lentes o de proyectores estereoscópicos. Aunque preferentemente usan un sistema de estereoscopia intermedio que permite que si el observador tiene lentes estereoscópicos, vea en 3D; pero si no cuenta con dicho dispositivo, aún puede apreciar el escenario. Generalmente este despliegue es en escala 1:1 y con una perspectiva natural, para lograr la sensación de inmersión y realismo.

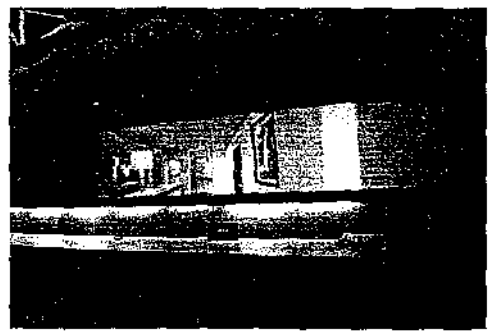


Figura I.2.7. Reality Center, en el cual se observa el recorrido virtual por un pasillo, el espectador puede percibir el realismo de la escena al ser esta en escala 1:1. *Fotos: Biól. José Luis Villarreal Benítez.*

Dispositivos de Despliegue

Los dispositivos de despliegue que producen la ilusión óptica de imágenes tridimensionales, requieren que un ojo reciba una imagen ligeramente diferente al otro; simulando el ángulo de diferencia con que ve cada ojo. La perspectiva de estas imágenes depende de la posición de la cabeza con respecto al objeto 3D. Así que es necesario determinar la distancia de la cabeza al objeto y los ángulos o vista que se tiene del objeto. De tal forma que si se mueve el objeto se recalculan las coordenadas de pantalla (como en cualquier otro dispositivo) pero si se mueve el observador, también será necesario recalcular las dos imágenes que recibe este. Así mismo, si se tienen varios observadores, es necesario dibujar diferentes pares estereoscópicos para cada quien.

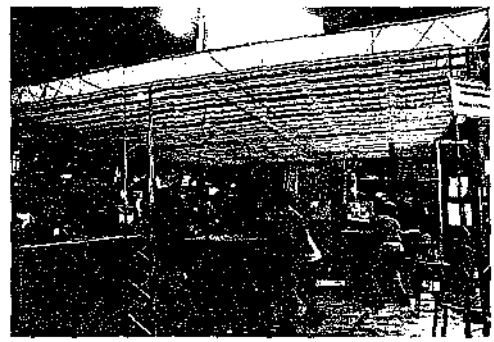


Figura I.2.8. En la imagen se aprecia una estructura que soporta en su techo una malla sensora infrarroja que realiza un "traking" sobre las cabezas de los espectadores. *Fotos: Biól. José Luis Villarreal Benítez.*

Casco Virtual

Los cascos de Realidad Virtual permiten sensaciones de inmersión en mundos tridimensionales. Los nuevos sistemas pueden hacer coincidir a diferentes personas en el mismo entorno virtual permitiendo la interacción entre las personas y de estas con el mundo virtual.

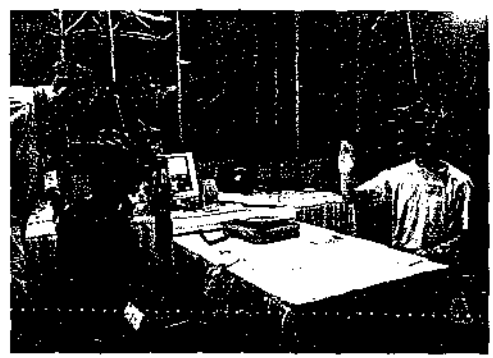


Figura I.2.9. En la figura se puede observar a dos personas compartiendo un mismo escenario ayudados por los cascos virtuales. *Fotos: Biól. José Luis Villarreal Benítez.*

Realidad Aumentada

La Realidad Aumentada permite obtener información del entorno (real o virtual) a través de sensores y visión por computadora, mientras presenta esta información en un mundo virtual al integrar la información textual, sonora o gráfica, al entorno de despliegue.

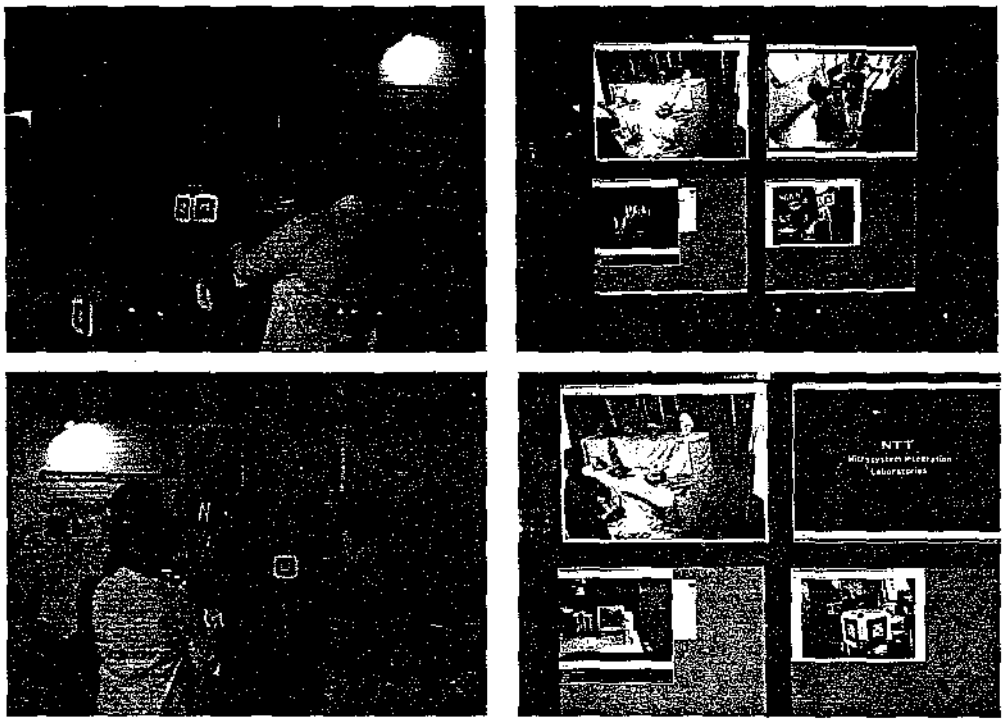


Figura I.2.10. En las fotografías se aprecia un sistema que interpreta símbolos captados previamente con una videocámara para después desplegarlos como máscaras en una pantalla portátil. *Fotos: Biól. José Luis Villarreal Benítez.*

I.3. Creación de ambientes virtuales.

La gran mayoría de las personas ha visto y disfrutado de las imágenes creadas en tercera dimensión, pudo haber sido por medio de la computadora o en la televisión. Muchas de estas personas incluyéndome, alguna vez nos preguntamos cuál es el elemento que permite la creación de éstas. En el mundo de las gráficas 3D la computadora usa *puntos* y *enlaces* que los hacen relacionarse para crear la estructura de los objetos, ya que todos los objetos tienen una estructura que les da forma.

Nube de puntos

Al conjunto de puntos que conforman a un objeto se les llama *nube de puntos* (figura I.3.1) ya que al eliminar las líneas que los conectan, la resultante es parecida a una nube en forma de puntos flotando en el espacio. Después de esto, la computadora une los puntos para crear la estructura que le dará forma al objeto, con esto, la estructura toma una forma más clara, aunque aún no se vería muy real debido a que en su interior estaría vacía, a esta etapa se le llama etapa de *delineado* (figura I.3.2).

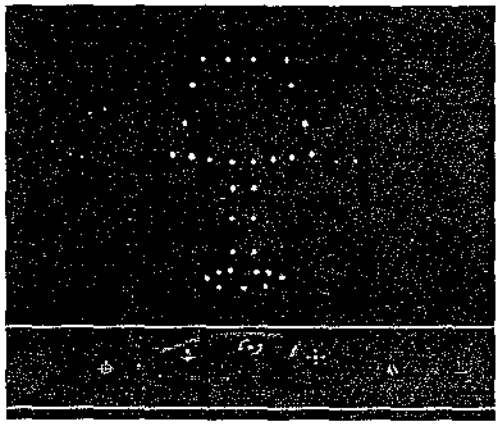


Figura I.3.1. Nube de puntos representando una lámpara

Polígonos en 3D

Son representaciones necesarias en la creación de objetos 3D. Ya se mencionó que los objetos se forman partiendo de una nube de puntos, los cuales se unen mediante líneas y al formar la estructura se cubren con una superficie. Estas superficies están compuestas por polígonos. Un polígono es la unidad básica de construcción del 3D y puede definirse como una superficie plana e infinitamente delgada. En graficación por computadora, el polígono favorito es el triángulo por ser el polígono más sencillo y aunque para estas máquinas es muy fácil asignarle color de relleno a este polígono, no lo es para uno de tipo irregular, esto nos lleva a que cuando se dibuja un polígono irregular en 3D con muchas caras, la computadora lo descompone en varios triángulos para trabajar más fácil con él, como se observa en la figura I.3.2.

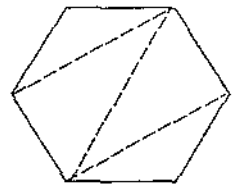


Figura I.3.2. Polígono dividido en cuatro triángulos regulares.

Hasta este momento sabemos como se crea un objeto a partir de la unión de puntos, estructuras y superficies, para después iluminarlo, ahora supongamos que sólo queremos ver la estructura del objeto, es decir, verlo en segundo plano, estaremos realizando lo que se conoce como *delineado o Wireframe*. La computadora usa líneas para conectar la nube de puntos que componen su estructura y eso será lo único que se desplegará.

Las computadoras hacen acabados de un objeto delineándolos con gran rapidez, por esto, usa este método para representar los modelos tridimensionales sin tener que esperar demasiado. Así tenemos que una estructura delineada por computadora y en esté caso VRML, tiene la apariencia mostrada en la siguiente figura:

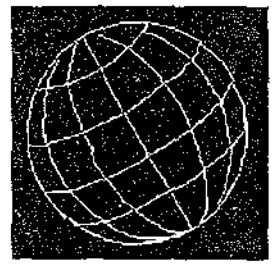


Figura I.3.3. Estructura de un objeto en segundo plano.

Delineado o Wireframe

A la etapa de delineado también se le puede asociar con el modelado "Wireframe". Este modelado consiste en líneas y arcos que se conectan a puntos para expresar la forma, pero no expresan sólido ni vacío. Estas líneas definen los límites de la forma, pero requieren más información que un dibujo sólo para poder entender la forma.

Los modelos de wireframe se usan por lo general como ayudas cuando se construyen modelos tales como senderos, o cuando se barren o se trazan objetos. Se pueden crear modelos de wireframe a partir de objetos 2D (planos) para después llevarlos a cualquier lugar dentro de un espacio 3D. Un ejemplo se observa en la escena 1.3.4.

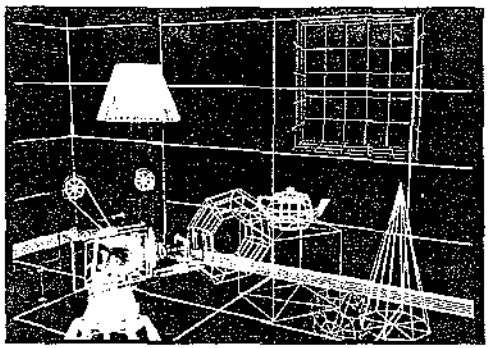


Figura I.3.4. Etapa de delineado de los objetos de una escena 3D ("Shutterburg". Pixar 1990).

Una vez que se tiene la estructura, necesitamos una superficie que rellene al objeto, la cual, puede tener diferentes calidades, dependiendo de que tipo de imagen se use y la resolución de la misma. Por otra parte, la computadora debe iluminar sus objetos por medio de una luz que ella misma debe generar para dar la impresión más real de lo que se esta viendo (figura I.3.5).

Sombreado

Cuando un objeto ya esta iluminado se le debe asignar un *sombreado,* cada superficie que cubre al objeto reflejará la luz de diferente manera y por tanto la computadora debe calcular este sombreado (figura I.3.6). Conjuntando estos elementos veremos que el objeto gradualmente toma una apariencia cada vez más real; es lógico observar que el grado de realismo que se le impregna a un objeto es proporcional a la cantidad de trabajo por computadora que se le dedique.

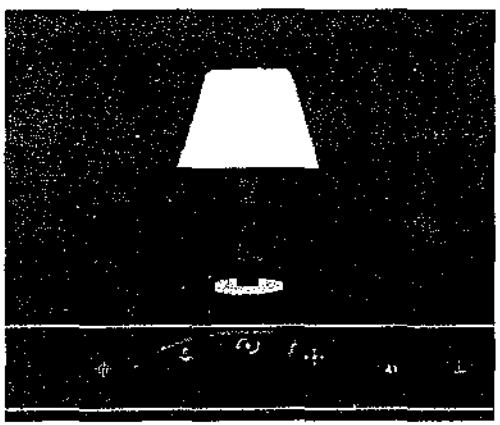


Figura I.3.5. Etapa de iluminación del objeto.

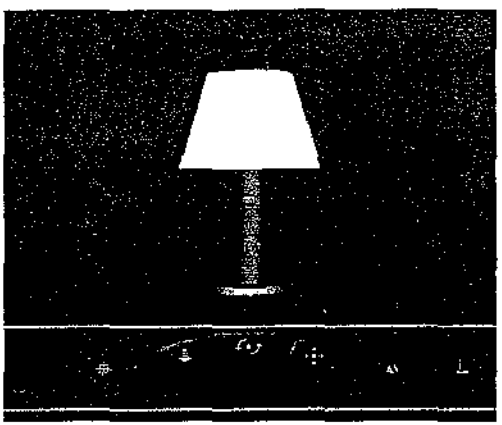


Figura I.3.6. Sombreado del objeto para darle mayor realismo.

Los objetos sólidos se componen - como ya se vio - de varias caras poligonales, la normal de estas caras determina como se refleja la luz en las mismas. La computadora trata de calcular la cantidad de luz reflejada en las caras del objeto, la cual, proviene de las fuentes luminosas presentes en la escena tridimensional, este proceso se conoce como *sombreado.* Existen diferentes tipos de cálculos que permiten establecer el sombreado de un objeto, algunos de ellos son muy simples y otros son complicados y lentos.

Sombreado plano

Es el más simple y se basa en la normal, donde la normal es el rayo luminoso que se proyecta hacia el exterior de la cara del polígono. En este caso, esta cara tendrá una apariencia muy brillante porque reflejará toda la luz que emite la fuente. Por otra parte, si el polígono cambia su ángulo oblícuo (como la luz del sol que llega a la Tierra), el objeto reflejará una menor cantidad de luz y será menos brillante.

El sombreado plano, aunque es muy rápido crea una apariencia artificial en los objetos. Este método de sombreado es el menos costoso computacionalmente. Cada nivel progresivo de sombreado aumenta el uso de la capacidad matemática de la computadora para crear una imagen más tersa.

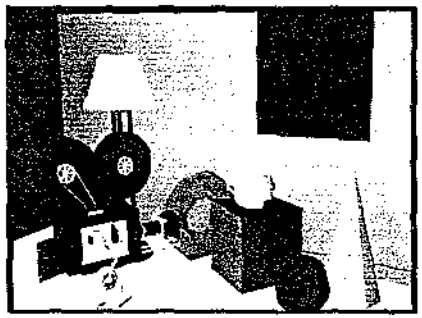


Figura I.3.7. Escena ("Shutterburg". Pixar 1990) renderizada con sombreado plano. El contorno de cada polígono es claramente visible.

Sombreado Gouraud

Aquí se usa la normal de la cara del primer polígono y la normal de las caras principales en los polígonos contiguos para establecer un sombreado promedio entre todas las caras. Este trabajo requiere de muchos cálculos matemáticos, por esta razón tarda en promedio diez veces más que el sombreado plano, pero proporciona una apariencia más realista en los objetos, aunque en ocasiones dependiendo de su complejidad presentaran irregularidades como franjas, esto se debe a una falla en la precisión de los cálculos matemáticos que implica el proceso. Aunque el uso de este sombreado es muy usado en el medio computacional, también genera una apariencia en cierto modo artificial.

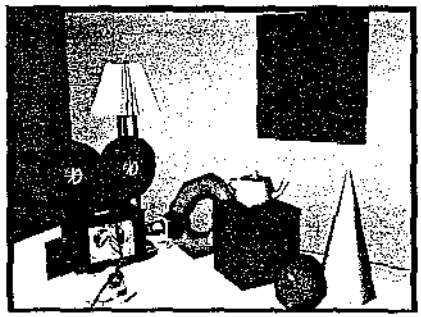


Figura I.3.8. Escena ("Shutterburg". Pixar 1990) renderizada con sombreado gouraud. Los objetos aun se ven en cierta forma artificiales.

Sombreado Phong

Fue creado para subsanar algunas deficiencias del gouraud, esto se logró aumentando el número de cálculos matemáticos, ya que, la computadora toma la normal de la cara del primer polígono y determina normales adicionales para cada esquina de la cara, si se trata de un polígono irregular se generara un gran número de normales, a fin de promediar los valores obtenidos con los de las normales de los polígonos adyacentes. Este proceso es a su vez diez veces más lento que el gouraud, pero los resultados son espectaculares en la mayoría de los casos.

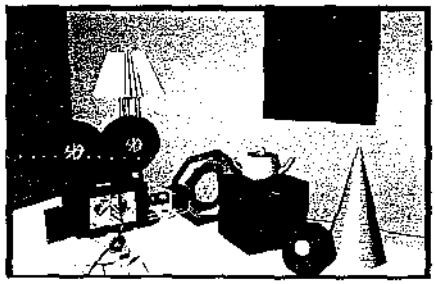


Figura I.3.9. Escena ("Shutterburg". Pixar 1990) renderizada con sombreado Phong. Los objetos se ven más realistas.

Mapeo de textura

Para las computadoras no es sencillo manejar el mundo real y detallado, por lo cual, es válido recurrir a trucos usando mapas de textura. Estos mapas son parecidos a un "papel tapiz" que se aplica a las caras de los polígonos, en ocasiones, un mapa de textura puede envolver a todo un objeto, es decir, se "pegan" imágenes a las superficies de los objetos.

Por ejemplo, si se quisieran modelar en 3D el planeta tierra, sería muy complejo llegar a tanto nivel de detalle, como querer volver a dibujar país por país en computadora, lo cual generaría miles o millones de polígonos. En lugar de esto se podría tomar una superficie del tamaño proporcional a una esfera 3D que simule el planeta y cubrirla con un detallado mapa de textura con la imagen de un planisferio. La gráfica generada no tendría profundidad pero si detalles importantes y se podría desplegar en la mayoría de las computadoras. Con los mapas de texturas se logran resultados excelentes en la simulación de gráficas complejas. Con ellos, se pueden crear escenas ricas en imágenes sin tener que elaborar un ambiente muy complicado. El ejemplo descrito se observa en las figuras I.3.10 y 1.3.11.

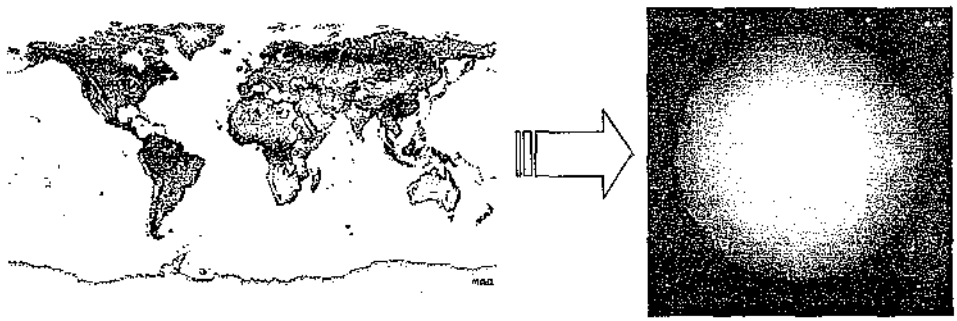


Figura I.3.10. Utilización de un mapa de textura sobre un objeto esfera.

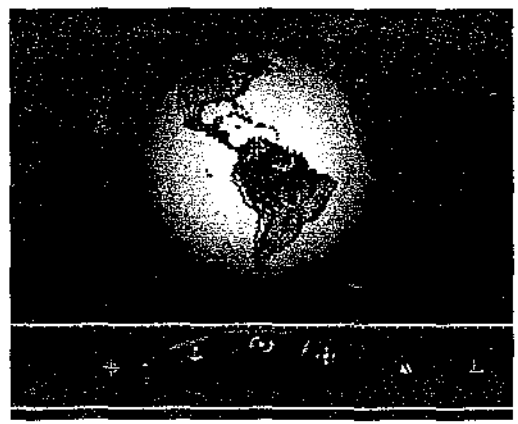


Figura I.3.11. Planeta tierra 3D, producto de texturizar una esfera con una tapiz en formato JPG.

Seis grados de libertad

Cada objeto 3D que se realice debe comenzar en un lugar en el espacio, es decir, debe tener una posición determinada de donde se comenzó y término de crear. Además de su posición, nuestra realidad tiene tres dimensiones llamadas anchura, altura y profundidad. La anchura es la característica de poder extenderse de lado a lado de cualquier objeto; la altura es la característica que comprende el espacio vertical que permite desplazamiento hacia arriba o abajo y la profundidad que es la característica observada al acercarnos o alejarnos del objeto. Las computadoras por convención, utilizan las letras x, y, z para representar estos valores.

Un objeto 3D puede moverse de seis maneras diferentes. Una manera fácil de demostrar estos movimientos es mediante la "regla de la mano derecha". Para comprender mejor esta regla obsérvese la figura I.3.12. Se debe poner la mano derecha de la siguiente forma: el dedo pulgar apuntando hacia la derecha, el dedo índice apuntando hacia arriba y el dedo medio señalando hacia adelante (enroscando los otros dos dedos y manteniéndolos apartados de los primeros tres).

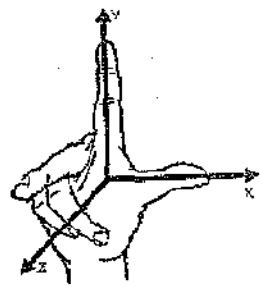


Figura I.3.12. Regla de la mano derecha

Esto muestra tres de las seis maneras en las que un objeto 3D puede moverse:

1) para adelante y para atrás a lo largo del dedo medio;
2) hacia la izquierda o hacia la derecha a lo largo del dedo pulgar;
3) para arriba y para abajo a lo largo del dedo índice.

Los otros tres movimientos del objeto pueden ilustrarse con el movimiento de la mano mientras se mantiene en la forma descrita en el párrafo anterior. A continuación las posiciones del objeto:

1) Flexionando la muñeca (con el dedo medio hacia adelante) moviendo el dedo pulgar para arriba y para abajo.
2) Al rotar la muñeca de modo que el dedo medio se mueva para arriba y para abajo.
3) Al torcer la muñeca hacia un lado y hacia el otro se representa otro movimiento.

Estos seis movimientos y posiciones muestran todas las maneras en las que se puede mover un objeto 3D.

Giros, ciclos y círculos en 3D

Haciendo una analogía que relacione nuestro entorno real con el de la computadora tenemos que, si estuviéramos en la sala de nuestra casa y estuviéramos observando la televisión, pero la luz que entra por una ventana entorpeciera esta visión, nos podríamos levantar a girar la televisión en otra posición o incluso movernos nosotros, lo que estaríamos haciendo independientemente de quien o que se mueva, sería cambiar la *orientación*. Los valores para representar o variar la orientación de un objeto están en función de los ejes x, y, z. Los valores de la orientación son: *giro, vuelta y rodado* o de manera más formal: movimientos de giro, de ciclo y circular.

El movimiento de giro se efectúa cuando se desplaza alrededor de un eje vertical, por ejemplo, la luz de un faro marino realiza este tipo de movimiento. El ciclo se efectúa cuando se gira en forma de traslación completamente en 360 grados como en el caso de la Tierra. Por último, el movimiento circular que se realiza simulando el movimiento de las llantas de un transporte, donde al mismo tiempo que gira, el objeto se desplaza.

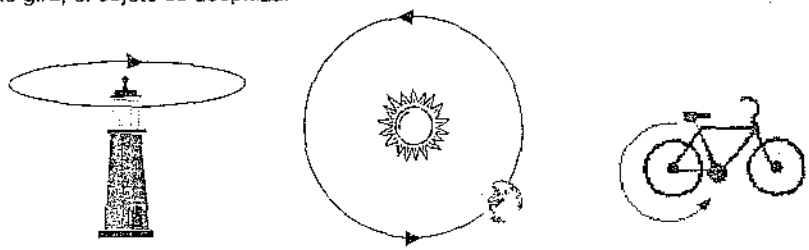


Figura I.3.13. Representación de los tipos de movimientos que puede tener un objeto.

Debido a que los movimientos son de forma circular, se deben medir de alguna forma, esto es generalmente en grados. Las variables x, y, z, dan la posición con el giro, el ciclo y el movimiento circular de la orientación para crear un conjunto llamado los *seis grados de libertad*, porque el mundo de la tercera dimensión por computadora necesita estos seis valores diferentes para especificar la posición y orientación de un objeto.

1.4. Transformaciones de objetos 3D.

La representación de objetos 3D se da a través de polígonos (puntos X, Y, Z y las conexiones entre puntos que forman el polígono). Para transformar estos objetos, basta con aplicar transformaciones afines a cada punto. Una transformación de un cuerpo rígido se logra al manipular todos los puntos con la misma transformación, una forma eficiente de lograrlo es con algebra matricial.

Suponiendo que se necesita girar una imagen alrededor de un punto que no es el origen. Si se tuviera la capacidad de trasladar toda la imagen de un punto a otro de la pantalla, se podría realizar este giro moviendo primero la imagen hasta que el centro de rotación coincida con el origen, luego se realiza la rotación y por último se devuelve la imagen a su posición original.

Desplazar la imagen recibe el nombre de *Traslación*. Se realiza de forma sencilla mediante la suma a cada punto de la cantidad que vamos a mover la imagen. En general, con el fin de trasladar una imagen (Tx, Ty), cada punto (X_1, Y_1) se convierte en uno nuevo (x_2, Y_2) donde:

$$X_2 = X_1 + Tx \qquad Y_2 = Y_1 + Ty$$

Desafortunadamente, esta forma de describir la traslación no hace uso de matrices, por lo que no puede ser combinada con las otras transformaciones mediante una simple multiplicación de matrices. Se puede observar que la rotación alrededor de un punto que no sea el origen puede realizarse mediante una traslación, una rotación u otra traslación. Lo ideal sería combinar estas tres transformaciones en una sola por motivos de eficacia y elegancia. Una forma de hacer esto es emplear matrices de 3 X 3 en vez de matrices de 2 X 2, introduciendo una coordenada auxiliar "*w*". Esté método recibe el nombre de *Coordenadas Homogéneas*.

En este método las coordenadas forman puntos definidos por tres coordenadas y no por dos. Así un punto (x, y) esta representado por la tripleta (xw, yw, w). Las coordenadas x e y se pueden recuperar fácilmente dividiendo los dos primeros números por el tercero respectivamente. Se sabe que con el sistema coordenado homogéneo y la representación de matrices, la ecuación fundamental de transformaciones en 3D se puede escribir en coordenadas homogéneas:

$$X' = (X)\,(M)$$

Donde $X' = [x'\ y'\ z'\ 1]$ (punto transformado)
$X = [x\ y\ z\ 1]$ (punto original)

Observe que transformando los puntos que definen el objeto, por sí mismo, es transformado. La transformación deseada es determinada únicamente fijando la matriz de la transformación M, apropiadamente. Podemos resumir las matrices de M para las transformaciones primarias como sigue.

Translación

Para trasladar un objeto una distancia Tx a lo largo del eje X y Tz a lo largo del eje Z. la matriz M es la matriz T, definida como:

$$T = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ Tx & Ty & Tz & 1 \end{pmatrix}$$

Figura I.4.1. Matriz de traslación.

Sustituyendo T por M en la ecuación: X' = (X) (M), obtenemos las ecuaciones lineales estándares de la translación:

$$x' = x + Tx$$
$$y' = y + Ty$$
$$z' = z + Tz$$

En la notación del vector, esta transformación se puede considerar como la adición de un desplazamiento del vector D, a un vector de la posición original x. para mover el objeto a x'. Matemáticamente, esto se escribe:

$$x' = x + D \qquad \text{donde:}$$

x = (x, y, z) (coordenadas originales del objeto)
x' = (x', y', z') (coordenadas del objeto después de la translación)

El vector del desplazamiento D, tiene las componentes Tx, Ty y Tz como se ilustra:

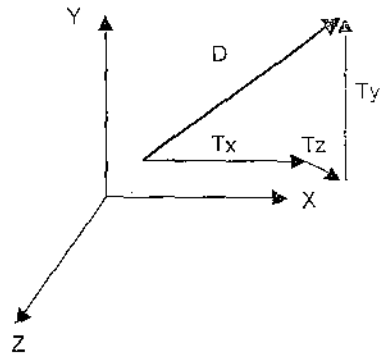


Figura I.4.2. Vector que indica la forma en que las coordenadas de un objeto se desplazan de una posición a otra.

Escala

Generalizando las transformaciones de escala de 2D a 3D, esto involucra las suma del factor Sz, para las previamente definidas Sx y Sy para la escala independiente a lo largo del nuevo eje z. La matriz M de la ecuación X' = (X) (M) está entonces representada por la matriz S siguiente:

$$S = \begin{pmatrix} Sx & 0 & 0 & 0 \\ 0 & Sy & 0 & 0 \\ 0 & 0 & Sz & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

Figura I.4.3. Matriz de escala.

Sustituyendo S por M en la ecuación X' = (X) (M) y multiplicando obtenemos las ecuaciones lineales estándares para escalar:

$$x' = (Sx)\ (x),$$
$$y' = (Sy)\ (y),$$
$$z' = (Sz)\ (z)$$

Para la ampliación uniforme comúnmente usada de un objeto por un factor de posicionamiento S, los tres factores de posicionamiento se fijan iguales a S. Sin embargo, el escalamiento diferencial también proporciona una herramienta útil para las deformaciones del objeto como se demuestra a continuación:

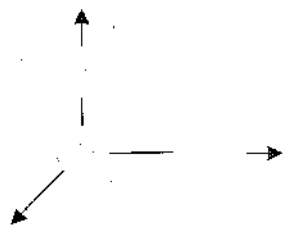


Figura I.4.4. Representación de una esfera deformada por efecto de la matriz de escala en sus coordenadas.

Rotación

Se nota fácilmente que la matriz de la translación se ha expandido para acomodarse en la tercera dimensión. Sin embargo, al extender rotaciones a 3D encontramos algunas complejidades:

- Ahora hay tres ejes sobre los cuales la rotación puede tomar lugar. Las rotaciones de θx, θy y θz son ahora posibles sobre los tres ejes independientes.
- El primer problema es relacionado con un segundo e interesante aspecto de rotaciones, que es la rotación de 90° sobre el eje X seguido por una rotación de 90° sobre el eje Y obtenemos una configuración diferente que si hiciéramos una rotación de 90° sobre el eje Y seguida de una rotación sobre el eje X. El orden de rotación es importante.

- La rotación por θ grados sobre un eje arbitrario es una transformación valiosa pero no tiene ninguna representación simple en términos de θ. Esto puede, sin embargo, ser expresado en términos de una simple matriz compuesta como una función de θ y de los parámetros de la orientación del eje arbitrario.

La solución del primer problema es definir las tres matrices de rotación Rx, Ry y Rz, correspondientes a una rotación para θ sobre los ejes x, y , z, como se muestra en la figura I.4.5.

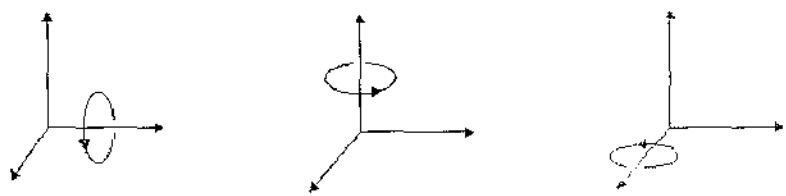


Figura I.4.5. Representación de las matrices de rotación en los tres diferentes ejes x, y, z.

El último de los renglones, Rz, es una simple extensión de la matriz R en 2D porque el movimiento en ambos R y Rz ocurre de forma paralela al plano x-y. Las tres matrices se muestran a continuación:

$$Rx = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos\theta & \text{sen}\,\theta & 0 \\ 0 & -\text{sen}\,\theta & \cos\theta & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \quad Ry = \begin{pmatrix} \cos\theta & 0 & -\text{sen}\,\theta & 0 \\ 0 & 1 & 0 & 0 \\ \text{sen}\,\theta & 0 & \cos\theta & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \quad Rz = \begin{pmatrix} \cos\theta & \text{sen}\,\theta & 0 & 0 \\ -\text{sen}\,\theta & \cos\theta & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

Figura I.4.6. Matrices de rotación correspondientes a Rx, Ry y Rz.

Sustituyendo estas matrices de rotación por M en la ecuación X' = (X) (M) y multiplicando, tenemos ecuaciones familiares para rotación sobre el eje X:

$$x' = x$$
$$y' = y \cos\theta - z \,\text{sen}\,\theta$$
$$z' = y \,\text{sen}\,\theta + z \cos\theta$$

Cualquier lenguaje que utilice 3D como en este caso VRML, trabaja internamente regido por estas matrices que son las que establecen las relaciones de rotación, translación y escala entre los objetos que quieran sufrir estas modificaciones a partir de su estado inicial.

Así, tenemos que en el apartado "capacidades del sistema" se establece que el usuario tiene la libertad de cambiar a cualquier posición los muebles de su espacio funcional con tan sólo arrastrarlo con el cursor del Mouse, VRML logra esta tarea con ayuda del nodo "PlaneSensor", este nodo es importantísimo ya que proporciona la libre translación de un objeto en un plano paralelo a Z delimitado por coordenadas en 2D, como se ilustra en la figura I.V.7.

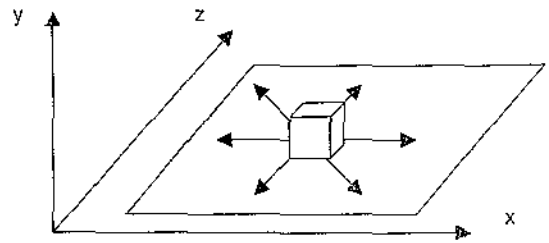


Figura I.4.7. Representación del nodo PlaneSensor sobre un cubo.

Una vez explicada la translación ¿Cómo logra VRML la rotación? Aquí ya no es tan sencillo, porque VRML no cuenta con un nodo que realice esta acción, pero como ya mencionamos su potencial, VRML trabaja con scripts, aquí se utilizó uno, este contiene una variable que se inicializa en cero, pero que se va incrementando de 90 en 90 grados cada vez que detecta un clic del Mouse sobre el objeto que contiene al script en cuestión. Con estos datos, el objeto efectúa un movimiento de 90 grados cada vez que se de un clic, pero no hemos establecido que sucede con esos 90 grados ni con referencia a que se realizará el movimiento, entonces, en el código se establece un evento "set_rotation" y un nodo "Billboard", el primero se encarga de hacer una rotación de 90 grados y el segundo de hacerlo con referencia al eje Y.

De esta manera tenemos el script más completo que realizará la acción de rotar al objeto como se ilustra:

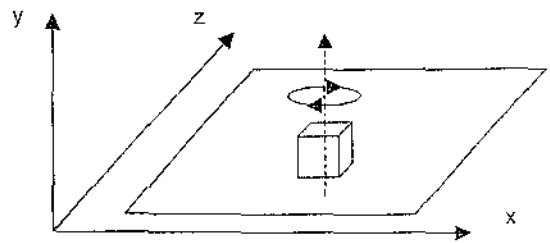


Figura I.4.8. Efecto de un script de rotación sobre el cubo.

La escala es la propiedad de modificar el tamaño de los objetos; fue utilizada en el espacio funcional que complementa este trabajo, debido a que la mayoría de los objetos inicialmente fueron creados en una escala estándar para su mejor manipulación. Al llevarlos al espacio funcional se les tuvo que escalar de manera que quedaran acordes a los demás objetos. VRML cuenta para esto, con un nodo llamado "scale", el cual se coloca junto a nodos como translation, rotation, children, entre otros. Se hace esta mención porque se observará que VRML tiene un nodo translation y un rotation, con lo que se podría preguntar ¿para qué entonces todo lo explicado en los párrafos anteriores?.

La explicación es simple, los nodos propios de VRML translation y rotation, si trasladan y rotan los objetos, pero una sola vez y recordemos que en este sistema el usuario los debe poder mover y rotar con toda "libertad"; cosa que los nodos ya mencionados no logran sin establecer los procesos ya mencionados.

Regresando al nodo "scale", este nodo cuenta con dos características. permite establecer tres valores que representan el alto, ancho y largo que se quiere escalar (o quizá podamos decir "deformar") al objeto, con valores que van de cero a infinito. Es importante mencionar que se debe tomar como referencia que un objeto con una escala normal inicial tiene los valores (1, 1, 1), que representan en ese orden a x, y, z. De esta manera, si por ejemplo se quiere escalar al triple de alto, se establecen los nuevos valores (1, 3, 1), respectivamente, como se ilustra:

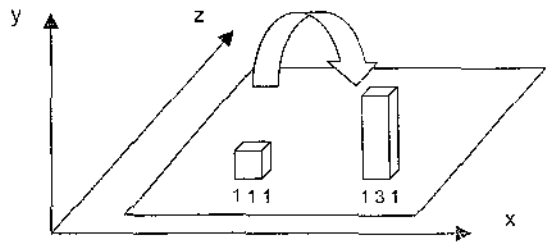


Figura I.4.9. Efecto del nodo escala de VRML sobre un cubo para cambiar su tamaño.

I.5. Aplicaciones del VRML.

La Realidad Virtual, hablando no sólo del VRML, tiene multitud de aplicaciones que la hacen valiosa. La mayoría de estas aplicaciones son indirectas a nosotros pero finalmente sirven para mejorar nuestra vida; en este apartado se citan algunos ejemplos en los que se contempla el uso del VRML como herramienta.

Una aplicación de Realidad Virtual (RV) puede facilitar muchas actividades; por ejemplo, se pueden capacitar eficazmente a empleados de una empresa, generando menos costos, por ejemplo, si una empresa se dedica a la enseñanza de conducción automotriz, entonces podrá capacitar estudiantes en un simulador que contenga un ambiente virtual y después hacer la práctica real en un automóvil; disminuyendo así los gastos y previniendo daños en los automóviles reales de la escuela de conducción. También en áreas de diseño y presentación de datos, medicina, arquitectura, educación, publicidad, televisión y en general en cualquier organización que manipule información compleja o que desee presentar de una forma más natural los procesos de ésta.

En un entorno virtual se puede aprender a manipular equipos sofisticados, aprender a conducir, realizar operaciones, conocer una construcción antes de que se realice y muchas cosas más. Actividades que parecen imposibles, como el recorrer un planeta, hacer un viaje entre las arterias humanas o simplemente estar entre la estructura molecular de algún elemento químico, se convierten en posibilidades al alcance de cualquier persona que tenga acceso a una computadora y haga uso de la RV. Los equipos especiales en un programa de RV hecho en VRML son opcionales y dependen de la tarea que se pretenda realizar. Así que no se está obligado a conseguir equipos costosos como cascos, guantes u otros implementos.

VRML se ha desenvuelto desde la nada a casi un estándar internacional en poco más de cinco años y es cierto que llegó a ser la norma para describir ambientes 3D en Internet en un tiempo. Con VRML, la habilidad del lenguaje permite a los objetos, exhibir conductas al empezar a ser explotadas en ambientes virtuales compartidos y por cierto, este es usado para la visualización de datos que están ya bien establecidos.

En la actualidad el Web es utilizado para desplegar catálogos con hojas de especificaciones y diferentes tipos de literatura publicitaria. A pesar de que este es un muy buen uso de la tecnología de Web, no está siendo explotada en su totalidad. Con la explosión del comercio electrónico, el Web se a encontrado con nuevas aplicaciones; por ejemplo la visualización física de productos, ya sea para su venta en línea o para su demostración. A través del uso de VRML, la demostración de productos en línea toma las siguientes características:

La interactividad entre el usuario y el producto que desee adquirir, observarlo de diferentes ángulos y cambiándolo de posición. La Integración de Multimedia, en la que VRML provee la integración de elementos multimedia tales como audio e imágenes. Por ejemplo, el lenguaje de programación Java puede ser utilizado para manipular objetos tridimensionales y dar detalles del producto a través de pistas de audio. Un Ancho de banda aceptable a través del uso eficiente de VRML y mundos optimizados, el tiempo de transmisión se puede decrementar enormemente, evitando que el usuario tenga que esperar mucho tiempo perdiendo el interés. Citaré a continuación aplicaciones diversas.

En el Web

Hasta ahora es muy común que las campañas de publicidad en el Web utilicen los llamados "banners", "applets" o "pop-ups" que no son más que las pequeñas ventanas con anuncios que cualquier usuario se encuentra al navegar en Internet, también se pueden encontrar planos o imágenes animadas para atraer a los internautas (usuarios de Internet) a sus sitios. Estos applets entregan poca información y su transferencia puede llegar a ser muy lenta además de molesta. Con VRML es posible generar animaciones de mayor impacto y de menor tamaño. Además el hecho de que la animación se realice en un ambiente 3D provee de mucho mayor información al usuario, logrando con esto un mayor impacto publicitario.

En el comercio electrónico

Cada vez se empiezan a ver más centros comerciales virtuales, VRML provee de nuevas opciones para que el comerciante llegue a su público objetivo. Ahora millones de usuarios conectados en línea pueden acceder a los centros comerciales ubicados en cualquier parte del mundo, pasear entre las tiendas, visualizar los productos para comprarlos o interactuar con otros compradores o vendedores.

Se han traspasado las fronteras y el concepto de "entrega a tu domicilio" se ha complementado con "visita desde tu domicilio" en Internet, aunado a esto, gracias al intercambio digital de datos, podemos saber quién visita que y cuándo lo hace, así como saber que necesita y cómo lo quiere, sin importar en que parte del mundo se encuentre.

En la industria automotriz

En la industria automotriz se valen de la RV utilizando simuladores virtuales que se conectan a elementos corporales de individuos para proporcionar así sensaciones reales. Se puede imitar todo un automóvil sin necesidad de construirlo completamente, solamente quizá, el tablero de mando, con esto, la persona que virtualmente maneja el vehículo puede experimentar sensaciones reales de movimiento, velocidad, confort y demás detalles, pero con ciertas ventajas como, no poner en riesgo su vida, no caer en gastos innecesarios de diseño o construcción, etc.

Figura I.5.1. Simulación Virtual del interior de un automóvil con el que el conductor interactúa.

Existe un prototipo virtual de interiores automotrices en el que se utilizan maquetas detalladas del interior para estudiar diferentes diseños, evaluar factores humanos y ediciones ergonómicas. Estos prototipos físicos son costosos, desperdiciadores de tiempo y difíciles de modificar. La realidad virtual de inmersiva proporciona un alternativa eficaz, porque con un prototipo virtual puede sustituir una maqueta física para el análisis de los aspectos del diseño como: disposición y eficacia de las bolsas de aire; visibilidad de instrumentos, de controles y de espejos; efectos de colisiones; funcionamiento humano; estética y más.

En la industria de la manufactura

Otro ejemplo, lo encontramos en la industria, hablando de la construcción de piezas metálicas, se sabe que existen empresas en las que se requiere que los trabajadores no tengan ningún tipo de falla en el moldeado de estas debido a su alto costo, lo cual, implicaría grandes pérdidas. Para impedir estas pérdidas, se puede entrenar a la gente virtualmente en el proceso, con visores que simulen la operación completa incluyendo sensaciones de movimiento, imagen y sonido de manera que el trabajador quede totalmente sumergido en el ambiente que tendrá enfrente cuando elabore la pieza real (la simulación observada en el visor sería creada en VRML); esta simulación tiene su principal ventaja y se justifica tan sólo en el costo que ahorra al evitar las cantidades de material que se pueden desechar si los trabajadores intentaran aprender el proceso en forma real.

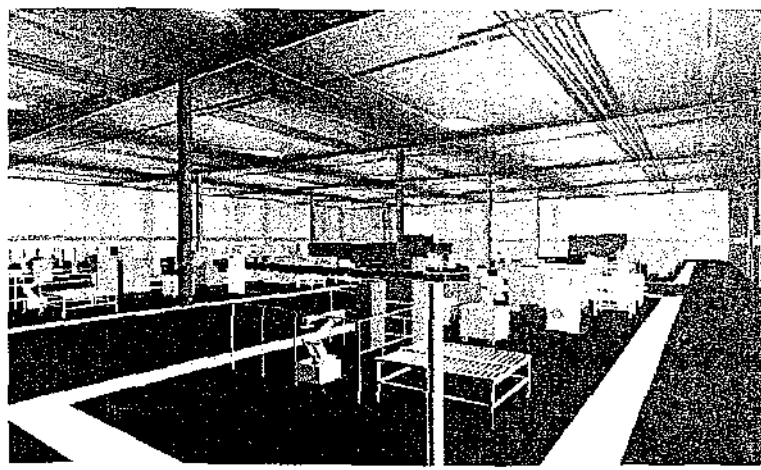


Figura I.5.2. Representación virtual de toda una fabrica de manufactura.

En la aviación

En la aviación por decirlo de alguna manera, se han simulado con VRML los planos previos a la construcción de aeropuertos, contemplando desde la vista exterior con las pistas de aterrizaje, hasta los detalles de salas de espera, baños, taquillas, cafeterías, escaleras eléctricas; incluso donde sería más óptimo colocar alguna planta de ambiente o las salidas de emergencia, en fin, todo lo necesario para hacer un aeropuerto confortable y seguro. Este ambiente previo serviría como prototipo, permitiendo la libre manipulación de los objetos, sitios específicos donde se colocarán los objetos y todo lo necesario que sirva de base para la construcción formal.

En la Universidad de Michigan [37] se trabaja la RV en el Virtual Reality Laboratory, una de sus últimas creaciones es la ampliación del aeropuerto del metropolitano de Detroit del condado de Wayne .

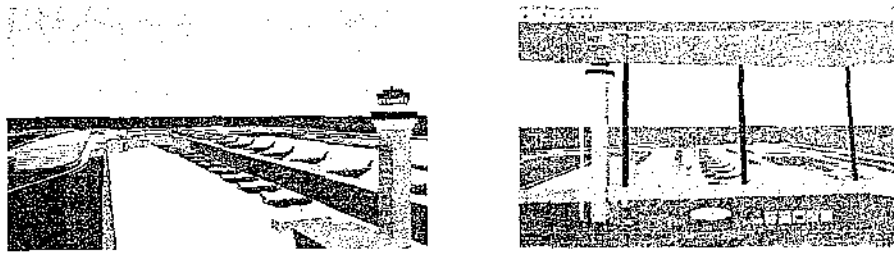


Figura I.5.3. Pistas de aterrizaje (izq.) y torre de control (der.), representadas con realidad virtual para simular una construcción futura.

El proyecto también incluye la construcción de una nueva terminal de aviación internacional y doméstica con 97 puertas, conexiones del campo de aviación, una estructura grande del estacionamiento con 11.500 espacios, un sistema de varios niveles de las vías de acceso a la nueva terminal y una central eléctrica. En cooperación con las líneas aéreas del noroeste, el laboratorio virtual ya mencionado desarrolló un modelo virtual para que el proyecto terminal de Detroit Midfield asista a la evaluación del diseño y apoye un proceso de toma de decisiones en la construcción del complejo.

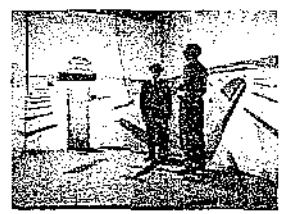


Figura I.5.4. Habitación inmersiva (tipo CAVE) en la que una persona puede sentirse dentro del mundo virtual e interactuar con los objetos que lo componen.

Un modelo de realidad virtual inmersivo tridimensional VRML, puede también observarse usando tecnologías virtuales inmersivas que consisten en ayudarse de dispositivos de exhibición montados en la cabeza o con un sistema de proyección. Tales sistemas proporcionan una representación realista y completa del ambiente e incluyen la visión estereoscópica como se muestra en la figura I.5.4, donde se observa una de las vistas creadas con RV para el aeropuerto ya mencionado, pero con un sistema de proyección, proporcionando la sensación de estar ahí.

En los juegos mecánicos

También, aunque parezca increíble, juegos mecánicos como la montaña rusa han sido diseñados con ayuda del VRML, porque no podemos imaginar subir a este tipo de artefacto recreativo a una persona como conejillo de indias, cuando está en etapa de diseño. Con un prototipo virtual hecho por computadora se puede saber que movimientos bruscos o no llevará el tripulante, como serán sus movimientos de inercia, trayectorias, efectos emocionales, si algún giro del carro pudiera provocar alguna lesión, etc.

En el entrenamiento de soldados

El ejército estadounidense es uno de los principales clientes de las tecnologías de RV hechas en una computadora. De hecho, en muchas ocasiones ha encargado a las compañías de simulación gráfica, juegos basados en estas técnicas para utilizarlos en la instrucción de los soldados que formarán parte de las fuerzas de elite. El objetivo es entrenar a los soldados para que desarrollen la capacidad de tomar decisiones adecuadas en periodos de tiempo mínimos en medio de situaciones críticas.

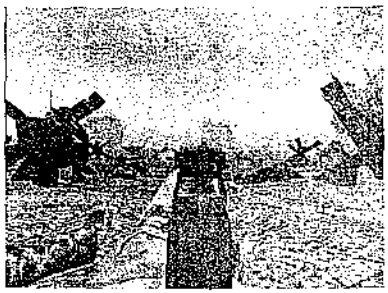


Figura I.5.5. Simulación de disparos a enemigos por computadora.

Un ejemplo de este tipo de aplicaciones la encontramos en cualquier centro comercial donde haya vídeo-juegos, existe una simulación en la que se dota a los participantes de un chaleco especial con sensores, un casco y una pistola con frecuencia láser, el juego o la batalla se realiza entre seis competidores, constituidos por equipos de tres personas cada uno, los cuales se internan en una sala aproximadamente de 30 m^2 a oscuras, en ella existen barreras para cubrirse de los ataques enemigos y unas bases en las que se recargan las pistolas láser, ya que estas sólo proporcionan cerca de 10 disparos cada una. Cada disparo acertado en un contrincante es detectado por los sensores del chaleco que lo envían a un panel que va indicando el marcador de los equipo, después de transcurrir unos 15 minutos se indica que el juego ha terminado y el equipo ganador es el que logro más disparos en el equipo contrario. Es sabido que existe una actividad del mismo tipo, pero en campo real, con armas que disparan balas de pintura muy dolorosas, así que el hacerlo con RV sigue teniendo sus ventajas.

En la física y química

Dentro del área de la física existen proyectos con distintos enfoques, aquí se describe una aplicación muy común: la visualización de fluidos de partículas. Una aplicación en el área de visualización es el fluido de partículas a través de cuerpos rígidos. Existen proyectos que modelan este tipo de fenómenos utilizando el VRML [32], donde el propósito principal es el fácil análisis de una gran cantidad de datos que facilitan el estudio de los modelos. Se cuenta con una herramienta auxiliar que permite visualizar modelos complicados de interpretar si sólo se analizan tal cual. En este tipo de proyectos, las partículas y demás elementos se logran con facilidad gracias a las geometrías de VRML. El VRML se ha aplicado a muchas áreas de la química. Los primeros modelos del VRML fueron publicados en diciembre de 1994.

De hecho navegando por Internet me he percatado de la existencia de varios sitios en los que, profesores de nivel medio superior y superior utilizan la computadora y en específico al VRML para crear simulaciones y representaciones de partículas, átomos, células, moléculas, ionizaciones y demás elementos relacionados con la física o química como se muestra en la figura I.5.6.

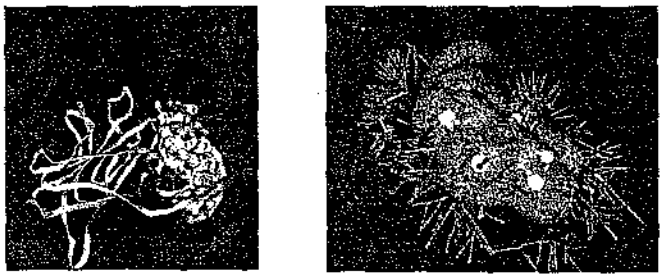


Figura I.5.6. Modelos VRML que ilustran las características dominantes de la proteína del supresor del tumor p53 (izq.) y un diagrama donde se demuestran interacciones intermoleculares en la vinculación del hidrógeno alrededor de sistemas aromáticos (der).

En la medicina

Las técnicas más precisas de la RV se están aplicando en la medicina, tanto para realizar prótesis para los disminuidos físicos como para la exploración e intervención médica a niveles celulares y genéticos. Plantas médicas virtuales, equipos virtuales y pacientes virtuales proporcionan a los estudiantes posibilidades de estudiar las experiencias de alto riesgo (figura I.5.7). Pero las aplicaciones reales también existen. De hecho, las endoscopias en estéreo pueden transmitir dibujos tridimensionales a los ojos del médico a través de un casco para que pueda hacer una cirugía, como si estuviera dentro del paciente. La RV y las tecnologías de las micromáquinas junto con el control remoto están sirviendo para que cirujanos y especialistas ejecuten las tele operaciones con dispositivos en pacientes humanos.

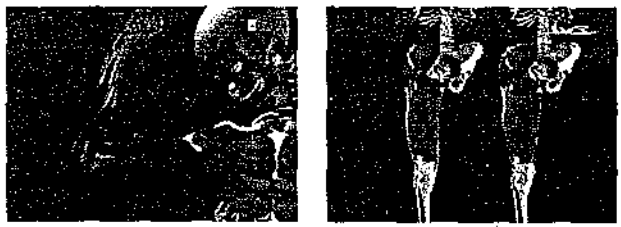


Figura I.5.7. Representaciones virtuales del sistema muscular.

En Estados Unidos, algunos psicólogos han comenzado a explorar las posibilidades de la RV para curar las fobias y traumas que sufren algunas personas. Como la compañía Virtually Better [36], dedicada a investigar las simulaciones por ordenador en el tratamiento de las fobias más extremas. En la "Virtuoterapia", cualquiera puede enfrentarse a sus temores sentado cómodamente, desde el consultorio. El sistema de RV está compuesto por un casco, unos auriculares y un sillón colocado sobre una plataforma móvil, en el que el paciente puede padecer una simulación tridimensional por ordenador con sonido envolvente de la situación que le produce angustia o simplemente ver su fobia en la pantalla de la computadora; por ejemplo, la vista simulada de un precipicio o una araña como se observa en la figura I.5.8.

Figura I.5.8. Simulación virtual de una araña para superar la fobia a estos insectos.

El sistema de Virtually Better ya se está utilizando en muchas clínicas y consultas psicológicas de Estados Unidos. Otra aplicación es el amaestrador médico con Realidad Virtual, consiste en un proyecto interdisciplinario en la Universidad de Michigan [37] que en conjunto con el departamento de medicina de emergencia, se dan a la tarea de unir medios y un laboratorio virtual.

El objetivo es el desarrollo de un "amaestrador médico virtual" que integre tecnologías avanzadas como los simuladores pacientes humanos, los sistemas virtuales inmersivos, tecnología del Internet de la generación siguiente, la comunicación video virtual y todo lo concerniente al contexto de los ambientes virtuales distribuidos y compartidos para el entrenamiento del personal de la emergencia en una variedad de campo común, así como situaciones extremas. En la figura I.5.9 se pueden observar los laboratorios médicos de cirugías elaborados totalmente con Realidad Virtual, donde una herramienta para su creación puede ser VRML.

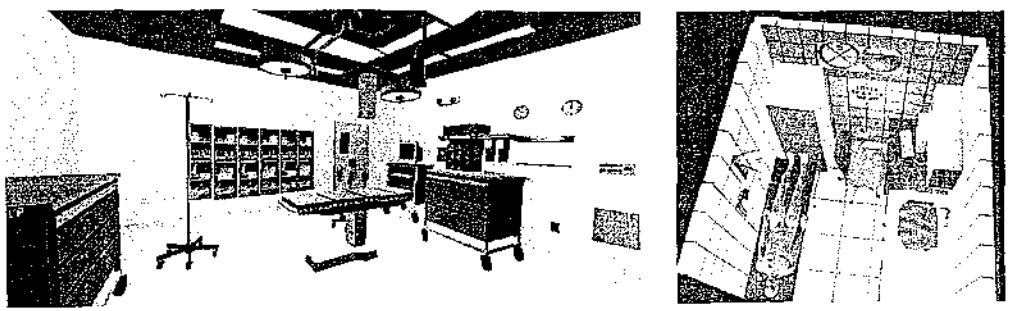


Figura I.5.9. Laboratorios Virtuales construidos para simular operaciones reales y entrenar a nuevos médicos.

En geografía

Dentro del área de Ciencias de la Tierra se realizan proyectos para algunas de las áreas de aplicación, como lo es la visualización de fenómenos volcánicos o la modelación de relieves topográficos. Sin duda, el riesgo de potenciales erupciones volcánicas es un problema que se tiene en todo el mundo. Las simulaciones de fenómenos volcánicos permiten analizar la pérdida de vida y la destrucción de cualquier infraestructura. Construyendo modelos de flujos se permite estimar los movimientos de materiales volcánicos dentro y sobre una superficie, este tipo de aplicaciones permite el entendimiento de los peligros de estos fenómenos antes de que sucedan, además del desarrollo de mapas de riesgo, asistencia en crisis y reconstrucción post-crisis.

En la Universidad de Buffalo, se desarrolló un sistema de visualización de estos fenómenos y es utilizado para el análisis de varios tipos de flujos que van desde lava de movimiento lento y saturado, que le permitirá a oficiales públicos, científicos y la población en general entender el efecto de varios fenómenos volcánicos en sus áreas locales y diseñar planes apropiados para ponerse fuera de peligro. En México se han escogido tres volcanes para desarrollar y probar los modelos científicos de este estudio, estos son el Popocatepetl, Colima y Pico de Orizaba interactuando con vulcanólogos de la UNAM.

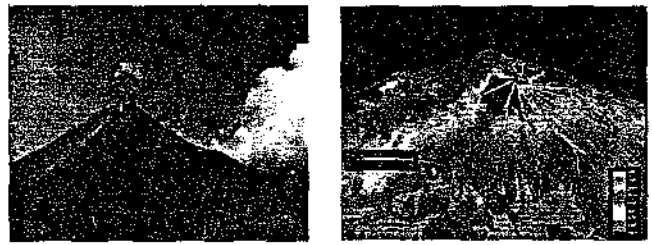


Figura I.5.10. Simulación de los fenómenos que presenta el volcán de Colima.

En la oceanología

Utilizando VRML en proyectos de oceanología se puede visualizar una estructura tridimensional de la superficie del océano, donde se puede modelar, por ejemplo, el comportamiento de mareas, tener una simulación de cómo el viento afecta las olas, u observar fenómenos como El Niño o La Niña, observando temperaturas , dirección de vientos o velocidad. Estas aplicaciones se pueden observar haciendo una visita al Instituto Meteorológico Nacional en México, donde se utilizan para hacer predicciones.

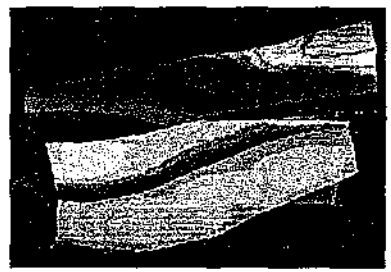


Figura I.5.11. Modelo virtual que representa el comportamiento de fenómenos oceánicos.

En medios de comunicación

Los medios de comunicación, en particular la televisión, aprovechan la tecnología de realidad virtual para obtener mayor publicidad y estar incorporando tecnología innovadora a sus transmisiones. Así, se tiene que en algunas transmisoras de Televisión en sus partidos de fútbol, en determinados momentos, se repiten jugadas en forma tridimensional, reconstruidas en diferentes ángulos para su análisis; con lo cual obtienen la ventaja de tener las jugadas desde diferentes perspectivas que a veces pueden pasar desapercibidas por las propias cámaras de televisión. Una aplicación más es el llamado "set virtual", en el que, un comentarista hace referencia a las jugadas del primer tiempo de juego ayudado por elementos virtuales en una pantalla. Debemos hacer la observación de que estos ambientes virtuales pueden ser hechos en VRML u otro lenguaje que proporcione ventajas 3D.

Las televisoras también han incorporado personajes virtuales dentro de sus programas, éstos personajes virtuales interactúan con actores reales. Un personaje virtual para televisión puede realizarse gracias a un sistema de captura de movimiento con el que una persona dá vida al ciberpersonaje, este puede ser generado mediante un sistema de Motion Capture, integrado a un software que reconoce el movimiento de los sensores; el hardware se compone de un traje con 12 o más sensores magnéticos cuyo movimiento es replicado por el cuerpo del personaje; por otra parte con un hardware consistente en un par de guantes con seis sensores cada uno se controlan las expresiones y movimientos de la cara así como los efectos preprogramados respectivamente.

Con ayuda de cuatro o más cámaras equipadas con emisores infrarrojos un sistema detecta la posición y movimiento del personaje y de esta manera permite situar en forma congruente la escenografía virtual con los diferentes puntos de vista. Las cámaras se montan en tripies con cabezales computarizados cuyos mecanismos transfieren las coordenadas de su posición al sistema. El set virtual de la televisora emplea técnicas de rastreo de movimiento: óptica, mediante la lectura de una retícula; otra a través de sensores infrarrojos; y otra mediante cabezales computarizados. Esta combinación de elementos reales con virtuales en tiempo real aunada a la libertad total de movimiento de cámaras, representan la vanguardia tecnológica en herramientas para la producción, en vivo, de programas de televisión.

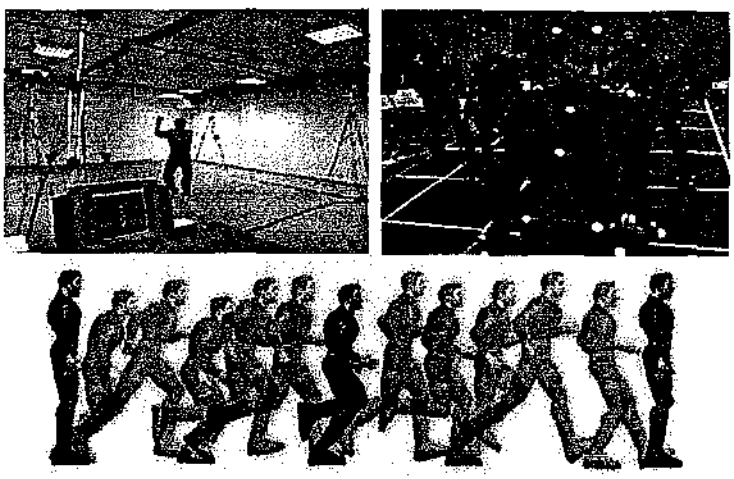


Figura I.5.12. Animación virtual con el sistema "Motion Capture" utilizado en televisión, Cine y videojuegos.

En el arte

El VRML juega un papel importante para el conocimiento y es utilizado por museos, planetarios y centros de ciencia. Con VRML estos centros realizan exposiciones virtuales donde se pueden hacer recorridos en templos antiguos, palacios, galaxias, aprender de diversas áreas de conocimiento, entre otras. En algunos de los proyectos realizados en los centros, se experimenta con situaciones más cotidianas o con las que los visitantes (principalmente los jóvenes y niños) pueden identificarse, por ejemplo, se puede diseñar una montaña rusa y posteriormente experimentar el viaje como si físicamente se estuviera en la montaña, así, mientras se disfruta del viaje se pueden aprender leyes de la física.

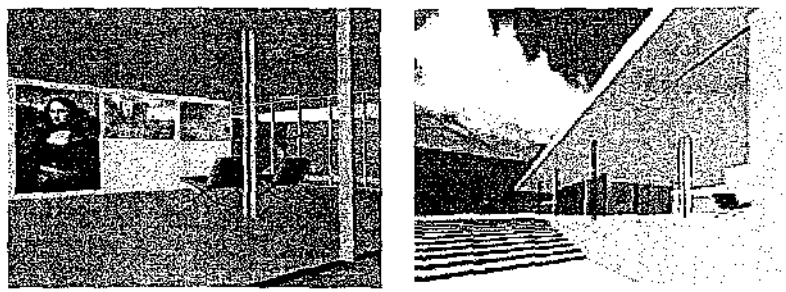


Figura I.5.13. Museo virtual en el que se pueden preestablecer los espacios de las obras de arte (izq) y representación virtual de la entrada principal del pabellón de Barcelona (der).

Otro de los enfoques que se le da a la realidad virtual, es el de experimentar visitas virtuales a lugares o templos antiguos que por alguna razón no están disponibles al usuario o se presenta una propuesta de restauración. Algunos de estos impedimentos son: destrucción, restauración o distancia. Muchas veces, los museos también cuentan con exposiciones virtuales (colecciones de arte, objetos históricos, etc) a través del Web, con lo que abren la posibilidad de llevar cultura y conocimiento a personas que por alguna razón no puedan visitarlos físicamente, también amplían la percepción de otras culturas y/o formas de vida antiguas, al permitir los recorridos virtuales por lugares históricos. Un ejemplo de este tipo lo encontramos en el Pabellón Barcelona (figura I.5.13), diseñado por Mies Van Der Rohe, era una entrada alemana para una exposición en 1929 del mundo, en Barcelona. El pabellón existió por seis meses y desapareció durante el envío de nuevo a Alemania. Aunque su existencia ahora se confino a Internet, el pabellón Barcelona todavía se considera la obra maestra más famosa de la arquitectura moderna, ya que cuenta con gran técnica de dibujo, información de fondo y un viaje vía HTML o VRML [35]. Este pabellón está desarrollado basado en los datos obtenidos por la empresa SGI para experimentar con realidad virtual inmersiva y para el estudio de los siguientes elementos: simplificación de geometría, técnicas de RV inmersiva, representación de la estructura en RV inmersiva, navegación alrededor del pabellón, utilización de texturas en superficies planas y curvas, animaciones y la representación del modelo en VRML 1,0 y 2,0.

En la arquitectura

La manera en que los arquitectos comunican sus ideas la mayor parte de tiempo es en forma visual, el utilizar alguna forma de visualización facilita la comprensión de información compleja y facilita la comunicación. Hoy en día, cada vez son más los arquitectos que utilizan a la realidad virtual como una herramienta más para participar a los demás de sus ideas y trabajos.

Figura I.5.14. Panorama arquitectónico virtual.

Algunos de los enfoques más comunes que los arquitectos dan al uso de realidad virtual es en el modelado virtual de sus diseños de casas y edificios, donde además de hacer los diseños tradicionales como planos y maquetas, elaboran un modelo tridimensional interactivo, donde sus clientes pueden contemplar de una manera más "real" los diseños o inclusive adentrarse en estos edificios o casas y recorrerlos libremente, teniendo así una visión más clara de las ideas que se tratan de expresar.

Los dibujos por computadora y las técnicas de visualización han sido usadas por los arquitectos durante casi dos décadas y con el paso de los años, los profesionales de campos como la ingeniería de diseño o la arquitectura, han sabido valorar los avances en tecnología gráfica y explotarlos para ahorrar los costos de desarrollo y tiempo. Al principio, sólo se usaban para construir planos sencillos y para producir la impresión de programas por computadora, desarrollados para generar dibujos bidimensionales o imágenes en una pantalla, permitiendo así al usuario acercarse rápidamente a un determinado punto para observarlo en detalle o alejarse para obtener una visión global del dibujo. La mayoría de los programas de diseño arquitectónico que se usan hoy en día son de este tipo.

Existen otros programas de computadora, adaptados por arquitectos y diseñadores, incluyen programas que generan imágenes elaboradas y tridimensionales que pueden ser rotadas, alteradas o combinadas por el usuario mediante órdenes, los diseñadores que empiezan a adquirir cierta habilidad con estas herramientas sienten el deber de poner a prueba incluso sus propias creaciones y son estimulados para probar ideas nuevas e incluir de una manera más activa a sus clientes en los procesos de planificación y diseño. En Internet se pueden encontrar ejemplos de arquitecturas realizadas con la tecnología virtual [35], tales como el pabellón "Karls Platz" en Viena y la capilla "Notre Dame Du Haut".

En el cine

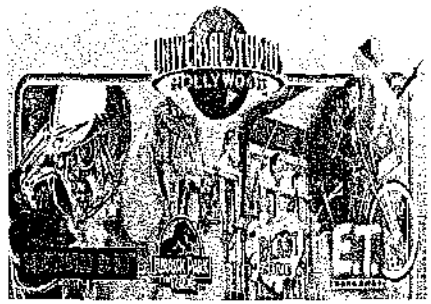


Figura I.5.15. Algunas de las películas que han requerido de efectos virtuales para lograr el éxito.

La empresa sgi ha sido la creadora de las estaciones de trabajo más poderosas que han existido. En sus máquinas y con software de RV se ha dado vida a películas como Star Wars, Jurassic Park, Toy Story, etc. Estas producciones han logrado mucho éxito basándose en las bondades de la RV; ejemplo de esto se observa cuando en escenas de acción se sustituye a personajes reales por representaciones hechas en computadoras para no exponer a un actor al peligro. Otra aplicación es realizar replicas exactas de estructuras que existieron hace mucho tiempo y que hoy sería imposible ver; por ejemplo el Titanic o el Coliseo Romano. No se puede dejar de mencionar que se pueden simular multitud de elementos como precipicios, cascadas, desastres naturales, vuelos por el universo, animales gigantes y un sin fin de aplicaciones que significarían grandes costos o producciones de baja credibilidad por falta de realismo.

Aplicaciones en Ingeniería

Dentro de las áreas de ingeniería hay proyectos de manipulación remota como lo son la manipulación de robots, o procesos de ensamblado; también existen áreas dedicadas al desarrollo de prototipos virtuales. Estas aplicaciones facilitan la automatización de diferentes formas; por ejemplo, cuando se tiene un proceso de ensamblado de algún producto, se presentan distintos acontecimientos como puede ser las deformaciones de plástico, fricción externa, fenómeno termal, absorción y factores como el desgaste de herramientas, ocasionando errores de dimensión y forma.

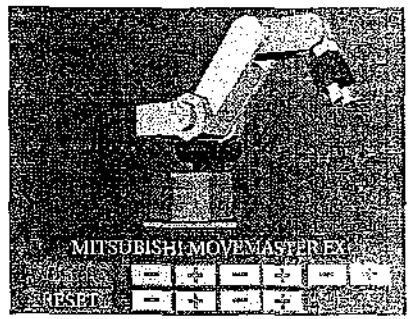


Figura I.5.16. Brazo prototipo para ensamblado, construido inicialmente en forma virtual.

Si se tiene información adicional sobre el efecto de los parámetros antes mencionados sobre la variación en los valores de tolerancia y dimensión, se pueden desarrollar mecanismos para el ensamblado automático. Usando un modelo de elementos finitos se pueden visualizar las fuerzas que actúan en el proceso de manufactura y la deformación del equipo bajo la acción de estas fuerzas. Si se tiene un ingeniero en diseño y manufactura que pueda observar el ensamblado de una de las partes por medio de la computadora y dispositivos especiales, puede sugerir cambios en la tolerancia de los valores basándose en las condiciones de las máquinas, herramientas, fisuras y requerimientos de diseño. Un tipo de aplicación como ésta puede permitir obtener un configuración de ensamblado óptimo para satisfacer los requerimientos funcionales, por lo que, es un tipo de herramienta efectiva para el proceso de toma de decisiones. Este tipo de proyectos son totalmente inmersivos.

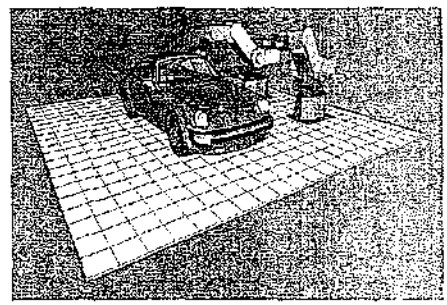


Figura 1.5.17. Representación virtual del funcionamiento del brazo.

En ingeniería se utilizan mucho este tipo de aplicaciones, aquí, el diseño de prototipos es combinado con un modelado virtual de estos, permitiendo al diseñador participar activamente en el detallado del diseño y la optimización del proceso. Las bondades de VRML permiten generar ambientes computarizados para que el diseñador investigue y pruebe múltiples cambios a sus diseños que está realizando mientras observa y manipula objetos virtuales al usar movimientos humanos naturales. Los diseños interactivos permiten cambiar los parámetros de diseño e inmediatamente determinar el efecto de los cambios.

Existen compañías que utilizan la realidad virtual como una herramienta en el diseño de dispositivos de control de contaminación y de calderas. De esta forma, la compañía puede garantizar el funcionamiento de sus productos, incluso antes de haberlos construido. Trabajando con las especificaciones de los productos, se modelan nuevas calderas y se simula su temperatura, dirección y velocidad de consumo de los gases. De esta forma, al realizar distintos experimentos con la colocación de los inyectores y otras características físicas, se crea el mejor sistema controlador de contaminación para la caldera y se integra dentro del diseño antes de que la caldera sea construida. Antes de la realidad virtual, se utilizaban modelos computacionales estadísticos que tomaban semanas para calcular. Con este proceso, se puede completar el análisis en un día o menos, incluso con mayor exactitud.

Existen empresas dedicadas a crear y explotar los beneficios del software de 3D para crear sistemas de modelado, una de ellas es GraphicsNet. En este tipo de aplicaciones se realizan productos simulados gracias a la interacción hombre-máquina. Una vez creado un producto se tiene interacción con él, usando las propiedades de reconocimiento de gesturas por parte del software. Avanzados sistemas de modelado 3D son la base para la planificación y detallado diseño de productos. Las actividades de Ingeniería virtual dentro del GraphicsNet se apuntan a desarrollar nuevas técnicas de innovación para la interacción del humano-computadora para soportar el proceso de desarrollo del producto entero.

La interacción inmersiva 3D, cuenta con reconocimiento de gestos y la realización de esbozos a pulso para facilitar las fases tempranas del diseño de un objeto. La conjunción de la interacción 3D y las técnicas de modelado abren paso a nuevas formas de realizar un producto en forma intuitiva tridimensional, obteniendo así un desarrollo de alto nivel. De esta manera surgen nuevos conceptos del lugar de trabajo: las áreas de trabajo virtuales, dentro de las que el humano es capaz de presentar prototipos virtuales, repasarlos y modificarlos.

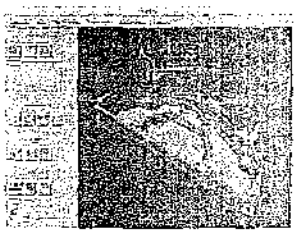


Figura I.5.18. Prototipo 3D creado en computadora pudiendo de esta manera interactuar con él antes de crear el objeto real.

Algunos ejemplos de aplicaciones que tienen estos productos 3D en la ingeniería son los siguientes:

ARCADE (Advanced Realism CAD Environment) Ambiente Avanzado CAD de Realismo. Sistema de modelado 3D que permite - como ya se menciono – que un desarrollador tenga la libertad y comodidad de crear y modificar modelos 3D usando dispositivos de salida 3D. También se puede dar otra forma de manipulación de este sistema trabajando en una llamada "mesa virtual" donde se pueden esbozar objetos a mano alzada y dibujo libre. De esta manera, tan sólo con los movimientos de las manos el usuario puede crear objetos dentro de un espacio 3D.

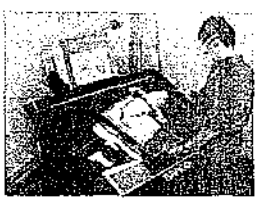


Figura I.5.19. Mesa virtual en la que se pueden realizar bosquejos 3D a mano alzada.

Gesturas basadas en interacciones dinámicas 3d.- Cuando se crean manualmente objetos en el mundo real, se llevan a cabo diversos movimientos corporales y con las manos para ir dando forma a un producto final. Refiriéndonos a un trabajo parecido al que realiza un artesano y panadero que con cada gestura que realiza con las manos se va formando una figura de forma determinada, ahora también es posible con la RV. Dichos movimientos son usados para dar vida a una amplia variedad de aplicaciones 3D. Con ayuda de una caja de herramientas que reconoce los movimientos dinámicos de las manos, una interfaz y métodos de inteligencia artificial (como redes neuronales y lógica), se superan las desventajas y deficiencias que se tendrían usando un mouse.

ARCADE.- El sistema de modelado de objetos 3D ARCADE permite una planeación cooperativa y simultánea a varios usuarios en la elaboración de un espacio 3D. En una interfaz se muestran las manipulaciones y cambios que van realizando los usuarios en tiempo real. Existe un protocolo de control que permite a cada usuario un control dinámico de un objeto cada vez, es decir, cada usuario tiene el control cada vez sin que nadie más intervenga para tener un orden, de esta manera, cuando el usuario termina su operación, el objeto queda accesible para otro participante. Dos elementos son importantes en esta interacción: los avatares (representaciones virtuales de los participantes) y la videoconferencia integrada, estos elementos permiten que cada usuario sepa la ubicación tridimensional de los demás participantes y como cada uno ve el o los objetos 3D. Se concluye de esta manera que el medio de comunicación e interacción de ARCADE es el Internet.

AUTOCAD.- Permite el uso de un plugin de conferencia en el que pueden conectarse dos o más aplicaciones transmitiendo una gran cantidad de datos entre los participantes que se concentran en una sesión. En este sistema cada usuario trabaja en su propio objeto pudiendo cambiar configuraciones en el sistema y hacer este trabajo colaborativo.

Figura I.5.20. Sistema CAD donde un usuario trabaja en su proyecto individualmente pudiendo hacerlo colaborativo.

Los componentes importantes de desarrollo del producto integrado son los prototipos digitales, el sistema DMU (Alta Simulación Digital) es usado en industrias como aviación y automovilismo, por ejemplo, en el se pueden realizar presentaciones de alta calidad, simulación de posibles colisiones o de otros tipos, donde estas herramientas servirán de base para tomar ciertas decisiones al momento de crear un producto final. Las tecnologías de Realidad virtual permiten presentaciones y representaciones en tres dimensiones y tiempo real. Un ejemplo de este tipo de prototipo digital virtual es llamado VIRGO, se compone por un auto y un humano digital con el que se puede analizar la ergonomía del vehículo.

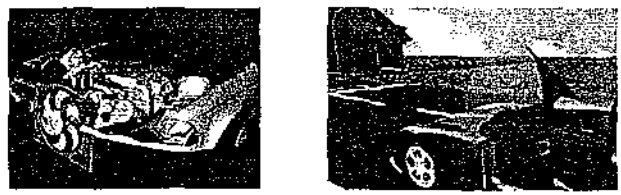


Figura I.5.21. Representación virtual de un auto en el que se evalúan diversas características para elaborar el producto final.

Soporte en grupo para empresas virtuales

Algunas empresas comienzan sus proyectos con bajos presupuestos, desconfianza o simplemente no desean realizar las inversiones que implican el instalar y configurar la gestión de redes avanzadas y soluciones en grupo dentro de su organización. En su lugar utilizan las bondades del Internet como el correo electrónico, las páginas Web y la transmisión por FTP. En este ámbito surgen otros factores que intervienen en el proceso, como lo son una comunicación asíncrona, amenazas a la seguridad de la red o usuarios que no desean trabajar en la instalación y mantenimiento de la red. En una red de Internet o Intranet sólo se usa un navegador Web, protocolos HTTP, mecanismos de seguridad, conectividad con una base de datos y enlaces dinámicos para lograr un buen producto a la hora de comunicarse con los distintos colaboradores sin recurrir a gastos excesivos.

Concluyendo, tenemos que, con la invención de la Realidad Virtual en los años sesenta y del VRML en 1994, varios desarrolladores han explotado su creatividad al crear varios trabajos en interfaces VRML sabiendo que su potencialidad como medio interactivo va más allá de una aplicación local, porque va hacia Internet. En un principio la realidad virtual fue usada en su mayoría para aplicaciones militares o incluso de entretenimiento; sin embargo, en los últimos años se han diversificado las áreas en que se utiliza y se ha colocado al alcance de usuarios comunes.

II. DEFINICIÓN Y CARACTERÍSTICAS DE VRML.

II.1. Funcionamiento de un visualizador VRML.

Para conocer como funciona VRML se necesita saber en forma general como trabaja el Web y saber el manejo básico de archivos VRML. Tomando como base la forma en que funciona el Web, debemos plantear que el proceso de navegación implica a dos elementos fundamentales: un navegador y un servidor. El navegador se encarga de interpretar las páginas Web y hacer peticiones de datos a un servidor mientras que este último se encarga del envío y recepción de datos dependiendo del tipo de petición que haya hecho el navegador.

Cuando el servidor recibe una petición, debe analizarla para generar la información requerida, es decir, dentro del envío incluye el "tipo de contenido", el cual, no es más que información que le permite al navegador saber de que tipo de datos se trata y poder diferenciar entre un texto que son sólo caracteres o una imagen que son datos binarios, etc; de otra manera el navegador no sabría exactamente que es lo que esta recibiendo o no lo podría presentar correctamente.

La mayoría de los navegadores de Internet, llámense Opera, Navigator o Explorer tienen una operación similar, reciben el tipo de contenido para saber si despliegan un archivo, por ejemplo de sonido, wav, mpg, mp3 u otro tipo. En caso de que en su acervo interno, no contemple correctamente el tipo de contenido del archivo a desplegar, éste generará una ventana (que la mayoría de internautas ha visto) indicando con que tipo de aplicación queremos ejecutar dicho archivo que no puede reconocer, o en su defecto, preguntará si deseamos guardarlo en el disco duro.

Por su parte los archivos de VRML cuya extensión es .WRL, no necesitan alterar el funcionamiento de los servidores, lo que lleva a poder agregar de una manera fácil y rápida archivos VRML a Internet. La única modificación que debe hacer el usuario es indicarle al servidor la extensión "wrl" e incluir el tipo MIME (x-world/x-vrml), datos con los que el servidor podrá reconocer los archivos de VRML y podrá enviarlos. Este único cambio ha representado para los administradores de sitios de Internet algo muy accesible y es la razón que llevó a VRML a tener tanta popularidad entre ellos.

VRML, no sólo puede representar una escena o un ambiente virtual, también contempla enlaces a páginas Web o incluso entre mundos VRML. Otra característica es el llamado "punto de vista", que permite observar el mundo VRML desde una o varias posiciones predefinidas, esto funciona como un atajo para llegar a una posición determinada sin tener que realizar un recorrido por corto o largo que sea. Desde el surgimiento de este lenguaje de interpretación 3D, han surgido varios visualizadores, aunque no todos con gran aceptación, algunos de ellos se muestran en la figura II.1.1.

Figura II.1.1. Algunos de los visualizadores más usados por los programadores de VRML: Cosmo Player, Blaxxun y Cortona.

Continuando con el funcionamiento, tenemos que para visualizar un archivo wrl, se realiza la petición desde un visualizador que no tiene que ser necesariamente VRML sino que también puede ser HTML, una vez hecho esto, el servidor analiza la petición y transmite el archivo con el tipo de contenido implícito, una vez recibido por el visualizador, éste lo analiza sintácticamente y posteriormente si es correcta, elabora una representación visual del contenido del archivo pudiendo observarse un mundo VRML en pantalla, se puede ver en la figura II.1.2 su representación.

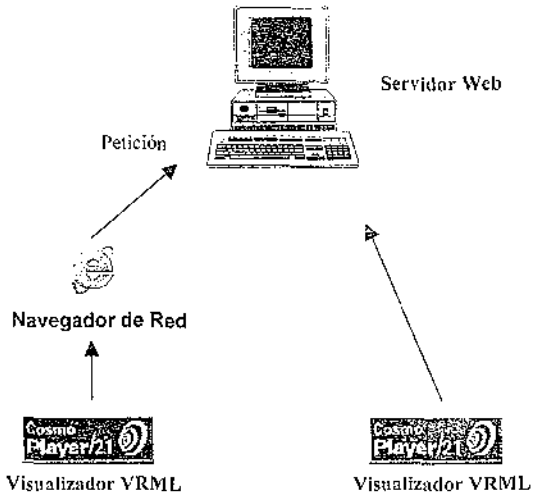


Figura II.1.2. En esta figura se observa cómo un visualizador VRML puede realizar la petición de una escena u objeto virtual por medio de un navegador o directamente.

Se ha mencionado que VRML maneja enlaces del mismo modo que HTML, ya que la función de un navegador es intentar desplegar los datos recuperados del Web; sin embargo, un visualizador VRML sólo puede trabajar con datos de VRML, es decir, objetos tridimensionales. De esta manera, cuando un usuario navega por un mundo VRML puede dar clic en un enlace o vínculo representado por un objeto 3D para activar alguna aplicación que procese los datos correspondientes.

Por último, es importante dejar claro que VRML es un lenguaje para *descripción de escenas* y no un lenguaje de programación, porque algunos lenguajes de programación como el "C", primero compilan el programa (lo traducen en lenguaje que la máquina pueda entender) y después se ejecutan. En cambio VRML pasa por un análisis sintáctico (para convertirlo en objetos que la computadora pueda manejar) antes de ser desplegado en la pantalla. La descripción de escenas es un proceso estático, porque los elementos no cambian cuando se carga el archivo VRML. Otra característica es que lo que se observa al descargar un archivo VRML es el resultado puramente final, lo cual, significa que no existen sorpresas desagradables como los virus que se ocultan en documentos y otros. También, a diferencia de muchos programas que pueden tardar minutos para generar la aplicación ejecutable del código, VRML proporciona resultados tan pronto como se termine de programar el mundo 3D.

II.2. Iluminación en VRML.

Hablar de iluminación implica el uso de las propiedades de la luz que influye en la manera en que se ven los objetos, en este caso, de un mundo u objeto tridimensional. Dentro de estas técnicas existen las dedicadas a la iluminación global encargadas de dar una apariencia más real a los objetos de una escena VRML.

Lo novedoso de esta técnica es que utiliza intensa y exclusivamente texturas para representar luz. Combinando lo anterior con métodos de filtrado y reconstrucción, se logra mejorar la calidad visual del dibujado (en inglés rendering) de las texturas.

VRML permite la fácil transferencia de ambientes 3D que cruzan por Internet. Lo cual genera una plataforma y un lenguaje abierto, poderoso y suficiente para describir ambientes sofisticados. Los navegadores actuales permiten ver estos mundos VRML por medio de un API. La calidad de dibujo que tengan las escenas depende del buen manejo del lenguaje. Por ejemplo el uso de VRML en el campo de la arquitectura está restringido a crear mundos donde sólo se pueden explotar los modelos de iluminación y prototipos e interactivamente sólo se puede caminar en una escena. Al estar estos mundos restringidos a caminar en una escena prácticamente estática, surge la necesidad de desarrollar un visualizador mejor, más hábil para manejar correctamente las luces y materiales.

Para lograr lo anterior se debe evaluar un modelo de iluminación global que pueda representar ambientes VRML 3D y por otra parte se debe privilegiar la información luminosa de alta frecuencia. Una de las herramientas utilizadas son los mapas de luz, para depositar datos luminosos y manejar la calidad del proceso de iluminación. Al hacer uso de los mapas de luz, se permite implantar directamente filtros y métodos de reconstrucción con lo cual se puede mejorar la calidad y precisión visual de dibujo.

Los mapas de luz se comunican por medio de texturas para no sobrecargar el procesador. De esta manera cada método de iluminación enfocado a la visualización ha sido tratado para ser procesado con texturas. Existen diversos autores que han propuesto diferentes usos de optimización con texturas o mapas de luz, entre ellos, Arvo, Heckbert, Myszowski, Kunn, entre otros. Los trabajos recientes sobre el tema apuntan a utilizar las texturas en el proceso de iluminación global para mejorar la calidad y velocidad de dibujo de escenas con iluminación compleja.

Tomando en cuenta esta información, se evalúa rápidamente la información de la alta frecuencia fotométrica e inicia el proceso de iluminación global. Contrario a los clásicos métodos de radiosidad multi-resolución, se ha hecho gran uso de los mapas de luz multi-resolución. Estos mapas de luz por un moderado costo de memoria permiten un control permanente y fino en la calidad de iluminación del dibujo.

Los procesos de filtrado implantados en las actuales plataformas son esencialmente dos: filtrado de borde para mejorar precisión visual de zonas discontinuas calculadas durante la etapa de iluminación y una etapa de reconstrucción de mapa de luz al proceso indirecto de iluminación. El usar los mapas de luz para representar la información luminosa durante el proceso de dibujo de la escena no es reciente. Las texturas permiten separar sombreado y geometría, para tener un mejor control sobre los métodos de interpolación, incrementar el desempeño del dibujo con actuales elementos de hardware gráfico y permitir la manipulación de grandes números de celdas fotométricas elementales.

El uso de mapas de luz es decisivo durante la elaboración de modelos dinámicos, ya que son enteramente construidos antes del proceso de iluminación y es viable su uso para representar la incidencia de la luz sobre superficies. Las tuberías de iluminación afectan de forma diferente al motor de iluminación y dibujo.

El mezclado del mapa de luz terminal y la textura del material es hecho después del proceso de dibujo por la interfaz de aplicación del navegador. Cuando se mezclan los mapas de luz intermedios, tenemos un resultado hecho al vuelo debido al proceso de iluminación global y la textura material. La ventaja de este método es la disociación de la representación fotométrica y los detalles de superficie. Pero surgen dos desventajas importantes: la sobrecarga del proceso de dibujo con la mezcla de funciones y la representación incorrecta de la incidencia de luz.

El área válida para los componentes de los mapas de luz dados en código RGB (Red Green Blue) es [0.0, 0.1]. Componentes arriba del valor 1.0 producidos por la iluminación son sujetados a 1.0. Por tanto, los componentes texel[1] RGB de las texturas de color pueden solamente decrementarse. Para corregir estas últimas desventajas tenemos que escoger otro modelo de tubería en el cual la mezcla es hecha completamente durante el proceso de iluminación. Una desventaja de este proceso es la necesidad de convertir una textura morfológica (resolución, filtrado, interpolación y reconstrucción) para yuxtaponerla[2] en la función de mezclado.

Las luces de los archivos VRML, son implantadas fácilmente porque son emitidas desde un punto sencillo o son direccionales y no extendidas. Las luces extendidas son manejadas directamente por la etapa de iluminación indirecta. Este tipo de iluminación es la más costosa en términos de consumo de recursos del CPU. Gracias a la primera etapa de iluminación, sólo necesitan ser evaluadas las interacciones fotométricas de segundo orden.

Cuando se desea iluminar todo un entorno, se tienen dos opciones básicas: iluminación pixel a pixel, o a nivel de primitiva de dibujado. El iluminado pixel a pixel conlleva crear un mapa de iluminación para la escena, indicando la cantidad de luz y color que cae sobre cada lugar, para colorear la imagen correctamente. Esto le permite proyectar luz con la forma e intensidad que desee, por lo que es sencillo tener objetos con luz propia; por ejemplo: antorchas, focos y efectos de cualquier forma extraña e irregular que pueda concebir. Obviamente hay una gran cantidad de procesado necesario para dibujar una imagen separada para las luces y si se toma este camino, se perderá toda la información direccional de la luz.

Un mapa de luz funcionará bien si tienen muchas luces de formas extrañas, afectando a un número relativamente reducido de pixeles. Cuando se tienen varios pixeles y pocas luces simples (ej: omnidireccionales), podría ser mejor iluminar las cosas a medida que dibuja, calculando la cantidad de luz que caería en cada objeto y coloreando todo en una pasada.

En modos de 256 colores de resolución de monitor, es sencillo usar un valor de 0 a 255 para el nivel de luz y un cuadro precalculado de 64k para mezclar éste con cada pixel. Es casi seguro que no merece la pena iluminar una imagen de 15 ó 16 bits directamente, porque el rendimiento será atravesado completamente por las interminables separaciones en componentes RGB individuales, sus transformaciones y respectivas combinaciones en pixeles con formato empaquetado. Sería más lógico trabajar con formatos de 24 bits en memoria (ó quizás 32 para mantener pixeles alineados) y reducir éste a 15 o 16 bits mientras copia la imagen procesada a la memoria de vídeo.

Las luces con color son ciertamente atractivas, pero actualmente están sobrevaloradas en importancia. Recientemente las máquinas se han hecho lo suficientemente rápidas como para soportar iluminación RGB completa a una velocidad razonable, por lo que tenemos juegos como Incoming, Forsaken y Unreal lanzando destellos verdes y púrpuras cada vez que dispara un misil. No se puede negar que eso se vez bien, pero podría estarse abusando de ello y dentro de unos años miraremos atrás y criticaremos lo vacíos que son estos juegos.

[1] (Texture Element) Unidad base de una imagen. Mientras que los pixeles son elementos básicos de cualquier imagen, los texels son sus equivalentes en un mapa de textura.
[2] Colocar un objeto junto a otro.

Observando prácticamente cualquier película o programa de televisión: la vasta mayoría de las luces son blancas y hay buenas razones para ello. Sutiles tonos de naranja, azul o rosa pueden hacer maravillas para añadir atmósfera, pero puede alterar los gráficos originales para conseguir el mismo efecto. No hay necesidad de luz roja alrededor de una chimenea, cuando puede obtener el mismo efecto aplicando luz blanca a un gráfico de fondo rojizo.

El problema de dibujar un mapa de luz es que hay que hacerlo todo el tiempo, por cada pixel de la pantalla. Este procedimiento estándar además limita la iluminación de un pixel a un color entre negro y el color original, pero no más brillante: exactamente así trabaja el mundo real, pero tiende a enfadar a los artistas porque significa que deben dibujar todo excesivamente brillante y no pueden predecir como se verá hasta que vean el juego en acción, con un nivel ambiente mucho más oscuro.

Una manera para evitar esto sería dibujar las imágenes tal y como deberían verse normalmente y entonces añadir luz donde las cosas son particularmente brillantes, al contrario del procedimiento habitual de quitar luz donde las cosas son oscuras. Cuando se iluminan las cosas por encima de su tono original (ej: multiplicando por un valor mayor que uno), se arriesga al desborde y tener que recortar el valor del color y esto distorsionará su tono si alguno de sus componentes debe ser recortado antes que los demás hayan llegado a sus máximos. Amplificar lo que originalmente es una imagen oscura también podría fastidiar el contraste y tono del gráfico y cuando las cosas se dibujan muy oscuras, sufren errores de cuantización debido a la limitación de bits, lo que puede resultar malo cuando se añada más luz.

Luces en 3D

Cuando la computadora ya tomó los puntos del espacio, ya los conectó y asignó una superficie, el objeto debe iluminarse con una luz generada por la misma computadora, a fin de crear un efecto de reflexión, el brillo y sombras. Al igual que en el mundo real ningún objeto es visible mientras no esté iluminado, es por esto que las aplicaciones en las que se genera 3D se incluye una luz ambiental que proporciona una iluminación uniforme cuando no existe otra fuente luminosa.

Cuando la iluminación es bien utilizada se puede crear un ambiente lo más real posible y al igual que en el mundo real, una iluminación brillante puede iluminar toda una habitación, mientras que una lámpara direccional sólo resaltará porciones específicas del ambiente. Para citar cada ejemplo, siempre existirá una analogía en el mundo de las gráficas tridimensionales.

Las *luces apuntadoras*, son aquellas que irradian la misma luminosidad en todas direcciones.

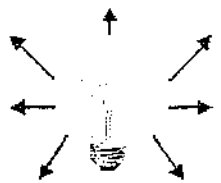


Figura II.2.1. Luces apuntadoras como las usadas en una habitación.

La *luz apuntadora paralela* es parecida a la apuntadora simple con la diferencia de que los rayos emitidos son paralelos entre si, lo cual, significa que la luz actúa como si la fuente estuviera muy lejana, esto es de gran utilidad cuando se quiere imitar la luz del sol.

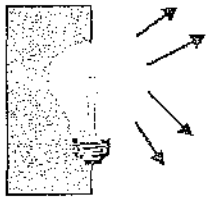


Figura II.2.2. Luz apuntadora en forma paralela sirve para simular fuentes de iluminación lejana.

Las *luces direccionales* tienen una posición (asignada por las coordenadas x, y, z) y una orientación (en grados y mediante las variables de giro, vuelta y rodado); todos estos factores indicarán donde se encuentra la fuente de iluminación, así como su dirección de enfoque.

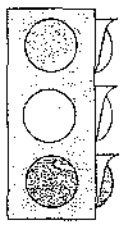


Figura II.2.3. Luz direccional. la cual se dirige orientada en una dirección determinada.

Las *luces fijas* son una variación de las direccionales, ya que además de contar con una dirección hacia la que proyectan la luminosidad, incluyen un enfoque llamado *umbra*[3], la cuál, determina el ancho del rayo de luz y la velocidad con que cruza el espacio del mundo generado por la computadora.

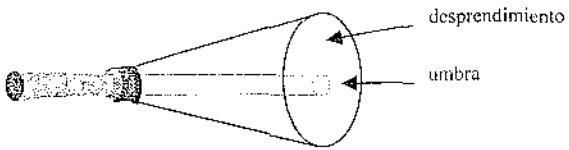


Figura II.2.4. Tipo de Luz fija compuesta por la umbra.

Las geometrías y objetos VRML son iluminados por la suma de las luces que se presentan en el mundo en el que son creados. Esto incluye la contribución de la iluminación directa y la iluminación ambiental. La iluminación ambiental es el resultado de esparcir la reflexión de luz emitida directamente por las fuentes luminosas. La cantidad de luz ambiental es asociada a las luces individuales de la escena. Ésta es una aproximación parecida a como ocurre realmente la reflexión ambiental en la naturaleza.

[3] adj. relativo a que causa sombra.

El efecto de iluminación en VRML se logra combinando velocidad y exactitud, depositando más énfasis en la velocidad. Un modelo de iluminación más exacto requiere cálculos de iluminación excepcionales, produciendo un dibujado y procesamiento más lento. El modelo de iluminación de VRML es similar a los usados en software y hardware de gráficos actuales.

Los tipos de nodos que sirven para declararse como fuentes luminosas son:

- DirectionalLight
- PointLight
- SpotLight

Todos los nodos de iluminación contienen una intensidad, un color y un campo de intensidad ambiental (ambientIntensity). El campo de intensidad especifica el brillo de la emisión directa de la luz y el ambientIntensity especifica la intensidad de la emisión ambiental de la luz. La intensidad puede ir de 0.0 (emisión mínima) a 1.0 (la intensidad máxima). El campo de color especifica las propiedades del color espectral de los campos mencionados con valores RGB.

La sintaxis de DirectionalLight es la siguiente:

```
DirectionalLight {
        ambientIntensity    0       [0,1]
        Color color         1 1 1   [0,1]
        direction           0 0 -1  (- ∞ , ∞ )
        intensity           1       [0,1]
        on                  TRUE
}
```

El nodo DirectionalLight define una fuente de luz direccional que ilumina un vector 3D dado usando rayos paralelos. El campo *direction* especifica el vector de la dirección de la iluminación que emana la fuente de luz en el sistema de coordenadas local. La luz se emite lejos a lo largo de los rayos paralelos de una distancia infinita. Una fuente de luz direccional ilumina sólo los objetos que se encuentran en su grupo. La luz puede iluminar todo dentro de este sistema de coordenadas, incluso todos los hijos y descendientes del grupo padre. La forma en que trabaja este nodo se muestra en la figura II.2.5.

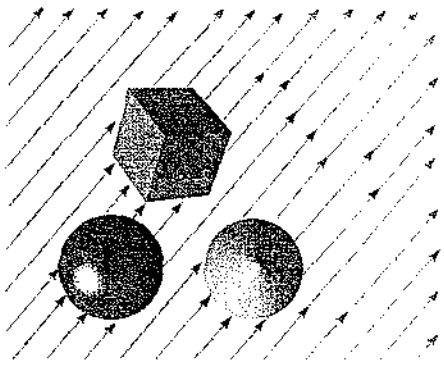


Figura II.2.5. Representación de la forma en que trabaja *DirectionalLight*

El nodo PointLight especifica un punto como fuente luminosa en una escena 3D. Una fuente de este tipo emite la luz igualmente en todas las direcciones; es decir, una luz omnidireccional. Un nodo PointLight ilumina una geometría dentro de su radio. La iluminación del nodo PointLight se pierde con la distancia. Un valor de atenuación de (0, 0, 0) es idéntico a (1, 0, 0), por lo que los valores de atenuación deben ser mayores a 0.0. La forma en que actúa este nodo se muestra en la figura II.2.6.

La sintaxis de PointLight es la siguiente:

PointLight {

ambientIntensity	0	[0,1]
attenuation	1 0 0	[0, ∞)
Color color	1 1 1	[0,1]
intensity	1	[0,1]
location	0 0 0	(-∞ , ∞)
on	TRUE	
radius	100	[0, ∞)

}

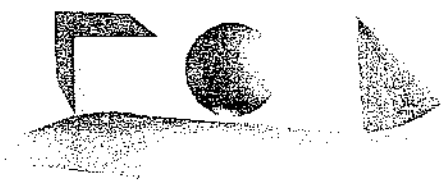


Figura II.2.6. El nodo *PointLight* emite la luz en todas direcciones sobre los objetos.

Por su parte, el nodo SpotLight define una fuente que emite la luz de un punto específico a lo largo de un vector en una dirección específica y delimitado en un ángulo sólido. Los nodos del SpotLight se especifican en un sistema de coordenadas local y son afectados por transformaciones anteriores a él.

La sintaxis de SpotLight es la siguiente:

SpotLight {

ambientIntensity	0	[0,1]
attenuation	1 0 0	[0, ∞)
beamWidth	1.570796	(0, ∞ /2]
color	1 1 1	[0,1]
cutOffAngle	0.785398	(0, ∞ /2]
direction	0 0 -1	(-∞ , ∞)
intensity	1	[0,1]
location	0 0 0	(-∞ , ∞)
on	TRUE	
radius	100	[0, ∞)

}

El campo *location* especifica el desplazamiento del punto del centro de la fuente de luz desde el origen de sistema de coordenadas local. Este punto es el ápice del ángulo sólido que limita la emisión de la fuente de luz dada. El campo *direction* especifica el vector de la dirección del eje central de la luz definido en el sistema de la coordenada local. El campo *on* especifica si la fuente emite o no la luz. Si es VERDAD, la fuente esta emitiendo la luz y puede iluminar la geometría en la escena. Si es FALSO, la fuente ligera no emite la luz y no ilumina ninguna geometría.

El campo *radius* especifica la magnitud radial del ángulo sólido y la distancia máxima desde donde *location* puede ser iluminado por la fuente del luz. La fuente de luz no emite fuera de este radio. El radio será >= 0.0. Ambos nodos *radius* y *location* son afectados por las transformaciones anteriores a estos (las escalas afectan a *radius* y transformaciones afectan a *location*). El campo *cutOffAngle* especifica el límite exterior del ángulo sólido. La fuente de luz no emite fuera de este ángulo sólido. El campo *beamWidth* especifica un ángulo sólido interno en que la fuente de luz emite a una intensidad completa uniforme.

Si *beamWidth* es mayor que *cutOffAngle*, *beamWidth* se iguala a *cutOffAngle* y la fuente de luz emite con una intensidad completa dentro del ángulo sólido entero definido por el *cutOffAngle*. Ambos *BeamWidth* y *cutOffAngle* deben ser mayores que 0.0 y menores o iguales a (∞/2). En la figura II.2.7 se observan los campos *beamWidth*, *cutOffAngle*, *direction*, *location* y *radius* del nodo SpotLight.

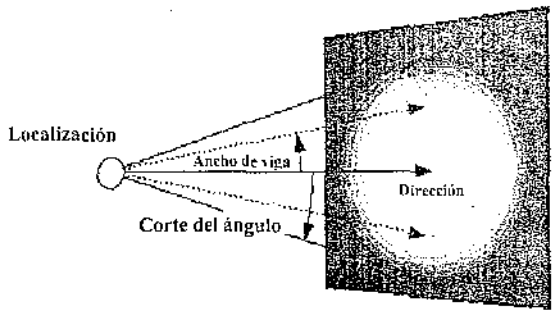


Figura II.2.7. Nodo *Spotlight* y sus campos.

En si, DirectionalLight permite aplicaciones eficaces que permiten una cantidad razonable de luz sobre un objeto o geometría de una escena virtual. DirectionalLight no intenta duplicar su posición en la jerarquía de la escena de los objetos que ilumina. Esto puede producir un resultado poco realista. Por ejemplo, una luz direccional que ilumina todo dentro de un cuarto, no iluminará un objeto que viaja en el cuarto a menos que ese objeto esté en la parte del cuarto de la jerarquía de la escena y un objeto que se mueve fuera del cuarto continuará siendo encendido por la luz direccional hasta que se mueva fuera del grupo del cuarto.

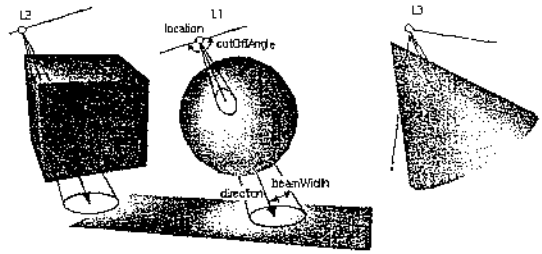


Figura II.2.8. Efecto de una luz direccional.

Cabe mencionar que estos nodos de iluminación muestran el potencial del lenguaje VRML cuando de dar realismo a sus escenas se trata. Tan sólo se deben observar las características que tienen los nodos planteados; por ejemplo; DirectionalLight que otorga al programador el poder de color, dirección e intensidad de la luz que se emite. PointLight que puede especificar un punto cualquiera en la escena como origen de una fuente de luz, la cual irradia en todas direcciones, pero perdiéndose con la distancia; lo cual, es parecido a como sucede en realidad. Por último SpotLight que genera un punto origen de emisión de luz a lo largo de un vector.

II.3. Transformaciones en VRML.

En VRML el nodo usado para transformaciones de los objetos es el llamado "Transform", su función colocar a un objeto o varios objetos en un sistema coordenado ayudado de nodos children (hijos) siguiendo una jerarquía de nodos descendientes. Contiene nodos que se combinan para generar diferentes efectos; como centrado, rotación, traslación y escala sobre los objetos que le pertenecen:

Nodo	Ejes	Rango	Ángulo	Rango
Center	x,y,z	(-∞ , ∞)	NO	NO
Rotation	x,y,z	(-1 , 1)	SI	(-∞ , ∞)
Translation	x,y,z	(-∞ , ∞)	NO	NO
scale	x,y,z	(0 , ∞)	NO	NO

Cuadro II.3-1. Nodos pertenecientes al nodo Transform.

Las operaciones de translation, rotation y scale ocurren como una operación natural y son independientes entre sí. Si se desea aplicar los tres campos (translate/rotate/scale) a un objeto, se pueden declarar antes de que se declare el children que agrupa a el o los objetos que se quieren trasladar, rotar y escalar.

Si se escribe en el nodo translation la coordenada (1, 0, 0), entonces los objetos que en ese momento pertenezcan al nodo padre Transform serán trasladados una unidad a la derecha, sin importar los campos de rotación o escala. Quiere decir que translation trabaja moviendo los objetos a nuevas posiciones sobre los ejes x, y ó z, determinadas por las nuevas coordenadas establecidas.

En un comedor virtual, por ejemplo, las coordenadas de las sillas, se modificarían para ubicarlas cerca de la mesa y en posiciones adecuadas, utilizando translation:

Esto se lograría con la siguiente sintaxis del código obteniendo el desplazamiento que se observa en la imagen :

```
# se declara el nodo de transformación
Transform {

# se traslada el objeto silla 3 unidades positivas con respecto a su posición inicial
        translation 3 0 0

# se declara un nodo hijo del nodo transform
        children [

# se llama al código del objeto silla para no escribir todo el código en esta parte
                Inline {url ["SillaComedor.wrl"]} }

# se cierra el nodo hijo
                ]

# se cierra el transform
        }
```

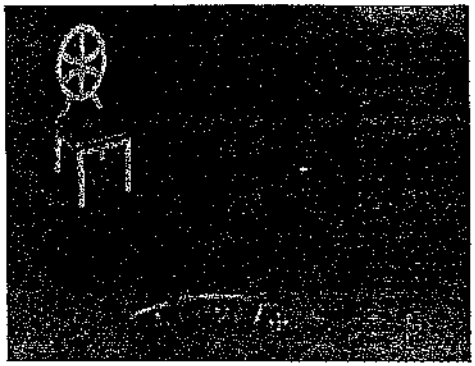


Figura II.3.1. Objeto Silla sin ser afectado por el nodo Transform.

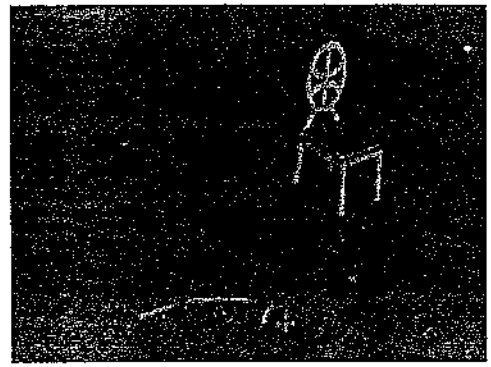


Figura II.3.2. Objeto Silla después de ser afectado por el nodo Transform.

La operación del escalamiento trabaja con escalas uniformes en los tres ejes, dejando claro que el tamaño normal de un objeto sin alterar su escala siempre es por omisión para VRML : (1, 1, 1). Entonces, para escalar un objeto 1:3 (uno a tres), se debe escribir el parámetro (3, 3, 3) en el nodo scale y aumentará su tamaño tres veces a lo alto, ancho y largo de manera uniforme. De igual manera se pueden escalar objetos con la intención de reducir su tamaño. Los valores de escala negativos no están permitidos, así que el truco a usar será definir valores menores a la unidad, con lo que, si se quiere reducir un objeto a la mitad se debe escribir (0.5, 0.5, 0.5).

Ahora una lámpara cuyo tamaño se disminuirá para verse adecuada al tamaño de una sala :

```
# se declara el nodo de transformación y una posición inicial en (0, 0, 0)
Transform { translation 0 0 0

# el nodo escala disminuyendo a la mitad el tamaño de la lámpara
                scale 0.5 0.5 0.5

# comienza el nodo hijo de transform
                children [

# se llama al código del objeto lámpara para no escribir todo el código en esta parte
                        inline {url ["lampara.wrl"]} }

# se cierra el nodo hijo
                                ]

# se cierra el nodo padre transform
                                }
```

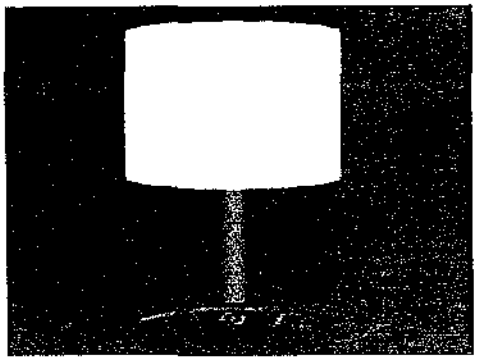


Figura II.3.3. Objeto lámpara sin ser afectado por el nodo scale.

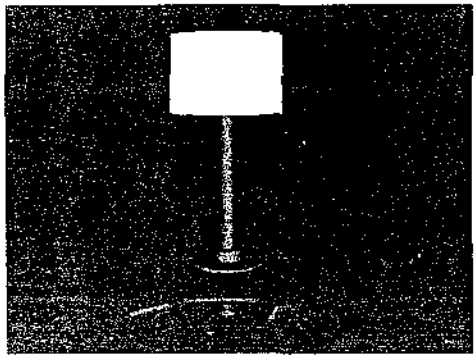


Figura II.3.4. Objeto lámpara después de ser afectado por el nodo scale.

El nodo rotation funciona rotando (girando sobre su eje), al objeto sobre el cual tiene efecto el nodo. La sintaxis de este nodo debe contener sobre que eje se hará la rotación y a que ángulo. Es importante mencionar que VRML no maneja grados sino radianes, por lo que se debe realizar la respectiva conversión teniendo en cuenta la siguiente fórmula: (grados x 6.28) / 360. VRML interpreta si el ángulo es positivo o negativo para saber en que sentido realizará el giro.

Para ejemplificar este nodo usaremos un objeto lavabo o lavamanos, en el cual se usaría la rotación para acomodarlo de manera que no quede de frente con el retrete si este fuera el caso:

```
        # se declara el nodo de transformación y una posición inicial en (0, 0, 0)
Transform { translation 0 0 0

        # se declara el nodo que rotará 45° positivos al objeto lavabo sobre el eje "y".
                rotation 0 1 0  0.785

# comienza el nodo hijo de transform sobre el que tendrá efecto rotation
                children [

# se llama al código del objeto lavabo para no escribir todo el código en esta parte
                        Inline {url ["lavabo.wrl"]} }

# se cierra el nodo hijo
                                ]

# se cierra el nodo padre transform
                                }
```

El resultado es el mostrado en las imágenes:

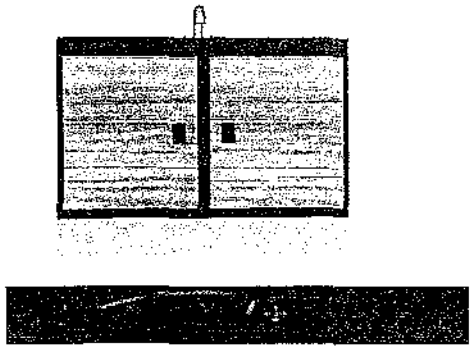


Figura II.3.5. Objeto lavamanos sin ser afectado por el nodo rotation.

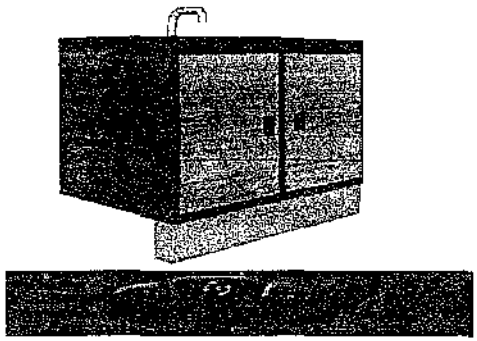


Figura II.3.6. Objeto lavamanos después de ser afectado por el nodo rotation.

Como se observa, las transformaciones en VRML son de gran utilidad para dar movilidad a los objetos que conforman los escenarios virtuales. Además se pueden modificar los tamaños y posiciones, cada cambio que se realice tendrá efecto al momento de recargar el navegador y por ende, el visualizador, ya que todo sucede en tiempo real. Más adelante se abordan más nodos que conforman a VRML y que lo justificaron para ser el medio virtual de este trabajo.

II.4. Formas, apariencia y geometrías en VRML.

El nodo que determina la forma de cualquier objeto en VRML es SHAPE (forma), el cual tiene dos campos: apariencia y geometria; usados para crear los objetos dados en un mundo. El campo de la apariencia contiene un nodo de Apariencia que especifica los atributos visuales (por ejemplo, material y textura) para ser aplicados a una geometría. Por su parte el campo de la geometría establece la figura o tipo de geometría que se utilizará para dar forma a un objeto.

La sintaxis del nodo Shape es la siguiente:

```
Shape {
        appearance   NULL
        geometry     NULL
        }
```

El nodo de la Apariencia especifica las propiedades visuales de la geometría de un objeto dadas a su vez por los nodos: material, texture y textureTransform. El valor para cada uno de los campos en este nodo puede ser nulo o no. A continuación la sintaxis del nodo appearance:

```
Appearance {
                material          NULL
              texture             NULL
                textureTransform  NULL
                    }
```

Si se especificó el campo llamado Material, se contendrá un nodo Material. Si el campo material es NULO o no especificado, la iluminación será nula también (todas las luces se ignoran durante el dibujado del objeto que hace referencia a esta apariencia) y el color por defecto es (1, 1, 1), o sea blanco. Estas características se detallan más a fondo en el apartado "Iluminación en VMRL".

Por otra parte, si se declara el campo de la textura, contendrá uno de los varios tipos de nodos de la textura: ImageTexture, MovieTexture, o PixelTexture. Si el nodo de la textura es NULO o no especificado, el objeto no tendrá textura alguna. Al especificar el nodo textureTransform, este, contendrá un nodo de TextureTransform (obsérvese que el nodo comienza con mayúscula), el cual, ayudará a determinar las coordenadas en 3D para la textura sobre la superficie del objeto.

El nodo Shape asocia un nodo de geometría con nodos que definen la apariencia de esa geometria. Los nodos de Shape deben ser parte de la jerarquía de la transformación o nodo Transform para tener cualquier resultado visible mientras que la jerarquía de la transformación debe contener los nodos de Shape para que cualquier geometría sea visible. Un nodo de Shape debe contener un nodo de geometría en su campo de la geometría. Estos son nodos de geometría válidos:

- Box
- Cone
- Cylinder
- Extrusion
- IndexedFaceSet
- IndexedLineSet
- PointSet
- Sphere
- Text

Es importante saber que si el nodo Geometry, se especifica como nulo o no, en caso de ser nulo el objeto no será dibujado.

El nodo Shape se puede usar en cualquier objeto, incluyendo los que componen un espacio funcional, para darle un aspecto agradable y lo más parecido posible a la realidad. El nodo de apariencia tuvo efecto en varios objetos que necesitaban color, para esto se utilizó el nodo material y un código de colores RGB que maneja VRML. En la figura II.41 se muestran códigos de los colores RGB que se pueden usar en VRML.

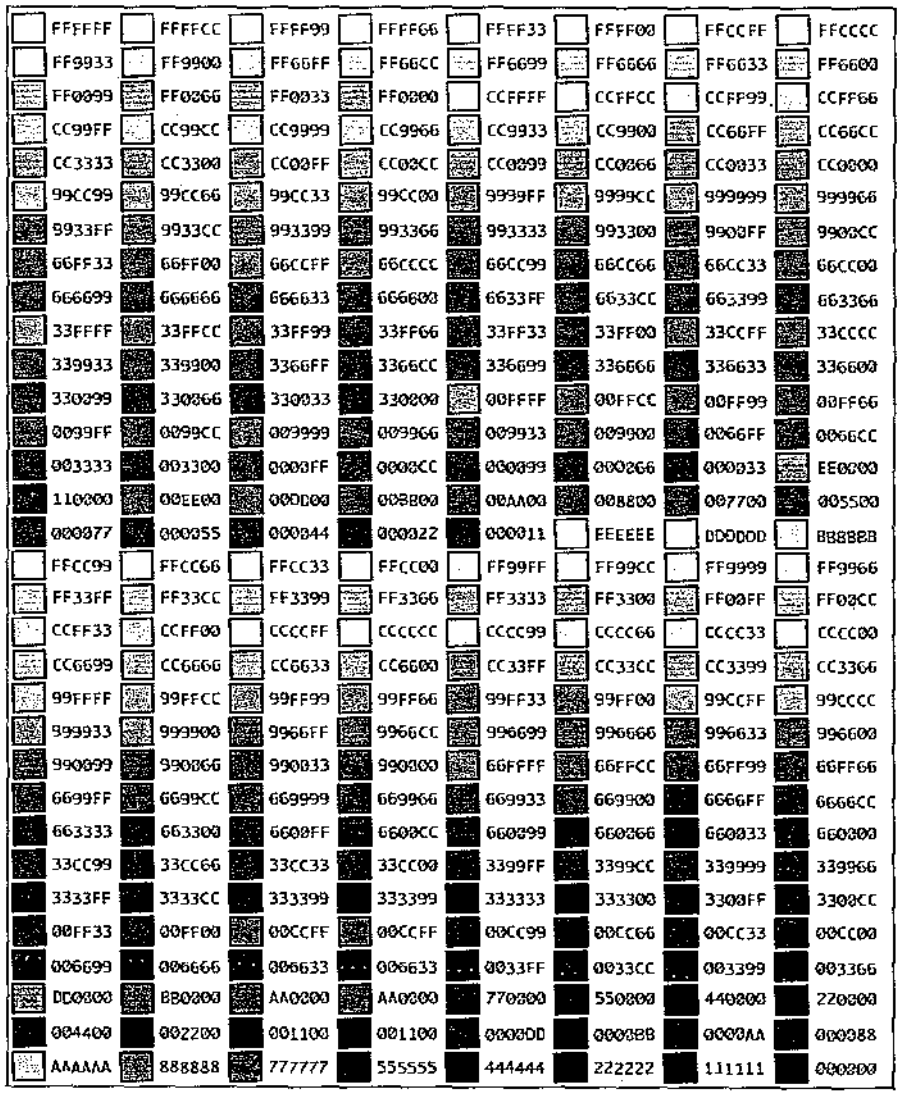


Figura II.4.1. Colores RGB que maneja el código VRML.

Es importante señalar que los colores mostrados en la tabla están dados en hexadecimal, por convención se debe realizar la equivalencia correspondiente a decimal para establecer dichos colores en código VRML. Esta conversión se realizará tomando en cuenta que los números de la tabla están dados en tres pares y VRML solo utiliza tres números, la otra consideración es que VRML maneja un rango de cero a uno y no de cero a quince como se haría en hexadecimal. Tomando en cuenta estas observaciones se concluye que realizando las equivalencias correspondientes se pueden utilizar los colores mostrados en la tabla sin ningún problema sobre los objetos VRML. A continuación ejemplos de la conversión.

HEXADECIMAL	DECIMAL	COLOR
FF0000	1 0 0	ROJO
00FF00	0 1 0	VERDE
0000FF	0 0 1	AZUL
FFFF00	1 1 0	AMARILLO
FF00CC	1 0 0.8	ROSA
CC00FF	0.8 0 1	VIOLETA

Cuadro II.3.2. Equivalencias entre código RGB hexadecimal y decimal para VRML.

Por ejemplo un objeto "estufa" puede ser un simple cubo de color blanco, pero forrado o texturizado con un tapiz que logra simular una estufa por el frente y parte superior. Como se ve en el código de este objeto, el nodo "appearance" se encarga de llamar a la textura y si es necesario escalarla o rotarla según sea necesario. El resultado se observa en las figuras 2 y 3.

```
#VRML V2.0 utf8

# CUERPO DE LA ESTUFA

Group {children [ Transform {translation 0 0 0
children[

#FRENTE
Transform { translation 0 -0.15 0
        children [Shape {appearance Appearance {material Material{emissiveColor 1 1 1}
                                #AQUI ACTUAN LOS NODOS APPEARANCE Y MATERIAL
                                }
                        geometry Box {size 0.9 0.7 0.9
                                }
                        }
                ]
        }


#QUEMADORES
Transform { translation 0 0.2 0
        children [Shape {appearance Appearance {material Material{}
                texture ImageTexture {url "quemador.jpg"        # AQUI ACTUA EL NODO
          TEXTURE QUE LLAMA LA IMAGEN QUE TEXTURIZA AL CUBO Y DA FORMA
                                                A LA PARTE SUPERIOR DE LA ESTUFA
                                }
                                }
                        geometry Box {size 0.85 0.02 0.85
                                }
```

```
                    }
                ]
        }
#FRENTE
Transform { translation 0 -0.15 0.45
        children [Shape {appearance Appearance {material Material{}
                texture ImageTexture {url "estufa.jpg"                    # AQUI ACTUA EL
    NODO TEXTURE QUE LLAMA LA IMAGEN QUE TEXTURIZA AL CUBO Y DA FORMA
                                                    A LA PARTE FRONTAL DE LA ESTUFA

                                }
                                        }
                        geometry Box {size 0.89 0.69 0.02
                                }
                    }
                ]
        }
#BASE PISO
Transform { translation 0 -0.58 0.425
        children [Shape {appearance Appearance {material Material{}
                                        }
                        geometry Box {size 1 0.17 0.1
                                }
                    }
                ]
    }

]]
```

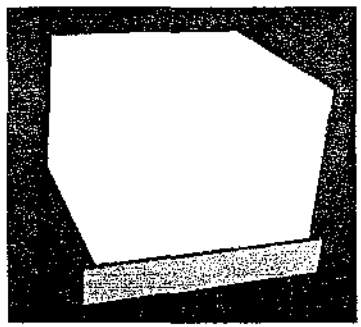


Figura II.4.2. Objeto estufa sin textura.

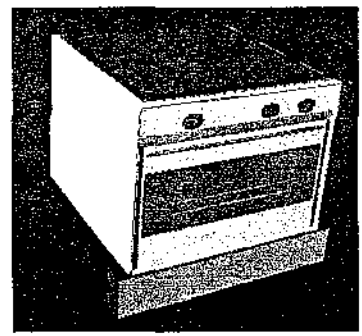


Figura II.4.3. Objeto estufa texturizado.

II.5. VRML como modelo de flujo de datos para ambientes virtuales.

Dentro de los IVE (Ambientes Virtuales Inmersivos), tenemos a VRML que es un MDF (Modelo de Flujo de Datos) porque en él transitan datos y dependiendo del efecto de las funciones que actúan sobre ellos, se transforma el estado del sistema. De otra manera los datos que fluyen por el sistema forman gráficas internas conformadas por los nodos de las funciones y arcos, dichas gráficas indicarán posibles rutas a tomar por los datos.

La versión actual de VRML 2.0, describe sus ambientes virtuales utilizando también un MFD de operación. En VRML el funcionamiento del MFD parte de saber que los nodos son la representación de funciones, las cuales, describen conductas, interacciones, geometría y apariencia. Por su parte, los arcos se encargan de representar cómo se pueden modificar las propiedades de los nodos.

Modelo de flujo de datos

Los elementos mediadores en el proceso que realizan los MFD, sin duda son los nodos, que no son más que elementos sensores que reaccionan a ciertos eventos dentro del ambiente virtual, cuando el sensor recibe un evento cualquiera, fluyen los datos del sistema con la finalidad de activar a los nodos implicados en el proceso que gobierna dicho evento. Entonces, los datos activan a los nodos y los nodos procesan a los datos, pero hay que tener cuidado, ya que si un nodo no genera una buena salida, el flujo de datos se detendrá hasta que el nodo reaccione correctamente.

Entiéndase "sensor", cómo un recurso de VRML que provee mecanismos para que el participante pueda interactuar con objetos en el "mundo" virtual al cual ingresa. En una gráfica de flujo de datos completa, pueden darse dos situaciones: la primera es que los datos provengan de diversas fuentes y se dirijan hacia un mismo destino. La segunda situación es que los datos provengan de una misma fuente y se dirijan hacia varios destinos. A esto se le conoce cómo *fan-in* y *fan-out* respectivamente.

VRML 2.0 aun conserva las bases de VRML 1.0, estás bases las conforman elementos con los cuales se representa a los objetos de sus escenas, hablando de geometría, apariencia, luces y transformaciones, ahora bien, si a estos elementos les agregamos el MFD, el cual implica sensores, interpoladores, scripts y rutas, el resultado será lo que siempre se ha perseguido con la RV: dinámica e interactividad. Los eventos *eventIn* y *evenOut* representan para VRML procesos de entrada y salida respectivamente. A *eventOut* se le puede considerar consecuencia de *eventIn,* ya que eventIn actúa al recibir datos, estos actúan entonces sobre ciertos campos encargados de detectar el cálculo de funciones internas que una vez ya calculadas, generaran valores hacia otros campos del tipo *eventOut* quienes finalmente son propagadas a los demás nodos que actúan en el proceso.

Cabe mencionar que *eventIn* no recibe cualquier dato a menos que sea del tipo boolean, un color, una posición, una orientación u otro tipo relacionado. Está temática está orientada hacia la interactividad que VRML logra gracias a sus eventos, nodos y demás elementos que la hacen posible. Ahora un ejemplo, utilizando los nodos *PlaneSensor, TouchSensor, Billboard* y el evento *ROUTE.* Existen sistemas interactivos con un potencial en el que un usuario puede dar clic en un objeto y a continuación producirse algún evento deseado, en este caso, el usuario tiene la libertad de desplazar libremente a un objeto cama dentro del área delimitada habitación, esto se logra con el nodo *PlaneSensor,* el cual proporciona a cualquier objeto que se declare cómo su nodo descendiente la libertad de desplazamiento en un área de dos dimensiones paralela al eje Z.

Además de está propiedad, el usuario puede rotar cualquiera de los objetos que tenga está propiedad para adecuarlos a cualquier otra posición a la que sean desplazados gracias a que VRML opera de la siguiente forma: primeramente acciona el nodo *TouchSensor*, este nodo actúa cuando el usuario pasa el cursor del Mouse sobre el objeto que el diseñador del sistema consideró apto para realizar la rotación, también se elige un objeto situado en un punto estratégico y accesible, entonces al pasar el Mouse sobre la esfera el cursor toma la forma de un sol pequeño (en el visualizador Cosmo Player 2.1.1 ya que en Cortona 4.0 toma la forma de una manita), lo cual, indica que si se mantiene presionado el clic, se generará el evento de rotación, donde *TouchSensor* genera un valor que se dirigirá hacia el nodo *Billboard* encargado de asignar una rotación al eje especificado.

De alguna manera se tiene que indicar en VRML que cada vez que el usuario de un clic sobre la esfera, se genere la rotación del objeto en cuestión; cómo es un evento compuesto, se creo un script con lo cual se demuestra que VRML no está solo y que tiene poder de compatibilidad con scripts y muchas herramientas más. El script contiene las instrucciones que combinadas con los nodos anteriores establecerán un ángulo de inicio cero y una rotación de 90° cada vez que se de un clic sobre la esfera. A continuación las representaciones gráficas de lo descrito (figuras 1 y 2):

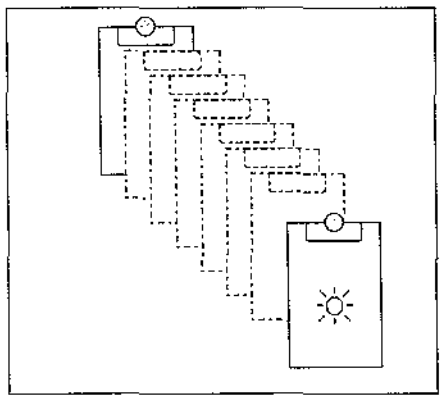


Figura II.5.1. Traslación de un objeto arrastrándolo a una nueva posición.

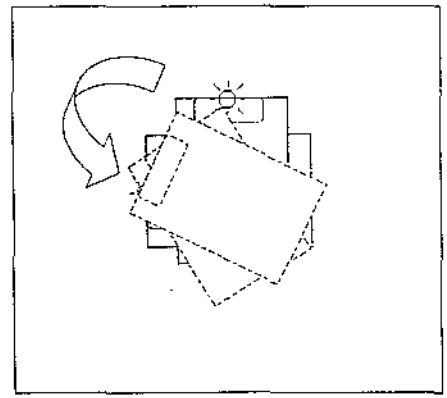


Figura II.5.2. Rotación de un objeto sobre su eje.

La representación de la gráfica del nodo de traslación es la siguiente (figuras 3, 4 y 5):

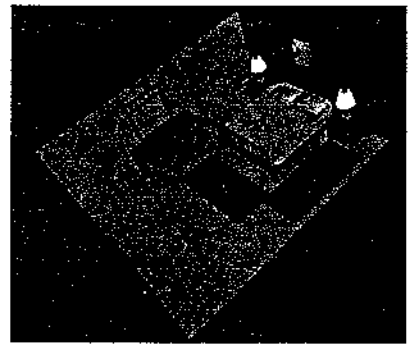


Figura II.5.3. Objeto cama en su posición original.

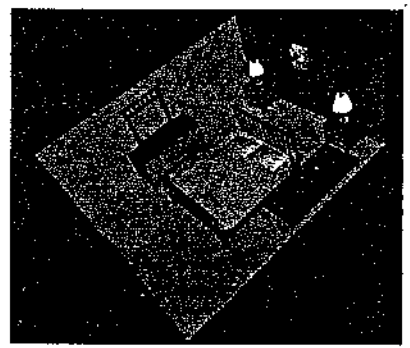


Figura II.5.4. Objeto cama desplazado con ayuda del nodo translation.

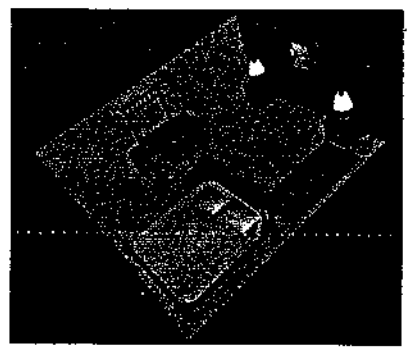


Figura II.5.5. Objeto cama desplazado a su posición final.

La representación de la gráfica del nodo de rotación aplicada en una recamara virtual es la siguiente (figuras 6, 7 y 8):

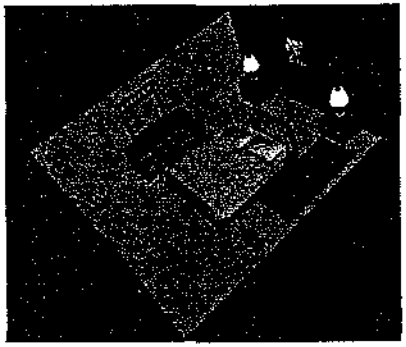


Figura II.5.6. Objeto cama en una posición inicial antes de ser rotado.

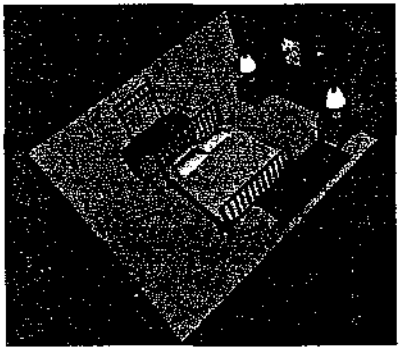


Figura II.5.7. Objeto cama rotado 90° sobre su eje.

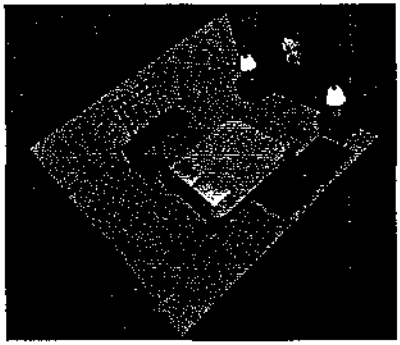


Figura II.5.8. Objeto cama rotado nuevamente 90° más, sobre su eje.

Este es el código en el que se muestra en el editor Xemacs la interconectividad entre nodos y su potencial:

```
#VRML V2.0 utf8

NavigationInfo {type "EXAMINE"}

#PLANE SENSOR                                   #Declaración del nodo
Transform {translation 0 0 0                    #Se establecen las coordenadas iniciales
rotation 1 0 0 1.57                             #Se rota el sensor al plano XZ
children [                                      #agrupación del planesensor y el objeto cama
DEF PS1 PlaneSensor {minPosition -1.6 0
                     maxPosition 1.7 5.5
                    }

#CAMA
DEF T1 Transform {translation 0 0 0             #Se declaran las coordenadas iniciales de los objetos agrupados (cama-esfera)
children[
Transform { translation 0 0 0
rotation 1 0 0 -1.57                            # rota el objeto 90° sobre el eje X
children [ Inline {url ["CAMASHAKER.wrl"]}      #Se invoca al objeto cama
        ]
                }

#ESFERA
Transform { translation 0 -2  -1.4              #Se establecen las coordenadas de la esfera que dará rotación a la cama
children [
DEF TOUCH9 TouchSensor {}                       #Se define el nodo Sensor
Billboard { axisOfRotation 1 0 0
children [
Billboard
{ axisOfRotation 1 0 0
children [
Shape { appearance Appearance { material Material { diffuseColor 0 1 0}}        #Se establece el color de la esfera cómo verde
        geometry Sphere {radius 0.07}
      } ] } ] } ] } ] } ] }

#SCRIPT
DEF miscript Script {                                   #Se define el script
eventIn SFBool isActive                         #Eventos booleano, temporizador, de traslación, y de valores flotantes activos
eventIn SFTime touchTime
eventIn SFVec3f translation_changed
eventIn SFFloat fraction_changed
field SFNode T1                                 #Se invoca al nodo T1
USE T1
field
SFFloat mirotacion 0                            #Se declara un campo de rotación inicial en 0°
field SFFloat angulo -1.57                      #Se declara un campo de rotación con valor de 90°
field SFFloat flag 0
directOutput TRUE
url"vrmlscript:
function touchTime()

mirotacion+=angulo;                             #Aquí es donde se rota 90 grados cada vez que se da clic en la esfera
T1.set_rotation=new SFRotation(0,0,1,mirotacion); }
function translation_changed(val)

valolr1=new SFVec3f(val[0],val[1],val[2]);      #Aquí es donde se invierten los ejes para rotar paralelamente al eje Z
T1.set_translation=new SFVec3f(val[0],val[2],val[1]);
} "}

ROUTE PS1.translation_changed TO T1.set_translation   #Ruteador del nodo PlaneSensor al objeto cama
ROUTE TOUCH9.touchTime TO miscript.touchTime          #Ruteador del nodo TouchTime al script
```

Cabe mencionar que este código es solo de la cama, cada objeto que tenga las mismas propiedades de desplazamiento y rotación, se componen de un código similar.

Características del Vrml 2.0 como MDF

- ✓ Puede describir AV (Ambientes Virtuales)
- ✓ Trata con URL's (Direcciones de Internet)
- ✓ Trata con scripts
- ✓ No tiene un amplio API (Interfaz de aplicación)
- ✓ Asigna a sus objetos propiedades como color posición y animación
- ✓ Basa sus datos de origen a partir de la gráfica de escena
- ✓ Tiene propiedades de jerarquía

Colisiones en VRML

En algunas fuentes consultadas se ha establecido que VRML no posee la propiedad de detectar colisiones (Virtual Worlds On the Internet, pag.17), particularmente quiero aclarar lo siguiente: VRML es el lenguaje base del presente trabajo y por tanto se puede afirmar que: sí permite la detección de colisiones pero entre la cámara y los objetos, al utilizar la navegación tipo "Go" descrita en el manual de usuario (visualizador Cosmo Player), sin embargo, es verdad que cuando programamos eventos en donde se pueden mover objetos libremente, no existe detección de colisiones entre estos objetos, aún así no se puede asegurar que no detecte colisiones al cien por ciento, esto dependerá del medio de navegación.

En las siguientes imágenes (figuras 9 y 10) se observa cómo al intentar atravesar un sofá virtual usando el botón del control "Go" , chocamos contra el objeto, es decir, solo llegamos hasta un cierto acercamiento y no podemos avanzar hacia delante, solo hacia los lados o rodeando el objeto.

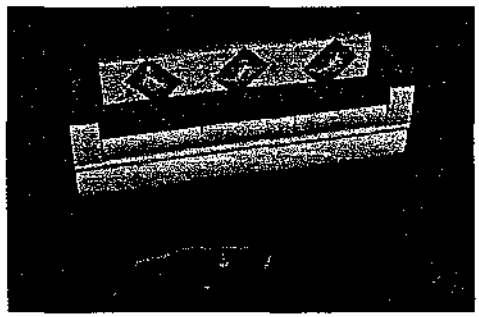


Figura II.5.9. Objeto sofá antes del acercamiento con el control "Go".

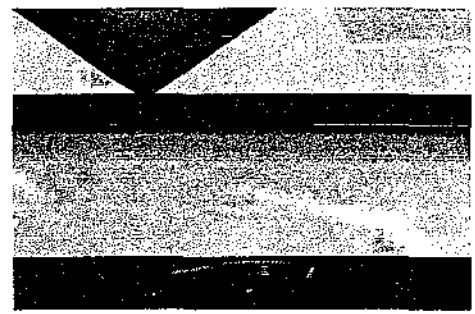


Figura II.5.10. Vista del acercamiento máximo al objeto sofá sin poder atravesarlo.

Por otra parte, al utilizar el botón de control "zoom", se logra atravesar con facilidad cualquier geometría del ambiente virtual (figuras 11 y 12). Si en cambio cómo en la presente aplicación, ciertos objetos tienen la facilidad de desplazamiento, tampoco existe colisión entre ellos, por lo que se vera cómo si fueran fantasmas que se atraviesan, a continuación un ejemplo:

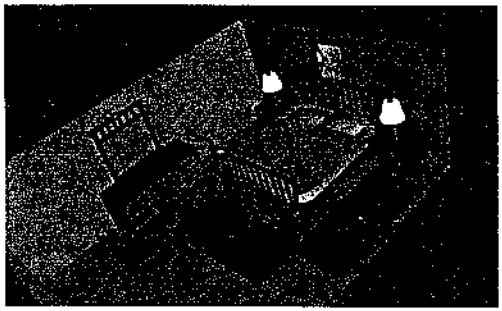


Figura II.5.11. Objeto cama en su posición inicial.

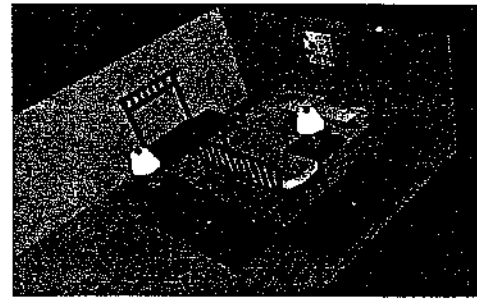


Figura II.5.12. Objeto cama colisionándose con los objeto buró y luna.

Continuando con sus características, se tiene que VRML soporta muy bien el cambio de propiedades, debido a que cuenta con la propiedad "*graph structure*" donde cada elemento del tipo textura y geometría son elementos adecuados para incluirlos en el flujo de datos. Otro punto a favor de VRML son sus extensiones, con las que se puede implementar un API[2], la cual permite el control externo de una escena VRML al instalarla en un navegador, este control lo puede tomar aplicaciones diversas para alterar la gráfica de escena independientemente del modelo de flujo de datos.

En cuanto a colisiones, VRML puede realizar prototipos (tratándose de colisiones entre objetos), en cambio al tratar con sensores de información puede requerir de clases, un navegador API, o podría generar nodos que modelen el cuerpo del participante, en donde está última opción proporcionaría ventajas de despliegue en los navegadores trayendo cómo consecuencia que el usuario tenga un mayor rango de modelado de su ambiente. Tratándose de mapeo de acciones y técnicas de interacción VRML no las puede ejecutar sin un nuevo navegador que interprete los nodos, ya que el navegador es la base sobre la que trabaja el visualizador de este lenguaje.

Vrml : ¿réplica o representación?

La realidad virtual se compone de dos palabras contradictorias: virtual y realidad. Webster define que la realidad es algo que existe independientemente de las ideas que lo conciernen y lo virtual como el estar por efecto, pero no por hecho; en un estado de posibilidad dentro de un ser real. Hablando en terminologías, virtual se define como: (adj.) Indica que una entidad imita las características de otra. En el contexto de un mundo virtual, cualquier objeto en ese mundo puede decirse que es virtual. La realidad virtual se describe también como un medio compuesto de simulaciones interactivas en una computadora qué perciben la posición del participante y reemplaza o aumenta los estímulos a los otros sentidos dando la impresión de estar inmerso o presente en la simulación.

[1] API. Biblioteca específica con acuerdos de llamado para que un programa pueda enlazarse y crear una aplicación ejecutable. Por ejemplo, OpenGL, Reality Labs, renderWare, etc.

Figura II.5.13. Lentes 3D para observar mundos virtuales interactivos.

La RV se puede obtener en películas, radio, televisión, computadoras, simuladores de vuelo o computacionales, juegos entre otros. Ahora se definen algunos elementos que conciernen a la realidad virtual:

- Un mundo virtual se define de la siguiente manera:

    1. El contenido de algún medio;
    2. Un espacio que existe en la mente de su creador - manifestado en algún medio -.
    3. La descripción de una colección de objetos en un espacio considerando las reglas y relaciones que gobiernan a estos objetos.

- Ambiente virtual también se define de forma parecida:

    1. Un mundo virtual.
    2. Una instancia de un mundo virtual presentada en un medio interactivo como una realidad virtual.

- Un Avatar:

    1. Objeto virtual usado para representar a un participante en un mundo virtual. Esta representación visual puede tener cualquier forma.
    2. El objeto representa o envuelve a un participante.
    3. Esta palabra proviene de la palabra que indica una deidad hindú es terrenal o encamación.

- Otras definiciones de RV:

    - Howard Rheingold, en su trabajo "virtuelle de Réalité" (1993), define el mundo virtual como "un mundo calculado por una computadora a la que nosotros comandamos no escribiendo programas, sino con gestos naturales, para viajar el, explorándolo, mirándolo y usando las manos para manipular los objetos que contiene."

    - Steve el Aukstanalkis & David Blatner: "para los humanos, la realidad virtual es querer visualizar, manipular e interactuar con las computadoras y datos extremadamente complejos".

- Myron Krueger en 1991: "La promesa de realidad artificial no es reproducir la realidad convencional o para actuar en el mundo real. Es precisamente la oportunidad de crear la realidad sintética para lo cual no hay antecedente reales, es decir, realidades que nunca existieron."

Si se desea una definición más satisfactoria:

"Simulación por computadora de un ambiente que utiliza imágenes 3D y dispositivos externos, como los guantes de los datos y cascos, para permitirle a un usuario interactuar con la simulación. Los usuarios pueden moverse a través de los ambientes de realidad virtuales igual que como lo harían en el mundo real, caminando a través de estructuras e interactuando con los objetos de este ambiente en el tiempo real."

El desarrollar ambientes virtuales, se ha visto cómo un trabajo de diseño en el que lo que se construye además de significativo debe ser lo más parecido al mundo real, es decir una replica. Cuando un usuario, ya sea desarrollador o no, interactúa con un ambiente virtual, su primera impresión es sentirse inmerso en una reconstrucción de la vida real, pero lograda gracias a la computadora e inventiva del desarrollador. Está representación de la realidad quiérase o no, siempre será una virtualidad o más crudamente la podemos llamar una "falsedad". Suena agresivo el termino "falsedad", pero debemos aceptarlo, y es que queramos o no, la RV es una replica de la realidad pura.

Al desarrollar estos ambientes tenemos cómo regla el implicar la selección, la abstracción y la libertad que cualquier diseñador tiene al construir una representación, está libertad tiene la gran ventaja que hacemos divertir cómo niños: el poder siempre hacer cualquier cambio, ya que nosotros somos sus creadores, tomemos entonces, nuestro papel de diseñadores. Podemos hacer una analogía para dejar más clara está situación, el diseño de un ambiente virtual es cómo el teatro, donde se recrea un escenario, personajes, objetos, posiciones de los mismos, etc, ¡somos libres!, pero recordemos, está libertad implica una responsabilidad, la de crear una representación lo más cercana a la realidad.

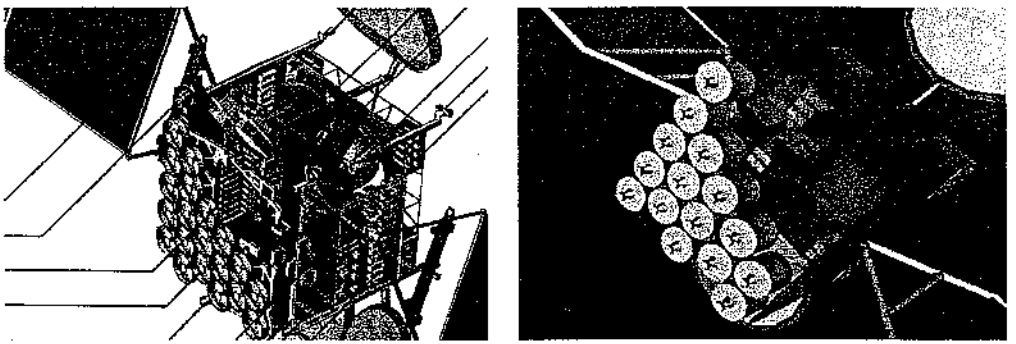


Figura II.5.14. Imagen de la estructura interna de un satélite (izq.). Representación de la estructura en forma tridimensional en VRML 2.0 (der.).

En estas imágenes se puede apreciar del lado izquierdo una representación real de la estructura del satélite Morelos 1, y a la derecha como se logra una representación virtual en una computadora gracias a VRML. A primera vista parece no haber diferencia, pero la ventaja es que en la izquierda se tiene una imagen estática y en la derecha un objeto 3D con movimiento tridimensional.

III. DISEÑO DE LOS ESPACIOS FUNCIONALES.

III.1. Ambientes virtuales basados en VRML.

Para colocar un ambiente virtual en Internet, necesitamos de varios elementos, primeramente una computadora personal con características definidas (ver "capacidades del sistema"), que pueda soportar al ambiente virtual en cuánto a velocidad de procesamiento y gráficos, también un software que permita la aplicación en la que un usuario pueda interactuar con el ambiente virtual y finalmente un código estandarizado para tal fin.

Estas razones hicieron que VRML no tuviera buena suerte al nacer, ya que no estaba en su tiempo - por llamarlo de alguna manera -, en este entonces, hablando del año 1995, los usuarios no tenían en sus computadoras el suficiente poder de gráficos, procesamiento y velocidad de Internet para garantizar su éxito total.

Para cualquier desarrollador que ha programado y generado gráficas por computadora, VRML consiste en el poder de cargar un elemento virtual en Internet, para lo cual debe haber un motor que tenga la capacidad de codificar información en tres dimensiones a un archivo que se pueda trabajar en el Web. También es importante saber que en la estructura de cualquier archivo de VRML, existen elementos que componen una escena gráfica jerárquica con la que se pueden describir las geometrías 3D que VRML utiliza en la creación de sus objetos, estas geometrías pueden ser texturas, iluminación y efectos de sonido.

Pero VRML tiene más potencial, los elementos de una escena pueden ser manipulados e incluso programar animaciones o eventos, gracias a que maneja lenguaje script; así, a través de eventos de tiempo y mensajes diversos se comunican sus nodos.

La compatibilidad con Java y JavaScript queda de manifiesto al otorgar la posibilidad de sumar o restar nodos de escena gráfica, generar eventos y recibir notificaciones de los mismos. De manera general VRML funciona de la siguiente forma: cuando se toma un archivo (.wrl) del Web es interpretado por un Visualizador VRML que dibuja la escena gráfica y soporta la navegación que realiza el usuario a través del ambiente virtual.

En cuánto a la compatibilidad con Java, tenemos que VRML puede trabajar conjuntamente de la siguiente manera:

- Creando nuevas geometrías.
- Enviando eventos a los nodos en la escena.
- Leyendo los valores más recientes de eventos enviados desde nodos en la escena.
- Recibiendo notificaciones cuando los eventos son enviados desde nodos en la escena.

Para poder desplegar un mundo VRML se necesita un visualizador que interprete el código para poderlo cargar en el Web, existen varios (véase "Introducción"), afortunadamente hasta el momento existen varios gratuitos y compatibles con los navegadores más comerciales conocidos como lo son Netscape e Internet Explorer. El Visualizador VRML más popular y agradable es el de sgi: Cosmo Player 3D en comparación con otros conocidos como el de Intervisa, el de Netscape, el de ParallelGraphics y el de Blaxxun entre los más conocidos (véase cuadro IV.1.2 comparativo en "Antecedentes y evaluaciones").

Hasta 1999, las limitaciones de hardware de la mayoría de las máquinas conectadas a Internet y la falta de investigación de la tecnología en red VRML, fueron un obstáculo para que diferentes organizaciones pudieran experimentar los beneficios en red del VRML. Los pocos sistemas que habían sido desarrollados, tenían un número de usuarios menor a 16 en la mayoría de los casos, cuando de conectar mundos VRML en red se trataba.

Para generar ambientes VRML más elaborados, en ocasiones se requerirá de la ayuda de lenguajes como Java o lenguaje C, cuestión que sería difícil de lograr sin la compatibilidad ya mencionada. Existen empresas e instituciones que han trabajado al VRML en red y lo han combinado con Java, logrando distintas aplicaciones como el uso de avatares y mundos con objetos solitarios o acompañados de presencias, algunas de ellas: La Escuela Naval de Posgraduados, en Estados Unidos; la comunidad "Mundos Vivientes", Blaxxun Interactive [2], Sony Community Place [7], Open Community, VNet por mencionar sólo algunas. Cabe mencionar que el sistema que complementa este trabajo no ha necesitado hacer uso más que del lenguaje VRML puro.

Existe un proyecto que utiliza un nuevo protocolo, el Protocolo de Transferencia de Realidad Virtual (VRTP) que consiste en intentar hacer de VRML una aplicación más general, pudiendo llegar a más computadoras conectadas en red y por ende permitiendo interoperabilidad y compatibilidad entre los usuarios y sus mundos en Internet. Esto hace que puedan ser más extensos y mejores. El objetivo para la creación de este protocolo es ayudar a soportar la transferencia de cualquier tipo de información de los ambientes virtuales usando una misma estructura. El primer elemento que compone al VRTP es el Procesamiento del Estado de la Entidad, que son mensajes integrados por estado, acontecimiento y la información de control [John Vince, 1998].

Otros son los *Objetos Obstaculizadores*, datos grandes que se oponen a la transmisión de datos, a menos que esta transmisión sea orientada en una conexión confiable. Los datos se entregan como respuesta del protocolo HTTP respondiendo a una petición del indicador de la red. Los Indicadores de red pueden ser usados para contestar preguntas repetidas por otros miembros del grupo, en vez de los servidores. A diferencia de las interacciones ligeras, los indicadores de red no contienen a un objeto completo, solamente una referencia a un objeto. Por último, los *flujos en tiempo real*, que son, el vídeo vivo, el audio, las imágenes secuenciales de los gráficos, o el tráfico continuo del flujo se tienen que entregar en tiempo real, ordenando y sincronizado. Estos elementos se ponen en ejecución típicamente usando los canales del multicast[1].

La estructura del VRTP se compone de una colección de módulos de protocolos y un protocolo de aplicación que provee la conectividad necesaria para un ambiente virtual cliente. Al conjuntar a todos los elementos de comunidades, códigos de optimización, protocolos de comunicación y estandarización.

Ahora bien ¿Cuál ha sido una de las aplicaciones que más éxito ha tenido al hablar de RV en red? sin duda han sido los juegos en computadora, sobre todo cuando de jugar en red se trata. Si una aplicación cumple con los requerimientos de interactividad y ha tenido más éxito estos sin duda son los juegos de Internet. Estos comúnmente se cargan en la red de dos formas, cargado en red o libre en servidor. Aquí la importancia radica en compartir todos los juegos para demostrar la habilidad de participar en un combate con un oponente que se encuentra en una localidad remota, de muy diversas formas, quizá con una ventana de chat para comunicarse y coordinarse o en un mapa que simula un escenario de combate entre muchas otras.

[1] Emisor y receptor de información a través de varios canales.

Si se tienen deficiencias de hardware y software, los juegos sufren serios problemas de retardo. Por ejemplo, en un juego llamado Age Of Empires II: los soldados deben mostrar su habilidad de pelea para jugar con un oponente en red en donde existen imperios y cada uno de estos debe exterminar a su enemigo de muy diversas formas. Con un retardo de algo próximo a 300 y 400 ms, no habría oportunidad de atacar al oponente como se quisiera sólo cuando el juego se ejecutaba conectado sobre una red LAN relativamente a alta velocidad de transmisión. Aunque después de varios análisis en jugadores, se concluyó que la mayoría de ellos aprender a convivir en el juego con el retraso de la red y es que ahora cualquier adolescente aficionado a los juegos en red, es capaz de adaptarse a las situaciones con tal de no perder la contienda.

Figura III.1.1. Presentación del juego en Red y de estrategia: "Age of Empires".

Otra experiencia se recogió en una muestra de cómputo, donde se conectaron en red, seis máquinas con el famoso juego Resident Evil. Se pudo constatar que el retraso y la resolución se vuelven problemas, que incluso a veces, se pueden convertir en aliados de los participantes con mayor capacidad de reacción al encontrarse en situaciones donde puede quedar acorralado por sus adversarios y aprovechando el retardo, poder escapar.

Figura III.1.2. Escena del juego 3D, Resident Evil (izq) y personaje de "Final Fantasy" (der).

Un ejemplo más y de gran demanda entre los jugadores en red, es el fútbol, En esta categoría se encuentra con mucha aceptación a las series que salen año con año de FIFA, este juego ha sido desarrollado por la empresa EA Sports y de verdad que es un juego muy entretenido, donde para jugar en red se necesita de dos o más PC's para poder jugar en tiempo real, aunque también pueden conectarse los contrincantes a través de la línea telefónica a cambio de unos segundos más de retardo.

El objetivo final es el ya conocido por lo aficionados al fútbol, anotar en la portería del contrario la mayor cantidad posible de goles en un tiempo determinado, cabe mencionar que FIFA, es un juego bastante real ya que permite cambio de alineación, estadios, apariencia, clima, tiempo, rudeza, lesiones, control y demás detalles que se tienen en un partido real.

Figura III.1.3. Escena del juego FIFA Soccer, donde se permite personalizar el uniforme y apariencia de los jugadores de cada equipo.

Aún queda muchísimo por desarrollar, pero es muy notorio el avance que ha habido en el ámbito de la RV local y en red, es por esto que a los usuarios que no se dedican a estos avances, sólo les resta esperar, aportar sugerencias y hacer saber de sus inquietudes e incomodidades con los sistemas en red ya existentes para que sean corregidos y mejorados.

III.2. Capacidades del sistema "Apartamento Virtual".

Requerimientos

Para obtener los mejores resultados de interacción con esta aplicación se requerirá que el equipo de cómputo utilizado tenga las siguientes características mínimas:

- Procesador Pentium 75 Mhz o más rápido.
- Sistema Operativo Microsoft Windows 95, 98, Server 2000, Me, Linux, Unix o equivalentes.
- 16 Megabytes de RAM como mínimo.
- 14 Megabytes de espacio libre en Disco Duro
- Monitor Super VGA a 256 colores de 15 pulgadas o superior.
- Navegador de Internet versión mayor a Explorer 3.0, Netscape 4.0 o equivalente.
- Visualizador o Plug-in de VRML.

Es natural que no se cuente con un Visualizador de VRML, el cual es necesario para poder ejecutar el espacio virtual al que hacemos referencia, esta es una de las primeras carencias que debemos cubrir para tener la aplicación completa.

Por esta razón en la página web que alberga este espacio funcional se hace referencia a enlaces en forma de iconos que representan tres de las compañías más representativas creadoras de visualizadores VRML como los son Cosmo Player, Parallell Graphics y Blaxxun. Al hacer clic sobre el icono de la compañía elegida automáticamente se abrirá una nueva página que hará referencia al portal del Visualizador, de donde se podrá descargar dicho software en forma gratuita.

El espacio funcional que complementa el presente trabajo es un espacio con las dimensiones estándar, pero el diseño de los interiores será interactivo. Las capacidades de esta aplicación son:

1. **Recorridos fijos** El usuario dispondrá de recorridos fijos programados, con la finalidad de que tenga una navegación más adecuada o de otra manera por si tiene dificultades de navegación.

2. **Cambio en la posición de los muebles** Se puede cambiar la posición de los muebles previamente elegidos para dar diferentes opciones de presentación a las habitaciones, según el gusto del usuario.

3. **Ficha técnica de cada mueble** En cada mueble se proporciona su ficha técnica con sólo hacer clic en una etiqueta, esto con ayuda del mismo código VRML que invoca una sub-ventana Web de la aplicación.

Partiendo de esta idea, otra opción que se puede desarrollar con VRML en relación a espacios funcionales es un museo virtual con la estructura estándar de la mayoría de los museos y en el se incluirían las siguientes capacidades:

1. **Recorridos fijos** El usuario dispondría de recorridos fijos programados, con la finalidad de que tenga una navegación más adecuada o de otra manera por si tiene dificultades de navegación.

2. **Elección de mobiliario** El usuario podría agregar elementos necesarios para montar una exposición tales como: pedestales, vitrinas y mamparas al dar clic en el menú del lado izquierdo del mundo virtual. En este caso, los objetos aparecerán en una posición predeterminada desde la cuál podrán ser arrastrados a nuevas posiciones adecuadas, y de esta manera ir dando forma al museo.

3. **Cambio en la posición de mobiliario** Cada objeto podría ser desplazado libremente por el museo dando clic al objeto y arrastrándolo. De manera similar a como sucede en el apartamento, al dar clic sobre el objeto, el cursor cambiará su forma indicando que se puede arrastrar el objeto.

4. **Ficha técnica de cada mueble** Se puede observar, que en esta aplicación no sería necesaria la opción de proporcionar una ficha técnica de los objetos debido a que solo se trata con elementos mobiliarios que se supone todo museo debe tener, entendiéndose de esta manera que la finalidad de la aplicación es puramente de prediseño.

Otro ejemplo, por solo mencionarlo podría ser desarrollar un centro de exposiciones con las siguientes capacidades:

1. **Recorridos fijos** Este espacio funcional no requeriría de recorridos predefinidos ya que de inicio solo se vería un área delimitada vacía que iría tomando forma hasta que vayan apareciendo los objetos de la exposición. Esta situación no permite saber de inicio que stands se crearán y donde se ubicarán impidiendo de esta manera establecer recorridos virtuales.

2. **Elección de mobiliario** En un menú del lado izquierdo de una página Web, el usuario daría clic en la imagen del objeto deseado y al momento aparecería al centro del escenario dicho objeto listo para ser desplazado hasta la opción de preferencia.

3. **Cambio en las posiciones del mobiliario** Estos desplazamientos se realizarán dando clic sobre la superficie del objeto en cuestión y sin soltar el botón del mouse arrastrarlo hasta la posición deseada. Para rotarlos se procederá de la misma forma que en el apartamento.

4. **Ficha técnica de cada mueble** Se puede observar, que en esta aplicación no sería necesaria la opción de proporcionar una ficha técnica de los objetos debido a que solo se trata con elementos mobiliarios que se supone toda exposición debe tener, entendiéndose de esta manera que la finalidad de la aplicación es puramente de prediseño.

Con proyectos de este tipo se pueden desarrollar más ideas sobre espacios funcionales que evitarían gastos innecesarios en la etapa de planeación y diseño de los mismos. El apartamento como toda aplicación bien planeada, este debe tener una presentación acorde a la aplicación. En este caso consiste en una página Web la cuál contiene diferentes secciones referentes a la presente aplicación. La página principal contiene un menú con botones, los cuales conducen a diferentes páginas con explicaciones referentes a la aplicación, además de una introducción. Los botones hacen referencia a temas como: antecedentes e historia del VRML, catálogo de muebles de la aplicación, asesor virtual, enlaces a sitios VRML y sitios de descarga de visualizadores.

En cuánto a los antecedentes e historia, se hizo una recopilación de información sobre el comienzo de este poderoso lenguaje de modelado en 3D, así como las bases que dieron origen a este lenguaje y una reseña de los pormenores de su nacimiento. Ahora se explica en que consiste cada una de estas características.

Cambio en la posición de los muebles

El usuario puede elegir libremente la posición de los muebles al tener la posibilidad de rotarlos y cambiarlos de lugar para darle la capacidad de no quedar encajonado o sujeto al diseño del programador y así tener la opción de modificar su casa habitación como él desee, esto es parte de la magia que se logra con VRML. Cada mueble tiene una posición ya predefinida, es decir, un lugar y un ángulo de posición determinado, al moverlo, es probable que además de su posición se tenga que rotar para que este quede en un nuevo lugar en posición adecuada. Decimos que es probable que se tenga que rotar, porque no a todos los muebles se les deberá rotar, por ejemplo, si se trata de un taburete o una lámpara, al ser estos objetos, simétricos, será indiferente realizar una rotación en ellos, ya que se verá que no tendrá importancia haberlo hecho. Cabe mencionar que esta aplicación se ha programado para modificar la posición de los muebles dentro de un rango de superficie predeterminado coherente, esto con el fin de que el usuario no se preocupe de que al mover el objeto, éste vaya demasiado lejos.

Ficha técnica de los muebles

Para saber que características tienen los muebles que componen el espacio funcional, se usa una capacidad de la presente aplicación de tipo informativa, la cual permite al usuario obtener los datos del mueble que elija de cualquiera de los que componen su espacio virtual. Estos datos contemplan las dimensiones en general del mueble elegido, su color, su textura, el modelo, tipo de mueble e incluso su precio en el mercado. Este último dato es importante para que dar oportunidad a que el comprador lleve su propia cotización o sepa si este mueble le conviene o no con base en su presupuesto. Para activar esta propiedad, el usuario sólo necesita hacer clic sobre el nombre del mueble en el menú correspondiente de la página Web, en seguida de hacer clic sobre ella, aparece una ventana Web informativa con los datos técnicos (ya mencionados) del mueble elegido.

Expectativas

Esta aplicación puede ser de mucha utilidad para alguna o algunas de las empresas importantes que actualmente operan en el país en el ramo de muebles e inmuebles, debido a que al navegar por sus páginas, se observa que pueden crecer en aspectos atractivos al consumidor; por ejemplo mostrar buenas cotizaciones o interactividad entre el posible comprador y los productos.

Se puede denotar, que las empresas del ramo, han invertido más en comerciales televisivos que en cualquier otra forma de comercialización. Una buena opción es invertir también en un sistema que permita consultar precios y mobiliario desde la comodidad del hogar; además cada día son más las familias que gozan de una computadora en casa y esto debe ser observado por las empresas. No se debe olvidar que Internet cada día se convierte en una opción más cómoda de compra, existen ejemplos como los libros de Amazon, artículos Nike, cotizaciones de autos en la página de VolksWagen entre otros.

Si alguna persona se interesará en saber más acerca de este lenguaje de realidad virtual, también se proporciona una sección donde habrán vínculos o ligas hacia sitios referentes a páginas con información de VRML, donde se puede encontrar desde información de los inicios del lenguaje, hasta manuales y tutoriales para aprender a programar sitios VRML.

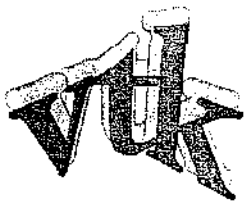
III.3. Gráficas de los objetos VRML.

Además de la temática, se desarrolló una aplicación complementaria y los diferentes objetos que lo componen de manera programada en código VRML puro, es decir, no se utilizó un modelador de objetos VRML o algún programa de interfaz para crearlos, esto debido a que se consideró más cómodo y provechoso en el aprendizaje y práctica del lenguaje. Varios objetos que componen el espacio funcional tienen complejidad para ser creados con las geometrías normales de VRML ya que en ocasiones requieren ser deformados o toman aspectos curvos o irregulares, para solucionar este problema, VRML cuenta con una herramienta que permite graficar un objeto en 2D y posteriormente asignarle volumen y convertirlo en 3D, este nodo es llamado "Extrusión".

Para que VRML logre lo anterior se siguen una serie de pasos: el usuario necesita primeramente, graficar el objeto en 2D para tenerlo en un plano, para tal fin se pueden recurrir a distintas herramientas: una hoja de papel milimétrico o cuadriculada sencilla o incluso un programa que permita dibujar sobre una gráfica el objeto plano en 2D y una vez logrado esto, permita graficar sobre él los puntos que lo componen e ir trasladando sus coordenadas al código VRML.

En este trabajo se recurrió a un software de graficación creado en el Departamento de Visualización de la Dirección de Servicios de Computo Académico de la UNAM denominado **VisModel 3D.** El cual fue creado con la unión de software de alta calidad como lo son: Java y VTK. La programación se llevó a cabo en lenguaje Java puro y el proceso de visualización tridimensional se logra gracias al software de visualización VTK. Se manejo la versión 1.2 de Java y 4.0 de VTK. Como parte del presente trabajo se desarrollaron módulos sobre *VisModel3D*, para modelar y generar archivos de objetos 3D para VRML.

VTK (Visualization Toolkit) es un sistema que trabaja con gráficos 3D y Visualización. La versión más actual es la 4.0 y su descarga esta disponible en su sitio oficial [29].



VTK es código abierto, libremente disponible para gráficos 3D, procesamiento de imagen y visualización. VTK se compone de una biblioteca de C++ y fragmentos de código en Tcl/Tk, Java y Pitón. VTK es compatible con varias plataformas para su instalación como Unix, PCs (Windows 98/ME/NT/2000/XP) y Mac OSX Jaguar o posteriores. Algunas de sus características son:

- Es código libre.
- Fácil para crear las aplicaciones de visualización gráfica.
- La interfaz del usuario puede crearse rápido con Tk o Java GUI clasifique las bibliotecas.
- Como apoyo una paleta extensa de 3D widgets.
- Muchos algoritmos avanzados muy útiles.
- Software integrado.
- Probado en aplicaciones del mundo real... no en código académico.

A continuación, un ejemplo paso a paso de este proceso y ejemplos de las gráficas creadas en esta aplicación virtual con el fin de que los usuarios lo tomen como una herramienta alternativa importante en el momento de crear mundos y objetos virtuales en VRML.

Como primer ejemplo se tiene un brazo de uno de los sillones de las salas mostradas en la aplicación, el cual debido a su forma irregular necesitó ser representado en 2D como se muestra en la figura III.3.1, para posteriormente poder crearlo en VRML. Para esto primero se consultan páginas de Internet, revistas o cualquier fuente que presente imágenes relacionadas con lo que queremos crear; en este caso muebles y más específicamente un sillón. Una vez hecho esto, se observa la imagen cuidadosamente en sus detalles para saber qué se puede crear con los nodos comunes de VRML y qué no. Las partes no complicadas se realizan primero, es decir, las que se pueden crear con las geometrías, para después pasar a la creación de las complicadas con ayuda del nodo "Extrusion".

Figura III.3.1. Brazo izquierdo de un sillón del sistema.

Como en este caso, el brazo del sillón es de forma irregular, dibujamos esta forma lo más parecida a como queremos que VRML la interprete, es importante saber que para tener mayor precisión en el objeto irregular que VRML creará, se deben graficar la mayor cantidad de puntos como se observa en la figura III.3.2.

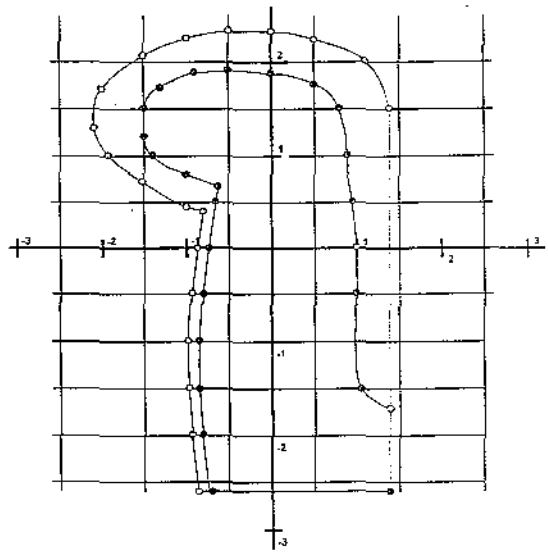


Figura III.3.2. Gráfica en VisModel 3D que muestra las coordenadas de la mayoría de puntos que componen al brazo.

Después de haber graficado los puntos en VisModel3D, se calculan las coordenadas de cada uno de los puntos que componen al objeto, para trasladarlas al nodo Extrusion donde el nodo "CrossSection" las grafica en 2D y después sus nodos descendientes "scale" y "spine" le dan tamaño y volumen. A continuación se observa un fragmento del código donde se escriben las coordenadas anteriores y el nodo *extrusión* hace su trabajo.

```
geometry Extrusion {crossSection [ 1.0   0.0
                                   0.95  0.5
                                   0.9   1.0
                                   0.8   1.5
                                   0.5   1.75
                                   0.0   1.88
                                   -0.5  1.9
                                   -0.9  1.88
                                   -1.33 1.75
                                   -1.5  1.5
                                   -1.5  1.2
                                   -1.4  1.0
                                   -1.0  0.8
                                   -0.6  0.7
                                   -0.65 0.5
                                   -0.75 0.0
                                   -0.8  -0.5
                                   -0.85 -1.0
                                   -0.85 -1.5
                                   -0.8  -2.0
                                   -0.7  -2.6
                                    1.4  -2.6
                                    1.4  -1.7
                                    1.05 -1.5
                                    1.0  -0.5
                                    1.0   0.0

                                   ]

spine [0 0 0, 0 1 0, 0 7.3 0,
                                   ]
                                  solid FALSE
                                  creaseAngle 1
                                  convex FALSE

                                 }
```

En la figura III.3.3 se observa el resultado de la interpretación de las coordenadas por el nodo de VRML conjuntando el brazo con demás elementos que componen un objeto sillón.

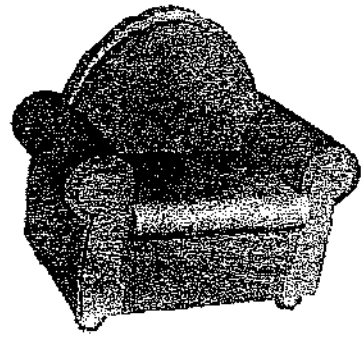


Figura III.3.3. Sillón programado en VRML con elementos creados con el nodo Extrusión.

Otro ejemplo lo encontramos en el sanitario del apartamento, ya que la forma que tiene la tapa es una geometría que VRML no puede crear a menos que utilice la unión de puntos en 2D y posteriormente *Extrusión* los gráfica y da volumen. Aquí se observa que la forma es de un trapecio, pero con las esquinas redondeadas, los puntos graficados son unidos por el nodo "crossSection" formando el objeto irregular, se le asigna volumen con "spine" y finalmente se escala (si es necesario) con el nodo "scale". Al ser este un objeto sencillo de crear, se establece más a fondo el proceso:

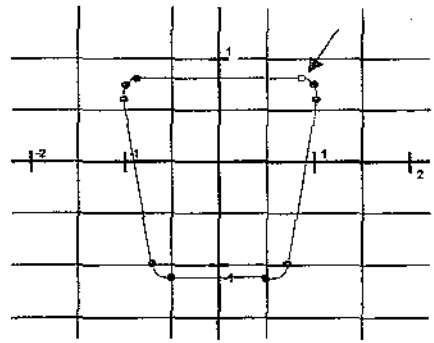


Figura III.3.4. Tapa del sanitario graficada en VisModel3D para encontrar las coordenadas.

Observando la gráfica, tenemos las siguientes coordenadas tomando como punto inicial al señalado por la flecha y en el sentido de las manecillas del reloj:

```
 0.88  1.3
 1.0   1.23
 1.0   1.1
 0.7  -1.0
 0.5  -1.12
-0.5  -1.12
-0.7  -1.0
-1.0   1.1
-1.0   1.23
-0.88  1.3
 0.88  1.3
```

Obsérvese que el punto final es el mismo punto inicial, esto es necesario para que "CrossSection" cierre la unión de puntos del objeto. Estos puntos se reescriben tal como están, dentro de "CrossSection", con lo cual se genera la forma irregular en 2D (ver en la figura III.3.5 de vista superior) pero con una altura que VRML asigna por regla y que en la mayoría de ocasiones no es la deseada.

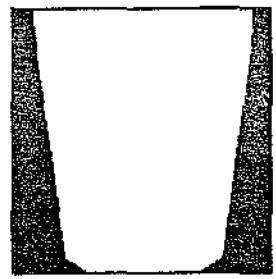


Figura III.3.5. Tapa en código VRML con una altura que el visualizador asigna automáticamente.

Utilizando el atributo "spine" se le da volumen al objeto. Al establecer cada coordenada que representará la altura, implícitamente de manera imaginaria se van creando algo como capas, donde cada capa tendrá coordenadas en 3D que modifican al objeto en su trayectoria horizontal, vertical y de fondo, dándole así volumen. Tenemos con esto la figura III.3.6:

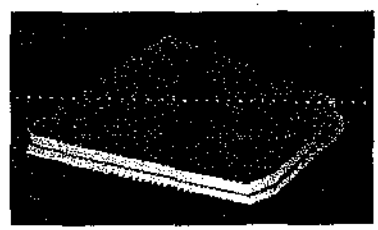


Figura III.3.6. Tapa con volumen asignado por el atributo *spine*.

Esta tapa, se combina con los demás elementos como el tanque, palanca y la base para formar todo el retrete, que finalmente queda como se ilustra en la figura III.3.7.

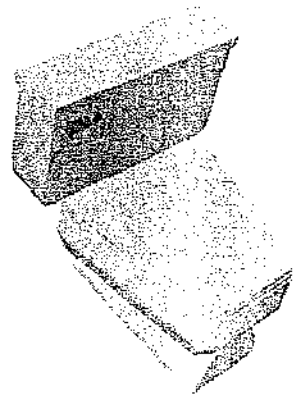


Figura III.3.7. Retrete completo donde las formas irregulares como la tapa, se crearon con *extrusion* y se graficaron en *VisModel3D*.

Uno más de los ejemplos interesantes del trabajo y que fue producto de imaginar como explotar estas herramientas de la graficación, fue cuando se tuvo que crear la puerta de la cocina, ya que este tipo de puertas regularmente no son totalmente sólidas en su estructura, sino que llevan el grabado de una cuchara, tenedor o algo alusivo a un implemento de cocina; el espacio funcional de este proyecto no podía ser la excepción. He aquí como se gráfico la puerta, para posteriormente crearla en 3D. Observemos que la cuchara tiene formas curvas, excépto la base y que debe ser creada de una sola pieza.

Una vez más se gráfica el objeto en VisModel 3D para encontrar sus coordenadas como se muestra en la figura III.3.8.

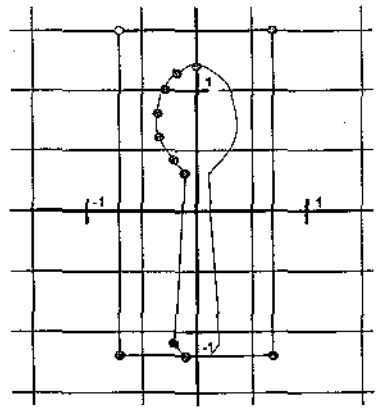


Figura III.3.8. Objeto cuchara graficado en VismModel3D.

En seguida, se capturan las coordenadas en el código VRML donde finalmente *extrusion* hace su trabajo (figura III.3.9).

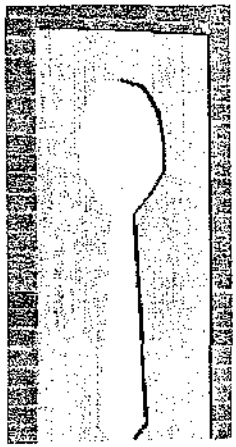


Figura III.3.9. Objeto puerta de la cocina donde la forma irregular interna es posible con el nodo *extrusión* y el software de graficación VisModel3D.

IV. IMPLANTACIÓN DE VRML Y SUS EVALUACIONES

IV.1. Antecedentes y evaluaciones.

Internet es el sistema que nos permite explorar y conocer casi todo, aún con estas maravillas no podía ser totalmente perfecto, tiene limitaciones y una de las más importantes y que nos trajo hasta aquí es la interactividad 3D. Las páginas actuales de Internet son en su mayoría estáticas, es decir, son páginas que no generan una interacción entre la computadora y el usuario. Un 80% de las existentes son de este tipo, basta con navegar dos horas visitando sitios con diferentes temáticas al azar y confirmaremos que al usuario no le queda más que ser espectador de texto, cuadros e imágenes fijas e inalterables. La Realidad Virtual (RV), es una alternativa para terminar poco a poco con esta monotonía.

Se podría pensar que existen aplicaciones como JavaScript o Flash que proporcionan interactividad, pero no es totalmente cierto, JavaScript genera applets, ventanas, etc; pero no son analogías de mundos reales, lo cual, limita la interactividad. Por su parte Flash, es animación en 2D donde el usuario interactúa con unos cuantos botones, un inicio y un fin determinado siempre por el programador de la animación, esta interacción es limitada. Así se pueden seguir mencionando aplicaciones que mientras no trabajen realmente en 3D, poco pueden hacer por proporcionar un ambiente análogo a la realidad, o una realidad alternativa.

El Internet se ha convertido en un importante medio para las aplicaciones académicas y comerciales. Hoy en día muchas empresas ofrecen sus productos con tecnología basada en el Web. Los mundos artificiales tridimensionales son considerados para ser uno de los más usados en forma amigable por los usuarios de la red que gustan de interacción con las computadoras. Un área de aplicación con una fuerte necesidad para usuarios de interfaces 3D es el comercio electrónico. Existe un gran número de productos que pueden ser presentados de manera clara y atractiva para los consumidores usando realidad virtual. Los usuarios pueden examinar de cerca el modelo de cada objeto e interactuar con él. Elementos adicionales como pósters, salas de chat y otros, pueden ser fácilmente integrados. Sin embargo hay un problema desde la construcción de estos mundos 3D.

Estos mundos se crean manualmente por expertos usando modelado 3D y herramientas de textura. Como estos viejos métodos no son suficientes, se debe contar con costos mínimos, rápidas actualizaciones, contenido personalizado y la garantía de correcciones totales, lo que se logrará con una nueva herramienta que logra la generación automática de mundos 3D: el VRML.

Los usuarios cada día necesitan y demandan más páginas, sitios, ambientes con los que se pueda hacer cualquier ocurrencia con sólo dar unos cuantos clics y manipular una escena de una manera amigable. Se necesita que exista para tal fin, un lenguaje o interfaz capaz de sumergirnos en un ambiente virtual que nos haga sentir inmersos en espacios con altura, ancho y profundidad, porque todo lo que existe sobre la Tierra vive sobre tres dimensiones. El presente trabajo tiene la intención de adentrar a cualquier persona en el uso de la Realidad Virtual como una alternativa para realizar implantaciones 3D en el Web y que esto sirva para dar a un proyecto una solución tridimensional, característica que con 2D no es posible y que hoy por hoy revoluciona a Internet.

Hablando de RV, se sabe que existen varias alternativas para su generación, desde una simple computadora con software especializado para tal fin, hasta implementos especializados de hardware. Se realizaron diferentes análisis para decidir como se trabajaría en el presente desarrollo. Cada alternativa cuenta con características propias y llevan a que su utilidad depende en gran medida del uso que se quiera dar a la RV.

Una alternativa para interactuar con realidad virtual es el uso de implementos de hardware (cascos, guantes, visores, etc.), ya que proporcionan gran inmersión e interactividad al usuario en una escena o mundo virtual, obviamente, este tipo de hardware compuesto por elementos electrónicos y un software integrado resulta muy costoso y la intención es proporcionar al usuario común una herramienta accesible en todos los aspectos. De esta manera, las alternativas nos llevan a una en la que sólo se requiere de una computadora personal y un software gratuito como medios de interacción. Esto representa una gran opción, porque aunque se quiera pensar que no todos los usuarios comunes en promedio tienen una computadora en casa, hoy en día, existen otras posibilidades como el acceder a la económica renta de una computadora en un café Internet a precio accesible, lo cual, es más viable que comprar un casco o visor virtual.

No se puede dejar de mencionar que Internet es el medio que hoy en día comunica a más computadoras a grandes distancias en tareas especializadas como consulta, comunicación o diseño para lo que utiliza transferencia de datos, vídeo, audio e imágenes, lo cual lo convierte en el medio adecuado para hacer llegar a cualquier persona las creaciones en RV. Una vez acordados los elementos, se establecen sus características. Primeramente se necesita una computadora personal con especificaciones mínimas de memoria, procesamiento, resolución de pantalla y periféricos, para su correcto funcionamiento, estas especificaciones se abordan de manera más completa en el apartado "capacidades del sistema".

El software debe manejar la RV lo mejor posible y ser accesible desde su instalación, pasando por la implantación y programación hasta su interactividad con un usuario final. Investigando sobre posibles programas y aplicaciones que pudieran generar un ambiente virtual como el que requerimos, se encontraron varios candidatos, entre los más viables, Java 3D, QuickTime VR [1], JavaView, VRML (Virtual Reality Modeling Languaje) y Autocad. Al analizar el funcionamiento de Java 3D, es sabido que genera gráficos de alta calidad, pero la programación en Java requiere un esfuerzo de desarrollo mayor, así como la demanda de muchos recursos de cómputo, lo cual ya implica demora en la construcción de un proyecto si no se cuenta con estos requerimientos.

QuickTime VR, es una mejora de QuickTime en la que se pretende (según sitio de Macintosh) la creación y visualización de ambientes de realidad virtual, consiste en una interfaz gráfica donde se añaden fotografías tomadas a 360° hasta conformar una animación rotativa a lo que ellos llaman realidad virtual. Estas aplicaciones no tienen nada de inmersivo, se consultó la página www.ibiza-spotlight.com, donde se presentan ejemplos de estos desarrollos y se puede ver que un objetivo importante es presentar geometrías dentro de un entorno tridimensional y este objetivo no lo cumple esta herramienta.

Autocad, es un sistema de interfaz gráfica con autolisp y herramientas que permiten la creación rápida y libre de estructuras, mapas, levantamientos, muros y cualquier elemento de diseño o construcción en 2D y 3D. Sin embargo, no es un medio para poder presentar dichas estructuras en Internet porque no fue creado para tal fin, tampoco es compatible con lenguajes de programación estructurada ni genera eventos en los ambientes.

Una alternativa más, la representa un nuevo sistema poco conocido pero de gran poder en representaciones tridimensionales, se trata de JavaView. Como su nombre lo indica fue construido en Java y permite modificaciones en su estructura al saber programar en este mismo lenguaje. JavaView es una interfaz gráfica del tipo de un applet (sub-ventana de un navegador de Internet) que permite la representación en tiempo real de figuras y geometrías regulares o irregulares a las que se les pueden modificar coordenadas, estructura, colores, escala, puntos de vista y demás características 3D.

Se supone que JavaView tiene la capacidad de importar archivos gráficos de varios tipos, pero en este trabajo se intentó que pudiera interpretar una estructura VRML y el resultado fue poco alentador debido a que JavaView no maneja nodos, eventos, animaciones ni texturas y no reconoció estas características, por lo que tampoco es una herramienta útil en las pretensiones de este trabajo.

Una vez que se examinaron las capacidades, facilidad de uso, beneficios y demás elementos relacionados de cada lenguaje, fue VRML el lenguaje que demostró ser la propuesta más viable por méritos propios para el presente trabajo al ofrecer mejores resultados que las aplicaciones ya mencionadas, como se ve en el siguiente cuadro comparativo:

	VRML	Java 3D	QuickTime VR	Autocad	Java View
Capacidad de presentar escenas 3D en Internet.	●	●	●		●
Facilidad de uso.	●	●	●	●	●
Facilidad de programación.	●				
Facilidad de desarrollo.	●		●	●	
Consumo de recursos mínimos.	●		●		
Interactivo en tiempo real.	●	●	●	●	●
Compatible con varios lenguajes de programación.	●	●			
Buena calidad de dibujo de escena.	●	●		●	●
Trabaja en 3D .	●	●		●	●
Genera animación.	●	●	●		●
Genera eventos.	●	●			

Cuadro IV.1.1. Comparación entre diferentes tipos de software que manejan 3D.

De manera más específica:

- VRML, fue creado para presentar escenas 3D, proyectos, animaciones y demás aplicaciones en Internet.
- VRML se puede programar tan fácil como programar una página Web en HTML (tarea que hasta en la preparatoria ya se resuelve).
- Los objetos y escenas de VRML son interactivas en tiempo real.
- Su código es compatible con cualquier lenguaje de programación desde Pascal hasta Java y lenguaje C.
- VRML maneja lenguaje script con lo cual provee animación a sus escenas y objetos.
- VRML trabaja siempre en tres dimensiones, algunas aplicaciones dicen hacer lo mismo pero en la realidad no lo hacen - por ejemplo, QuickTime VR - y en otros casos se tienen que realizar ciertas adaptaciones o conversiones como Autocad.

Se pueden seguir citando las bondades que ofrece VRML, pero queda claro con las ya expuestas, que se hizo la mejor elección al trabajar con este lenguaje que es accesible y poderoso al mismo tiempo.

Una vez que se encontró el lenguaje adecuado, se tiene que para poder observar los objetos creados se debe tener un visualizador del lenguaje VRML, es decir, una interfaz que interprete el código y lo convierta en las geometrías que construimos, en Internet se encuentran varios con sus propias formas de trabajar y formas de interacción. Como en el caso de la búsqueda del mejor sistema de RV, se analizaron los visualizadores más utilizados en el Web. De manera resumida, los visualizadores analizados fueron Cosmo Player, Cortona y Blaxxun. Se intentó analizar otros visualizadores como i3D del CERN, Live3D de Netscape, OZ Virtual de OZ Interactive, RealSpace de LivePicture, Vrealm de Ligos Technology, VR Web, WebOOGL de The Geometry Center, WebView de SGI y WorldView de Intervisa, pero no se pudo acceder a ninguno de sus sitios, al parecer, la gran mayoría ha desaparecido debido a la baja demanda de uso o quizá fueron demasiado deficientes. Se analizaron entonces a tres visualizadores más usados actualmente: Blaxxun, Cortona y Cosmo Player.

Blaxxun Contact [2], es un cliente de comunicación multimedia y 3D con el que se puede visualizar el estándar de VRML 97. Al analizarlo se encontró que sorprendentemente muestra el mundo virtual a toda la pantalla del navegador, es decir, que no utiliza un panel de control, sus herramientas de interacción se habilitan al dar clic derecho del Mouse, otra novedad es que trabaja totalmente en español con un lenguaje común con palabras del tipo "de prisa" mientras en otro visualizador dirá "very fast". Al realizar desplazamientos se notó una velocidad relativamente mejor que la de otro visualizador (Cosmo Player), el cursor en Blaxxun es una cruz que simplemente muestra la primera letra de la aplicación que se está usando. Otra novedad es que al abrir un archivo con eventos, el cursor sólo se convierte en una "manita" que parece indicar un hipervínculo, pero en la parte inferior izquierda del navegador se muestra textualmente que tipo de nodo sensor se esta activando cuando se pasa sobre un objeto que contiene un evento; por ejemplo, cuando se pasa el Mouse sobre un objeto que se puede desplazar libremente, se muestra que el comando se llama Touch. Aún con estas diferencias, no es de un uso ideal ya que es muy incómodo en su manipulación debido a que para cambiar de control se necesita dar clic derecho cada vez que se quiere elegir un nuevo control, aunque sólo sea para un pequeño movimiento y aunado a esto le faltan varios controles que Cosmo Player y Cortona si incluyen y que se muestran en el cuadro IV.1.2.

Cortona [3], tiene una presentación de medio margen de pantalla, formado por un panel de control del lado izquierdo y uno en la parte inferior, los cuales limitan el espacio de despliegue, estos paneles de control se pueden ocultar al igual que en CosmoPlayer, en Blaxxun no existen. Este visualizador tiene una velocidad de desplazamiento de los objetos más lenta que Cosmo Player y Blaxxun, esto se puede deber a que consume más recursos del procesador y memoria. Es importante mencionar que al probar la facilidad del uso de controles, se determinó que para la mayoría de desplazamientos de un objeto, se deben habilitar dos combinaciones o sucederá algo no deseado; por ejemplo, para hacer una rotación de 360° en las tres dimensiones, se debe habilitar el control "Walk" y después el control "Turn" ya que si se usa por ejemplo "Walk" pero con la combinación "plan", sucede un desplazamiento directo hacia el objeto, lo cual no se deseaba. Otra característica es que el cursor siempre tiene la forma común de una flecha hasta que no se realiza un movimiento, esto es muy incómodo porque no se sabe que acción se va a realizar a menos que se sepan todas las combinaciones de control de Cortona.

	Cosmo Player 2.1.1.	Cortona 4.0.	Blaxxun Contact 3D Ver. 5.1.
Acercamiento de cámara	Zoom	Plan	Caminar
Rotación de 360° en XYZ	Rotate	Walk - Turn	Examinar
Mover punto de vista	Pan	Pan	Volar
Inclinación de objeto	Tilt	Fly - Pan	Examinar
Mover hacia el objeto	Go	Walk - Plan, Pan	Caminar
Rotación de 360° en Y	Slide	Plan	Resbalar
Hacer y deshacer última acción	Undo / Redo Move	X	X
Reiniciar escena	Undo	Restore	Reiniciar
Sensor de interactividad	Starburst	Sphere Sensor	X
Acercamiento al objeto	Seek	Goto	Saltar
Lista de vistas predefinidas	ViewPoint	View	X
Gravedad	Gravity	Study	Gravedad
Flotando	Float	Float	Panorámica
Alineación 90° sobre X	Straighten	Align	X
Ayuda	Help	Help	Ayuda
Preferencias	Preferences	Preferences	Preferencias
Velocidad de carga en navegador	Aceptable	Media	Aceptable
Velocidad de desplazamiento	Aceptable	Lenta	Modificable
Facilidad del uso de controles	Buena	Buena	Media
Megabytes que consume en Disco Duro	3.18 Mb	1.56 Mb	5.47 Mb
Consumo de RAM	16 Mb	16 Mb	16 Mb
Compatibilidad con Navegadores	Explorer 4.x y Netscape 3.01 en adelante	Explorer 4.x y Netscape 3.01 en adelante	Explorer 4.x y Netscape 3.01 en adelante
Es Freeware	Y	Y	Y
X = No cuenta con el elemento o función.		**Y** = Cuenta con el elemento o función.	

Cuadro IV.1.2. Cuadro comparativo entre visualizadores de VRML.

A continuación se presenta una escena VRML, visualizada en un monitor con una resolución convencional de 600 X 800 pixeles. Se puede observar la escena manipulada por un panel de control de Cortona (color gris) y de Cosmo Player (color negro), la presencia de Blaxxun no se aprecia en la imagen debido a que no necesita panel de control para su manipulación, el cursor del mouse va cambiando su forma según la acción que este realizando. (Figura IV.1.1).

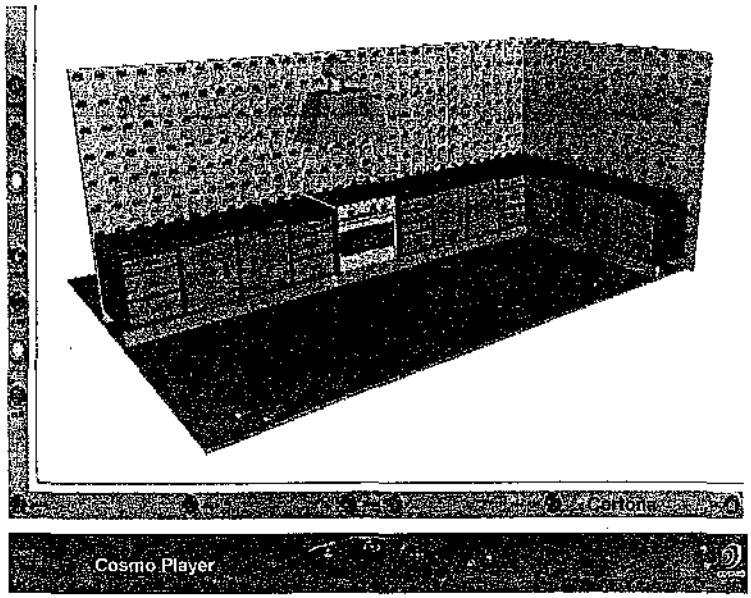


Figura IV.1.1. Diferentes vistas de los visualizadores Cosmo Player, Cortona y Blaxxun.

Obsérvese que con esta presentación, Cosmo Player permite aproximadamente un 5% en pixeles más de área de visualización de la escena, los botones están más cercanos para su fácil manipulación y las preferencias y ayuda en el mismo panel. Se concluye así que la mejor opción dentro de los visualizadores VRML analizados y evaluados, es Cosmo Player 2.1.1, por lo siguiente:

- Genera un ambiente de confianza, al saber que parte de la creación de VRML y Cosmo Player en su totalidad, fueron realizados por la misma empresa: sgi.
- Es un visualizador que tiene una presentación muy agradable por la forma en que están distribuidos sus controles, ayuda en tiempo real, panel de navegación, consola de errores, ocultación de barras de herramientas, botón de reinicio y las mencionadas.
- Se obtiene gratuitamente de Internet y su descarga es aproximadamente de 5 minutos en una conexión normal casera con un módem a 56 kbps.
- Fue creado específicamente para desplegar mundos VRML a diferencia de Blaxxun que es un cliente 3D en general.
- Se encuentra en actualización de sus versiones acorde al avance de VRML por Web3D Cossortium quienes lo evolucionarán a X3D.
- Una vez descargado el archivo cosmo_win95nt_eng.exe, su instalación es bastante fácil y rápida sin necesidad de reiniciar la computadora, detectando automáticamente los archivos de VRML que se tengan.

- Proporciona un manual de usuario (en inglés) para cuya consulta no se necesita estar conectado a Internet, además no se considera necesario ya que el uso del panel de control de Cosmo Player es muy accesible.

- Es el visualizador más recomendado por cualquier página que haga uso de VRML, basta con acceder a alguna página que contenga proyectos en VRML y se podrá verificar este hecho.

Teniendo ya un navegador convencional sea Explorer o Netscape y el visualizador de preferencia, sólo resta hacer uso de la creatividad personal (desde el punto de vista de diseñador), para comenzar a crear RV con VRML. Pasamos así, a la parte de la programación, partiendo de que existen otras alternativas para crear mundos 3D con VRML como los son: modeladores o constructores de VRML, editores de texto y código VRML y modeladores 3D con traductor de formato, aunque hay que dejar bien claro que no por ser más algunos método más fáciles quiere decir que sean los más adecuados, estas diferencias se muestran en el cuadro IV.1.4.

	Tipos de software.	Ventajas	Desventajas
Editor de Texto.	❖ Notepad. ❖ Word. ❖ WordPad. ❖ Xemacs. ❖ VrmlPad ❖ Otros...	❖ No es necesario comprar software adicional. ❖ Acceso a todas las características de VRML. ❖ Control detallado de la eficiencia del lenguaje.	❖ Difícil de construir elementos 3D. ❖ Requiere conocimiento del lenguaje: sintaxis y semántica.
Entornos de desarrollos VRML.	❖ Space Builder. ❖ 3D Viscape. ❖ Internet SupersCape. ❖ Otros...	❖ Fácil creación de los elementos 3D, animación e interfaz de usuario. ❖ No se necesita el conocimiento detallado de VRML.	❖ No se tiene acceso a todas las posibilidades del lenguaje VRML. ❖ El código que se genera no es el más eficiente. ❖ En versiones gratuitas limita el tamaño del código a manipular.
Modelador 3D y traductor de formato.	❖ 3D Max Studio. ❖ Autocad. ❖ JavaView. ❖ Otros...	❖ Buena capacidad de desarrollo de los elementos en 3D, animación e interacción. ❖ Se logran imágenes foto-realistas.	❖ No se tiene acceso a todas las posibilidades del lenguaje VRML. ❖ El código que se genera no es el más eficiente. ❖ El software no fue diseñado para soportar VRML. ❖ Normalmente solo existe una vía en la traducción al lenguaje.

Cuadro IV.1.3. Comparación entre diferentes métodos para crear mundos VRML.

Después de realizar evaluaciones entre estos métodos se determino que para el presente ningún método ofrece una buena asimilación de como trabaja el código VRML puro a menos que se programe en él y eso fue lo que se hizo. Haciendo una analogia, sería parecido a realizar una página Web con un modelador como FrontPage o Dreamweaver sin conocer el lenguaje HTML como tal. Se consideró más adecuado comprender el uso correcto y eficiente del lenguaje VRML y en un futuro, si es necesario y se desea, utilizar un modelador como herramienta directa o complemento.

Cada objeto, escena, mueble, luz, movimiento y demás elementos que componen la aplicación que complementa este trabajo, fueron creados con código VRML puro, escrito en un editor de texto avanzado (Xemacs) para su mejor manipulación. Xemacs es un sistema abierto, creado por la organización GNU, misma que participa en el desarrollo de Linux. Xemacs ofrece un gran potencial ya que permite la edición de texto normal y de código reconociendo idiomas y lenguajes de programación, esta última característica es muy importante ya que parecería que se está trabajando con el IDE de un compilador, Xemacs colorea las líneas de código según su tipo, además de diferenciar código de texto ASCII entre otras ventajas. Se recomienda este editor de texto ya que es sencillo y muy poderoso, se puede descargar en su sitio Web [4] rápidamente y en diferentes versiones, para Windows hay versión de interfaz gráfica así como para ambiente Linux. Es importante mencionar que en el menú de su se incluye un manual de usuario donde se explican todos los comandos para manipular este procesador de texto. A continuación su presentación preliminar y manipulando código VRML (figura IV.1.4).

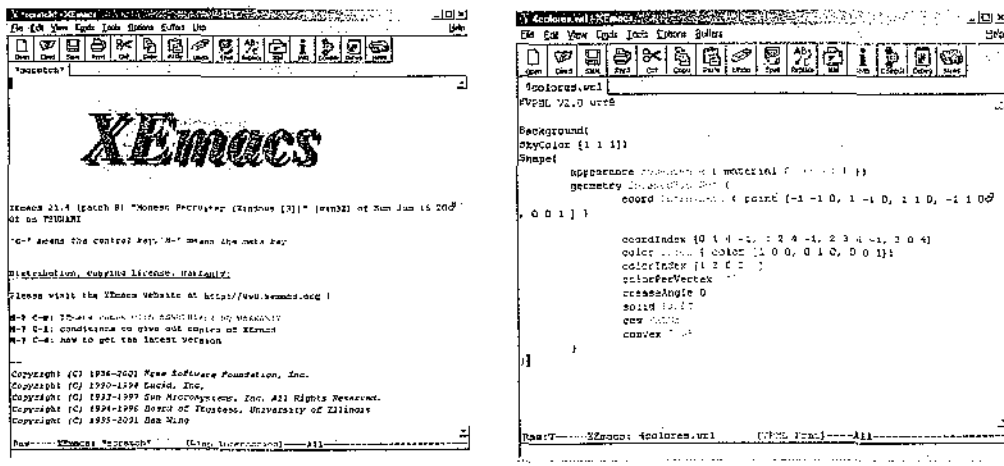


Figura IV.1.4. Presentación inicial del editor Xemacs (izq.) y la forma en que representa al código VRML en su interfaz.

Cómo se puede observar en el apartado "Gráficas de los objetos VRML", este lenguaje no puede crear objetos irregulares con las geometrías que tiene por definición (nodos primitivos), para lograr esto, cuenta con un nodo llamado "extrusión", el cuál, trabaja en base a la unión de puntos, puntos que conforman al objeto irregular, aquí la tarea es detectar las coordenadas de estos puntos, para esto, existen diferentes métodos como el usar un constructor de VRML, un software especializado en extrusiones como VRMLPad, o el finalmente elegido: la graficación directa de este tipo de geometrías con un software adaptado para tal fin, creado y desarrollado en el Departamento de Visualización perteneciente a la DGSCA en la UNAM.

Este software llamado *VisModel 3D* , es una aplicación en la que se tiene la libertad de introducir coordenadas en 3D y manipularlas en forma de dos planos cartesianos XY y XZ, con acercamientos y modificaciones en tiempo real, además de que al obtener las coordenadas finales de un objeto regular o irregular automáticamente se genera el código del nodo *Extrusion* de VRML, el cual, finalmente se encarga de darle forma virtual en el visualizador.

El presente trabajo incluyó las adaptaciones de *VisModel 3D*, para modelar objetos para VRML. Una vez que se encontraron las herramientas más idóneas se pensó en realizar una herramienta que sirva a cualquier usuario - aunque no sea un experto en diseño -, para construir espacios funcionales, en este caso tres: un apartamento, un museo y un centro de exposiciones. Los elementos intermediarios usados son algunos de los ya mencionados:

- Navegador de Internet: Explorer 5.0
- Visualizador de VRML: Cosmo Player 2.1
- Editor de texto: Xemacs
- Software de graficación tridimensional: VisModel 3D

El apartamento virtual partió de bosquejos, tarea primaria de cualquier diseñador. Posteriormente se les dedicó semanas de trabajo y de ensayo-error en cuestión de programación y finalmente se agregaron propiedades de interacción como desplazamientos, elección de objetos entre otras; para ello se contó con el apoyo de personas del área de diseño gráfico, programación en VRML y puntos de vista de usuarios finales.

El mundo virtual creado para el presente trabajo proporciona una solución a dificultades reales. Se resuelve la inquietud por previsualizar la decoración y amueblado del mismo para no caer en incoherencias con respecto a las dimensiones de los muebles y sus posiciones, esto se logra dando al usuario la posibilidad de modificar la escena como y cuantas veces quiera. De la misma manera que en un museo virtual se contaría con la ventaja de tener una panorámica previa del espacio que albergará los objetos de arte, cuadros, pedestales, señalizaciones, fichas descriptivas y todo lo que necesita un museógrafo para la mejor presentación de sus obras.

Otra alternativa de aplicación – como ya se mencionó en "Capacidades del sistema" se hallaría al montar una exhibición, en la cual con VRML se puede crear el espacio y la manera en que serán distribuidos los exhibidores (stands), posición de los mismos, ubicación de baños, cafeterías, centros de información, centros de registro de visitantes, conferencias, etc. Con dicha aplicación sólo quedaría modelar el centro de exhibición al gusto de los expositores e ir directamente a su implantación real, con esto se evitan bosquejos, croquis y maquetas innecesarias, además de poder usar la herramienta virtual para la comercialización del espacio de exhibición.

Hablando acerca del "boom" que han representado las ventas por Internet, VRML permite gran atracción por parte de los visitantes de un portal o sitio Web que contenga la representación virtual de cualquier objeto que se quiera comercializar, ya que no sólo verá una simple foto sino interacción real y vistas en 3D. Como se puede observar, en este proyecto se brindan distintas ideas logradas con un mismo lenguaje, un mismo visualizador y un mismo navegador, obteniendo compatibilidad con Internet, gran facilidad de programación, uso y poder de interacción en tiempo real, resolviendo necesidades comunes de hoy en día.

Recapitulando, la RV es un medio gracias con el que podemos explorar y examinar, en tiempo real, un mundo virtual (ambiente 3D generado por computadora) desde cualquier perspectiva e interactuar a la vez con los distintos elementos virtuales que lo componen. El usuario mantiene un control continuo de su movimiento y de su punto de vista.

Comúnmente, a la RV se le confunde con la animación y otras tecnologías que utilizan gráficas por computadora como la Visualización y la Graficación, pero recordemos que la RV es inmersiva, colocando al usuario en el interior del mundo creado por la computadora y que éste se pueda mover ahí, ver hacia arriba, abajo, derecha o izquierda como en el mundo real. También es interactiva en 3D, lo que permite al usuario manipular los objetos del universo virtual e interactuar con sus habitantes. A diferencia de otras tecnologías como la animación, la RV es en tiempo real, esto significa que lo que pasa se calcula exactamente en ese momento permitiendo libertad y realismo, que no puede dar la animación. Entonces la RV es en tiempo real, interactiva e inmersiva, lo que representa el utilizar la mayor cantidad de sentidos para crear la sensación de inmersión. Se debe hacer énfasis en los elementos característicos de un sistema de RV, mismos que se encuentran implícitos en la presente aplicación:

Interacción: Quiere decir que el mundo virtual y sus objetos o personajes reaccionan a las acciones del usuario y de ellos mismos. El usuario a su vez reacciona a ellos y esto le permite manipular el curso de la acción dentro de una aplicación de RV, logrando que un sistema virtual responda a los estímulos de la persona que lo utiliza; creando interdependencia entre ellos. Existen dos aspectos únicos de interacción en un mundo virtual: el primero de ellos es la *navegación*, que es la habilidad del usuario para moverse independientemente alrededor del mundo. Las restricciones para este aspecto las coloca el desarrollador del software, que permite un rango de grados de libertad, si se puede volar o no, caminar, nadar, etcétera. El otro punto importante de la navegación es el punto de vista del usuario y uno más es la interacción dinámica del ambiente, que no es más que las reglas de como los componentes del mundo virtual interactúan con el usuario para intercambiar información.

Inmersión: Es el proceso por medio del cual se genera la ilusión óptica y sensorial para que el usuario crea que esta formando parte de la experiencia virtual, es lo que crea la sensación de lo que podríamos llamar "estar ahí". Esta palabra significa bloquear toda distracción (en lo posible) y enfocarse sólo en la información u operación sobre la que se trabaja. Posee dos atributos importantes, el primero de ellos es su habilidad para enfocar la atención del usuario y el segundo es que convierte una base de datos en experiencias, estimulando de esta manera el sistema natural de aprendizaje humano (las experiencias personales).

Tridimensionalidad: Esta es una característica básica para cualquier sistema llamado de RV, tiene que ver directamente con la manipulación de los sentidos del usuario, principalmente la visión, para dar forma al espacio virtual; los componentes del mundo virtual se muestran al usuario en las tres dimensiones del mundo real: ancho, largo y alto, en el sentido del espacio que ocupan.

Con todo lo anterior se sabe lo que es capaz de hacer uno de los varios lenguajes de RV existentes como lo es el VRML. Después de trabajar con él, se encontraron varias características que llevan a concluir que es la aplicación más viable a utilizar para el proyecto. Con esto se logró el objetivo del presente trabajo: crear espacios en los que un usuario cualquiera que tenga una computadora personal e Internet a su alcance, pueda diseñar un ambiente apto a sus necesidades, sean de vivienda, exhibición o representación, partiendo de una superficie base y pudiendo elegir todos y cada uno de los elementos que lo conformaran, así como su lugar y posición hasta que se sienta de lo más a gusto e identificado posible con su espacio funcional. En el desarrollo del presente tema se han utilizado fuentes confiables que justifican el uso del lenguaje VRML y sus aplicaciones. Sabemos que el lenguaje se encuentra en desarrollo y su difusión es poco conocida por las personas ajenas al cómputo. Aún así, lo que hasta el momento existe es bastante funcional y se ha llegado a construir desde una simple oficina virtual hasta simulaciones de todo tipo. Además se sabe que VRML una vez más demostrará su poder, al presentarse una nueva versión que toma como base al VRML 2.0, pero con mejoras compatibles con Internet llamado X3D del cual se hablará más adelante. Las principales fuentes bibliográficas en que me he basado hasta el momento serán descritas en el apartado "Referencias electrónicas comentadas".

IV.2. Estructura del sistema.

En los últimos años un nuevo escenario para la representación de mundos virtuales en 3D se está reafirmando: la realidad virtual basada en Internet. Este avance ha permitido el acceso público a distintos logros y las últimas aplicaciones de la Realidad Virtual en áreas como la enseñanza, la simulación, el comercio, la industria, las ciencias entre otras. La Realidad Virtual por computadora es más accesible a los usuarios, ya que, elementos como guantes de datos, cascos estereoscópicos y otro tipo de sensores de altos costos se sustituyen por la pantalla, ratón y conexión a Internet usuales en cualquier equipo informático.

En el presente trabajo se mencionan aplicaciones posibles en el ámbito de la construcción o diseño de espacios y simulación. Se desarrolla un espacio virtual interactivo para la creación de espacios funcionales basado en el Web. Se propone la inclusión de este tipo de tecnologías en cualquier desarrollo inmobiliario, presentaciones previas de exposiciones en museos y en exposiciones empresariales.

El presente trabajo ha sido elaborado siguiendo una proceso estructurado en la creación de los espacios funcionales. Un espacio funcional es aquel espacio en donde convergen elementos u objetos que conforman un todo que servirá para cubrir una necesidad o aplicación a diferentes usuarios. En este caso, el objetivo de la creación de espacios funcionales se logro con la utilización de componentes de hardware y software así como una metodología de elaboración de cada uno de ellos.

Cabe mencionar que desde el inicio de la planeación de proyectos funcionales como el del presente trabajo, se analizan las necesidades y demandas tanto del programador o desarrollador y del usuario. Es lógico pensar que los usuarios de esta aplicación no quedarán satisfechos si observan que las únicas interacciones son girar el mundo virtual y observar desde diferentes perspectivas los objetos que lo componen. Un mundo 3D que solo se limitará a ello sería simplemente otra técnica de despliegue sin ninguna otra característica, por ello, para justificar el esfuerzo, las capacidades de 3D gráficos modernos y tecnología de interacción tienen que ser usadas para proporcionar varias ventajas que otras tecnologías no pueden ofrecer, por ejemplo:

- Darle al usuario la sensación de que realmente esta allí en el mundo virtual funcional. Esto incluye la presentación de los objetos y muebles completos que se encontrarían en una casa real.

- Mantener un incomparable nivel de libertad de navegación dentro de los espacios funcionales y de interacción con todos los elementos que lo componen. Todo ello sin incurrir en gastos innecesarios como el uso de dispositivos de rendimiento estereofónicos, guantes de datos, cascos sensoriales u otros

- Generar un código abierto lo más pequeño y entendible posible, siempre y cuando siga cumpliendo los requerimientos del sistema (interacción y tridimensionalidad).

- Dar a los usuarios la habilidad de jugar (desde el punto de vista amigable) con el espacio funcional en todos los aspectos. De hecho, el panel de navegación que presentan la mayoría de los visualizadores VRML esta diseñado para ello.

- Cualquier vendedor u ofertante de algún producto podrá darle mayor énfasis de venta a sus productos por medio de esta tecnología dentro de varios criterios :

(a) Debe crear un mundo virtual lo más cercano posible a una apariencia real de lo que representa su producto, por lo cual, cada objeto que lo compone debe tener las características mínimas claramente visible a los clientes o usuarios.

(b) Debe existir un manual de mantenimiento en el que los desarrolladores del proyecto actualicen continuamente al espacio funcional y todos los elementos que lo componen.

(c) Los elementos del espacio deben ser cambiables con un esfuerzo muy pequeño (refiriéndonos a código fuente) para poder mejorarlo y que siempre este actualizado y sea de un uso apropiado, esto es esencial responder a los desafíos de otros vendedores.

No se debe olvidar que una de las primordiales funciones de los modelos 3D deben ser el utilizarlos para planear las tareas en el mundo real (por ejemplo, para la colocación de estantes y productos, así como para la selección de colores y otros elementos de un espacio funcional).

Basado en estos resultados, los requisitos técnicos para generador el espacio funcional previamente definido se tiene que:

- ✓ Se debe mantener una variedad de niveles de calidad y rendimiento (por ejemplo, los niveles de detalle para los objetos 3D o las resoluciones y texturas) para compensar las demandas de los diferentes usuarios y las diferentes capacidades del sistema.
- ✓ Crear un modelo realista, de calidad superior con ayuda de las texturas, fuentes y un gran número de detalles.
- ✓ El individuo usuario de la aplicación podrá integrar elementos 3D para propósitos especiales o particulares libremente.

Primeramente se deben cubrir las necesidades del programador de los espacios, con la finalidad de que cuente con las herramientas necesarias. Estas herramientas comprenden una computadora con características mínimas para trabajar como se muestran en el cuadro IV.2.1. ya mencionados en el apartado "Capacidades del sistema".

Procesador	➢ Pentium 75 Mhz o más rápido.
Memoria RAM	➢ 16 Megabytes como mínimo.
Espacio en Disco Duro	➢ 14 Megabytes de espacio libre.
Monitor	➢ Super VGA a 256 colores de 15 pulgadas o superior.

Cuadro IV.2.1. Características mínimas de hardware para crear y ejecutar mundos VRML.

Los elementos de software se muestran a continuación.

Sistema Operativo	➢ Microsoft Windows 95, 98, Server 2000, Me, Linux, Unix o equivalentes.
Navegador de Internet	➢ Versión mayor a Explorer 3.0, Netscape 4.0 o equivalente.
Editor de texto	➢ Xemacs, Word. Word Pad, Bloc de notas entre otros.
Visualizador de VRML	➢ Cosmo Player 2.1.1, Cortona 4.0 o equivalente.

Cuadro IV.2.2. Características de software necesarios para crear y ejecutar mundos VRML.

El programador puede optar por las herramientas que considere más adecuadas. A manera de información el apartamento virtual se ha creado usando un sistema operativo Windows Me, Navegador Explorer 5.0, Editor Xemacs para windows, Software para graficación VisModel 3D y visualizador Cosmo Player 2.1.1. Estos elementos de software son los más usados por proporcionar los mejores resultados al trabajar en desarrollos virtuales en PC; pero no se puede omitir el hecho de que existen otros alternativos. La mayoría de este software esta al alcance de cualquier usuario con solo dedicar unos minutos de navegación en Internet por las páginas de los proveedores y cuyos sitios Web se proporcionan en el presente trabajo en la sección de Literatura citada.

Otra de las actividades del programador es establecer referencias de los espacios funcionales que se quieren elaborar, es decir, experiencias personales de presencia en lugares similares, fotografías, películas revistas u otras fuentes. Todo ello para tener una idea lo más cercana a la realidad. Partiendo de estos elementos se establecen una serie de pasos que se seguirán hasta alcanzar el objetivo final: la creación de un espacio funcional. A continuación la metodología utilizada para el desarrollo de la aplicación mencionada.

METODOLOGÍA

- ✓ Se establecen las fronteras espaciales, esto servirá para delimitar el área global del espacio y de la misma manera las áreas internas que integran un todo. El elemento, será un plano con medidas a escala que representaría lo más fielmente posible la estructura del espacio funcional. Este paso es el primero y más elemental ya que establece el área en la que residirán habitaciones o salas y a su vez los objetos que darán vida.

- ✓ Una vez establecidos los límites se establecerán los elementos que integrarán al espacio funcional en cada caso. Para el apartamento se construirán muebles y objetos para cada habitación ya delimitada. Dichos muebles y objetos se van creando de acuerdo a la habitación elegida sin un orden particular. Pro ejemplo, en la recamara habrá: tapetes, cuadros, cama, buró y lámparas. En la sala: sillones, lámparas, mesas, cuadros y tapetes. En el comedor: mesa, sillas y bancos. En la cocina: fregadero, estufa, campana y alacenas. En el baño: taza, portapapel, lavabo y espejo. De la misma manera se establecen límites para las demás aplicaciones.

- ✓ Una vez delimitadas las áreas y establecidos los objetos y sus comportamientos se procede a establecer recorridos de navegación predefinidos en el espacio funcional. Estos recorridos evitarán navegación incorrecta o de otra manera proporcionarán una navegación más cómoda y directa por el mundo virtual ya que con solo elegir un recorrido, automáticamente la pantalla se desplazará hacía alguno de los lugares más solicitados por los usuarios finales del espacio funcional. Cabe mencionar que esta propiedad esta sujeta al visualizador VRML que se este utilizando para que interprete el código que realiza el recorrido.

- ✓ Lo anterior se logra porque el visualizador de VRML proporciona la facilidad de establecer rutas como parte de un menú en la consola del mismo. Estas rutas aparecen en un menú que se genera mostrando el nombre del recorrido y al elegirlo el usuario se desplazará virtualmente.

- ✓ Cabe mencionar que debido a que esta aplicación es una muestra de lo que se puede hacer con la Realidad Virtual y en particular VRML, no todos los objetos tienen las mismas propiedades de interacción. Así, los únicos muebles que además de la movilidad que permite el visualizador con su panel de control tienen otros comportamientos como: rotación y desplazamiento son: muebles de la recamara y sala. En el apartado "capacidades del sistema" se detallan dichos comportamientos.

- ✓ Para asignar los comportamientos mencionados a los diferentes objetos se tiene que realizar una estructura dividida en tres partes para cada objeto: el código que da forma al objeto en sí, el código que da forma al elemento mediador con el que el usuario interactuará para manipular al objeto, en este caso una pequeña esfera verde y por último una sección de código script[1] quien contiene las instrucciones de desplazamiento y rotación según los movimientos del mouse por parte del usuario.

Desde el punto de vista del usuario, este tendrá como medio de interacción al visualizador de VRML e implícitamente al panel de navegación que proporciona. Desde el momento en que interactúa con la página de presentación y da clic en el enlace que carga alguno de los espacios funcionales se realizará una petición al visualizador para que represente las geometrías que dan forma al espacio.

Ya en el espacio el usuario interactuará con él y cada uno de los objetos de una manera con el panel de control y el mouse y de otra con las herramientas que se generan con scripts de VRML.

- *Libertades del usuario.*

  - Libre desplazamiento de los muebles en un área de acción delimitada, por ejemplo, los muebles de la recamara se pueden desplazar por ella con solo hacer clic sobre el objeto y arrastrar el mouse por el interior de la recamara.

  - Los muebles de la sala se pueden mover por el área en que esta delimitada con las paredes y así cada uno de los objetos que tienen permitida esta acción. Los objetos que tienen permitida esta acción tienen en su superficie una esfera pequeña de color verde. El objeto podrá ser desplazado cuando al pasar el mouse sobre él, cambie el cursor a la forma de un pequeño sol (usando Cosmo Player) , lo cual, indica que el script puede realizar la acción al detectar la posición del mouse.

  - Otra de las libertades, es la rotación de los muebles. Esta rotación es propiedad de los muebles que también tienen en su superficie una pequeña esfera verde que contiene implícitamente el script de rotación. Así, al observar un mueble que contiene dicha esfera se podrá dirigir el cursor del mouse hasta ella, y una vez hecho esto cambiará el cursor a la forma de un pequeño sol y entonces el usuario puede dar un clic y el mueble rotará 90° en el sentido de las manecillas del reloj. Es importante mencionar que se rota el mueble 90° con cada clic pudiendo con esto regresar a su posición inicial. El hecho de que gire sobre su eje 90° es debido, por convención, a la forma cuadrada común de las habitaciones. Con esta posibilidad el usuario puede adecuar la posición de los muebles después de haberlos desplazado hasta una posición deseada o viceversa.

[1] Pequeño programa creado con un lenguaje de alto nivel para dar instrucciones a otro programa y realizar así tareas específicas.

- Si el usuario requiere saber datos sobre el mueble que va a elegir necesitará dar clic sobre el nombre del mueble, que lo llevará a la página Web donde se desplegará la imagen del objeto con sus datos técnicos. En este enlace encontrará datos como: medidas, color, estilo, precio y otros.

- Recordemos que la presente aplicación es una herramienta en proyecto con comportamientos elementales libres de modificaciones y adecuación a algo más ambicioso, pero, que como tal es una herramienta para la creación de espacios funcionales, donde un usuario puede utilizar los beneficios de la realidad virtual en Internet por medio de una aplicación en la que él es quien finalmente creará su espacio de acuerdo a sus gustos y necesidades con solo dar unos clics.

- La expectativa de esta aplicación es el brindar una herramienta a diseñadores de casas, de interiores, arquitectos, decoradores e incluso a empresas del ramo inmobiliario que deseen colocar en su página Web esta herramienta para atraer más clientes.

Como se puede observar, el papel del programador es muy diferente al del usuario final de la aplicación. Por una parte el programador necesita saber conocimientos básicos de programación, perspectiva, diseño, tridimensionalidad, tener referencias de lo que construye y que herramientas le puede proporcionar al usuario final dependiendo de las ambiciones que persiga dicho sistema y a que tipo de usuarios va dirigido. De esta manera, tiene que echar mano de inventiva y lógica para resolver el problema de cómo darle al usuario una herramienta accesible que le ayude a crear un espacio funcional, que le sirva para tener un prediseño de espacios en forma virtual a manera de ahorrar costos y esfuerzos.

En la aplicación, se pensó que una de las herramientas que se podían usar para crear los espacios, era la realidad virtual por características propias que otros sistemas no proporcionarían, por ejemplo:

- Tener una perspectiva en 3D, es decir tres dimensiones para hacerlos más real y a la vez más atractivo.
- Al usar 3D cualquier persona puede darse una idea clara de lo que ve si tener que imaginar dimensiones como cuando se ve cualquier imagen en 2D.
- La realidad virtual permite por cualquiera de varios medios (cascos, visores, monitores, etc.), percibir una realidad simulada.
- Proporciona el poder de observar objetos, escenarios y situaciones reales pero simuladas, con lo que se pueden prever futuros malos resultados en sistemas.
- La realidad virtual ahora ya esta al alcance de más personas gracias al Internet sin tener que recurrir a gastos excesivos en hardware.
- Con el avance de los lenguajes de programación ahora es fácil y accesible el programar mundos 3D, situación imposible hace unas décadas.

El otro elemento importante en el desarrollo de este trabajo fue elegir un lenguaje de programación accesible en su entendimiento y manipulación al momento de crear realidad virtual en corto tiempo. Este lenguaje es VRML, y entre sus principales características esta el programarse de manera similar a HTML, crear sus objetos en tiempo real y su capacidad de crear elementos que se pueden cargar al instante en Internet.

Aunado a esto, los bajos requerimientos de hardware al ser VRML un lenguaje de realidad virtual que se construye en una computadora personal, se logran conjuntar todas las herramientas necesarias para dar al usuario elementos con los que solo se preocupe por saber como y en que posición quiere los elementos que van dando forma a sus espacios funcionales. VRML 2.0 es una versión de código 3D que se encuentra en constante actualización y libre de modificación en el código que genera, por esto, se piensa que seguirá siendo una plataforma de edición 3D estándar para Internet en un futuro para el diseño de cualquier mundo 3D imaginable por su facilidad de programación y compatibilidad con el Web, sufriendo por supuesto, de mejoras en su estructura y funcionalidad, ejemplo de esto es el naciente X3d.

Del otro lado de la presente aplicación donde se encuentra la persona que verá los presentes sistemas como una herramienta intermediaria, hará uso de su imaginación o de sus necesidades personales de diseño o marketing solo limitándose a dar unos cuantos clics para ir creando escenarios interactivos con propiedades como: escoger, agregar, quitar, reiniciar, desplazar, rotar, delimitar, etc. Este lado de la aplicación es sencillo si se trata de su manipulación, aunque el proceso de diseño tiene su importancia ya que de nada sirve un sistema con las propiedades ya mencionadas, sino se tienen buenas ideas.

Es importante mencionar que el usuario de la presente aplicación debe tener una clara idea de las ventajas que recibirá a cambio del buen uso de los mismos, refiriéndome a que debe tener conocimientos básicos de los que es el VRML como programa que genera escenas 3D, para ello se han colocado referencias de sitios Web donde puede descargar manuales y tutoriales. Otra de las actividades que se deben realizar sino se ha interactuado antes con este tipo de aplicaciónes, es leer el manual de usuario que se presenta en este trabajo a fin de se familiarice con la forma en que se opera el panel de control del visualizador en el que se cargará el espacio funcional, en este caso, Cosmo Player por ser - a mi juicio - el más estándar. Finalmente revisar el contenido del apartado llamado "Capacidades de los Sistemas" para que se observen los requerimientos mínimos de hardware y software además de las características y posibilidades que brinda cada aplicación.

IV.3. Aportaciones.

Después de analizar la presente aplicación en sus capacidades y alcances, se reconoce que el presente trabajo hace las siguientes aportaciones a la ingeniería en computación:

- Se crearon "scripts" para dar movimientos como: libre desplazamiento y rotación a los objetos.
- Se adapto el funcionamiento del software VisModel 3D proporcionado por el Departamento de Visualización en D.G.S.C.A. para realizar las gráficas de los objetos 3D.
- Se utilizó el editor de texto Xemacs para realizar una revisión sintáctica y semántica de los códigos VRML.
- Se realizaron comparaciones entre los visualizadores con mayor demanda en Internet, proporcionándose un cuadro (IV.1.2) de resultados.
- Análisis de la forma en que trabaja VRML en un editor de texto y utilizando un constructor o modelador.
- Evaluación de alternativas de software modelador con traductor de formato VRML para la creación de espacios funcionales proporcionándose un cuadro (IV.1.1) de resultados.
- Se observo que las páginas existentes del ramo inmobiliario no manejan VRML para hacer atractivos sus sitios.
- Se concluye que los recursos con que se cuenta en el medio inmobiliario y de diseño de espacios no tienen interactividad 3D y al mismo tiempo libertad de manipulación de objetos.

Se espera que las anteriores aportaciones sirvan para que los usuarios del espacio funcional logren comprender que hacen falta elementos interactivos con lenguajes y aplicaciones accesibles y que con herramientas de este tipo se generen más desarrollos que ayuden a resolver problemáticas en general y no solo de generación de espacios.

Conclusiones.

Se ha incursionado en el área de diseño y construcción de espacios con el propósito de desarrollar una herramienta que permita a los usuarios e internautas de páginas inmobiliarias contar con una aplicación que les permitan crear espacios funcionales, lo cual, representa el tener un software aún inexistente en el mercado.

El presente trabajo es el resultado de la detección de una problemática real detectada a través del navegar por páginas que pretenden mostrar muebles y elementos de diseño en una forma poco atractiva y estática - como se menciono al principio -, además de detectar que las personas relacionadas con el diseño, no cuentan con herramientas de este tipo.

En el presente trabajo ilustro una de las varias aplicaciones en el contexto del diseño de espacios funcionales: un apartamento virtual. Se desarrollo con la finalidad de proponer la inclusión de estas tecnologías en inmobiliarias como ejemplo de lo que se puede realizar en muchas otras áreas y los grandes beneficios que conlleva usar la Realidad Virtual como medio para hacer el Web más amigable a los usuarios. La aplicación tiene como objetivo previamente analizado el ser una herramienta que pueda servir a un arquitecto, diseñador, decorador, vendedor de inmuebles o cualquier a fin a la rama de exposición y diseño.

Cabe mencionar que VRML fue una interesante línea de desarrollo ya que es una de las alternativas más viables desde mi punto de vista al desarrollar un entorno virtual, no se descarta el poder utilizar otras aplicaciones, pero reitero que para mi fue la más interesante.

Con la implantación de esta aplicación se alcanzarán las siguientes ventajas:

- Reducción de tiempos en la interacción, gracias al VRML como lenguaje de realidad virtual.
- Confiabilidad de que el usuario es dueño de la manipulación del espacio.
- Inmersión total en el entorno 3D.
- Libertad de desplazamientos.
- Capacidad de modificar el ambiente 3D.
- Utilización de recorridos automáticos por el mundo 3D.
- Libres modificaciones de los objetos en su tipo, textura, color y posición.
- Paginas inmobiliarias totalmente interactivas con elementos más atractivos para los visitantes, si de vender por Internet se trata.

Después de implantar esta herramienta en computadoras de personas involucradas con el diseño se encontraron los siguientes resultados:

- Menores problemas de manipulación que en ambientes equivalentes.
- Mayor confort de navegación gracias a la forma de operación del visualizador Cosmo Player.
- Un aprendizaje rápido en la forma de manipular el visualizador.

- Capacidad por parte de los usuarios de modificar las aplicaciones en su código, con solo dar lectura al manual de VRML adjunto en el presente trabajo.

- Se encontraron opiniones de que este tipo de aplicaciones deberían implantarse y tomarse como una herramienta alternativa de diseño de interiores.
- Se considero al lenguaje VRML muy accesible parecido a programar en HTML.
- Se concluyo que este tipo de aplicaciones si representan una alternativa real para vender por Internet y es lo que la gente está esperando.

Se puede concretar entonces, que se ha desarrollado una aplicación que cubre las necesidades de creación de espacios funcionales que se requieren en el campo del diseño, así como la explotación de los recursos de interactividad, inmersión y tridimensionalidad.

La conclusión entonces es que la creación de espacios funcionales con realidad virtual es una alternativa real y viable por las ventajas que ofrece y los resultados arrojados después de implementar y evaluar la aplicación, encontrando en ella: eficiencia, flexibilidad y consistencia en su uso. Se espera el haber aportado alternativas importantes al ramo de la ingeniería en computación con el presente trabajo que sirvan para su uso o como idea para la creación de desarrollos alternativos.

ANEXO I.
Uso de los visualizadores VRML.

Primeramente para navegar en Internet y a su vez en mundos 3D del VRML, tenemos que hacer uso de un visualizador . Los visualizadores tienen en este caso, la función de interpretar los mundos virtuales y lograr que los podamos ver en nuestra computadora como tales, porque en realidad los mundos son programas.

Existen en este momento varios simuladores conocidos y desconocidos. Cada visualizador funciona de diferente forma y posee controles para manipular los mundos VRML diferentes, desde sencillos hasta elaborados. Actualmente existen en el medio visualizadores como: *Web Space, WorldView, Qmosaic, Netscape VRML, Live 3D, Blaxxun, Cosmo Player entre otros.* En este apartado se trata a Cosmo Player Versión 2.1.1 al ser el visualizador que se esta recomendando y utilizando por la comunidad Web al manipular mundos en VRML (ver cuadro (A.1.2) comparativo en "Antecedentes y evaluaciones"). Cosmo Player permite un panel controlador para manipular los mundos u objetos creados con mucha facilidad y desde diferentes perspectivas.

Instalación

Debido a que Cosmo Player es un software de libre difusión, creado por la empresa Silicon Graphics, para obtenerlo sólo se necesita accesar a su sitio Web en www.cosmosoftware.com y ejecutar los siguientes pasos:

- Dar clic en el botón con la siguiente leyenda: "Free Download for Win 95/NT"
- En la siguiente ventana se deben llenar las cajas de texto: "First Name, Last Name y Email"
- Después de llenar los datos requeridos se debe dar clic en el botón "Download Now" y enseguida en el idioma en que se desea la descarga (preferible "english")
- Elegir "Guardar este programa en disco" y la ruta deseada (preferentemente en C:)
- Una vez terminada la descarga, proceder a su Instalación.

Una vez que el archivo aparece en la carpeta que se especificó (este archivo es un .exe), se le da doble clic y comenzará el proceso de instalación. Cuando se ha terminado de instalar, se abre el navegador y se revisa que se instaló correctamente dando clic en el menú de "Ayuda" y luego en "Acerca de los conectores o plugins". Si no aparece, obviamente no se instaló correctamente y habrá que repetir la instalación.

El panel de control

A continuación en la figuras A.4.1, A.4.2 y A.4.3 se presenta el panel de control del visualizador Cosmo Player con todos sus controles para manipular los mundos virtuales.

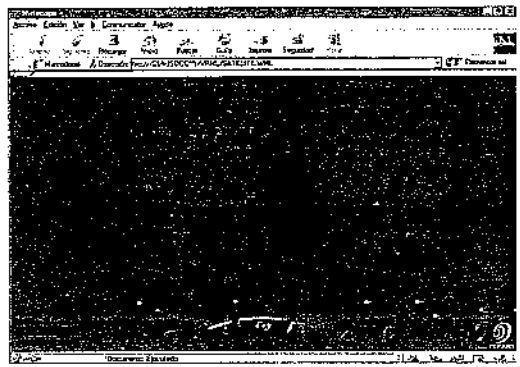
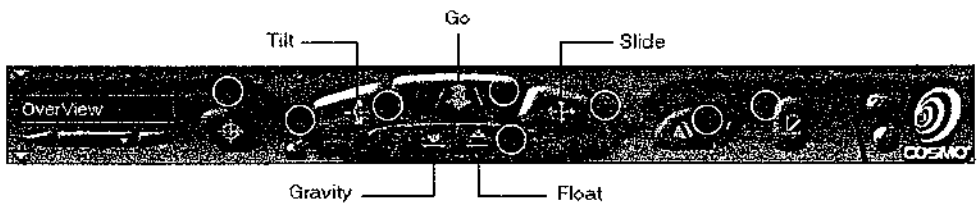


Figura A.4.1. Panel de control del Visualizador Cosmo Player utilizando el navegador Netscape.

En la figura A.4.1 se puede observar la presentación del visualizador Cosmo Player, el cual, tiene en su parte inferior un panel de control que opera en dos modos diferentes. Estos dos modos de operación se pueden observar en las figuras A.4.2 y A.4.3.

En la figura A.4.2 se muestra el panel que aparece por omisión al abrir la interfaz de Cosmo Player, es llamado "Moviéndose alrededor del mundo virtual", sus controles se explican más detalladamente en el cuadro A.4.1:



En la figura A.4.3 se observa el segundo modo del panel de control de Cosmo Player, el cual, aparece al dar un clic sobre el control número 2 del panel mostrado en la figura A.4.2.

Figura A.4.3. Vista del segundo panel de control de Cosmo Player.

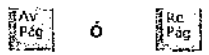
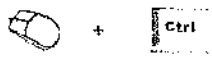
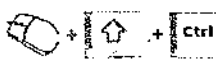
Acciones de los controles.

A continuación un cuadro donde se muestra una breve descripción de las acciones que realiza cada control con la finalidad de manipular los objetos y mundos VRML utilizando Cosmo Player 2.1.

1. SEEK	Este control se encarga de acercar el objeto hacia el usuario, se hace clic sobre éste control y a continuación sobre el objeto que se quiere acercar.
2. CHANGE CONTROLS	Switch o palanca que intercambia el modo de los paneles de control al hacer clic sobre él.
3. TILT	Haciendo clic en este control, el objeto es arrastrado para observar sus vistas lado a lado.
4. GO	Mueve el objeto en cualquier dirección.
5. SLIDE	Arrastra la vista del objeto hacia arriba, abajo, izquierda o derecha.
6. FLOAT/ GRAVITY	Al navegar en un mundo virtual, con el control GRAVITY el mundo virtual se mantiene sobre un piso o Tierra. Con el control FLOAT el mundo virtual "flota" de cierta manera, ya que no existe Tierra o piso para nuestro objeto virtual.
7. REDO MOVE/ ENDO MOVE	Estos controles sirven para "regresar o adelantar" movimientos, es decir, después de haber realizado diferentes movimientos con el objeto virtual a través de los controles, podemos regresar a algún movimiento de los que realizamos utilizando estos controles.
8. STRAIGHTEN	Su función es mover el objeto de manera que quede en forma perpendicular con respecto al eje horizontal, no importando a que ángulo se encuentre inclinado el objeto.
9. ZOOM	Su función es acercar o alejar el objeto, esto se logra haciendo clic sobre el control y después sobre el objeto y sin soltar el mouse desplazarlo hacia arriba y abajo, de esta manera el objeto se alejará o acercará hacia nosotros.
10. ROTATE	Haciendo clic sobre este control y después sobre el objeto y sin soltar el mouse, hacia cualquier lado que hagamos movimientos el objeto será rotado. Esta función es de gran utilidad ya que permitirá tener diferentes vistas del objeto en sus 3 dimensiones.
11. PAN	Después de seleccionarlo (hacer clic sobre este control), se coloca sobre el objeto y sin soltar el mouse, el objeto será arrastrado paralelamente con el cursor del control en la interfaz, hacia cualquier posición deseada.

Cuadro A.4.1. Controles del panel de Cosmo Player 2.1.

Cabe mencionar que Cosmo Player no se limita al uso del Mouse, ya que provee la posibilidad de permitir interactuar con los mundos 3D por medio del teclado también, una manera rápida y a mena de explicar este tipo de interacción se muestra en el cuadro A.4.2.

	Pulsa sobre estas teclas para moverte de forma **guiada** por el escenario
	Mantén pulsado el botón Izquierdo del ratón sobre el escenario y al mismo tiempo mantén pulsada la tecla Shift. Mueve al ratón hacia derecha e izquierda (desplaza) y arriba/abajo (zoom).
	Mantén pulsado el botón Izquierdo del ratón sobre el escenario y al mismo tiempo mantén pulsada la tecla Ctrl. Mueve al ratón hacia derecha/izquierda/arriba/abajo para girar tu punto de vista.
	Mantén pulsado el botón Izquierdo del ratón sobre el escenario y al mismo tiempo mantén pulsadas estas dos teclas para moverte.

Cuadro A.4.2. Formas de operación de Cosmo Player con el teclado.

ANEXO II.
Manual de VRML .

El código VRML es muy accesible, ya que se puede visualizar su contenido con un simple editor de texto, no así sus objetos, para ello se requiere de una de dos tipos de aplicaciones diferentes: un visualizador o un *plug-in* instalado en un navegador de Internet. La opción a usar en la mayoría de las veces es la del *plug-in*, de este modo, el Netscape Communicator apartir de su versión 4.01 incorpora como *plug-in* de VRML el Cosmo Player de Silicon Graphics, mientras que para Internet Explorer habrá que descargarlo. Para averiguar que *plug-ins* se encuentran instalados en un navegador basta con desplegar el menú *Help / About plug-ins.*

Documentos VRML

Como se ha mencionado ya en este trabajo, VRML es un lenguaje de descripción de escenas en el que cada escena se compone de un número de objetos. Los objetos pueden ser formas sólidas situados y orientados de determinada forma u elementos intangibles que afectan a la escena como luces, sonido y distintos puntos de vista. Para crear estos mundos de realidad virtual se utilizan archivos de texto, cuya extensión será siempre ".wrl", los cuales pueden ser desarrollados mediante cualquier editor o procesador de textos. Además existe la posibilidad de utilizar programas de diseño gráfico, los cuales generan automáticamente archivos en formato VRML.

Todo documento VRML está compuesto por los siguientes elementos:

- Cabecera
- Comentarios
- Nodos

Cabecera:

La cabecera de todo archivo VRML es siempre la misma: ***#VRML V2.0 utf8.***

donde VRML V2.0 indica el estándar empleado y utf8 autoriza el uso de caracteres internacionales. Es importante resaltar que no debe existir ningún espacio en blanco entre el símbolo "#" y la palabra "VRML".

Comentarios:

En VRML un comentario se escribe en una sola línea, la cual comienza con el símbolo #. Se pueden tener tantas líneas de comentarios como se desee.

Nodos:

Un nodo es la estructura mínima indivisible de un archivo VRML y tiene como misión la de definir las características de un objeto o bien las relaciones entre distintos objetos. La mayoría de los nodos pueden repetirse tantas veces como sea necesario en una escena salvo excepciones.

Por otra parte, no todos los nodos afectan al aspecto visual del mundo. Por ejemplo, existen nodos que actúan como sensores que detectan acciones del usuario e informan de ellas a otros objetos y otros que se encargan de modelar sonidos. Los nodos a su vez contienen campos que describen propiedades. Todo campo tiene un tipo determinado y no se puede inicializar con valores de otro tipo.

De este modo, cada tipo de nodo tiene una serie de valores predeterminados para todos sus campos, de forma que cuando lo utilicemos en una escena sólo debemos indicar aquellos campos que se quieran modificar. Los campos pueden ser simples o campos que indiquen a vectores u otros nodos.

Estilo de escritura de los programas:

VRML es un lenguaje sensible a mayúsculas y minúsculas, lo cual ha de ser tenido en cuenta a la hora de asignar nombres. Todos los nodos han de comenzar siempre con letra mayúscula. Los campos de los nodos deben comenzar siempre con letra minúscula. Los números se escriben en punto flotante. Utilizar una línea distinta para cada nodo, para cada campo y para cada valor de cada campo. Indexar cada línea, según su jerarquía. Colocar cada símbolo de cierre en el nivel de indexación que le corresponda. Poner las líneas de comentario necesarias al mismo nivel que lo que se comenta. Poniendo nombres propios a los nodos

Un ejemplo de programa VRML sencillo sería el siguiente:

```
#VRML V2.0 utf8
#Esto es una línea de comentarios
Shape{
        appearance Appearance{
        material Material{ }
                                }
  geometry Cylinder{
     height 2.0
     radius 1.5
                        }
       }
```

Construcción de formas primitivas

Las *formas* (Shapes) son los elementos que nos permiten visualizar los objetos en los mundos VRML. La sintaxis del nodo Shape es la siguiente:

```
Shape{
appearance ...
geometry ...
          }
```

El campo appearance especifica las propiedades en cuanto a textura, material, etc del objeto que se describe en el campo geometry. Hablamos de formas primitivas cuando *Shape* utiliza nodos geométricos primitivos para construir una figura. Los nodos geométricos primitivos son los siguientes:

Box	(Caja)
Cone	(Cono)
Cylinder	(Cilindro)
Sphere	(Esfera)

Mediante la combinación de estas formas geométricas básicas se pueden obtener otras formas de mayor complejidad.

Nodo primitivo Box:

```
Box{ size 0 0 0
   }
```

Ejemplo:

```
Box{ size 2.0 0.5 3.0
   }
```

Las dimensiones que se manejan en VRML son dimensiones abstractas pero lo normal es suponer que la unidad de medida es el metro. De esta forma, en el ejemplo anterior estaríamos definiendo una caja de 2 metros de ancho, 0.5 metros de alto y 3 metros de profundidad.

Nodo primitivo Cone:

```
Cone {
height        #altura
bottomRadius  #radio_de_la_base
bottom        #valor_lógico
side          #valor_lógico
}
```

Mediante los campos bottom y side se indica si se desea dibujar la base y la superficie lateral respectivamente. Por defecto estos campos toman el valor **TRUE**, lo cual indica que se dibuja el cono completo.

```
Cone{
    height 3.0
    bottomRadius .75
   }
```

Nodo primitivo Cylinder:

Sintaxis:

```
Cylinder{
height        #altura
radius        #radio
bottom        #valor_lógico
side          #valor_lógico
top           #valor_lógico
}
```

Mediante los campos bottom, side y top se indica si se desea dibujar la base inferior, la superficie lateral y la base superior del cilindro. Por defecto estos campos toman el valor TRUE, lo cual indica que se dibuja el cilindro completo.

```
Cylinder  {
height 2.0
radius 1.5
          }
```

Nodo primitivo Sphere:

Sintaxis:

```
Sphere{  radius #radio
         }
```

Ejemplo:

```
Sphere{  radius 1.0
         }
```

Sin embargo, la definición de un nodo primitivo implica la definición de un objeto, pero no su visualización. Es por ello por lo que se han de englobar dentro de un nodo Shape, el cual determina la apariencia de estos objetos.

Ejemplo:

```
#VRML V2.0 utf8

        Shape{
          appearance Appearance{
            material Material {}
                                  }
          geometry Cylinder{
            height 2.0
            radius 1.5
                              }
             }
```

A este archivo podríamos darle el nombre de **"cilindro.wrl".**

Construcción de formas de texto

En los mundos virtuales es necesario en ocasiones usar textos para guiar al visitante Para ello existe un nodo específico, el nodo Text. Una de las principales características de los textos es que son planos, es decir, no tienen profundidad.

Nodo Text:

Como en cualquier procesador de textos, se nos permitirá indicar el tipo de fuente, su estilo, su tamaño, el espaciado entre caracteres, justificación de los párrafos, etc.

Sintaxis:

```
Text {
        string ["linea_texto 1",
                "linea_texto 2",
                        .
                        .
                        .
                "linea_texto N",]
        fontStyle FontStyle {
                family  "Nombre_Fuente",
                style   "Estilo_Fuente",
                size    Tamaño_Fuente
                spacing espaciado_entre_caracteres
                justify  "justificación_del_texto"
               }
}
```

Como se puede apreciar el nodo *Text* posee dos campos: string, que es donde se introduce el texto que se desea visualizar y FontStyle, un campo opcional, de manera que si se omite, el texto tendrá el estilo de la fuente por defecto.

Nodo FontStyle:

Sintaxis:

```
FontStyle {
        family "Nombre_Fuente",
        style "Estilo_Fuente",
        size Tamaño_Fuente
        spacing espaciado_entre_caracteres
        justify "justificación_del_texto"
       }
```

Los posibles valores de los campos del nodo *FontStyle* son los que se muestran a continuación:

family: Determina la fuente que se va a utilizar para el texto. Se puede escoger entre "SERIF", "SANS" o "TYPEWRITER". Obsérvese que los nombres están en mayúsculas.

style: Se puede escoger entre "BOLD" (negrita), "ITALIC" (cursiva), "BOTH" (negrita y cursiva) o "NONE" (tipo de letra normal).

size: Determina el tamaño de la fuente, pero en unidades VRML.

spacing: Determina la separación entre líneas, también en unidades VRML.

justify: Determina la justificación del texto. Puede ser "BEGIN" (Alinear a la izquierda), "MIDDLE" (centrar el texto) o "END" (Alinear a la derecha).

Ejemplo: Text {

```
string ["Esta es la primera fila de texto",
 "esta es la segunda fila",
 "etc."]

    fontStyle FontStyle {      family "SERIF",
                               style "BOLD",
                               size 1.0
                               spacing 1.0
                               justify "BEGIN"
                              }

}
```

De igual manera que con los nodos geométricos primitivos, mediante el nodo Text lo único que se consigue es definir la estructura del texto, sin embargo no se puede visualizar, ya que no hemos indicado como se ha de presentar en el mundo virtual. Para conseguir esto, se integra en el nodo Shape, de la misma manera que se hacía con los nodos primitivos:

```
Shape {
appearance ...
geometry Text { ... }
        }
```

Una vez que el texto se encuentra en el mundo virtual se puede manipular como cualquier otro objeto (girándolo, etc.), ya que lo único que lo diferencia de los nodos primitivos es que posee dos dimensiones en lugar de tres.

Ejemplo:

```
#VRML V2.0 utf8
Shape{  appearance Appearance{
          material Material {}
                                    }
geometry Text {

    string ["Esta es la primera fila de texto",
        "esta es la segunda fila",
        "etc."]

    fontStyle FontStyle {
        family "SERIF",
        style "BOLD",
        size 1.0
        spacing 1.0
        justify  "BEGIN"
                          }

            }
          }
```

A este archivo podríamos darle el nombre de **"texto.wrl"**.

Agrupación de nodos.

Hasta ahora hemos visto los objetos aisladamente. Veamos ahora cómo podemos agruparlos para conseguir formas más complejas. Existen diversos nodos que nos permiten agrupar objetos:

- Nodo Group
- Nodo Transform
- Nodo Switch
- Nodo Billboard

Nodo Group:

El nodo Group permite unir un conjunto de nodos de forma que actúen como una entidad única, pero sin efectuar ninguna transformación en ellos. La principal característica de este tipo de grupo es que los objetos son creados todos en el mismo punto (en el centro del escenario de realidad virtual).

Sintaxis:

```
Group {
        children [ ... ]
        }
```

El campo children contiene la lista de los objetos que se quieren agrupar, representados por sus nodos Shape respectivos:

Sintaxis:

```
Group {
    children [
            Shape { ... },
            Shape { ... },
                    ...
            ]
    }
```

Ejemplo:

```
#VRML V2.0 utf8
#Ejemplo de agrupación de una caja y un cono
Group {
   children [  #Aquí empieza la caja:
        Shape {
         appearance Appearance {
         material Material {  }
                                    }
         geometry Box { size 2.0 0.5 3.0
                              }
               },
      #Aquí empieza el cono:
      Shape {
           appearance Appearance {
           material Material {  }
              }
           geometry Cone {
           height 3.0
           bottomRadius 0.75
         }
      }
   ]
}
```

El archivo asociado a este ejemplo es "group.wrl".

Nodo Transform:

Por defecto todos los objetos (Shapes) se construyen en el centro del escenario virtual. El nodo Transform nos va a permitir evitar esto, indicando la posición, orientación y tamaño de los diferentes objetos que va a crear.

Sintaxis:

```
Transform{
    translation #Eje_X Eje_Y Eje_Z
    rotation    #Eje_X Eje_Y Eje_Z Ángulo
    scale       #Eje_X Eje_Y Eje_Z
    children[...]
  }
```

Cada grupo creado mediante el nodo Transform va a poseer su propio sistema de coordenadas, cuyos atributos se determinan a través de los campos translation, rotation y scale, los cuales son optativos.

El campo translation permite indicar la posición del origen del nuevo sistema de coordenadas perteneciente al grupo dentro del sistema de coordenadas de nodo que lo engloba (nodo padre). A través del siguiente ejemplo esta idea quedará más clara:

Ejemplo:

```
Transform{
    # Ejes:       X  Y  Z
      translation 2.0 0.0 0.0
      children [...]
        }
```

Mediante este ejemplo se consigue que el grupo creado tenga un sistema de coordenadas idéntico a el sistema de coordenadas principal, con la única salvedad de que su origen se encontraría desplazado dos unidades sobre el eje X del sistema principal. También es posible apreciar en este ejemplo que sólo se han de contemplar los campos que interesen, ignorándose el resto.

El campo rotation nos permite girar el sistema de coordenadas del grupo alrededor de uno de los ejes del sistema de coordenadas del nodo padre. Para ello, además de indicar sobre que eje se desea realizar el giro, se ha de hacer referencia al grado de inclinación de dicho giro (en radianes).

Ejemplo:

```
Transform{
    # Ejes:   X  Y  Z  Ángulo
      rotation 0.0 0.0 1.0  0.52
      children [...]
          }
```

Con este ejemplo se pretende hacer girar el sistema de coordenadas del grupo sobre el eje Z a 0.52 radianes. Nótese que solamente uno de los ejes puede tomar un valor (que ha de ser forzosamente la unidad) mientras el resto ha de permanecer a cero. De esta forma sólo hay tres combinaciones posibles con las que rellenar los ejes del campo *rotation*:

Rotación sobre el eje X	1.0	0.0	0.0
Rotación sobre el eje Y	0.0	1.0	0.0
Rotación sobre el eje Z	0.0	0.0	1.0

A través del campo scale podemos aumentar o reducir el tamaño de los ejes del sistema de coordenadas del grupo utilizando factores de escala que toman como referencia los ejes de coordenadas del sistema del nodo padre. De esta forma aumentamos o disminuimos el tamaño de los objetos que se crean.

Ejemplo:

```
Transform{
    # Ejes:   X  Y  Z
      scale     0.5 0.5 0.5
      children [...]
    }
```

El ejemplo anterior hace que los ejes del sistema de coordenadas del grupo sean un 50% (0.5) más pequeños que los ejes del sistema principal y por lo tanto el objeto diseñado en estos ejes reducirá su tamaño a la mitad. Si se hubiese querido que fuesen el doble de grandes, el ejemplo hubiese sido el siguiente:

Ejemplo:

```
Transform{
        # Ejes:   X  Y  Z
          scale    2   2   2
          children [...]
        }
```

Por último, se muestra un ejemplo en el que se unen las diferentes modificaciones sobre el sistema de coordenadas de un grupo:

Ejemplo:

```
Transform{
        # Ejes:    X   Y  Z    Ángulo
          translation 2.0 0.0 0.0
          rotation 0.0 0.0 1.0   0.52
          scale      0.5 0.5 0.5
          children [...]
        }
```

Nodo Switch:

La principal característica de un nodo Switch es la de mostrar únicamente uno de los nodos hijos del nodo, el cual ha debido ser seleccionado previamente. Se pueden utilizar para conectar o desconectar los efectos de una determinada propiedad o para alternar entre propiedades diferentes.

El campo whichChild especifica el hijo choice que se va activar, siendo el 0 el del primer hijo. Su valor por defecto es -1, lo cual indica que ninguno de los hijos está seleccionado. El campo *whichChild* es una entrada y por lo tanto puede ser modificado por otro nodo.

Sintaxis:

```
  Switch{
whichChoice 0
choice[...]
        ...
choice[...]
        }
```

Nodo Billboard:

El nodo Billboard permite crear un grupo con un sistema de coordenadas especiales, ya que a través del campo axisOfRotation (eje de rotación) indicamos el eje sobre el que a de girar el objeto, de forma que, siempre esté de cara al espectador. Todos los hijos agrupados mediante este nodo serán visualizados.

Sintaxis:

```
Billboard{
axisOfRotation  Eje_X  Eje_Y  Eje_Z
children[...]
  }
```

Al igual que para el campo rotate del nodo transform, únicamente uno de los ejes puede tomar como valor la unidad, permaneciendo el resto a cero:

Rotación sobre el eje X	1.0	0.0	0.0
Rotación sobre el eje Y	0.0	1.0	0.0
Rotación sobre el eje Z	0.0	0.0	1.0

Un ejemplo completo se encuentra en el archivo **"robobill.wrl"**.

Poniendo nombres propios a los nodos:

Hay una solución prevista para simplificar las repeticiones de objetos dentro de un escenario virtual. Esta solución consiste en asignar un nombre arbitrario al nodo que se piensa repetir en el código. Supongamos, por ejemplo, que se van a utilizar repetidamente en un escenario unos cilindros exactamente iguales (que podrían ser las columnas de una casa). Dichos cilindros tendrán el siguiente código:

```
Shape { appearance Appearance {
          material Material { }
                                }
        geometry Cylinder { height 2.0
                            radius 0.5
                          }
      }
```

Este es el código de un cilindro con la apariencia por defecto, de 2 unidades de alto y una base de radio 0.5.

Se puede definir, para el ámbito de un documento VRML, que este tipo de cilindro tenga un nombre arbitrario, por ejemplo ColumnaRepetida (o como se desee, con tal de que comience por mayúscula), de la siguiente manera:

```
DEF ColumnaRepetida Shape {
                        appearance Appearance {
                    material Material { }
                                        }
                    geometry Cylinder {
              height 2.0
              radius 0.5
                                          }
                                        }
```

Entonces, cada vez que se quiera usar este nodo en otra parte del documento, basta con poner lo siguiente:

USE ColumnaRepetida

En el ejemplo anterior de la caja y el cono, aparece el nodo *Appearance* repetido. Vamos a definirlo, en la primera ocasión que se utiliza con el nombre, "PorDefecto" y la segunda vez que se usa lo invocaremos mediante el comando *USE*:

```
#VRML V2.0 utf8
#Ejemplo de agrupación de una caja y un cono,
#haciendo uso de los comandos DEF y USE.
Group {
  children [
    Shape {
      appearance DEF PorDefecto Appearance {
        material Material {  }
          }
      geometry Box {
        size 2.0 0.5 3.0
                  }
                                                  },
    Shape {
      appearance USE PorDefecto
      geometry Cone {
                          height 3.0
                          bottomRadius 0.75
                  }
          }
        ]
      }
```

En este caso concreto, la simplificación no ha sido muy grande (sólo un par de líneas menos de código), pero ha servido para ilustrar el concepto.

El color de los objetos

Anteriormente se ha comentado que el nodo *Shape* contenía dos campos, *appearance* y *geometry*, de los cuales el segundo indicaba el tipo de objeto a representar y del que se ha hablado ya ampliamente. Sin embargo la misión del campo *appearance* apenas se ha comentado, por lo que procederemos a analizarla con más detalle en este punto.

El campo appearance va a permitir seleccionar el color y la textura del objeto que va a ser representado dentro del escenario virtual. Este campo toma como valor un nodo de tipo Appearance, el cual a su vez, posee un campo denominado material que toma como valor un nodo de tipo Material.

El nodo Material es el que controla las propiedades del color (selección del color, del brillo, del grado de transparencia, etc.) que se van a dar al objeto.

Los colores que se le dan a los objetos son colores RGB, es decir, vienen dados por tres valores en coma flotante, cada uno de los cuales representa uno de los colores primarios (Red, Green, Blue) [Rojo, Verde y Azul]. El valor 0.0 representa la ausencia de color y el 1.0 la máxima intensidad.

Muchos programas de dibujo darán un valor para cada color RGB en un formato 256x256x256. Para poder utilizar estos colores en VRML es preciso convertirlos, dividiendo cada valor por 256 y colocando el resultado en su campo correspondiente dentro del nodo *Material*.

Nodo Material:

Sintaxis:

```
Shape{
        appearance Appearance{
        material Material{
diffuseColor        #color_RGB
emissiveColor       #color_RGB
specularColor       #color_RGB
ambientIntensity    #valor
transparency        #valor
shininess           #valor
                        }
                                }
 geometry ...
        }
```

Cada uno de los seis campos del nodo *Material* tiene su propio efecto específico sobre un objeto y todos son opcionales.

diffuseColor: Este campo representa lo que la mayoría de los usuarios llamarían como el color del objeto.

emissiveColor: Se utiliza para fijar el color del brillo del objeto, cuando dicho objeto necesite ser visible en la oscuridad. De esta forma se consigue un efecto en donde la figura representada parece iluminada desde el interior mediante una luz de un determinado color. La configuración por defecto de este campo es el negro, ya que la mayoría de los objetos normalmente no brillan.

specularColor: El campo *specularColor* es un parámetro avanzado que permite indicar qué color de luz **refleja** el objeto. Por ejemplo, una cama roja no refleja un color rojo, pero una olla rojiza si puede reflejar su color.

ambientIntensity: Este campo es otro parámetro avanzado que indica la cantidad de luz ambiental (producida por los diferentes focos de luz del escenario virtual) es reflejada por el objeto. Toma valores en coma flotante entre 0.0 y 1.0.

shininess: El campo *shininess* controlan el brillo de un objeto. Toma valores en coma flotante entre 0.0 y 1.0.

transparency: El campo *transparency* indica el nivel de transparencia del objeto. Toma valores en coma flotante entre 0.0 y 1.0, siendo el 1.0 el nivel máximo de transparencia (objeto invisible) y el 0.0 el nivel mínimo (objeto totalmente opaco). El valor por defecto es el 0.0.

Objetos almacenados en archivos

Para facilitar el diseño de mundos virtuales habrá ocasiones en las que convendrá almacenar cada objeto en su propio archivo. De este modo, si por ejemplo se deseara modelar una habitación, los diferentes elementos que van a aparecer dentro de ella: paredes, puertas, mesas, sillas, etc, son objetos independientes entre si pero que se engloban dentro de un mismo espacio y, que además, pueden aparecer varias veces en el diseño de todo mundo virtual.

Es por ello por lo que sería de interés crear un archivo donde almacenar el objeto mesa ("mesa.wrl"), otro archivo para el objeto silla ("silla.wrl"), etc. En el archivo "comedor.wrl" se realizan llamadas tanto al archivo que contiene la silla como al que contiene la mesa.

Nodo Inline:

El nodo Inline va a permitir crear un grupo en donde los hijos, almacenados en distintos ficheros VRML, son recuperados indicando su dirección URL.

Sintaxis:

```
Inline{
url "dirección_url"
}
```

Ejemplo:

```
...
Inline{
        url"mesa.wrl"
       },
        ...
        Transform{
                translation ...
                        children [
                                Inline{url"silla.wrl"}
                                  ]
           }
                            ...
```

Adición de enlaces en el mundo virtual

Nodo Anchor:

El nodo Anchor crea un grupo especial ya que seleccionando cualquier objeto perteneciente a dicho grupo se salta hacia otro lugar del escenario virtual o hacia otro mundo virtual almacenado en un fichero VRML (al cual accedemos a través de su dirección URL). Cualquier objeto o grupo de objetos se puede convertir en un enlace. Estos enlaces son los equivalentes en el mundo tridimensional a los enlaces existentes en las páginas Web realizadas mediante HTML. Además, todo nodo Anchor posee un campo denominado *description* en el que mediante una cadena de texto se describe brevemente el objeto.

Sintaxis:

```
Anchor{
        url"dirección_URL"
        description "descripción_del_enlace"
        children[...]
 }
```

Ejemplo:

```
Anchor{
        url"escalera.wrl"
        description "Escaleras Flotantes"
        children[...]
}
```

Construcción de formas complejas.

Se ha visto en temas anteriores como mediante la superposición y unión de diferentes nodos geométricos básicos es posible construir objetos más complejos. Sin embargo, si la única manera de generar formas complejas fuese ésta, la construcción de mundos virtuales será bastante ardua. Es por ello por lo que aparecen una serie de nuevos nodos, los cuales, además de facilitar el diseño (al tener más control sobre ellos y al ser más flexibles), generan mundos VRML de una forma más eficiente. Lo más recomendable a la hora de diseñar un objeto es describir su geometría en dos pasos aislados:

- Especificación de las coordenadas
- Creación de la forma geométrica

Especificación de las coordenadas:

En este paso se ha de indicar mediante el nodo Coordinate la posición de los puntos que se van a utilizar para construir el objeto. Estos puntos no son visibles en el escenario virtual.

Nodo Coordinate:

Sintaxis:

```
Coordinate {
 point [
 Eje_x Eje_Y Eje_Z,
 Eje_x Eje_Y Eje_Z,
       ...
 Eje_x Eje_Y Eje_Z
       ]
      }
```

El campo point puede poseer varios puntos, cuyas coordenadas están separadas por comas.

Ejemplo:

```
Coordinate {
 point [
       12.0 11.0 17.1,
       20.5 13.8 5.3,
       14.0 6.1 22.9
       ]
      }
```

Creación de la forma geométrica:

Una vez definidos los puntos que forman el esqueleto del objeto, se han de conectar. Existen tres maneras de unir estos puntos, cada una de las cuales está asociada a un nodo diferente:

- **Nodo PointSet**
- **Nodo IndexedLineSet**
- **Nodo IndexedFaceSet**

Estos tres nodos poseen un campo denominado coord que acepta como valor un nodo *Coordinate*.

Nodo PointSet:

Representa un conjunto de puntos situados en las coordenadas indicadas dentro del sistema de coordenadas del nodo padre.

Sintaxis:

```
PointSet {
coord Coordinate {
                point [ . . . ]
                  }
color Color {
                color [ . . . ]
            }
    }
```

El campo coord toma como valor un nodo de tipo *Coordinate*, el cual define los puntos que se desean representar. El campo color sirve para definir el color de cada uno de los puntos. Este campo toma como valor un nodo Color, que a su vez contiene un campo de tipo color de nuevo. Este último campo describe una lista de colores RGB, de forma que el primer color corresponde al primer punto descrito por el nodo *Coordinate* del campo *coord* ; el segundo color corresponde al segundo punto y así sucesivamente.

Ejemplo:

```
PointSet {
        coord Coordinate {
                point [
                        12.0 11.0 17.1,  #1º punto
                        20.5 13.8 5.3,   #2º punto
                        14.0 6.1 22.9    #3º punto
                       ]
                          }
 color Color {
   color [
      1.0 0.0 0.0, # 1º punto rojo
      0.0 1.0 1.0, # 2º punto verde
      1.0 1.0 0.0  # 3º punto amarillo
         ]
            }
         }
```

Si se incluirá este código, tal como está, en un documento VRML, no podríamos ver ninguno de estos puntos, ya que como se ha visto anteriormente, para crear un objeto visible se debe utilizar el nodo *Shape*. Obsérvese que se ha prescindido del campo appearance del nodo Shape ya que no es necesario, pues los puntos no van a tener la apariencia por defecto, sino la que se determina en el campo color.

Ejemplo:

```
#VRML V2.0 utf8
#Ejemplo de un grupo de tres puntos con colores
        Shape {
              geometry PointSet {
       coord Coordinate {
                     point [
                            12.0 11.0 17.1,  #1° punto
                            20.5 13.8 5.3,   #2° punto
                            14.0 6.1 22.9    #3° punto
                           ]
                                 }
       color Color {
                     color [
                            1.0 0.0 0.0, # 1° punto rojo
                            0.0 1.0 1.0, # 2° punto verde
                            1.0 1.0 0.0  # 3° punto amarillo
                           ]
                    }
                                    }
             }
```

Nodo IndexedLineSet: Permite unir los diferentes puntos especificados en su campo *coord* mediante líneas poligonales.

Sintaxis:

```
      IndexedLineSet{
coord Coordinate {
                    point [ . . . ]
                     }
coordIndex  [...]
colorPerVertex    #valor_lógico
color Color {
    color [ . . . ]
                }
colorIndex  [...]
                    }
```

El campo coord toma como valor un nodo de tipo *Coordinate*, el cual define los puntos que sirven como esqueleto de una figura.

El campo coordIndex se utiliza para especificar entre qué puntos se han de trazar las líneas. Una línea puede ser trazada utilizando más de dos puntos, de forma que se dibuja una línea entre el primer y el segundo punto, otra entre el segundo y el tercer punto, y así sucesivamente. Un índice con valor -1 indica que ha finalizado la línea actual y que comienza la siguiente.

Para indicar el primer punto definido en el nodo *Coordinate* se utiliza el 0, para el segundo el 1 y así sucesivamente. El campo colorPerVertex indica como se han de aplicar los colores sobre las líneas:

Nodo IndexedFaceSet:

Permite unir los diferentes puntos especificados en su campo *coord* mediante caras poligonales.

Sintaxis:

```
    IndexedFaceSet{
coord Coordinate {
    point [ . . . ]
                    }
coordIndex  [...]
colorPerVertex    #valor_lógico
color Color {
    color [ . . . ]
            }
colorIndex  [...]
                    }
```

El campo coord toma como valor un nodo de tipo *Coordinate*, el cual define los puntos que sirven como esqueleto de la figura. Utiliza los índices de su campo coorIndex para especificar las caras poligonales. Un índice con valor -1 indica que ha finalizado la cara actual y comienza la siguiente.

Utilización de texturas

La textura es la posibilidad que tenemos de envolver un objeto con una imagen determinada que se encuentra almacenada en un fichero, al cual accedemos mediante su URL. Los tipos de imagen que soporta VRML son: **JPEG, GIF y PNG.**

Hasta ahora, para definir un objeto visible se ha utilizado el nodo *Shape* de la siguiente forma:

```
    Shape {
appearance Appearance {
   material ...
                        }
geometry ...
            }
```

en donde el nodo *Appearance* tiene un solo campo, *material*, con el que se definen el color y la transparencia, según se ha visto en temas anteriores.

Pero en realidad puede tener también otros dos campos: **texture** (cuyo valor suele ser un nodo de tipo *ImageTexture* o de tipo *MovieTexture*) y **textureTransform**, con los que se define la textura de los objetos:

```
Shape {
appearance Appearance {
   material ...
   texture ImageTexture{...}
   textureTransform {...}
                        }
   geometry ...
            }
```

Nodo ImageTexture:

Sintaxis:

```
ImageTexture{
        url       "direccion_URL"
        repeatS   #valor_lógico
        repeatT   #valor_lógico
         }
```

El campo url contiene la dirección URL del fichero gráfico que se va a usar como textura. Los formatos gráficos que admite VRML son jpeg,gif y png. Las texturas definen un conjunto de coordenadas 2D (Ejes S y T) que se emplean para trazar texturas sobre la superficie de los objetos. Las coordenadas de textura van de 0 a 1. La coordenada horizontal se denominada S y la coordenada vertical T. El lado inferior de la imagen se corresponde con el eje S y el lado izquierdo con el eje T. La esquina inferior izquierda de la imagen, según este sistema de coordenadas, vendría dada por el punto (S=0,T=0) y la esquina superior derecha por (S=1,T=1).

Los campos repeatS y repeatT determinan como envuelve la textura en las direcciones S y T. Si repeatS es TRUE, la textura se repite (cada vez que S=1) en la dirección S hasta cubrir la superficie del objeto. Si repeatS es FALSE, la textura se adapta a toda la superficie del objeto a lo largo de la dirección S, sin tener en cuenta el valor de las coordenadas S y T. El campo repeatT haría lo mismo sobre el eje T.

Nodo TextureTransform:

Este nodo define una transformación 2D aplicada a las coordenadas de textura. Esto afecta a la forma en que se aplica la textura a las superficies de los objetos. La transformación consiste (por orden) en un ajuste de la escala no uniforme sobre un punto central arbitrario, una rotación sobre ese mismo punto y una translación. Esto permite al usuario modificar el tamaño, orientación y posición de las texturas de los objetos.

Sintaxis:

```
TextureTransform{
   center       Eje_S  Eje_T
   rotation     ángulo
   scale        Eje_S Eje_T
   translation  Eje_S  Eje_T
                }
```

El campo center especifica un punto en el sistema de coordenadas de la textura (S,T) sobre el que los campos rotation y scale van a ser aplicados. El campo scale contiene dos factores de escala, uno para el eje S y otro para el eje T de la textura. Puede tomar cualquier valor real en ambos ejes.

El campo rotation determina la rotación en radianes de los ejes de coordenadas de la textura con respecto al punto marcado por el campo center, después de haber aplicado el campo *scale*. Por último, el campo translation provoca un desplazamiento del sistema de coordenadas de la textura.

Ejemplo gráfico:

Es importante destacar que todas estas modificaciones aparecen invertidas cuando la textura se aplica sobre la superficie de un objeto. De esta forma si duplicásemos los ejes S y T de una textura (*rotate 2 2*) cuando se aplicase sobre un objeto se observaría que su tamaño se ha reducido a la mitad. De igual modo, si desplazáramos la textura 0.5 unidades con respecto al eje S (*translation 0.5 0*) cuando se aplicase sobre un objeto se podría apreciar que se ha desplazado -0.5 unidades con respecto a la superficie del objeto.

Imágenes distintas en un mismo objeto:

Puede interesar que un objeto tenga imágenes diferentes en alguna o algunas de sus caras, para ello hemos de diseñar la figura de forma que cada una de sus caras sean objetos independientes. Supongamos que se desea poner una determinada textura sobre las bases de un cilindro y una textura diferente sobre la superficie lateral, los pasos a seguir serían los siguientes:

- Se define un cilindro con la textura de la base, anulando las superficies superior e inferior.
- Se define otro cilindro idéntico con otra textura, anulando la superficie lateral.
- Por medio del nodo *Group* se agrupan ambos cilindros, con lo que el resultado es aparentemente un cilindro con imágenes distintas.

Utilización de fondos

La utilización de fondos en el mundo virtual, nos permite dotarlos de un cielo y de un suelo, añadiendo realismo de esta forma a la escena que se pretende crear. Estos fondos se van a caracterizar porque siempre le van a dar al visitante la sensación de que se encuentran a una gran distancia. Por otra parte, su coste es menor que el uso de geometrías. Se distinguen dos tipos de fondos:

Backdrop

Este tipo de fondo se caracteriza por definir un cielo y un suelo cuyos colores vienen dados mediante gradientes. Se realiza dentro de una esfera (Sphere).

Panorama

Este tipo de fondo se caracteriza porque define una serie de imágenes que rodean la escena. Se realiza dentro de una caja (Box).

Nodo Background:

Incorpora un plano de suelo sombreado, texturas y cielo escénico. Sólo se emplea el primer nodo Background que se encuentre, debiéndose especificar en el archivo principal.

Sintaxis:

```
Backgroud{
  groundAngle    [ ]
  groundColor    [ ]
  skyAngle       [ ]
  skyColor       [ ]
  backUrl        "dirección_URL"
  bottomURL      "dirección_URL"
  frontUrl       "dirección_URL"
  leftUrl        "dirección_URL"
  rightUrl       "dirección_URL"
  topUrl         "dirección_URL"
        }
```

Todos estos campos no se utilizan simultáneamente, sino que su uso dependerá del tipo de fondo que se desee crear.

Backdrop

Sintaxis:

```
Backgroud{
  groundAngle    [ ]
  groundColor    [ ]
  skyAngle       [ ]
  skyColor       [ ]
        }
```

Ejemplo:

```
Backgroud{
        groundAngle    [.785]
        # color_RGB1,color_RGB2
        groundColor    [ .14 .28 0,.09 .11 0 ]
          skyAngle       [.785 ]
        # color_RGB1,color_RGB2
        skyColor       [.02 0 .26, .02 0 .65 ]
    }
```

El fondo de tipo Backdrop se construye mediante una esfera incompleta (el suelo) inmersa dentro de otra esfera completa (el cielo), en donde el visitante se sitúa en el centro de ambas. Estas esferas tienen un radio infinito.

El campo skyColor determina el color del cielo en los distintos ángulos de la esfera que lo contiene. El primer valor de este campo determina el color del cielo en los 0.0 radianes de la esfera, es decir, el color que tiene el cielo en el lugar donde se une con el suelo.

El campo skyAngle es utilizado para indicar los ángulos (en radianes) en los que un nuevo color debe aparecer. El campo *skyColor* debe poseer N+1 colores si en *skyAngle* se han definido N ángulos, ya que el ángulo 0.0 (que es el que se corresponde con el cielo del horizonte) no se incluye en este último.

Los campos groundColor y groundAngle, son equivalentes a *skyColor* y *skyAngle* respectivamente, pero referidos a la esfera que representa el suelo. Si se especifica más de un color, se interpolará el color del suelo entre los colores de 0 a 90 grados en el horizonte (plano X-Z). De forma similar se interpretan los colores del cielo, de 90 a 180 grados en la vertical (plano X-Y).

Sintaxis:

```
Backgroud{
   backUrl       "dirección_URL"
   bottomURL     "dirección_URL"
   frontUrl      "dirección_URL"
   leftUrl       "dirección_URL"
   rightUrl      "dirección_URL"
   topUrl        "dirección_URL"
         }
```

Ejemplo:

```
Backgroud{
   backUrl          "ba_image.jpg"
   bottomURL        "bo_image.jpg"
   frontUrl          "f_image.jpg"
   leftUrl           "l_image.jpg"
   rightUrl          "r_image.jpg"
   topUrl            "t_image.jpg"
         }
```

El fondo de tipo Panorama consiste en seis imágenes, cada una de las cuales ocupa una de las caras de una inmensa caja centrada en el eje de coordenadas. Las imágenes que ocupan cada una de las caras vendrán determinadas por los valores de los campos backUrl, bottomUrl, frontUrl, leftUrl, rightUrl, topUrl. Todos estos campos toman como valor una dirección URL de un fichero que contiene una imagen en formato jpeg, gif o png.

Nodo Collision:

Indica al navegador que objetos de la escena no se van a poder atravesar. Esto permite evitar, por ejemplo, que los visitantes traspasen las paredes de un edificio. La respuesta a la colisión la define el navegador (haciendo que se rebote en el objeto, deteniéndose simplemente, y otros).

Dado que es muy costoso el cálculo de una colisión con una geometría compleja, para aumentar la eficacia se puede utilizar un método que consiste en definir una geometría alternativa que sirva como sustituto para colisiones. Esta geometría podría ser tan imperfecta como un simple cuadrado o una esfera. Este volumen alternativo se usa solamente para calcular la colisión con el visualizador. VRML ofrece volúmenes alternativos de colisión para objetos mediante el nodo *Collision*.

Sintaxis:

```
Collision{
  collide      valor_lógico
  proxy        nodo
  children     [...]
       }
```

Si el valor del campo collide es FALSE entonces no se realiza colisión con la geometría afectada; si es TRUE el campo proxy define la geometría contra la que se hace la prueba de colisión. Si el valor de proxy no está definido, se colisionará con la geometría real, y si esta definido, contendrá la geometria que se emplea para calcular las colisiones.

Cabe mencionar que este manual tiene un nivel básico de aprendizaje del lenguaje, pero lo considero suficiente como base. Se debe recordar que siempre se le deben dar bases de este tipo a los usuarios que se inician en una temática y que dependiendo de su interés, el mismo usuario buscará más alternativas y profundidad para desarrollos más avanzados. Para ello sugiero las ligas o vínculos citadas en "Guia de Internet", que ayudarán a complementar el conocimiento de aquellos que deseen llegar más lejos en este lenguaje y a su ves en la Realidad Virtual.

Glosario.

3D.
Tridimensional. En tres dimensiones.

API (Interfaz para programa de Aplicación).
Conjunto de convenciones de programación que definen cómo se invoca un servicio desde un programa, así también es una biblioteca específica con acuerdos de llamado para que un programa pueda enlazarse y crear una aplicación ejecutable. Por ejemplo, OpenGL, Reality Labs, renderWare, etc.

Applet.
Aplicación escrita y compilada en Java que se difunde a través de la red para ejecutarse en un navegador cliente.

Aplicación.
Software que realiza una función útil. Los programas que se utilizan para realizar alguna función (como correo electrónico, FTP, etc.) son llamadas aplicaciones cliente.

CERN.
Laboratorio Europeo de Física de Partículas. Fue el desarrollador inical del World Wide Web. Actualmente los estándares del Web son desarrollados por la World Wide Web Organization (3W0). El web site del CERN se encuentra en http://www.cern.ch.

CGI (Common Gateway Interface).
Interfaz escrita en un lenguaje de programación (perl, c, c++,visual basic, etc.) y posteriormente ejecutada o interpretada por una computadora servidor para contestar a pedidos del usuario desde una computadora con una aplicación cliente; casi siempre desde el World Wide Web. Esta interfaz permite obtener los resultados pedidos, como los que resultan al consultar una base de datos.

Ciberespacio.
Termino originado por William Gibson en su novela Neuromancer. La palabra Cyberspace es ampliamente usada para descubrir los recursos de información disponibles a través de Internet.

Cliente.
a) Una aplicación que permite a un usuario obtener un servicio de un servidor localizado en la red.
b) Un sistema o proceso que solicita a otro sistema o proceso que le preste un servicio.

DOF (grado de libertad).
Representa la capacidad de variación en un parámetro. Los cuerpos libres tienen tres niveles de libertad para posicionarse y orientarse en cualquier lugar del espacio.

FTP (Protocolo de Transferencia de Archivos).
Protocolo de alto nivel que permite el copiado de archivos entre sistemas y que requiere de componentes para cliente y servidor.

Hipermedia.
Combinación de texto y multimedia. Actualmente es un recurso ampliamente explotado en el World Wide Web.

HTML (Hiper Text Markup Languaje).
Lenguaje de marcado de hipertexto, es el lenguaje con que se escriben los documentos en el World Wide Web. A la fecha existen tres versiones de HTML. HTML 1, donde se sientan las bases para la disposición del texto y las gráficas, HTML 2 donde se agregan formas y HTML 3 (llamado también extensiones Netscape) donde se añaden cuadros, mapas, etc.

HTTP (Hiper Text Transfer Protocol).
Protocolo de Transferencia de Hipertextos. Es el protocolo usado por el Word Wide Web para transmitir páginas HTML.

Hypertexto.
Documentos que contienen vínculos con otros documentos, al seleccionar un vínculo automáticamente se despliega el segundo documento.

Iluminación.
Proceso relacionado con el cálculo de la cantidad de luz que llega a una superficie y el tipo de luz que se refleja en ella.

Internet.
Es una red de cómputo a nivel mundial que agrupa a distintos tipos de redes usando un mismo protocolo de comunicación. Los usuarios en Internet pueden compartir datos, recursos y servicios. Internet se apoya en el conjunto de protocolos TCP/IP. Las computadoras que lo integran van desde modestos equipos personales, minicomputadoras, estaciones de trabajo, mainframes hasta supercomputadoras. Internet no tiene una autoridad central, es descentralizada. Cada red mantiene su independencia y se une cooperativamente al resto, respetando una serie de normas de interconexión. El organismo que se encarga de regular, establecer estándares, administrar y hacer operacional a Internet es la ISOC (Internet Society).

IP.
Protocolo de Internet, junto con TCP, es uno de los protocolos fundamentales en el manejo de redes. IP es responsable de direcciones y envío de datos en Internet.

Java.
Lenguaje de programación que permite ejecutar programas escritos en un lenguaje muy parecido al C++, llamados applets, a través del World Wide Web. La diferencia contra un CGI es que la ejecución se realiza totalmente en la computadora cliente, en lugar del servidor. Java fue originalmente desarrollado por Sun Microsystems (http://www.sun.com). El principal objetivo de JAVA fue hacer un lenguaje que fuera capaz de ser ejecutado de una forma segura a través de Internet. Esta característica requiere la eliminación de muchas construcciones y usos de C y C++. El más importante, es que no existen punteros. Java no puede acceder arbitrariamente a direcciones de memoria. Java es un lenguaje compilado en un código llamado "codigo-byte" (byte-code). Este código es interpretado "al vuelo" por el interprete Java.

Luz.
Fuente luminosa que se define mediante los siguientes factores: posición, vector de dirección, ángulo de descenso, ángulo de diseminación y color.

Mapa de Luz.
Consiste en un proceso de iluminación pixel a pixel para la escena, indicando la cantidad de luz y color que cae sobre cada lugar, para colorear la imagen correctamente.

Mapa de Textura.
Imagen bitmap escaneada o dibujada que proporciona a un material cualidades únicas que no están disponibles simplemente variando los atributos de superficie. Es una forma de controlar el color difuso de una superficie pixel por pixel, en lugar de asignar un valor total.

Matriz de transformación.
En VRML, es un arreglo de 4x4 de números con punto flotante que permite representar la transformación de alguna orientación y posición a una orientación y posición diferentes; otra aplicación que tiene, es aumentar o disminuir la escala de los objetos. También se conoce como transformador.

MIME (Multipurpose Internet Mail Extensions).
Extensiones de Correo de Internet de Múltiples propósitos. Técnica para codificar archivos y anexarlos a un mensaje de correo electrónico. Permite principalmente enviar archivos binarios como parte de un mensaje.

Mosaic.
Visualizador para el World Wide Web. Fue el primer visualizador para los ambientes Macintosh, UNIX y Windows, desarrollado por la NCSA (http://www.ncsa.uiuc.edu/).

NCSA (National Center for Supercomputing Applications).
Centro Nacional de Aplicaciones de Supercómputo. Desarrolladores del visualizador Mosaic para el World Wide Web.

Orientación.
Posición de un objeto. Esta puede especificarse como giro, vuelta y rodado; como rotación en las coordenadas x, y, z; como ángulos Euler (azimut, altitud y movimiento giratorio); o como un vector de dirección.

Página Web.
Es el resultado en hipertexto e hipermedia que proporciona un visualizador de World Wide Web después de obtener la información solicitada.

Parser.
Intérprete.

Pixel.
Abreviatura de Picture Element (elemento de imagen). Se trata de puntos aislados en la pantalla de la computadora; el color de los mismos se determina con el valor de cada pixel.

Plugins.
Programas que se agregan a un visualizador del Word Wide Web que realizan funciones determinadas. Estas pueden ser visualización de archivos multimedia, soporte a archivos gráficos no estándares con el visualizador, etc.

Posición de cámara.
Posición en el espacio donde se coloca la cámara o el enfoque visual. También conocido como punto de vista, punto visual, posición visual o posición de vista.

Protocolo.
Serie de reglas que controlan la transmisión y recepción de datos en red.

Radiosidad.
Técnica de iluminación que distribuye cantidades equivalentes de cada fuente de luz en la escena. De hecho, los objetos en la escena también se consideran como fuentes luminosas porque reflejan la luz que reciben.

Red.
Agrupación tanto de equipos como de programas que comparten recursos entre sí, observando "reglas de comportamiento" a partir del uso de un lenguaje y medios de transmisión comunes, sin importar - en lo esencial - la naturaleza de cada elemento dentro de la red.

Reflexión.
Aproximación matemática de la cantidad de luz que después de llegar a un objeto se proyecta en otras direcciones, lo cual, permite observarla. En VRML, existen tres tipos de reflejo: ambiental, difuso y especular.

Render.
Proceso en el que la computadora interpreta todos los datos de luces y objetos para crear una imagen final en el visor seleccionado. La imagen resultante puede ser fija o un cuadro de una animación.

RGB.
Gama de colores compuesta del rojo, verde y azul usados en VRML y otras aplicaciones gráficas.

Rotación.
Movimiento lateral de un objeto. En las gráficas por computadora, la rotación es el ángulo de giro sobre los ejes longitudinales.

Script.
Pequeño programa creado con un lenguaje de alto nivel para dar instrucciones a otro programa y realizar así tareas específicas.

Servidor.
Computadora dedicada a gestionar el uso de la red por otras computadoras llamadas clientes. Contiene archivos y recursos que pueden ser accesados desde otras computadoras (terminales).

Sistema de coordenadas.
Sistema de ejes y unidades que define la posición de los objetos. En VRML, los objetos se definen en un espacio local que incluye a la cámara.

Sombreado.
Parte del acabado que controla la conversión de los datos en colores que pueden ser desplegados. Los sombreados más comunes son: Opaco, Garaud y Phong.

Superficie.
Cualquier figura que define un área, pero que no tiene volumen. Las superficies pueden ser facetas (conjunto de caras) o paramétricas, es decir, las que se definen mediante curvas.

Transparencia.
Aproximación matemática del efecto que resulta al hacer una emisión luminosa que cruza una superficie no opaca (traslúcida). En VRML, este valor se controla por medio del campo Transparency del nodo Material.

UNIX.
Sistema operativo especializado en capacidades de multiusuario y multitarea. Fue la base inicial de Internet. Entre sus caracteristicas más importantes se encuentran: Redireccionamiento de Entrada/Salida de alta portabilidad al estar escrito en lenguaje C, lo que lo hace independiente del hardware.

URL (Uniform Resorce Locator) Localizador Uniforme de recursos.
Sistema de direccionamiento estándar para archivos y funciones de Internet, especialmente en el Word Wide Web. El url está conformado por el servicio (p.ej; http://), más el nombre de la computadora (p. ej; www.uvimar.cl), más el directorio y el archivo referido.

Usuario.
Un usuario es la persona que tiene una cuenta en una determinada computadora por medio de la cual puede acceder a los recursos y servicios que ofrece una red. Un usuario que reside en una determinada computadora, tiene una dirección electrónica única.

Vértice.
Punto en la "esquina" de una cara o conjunto de líneas. Los vértices conectan los bordes de las figuras.

Vínculo (link).
Es un indicador de texto o una imagen que sirve como enlace a otro documento

Visualizador (Browser).
Programa que despliega la información almacenada en páginas HTML que se encuentran disponibles en servidores del World Wide Web. Como ejemplo de visualizadores tenemos Cello, Internet Explorer, Mosaic, Netscape, Opera, entre otros.

VRML (Virtual Reality Modeling Language antes Virtual Reality Markup Language).
Lenguaje de programación utilizado para hacer presentaciones de realidad virtual en el Word Wide Web. Puede ser un visualizador propio o integrado a los visualizadores WWW a través de un Plugin.

Web (WWW).
Sistema basado en hipertextos cuya función es buscar y tener acceso a documentos a través de la red.

Literatura citada.

VIRTUAL WORLDS ON THE INTERNET
John Vince
Rae Earnsshaw
IEEE Computer Society
1998.

SPECTRUM
Magazine
IEEE Computer Society
2002.

THE VISUALIZATION TOOLKIT 2nd Edition
WillSchroeder
Ken Martin
Bill Lorensen
Prentice Hall PTR.
1998.

VIRTUAL AUTOMATION ENVIRONMENTS
Herwing Mayr
Marcel Dekker Inc.
2002.

NETWORKED VIRTUAL ENVIROMENTS
Design and implementation
Sandeep Singhal,Michael Zyda.
Ed. Addison-Wesley
ACM 1999.

VRML PARA INTERNET
Mark Pesce
Prentice Hall
1996.

VRML
Biblioteca del Programador
Kris Jamsa
Phil Schamauder
Nelson Yee
Ed. McGraw Hill.
1998.

Guía de Internet.

[1] - http://www.apple.com (Página oficial de Macintosh).

[2] - http://developer.blaxxun.com/download/index.html (Descarga de visualizador Blaxxun).

[3] - http://www.parallelgraphics.com (Descarga de visualizador Cortona).

[4] - http://www.xemacs.org (Descarga del editor Xemacs).

[5] - http://www.web3d.org/www-vrml/ (Ligas y elementos de VRML).

[6] - http://www.wed3d.org (Página oficial de Web 3D Consortium).

[7] - http://www.sony.com (Página oficial de la empresa Sony).

[8] - http://www.acm.org/crossroads/espanol/xrds3-3/vrhci.html (Interfaces actuales de RV).

[9] - http://www.activamente.com.mx/vrml (¿Qué es la RV?).

[10] - http://www.computo2000.unam.mx/rv_hernando.htm (Articulo de RV).

[11] - http://www.contactomagazine.com/realidadvirtual0417.htm (Realidad Virtual de los Programas Militares a Videojuegos e Internet).

[12] - http://www.etsimo.uniovi.es/links/vr.html (Realidad Virtual: Libros, Tutoriales, Congresos).

[13] - http://exodus.dcaa.unam.mx.

[14] - http://www.fisc.utp.ac.pa/unidades/uim/rv.html (Elementos de RV).

[15] - http://icarito.tercera.cl/profes/informatica/doc/Metodologia.doc (Evolución del VRML).

[16] - http://www.realidadvirtual.freeservers.com/cgi-bin/framed/4156.

[17] - http://www.rvltda.com/FramesEsp.htm (FAQ de VRML).

[18] - http://www.telematica.cicese.mx/computo/super/realvirtual.

[19] - http://www.ucm.es/info/Psyap/Prieto/alum9798/virtual/rv/rv.htm (Trabajo sobre RV).

[20] - http://www.um.es/undis/jornadas/p3espanol.html (Elementos básicos de la Realidad virtual).

[21] - http://www.unitec.edu.co/biblioteca/rv/principal.html (Trabajo sobre RV).

[22] - http://www.web3d.org/vrml/vrml.htm (Repositorio Web 3D).

[23] - http://www.frag.cl/gametech/Glosario/t.html (Glosario de Internet).

[24] - http://alleg.sourceforge.net/docs/lighting_effects.es.html (¿El camino hacia la iluminación?).

[25] - http://www.web3d.org/vrml/vrml.htm.

[26] - http://www.amazon.com (Venta de libros, audio y accesorios por Internet).

[27] - http://www.nike.com (Página oficial de la empresa Nike).

[28] - http://www.vw.com.mx (Página oficial de la empresa VolksWagen).

[29] - http://www.vicon.com (Página oficial de Vicon).

[30] - http://www.vtk.org/files/release/4.0 /.

[31] - http://www.ladei.com.ar/D3D/Intro_TecnSombreado.html (Técnicas de sombreado).

[32] - http://www.ch.ic.ac.uk/rzepa/vrml/ (portal de aplicaciones de química usando VRML).

[33] - http://sim.di.uminho.pt/vrmltut/ (Buen tutorial de VRML).

[34] - http://www.es.embnet.org/Doc/Training/VRML_training/index.es.html (Aplicaciones de VRML en biología).

[35] - http://www.galiciacad.com/vrml/barcelona.php3 (Sitio de arquitectura española en VRML).

[36] - http://www.virtuallybetter.com (Compañía dedicada a crear simulaciones virtuales de fobias)

[37] - http://www-vri.umich.edu (Proyectos virtuales en la Universidad de Michigan)

[38] - http://www.fortunecity.es/virtual/juegos/567/jvrml/main2.html

Epílogo.

Consideraciones de diseño.

A continuación describiré algunas aportaciones sobre VRML, sobre como se diseña en este lenguaje, como compartir eventos y objetos además de la interfaz entre HTML y VRML.

Se puede considerar que los ambientes 3D fueron creados desde sus inicios para mejorar la comunicación entre la gente. Ya que gracias al 3D puede encontrarse e interactuar en lugares virtuales, también pueden simular objetos reales. En estas representaciones virtuales, la gente puede mostrar, enseñar o aprender de una manera efectiva.

El crear un mundo virtual 3D le permite a un usuario ver a otras personas en ese mismo mundo ya que todos accesan como avatares[1] situándose en cierta posición y orientación. Cada uno de los avatares puede moverse y observar los objetos que se encuentran en el mundo 3D. Sin embargo el construir o diseñar ambientes 3D como un lugar de reunión requiere de varios cuidados. Por ejemplo, una puerta debe permitir que dos avatares quepan al mismo tiempo al pasar por ella, esto como medida estándar, deben crearse Viewpoints para evitar que los usuarios que no puedan navegar cómodamente lo hagan de forma más simple, entre otros. En un espacio 3D pueden existir diferentes tipos de eventos como abrir puertas, encender y apagar luces, abrir el CD de una computadora, etc.

Reglas de diseño

Para realizar aplicaciones VRML exitosas se necesita de una combinación de rendimiento y calidad. Aún cuando hoy en días las capacidades técnicas de las computadoras han aumentado, los gráficos 3D en tiempo real todavía no son lo que se espera. La calidad de despliegue de objetos o mundos VRML en una PC depende de la velocidad del CPU y la tarjeta gráfica o de video.

Una vez analizado lo anterior se tiene un elemento que funge como filtro en las aplicaciones de tiempo real, este es el tipo de conexión a Internet. La meta es cargar los mundos en un mínimo de tiempo, para que un usuario mantenga un balance entre velocidad y calidad de escena. Para optimizar el funcionamiento, un diseñador de un mundo 3D puede seguir los siguientes consejos:

- Crear modelos con la menor cantidad de polígonos posibles.
- Usar texturas para simular detalles de geometría.
- Reemplazar la iluminación con las texturas apropiadas.
- Usar objetos auxiliares de VRML en lugar de geometrías convencionales.
- Optimizar los archivos VRML con ayuda de funciones proporcionadas por el mismo VRML.
- Comprimir los archivos VRML (gzip)

Acerca de las dimensiones

La unidad de distancia en VRML es el metro. Por tanto cuando por ejemplo se crean avatares, estos deben tener una dimensión aproximada de 1.75 metros de alto. Se debe ser relativamente generoso con los espacios establecidos a la hora de crear los mundos, especialmente para puertas y el ancho de los lugares por donde se navega o pasan los avatares. Debemos asegurarnos de que tampoco encontraran obstáculos los visitantes, recordemos que la mayoría de ellos no tienen experiencia navegando en mudos 3D.

---

[1] Avatar. Representación virtual de un humano.

En cuanto a los archivos VRML siempre procuremos usar letras minúsculas para nombrarlos, incluyendo a las carpetas y las texturas relacionadas. Algunos sistemas operativos como UNIX y LINUX son sensibles al tamaño de las letras y si estos parámetros no se respetan pueden arrojar errores.

La escena gráfica VRML

Recordemos que VRML es un estándar internacional para gráficos interactivos 3D en Internet. Los detalles sobre sus especificaciones se pueden encontrar en el sitio http://www.vrml.org/fs_specifications.htm. Para la mejor manipulación de las escenas se debe programar el código VRML. Esto se debe hacer en un editor de texto, pero existe una buena alternativa que recomiendo para lograr de una manera más amena este cometido: Vrmlpad.

El VrmlPad se puede descargar de http://www.parallelgraphics.com/vrmlpad, consiste en un programa editor capaz de reconocer la sintaxis de VRML coloreando el código y reconociendo errores. Reconoce nodos, los organiza jerárquicamente, agrupa y describe la funcionalidad de cada nodo. En la siguiente tabla se describen algunos componentes de una escena VRML.

#VRML V2.0 utf8	Esta línea debe aparecer al comienzo de cada archivo VRML para identifcarlo como tal.
DEF	Esta instrucción es usada para asignar un nombre a un nodo: este nodo puede ser usado más adelante con la instrucción USE optimizando el tamaño del archivo.
Transform	Este nodo toma un objeto o bloque de objetos y los mueve a cierta posición en el espacio 3D.
Translation	Especifica el desplazamiento en X, Y y Z. En el sistema coordenado VRML, X y Z forman el plano horizontal mientras Y es vertical.
Rotation	Define una rotación con 4 valores numéricos: una rotación sobre alguno de los ejes X Y Z y un ángulo W dado en radianes.
Scale	Define un factor de escala para los ejes X Y Z.
Shape	Este nodo describe un objeto. Contiene los campos appearance y geometry.
Material	Este nodo proporciona detalles acerca de la apariencia de los objetos.
diffuseColor	
ATRIBUTOS ADICIONALES	
ambientIntensity	Nivel de brillo (0 – 1); donde 1 es el valor mas alto.
shininess	Especifica el nivel de contraste (0 – 1)
specularColor	Indica el color espectral.
transparency	Especifica la transparencia de un objeto (0 - 1), donde 0 = opaco y 1 = trasnparente.
emissiveColor	Especifica la emisión de color.
Box	Este nodo especifica el nodo promitivo "caja". Otros nodos de este tipo son: Sphere, Cone y Cylinder.
IndexedFaceSet	Este nodo (conjunto de polígonos) es usado para describir objetos que no pueden describirse por nodos primitivos.

Consideraciones al crear geometrías.

Se debe poner atención en el número de polígonos que se usan al dibujar una geometría. Es más complicado crear una escena atractiva cuando esta contiene pocos polígonos que cuando contiene más. De esta manera se entiende que será más fácil agregar complejidad y detalles a una escena que quitársela.

Las curvas pueden ser creadas con ayuda de pocos polígonos usando el campo de VRML denominado *creaseAngle 3.14* en el nodo *geometry* y de esta manera suavizar los bordes.El código debe optimizarse borrando polígonos que nunca serán vistos por los usuarios, pero teniendo cuidado de eliminarlos con orden para evitar dejar intervalos. En lugar de geometrías complejas, se pueden usar texturas o billboards.

El campo *solid* del nodo *geometry* especifica si un objeto tiene o no dos lados. Este parámetro declarado con un valor lógico *TRUE* indicará que el objeto es cerrado y solo será desplegado un lado. Por ejemplo, un objeto parecido a un disco plano podría verse invisible al verse por debajo. Para hacerlo visible por ambos lados se deberá declarar el valor lógico *FALSE* para forzar al trazado de la geometría en tiempo real, a calcular el polígono dos veces.

Consideraciones al usar texturas.

VRML soporta los siguientes formatos de imagen para texturas:

- JPG (Joint Photographic Experts Group)
- PNG (Portable Network Graphics)
- GIF (Graphics Interchange Format)

Es requerido un canal Alfa para usar texturas con regiones transparentes. Los formatos PNG y GIF proveen esta funcionalidad. Mantener texturas pequeñas (tamaño de imagen y número de colores) para reducir el tiempo de descarga para el usuario. La resolución de las imágenes debe ser 72 dpi para ser óptima y acorde a la de las páginas Web.

Se debe hacer un uso eficiente de la memoria de textura en las tarjetas gráficas, para ello las texturas deben dimensionarse en potencias de dos; por ejemplo, 32 X 32, 64 X 64, 64 X 128, etc. Si es posible, estas texturas deben ser cuadradas. El tamaño máximo ideal para una textura es de 512 X 512 pixeles. Es ideal usar una larga textura que contenga las imágenes de varios objetos. Esto es más eficiente que usar varios pequeños archivos de texturas invocados remotamente de un servidor de Internet.

Consideraciones sobre los puntos de vista (Viewpoints).

Con la herramienta "Viewpoint" se puede determinar directamente la posición de los usuarios en una escena. La mayoría de las herramientas 3D trabajan con cámaras, las cuales son convertidas a "Viewpoints" al exportarse a VRML. Los "Viewpoints" pueden ser animados para proporcionar movimientos automáticos como en un tour o recorrido virtual.

Los "Viewpoints" deben posicionarse a una altura de 1.75 metros sobre el nivel del suelo de una escena; esta es la vista normal de un avatar. Se debe usar el campo "descripción" para especificar el nombre de un "Viewpoint" ya que este nombre será desplegado en el menú del visualizador VRML usado (siempre y cuando contenga esa funcionalidad).

Consideraciones sobre Luces.

El uso de luces incrementa el tiempo de rendering. Se debe tener cuidado en seleccionar los lugares adecuados donde sean necesarias las luces. Los visualizadores VRML tienen una iluminación por defecto, la cual, permite mostrar las propiedades del objeto. Los objetos con texturas no son afectados por las luces, a menos que tengan propiedades de material adicionales.

Consideraciones sobre Animación.

Al crear animaciones se deben usar pocos frames y cambiar los intervalos de tiempo de la animación. Si algunos objetos poseen la misma animación, se deben combinar en un grupo para que los interpoladores optimicen su trabajo.

Optimización.

Algunos de los siguientes pasos para la optimización de VRML pueden ser ajustados mientras se crea el modelo:

- Crear un modelo con pocos polígonos.
- Simular detalles de geometría con texturas.
- Sustituir texturas fotorrealistas por iluminación.
- Reducir y simplificar las geometrías.
- Eliminar caras de los objetos innecesarias.
- Usar constantemente los nodos DEF y USE.
- No usar "solid FALSE" cuando no es necesario.
- Compartir nodos idénticos como texture, material y apperance.
- Intentar combinar varias texturas en una imagen adaptando las coordenadas de las texturas.
- Escalar las texturas en potencias de dos.
- Limitar el número de luces. Las luces direccionales son más rápidas.
- Ayudar al navegador limitando la cantidad de nodos activos:

a) Usando nodos "Switch"
b) Las partes invisibles pueden ser habilitadas o deshabilitadas.
c) Las animaciones complejas pueden deshabilitarse anulando los sensores de tiempo (TimeSensor).
d) En ocasiones se pueden sustituir los nodos de texto con texturas que representan ese texto.
e) Algunos plug-in permiten animaciones del tipo de las imágenes GIF.
f) El sombreado plano puede forzarse con el campo "creaseAngle" en cero y el normal "PerVertex" con el valor lógico FALSE.
g) Las formas con más de 2020 vértices o colores por cara requieren un poco de procesamiento extra.

Aspectos Multiusuarios.

Crear mundos 3D para ambientes multiusuarios es muy diferente a crear mundos u objetos para un solo usuario. Debemos asegurarnos de que nuestro concepto sea cuidadoso en los siguientes aspectos:

- Usar en dimensiones correctas.
- El mundo 3D debe ser adecuado para albergar varios usuarios al mismo tiempo. Dimensiones pequeñas pueden resultar incomodas.
- Se debe pensar en las interacciones que tendrán los multiusuarios.
- Asegurarnos de que la interfaz es entendible.
- Considerar un apropiado tiempo de descarga.