UNIVERSIDAD POLITÉCNICA DE PUEBLA

PROGRAMA ACADÉMICO DE
INGENIERÍA EN INFORMÁTICA

Desarrollo de un ambiente 3D para simular problemas de robótica

Alejandro Mugarte Falcón

Reporte Técnico PII-02-04-09

COMITÉ EVALUADOR

Dr. Antonio Benítez Ruiz (*Asesor*)
Dr. Jorge de la Calleja Mora (*Sinodal*)
Dra. María Auxilio Medina Nieto (*Sinodal*)

PROFESOR(A) DE PROYECTO DE INVESTIGACIÓN II

Dra. María Auxilio Medina Nieto

Juan C. Bonilla, Puebla
Abril 2009

Resumen

Los ambientes virtuales son simulaciones tridimensionales generadas por computadora las cuales representan algún aspecto del mundo real permitiendo al usuario interactuar con los objetos dentro del espacio virtual del ambiente.

Este trabajo describe el desarrollo de un sistema para la representación de ambientes virtuales llamado Graphic Environment for Motion Planning Algorithms (GEMPA), las funcionalidades de esta herramienta comprenden la manipulación del ambiente por medio del ratón y teclado, la recuperación de información desde diferentes tipos de archivos como son txt y off, iluminación bajo el modelo de Lambert, manejo de texturas, y manipulación, almacenamiento y recuperación de ambientes formados por figuras predeterminadas del sistema.

Capítulo 1. Planteamiento del problema de investigación.

1.1. Introducción

La robótica puede ser definida como la ciencia o estudio de la tecnología principalmente asociada con el diseño, fabricación, teoría y aplicación de los robots [Dowling 2003]. Mientras otros campos como la matemática contribuyen al cálculo, la robótica crea el producto final.

La planificación de movimientos es una rama de la robótica que estudia el control y decisión que el robot debe hacer para realizar un movimiento en tareas del mundo real, por ejemplo, mover cosas o cruzar zonas cambiantes. Esta rama hace posible una de las metas de la robótica, que es hacer robots autónomos, cada uno con la capacidad de trabajar sin ayuda de un ser humano [Latombe 1991].

Hacer pruebas de movimientos con robots reales conlleva gastos muy elevados, por esta razón, la simulación por computadora es una alternativa económica de gran ayuda. La simulación consiste en generar un ambiente virtual el cual representa situaciones de la vida real; los objetos que están dentro del ambiente se comportan e interactúan como en el mundo real [Aldrich 1996].

Existen dos diferentes tipos de ambientes virtuales, el primero es llamado *ambiente virtual inmersivo*, éste es un espacio 3D con el cual el usuario puede interactuar, produciendo la sensación de estar dentro del ambiente. El segundo es llamado *ambiente virtual no inmersivo*, este tipo de ambiente tiene un alto nivel de interactividad y el espacio es generado a escala real 1 a 1[Ramos 2003].

Los ambientes inmersivos pueden ser descritos por archivos con extensión *off* (object file format), éstos representan colecciones de polígonos planos con vértices compartidos para describir poliedros [Bourke 2006].

Es importante mencionar que a pesar de que ya existen herramientas gráficas que permiten representar el comportamiento de algoritmos de planificación de movimientos [CAMELOT 2006] [La Valle 2003], éstas no son los suficientemente flexibles para incorporar funcionalidades tanto de navegación como de pintado. Por esa razón es que se decidió iniciar el desarrollo de una herramienta gráfica en 3D, la cual se describe a continuación.

1.1.1. Antecedentes

El ambiente Gráfico para Algoritmos de Planificación de Movimientos (Graphic Environment for Motion Planning Algorithm, GEMPA) es un ambiente virtual inmersivo. El desarrollo de GEMPA inicia en primavera del 2002, su principal objetivo fue contar con un simulador en 3 dimensiones capaz de reproducir el comportamiento de algoritmos de planificación de movimientos.

Los algoritmos de planificación de movimientos se encargan de buscar una ruta libre de colisión entre dos configuraciones, una configuración inicial y una configuración final (una configuración representa tanto la posición de un objeto en 3 dimensiones como su orientación). La ruta encontrada por el algoritmo de planificación de movimientos entre una configuración inicial y una configuración final se almacena en un archivo de texto (archivo de configuraciones), el cual está formado por un conjunto de configuraciones, mismas que se encuentran representadas a través de tuplas que contienen 6 elementos, los primeros 3 especifican la posición del objeto y los últimos su orientación.

GEMPA permitió evaluar de manera eficiente los algoritmos de planificación de movimientos desarrollados por un grupo de robótica en un curso de maestría en la Universidad de las Américas Puebla en otoño del 2003. Posteriormente, fue utilizada por un tesista de maestría [De los Santos 2004] y en un trabajo de doctorado [Benitez 2005] para simular el comportamiento de algoritmos de planificación.

Aunque la herramienta cumplió con su objetivo inicial (simular el comportamiento de los algoritmos de planificación de movimientos), tiene algunas deficiencias tanto en los controles de navegación como en los efectos de iluminación y sombreado para los objetos representados en el ambiente. Actualizar GEMPA añadiendo algunas funcionalidades de navegación e iluminación permitirá lograr un mayor realismo del ambiente virtual.

Actualmente, GEMPA está siendo utilizada para cursos de Temas Selectos de Informática en la Universidad Politécnica de Puebla, en donde se desarrollan técnicas de simulación, tanto para objetos rígidos como para cuerpos articulados. Esto ofrece la oportunidad de mejorar sustancialmente las funcionalidades de GEMPA para poder trabajar también con cuerpos articulados, proponiendo así el objetivo general descrito en la siguiente sección.

1.2. Objetivo General

Rediseñar e incorporar texturas, iluminación, transformaciones geométricas y nuevas interfaces de usuario al ambiente de simulación GEMPA a través del uso de herramientas de programación gráfica y de la migración del código a una plataforma de software libre, para dotar al ambiente virtual de mayor realismo y portabilidad.

1.3. Objetivos Específicos

- Proponer el diseño e implementación de una nueva interfaz de usuario para GEMPA, la cual incorpore el manejo de luces, sombras, efectos de transparencia y texturas para lograr un mayor realismo, bajo la plataforma Dev C++.
- Proponer e implementar una interfaz de usuario para el control de la navegación dentro del ambiente virtual (operaciones de inmersión en el ambiente como son traslaciones y rotaciones) a través del uso del teclado, ratón y utilizando programación visual.
- Incorporar a la interfaz la capacidad de poder definir, manipular y almacenar figuras geométricas tridimensionales soportadas por las primitivas de OpenGL.
- Colaborar con la migración de las funcionalidades existentes en GEMPA hacia la plataforma Dev C++, así como incorporar al ambiente la capacidad de leer archivos de tipo off.

1.4. Justificación

Muchas universidades no cuentan con robots reales y recurren al uso de software para representar y hacer un estudio de conceptos de robótica. Esto les permite comprobar la teoría matemática con los resultados arrojados por pruebas de simulación, en donde áreas como inteligencia artificial y simulación por computadora apoyan la formación académica de los estudiantes. Buscando la construcción de una herramienta que pueda servir de material didáctico para introducir algunos conceptos de robótica, es que se hace necesario

el desarrollo de un ambiente virtual capaz de representar la interacción y movimientos que objetos como cubos, esferas, entre otros, dentro de un espacio virtual.

1.5. Metodología

A continuación se muestra el procedimiento para la realización del sistema propuesto.

1.5.1. Cronograma

Tabla 1. Actividades para Proyecto de Investigación 1

		SEP				OCT				NOV				DIC			
ACTIVIDADES	**PRODUCTO**	S 1	S 2	S 3	S 4	S 1	S 2	S 3	S 4	S 1	S 2	S 3	S 4	S 1	S 2	S 3	S 4
Analizar librerías de Open GL	----------																
Presentar propuesta de protocolo	Documento																
Rotación por teclado	Rot. teclas flecha izq. y der.																
Traslación por teclado	Tras. teclas flecha arriba y abajo																
Analizar métodos de iluminación	----------																
Implementar iluminación básica	Función para cálculo vector normal de triángulos iluminación Lambert																
Revisión de literatura	----------																
Entrega de Protocolo	Documento																
Implementar menú en la interfaz	Implementar barra de menú en parte superior de interfaz																
Implementar efecto traslucido	Figuras traslucidas																
Implementar figuras predeterminadas	Figuras predeterminadas disponibles en el menú																
Presentar protocolo ante comité evaluador	Documento																
Elaboración del reporte de investigación	Protocolo de investigación																

Tabla 2. Actividades para Proyecto de Investigación 2

ACTIVIDADES	PRODUCTO	ENE				FEB				MAR				ABR			
		S 1	S 2	S 3	S 4	S 1	S 2	S 3	S 4	S 1	S 2	S 3	S 4	S 1	S 2	S 3	S 4
Implementar rotación por controles gráficos	Barras para control de rotación																
Implementar traslación por controles gráficos	Barras para control de traslación																
Analizar métodos de mapeo de texturas	----------																
Implemenar texturas	Pegado de imágenes mapa de bits sobre figuras																
Migrar lectura de mallas con extensión .txt	Dibujado de malla																
Migrar lectura de configuraciones con extensión .txt	Dibujado de configuraciones																
Migrar lectura de ambientes con extensión .txt	Dibujado de ambiente																
Implementar lectura de mallas con extensión .off	Dibujado de malla																
Implementar lectura de ambientes con extensión .off	Dibujado de ambiente																

1.6. Recurso de Hardware y Software

- Software: Dev C++ 4.9, Librerías Glut 3.7 de OpenGL, Sistema Operativo Windows Vista
- Hardware: procesador AMD Semprom 3800+ a 2.2GHz, 2GB RAM DDR2, GeForce 6150SE nForce 430 128MB VRAM.

1.7. Alcances y Limitaciones

- El sistema podrá ser soportado por diferentes sistemas operativos como Linux y Windows.
- La interfaz del sistema podrá cambiar de resolución.
- El sistema incluirá figuras predeterminadas soportadas por las primitivas de OpenGL como tetera, esfera, cubo, entre otras.

- El sistema incorpora luces, texturas y transparencias para dar mayor realismo a los ambientes virtuales que se representen.
- El sistema queda abierto para posteriormente soportar el manejo de objetos articulados.
- El sistema incluirá una interfaz de usuario que controle la navegación dentro del ambiente a través del teclado y ratón.
- El sistema no contempla la lectura de datos desde archivos con formato XML.

Capítulo 2. Marco Teórico

2.1. Ambientes Virtuales

Se nombra *ambiente 3D* a una simulación tridimensional generada por computadora de algún aspecto del mundo real, en el cual el usuario puede interactuar con él. Por ejemplo, un usuario puede realizar acciones dentro de un modelo virtual como son desplazar y mover objetos para experimentar situaciones que se asemejan al mundo real [Suárez 2003].

2.1.1. Tipos de Ambientes Virtuales

Existen diferentes tipos de ambientes virtuales, el primero de éstos es llamado *ambiente inmersivo*. Los ambientes inmersivos son aquellos sistemas donde el usuario se siente dentro del mundo virtual que está explorando. Este tipo de ambientes utiliza diferentes dispositivos o accesorios como pueden ser guantes, trajes especiales, visores o cascos; éstos últimos le permiten al usuario visualizar los mundos a través de ellos. Los cascos son el principal elemento que hacen al usuario sentirse inmerso dentro de los mundos. Este tipo de sistemas son usados para aplicaciones de entrenamiento o capacitación.

El segundo tipo de ambiente es llamado *ambiente no inmersivo*. Los ambientes no inmersivos o de escritorio, son aquellos donde el monitor es la ventana hacia el mundo virtual y la interacción es por medio del teclado, micrófono, ratón o joystick. Se emplean para visualizaciones científicas, también son usados como medio de entretenimiento y aunque no ofrecen una total inmersión, son una alternativa de bajo costo [Ramos 2003].

2.1.2. Herramientas de Graficación

OpenGL es un software de interfaz para graficar objetos; está formado de 700 comandos distintos que se usan para especificar los objetos y operaciones necesarias para producir aplicaciones de tres dimensiones [Shreiner 2008].

La estructura básica de un código consta de una inicialización de estados que controlan cómo OpenGL dibujará a los objetos y los objetos que serán renderizados (al proceso por el cual una computadora genera imágenes en base a modelos, se le nombrará *renderización*).

Los modelos u objetos están construidos a base de primitivas geométricas (puntos, líneas y polígonos) que son especificadas por sus vértices. Las imágenes renderizadas consisten de pixeles (elemento visible más pequeño de la pantalla) iluminados en la pantalla; cada pixel que forma la imagen tiene un color uniforme [Buss 2003].

2.2. Graficación

La graficación es la representación de figuras en un plano. El plano consta de ejes (*X Y Z*); dentro de estos ejes se localizan coordenadas que permiten el trazado de imágenes (gráficos) en la superficie del plano.

Para el modelado de gráficos en 3D [Cruz 1993] define tres pasos que se deben llevar a cabo:

- **Modelado en 3D.-** Durante este paso se lleva a cabo la representación de los objetos que contendrá la escena con coordenadas tridimensionales, es decir, puntos que componen el objeto con coordenadas sobre los ejes *X Y Z.*
- **Proyección en 2D.-** La pantalla de una computadora es bidimensional, para lograr una representación de objetos 3D sobre la pantalla, es necesario hacer una proyección de los objetos tridimensionales sobre el plano bidimensional.
- **Representación y animación.-** Durante esta etapa se coloca la iluminación de los objetos, el manejo de texturas y la animación de objetos.

2.2.1. Transformaciones Geométricas

Una línea en la computadora se representa por dos puntos (x_1, y_1, z_1) y (x_2, y_2, z_2) en un espacio 3D. Aunque la pantalla de una computadora es plana y en 2D (eje *X*, eje *Y*), la representación de la profundidad (eje *Z*) se aprecia al hacer más pequeño el objeto a medida que se aleja del punto de vista del observador (Figura 1).

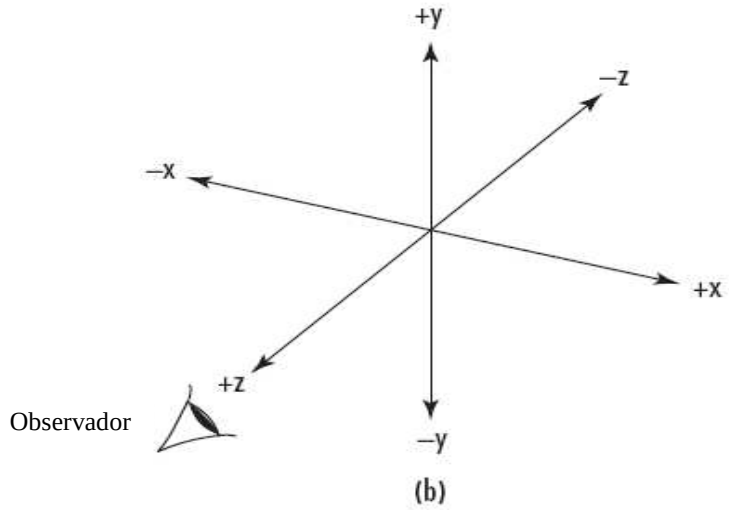


Figura 1. Punto de vista del observador

Una transformación geométrica es la operación u operaciones matemáticas que permiten deducir una nueva figura de la original, a la figura obtenida se le nombra *homólogo del original* [Wright 2003]. Ejemplos de transformaciones geométricas son la rotación y traslación, las cuales son descritas a continuación.

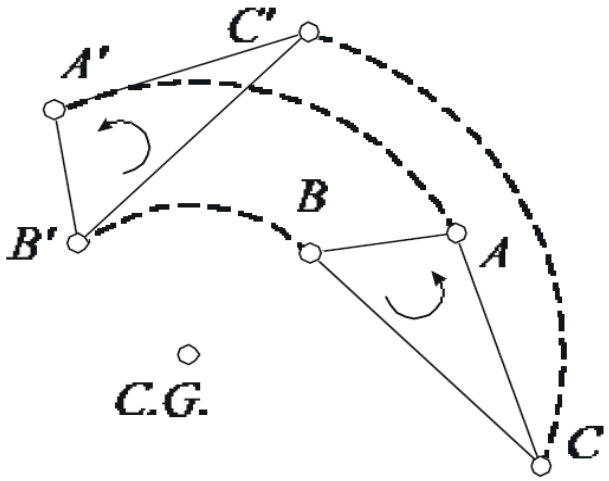


Figura 2. Rotación o giro

a) <u>Rotación</u>: Como se observa en la Figura 2, para obtener un homólogo de un punto en la rotación, se traza por el punto un arco con centro en el Centro de Giro (*C.G.*) que abarque

el ángulo de giro indicado. Una rotación se considera positiva si está en sentido contrario a las manecillas del reloj. La rotación puede ser aplicada en los ejes *X Y Z*. En la Figura 3, considere que el punto P(x_1, y_1, z_1) se rota al punto Q(x_2, y_2, z_2).

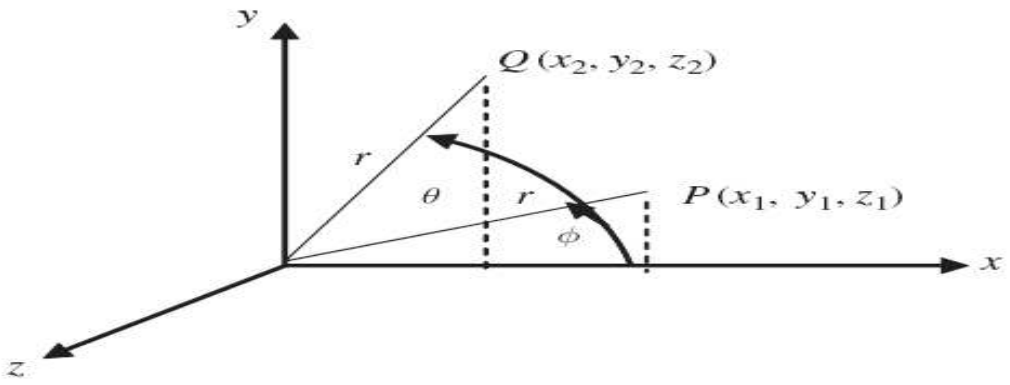


Figura 3. Rotación en un espacio

[Whitrow 2008] propone las definiciones siguientes:

$$x_1 = r\ \cos\phi \qquad y_1 = r\ \sin\phi$$
$$x_2 = r\ \cos(\theta + \phi) = r\ \cos\theta\ \cos\phi - r\ \sin\phi\ \sin\theta$$
$$y_2 = r\ \sin(\theta + \phi) = r\ \sin\theta\ \cos\phi + r\ \cos\theta\ \sin\phi$$

al eliminar las referencias a ϕ se obtiene:

$$x_2 = x_1\ \cos\theta - y_1\ \sin\theta$$
$$y_2 = x_1\ \sin\theta + y_1\ \cos\theta$$

su notación matricial es:

$$\begin{pmatrix} x_2 \\ y_2 \end{pmatrix} = \begin{pmatrix} \cos\theta - \sin\theta \\ \sin\theta + \cos\theta \end{pmatrix} \begin{pmatrix} x_1 \\ y_1 \end{pmatrix}$$

En la Figura 3 se observa que θ incrementa la rotación en sentido contrario a las manecillas del reloj a partir del origen. Para el caso general de la rotación en el eje z, en cualquier plano paralelo al plano (*x, y*) la matriz se denota:

$$\begin{pmatrix} x_2 \\ y_2 \\ z_2 \\ 1 \end{pmatrix} = \begin{pmatrix} \cos\theta & -\sin\theta & 0 & 0 \\ \sin\theta & \cos\theta & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 1 \end{pmatrix} \begin{pmatrix} x_1 \\ y_1 \\ z_1 \\ 1 \end{pmatrix}$$

En el momento que el ángulo de rotación aumenta en el eje *X* (en el plano *Y*, *Z*) y en el eje *y* (en el plano *X*, *Z*), las coordenadas *y* en el plano *X* se convierten en coordenadas *z* del plano (*Y*, *Z*), y las coordenadas *x* se convierten en coordenadas *y*, obteniendo lo siguiente:

$$\begin{aligned} x_2 &= x_1 \cos\theta - y_1 \sin\theta \\ y_2 &= x_1 \sin\theta + y_1 \cos\theta \end{aligned}$$ en el plano (*x*, *y*).

$$\begin{aligned} y_2 &= y_1 \cos\theta - z_1 \sin\theta \\ z_2 &= y_1 \sin\theta + z_1 \cos\theta \end{aligned}$$ en el plano (*y*, *z*).

La representación matricial de la rotación en el eje *x* que está en el plano (*y*, *z*) es:

$$\begin{pmatrix} x_2 \\ y_2 \\ z_2 \\ 1 \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos\theta & -\sin\theta & 0 \\ 0 & \sin\theta & \cos\theta & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x_1 \\ y_1 \\ z_1 \\ 1 \end{pmatrix}$$

La matriz correspondiente a la rotación en el eje *y* en el plano (*z*, *x*) es:

$$\begin{pmatrix} x_2 \\ y_2 \\ z_2 \\ 1 \end{pmatrix} = \begin{pmatrix} \cos\theta & 0 & \sin\theta & 0 \\ 0 & 1 & 0 & 0 \\ -\sin\theta & 0 & \cos\theta & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x_1 \\ y_1 \\ z_1 \\ 1 \end{pmatrix}$$

Las rotaciones que se han mostrado están colocadas en el origen del plano.

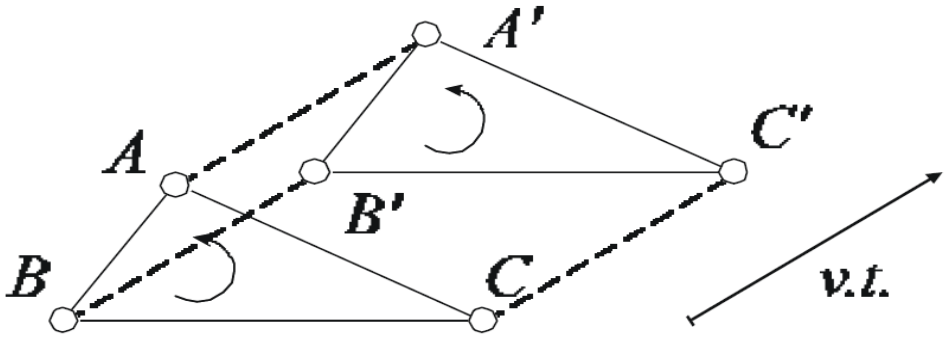


Figura 4. Traslación

b) Traslación: Para obtener un homólogo de un punto en la traslación, se sitúa el extremo de un segmento equivalente al vector de traslación (*v.t.*) sobre el punto, y el homólogo (puntos *A', B', C'*) quedará situado en el otro extremo, el vector de traslación define la dirección en que se va a hacer la traslación (Figura 4). La traslación puede ser aplicada en los ejes *X Y Z* [Hearn 1994].

Una traslación es el proceso de mover un punto de una posición a otra dentro de un espacio (3D en particular) [Whitrow 2008]. Considérese dos puntos P y Q en la Figura 5.

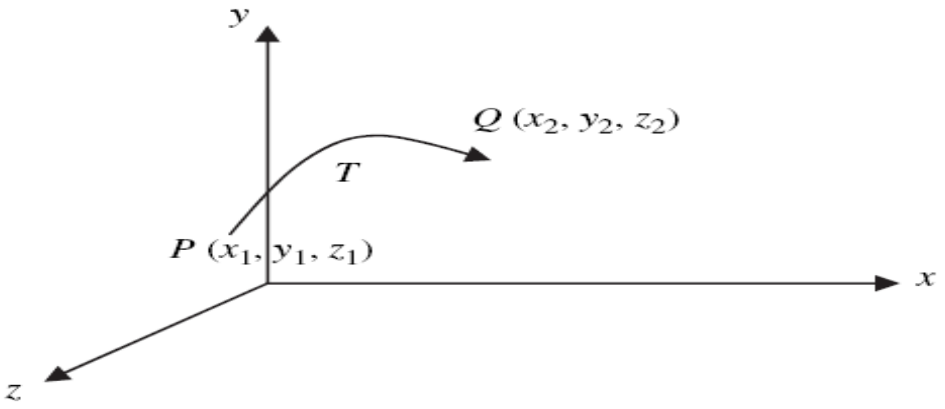


Figura 5. Movimiento de puntos en el espacio

La representación homogénea es:

$$P = \begin{pmatrix} x_1 \\ y_1 \\ z_1 \\ 1 \end{pmatrix} \quad \text{and} \quad Q = \begin{pmatrix} x_2 \\ y_2 \\ z_2 \\ 1 \end{pmatrix}$$

Si se quisiera mover un punto una distancia D (*dx, dy, dz*), el punto Q sería:

$$\begin{pmatrix} x_2 \\ y_2 \\ z_2 \\ 1 \end{pmatrix} = \begin{pmatrix} x_1 + d_x \\ y_1 + d_y \\ z_1 + d_z \\ 1 \end{pmatrix}$$

Para obtener la ecuación anterior se usa una matriz que contenga los valores de D, multiplicado por el vector P, como se muestra a continuación:

$$\begin{pmatrix} x_1 + d_x \\ y_1 + d_y \\ z_1 + d_z \\ 1 \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 & d_x \\ 0 & 1 & 0 & d_y \\ 0 & 0 & 1 & d_z \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x_1 \\ y_1 \\ z_1 \\ 1 \end{pmatrix}$$

Un caso particular de traslación es la escala, la cual permite la magnificación del punto u objeto en la misma medida en que se multiplique el punto P por alguna escala. La representación matricial es:

$$\begin{pmatrix} x_2 \\ y_2 \\ z_2 \\ 1 \end{pmatrix} = \begin{pmatrix} s_x & 0 & 0 & 0 \\ 0 & s_y & 0 & 0 \\ 0 & 0 & s_z & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x_1 \\ y_1 \\ z_1 \\ 1 \end{pmatrix}$$

Las trasformaciones geométricas ofrecen una mejor apreciación del objeto o ambiente que se esté observando.

2.2.2. Luces y Sombras

El tipo de superficie del objeto, la posición de la fuente de luz, el color de ambos y la posición del observador contribuyen a la percepción total de la imagen. La superficie del objeto puede ser brillosa y reflejar la luz en direcciones bien definidas o ser áspera y reflejarla de manera difusa. También existe una superficie donde la luz es reflejada y transmitida a través del material, al cual se le llama *traslucido* [Buss 2003].

En OpenGL la iluminación se considera resultado de tres componentes diferentes: 1) luz ambiental, 2) luz difusa y 3) luz especular [Whitrow 2008]. La luz ambiental es la iluminación uniforme de fondo en el ambiente. La luz difusa es direccional y se refleja uniformemente en la superficie hacia todas direcciones, haciéndola más brillante en la dirección en que recibe la luz. La luz especular, refleja abruptamente un punto de la superficie (un laser por ejemplo).

Es necesario tomar en cuenta que la reflexión de los componentes RGB (Red Green Blue) será diferente para cada superficie dada y para cada uno de los tres componentes de la que definen a la luz. La transparencia es otro efecto de iluminación, aunque no es parte del modelo de iluminación; se presenta cuando un objeto permite tanto la reflexión como la transmisión de la luz. La transparencia se simula agregando un cuarto componente al modelo RGB, el componente A (componente llamado Alpha). La sombra se forma cuando la luz ha sido detenida en su trayectoria a un plano por un objeto ubicado entre el plano y dicha luz proyectando una representación plana de las partes del objeto que sean iluminadas. Las sombras agregan realismo a cualquier imagen.

2.2.3. Mapeo de Texturas

El proceso de mapeo de texturas (colocar algo sobre una superficie, en este caso una imagen sobre un objeto), involucra el movimiento del espacio de la textura al espacio cartesiano usado por el objeto. Las imágenes mapeadas sobre superficies frecuentemente son un arreglo de pixeles (mapa de bits). A las imágenes que no tienen correspondencia uno a uno con la superficie en las que serán adheridas (la imagen es cuadrada y la superficie triangular, por ejemplo) se les refiere como un *arreglo de texels* (Figura 6).

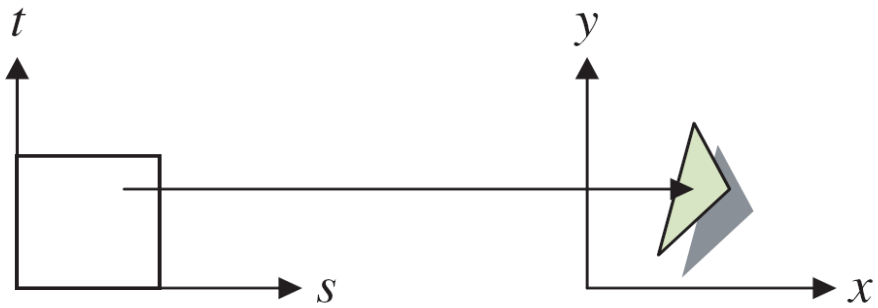


Figura 6. Mapeo de arreglo de texels.

El mapa de bits es definido en un espacio de texels (plano *s, t*) y requiere de una función de mapeo que pegue los texels a la superficie del objeto. Las posiciones en el plano cartesiano (x, y) están en función de *s* y *t.*

Para poder incorporar capacidad de texturizado, [Shreiner 2008] propone:

- Tener una imagen capturada dentro de un arreglo de memoria y especificar cómo cada fila de pixeles en la imagen está alineada.
- Identificar cada objeto para ser usado bajo un único nombre.
- Especificar controles requeridos para el cambio de espacio de los pixeles de la imagen al espacio de textura.
- Definir la imagen que está siendo usada en términos de dimensiones, formato de almacenamiento, tamaño y ubicación de memoria.

2.3. Trabajo Relacionado

El desarrollo de herramientas de simulación permite ahorrar recursos en el proceso de pruebas de los resultados arrojados por algoritmos de planificación de movimientos desarrollados para algún fin. Ropsim y Motion Strategy Library, se describen como un ejemplo de tales herramientas.

2.3.1. Ropsim

Ropsim es un sistema comercial para la simulación y programación de robots industriales (brazos robóticos). El propósito de Ropsim es hacer programas para robots que puedan ser descargados al robot dado. Para programar en este sistema se deben considerar tres fases: modelado, programación y generación de código. En la primera fase se diseñan los productos para que se puedan automatizar. En la segunda fase se puede usar y controlar el robot de forma física para las pruebas. La última fase genera programas para los respectivos robots. Ropsim genera un archivo que contiene el programa para el robot; el archivo puede

ser transferido al robot por medio de un disco, comunicación serial o por una red de información, para así para controlarlo de forma automática [CAMELOT 2006].

2.3.2. Motion Strategy Library

Motion Strategy Library (MSL) es un sistema libre que permite el desarrollo y pruebas de algoritmos de planificación de movimientos. La arquitectura del sistema es orientada a objetos, fue desarrollado para el sistema Linux usando el lenguaje de programación C++.

MSL consiste de siete clases jerárquicas como se observa en la Figura 7. A continuación se describen estas clases.

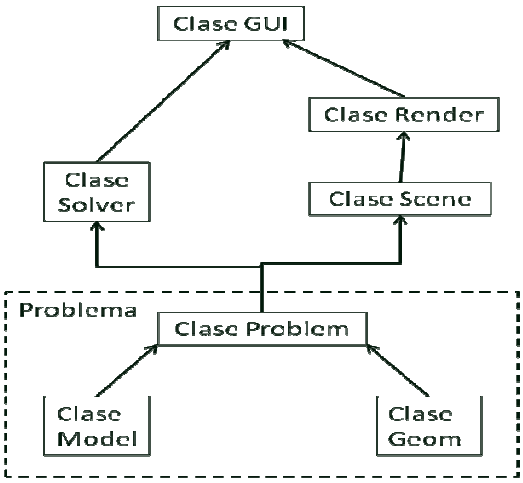


Figura 7. Clases de MSL

Clase *Model*.- Contiene simuladores incrementales que representan la variedad de cinemática y dinámica de los sistemas mecánicos (un robot, por ejemplo).

Clase *Geom*.- Define las representaciones geométricas de todas las partes que forman al mundo y al sistema mecánico.

Clase *Problem*.- Muestra el diseño abstracto de un algoritmo de planeación. Cada instancia del problema incluye la clase *Model* y *Geom*, un estado inicial y un estado final.

Clase *Solver*.- Contiene el método que resuelve el problema, generalmente es un algoritmo de planificación de movimientos.

Clase *Scene*.- Representa la interfaz que muestra las configuraciones de todos los cuerpos. Esta clase recibe la mayoría de la información de la clase *Problem*.

Clase *Render*.- Contiene diferentes implementaciones de renderización.

Clase *GUI*.- Es la interfaz grafica de usuario, está diseñada para habilitar interfaces específicas para problemas de planificación de movimientos y sus algoritmos. Cada instancia de la clase incluye una clase *Render*.

2.3.3. Comparaciones

La Tabla 3 presenta las principales diferencias que existen entre Ropsim, MSL y GEMPA.

Tabla 3. Diferencias entre GEMPA, Ropsim y MSL.

	GEMPA	Ropsim	Motion Strategy Library (MSL)
Costo	No comercial	Necesita licencia	No comercial
Extensibilidad	Simula la información de archivos off y txt	Simulación de brazos mecánicos propios	Permite simulación de espacios complejos
Compatibilidad	Multi-plataforma	Windows	Multi-plataforma
Adaptabilidad	Permite modificar y agregar módulos al código del programa.	No permite la modificación de su código fuente ni la incorporación de módulos ajenos al sistema	Permite modificar y agregar módulos al código del programa

Capítulo 3. Diseño del Sistema

Este capítulo describe el diseño del sistema para la simulación de ambientes virtuales en base a los requerimientos funcionales y no funcionales.

3.1. Requerimientos

Las tablas 4-7 describen requerimientos funcionales del sistema propuesto.

a) Requerimientos Funcionales:

Tabla 4. Requerimiento funcional para visualizar figuras predeterminadas

Identificador:	RF-1
Nombre de requerimiento:	Graficación de figuras predeterminadas
Descripción corta:	Muestra una figura
Descripción detallada:	1.- El usuario selecciona el menú *Figuras* 2.- El usuario elige una figura (cubo, tetera, toroide, cono o esfera) 3.- El usuario ve la figura seleccionada en la pantalla

Tabla 5. Requerimiento funcional para cambiar la resolución

Identificador:	RF-2
Nombre de requerimiento:	Cambio de resolución de la ventana
Descripción corta:	Cambia la resolución de la ventana
Descripción detallada:	1.- El usuario selecciona el menú *Pantalla* 2.- El usuario elige la opción resolución 3.- El usuario elige la resolución deseada (640 x 480, 800 x 600, 1024 x 768)

Tabla 6. Requerimiento funcional para explorar el ambiente

Identificador:	RF-3
Nombre de requerimiento:	Rotación y traslación
Descripción corta:	Mueve o rota la figura en la pantalla
Descripción detallada:	1.- El usuario presióna el botón izquierdo del ratón (rotación) 2.- El usuario presiona el botón derecho del ratón (traslación) 3.- El usuario presiona el botón izquierdo y el botón derecho (zoom)

Tabla 7. Requerimiento funcional para activación de luz

Identificador:	RF-4
Nombre de requerimiento:	Iluminación
Descripción corta:	Activa las luces del sistema
Descripción detallada:	1.- El usuario presiona la tecla 'L' para activar o desactivar las luces del sistema

b) Requerimientos no funcionales:

- *Eficiencia.*- La respuesta del sistema deberá ser instantánea, excepto en lectura de mallas (archivos .off y .txt), pues dependerá del tamaño del archivo.
- *Compatibilidad.*- El sistema se desarrollará con el sistema operativo Microsoft Windows en lenguaje C de ANSI con librerías de OpenGL.
- *Portabilidad.*- El sistema podrá ser utilizado en plataformas Linux.
- *Costo.*- El desarrollo del sistema no requerirá de licencias, se generará un software libre con fines educativos.

3.2. Casos de Uso

La Figura 8 muestra los casos de uso (CU_) del sistema, cada uno se implementa en un módulo como se describe a continuación.

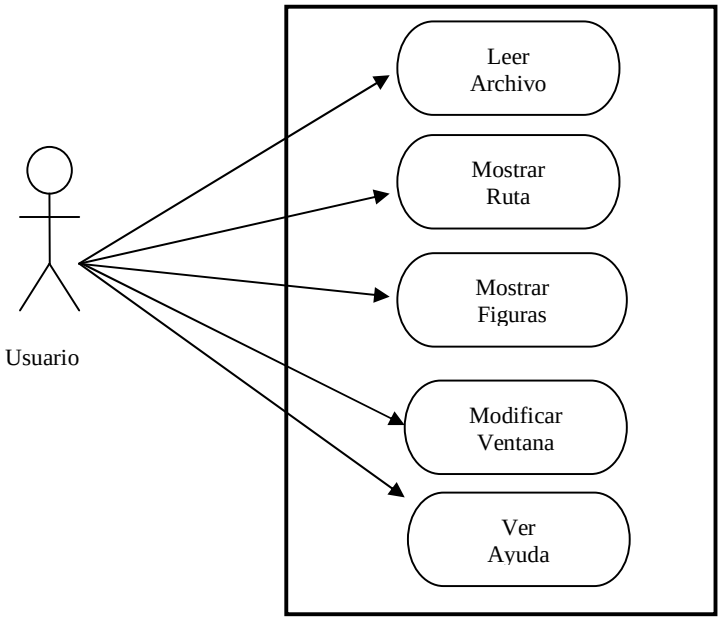


Figura 8. Casos de uso del sistema

Módulo Archivo: Lee las mallas de las figuras o ambientes a representar en el sistema, (ver Figura 9).

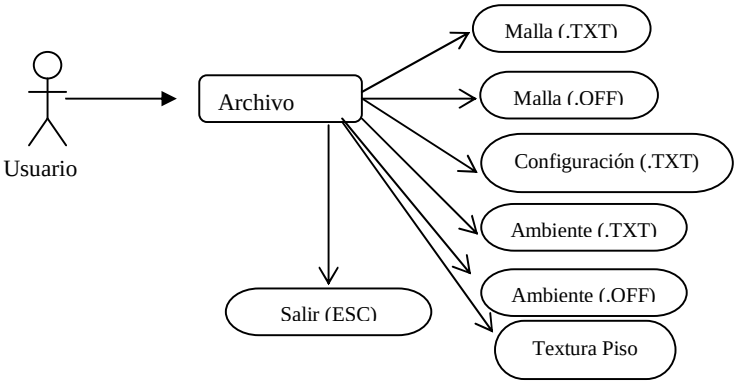


Figura 9. Funciones del módulo archivo

Módulo Ruta: Presenta el resultado de un algoritmo de planificación de movimientos, (ver Figura 10).

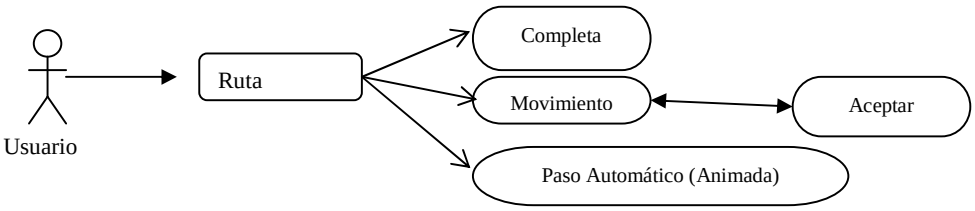


Figura 10. Funciones del módulo Ruta

Módulo Figuras: Muestra las figuras predefinidas de OpenGL, (ver Figura 11).

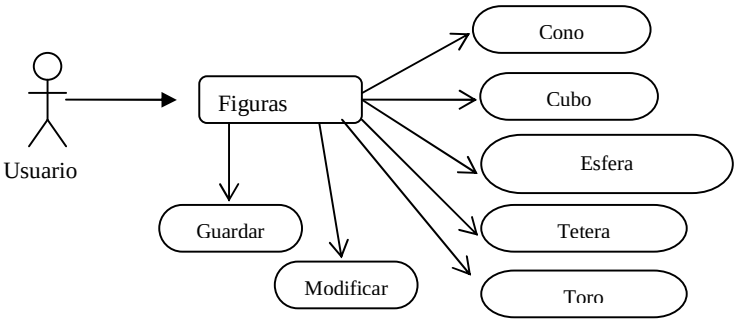


Figura 11. Funciones del módulo Figuras

Módulo Ventana: Permite el cambio en la resolución de la ventana del sistema, modificar el color del fondo y de la luz del ambiente (ver Figura 12).

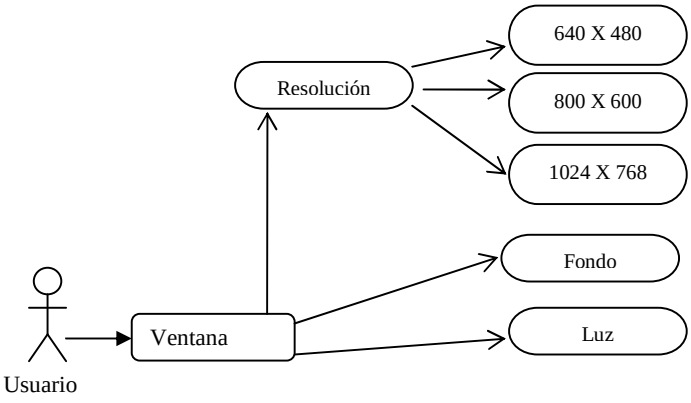


Figura 12. Funciones del módulo Ventana

Modulo Ayuda: Contiene y muestra la ventana de ayuda del sistema.

En las tablas 8-11 se describen algunos casos de uso del sistema

Tabla 8. Caso de uso iluminación

Identificador:	CU_01 Iluminación
Descripción:	Activa o desactiva la iluminación de la figura en la pantalla
Personal involucrado:	Usuario: decide el momento en que será activada o desactivada la iluminación
Precondiciones:	Tener un teclado
Garantías de éxito: **(Post condiciones)**	La imagen en la pantalla muestra color y reflejos de iluminación
Escenario principal de éxito o flujo básico:	1.- El usuario presiona la tecla 'L' de su teclado 2.- El usuario observa los reflejos de luz de la figura que está en la pantalla del sistema
Extensiones o flujos alternos:	No se muestra la iluminación a) Reintentar b) Probar con otra malla c) Usar mallas definidas por triángulos d) Cambiar el color o posición de la luz

Tabla 9. Caso de uso lectura de archivo

Identificador:	CU_02 Lectura de archivo
Descripción:	Recupera la información de un ambiente o figura desde archivos .off o .txt

Personal involucrado:	Usuario: elige la malla (archivo .off o .txt) a leer
Precondiciones:	Contar con archivos de malla con extensión .off o .txt definidas por triángulos
Garantías de éxito: **(Post condiciones)**	La información recuperada se muestra como una figura o ambiente en la pantalla del sistema
Escenario principal de éxito o flujo básico:	1.- El usuario elige el tipo de malla que leerá el sistema 2.- El usuario observa la información que el archivo de malla contiene representada en una figura.
Extensiones o flujos alternos:	No se muestra la figura o ambiente a) Esperar a que termine de cargar la información del archivo de la malla b) Usar mallas definidas por triángulos c) Probar con otra malla

Tabla 10. Caso de uso figuras

Identificador:	CU_03 Figuras
Descripción:	Muestra en la pantalla una figura predefinida por primitivas de OpenGL
Personal involucrado:	Usuario: elige la figura a mostrar (cubo, tetera, toroide, cono o esfera) y sus atributos (posición, tamaño, entre otros).
Precondiciones:	----------
Garantías de éxito: **(Post condiciones)**	La figura elegida se muestra en la pantalla del sistema
Escenario principal de éxito o flujo básico:	1.- El usuario selecciona el menú *Figuras* 2.- El usuario elige la figura a mostrar

Extensiones o flujos alternos:	No se muestra la figura a) Hacer traslación en eje z (“zoom”) b) Probar con otra figura

Tabla 11. Caso de uso pantalla

Identificador:	CU_04 Pantalla
Descripción:	Cambia la resolución de la pantalla del sistema
Personal involucrado:	Usuario: selecciona la nueva resolución (640x480, 800x600, 1024x768) de la pantalla
Precondiciones:	Tener la ventana del sistema abierta
Garantías de éxito: **(Post condiciones)**	La resolución de la pantalla cambia a la resolución elegida
Escenario principal de éxito o flujo básico:	1.- El usuario selecciona el menú *Pantalla* 2.- El usuario elige la opción resolución 3.- El usuario acepta la selección
Extensiones o flujos alternos:	No cambia la resolución a) probar con una opción de resolución diferente

3.3. Casos de Pruebas

Las tablas 12-15 son casos de pruebas (C.P.) de los casos de uso (C.U.) descritos en la sección anterior.

Tabla 12. Caso de prueba de lectura para el CU_02

Datos Entrada:	Lamp.txt (17 KB)

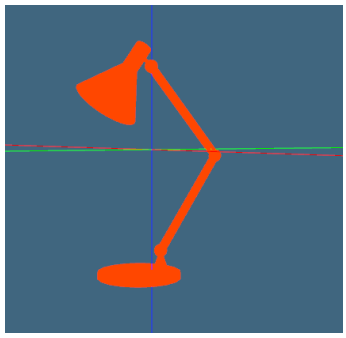
Datos Salida:	

Tabla 13. Caso de prueba iluminación para el CU_01

Datos Entrada:	Tecla ‘L’ Conejo.txt (75 KB)
Datos Salida:	

Tabla 14. Caso de prueba figuras-tetera para el CU_03

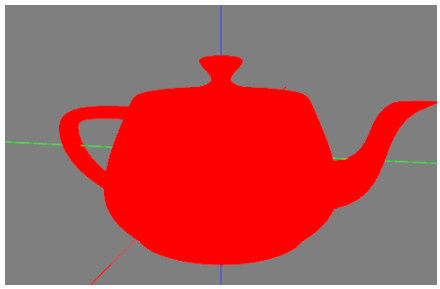
Datos Entrada:	Selección de la opción *Tetera* en el menú *Figuras*
Datos Salida:	

Tabla 15. Caso de prueba pantalla para el CU_01

Datos Entrada:	Selección de la opción *Resolución* del menú *Pantalla*
Datos Salida:	

El usuario (descrito en la Tabla 16) podrá leer archivos de texto con extensiones .txt y .off que representen mallas o ambientes. Los archivos .off que se usarán en el proyecto son de distribución gratuita extraídos de páginas relacionadas con graficación y desarrollo de ambientes[1]. Los archivos .txt han sido obtenidos de [Ramírez 2005], tampoco representan ningún costo.

Tabla 16. Descripción del usuario

Nombre del actor:	Usuario
Descripción:	Persona que usa el sistema
Características:	Persona que dispone de mallas para hacer pruebas de simulación
Relaciones:	Interactúa con el sistema para probar resultados de algoritmos de planeación de movimientos, mostrar mallas y simular ambientes

[1] http://shape.cs.princeton.edu/benchmark/index.cgi

3.5. Arquitectura

Los módulos de las tablas 17-20 corresponden a la arquitectura del sistema.

Tabla 17. Módulo Archivo

Descripción:	Lee una malla (figura o ambiente) o textura
Datos de entrada:	Nombre del archivo acorde a la lectura seleccionada
Datos de salida:	La figura, ambiente o textura acorde al contenido del archivo

Tabla 18. Módulo Ruta

Descripción:	Contiene las opciones de simulación de planeación (movimiento de un objeto entre obstáculos)
Datos de entrada:	Número de pasos a avanzar, en el caso de ruta a pasos
Datos de salida:	La figura desplazándose en la ruta definida y en la propia ruta

Tabla 19. Módulo Figuras

Descripción:	Tiene figuras predeterminadas por primitivas de OpenGL y opciones para modificarlas
Datos de entrada:	Selección de la figura, color, posición, tamaño, y rotación de ésta
Datos de salida:	La imagen 3D de la figura seleccionada con los datos especificados

Tabla 20. Módulo Pantalla

Descripción:	Maneja diferentes resoluciones para la pantalla del sistema, opciones de color de fondo y luz
Datos de entrada:	Selección de la nueva resolución, selección de un nuevo color para fondo o luz
Datos de salida:	La pantalla cambia a la resolución seleccionada, se muestra el cambio en el color de fondo del sistema o el cambio en el color de la luz

Capítulo 4. Implementación del Sistema

Este capítulo describe algunos detalles de implementación del sistema como los controles de navegación, cajas de diálogo, barras de menú, espacio de visualización, así como los elementos utilizados para la iluminación.

4.1. Controles de Navegación

Rotación, Traslación y escala: Para permitir el movimiento de las figuras dentro del espacio virtual se deben hacer cálculos matemáticos por cada vértice que forman a la figura. Cada vértice debe multiplicarse por la matriz que represente la transformación deseada (véase sección 2.2.1) para así obtener el efecto para la figura (una rotación, una escala o una traslación). En el instante en que se lee un archivo asociado a una malla, los vértices son procesados para así poder posicionar cada figura en el centro del espacio del ambiente virtual (coordinas x=0, y=0, z=0) con orientación hacia el usuario.

Cada vez que el usuario aplica una operación de traslación, rotación o escala para el ambiente (controles de navegación del sistema, ya sea por ratón o por teclado), los vértices asociados a cada figura contenida en él son recalculados utilizando las transformaciones mencionadas. La Tabla 21 muestra las teclas y botones que actualmente emplea el sistema para el control de la navegación.

Tabla 21. Control por teclado y ratón

Teclas	Descripción
Flechas	arriba, abajo (rotación sobre el eje X) , izquierda, derecha(rotación sobre el eje Y)
avpág	acercamiento de la cámara (traslación sobre el eje +Z)
repág	alejamiento de la cámara (traslación sobre el eje -Z)
W,w	activa/desactiva modelo en alambre
L,l	activa/desactiva luces del ambiente
F2	traslada el ambiente a la izquierda (traslación sobre el eje X)
F3	traslada el ambiente a la derecha (traslación sobre el eje X)
F4	traslada el ambiente hacia arriba (traslación sobre el eje Y)

F5	traslada el ambiente hacia abajo (traslación sobre el eje Y)
V,v	activa/desactiva el dibujado de los ejes XYZ
B,b	activa/desactiva la transparencia
Botones de Ratón	**Descripción**
Botón izquierdo	mientras se presiona activa la rotación en ejes XY del ambiente en dirección del arrastre del puntero
Botón derecho	mientras se presiona activa la traslación en ejes XY del ambiente en dirección del arrastre del puntero
Botón derecho + Botón izquierdo	Mientras se mantienen presionados ambos se activa la rotación con respecto a Z (moviéndolo de izquierda a derecha) y traslación en el eje Z (moviéndolo de arriba hacia abajo)

4.2. Cajas de Diálogo

Las cajas de diálogo permiten generar controles que se necesiten en un momento específico para definir o cambiar parámetros para la visualización. El uso de ventanas pretende que los controles no ocupen espacio en la interfaz cuando no se necesitan. Los encabezados de algunas de las funciones que administran las cajas de diálogo son:

```
LRESULT CALLBACK DialogProc ( HWND hwnd, UINT msg, WPARAM wParam, LPARAM lParam )
```

```
LRESULT CALLBACK AyudaProc ( HWND hwnd, UINT msg, WPARAM wParam, LPARAM lParam )
```

```
LRESULT CALLBACK PanelProc ( HWND hwnd, UINT msg, WPARAM wParam, LPARAM lParam )
```

Las cajas de diálogo son utilizadas para obtener la ubicación y el nombre de los archivos de que contienen la información para el pintado de espacios virtuales. Esta información esta contenida en archivos con extensión off o txt. Con las cajas de diálogo se definen además propiedades de las figuras predeterminadas como son: nombre, posición, orientación, color y tamaño. Se utilizan también para definir parámetros para la luz y el color de fondo del ambiente.

4.3. Barra de Menú

La barra de menú tiene opciones para activar las funciones siguientes:

Menú *Archivo*.- Contiene las opciones de lectura de archivos de mallas para la representación de espacios virtuales, así como la lectura de texturas. Cada opción determina el tipo de archivo que se leerá. Una vez seleccionada la opción, se abre una caja de diálogo de apertura de archivos.

Menú *Ruta*.- Muestra la ruta (secuencia de configuraciones) para un objeto dentro del ambiente. El ambiente y la ruta con las configuraciones para simular el problema de planeación deben ser leídos previamente en ese orden.

Menú *Figuras*.- Contiene figuras predeterminadas de openGL. Cada vez que se selecciona alguna, se introducen los datos de sus propiedades (nombre, posición, orientación, color y tamaño). Es posible colocar más de una figura dentro del ambiente, además incluye opciones para modificar las figuras predeterminadas y funciones de guardado y apertura de archivos creados por el usuario.

Menú *Ventana*.- Permite el cambio de resolución de la pantalla del sistema, el cambio de la luz del espacio virtual y su color de fondo.

Menú Ayuda.- Muestra la ayuda del sistema, contiene la información que se muestra en la Tabla 21.

4.4. Espacio de Visualización

El espacio de visualización utiliza los controles descritos en secciones anteriores, contiene las figuras que el usuario desee colocar o leer. El incremento en la cantidad de figuras o incluso en la cantidad de polígonos que definen a una figura puede afectar la velocidad de respuesta de los controles del sistema. El espacio puede contener uno o varios objetos, ya sea leídos desde un archivo off o txt, o bien, definidos por el usuario usando las figuras predeterminadas del sistema. Este sistema contiene algunas propiedades que serán descritas a continuación.

Color.- Es una propiedad que poseen las figuras predeterminadas, el fondo y la luz del ambiente, la gama de colores están definidos por el modelo RGB (Red, Green, Blue). La saturación completa de los tres canales da como resultado blanco y la falta de éstos el color negro.

Textura.- Es una propiedad del piso del ambiente, son imágenes de mapa de bits (bmp) que se adhieren a un polígono que define el piso.

Luz.- Se percibe por el reflejo del brillo que muestran las figuras del ambiente. La luz puede percibirse de modo distinto en cada figura dependiendo del color de la propia figura y del color de la luz.

Modelado.- Las figuras dentro del ambiente pueden ser vistas en tres diferentes modelos:

1. Alambre: muestra únicamente las aristas de los polígonos que definen a la figura
2. Flat (plano): muestra la figura de un solo color sólido y uniforme. En este modo se pierde la percepción 3D de las figuras
3. Lambert: es un efecto de iluminación que permite dar mayor realismo al ambiente al apreciar el reflejo de la luz sobre la superficie de los triángulos que definen a las figuras. Este modelado se explicado en la siguiente sección

4.5. Iluminación del Espacio de Visualización

La iluminación del espacio virtual considera el modelo de Lambert. Este modelo calcula el vector normal de cada triangulo con respecto a la posición del vector de la luz. El propósito es obtener un color diferente para cada triángulo. La normal de los triángulos queda definida por la formula siguiente:

$N=\overrightarrow{AB} \times \overrightarrow{AC}$

donde A, B y C son las coordenadas de los tres vértices que conforman cada triangulo.

El valor del vector $\overrightarrow{AB}=B-A$ se obtiene restando el valor del punto B menos el punto A, similar es el caso de $\overrightarrow{AC}=C-A$.

El cálculo de la normal se hace para cada triangulo cada vez que se hace una transformación en la figura, por tal razón, el uso de mallas con una gran cantidad de triángulos pueden afectar la velocidad de respuesta del sistema.

Capítulo 5. Resultados

Una vez implementadas operaciones para la navegación dentro del ambiente virtual y para la visualización de diferentes tipos de objetos, se integran ambas operaciones para obtener un una interfaz gráfica para la visualización de objetos y ambientes tridimensionales.

5.1. Espacio de Visualización

El espacio de visualización es la parte de la ventana en la que se aprecian las figuras que son colocadas o leídas por el sistema. El fondo por omisión es de color azul opaco y muestra los ejes X Y Z. La Figura 13 muestra una malla, la barra de menú y los ejes del espacio virtual, siendo la línea azul el eje Y, la línea verde el eje X, y la línea roja el eje Z

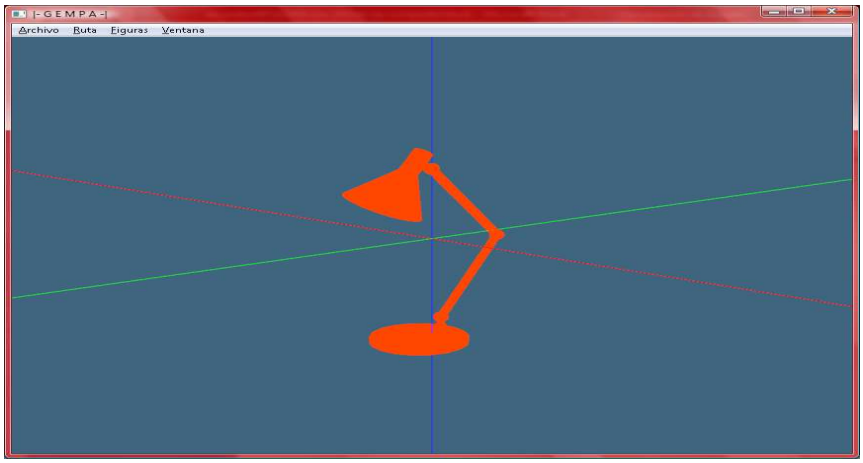


Figura 13: Interfaz con malla .txt sin luz activada

La Figura 14 muestra una ventana de MS-DOS que indica la cantidad de objetos que definen el ambiente virtual. La matriz representa la posición y rotación en la que será colocada la figura y los valores massx, massz y massy forman el centro de masa de la figura mostrada en el espacio virtual.

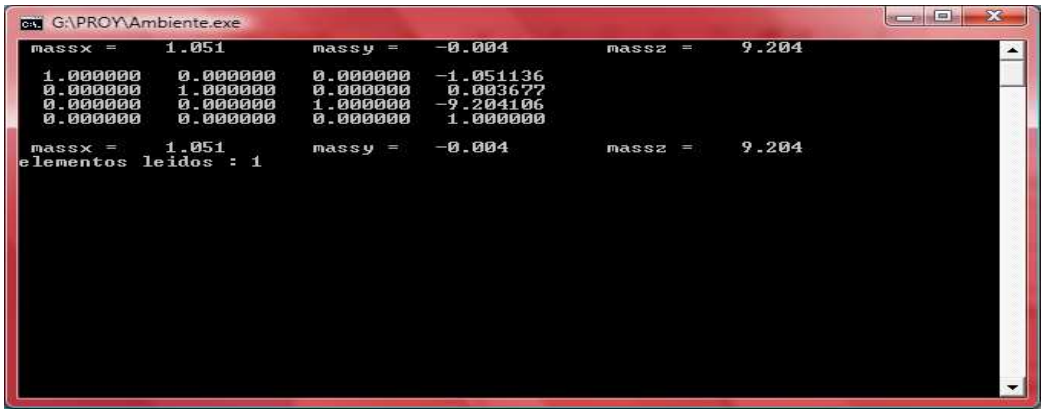


Figura 14. Consola MS-Dos generada por GEMPA

5.2. Lectura de Archivos

El sistema puede leer archivos de malla con extensión .txt y .off. Los archivos definen un ambiente virtual que puede contener una sola figura o bien un grupo de figuras dispersas dentro del espacio virtual. Aquellas que definan una sola figura se leen con la opción *Malla* (.off / .txt), las que contengan información para representar más de una figura se leerán con la opción *Ambiente* (.off/ .txt). Una vez que se haya leído un archivo de ambiente, si se dispone de la configuración de planeación de movimientos ("planning"), ésta será leída por medio de la opción *Configuración*. La Figura 15 muestra ventanas con una y varias figuras en el espacio virtual.

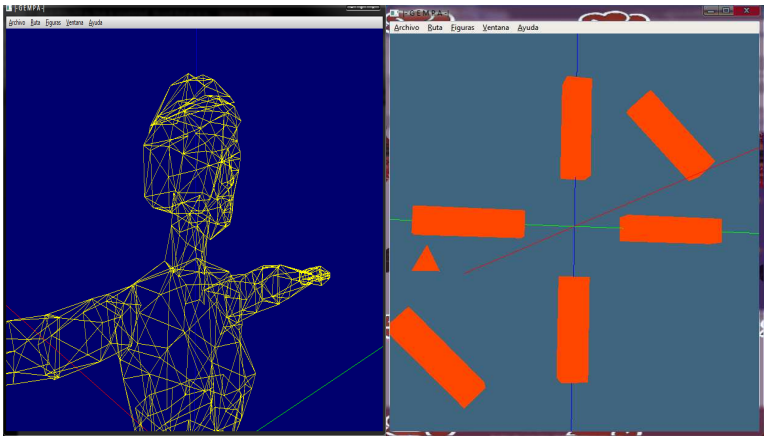


Figura 15. Ventana con malla .off en alambre (izq.), ventana con malla de ambiente .txt (der.)

5.3. Navegar en el Espacio Virtual

La rotación puede ser controlada por medio del teclado o del ratón, el usuario puede apreciar el efecto de rotación dentro del espacio virtual hasta colocar el ambiente en la posición deseada. La Figura 16 muestra un toroide antes y después de rotar el ambiente.

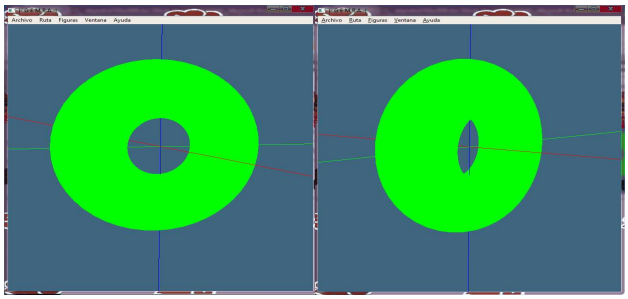


Figura 16. Toroide antes y después de rotar el ambiente.

La traslación, al igual que la rotación, puede ser controlada por teclado o ratón, afecta todo el ambiente permitiendo al usuario centrar la atención en alguna parte en particular (Figura 17).

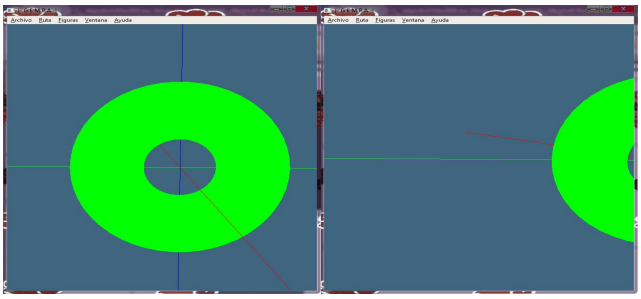


Figura 17. Antes y después de trasladar el ambiente

El caso particular de traslación en el eje Z representa hacer un zoom sobre el ambiente, dicho zoom se controla por medio de teclado o ratón apreciando así el efecto de acercamiento o alejamiento de alguna zona particular hasta la posición deseada (Figura 18).

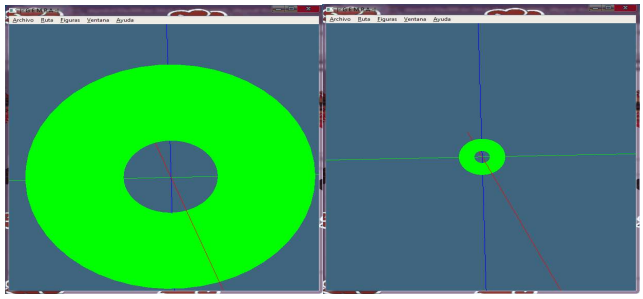


Figura 18. Antes y después de hacer zoom en el ambiente

La rotación del eje Z en particular sólo puede ser controlada por medio del ratón y afecta, al igual que las transformaciones anteriores, a todo el ambiente (Figura 19).

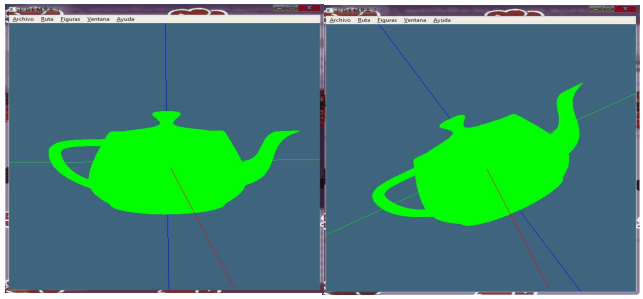


Figura 19. Antes y después de hacer rotación sobre el eje Z

5.4. Cajas de Diálogo

Para la apertura de archivos se genera una ventana que muestra una barra de direcciones, carpetas y archivos compatibles con el formato de archivo que se desee leer (.txt / .off / .bmp). La Figura 20 muestra la ventana para apertura de archivos.

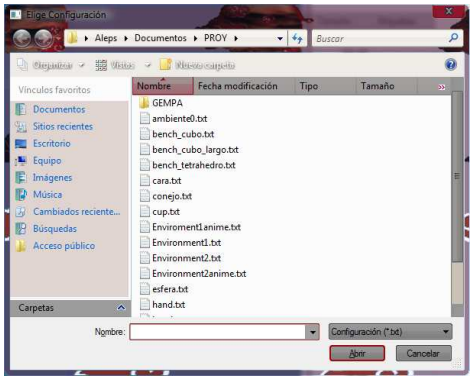
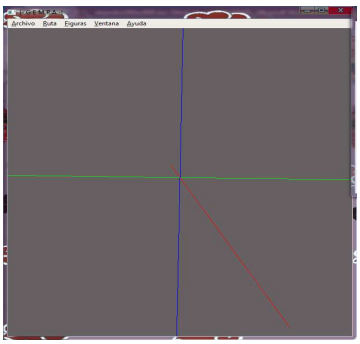
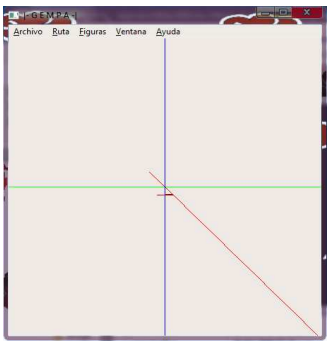


Figura 20. Ventana de apertura de archivos

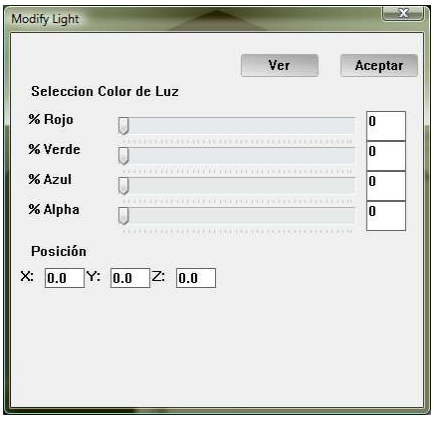
La ventana principal contiene ciertas propiedades que pueden modificarse durante el tiempo de ejecución. Estas propiedades se describen a continuación.

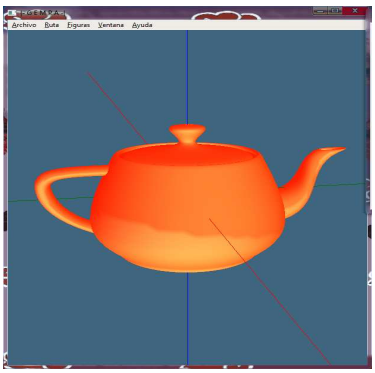
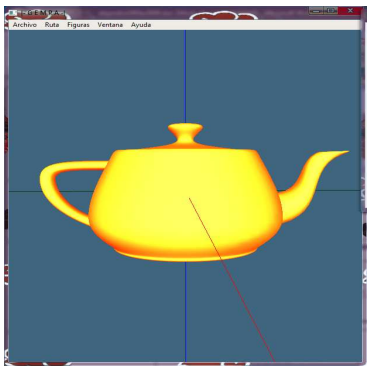
a) Fondo

Descripción	Dentro de esta caja se muestran barras de desplazamiento que definen el color RGB del fondo y el canal Alpha
Visualización	

Resultado	Antes: 	Después: 

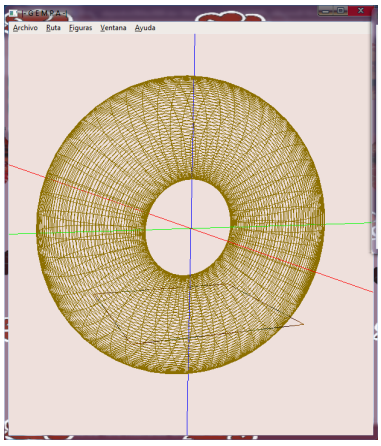
b) Luz

Caja	Caja de Diálogo para Luz
Descripción	Dentro de esta caja se muestran barras de desplazamiento que definen el color RGB de la luz y el canal Alpha (indicar que es un nivel de transparencia), además de cajas de texto para introducir la posición de la fuente de luz
Visualización	

Resultado	Antes: 	Después: 

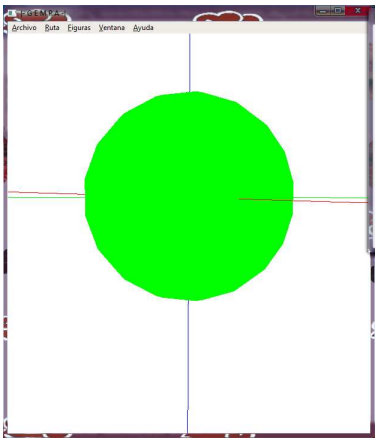
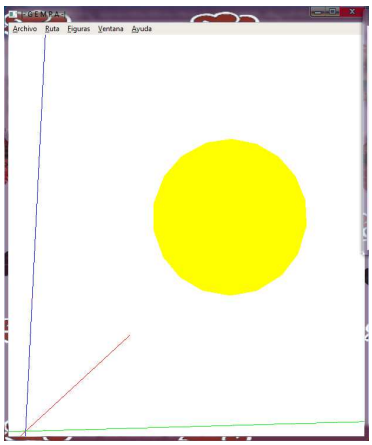
5.4.3. Figuras

Caja	Caja de Diálogo para Figuras
Descripción	En esta caja se muestran las especificaciones para la figura que se desee (toro, en este caso), se definen cajas de texto para el nombre, colores, posición y rotación de la figura, además de barras de desplazamiento para color y tamaño. Si el usuario no define un nombre, se le asigna uno automáticamente
Visualización	

Resultado	

5.4.4. Modificar

Caja	Caja de Diálogo para Modificar Figuras
Descripción	En esta caja se muestran las especificaciones de la figura que se desee modificar. Se selecciona una figura por nombre en una lista, se definen cajas de texto para el color, posición y rotación de la figura, además de barras de desplazamiento para color y tamaño
Visualización	

Resultado	Antes: 	Después: 

5.4.5. Guardar

Presenta una caja de diálogo en la que se escribe el nombre que se desea para el archivo que almacenará la información del ambiente que el usuario formó con las figuras predeterminadas. No se debe colocar ninguna extensión al nombre del archivo que se desee almacenar, puesto que se le asigna .txt por defecto.

Caja	Caja de Diálogo para Guardar Figuras
Descripción	Esta caja contiene una caja de texto en la que se coloca el nombre deseado al archivo sin extensión.
Visualización	

Resultado	 archivo de texto con el nombre definido

Capítulo 6. Conclusiones y Trabajo Futuro

En este proyecto se desarrolló exitosamente una interfaz gráfica utilizando el lenguaje C de ANSI y librerías de OpenGL como herramientas de desarrollo. El sistema tiene una ventana en la cual se puede visualizar el espacio para las figuras y/o ambientes virtuales, así como diferentes ventanas de diálogo para modificar los parámetros utilizados durante la visualización.

El sistema es una herramienta gráfica dotada de un conjunto de funciones y comandados que pueden introducirse por teclado o por ratón, lo cual permite una navegación sencilla dentro del espacio virtual. De esta manera, el usuario puede interactuar con el ambiente virtual para percibir efectos de inmersión.

Para recuperar información acerca de las malla de triángulos a ser dibujados dentro del ambiente virtual, GEMPA incluyen funciones de lectura de archivos con extensión off y txt (siendo éstos los formatos de archivos más frecuentemente utilizados por la comunidad en el área de graficación para la representación de malla). La nueva versión de GEMPA permite al usuario manipular figuras predeterminadas como son: tetera, cono, toroide, cubo y esfera. Estas figuras se definen en base a parámetros, mismos que pueden ser modificados a través de ventanas de diálogo. Así, el usuario puede trasladar, rotar, escalar y cambiar de color cualquier figura que desee incorporar al ambiente. Además, si el usuario lo desea, la herramienta le permite guardar la información necesaria de todos los objetos colocados para construir un ambiente utilizando figuras predeterminadas, dándole la posibilidad de poderlo recuperar posteriormente para su modificación.

Para darle mayor realismo al ambiente, se agregaron efectos de iluminación (Modelo de Lambert) y textura a un piso. Con el uso del piso se limita la percepción de espacio abierto. Este piso está definido con una textura la cual, si el usuario lo desea, puede ser cambiada por alguna de formato similar (imagen de mapa de bits -bmp-, con una resolución no mayor de 256x256 pixeles). Las propiedades de la luz (color y ubicación) pueden ser modificadas para mejorar la visualización de las figuras dentro del espacio iluminado, integrando también efectos de transparencia a los objetos, si el usuario lo requiere, puede además cambiar el color del fondo del ambiente.

Como se mencionó en la introducción, GEMPA se desarrolló inicialmente con el objetivo de ser utilizado para visualizar el comportamiento de algoritmos de planificación de movimientos para objetos rígidos. La segunda versión de GEMPA, misma que se desarrolló durante este proyecto de investigación, mejora significativamente a la primera en los siguientes aspectos:

- La carga de archivos off y txt, ampliando la posibilidad y gama de objetos que pueden ser pintados y manipulados desde GEMPA.
- El sistema de navegación a través de uso de ratón y teclado.
- Modelo de iluminación de Lambert, incorporando el cálculo de normales para cada triángulo de cada objeto en el ambiente.
- El manejo de texturas para el piso dentro del ambiente.
- La lectura de ambientes gráficos construidos a través de mallas con formato txt.
- La lectura de ambientes gráficos construidos a través de mallas con formato off.

Sin embargo, esta segunda versión aún no incorpora el pintado de rutas para algoritmos de planificación de movimientos, aunque define las estructuras de datos y funciones de pintado de objetos necesarias para esta operación. Ya que la primera versión de GEMPA si incluye opciones de pintado de rutas, será relativamente simple migrar esas operaciones hacia la nueva versión que está desarrollada en Dev C++, e implementa librerías de OpenGL.

Esta versión permite la manipulación del ambiente por medio de opciones visuales por medio de ratón y teclado que ayudan al usuarios a tener un mejor control en la exploración de los ambientes que represente el sistema, contiene iluminación, cambio en la gama de colores de diferentes componentes, texturas, uso y almacenamiento de ambientes con figuras predeterminadas, así como la recuperación de información desde archivos de mallas (.off y .txt).

Lamentablemente dentro de la iluminación no se incluye la propiedad de sombreado, esto porque la implementación de sombras implica una cantidad inmensa y repetitiva de

cálculos que propician una reducción agravante en la velocidad de respuesta de la interfaz, además de un incremento en los requerimientos de hardware.

Como trabajo futuro, se pueden describir algunas funcionalidades que harían de GEMPA una herramienta más completa y competitiva.

Algunas de estas funcionalidades se listan en orden de importancia.

- Incorporar algoritmos de planificación de movimientos como el PRM (Probabilistic Roadmap Method) , Visibility Roadmap y RRT.
- Dotar a GEMPA con la capacidad de representar objetos articulados.
- Integrar a GEMPA la capacidad de recuperar información desde archivos XML
- Insertar el manejo de algoritmos de detección de colisiones a GEMPA.
- Enriquecer la interfaz con el manejo de texturas y sombras.

Referencias.

- [Aldrich 2003] C. Aldrich. 2003. Learning by Doing : A Comprehensive Guide to Simulations, Computer Games, and Pedagogy in e-Learning and Other Educational Experiences. San Francisco. John Wiley & Sons.
- [Benitez 2005] Antonio Benitez Ruiz. 2005. Motion Planning Using Geometric Features. Tesis doctoral. Universidad de las Américas – Puebla. México.
- [Bourke 2006] Paul Bourke. 2006. OFF-A3D object file format. Disponible: <http://steve.hollasch.net/cgindex/formats/off.txt. Visitado 28/09/2008>
- [Buss 2003] Samuel R. Buss. 2003. 3-D Computer Graphics A Mathematical Introduction with OpenGL. Cambridge University Press. New Tork.
- [CAMELOT 2006] CAMELOT Robotics. 2006. Ropsim Handling demo. Disponible: <http://www.camelot.dk/Downloadropsim.aspx. Visitado: 29/09/2008>
- [Cruz 1993] Carolina Cruz-Neira et al. 1993. Suround-screen projection-based virtual reality: the design and implementation of the CAVE. Proceedings 20ST Annual Conference on Computer Graphics and Interactive Techniques
- [De los Santos 2004] Eric de los Santos de la Rosa. 2004. Heurística para la generación de configuraciones en pasajes estrechos aplicada al problema de los clavos. Tesis de Maestría. Universidad de las Américas – Puebla. México.
- [Deepak 2000] Tolani Deepak et al. 2000. Real Time Inverse Kinematics Techniques for Antropomorphic Limbs. Pennsylvania, Philadelphia. Academic Press.
- [Dowling 1996] Kevin Dowling. 1996. What is robotics?. Disponible: <http://www.cs.cmu.edu/~chuck/robotpg/robofaq/1.html. Visitado: 24/09/2008 >
- [Hearn 1994] Hearn Donald et al. 1994. Gráficas por computadora. Prentice-Hall Hipanoamericana. Mexico.
- [Latombe 1991] Jean-Claude Latombe. 1991. Robot Motion Planning. Springer.
- [La Valle 2003] Steven M. La Valle. 2003. Motion Strategy Library. Simulation freeware. Disponible: <http://msl.cs.uiuc.edu/msl/index.html. Visitado: 29/09/2008>

- [La Valle 2006] Steven M. La Valle. 2006. Planning Algorithms. Cambridge University Press.
- [Ramirez 2005] Ramirez Ortega Ma. del Carmen. 2005. Estudio e implementación de un algoritmo de detección de colisiones basado en esferas. Tesis de maestría. Universidad de las Américas-Puebla.
- [Ramos 2003] Ramos Nava Ma. del Carmen. 2003. Realidad Virtual. Entérate en línea. Año 2. Numero 24. Disponible: <http://www.enterate.unam.mx/Articulos/2003/noviembre/realivirt.htm. Visitado: 28/09/2008>
- [Shreiner 2008] Shreiner Dave, et al. OpenGL programming guide: the official guide to learning OpenGL. 2008. Addison-Wesley.
- [Suárez 2003] Suárez P. K. 2003. Animación y Visualización de Fenómenos Naturales. Tesis Licenciatura. Universidad de las Américas-Puebla. Mexico.
- [Whitrow 2008] Whitrow Robert. 2008. OpenGL Graphics Through Applications. Springer. Londres.
- [Wright 2003] Wright S. Richard, Jr et al. 2003. OpenGL Superbible Fourth Edition.Addison-Wesley.