UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO

FACULTAD DE INGENIERÍA

SOFTWARE DE OPERACIONES Y VISUALIZACIÓN PARA UN SATÉLITE EDUCATIVO

T E S I S

QUE PARA OBTENER EL TITULO DE

INGENIERO EN COMPUTACIÓN

P R E S E N T A :

SÁNCHEZ GUINEA ALEJANDRO

DIRECTOR:

DR. ESAÚ VICENTE VIVAS

Como parte del proyecto número 7126 financiado por CONACYT

México D.F. 2008

ÍNDICE

CAPÍTULO 1. ANTECEDENTES Y JUSTIFICACIÓN DEL SOFTWARE DE OPERACIONES Y VISUALIZACIÓN

El desarrollo de cualquier país que pretenda un buen crecimiento, dependerá en parte, de la inversión que se haga en el terreno de la ciencia y la tecnología, sin importar que sus resultados sean inmediatos o no.

La investigación espacial, ha representado una serie de aportaciones importantes para la humanidad, que con el tiempo se convirtieron en elementos fundamentales de otra clase de investigaciones así como de nuevos desarrollos tecnológicos sobre diversas áreas. Así también, la vida como hoy la conocemos, sería inconcebible sin las comunicaciones basadas en satélites, que permiten conectar al mundo entero en tantos y tan variados ámbitos como puedan imaginarse. Por ejemplo, hoy en día podemos contar con teléfonos celulares como el de la figura 1.1, que permite ver la televisión además de hablar por teléfono.

Figura 1.1. Teléfono celular japonés

Sobra entonces decir, que la investigación espacial es fundamental para el desarrollo científico y tecnológico de cualquier sociedad, representando: un canal para investigaciones futuras derivadas del conocimiento adquirido, un enorme campo para el desarrollo de técnicas de investigación del entorno, así como un medio que fomenta el desarrollo de nuevos sistemas de de comunicación y tecnología de punta en bastantes áreas. De esta manera, es muy importante la

inversión en el campo espacial para que México mejore su desarrollo como nación competitiva a nivel internacional.

Teniendo en cuenta que el recurso humano es uno de los más importantes en la realización de un proyecto tecnológico, desde hace un par de años en el instituto de ingeniería de la Universidad Nacional Autónoma de México se desarrolla un sistema satelital para entrenamiento de recursos humanos, de nombre SATERHUM.

SATERHUM fue pensado como un sistema portátil que se utilice en laboratorios escolares. Está compuesto por un dispositivo que emula los principales subsistemas que componen a un satélite real, denominado por su función explícita como SATEDU (Satélite Educativo), figura 1.2, y por un software que se ejecutará en una computadora portátil para que todo el sistema sea portable. Este software permitirá el control y la visualización operativa de SATEDU.

Figura 1.2. Maqueta SATEDU

Por otro lado, la idea del proyecto fue concebida a partir de la experiencia directa y previa en proyectos satelitales nacionales de diversas magnitudes. México, desde hace más de dos décadas, ha participado en el área de tecnologías satelitales, primero como usuario y recientemente en proyectos de investigación y desarrollo. Uno de los proyectos más recientes y ambiciosos ha sido SATEX, figura 1.3, el cual es un micro-satélite experimental que fue proyectado para realizar diversas investigaciones en el espacio exterior.

Dentro del proyecto SATEX, el instituto de ingeniería de la UNAM desarrolló exitosamente los sistemas de computadora de vuelo, el subsistema de sensores, los protocolos de comunicaciones, el software de vuelo y el software de estación terrena. Con esto, se obtuvieron bastantes conocimientos y experiencia en el área, además de una especial motivación para fundar las bases de un verdadero desarrollo en el terreno satelital.

El trabajo realizado para SATEX, hizo evidente la falta no sólo de experiencia y tecnología nacional en este campo, sino de las suficientes personas que participaran en esa clase de proyectos. Respecto a éste último rubro, se identificó la necesidad de crear nuevos medios de acceso para incorporar a un mayor número de estudiantes y para ofrecerles una mejor preparación tecnológica e ingenieril.

Figura 1.3. Proyecto Microsatelital Nacional.

SATERHUM entonces, se originó tomando en cuenta lo aprendido y desarrollado con anterioridad, especialmente en el proyecto SATEX, figura 1.3. Además, al ser SATERHUM un proyecto con fines didácticos, se le dio especial atención para presentar la información del prototipo satelital de manera sencilla y atractiva, mediante un modelo de construcción que es simple, fehaciente y económico.

En lo que resta del capítulo, se abordan temas que permiten un mejor entendimiento del Sistema Satelital para entrenamiento de Recursos Humanos, y se analiza además cada uno de los subsistemas que lo componen, incluyendo el Software de operaciones y visualización, tema central de esta tesis.

1.1 LAS CONVENIENCIAS DE DESARROLLAR UN SATERHUM

La educación es un factor muy importante en el desarrollo de un país que pretenda construir una realidad promisoria en el área científica y tecnológica. Siendo necesario, tanto profundizar en temas muy difundidos como abarcar aquellos que hayan sido desatendidos. Este es el caso de la investigación y el desarrollo satelital, figura 1.4; en donde los medios con que cuenta nuestro país para una óptima preparación estudiantil, son limitados.

Figura 1.4. Tecnología satelital contemporánea.

En este sentido, SATERHUM representa un camino bastante confiable para el establecimiento de programas que sienten las bases en el área de satélites pequeños en México. Presentando entre sus características: una alta capacidad para realizar las copias que resulten necesarias, una notable versatilidad en la interacción con los usuarios, además de una construcción altamente portable.

El diseño del Sistema Satelital para entrenamiento está compuesto por un prototipo que se aloja en un contenedor de discos compactos, el cual contiene la mayoría de los principales subsistemas empleados en satélites. Tiene además, un CD con el software para visualizar y controlar al dispositivo o prototipo satelital. De este modo, el sistema completo se compone de un hardware sencillo y económico, con el que se puede interactuar desde una interfaz de software sencilla, ilustrativa y funcional, instalado en una computadora convencional. En la figura 1.5 se muestra un sistema comercial desarrollado por la Fuerza Aérea Estadounidense que actualmente vende la compañía "Colorado Satellite Services" de EU, el cual tiene algunas similitudes con el hardware del SATERHUM.

Figura 1.5. Laboratorio de la Fuerza Aérea de EE.UU en donde desarrollaron el sistema denominado Eyasat; para SATEDU se prevé un escenario parecido.

1.2 SIMILITUDES ENTRE UN SISTEMA REAL Y SATERHUM

Al momento de realizar una simulación de un sistema real hecho por el hombre, se necesitan tomar en cuenta dos factores principales propios del sistema, los que forman parte del medio en que se pretende trabajar. Por un lado están la conformación de los elementos constitutivos del sistema, y por otro, el medio en que éste se desenvuelve.

La simulación de los elementos creados por el hombre, representa en la mayoría de los casos una interpretación que sólo cambia parámetros de costos, proporciones de conformación y de dimensiones del producto final. Haciéndola una tarea menos complicada, que la de lograr simular el medio de trabajo del sistema, sobre todo cuando se trata de ambientes poco comunes o muy extremosos.

Así, SATERHUM en la primera etapa de desarrollo en que se encuentra, tiene como objetivo lograr que sus usuarios comprendan las partes que conforman a un satélite real, con base en la

experimentación que realicen sobre el prototipo SATEDU. Dejando a un lado por el momento, la simulación de los factores del medio de trabajo de los satélites comerciales, es decir, el espacio exterior.

Sin embargo, el dispositivo SATEDU se ha desarrollado como un elemento que evolucionará hacia un pico-satélite[1], figura 1.6. Por lo que se han tomado en cuenta, en el diseño y fabricación de algunos subsistemas, las protecciones necesarias para que puedan operar en un medio altamente hostil en términos de radiación, temperaturas y vacío.

Figura 1.6. Picosatélite ColombianoLibertad 1,
comparado con un transmisor manual.

El sistema satelital para entrenamiento de recursos humanos, cuenta con los subsistemas de estructura, computadora de vuelo, potencia, estabilización por rueda inercial y bobinas de torque magnético, sensores de navegación inercial, comunicaciones, así como software de operaciones y visualización. Cada uno de los cuales son imprescindibles en satélites comerciales contemporáneos. De esta manera, quienes empleen el sistema para entrenamiento contarán con un elemento para experimentar y obtener buenos conocimientos no solo en el campo satelital sino en los campos vinculados a sus subsistemas.

1.3 ARQUITECTURA PROPUESTA PARA EL SATERHUM

Como ya se planteó, los puntos clave que se tomaron en cuenta en el diseño del SATERHUM fueron: la economía de los materiales empleados, la versatilidad en su uso y construcción; además de su capacidad para representar de manera confiable y útil, un sistema satelital real.

SATERHUM está compuesto por dos módulos fundamentales, mostrados en la figura 1.7. Por un lado, un prototipo de un satélite real desarrollado para operar en un ambiente de laboratorio, el cual cuenta con una serie de subsistemas alojados en circuitos impresos. Por otro, un software para el control y la visualización en tiempo real del prototipo mencionado, cuyo desarrollo es independiente en buena medida a la del SATEDU.

El hardware del Sistema para entrenamiento de recursos humanos, está compuesto por un conjunto de circuitos impresos de 8.9 cm^2, que representan los subsistemas del satélite. De esta forma, es posible acomodar las tarjetas una sobre otra, comunicadas con un par de conectores entre cada una y dispuestas dentro de una carcasa de acrílico transparente, que permite visualizar los elementos del prototipo completo, figura 1.8.

[1] Pico-satélite: satélite miniaturizado cuya masa va de los 10 gr. hasta 1 kg.

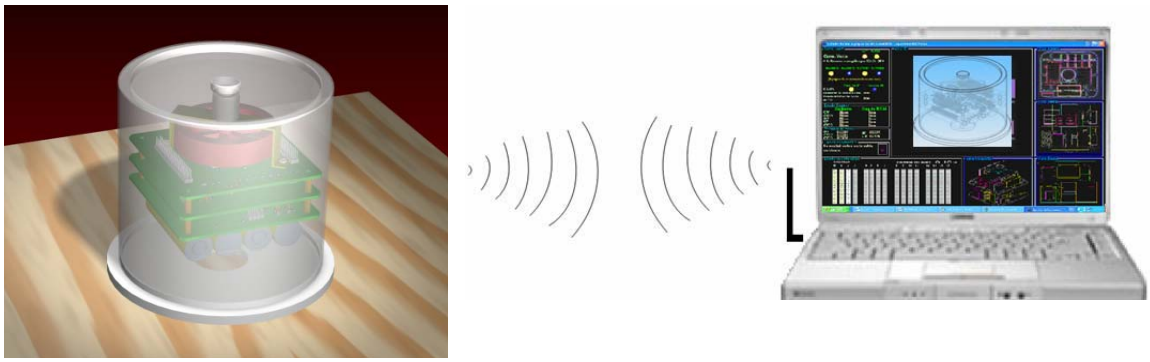


Figura 1.7. Arquitectura del SATERHUM

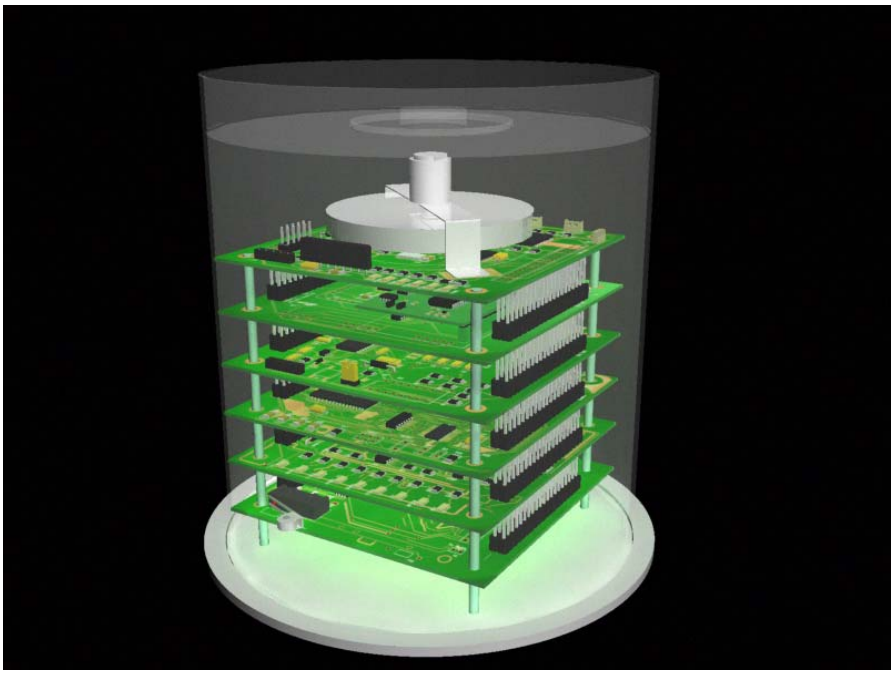


Figura 1.8. Modelo de SATEDU hecho con herramientas de realidad virtual.

En contraparte, el software del Sistema se diseñó pensando en que sus usuarios inviertan la mayor cantidad de tiempo en el empleo efectivo de sus prestaciones y no en tratar de averiguar cómo funciona.

La economía y la utilidad del prototipo fue solucionada con ciertos aspectos como:

- El empleo de una carcasa de acrílico ampliamente accesible en el mercado popular, en este caso una torre contenedora de discos compactos.
- Elementos electrónicos comerciales de uso automotriz, de telefonía celular y de tipo comercial, la mayor parte de ellos de montaje superficial.
- Un diseño único en cada elemento del sistema en su conjunto.

Finalmente, el diseño del SATERHUM está conformado por un prototipo satelital y un software. El prototipo como ya se indicó, se compone de: un subsistema estructural, computadora de vuelo, subsistema de potencia, subsistema de comunicaciones, subsistema de

estabilización y el subsistema de sensores. Mientras que el software contiene los elementos suficientes para dar seguimiento y control a cada subsistema del SATEDU, tanto de forma independiente (por módulos) como del sistema en conjunto.

1.3.1 SUBSISTEMA ESTRUCTURAL

La estructura empleada en SATEDU es una torre para discos compactos, la que en principio se planea instalar en una estructura de tipo giróscopo[2], que permita el libre desplazamiento del prototipo satelital a través de los 3 ejes coordenados, con una fricción reducida aunque no despreciable.

La estructura permitirá fijar dentro de ella a los circuitos impresos que constituyen los subsistemas de SATEDU, dispuestas a manera de pila, las cuales serán atornilladas a la base de la torre de CD's por 4 tornillos largos instalados en las esquinas de las tarjetas. De esta manera, se tiene un diseño estable, que permite inspeccionar cada circuito impreso, debido a que la estructura es de material transparente, además de que el diseño permite que la cubierta de acrílico se pueda separar con sencillez del resto del hardware, aspecto que permite entonces la visualización directa de los detalles del prototipo.

Por otro lado, la estructura pensada para visualizar físicamente el movimiento real de SATEDU permitirá el desplazamiento en tres ejes coordenados, figura 1.9, sin embargo, la fricción y la fuerza gravitatoria en este caso, son inconvenientes para realizar una simulación efectiva. Sin embargo, para fines didácticos puede considerarse como una buena aproximación y además sumamente económica.

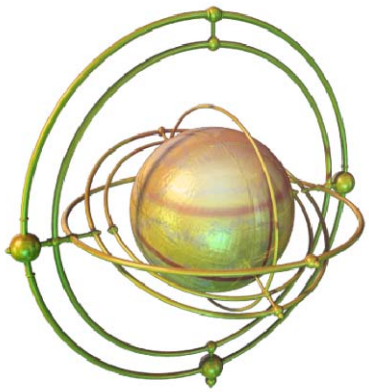


Figura 1.9. Estructura tipo giróscopo.

1.3.2 COMPUTADORA DE VUELO

El subsistema de computadora de vuelo se encarga de la gestión general de SATEDU, debido a que los comandos provenientes del software de estación terrena o de control y visualización, deben pasar por él para ser direccionado hacia el subsistema satelital final.

[2] giróscopo: dispositivo mecánico formado esencialmente por un cuerpo con simetría de rotación que gira alrededor de su eje de simetría.

El circuito impreso que alberga el subsistema en cuestión es de dos capas, dimensiones de 8.9cm x 8.9cm, con perforaciones en cada esquina para los tornillos que servirán como soporte para la estructura completa del Satélite Educativo, y con conectores en dos de sus lados para alojar todas las señales que requieren los subsistemas de SATEDU para su operación.

Para la computadora de vuelo que se presenta en la figura 1.10, se pensó desde sus fases de diseño que pudiera emplearse posteriormente y de forma directa en un proyecto satelital real. Para ello incorpora protecciones electrónicas heredadas del proyecto Satex para soportar niveles de radiación cósmica que se presentan en órbita baja (fenómeno latch-up'[3]). Si bien estas protecciones no tienen utilidad en un laboratorio, no generan ningún tipo de conflicto o costo adicional, debido a que los componentes electrónicos asociados no se instalan para la aplicación SATERHUM.

La computadora de vuelo contiene los siguientes elementos: un par de microcontroladores[4], uno de 16 bits (SAB80C166) que ejecuta las acciones de la computadora de vuelo y otro para apoyar la carga de nuevos programas al SAB; una memoria RAM para cargar los programas de operación de SATEDU; sensores de temperatura y arreglos de 'latch-up'; además de una memoria flash, para el almacenamiento de información permanente.

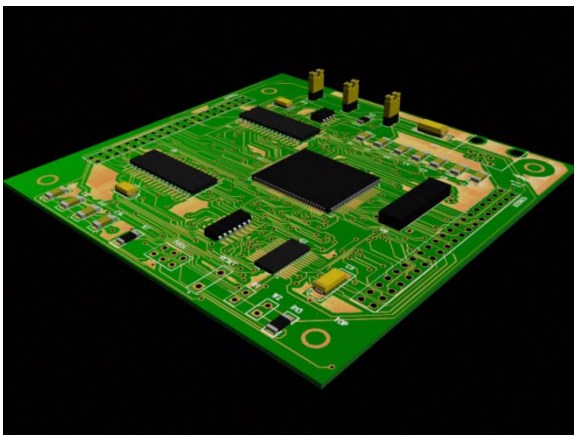


Figura 1.10. Modelo en realidad virtual de la Computadora de Vuelo.

1.3.3 SUBSISTEMA DE POTENCIA

El subsistema de potencia se subdivide en dos módulos que residen en circuitos impresos distintos, figura 1.11. Por un lado, un circuito impreso que contiene electrónica necesaria para controlar de forma inteligente a éste subsistema, así como medios de regulación y control de

[3] latch-up: fenómeno que consiste en la creación de corrientes parasitas, a través del semiconductor haciendo que este consuma mucha corriente hasta destruirse.

[4] microcontrolador: es un circuito integrado o chip que incluye en su interior las tres unidades funcionales de una computadora: CPU, Memoria y Unidades de E/S, es decir, se trata de un computador completo en un solo circuito integrado.

encendido de voltajes de alimentación para cada subsistema SATEDU, y por otro, un segundo circuito impreso que contiene una serie de baterías de Li+ con electrónica de carga de baterías ya sea desde celdas solares que se instalarán en el contenedor de acrílico o por medio de un cargador externo conectado a la línea externa de potencia.

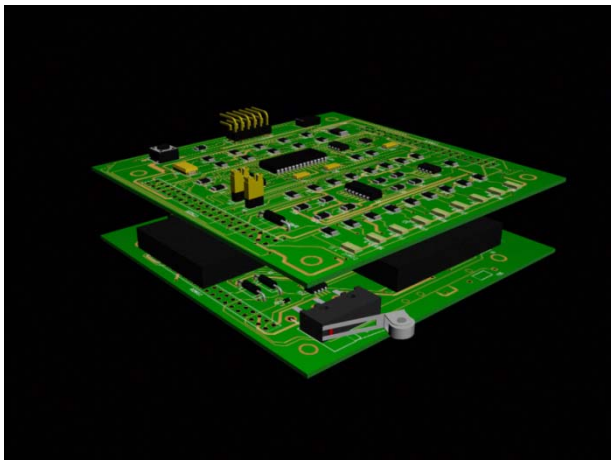


Figura 1.11. Modelo en realidad virtual del Subsistema de potencia.

Este subsistema fue diseñado para que pueda emplearse directamente en un picosatélite real. Para ello contiene circuitos de protección para el efecto 'latch-up', así como para asegurar su encendido al momento de la liberación del satélite en el espacio. Sin embargo, en el aspecto de las baterías sólo fueron consideradas ventajas económicas de compra y recarga, empleando 4 baterías de Li-On[5] recargables.

La lógica del subsistema de potencia la realiza un microcontrolador PIC18F2321, que mantiene comunicación con la Computadora de vuelo, para controlar los estados de apagado y encendido de las fuentes de energía de los subsistemas de SATEDU. Además, este subsistema contiene una serie de reguladores y un convertidor DC-CD de acuerdo con las necesidades de SATEDU.

1.3.4 SUBSISTEMA DE COMUNICACIONES

Las comunicaciones para el caso de SATEDU se diseñaron pensando sólo en el proyecto educativo, y no en la evolución hacia un satélite real. Por tanto, este subsistema permite que el Satélite Educativo sea capaz de operar por medios inalámbricos en un laboratorio, recibiendo instrucciones desde una computadora portátil también de manera inalámbrica.

Para ello, se optó por la utilización de transceptores[6] de radiofrecuencias – para un banda que es de libre utilización, la de 2.4 GHz –, uno a cada lado de la línea de comunicación; con interfaz serie-SPI del lado de SATEDU, y con interfaz USB-SPI del lado de la laptop.

[5] batería de Li-+: tipo de batería recargable en la que un ion de Litio se mueve entre el ánodo y el cátodo; es empleada comúnmente en aparatos electrónicos.

[6] transceptor: dispositivo que realiza, dentro de una misma caja o chasis, funciones tanto de trasmisión como de recepción, utilizando componentes de circuito comunes para ambas funciones.

De esta manera, el subsistema de comunicaciones, mostrado en la figura 1.12, se encarga de recibir y enviar los datos entre SATEDU y la computadora portable, sirviendo sólo de interlocutor entre el software de estación terrena y el satélite.
Para el proyecto SATERHUM se desarrollaron dos tarjetas de comunicaciones inalámbricas, una para uso en SATEDU y la otra para conectarse a la Laptop.

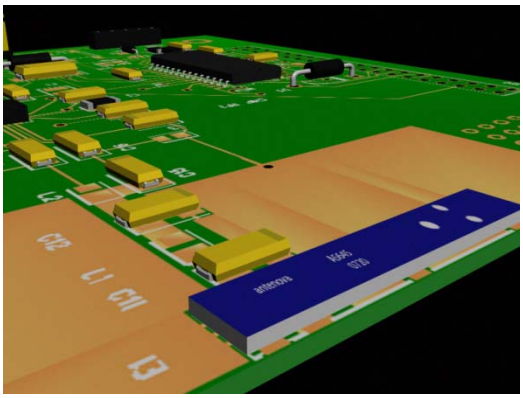


Figura 1.12. Modelo en realidad virtual del Subsistema de comunicaciones de SATEDU.

1.3.5 SUBSISTEMA DE ESTABILIZACIÓN

La estabilización es uno de los temas más cruciales y fundamentales en satélites previstos para trabajar en el espacio exterior, por lo que SATEDU cuenta con los elementos necesarios, para ilustrar a sus usuarios en este campo tecnológico.

El subsistema de estabilización de SATEDU emplea, mostrado dos métodos de estabilización activa, capaces de mover el prototipo satelital hacia un determinado punto de estabilización. El primero formado por una masa circular instalada en el eje de un motor de corriente directa, conocida como rueda inercial, figura 1.13; y el segundo, un juego de bobinas de torque magnético dispuestas a través de tres ejes coordenados.

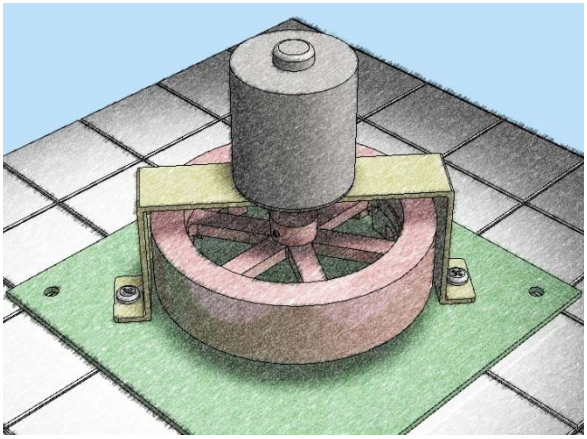


Figura 1.13. Bosquejo artístico del subsistema de estabilización por rueda inercial.

Con base en lo anterior será posible que los usuarios de SATEDU comprendan aspectos fundamentales del tema de estabilización satelital a través de la experimentación en condiciones terrestres.

1.3.6 SUBSISTEMA DE SENSORES DE ESTABILIZACIÓN

Al igual que los demás subsistemas que conforman al Satélite Educativo, el subsistema de sensores de estabilización que se muestra en la figura 1.14, reside en un circuito impreso de las medidas ya especificadas. Este subsistema permite leer los parámetros de movimiento y posición, es decir, de navegación inercial del prototipo satelital.

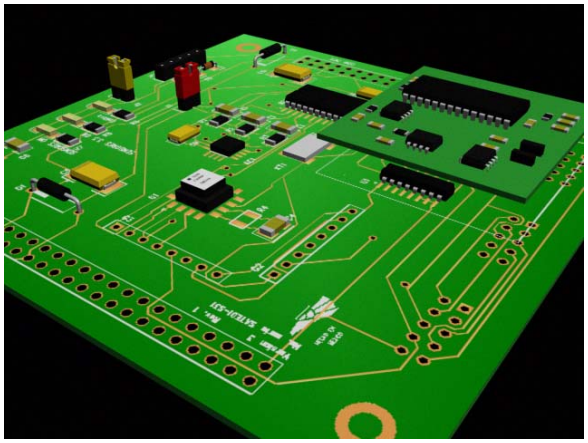


Figura 1.14. Modelo en realidad virtual del Subsistema de sensores.

Los sensores que contiene este subsistema, apoyarán al subsistema de estabilización así como al software de operaciones del SATERHUM. Los sensores que integra el subsistema son:

- Una brújula.
- Un acelerómetro triaxial.
- Un giróscopo triaxial.

Estos sensores permiten registrar el estado dinámico de SATEDU para identificar posteriormente puntos de estabilización del sistema.

1.3.7 SOFTWARE DE OPERACIONES Y VISUALIZACIÓN

Este software se instala y se ejecuta en una computadora personal y de preferencia en una laptop para interactuar con el prototipo satelital. Con el software que controla a SATEDU, se buscó obtener el mayor beneficio para el usuario del satélite educativo desde una aplicación de computadora. Se persiguió que el software tuviera la posibilidad de trabajar con algún subsistema en particular de SATEDU, o con SATEDU en conjunto.

El programa para operar y visualizar el satélite, emplea gráficos tridimensionales y realidad virtual para exponer en pantalla, los aspectos necesarios para comprender detalladamente el funcionamiento del Sistema de entrenamiento satelital, así como para interactuar en tiempo real con alguno o todos los subsistemas de SATEDU.

En la figura 1.15, se muestra la interfaz del software del SATERHUM, la cual se abordará en los capítulos posteriores de esta tesis.

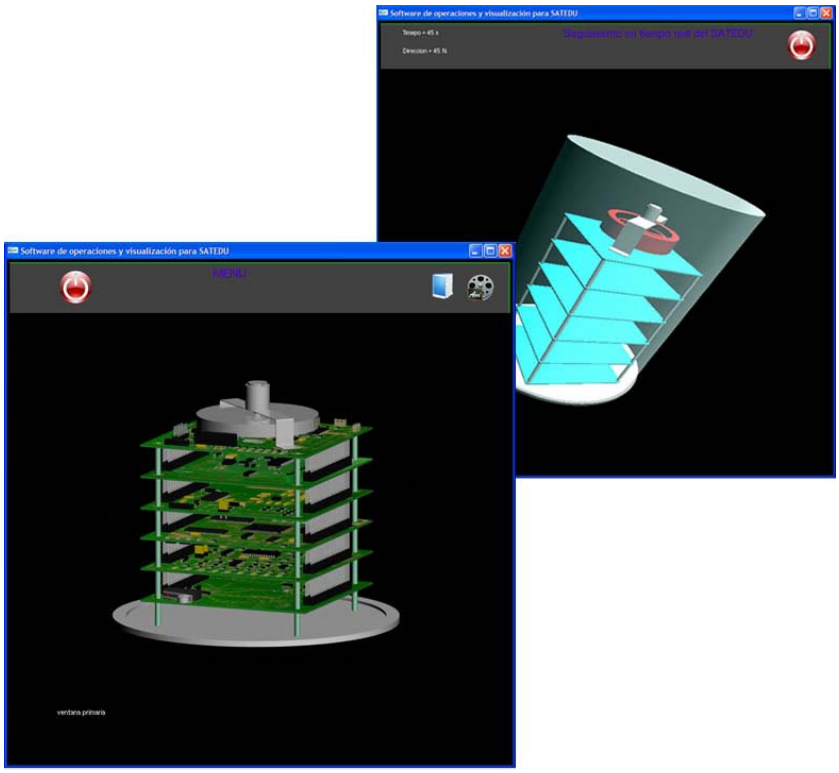


Figura 1.15. Software del SATERHUM.

1.4 LA NECESIDAD DE INCLUIR MEDIOS DE ESTABILIZACIÓN ACTIVA PARA EL SISTEMA DE ENTRENAMIENTO

El tema y el estudio de medios de estabilización es un tema muy importante en el terreno satelital, debido a que un satélite ya sea comercial, científico o milita, puede ofrecer sus servicios (comunicaciones o imágenes) en función de sus capacidades de apuntamiento o estabilización. Por tanto, si un satélite no puede permanecer en cierto punto con respecto a un determinado sistema coordenado, será imposible que realice actividades como: el estudio del entorno que rodea al satélite, recargar energía mediante la alineación correcta de los tableros solares y establecer comunicación con la base en Tierra.

1.5 LA REDUCCIÓN DE COSTOS COMO BASE DE DISEÑO GLOBAL

Tomando en cuenta el uso del SATERHUM como medio educativo, es preciso tener en cuenta el aspecto económico de su fabricación. Permitiendo que cualquier institución de educación pueda tener acceso a los beneficios de un sistema de apoyo para la instrucción de sus alumnos en el área satelital. Por otro lado, al tratarse de un proyecto que pretende evolucionar, resulta clave el cuidado del presupuesto en cada parte del proceso de desarrollo del mismo.

Es por ello, que se han elegido principalmente aquellas opciones que generen un diferencial positivo en cuanto a efectividad y bajo costo. Empleando materiales comerciales de uso convencional, tanto en el aspecto electrónico como en el estructural. Con diseños desarrollados

dentro del proyecto para cada subsistema y con la posibilidad de sustituir fácilmente sus elementos, así como de expandir el sistema en general.

CAPÍTULO 2. SOFTWARE DE OPERACIONES Y VISUALIZACIÓN PARA UN SATÉLITE EDUCATIVO, DISECCIÓN

La idea realizar un Sistema Satelital para Entrenamiento de Recursos humanos se enfocó en el desarrollo de un sistema integral que integra hardware y software de manufactura universitaria, en un campo lleno de retos como son los satélites artificiales.

Al hablar de un módulo de software como una de las dos grandes entidades del proyecto SATERHUM, se hace referencia implícita a una aplicación de computadora y no a la lógica de un subsistema físico. Sin que con esto exima la interacción que debe existir entre los programas ejecutados en cada una de las partes. Además, se debe subrayar que el software de control y visualización del proyecto de entrenamiento para recursos humanos, contiene un elemento destinado en parte a monitorear al prototipo SATEDU, cuya comunicación dependerá del buen entendimiento de programas tanto en SATEDU como en la computadora personal.

La mayoría del software ejecutable desde una computadora personal, requiere dos elementos constituyentes básicos, que cubrirán las necesidades de los usuarios finales. Uno correspondiente a las operaciones para las que fue creado el programa y, una interfaz de usuario, a través de la cual se interactúe con dichas operaciones de la mejor manera posible.

El software de control y visualización del SATERHUM, fue concebido para constituir una herramienta didáctica que permita realizar experimentos con la réplica de satélite, el cual representa a la estación terrena que permite la supervisión y el control de satélites comerciales. Por tanto, sus operaciones fundamentales estarán destinadas a monitorear y controlar al Satélite Educativo, reproduciendo de la forma más cercana las operaciones reales de una estación terrestre de control y monitoreo satelital. Este software establece una interfaz que permite la interacción entre usuario y SATEDU, y también constituye propiamente parte del material de experimentación y aprendizaje del sistema completo.

De este modo, el software que emula a una estación terrestre se plantea como una interfaz de fácil manejo y comprensión, cuya manipulación resulte intuitiva, en aras de fomentar un aprendizaje eficaz y efectivo en los usuarios. Así, el software de estación terrestre (SET) de SATEDU requirió la integración de diversos módulos que pretenden generar mejores resultados tanto en el terreno operacional como en el educativo.

2.1 ANIMACIÓN TRIDIMENSIONAL INTERACTIVA

En la actualidad una de las aplicaciones de computadora que más llaman la atención, es el de las animaciones basadas en realidad virtual. En donde trata de simularse un entorno real en el que nos movemos, con todos sus elementos asociados. Estos sistemas pueden encontrarse en ámbitos muy variados, desde videojuegos (en los que la interacción resulta tan intuitiva, que el rango de edades de los usuarios es irrestricto) hasta películas y simulaciones (en donde el nivel

de detalle visual, llega a veces, a ser un obstáculo para discernir entre el mundo virtual y el mundo real que nos rodea), figura 2.1.

Figura 2.1. Modelado de realidad virtual con 3DS MAX.

El software del SATERHUM emplea a lo largo de toda la aplicación y como parte de su diseño a elementos de realidad virtual, con el objeto de alcanzar mejores resultados de visualización. Así, se crearon dos modelos distintos del Satélite Educativo. El primero permite que la aplicación se beneficie del rápido procesamiento de ejecución en la computadora, lo cual resulta imprescindible para monitorear en tiempo real el movimiento físico que experimente SATEDU. En el segundo, se elaboraron con todo detalle, cada una de las piezas que conforman al prototipo satelital. Este último modelo permitirá que los usuarios interactúen con SATEDU en un ambiente virtual, conociendo, comprendiendo y manipulando sus diferentes aspectos físicos.

El modelo que se realizó para el seguimiento del satélite, figura 2.2, muestra una apariencia simple, presentando sólo un bosquejo representativo con los elementos básicos, pretendiendo dar tan solo una idea del modelo real. En este caso, la premisa de trabajo se basó en que el tiempo de respuesta de la animación fuese rápido para que el modelo virtual logre emular sin problemas los movimientos de SATEDU, tratando de evitar en lo posible, desfasamientos de movimiento entre las partes.

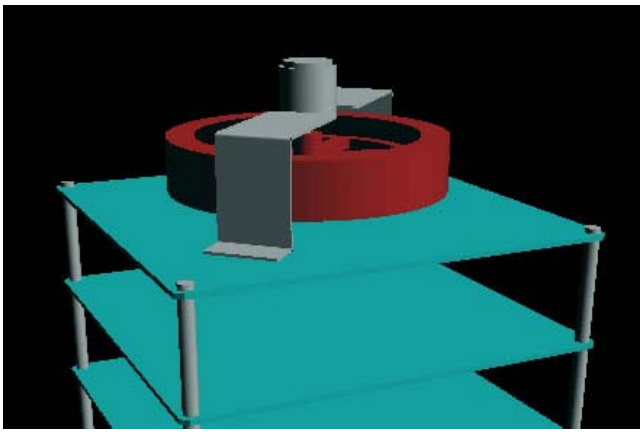


Figura 2.2. Modelo virtual de SATEDU sin detalles.

Para el segundo modelo virtual se diseñó una réplica prácticamente exacta del Satélite Educativo, cuidando cada uno de los detalles que lo componen visualmente. La idea fue lograr un elemento lo suficientemente preciso, para servir directamente como material exploratorio y didáctico, figura 2.3.

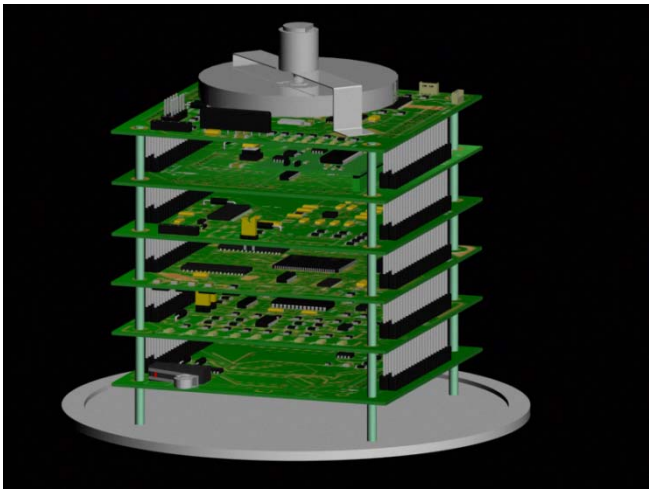


Figura 2.3. Modelo virtual detallado de SATDEDU.

Con el modelo detallado el usuario podrá escudriñar a placer todo el satélite, a través de una especie de paseo en primera persona cuya manipulación dependería de sus propias decisiones. Bajo este modelo se pueden observar incluso, aspectos tan minuciosos como los identificadores de cada uno de los componentes, tal y como se encuentran plasmados en los componentes reales. Partiendo de la misma exploración, se pensó en crear un video dividido en pequeños apartados, en el que se explicara en un ambiente virtual de tres dimensiones cada parte del dispositivo. La explicación se daría en términos de: su justificación dentro del satélite, sus posibles sustitutos y otros aspectos técnicos. El video puede ser visualizado integralmente como una sola entidad, o bien, desde cada uno de sus apartados accesibles desde la interfaz de usuario de la aplicación final.

2.2 INTERFAZ BASADA EN GRÁFICOS POR COMPUTADORA

El tono predominantemente didáctico que permeó las bases de diseño del software del SATERHUM, precisó el planteamiento de una interfaz de usuario. Que más allá de ser amigable y funcional, representa un escaparate inefable para mostrar los conceptos fundamentales del área satelital; desde su conformación básica, hasta sus principales aspectos operacionales.

En su evolución, el mundo de la computación gráfica ha desarrollado no sólo mejores y variados métodos para la realización de animaciones de realidad virtual, sino que además, ha puesto atención al aspecto de las acciones que los usuarios pueden llevar a cabo sobre dichas

animaciones. Por tanto, contando ya con un modelo virtual de SATEDU y con las herramientas necesarias para su libre manipulación, se pensó en desarrollar una interfaz basada en el modelo detallado. Esto con el objeto de que el manejo del software final fuera lo más intuitivo posible, es decir, que no se requirieran vastos conocimientos por parte del usuario, y que de igual forma no se requiriera gran conocimiento ni en el manejo del programa, ni en lo que a satélites respecta.

La interfaz de usuario para el control y la visualización del Satélite Educativo, requería solventar aspectos operacionales, tanto hacia el prototipo satelital como hacia el programa mismo, figura 2.4. Para lograrlo, se optó por emplear dos ventanas principales, sobre las que se desarrollarían todas las acciones. Una para la exploración y el reconocimiento de los componentes del satélite, y otra, para el monitoreo de sus operaciones. La interacción dentro de ambas ventanas funcionaría de manera sencilla. En la primera ventana manipulando a placer con el teclado el proceso de exploración, y con el ratón el control de opciones. En tanto que en la segunda se visualiza el movimiento en tiempo real de SATEDU, pudiendo modificar opciones con el ratón.

La interacción fue solventada en buena medida, como en la mayoría de las aplicaciones, por medio de menús de opciones, que en este caso fueron de dos tipos, dependiendo de la función que fueran a desempeñar. Por una parte, se implementó el uso de un menú que sólo aparece cuando el usuario lo requiere, adaptando sus funciones de acuerdo con la situación que se presente y con el objetivo de explorar fácilmente el modelo detallado del satélite. Luego, el otro tipo de menú es permanente, por lo que muestra en todo momento aquellas opciones que por su trascendencia deben permanecer visibles.

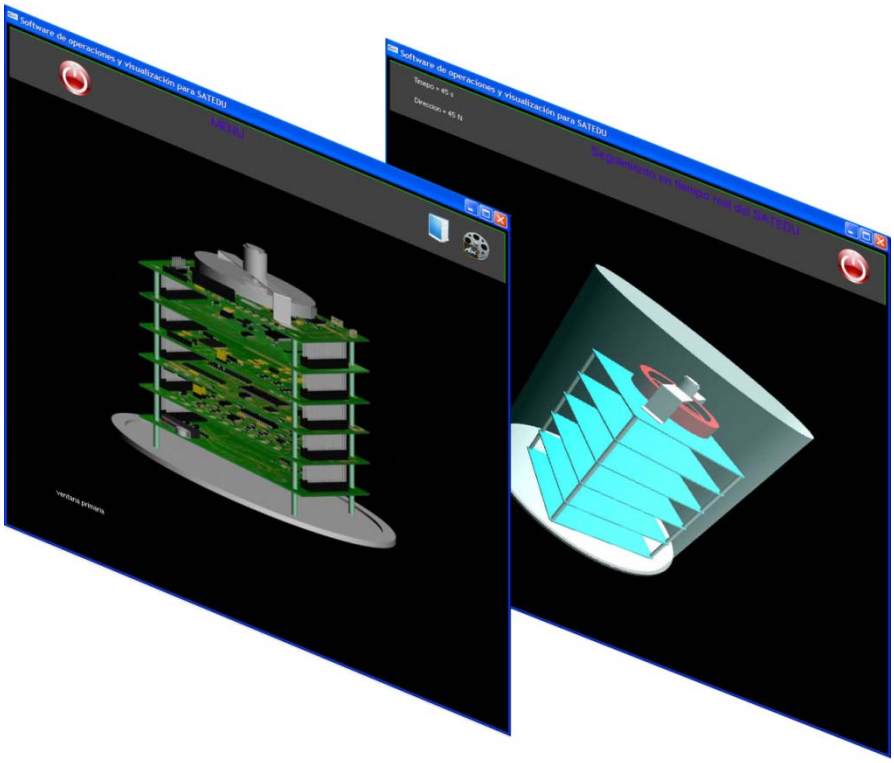


Figura 2.4. Software de visualización y operaciones del SATERHUM.

2.3 MONITOREO EFECTIVO

Uno de los objetivos fundamentales en el funcionamiento del proyecto SATERHUM es el monitoreo virtual efectivo del prototipo satelital. Al lograr este objetivo se obtiene un aspecto funcional único en su tipo y que incluso no tienen los sistemas de seguimiento satelital comercial que existen en el mercado. Por otro lado, se tenía planteado un seguimiento distinto a la simple toma y presentación de cifras o datos recibidos por telemetría. Por ello, resultó precisa la incorporación de un conjunto de módulos de diversa índole, que hicieron posible los resultados esperados.

Para monitorear cierta característica presentada en un dispositivo conectado a una computadora, es menester tener un determinado medio de comunicación, con su respectivo protocolo y programa receptor de datos. En el caso del sistema de entrenamiento, se empleó una comunicación serial, favoreciendo la velocidad de transmisión y la sencillez en la programación de la aplicación receptora, que por cierto, se encarga además de la interpretación de los datos recibidos.

En lo referente al software de SATERHUM, el programa encargado de la recepción e interpretación de la información proveniente del satélite (concerniente a movimientos de rotación) debía presentar los datos interpretados como una réplica instantánea del movimiento experimentado por el modelo virtual de SATEDU. Entonces, era necesario realizar dos procesos de manera paralela para no sufrir desfasamientos entre los cambios del modelo real y el virtual. El primero encargado de la comunicación serial entre SATEDU y SET, en tanto que el segundo se encarga de reproducir la animación tridimensional.

Si bien la actualidad ha traído consigo la popularidad del concepto de multi-procesamiento, debido en buena medida a la proliferación en el mercado de procesadores de dos o cuatro núcleos, capaces de realizar varios procesos a la vez (uno por cada núcleo); no es todavía, ni por mucho, un estándar en el hardware de uso popular. Sin embargo, algún tiempo tiene ya, el concepto de multi-hilos, el cual presenta una compatibilidad bastante extendida y brinda similares resultados (con las debidas reservas) al multi-procesamiento.

Considerando lo anterior, se decidió emplear la técnica de multi-hilos para manejar los dos procesos, de comunicaciones y de gráficos por computadora, de manera paralela. Cada proceso ejecutándose bajo un hilo diferente y compartiendo la información necesaria para un seguimiento efectivo. De un lado, un hilo con el proceso referido a las comunicaciones y del otro, un segundo hilo encargado del procesamiento de la animación del modelo virtual de SATEDU.

2.4 RECURSOS QUE INSTRUYEN

Todo proceso educativo lleva consigo el necesario uso de recursos que apoyan la correcta comprensión de los temas en cuestión; desde el establecimiento de las bases de conocimiento hasta la explicación exhaustiva de los puntos considerados como medulares.

Continuando con el empleo de los gráficos por computadora, y luego de tener modelado una réplica en realidad virtual del Satélite Educativo, pudo plantearse el desarrollo de una película de animación, figura 2.5. Esta explica con detalle todas las partes que componen al satélite

mediante modelos tridimensionales de las piezas del prototipo satelital. A su vez explica los fundamentos operacionales propios del sistema SATERHUM y los datos de justificación ya sea para un módulo en especial, o para la totalidad del sistema.

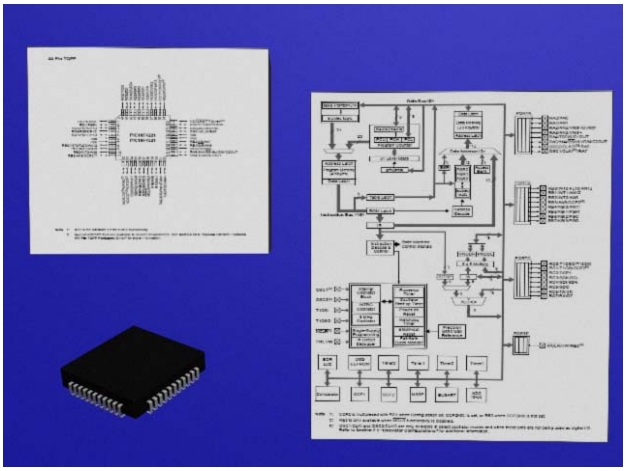


Figura 2.5. Parte de las explicaciones de la película de animación del SATERHUM.

La computación gráfica al día de hoy, ofrece entre sus herramientas algunas que permiten alcanzar modelos de realidad virtual. Cuyo grado de detalle resulta verdaderamente importante, sin un manejo de bajo nivel de los elementos que definen a los gráficos por computadora. Los resultados en cualquier situación, dependerán de por lo menos un par de elementos que apoyen la tarea completa del modelado. Por un lado, la herramienta en donde puedan crearse los modelos, con sus colores, texturas e iluminación asociados, figura 2.6; y por otro, la herramienta que dote a la escena de todos sus modelos asociados (acabado visual requerido) actividad que por cierto, recibe el nombre de renderizado.

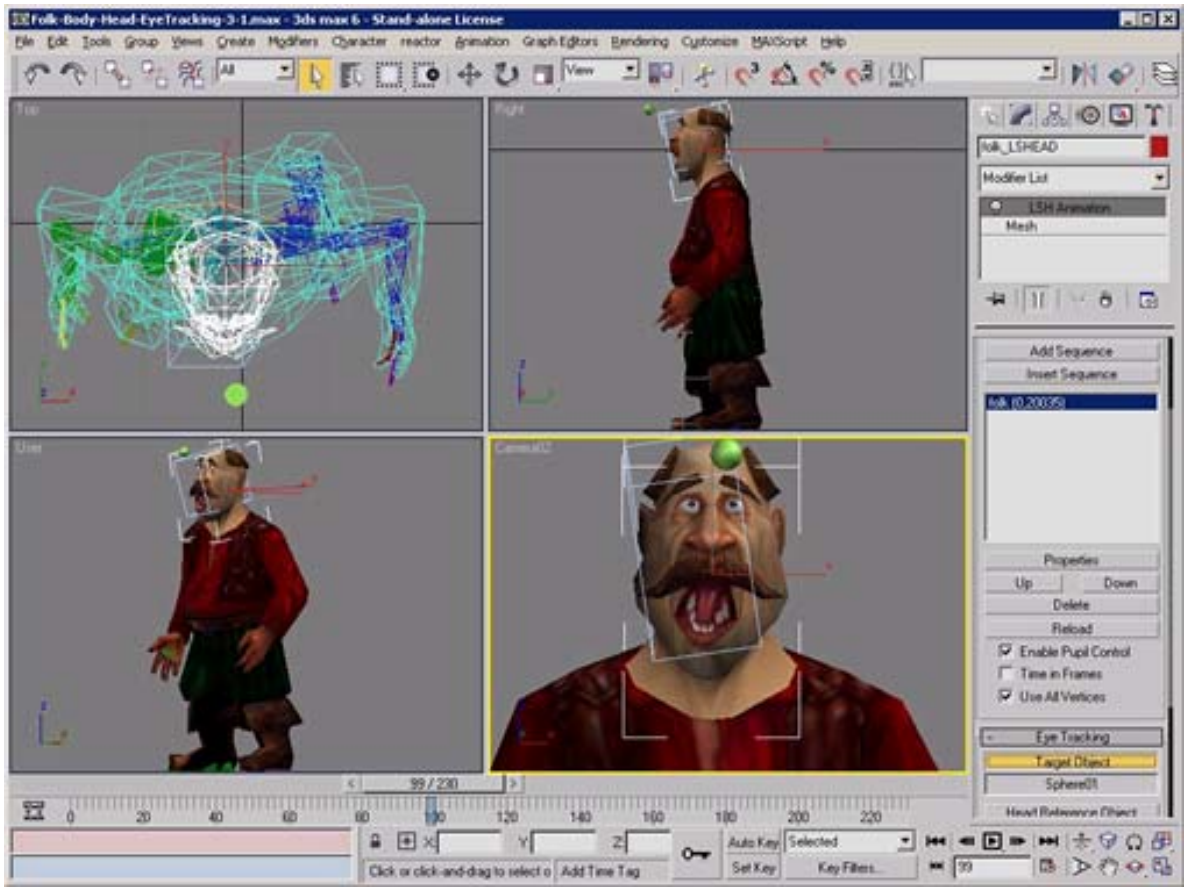


Figura 2.6. Entorno de desarrollo Autodesk 3DS MAX.

Si bien la parte más laboriosa de los desarrollos de realidad virtual, lo constituyen los trabajos de modelado y puesta a punto de los aspectos visuales; regularmente constituye sólo la primera fase de un proceso, en el que el producto final requerido es la animación de lo antes modelado. Así, las herramientas que se emplean para el desarrollo de los modelos mencionados, proveen un apartado especial para la animación de escenas. Basándose en cualquier caso, en la idea básica de la animación tradicional – empleada en dibujos animados, por ejemplo –, en donde la animación se concibe como una seguidilla de cuadros dibujados, con cambios pequeños de uno a otro, en donde sea necesario. En este sentido, las herramientas actuales que ayudan a la creación de animaciones, son capaces de crear interpolación entre cuadros, disminuyendo con ello el número de cuadros por dibujar.

Finalmente, luego de seguir el procedimiento mencionado, para el caso del modelo virtual del Satélite Educativo se culminó una película de animación con todos los detalles del proyecto. Película que de hecho, fue lograda desde la misma herramienta de modelado y animación, la cual provee esta prestación con dos opciones generales. Una, en la que se genera un 'previo', de película de formato popular, en donde puede apreciarse la animación completa, pero sin haber cubierto el renderizado. Y una segunda, empleada en la película para SATERHUM, figura 2.7; en la que el renderizado se realiza sobre cada uno de los cuadros de la animación, para luego crear la película, logrando muy buenos resultados visuales.

Figura 2.7. Parte de la película del proyecto SATERHUM.

2.5 EL SIGUIENTE PASO

Si bien la presente tesis tiene como tema, el desarrollo del software de operaciones y visualización para un Satélite Educativo, es importante mencionar, en este punto, que restan por terminar ciertas partes del mismo, lo que de hecho, fue previsto desde que se comenzó el trabajo de tesis.

La justificación de lo antes previsto, se centra en el hecho de que aún hay algunos aspectos del prototipo satelital, que no han sido terminados del todo, lo que ha obligado, hasta este momento, a marcar ciertas fronteras en el desarrollo del software de aplicación del proyecto SATERHUM. Sin embargo, los límites que fueron planteados, se hicieron desde la previsión de alcanzar un punto de desarrollo tal, que resten sólo adaptaciones de lo ya realizado, para la finalización total del software. Así por ejemplo, en lo concerniente al módulo encargado de las operaciones; se completaron satisfactoriamente, las funciones para dar seguimiento al movimiento del Satélite Educativo; las cuales pueden hacerse extensivas hacia un completo control del mismo dispositivo.

CAPÍTULO 3. MODELADO EN REALIDAD VIRTUAL DEL SATEDU, DIFERENTES PLANTEAMIENTOS

Debido a la naturaleza del proyecto SATERHUM que pretende alcanzar un proyecto atractivo, versátil y lo suficientemente ilustrativo para formar y reafirmar a estudiantes y entusiastas en el área satelital, fue necesario diseñar un software que llamara poderosamente la atención de sus usuarios, respetando el aspecto pragmático de su funcionamiento.

Por otro lado, la realidad virtual en las últimas dos décadas ha experimentado cambios realmente vertiginosos. Captando personas, animales, objetos y fenómenos del entorno real, de manera tan fiel que a veces es imposible distinguirlos de los reales; además de permitir que casi cualquier persona pueda crear animaciones tridimensionales realistas, con equipos de cómputo convencionales, figura 3.1.

Figura 3.1. PC QBEX VX-2000 de la marca QBEX Electronics Corporation, fabricante y proveedor de soluciones y productos para computación personal y electrónica de consumo

En lo que respecta al software de estación terrena, SATERHUM propone ocupar para la interfaz de usuario, modelos tridimensionales creados en computadora que representen de manera fiel el dispositivo SATEDU. De esta manera, quienes utilicen el Sistema para entrenamiento gozarán de un medio para manipular el Satélite Educativo por medio de su réplica tridimensional presentada en la pantalla de una laptop.

Además, el software del SATERHUM permite realizar las siguientes acciones:

- Seguimiento al estado del SATEDU.
- Indicarle al satélite que realice alguna tarea.
- Brindar información a los usuarios sobre el modo de operación de cada uno de los componentes del proyecto completo.

Para lo cual se adoptaron diversas formas de realizar el mismo modelo, de acuerdo a la función que se utilice.

En lo que se refiere al seguimiento del movimiento del Satélite Educativo, el software emplea un método que optimiza tiempos de respuesta y que permiten captar con bastante precisión los cambios de posición del dispositivo.

Por otro lado, para la exploración, manipulación y explicación de los elementos del SATEDU, se eligió un camino que permita mostrar el dispositivo tal y como se vería en la realidad.

3.0 UN POCO ANTES DEL MODELADO

Para hacer más comprensible el modelado del Satélite Educativo en computadora es importante abordar algunos conceptos básicos de la computación gráfica. La computación gráfica es el área de la computación que estudia los conceptos y algoritmos relacionados con la edición y producción de gráficos por computadora. Entendiendo como gráfico a elementos geométricos tridimensionales conjugados que buscan representar un hecho real.

Al hablar de producción de gráficos por computadora se hace referencia a un proceso que se realiza con apoyo de la computadora para mostrar en pantalla lo que se le haya especificado. Primero, se realiza la definición matemática del concepto por desplegar en pantalla, indicando los datos de las básicas geométricas que generarán el modelo deseado. Luego se lleva a cabo el paso del modelo matemático hacia el modelo geométrico, es decir, la abstracción en un espacio tridimensional del concepto definido. Este proceso recibe el nombre de renderización – término definido en el mismo seno de la computación gráfica, sin significado en el lenguaje convencional –. Y por último, el proceso de rasterización, figura 3.2, que significa la adaptación de la información de renderizado hacia el espacio bidimensional que representa el mapa de pixeles de la pantalla de la computadora.

De lo anterior, el único subproceso en el que influye directamente la mano del hombre, es el de la definición. Éste a su vez se compone de elementos como el establecimiento de los parámetros geométricos de dibujo, la manipulación de colores, la iluminación y las texturas del modelo en cuestión.

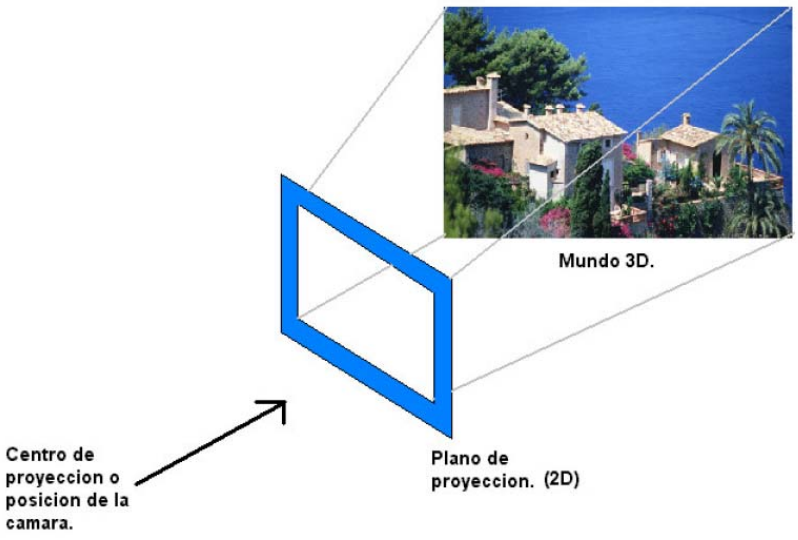


Figura 3.2. Modo en el que se reflejan en pantalla los gráficos por computadora.

3.1 MODELADO DEL SATEDU MEDIANTE OPENGL

Uno de los métodos empleados para realizar el modelo en realidad virtual de SATEDU es el que se utilizó para muestrearlo en tiempo real. En este caso se requirió contar con un modelo que no exigiera una gran capacidad de procesamiento, derivado de cálculos matemáticos que la computadora realiza durante la generación de figuras tridimensionales y de sus transformaciones asociadas.

Posteriormente se realizó este modelo mediante programación tradicional de computadoras, no desde bajo nivel, sino desde la perspectiva que permite producir cambios en el modelo durante el tiempo de ejecución y sin generar problemas de rendimiento. Es decir, se muestra sólo el cascarón del Satélite Educativo – carcasa, circuitos impresos sin componentes electrónicos, uniones entre las tarjetas y rueda inercial – dejando a un lado los detalles de cada subsistema, como se muestra en la figura 3.3.

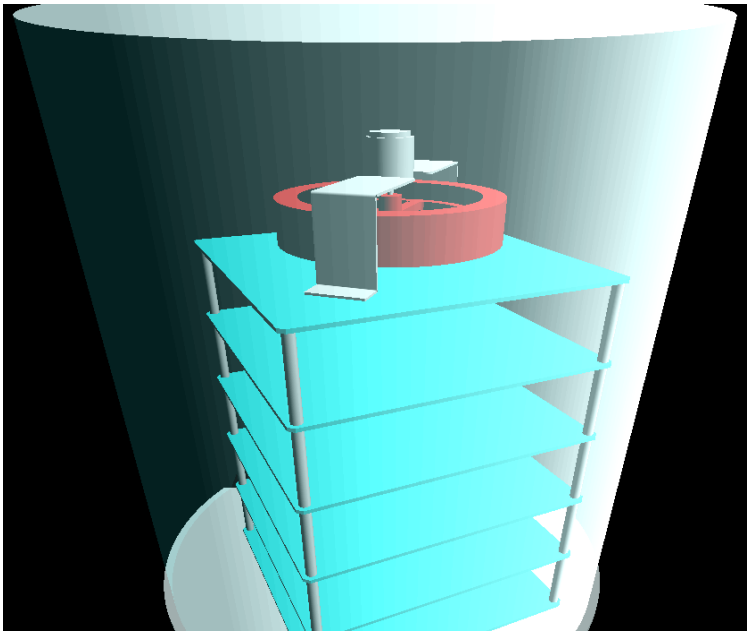


Figura 3.3. Modelo final de SATEDU para el módulo de seguimiento en tiempo real del software del SATERHUM.

En el terreno de Computación Gráfica se tienen actualmente dos librerías gráficas predominantes, DirectX y openGL. La primera se especializa en programación de videojuegos, mientras que OpenGL se utiliza como base en aplicaciones de CAD/CAM, figura 3.4, así como en la realización de modelos geométricos del entorno real. Es justamente esta última librería la que fue adoptada para el tratamiento de la animación en tiempo real del Satélite Educativo.

Figura 3.4. Software de CAD ejecutado en una PC.

Una librería gráfica como OpenGL contiene un conjunto de funciones de programación empaquetadas, que dotan a sus usuarios de elementos necesarios para dibujar figuras geométricas en la pantalla. Las funciones definen además la manera en la que se presentarán las figuras en cuanto a su: posición, ubicación, tamaño, composición, color, textura, iluminación, movimiento y modo de visualización.

Las funciones de una librería de este tipo deben utilizarse desde una aplicación creada con un lenguaje de programación compatible para contar con un entorno capaz de generar la aplicación, además de que maneje ventanas y los eventos que se realicen sobre ellas. Sin implicar con esto que se modificará la estructura de la librería con el lenguaje, sino que se adaptará de acuerdo con las circunstancias.

Debe señalarse que el acompañante más popular de OpenGL es C++, debido en buena medida a que es un lenguaje muy versátil, que ofrece amplios recursos de programación. Permitiendo que el programador vaya desde el alto hasta el bajo nivel y brindándole la mayor parte del control de la aplicación con una sola herramienta. En cuestión del control, tanto C++ como OpenGL buscan el mismo objetivo en su propio ámbito, pudiendo complementarse en aplicaciones gráficas.

Con lo anterior se justifica el que la programación del software de estación terrena del SATERHUM se haya realizado en C++.

3.1.1 DIBUJO A PARTIR DE VÉRTICES

El modelo geométrico de SATEDU como cualquier diseño realizado por el hombre, tiene como base estructural a figuras tridimensionales regulares, o bien, una composición de las mismas. Estas figuras a su vez pueden subdividirse en polígonos primero y más adelante en los vértices constitutivos de dichos polígonos. Es decir, mientras que en la naturaleza la mayor parte de los elementos son figuras irregulares, en aquellos de patente humana predominan figuras geométricas regulares o combinaciones de ellas.

Por otro lado, OpenGL ofrece algunos caminos para el modelado de cuerpos geométricos, siendo el más básico el que se logra dibujando la figura a partir de sus vértices. Este método es además el que representa mayor libertad para definir completamente la figura desde su conformación. Lo anterior es importante si se toma en cuenta que tanto OpenGL como el manejador de ventanas que comúnmente se le asocia, GLUT, proveen cuerpos volumétricos ya predefinidos, en los que sólo se necesitan definir sus dimensiones.

Los vértices que se indiquen en el dibujo de modelo de OpenGL deben corresponder a aquellos que forman parte de los polígonos constituyentes del mismo, y no a los que compongan a la figura. Lo anterior se debe a aspectos de iluminación, tema que será tratado más adelante. Por ahora basta partir de la idea de que los vértices deben componer polígonos básicos, los cuales a su vez generarán la figura deseada. Por ello se hace necesaria la creación de modelos matemáticos de elementos geométricos que cumplan con tales características.

OpenGL plantea una manera simple de definir un polígono, figura 3.5; predeterminando sus vértices, tal y como se muestra a continuación.

```
glBegin(GL_POLYGON);
glVertex2f(0.0, 0.0);
glVertex2f(0.0, 3.0};
glVertex2f(4.0, 3.0);
glVertex2f(6.0, 1.5);
glVertex2f(4.0, 0.0};
glEnd();
```

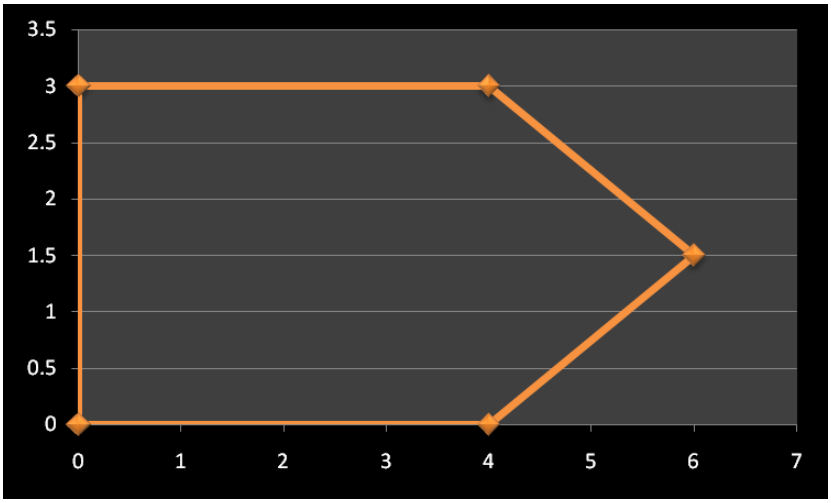


Figura 3.5. Ubicación de los vértices en una gráfica bidimensional.

Como puede verse en las anteriores declaraciones, se establece un polígono empleando el argumento GL_POLYGON[7] de glBegin(), cuyos vértices se definen con la sintaxis

[7] GL_POLYGON forma parte de los posibles argumentos de glBegin(), entre los que además se encuentran: GL_POINTS (puntos individuales), GL_LINES (pares de vértices interpretadas como segmentos de línea individuales), GL_LINE_STRIP (series de segmentos de línea conectados), GL_LINE_LOOP (igual que el anterior, pero con un segmento entre el final y el principio), GL_TRIANGLES (tripletas de vértices interpretada como triángulos), GL_QUADS (cuartetos de vértices interpretados como cuadrados).

glVertex'*(). Dicha sintaxis sustituye en ['], la cantidad de argumentos que podrá contener la función y en [*] el formato que deberán tener esos mismos argumentos, en cuyo caso, los más empleados son los números flotantes[8](f) o los apuntadores[9] a un arreglo[10] de flotantes(fv).

Se debe notar además que los delimitadores de la figura antes dibujada, son glBegin() y glEnd(), en donde glBegin() tendrá como argumento el tipo de figura por dibujar.

3.1.2 CREACIÓN DE FIGURAS TRIDIMENSIONALES

Como ya se indicó, los sólidos volumétricos necesarios para la construcción de un determinado modelo deben conformarse a partir de polígonos. Sin embargo, a pesar de que OpenGL presenta entre los posibles argumentos de glBegin() a: polígonos (GL_POLYGON), cuadrados (GL_QUADS), y triángulos (GL_TRIANGLES). Es necesario elegir los triángulos para conseguir resultados óptimos al momento de añadir iluminación a la escena[11] en donde aparezca el modelo final, figura 3.6 y figura 3.7.

Por estas razones fue necesario contar con las definiciones trigonométricas que generen las figuras tridimensionales básicas, para conformar el modelo deseado del SATEDU a partir de la unión de polígonos triangulares.

Algunos de los sólidos volumétricos que se obtuvieron, como los básicos para la composición del modelo geométrico completo del Satélite Educativo, se presentan en el siguiente código, en donde se bosqueja además, su forma de obtención.

```
....
float origen = 0.0;

float dim_x = 1.2;
float dim_y = 1.2;
float dim_z = 0.09;
float dim_esq = 0.1;

                /*****SE DEFINEN LOS VÉRTICES DE LOS TRIÁNGULOS QUE FORMARÁN LA FIGURA (UN
                                                PARALELEPÍEDO)**********/
static GLfloat tarjeta[8][3] = {
        {dim_esq, origen, origen},{(dim_x-dim_esq),origen,origen}, {dim_x,dim_esq,origen},
        {dim_x,(dim_y-dim_esq),origen},{(dim_x-dim_esq),dim_y,origen}, {dim_esq,dim_y,origen},      {origen,(dim_y-
dim_esq),origen}, {origen,dim_esq,origen} };

static GLfloat tarjeta_1[8][3] = {
        {dim_esq, origen, dim_z}, {(dim_x-dim_esq),origen,dim_z}, {dim_x,dim_esq,dim_z},
        {dim_x,(dim_y-dim_esq),dim_z}, {(dim_x-dim_esq),dim_y,dim_z}, {dim_esq,dim_y,dim_z},
        {origen,(dim_y-dim_esq),dim_z}, {origen,dim_esq,dim_z} };
```

[8] Número flotante: denominación que en programación se le da a los números con cifras decimales, los cuales presentan una precisión menor a los de doble precisión.

[9] Un apuntador es una *variable de programación* que contiene la dirección en memoria de otra variable. Se pueden tener apuntadores a cualquier tipo de variable.

[10] Un arreglo es un conjunto de variables de programación del mismo tipo, que pueden ser referenciadas a través de un mismo nombre

[11] En computación gráfica se le llama escena a todo el entorno en donde se desarrolla una determinada animación.

```
.....
/************SE REALIZA LA RUTINA PARA DIBUJAR LA FIGURA************/
for(int i=0;i<=7;i++)
            {
                        j += 2;

                        if(i==7)
                        {


/***********EN ESTE CASO SE UTILIZA LINE_LOOP PARA HACER EVIDENTES LOS TRIÁNGULOS*****/
              glBegin(GL_LINE_LOOP);
                                    glVertex3fv(&tarjeta[i][0]);
                          glVertex3fv(&tarjeta[0][0]);
                          glVertex3fv(&tarjeta_1[i][0]);
                          glEnd();

                                    glBegin(GL_LINE_LOOP);
                                    glVertex3fv(&tarjeta_1[i][0]);
                          glVertex3fv(&tarjeta[0][0]);
                          glVertex3fv(&tarjeta_1[0][0]);
                          glEnd();
.......
```

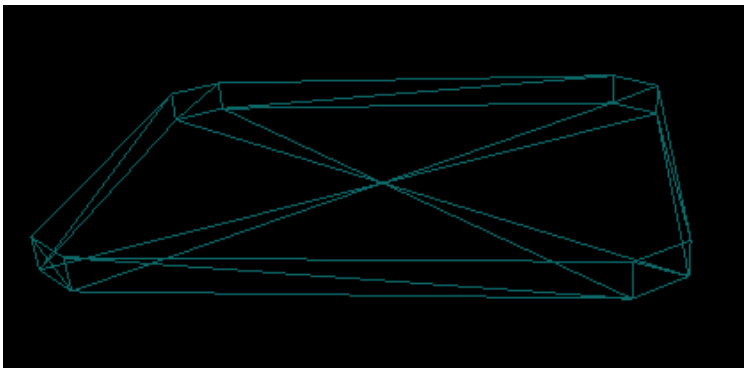


Figura 3.6. Paralelepípedo dibujado a partir de triángulos.

```
....
/***SE CALCULAN LOS VERTICES PARA CONFORMAR LOS TRIÁNGULOS UE HABRÁN DE FORMAR LA
FIGURA**/
while(ang_temporal<=360)
 {

  cilindro1[i][0] = cilindro1_1[i][0] = radioCil_1 * (cos(ang_temporal*(3.141592654/180)));
  cilindro1[i][1] = cilindro1_1[i][1] = radioCil_1 * (sin(ang_temporal*(3.141592654/180)));
                cilindro1[i][2] = 0;
                cilindro1_1[i][2] = altoCil_1;
               ang_temporal += gr_interval;
               i++;

 if((ang_temporal>360)&&((ang_temporal-gr_interval)!=360))
{
cilindro1[i][0] = cilindro1_1[i][0] = radioCil_1 * (cos(360*(3.141592654/180)));
cilindro1[i][1] = cilindro1_1[i][1] = radioCil_1 * (sin(360*(3.141592654/180)));
cilindro1[i][2] = 0;
cilindro1_1[i][2] = altoCil_1;
j=1;
}
 }
 tot_puntos=i-1+j;
.......
/***SE DIBUJA LA FIGURA***/
for(int i=0;i<=tot_puntos;i++)
            {

                        if(i==tot_puntos)
```

```
                {
                        glBegin(GL_LINE_LOOP);
                        glVertex3fv(&cilindro1[i][0]);
                        glVertex3fv(&cilindro1[0][0]);
                        glVertex3fv(&cilindro1_1[0][0]);
                        glVertex3fv(&cilindro1_1[i][0]);
                        glEnd();

                }
                else
                {
                        glBegin(GL_LINE_LOOP);
                        glVertex3fv(&cilindro1[i][0]);
                        glVertex3fv(&cilindro1[i+1][0]);
                        glVertex3fv(&cilindro1_1[i+1][0]);
                        glVertex3fv(&cilindro1_1[i][0]);
                        glEnd();
                }
        }
…….
```

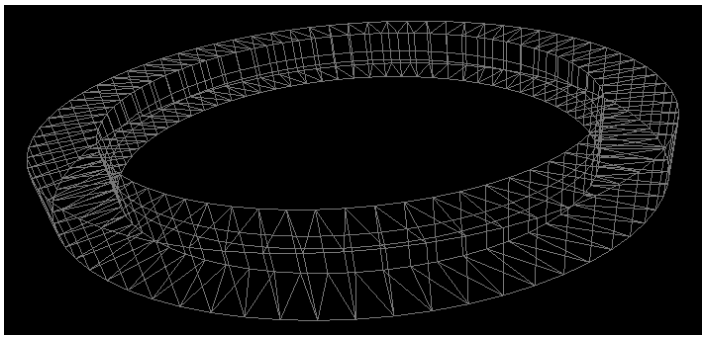


Figura 3.7. Cilindro con hueco en medio a partir de triángulos

3.1.3 COMPOSICIÓN DE MODELOS GEOMÉTRICOS COMPLEJOS A PARTIR DE FIGURAS Y MODELOS MÁS SENCILLOS

Una vez que se generaron los volúmenes básicos necesarios, se realizaron dos pasos para finalizar el dibujo adecuado. Primero se confeccionaron aquellas figuras de construcción más elaborada, partiendo del principio explicado – modelado matemático del volumen requerido, producido a partir de triángulos–. Segundo, se efectuó la composición final del modelo de SATEDU de manera incremental, hasta ensamblar cada una de sus partes, figura 3.8 y figura 3.9.

La primera fase, representó la realización de modelos matemáticos con un grado de complejidad ligeramente mayor al empleado para las básicas. Las que se concibieron pensando en la forma en la que se generaron las primeras figuras.

```
.......

/**** DEFINICIÓN DE LAS ASPAS ******/

ang_temporal=0;
 while(ang_temporal<=360)
 {
                if((ang_temporal>=0)&&(ang_temporal<=90))
                {
                        aspa_red[i][0] = aspa_red_1[i][0] = (aspa[3][0]) + (radioAspa*(cos(ang_temporal*(3.141592654/180))));
                 aspa_red[i][1] = aspa_red_1[i][1] =  (aspa[3][1] - radioAspa) + (radioAspa*(sin(ang_temporal*(3.141592654/180))));
                 aspa_red[i][2] = cilindro3[0][2];
                 aspa_red_1[i][2] = cilindro3_1[0][2];
```

```
/**asignacion para tapa**/
                tapa_aspa_1d[k][0] = tapa_aspa_2d[k][0] = aspa_red[i][0];
                tapa_aspa_1d[k][1] = tapa_aspa_2d[k][1] = aspa_red[i][1];
                tapa_aspa_1d[k][2] = aspa_red[i][2];
                tapa_aspa_2d[k][2] = aspa_red_1[i][2];
                k++;

                /**termina asignacion para tapa**/

               ang_temporal += gr_interval;
               i++;
               }

.....          .

 /*****termina aspa redondeada*******/
......
/***********EMPIEZA DIBUJO DE FIGURA COMPUESTA (RUEDA)*************/
glPushMatrix();
               DibujaAspa();
               glPopMatrix();
               glPushMatrix();
               glRotatef(55.0,0.0,0.0,1.0);
               DibujaAspa();
......
glPushMatrix();
DibujaCilindroHueco();  //FUNCIÓN DE CILINDRO CON UN HUECO EN MEDIO
glPopMatrix();
```

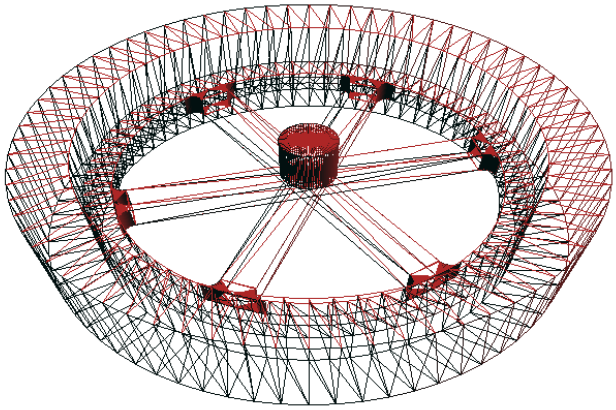


Figura 3.8. Figura compuesta (correspondiente a la rueda inercial de SATEDU).

```
......
/**********EMPIEZA CÁLCULO DE PIEZA EN FORMA DE Z*****************/

/*****empieza cálculo de cuadro redondeado DERECHO**********************/

/***SE FORMA UN CUADRO REDONDEADO BUSCANDO LAS COORDENADAS DE LAS ESQUINAS, PARA LUEGO
UNIRLAS; ES COMO CALCULAR 4 PARTES SEPARADAS DE UN CÍRCULO****/

int i=0;

ang_temporal=0;
 while(ang_temporal<=360)
 {

               if((ang_temporal>=0)&&(ang_temporal<=90))
               {
                        coord4[i][0] = coord4_1[i][0] = (ancho-radio) + (radio*(cos(ang_temporal*(3.141592654/180))));
              coord4[i][1] = coord4_1[i][1] =  (alto-radio) + (radio*(sin(ang_temporal*(3.141592654/180))));
```

```
		coord4[i][2] = 0;
		coord4_1[i][2] = radio1;

		ang_temporal += gr_interval;
		i++;
		}
		else if((ang_temporal>90)&&(ang_temporal<180))
		{
			coord4[i][0] = radio1_1;
			coord4_1[i][0] = 0;

		coord4[i][1] = coord4_1[i][1] = alto ;
		coord4[i][2] = 0;
		coord4_1[i][2] = radio1;

		ang_temporal =180;
		i++;
		}

.....

/******SE HACE LA UNIÓN ENTRE EL CUADRO DE ARRIBA Y EL DE EN MEDIO, CON UNA ESPECIE DE CODO,
CUYA CURVATURA SE REALIZA SOBRE EL PLANO X-Z******/

/***empieza arco union cuadro redondeado ARRIBA***/
 i=0;
ang_temporal=90;
while(ang_temporal<=180)
{
coord5[i][0] = coord5_1[i][0] = (radio1*(cos(ang_temporal*(3.141592654/180))));
		coord5[i][1] = 0;
		coord5_1[i][1] = 1;
		coord5[i][2] = coord5_1[i][2] = (radio1*(sin(ang_temporal*(3.141592654/180)))) ;


		ang_temporal += gr_interval;
		i++;
}
//tot_puntos5=i-1+j;
/********************/

.....

/******DIBUJO DE PIEZA EN FORMA DE Z**************/

/*****POR UN LADO SE DIBUJAN LAS PAREDES DE LA PIEZA, SEGÚN LOS CÁLCULOS ANTES REALIZADOS**/

/***dibuja PAREDES cuadro redondeado DERECHA 3d*****/

		for(i=0;i<=tot_puntos4;i++)
		{

			.....

			else
			{
				glBegin(GL_LINE_LOOP);
				glNormal3fv(&n_coord4[j][0]);
				glVertex3fv(&coord4[i][0]);
				glVertex3fv(&coord4[i+1][0]);
				glVertex3fv(&coord4_1[i][0]);
				glEnd();

				glBegin(GL_LINE_LOOP);
				glNormal3fv(&n_coord4[j+1][0]);
				glVertex3fv(&coord4[i+1][0]);
				glVertex3fv(&coord4_1[i+1][0]);
				glVertex3fv(&coord4_1[i][0]);
				glEnd();

			}
			j += 2;
		}
```

```
/***termina cuadro redondeado 3d*****/

......

/****LUEGO SE DIBUJAN LAS TAPAS *****/

/***dibuja tapa arriba cuadro redondeado 3d ***/

        for(int i=0;i<=tot_puntos4;i++)
        {

                if(i==tot_puntos4)
                {
                        glBegin(GL_LINE_LOOP);
                        glNormal3fv(&n_coord4_t2[i][0]);
                        glVertex3fv(&coord4_1[i][0]);
                        glVertex3fv(&coord4_1[0][0]);
                        glVertex3fv(&centroD_1[0][0]);
                        glEnd();
                }
                else
                {
                        ......
                }
        }

......
```

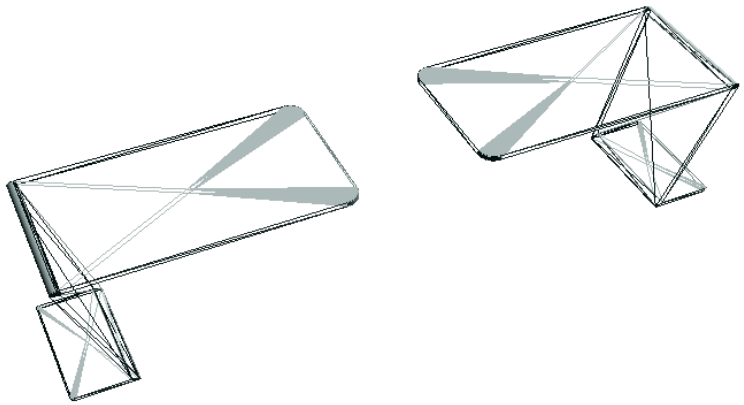


Figura 3.9. Figura compuesta, correspondiente al soporte de la rueda inercial.

El segundo paso correspondiente al ensamble de cada uno de los volúmenes geométricos del modelo del Satélite Educativo requirió la aplicación de una serie de transformaciones, las cuales forman parte de la ayuda de una librería gráfica por naturaleza. Dichas transformaciones no son otras que rotación, traslación y escalamiento, figura 3.10, las que después de aplicarlas conjuntamente llevan a generar el modelo geométrico uniforme que se buscaba desde un inicio.

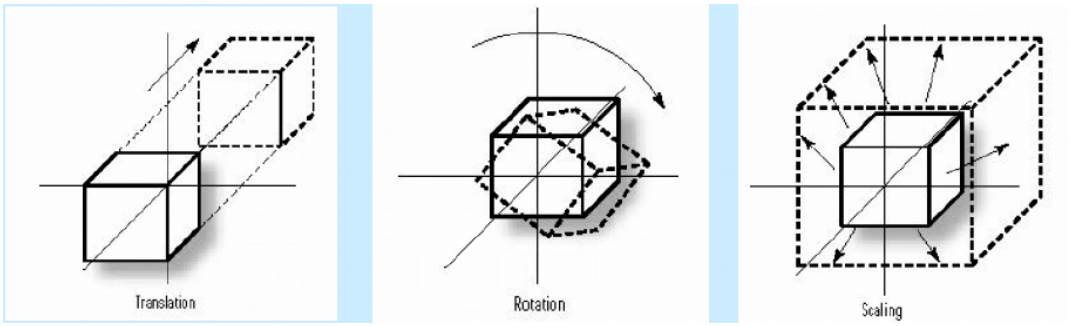


Figura 3.10. Transformaciones geométricas que forman parte de la librería OpenGL.

La manera en la que OpenGL resuelve las transformaciones se relaciona directamente con la teoría del algebra lineal, acerca de matrices de transformación[12]. A pesar de ello, no es necesario conocer dicha teoría para manejar correctamente las transformaciones, debido a que esta librería gráfica proporciona funciones sencillas para lograr los cambios requeridos en los objetos.

OpenGL emplea el concepto de pila[13] de matrices, figura 3.11, para las transformaciones de proyección[14] y las de modelos de vista; siendo ésta última en la que se generarán los cambios que incidirán directamente sobre los objetos dibujados. De manera natural, las funciones de transformación de modelo de vista, se realizan sobre la cima de la pila de forma acumulativa, figura 3.12; siendo dicha cima, la matriz que se aplicará al dibujo cuya función matemática se encuentre inmediatamente después.

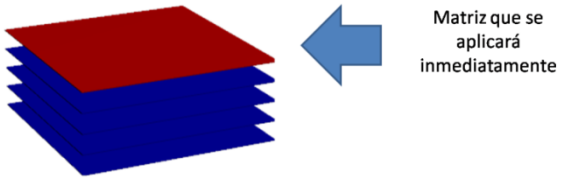


Figura 3.11. Pila de matrices de transformación.

[12] Con el apoyo de matrices y pensando que una matriz puede verse como una asociación de vectores de n dimensiones, se realizan operaciones de rotación, traslación y escalamiento.

[13] Una pila es una estructura de datos de tipo LIFO (del inglés Last In First Out, último en entrar, primero en salir) que permite almacenar y recuperar datos. Para el manejo de datos cuenta con dos operaciones básicas: apilar (push), que coloca un objeto en la parte alta de la pila, y su operación inversa, retirar o desapilar (pop), que retira el último elemento apilado.

[14] Transformaciones de proyección: se refieren a la manera en la que serán proyectados los objetos de la escena en el plano de proyección que representa la pantalla. La proyección empleada para simulaciones del mundo real se llama Perspectiva, en la cual, se presentan las proporciones de los objetos de acuerdo tanto a su tamaño, como a su distancia hacia los demás, y con la cámara que visualiza toda la escena.

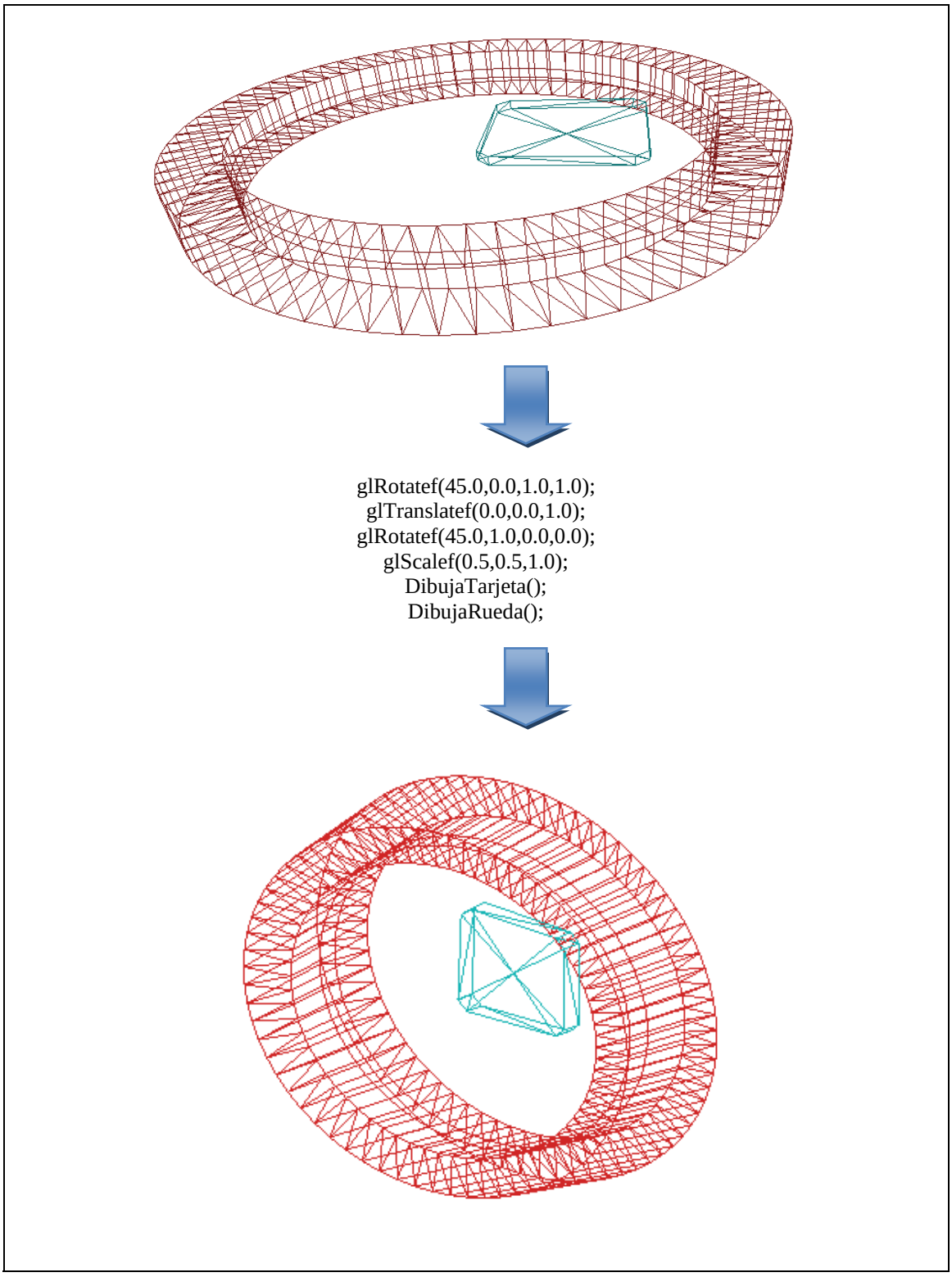


Figura 3.12. Transformaciones que se van acumulando en la matriz del tope de la pila,
para luego aplicarsele a los modelos geométricos;
el cambio de tonalidad se debe a la iluminación, tema que se explicará más adelante.

Como consecuencia de lo anteriormente planteado, se presenta un problema evidente, producto de la necesidad de generar modelos cuyos elementos requieran ser o no independientes, según corresponda. En este tenor, la utilización efectiva de la pila mencionada, se convierte en asunto indispensable, mas no complicado. Así, para aquellas transformaciones que deban aplicarse a una generalidad de elementos, tendrán que ser introducidas en la pila; para que en principio se apliquen aquellas transformaciones que sólo deban presentarse en uno o varios elementos particulares. Posteriormente se debe extraer la primera matriz de transformación de la pila, para regresar al punto donde se realizaron las transformaciones más generales, figura 3.13.

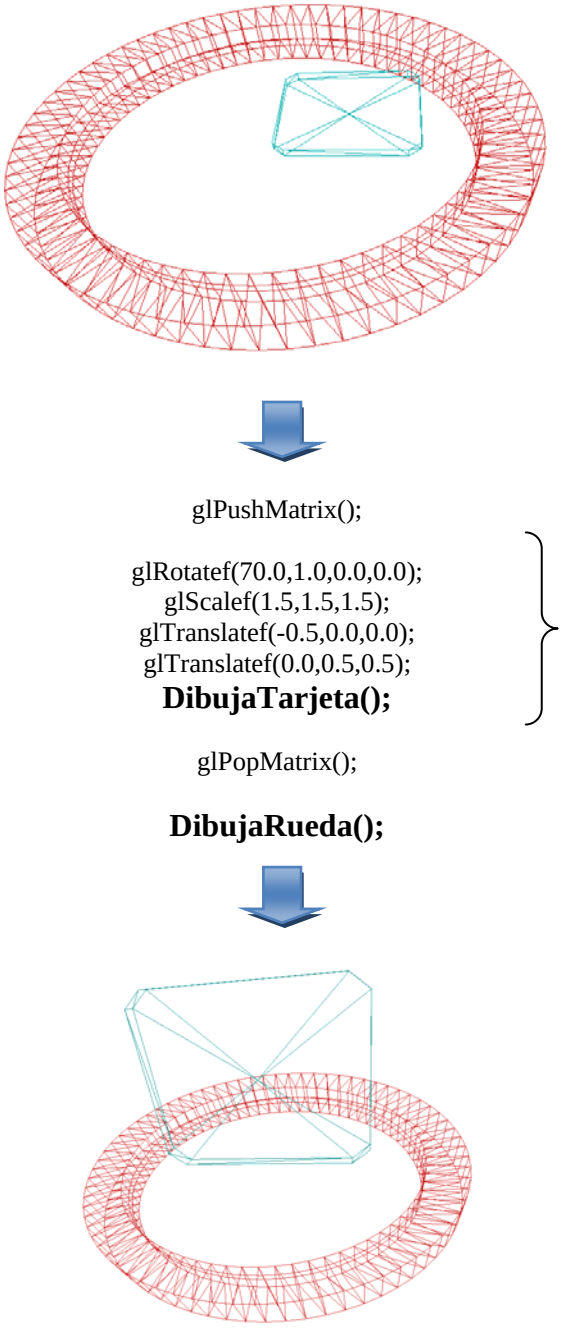


Figura 3.13. Aplicación de transformaciones con el empleo de las funciones de 'push' y 'pop' de la pila de modelo de vista de OpenGL.

De esta forma, el modelo geométrico final de la figura de SATEDU se logró con base en los conceptos planteados hasta aquí, quedando éste como lo muestra la figura 3.14.

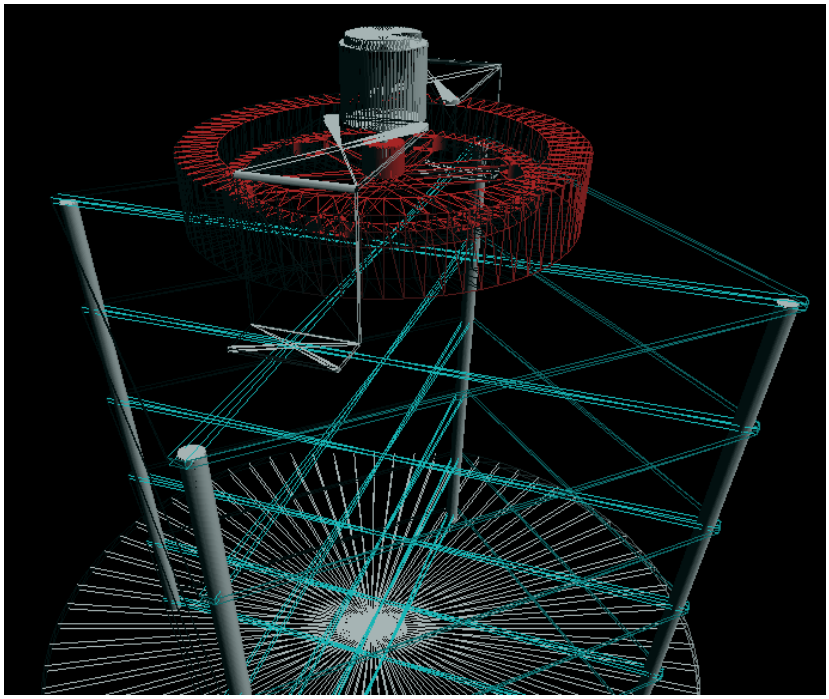


Figura 3.14. Modelo de SATEDU visualizando los triángulos que lo conforman.

3.1.4 DETALLES DEL MODELO TRIDIMENSIONAL DEL SATEDU, TRABAJANDO EN OPENGL

Durante el proceso de realización de modelos de computadora que representan elementos existentes en la realidad, la determinación de características geométricas se ve afectada por aquellos factores que mejoran su realismo. Tales factores pueden sintetizarse en aspectos de color, texturas, iluminación y materiales.

Para el modelo realizado con OpenGL se omitió el rubro correspondiente a las texturas de sus elementos, con la intención de sólo exponer una simulación de SATEDU, brindando con ello una idea general de su presentación real y exenta de detalles. Con ello se evitaron posibles dificultades en el tiempo de respuesta de la animación en tiempo real.

Por otro lado, los factores mínimos necesarios para tener en la pantalla de la computadora a un elemento que simule la realidad con el apoyo de OpenGL son el color y la iluminación. Los cuales guardan una estrecha relación entre sí, además del concepto de propiedades de los materiales.

La librería gráfica empleada en esta tesis maneja el color, la iluminación y los materiales. Por otro lado, cada una de estas características presentan una dependencia estrecha. Primero, cualquier objeto tridimensional que solo se defina solo con color, no permitirá distinguir los trazos de su estructura a lo largo de tres diferentes ejes coordenados; mostrando una figura

simplemente plana. Segundo, la forma en la que determinado modelo aparezca en la pantalla de la computadora, dependerá de las características de los elementos de iluminación de la escena en la cual se encuentre, aunado a las propiedades que se definan para su o sus materiales; constituyendo una mezcla entre la manera en la que incide la luz sobre los objetos y la forma en la que es reflejada por sus materiales.

Por ello, el modelado en realidad virtual de SATEDU se realizó pensando en que tanto la iluminación de la escena, como las características de los materiales empleados en los elementos del mismo, permitieran una correcta visualización del modelo.

Por ello, el modo en el que el espectador visualizará un objeto dependerá en la forma en que la luz le incide y la forma en la que el objeto refleja la luz. Tal concepto, no requiere mayor explicación, si se piensa en un ambiente real que carezca completamente de cualquier fuente de iluminación, llámese un foco, luz solar, etc., no será posible distinguir ningún objeto que se encuentre dentro del campo de vista. Adicionalmente, la visualización dependerá también de la reflexión de la luz y de los materiales de los mismos. Un objeto con un material opaco reflejará uniformemente la luz desde cualquier ángulo, en tanto que otro cuyo material sea por ejemplo metal brillante, presentará reflejos que concentrarán la luz en ciertas partes dependiendo del ángulo de incidencia de la luz, figura 3.15.

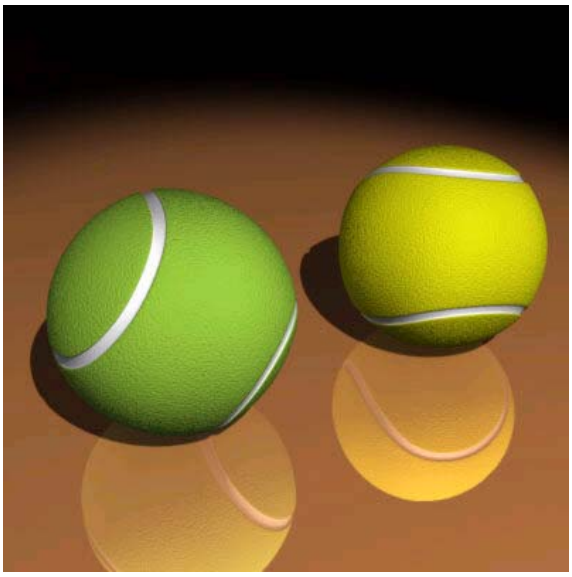


Figura 3.15. Imagen en la que se percibe un material opaco (en las pelotas) y por otro lado, un material que refleja la luz (la base sobre la que están las pelotas).

En el renglón de iluminación, OpenGL obliga la compaginación entre el modelo geométrico puro y sus características visuales. Esto se debe a que el cálculo de los ángulos de incidencia y reflexión de la luz, se realizan a partir de los vectores[15] normales[16] del modelo antes mencionado, figura 3.16. Es decir, OpenGL procesará el ángulo que haya entre las normales

[15] Vector: un vector geométrico es un segmento de una cierta magnitud, al cual se le asignan propiedades adicionales como la dirección y sentido.

[16] Vector normal: regularmente conocido como “normal” a una superficie, es un vector perpendicular a dicha superficie.

marcadas para el modelo geométrico y la luz que incide sobre ellas. Por lo anterior, resulta menester definir un vector normal por cada pequeña porción del modelo, que presente un ángulo con respecto al espacio tridimensional, de igual dimensión, sin importar la manera en la que pueda moverse en un momento dado. Así, cualquier modelo que se realice en OpenGL y que requiera de iluminación, deberá plantearse a partir de una función matemática que lo divida en porciones mínimas, sobre las que se calculará cada vector normal, figura 3.16. Si bien se vislumbra la necesidad de emplear definiciones de modelos geométricos a partir de sus vértices, resulta en muchos casos confusa o sin precisión, la labor de diseccionar hasta los vértices el modelo; por lo que se plantea la utilización de polígonos y de manera específica, triángulos.

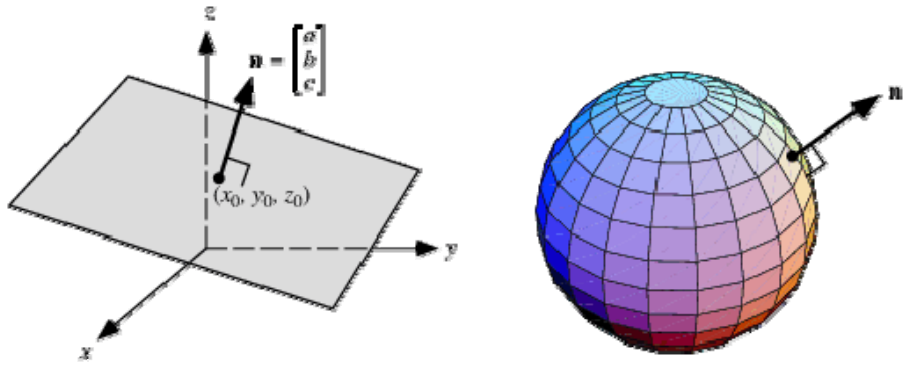


Figura 3.16. Vector normal.

Los polígonos más básicos de la geometría están representados por triángulos y cuadrados, siendo estos últimos en los que regularmente se piensa al pretender realizar un modelo geométrico. Esto se debe a que el cuadrado es quizás la figura que se presenta en la mayor parte de los objetos que percibimos en nuestra vida diaria; ejemplos de esto pueden ser, libros, paredes, puertas, aceras, etc. Sin embargo, para el caso del modelado geométrico por computadora, en donde se requieren las normales del modelo en cuestión para lograr una reflexión correcta de la luz, los cuadrados presentan un problema significativo, al ser objeto que bajo ciertas transformaciones rotacionales que colocan a sus vértices fuera de un mismo plano producen que el mismo polígono posea distintos vectores normales para cada uno de sus vértices, figura 3.17. En contra parte, los triángulos – cualquier tipo de ellos – poseen la propiedad de residir siempre sobre un mismo plano, sin importar las transformaciones que experimenten; convirtiéndose así, en la solución idónea para modelos cuyas normales sean de cálculo sencillo, figura 3.18.

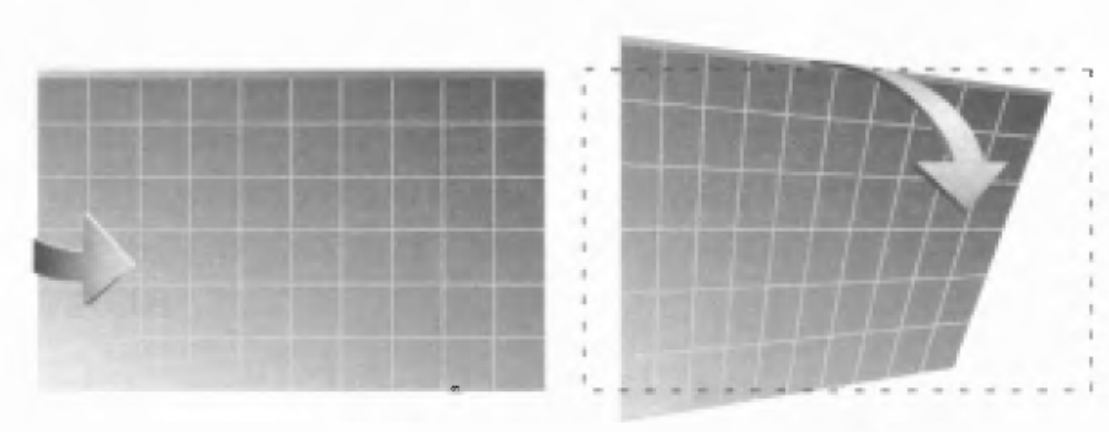


Figura 3.17. Vértices de un cuadrado que residen sobre distintos planos luego de una transformación.

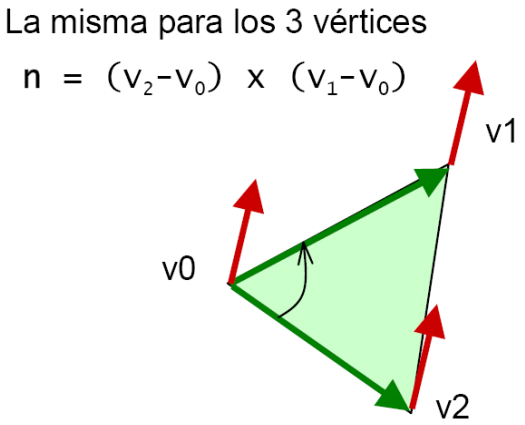


Figura 3.18. Vectores normales de un triángulo

Con base en lo planteado y partiendo del hecho de que el modelo geométrico de SATEDU fue pensado desde un inicio como una composición de polígonos triangulares; el establecimiento de la iluminación del mismo se redujo a sólo dos aspectos clave. El pertinente cálculo de las normales de cada uno de los triángulos que conforman el modelo total. Además del establecimiento de las características de iluminación de la escena, así como de los materiales del modelo mismo.

Como se había mencionado, el modelo que resultaba adecuado para este punto, carece de atractivo en cuanto a niveles altos de detalle. Pues define una iluminación versátil y simple para la escena en su conjunto, además de que cuenta con las mismas características de materiales para los elementos del modelo, Por ello brinda solo una visión adecuada de la simulación final. Sin embargo, fue menester realizar cambios para el modelo de la carcasa, debido al material transparente que presenta el dispositivo físico.

El material transparente se obtiene bajando la opacidad que se le ha asignado al elemento en cuestión, sin pasar por alto la deshabilitación del testeo de profundidad que regularmente se indica en OpenGL para que no requiera procesar aquellas detalles que no vayan a aparecer en

pantalla, figura 3.19. Así, en este punto queda de manifiesto uno de los principios básicos de la graficación por computadora; trabajar sólo sobre los detalles que se verán en pantalla.

```
.....

GLfloat mat_transparent[] = { 0.0, 0.8, 0.8, 0.6 };

.....

glMaterialfv(GL_FRONT, GL_DIFFUSE, mat_transparent);
glEnable(GL_BLEND);
glDepthMask(GL_FALSE);
glBlendFunc(GL_SRC_ALPHA, GL_ONE);
DibujaCilindro(5.3,15.0);
glDepthMask(GL_TRUE);
glDisable(GL_BLEND);
```

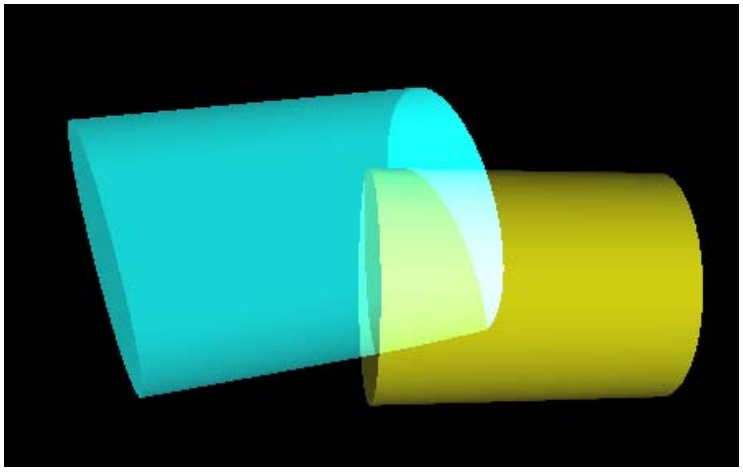


Figura 3.19. Dos cilindros; uno con material transparente (azul), otro con material opaco (amarillo)

3.2 MODELADO DEL SATEDU A DETALLE

Hoy en día, el número de herramientas que tiene el software para crear modelos tridimensionales de elementos reales es realmente extensa. Cada herramienta resulta más o menos efectiva, o inútil, de acuerdo con el objetivo que busquemos, debido a su propia naturaleza. Por otro lado, en el camino de la evolución del software, la computación gráfica ha sido un pasajero protagonista, beneficiándose del incremento en planteamientos y métodos para conseguir determinados resultados. Así, mientras otrora el tema de generar animaciones por computadora, era restrictivo de procedimientos de bajo nivel, el momento actual brinda posibilidades versátiles y de gran poder para todo tipo de necesidades, capaces de interactuar unas con otras para llegar a un objetivo definido. De esta manera, la elección de la herramienta idónea para cada actividad, se ha convertido en uno de los puntos trascendentales en el desarrollo de cualquier aplicación de software.

El segundo camino que había de seguirse para el modelado por computadora de SATEDU, ya no debía ocuparse del aspecto del rendimiento en lo que a tiempos de respuesta se refiere, sino concentrar los esfuerzos en lograr un modelo claramente fiel al original. Por consiguiente, fue necesario usar aplicaciones de computadora que apoyaran aquellos renglones en donde OpenGL, se convierte en extremo complicado y extenuante por su naturaleza misma, regresando a dicha librería para retomar la libre interacción necesaria.

Tomando como punto de partida a las características físicas de SATEDU, como un elemento preponderantemente electrónico y mecánico; resultó natural poner atención en alguna aplicación de computadora que facilitara los resultados requeridos, contando además, con posibilidades de manipulación externa de sus producciones. Entonces, durante el proceso de generación de una representación fiel de la realidad, se dispuso como primer paso, el trabajo con un programa especializado en diseño de piezas de manufactura humana, como es el caso de Solid Edge.

Solid Edge, figura 3.20, es una aplicación de computadora cuyo ámbito se concentra en el terreno del CAD/CAM, posibilitando la creación y manipulación de modelos de piezas de corte mecánico y electrónico a través de una interfaz completamente gráfica e intuitiva. Con este software el escenario principal se centró en el cuidado de cada uno de los detalles de las piezas.

Con este software se trabajó por un lado, con modelos de piezas ya existentes en los recursos mismos de la citada aplicación, o desde librerías de elementos prediseñados con fuente en Internet; y por otro, en el diseño total de aquellas piezas faltantes cuyo desarrollo se facilita con las prestaciones del software CAD/CAM.

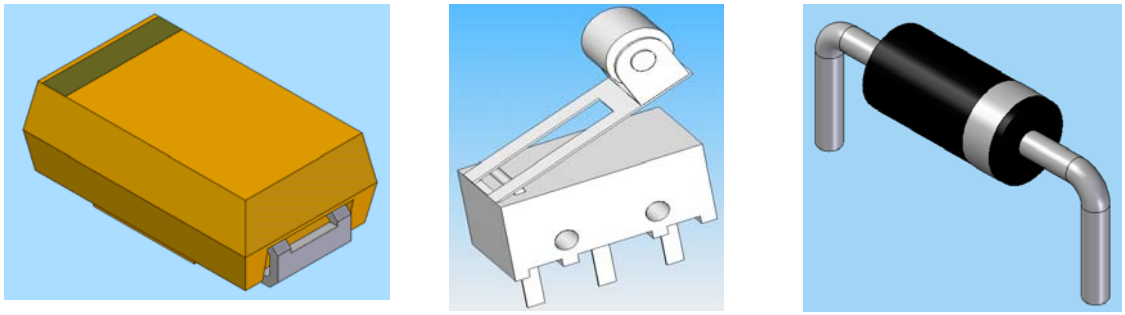


Figura 3.20. Algunas de las piezas realizadas con Solid Edge.

Si bien, a partir de un programa como Solid Edge es posible alcanzar una excelente presentación en la totalidad de un dispositivo como el Satélite Educativo, la meta perseguida por el Software de control de SATERHUM, requiere características de diseño detallado, mas también de una compaginación adecuada para con la manipulación en tiempo real de cada una sus partes; aspecto que de hecho Solid Edge no es capaz de solventar.

Por la naturaleza de lo pretendido para el modelo detallado de SATEDU, se presentaban dos opciones de buen alcance, compartiendo ambas el paso por un programa de computadora para realizar modelos y animaciones de realidad virtual. Dicho programa, tendría como primera tarea apoyar la aplicación de elementos visuales a las piezas diseñadas con antelación.

3DS MAX , se ha convertido desde hace varios años, en un compañero casi inseparable para las aplicaciones que emplean la librería gráfica OpenGL; siendo además, la herramienta empleada

por excelencia para realizar modelos tridimensionales complejos para videojuegos. Este comentario cabe al tomar en cuenta que dentro de las aplicaciones de computación gráfica, los videojuegos constituyen el ejemplo más difundido en el terreno de la interacción en tiempo real entre el usuario y la computadora. Por otro lado, este programa de computadora, creado y comercializado por la compañía Autodesk[17], permite trasladar modelos realizados desde casi cualquier programa de CAD/CAM, hasta su utilización dentro de OpenGL; convirtiendo un archivo IGES[18] por ejemplo, en uno perfectamente manejable desde dicha librería gráfica, como lo es el de tipo 3DS[19].

Con el apoyo de 3DS MAX se convirtió el tipo de archivo de cada pieza realizada en Solid Edge, para aplicarles texturas[20] a cada una de ellas, con el objeto de contar con un grado de detalle visual realmente importante, figura 3.21. Además, se crearon desde cero aquellas piezas cuyo nivel de complejidad, no requería una ayuda mucho mayor en su diseño.

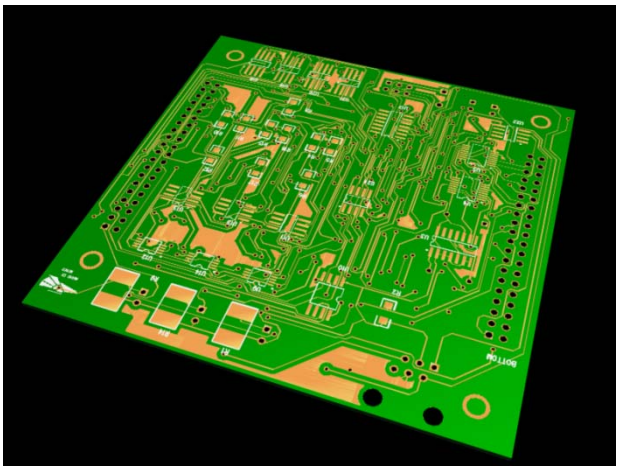


Figura 3.21. Paralelepípedo con textura aplicada en una de sus caras,
imagen correspondiente a una de las tarjetas del modelo virtual de SATEDU.

Una vez conseguido el modelado final de cada una de las piezas del Satélite Educativo, el siguiente paso se bifurcó de acuerdo a dos intenciones específicas perseguidas. Por un lado, se ensambló el modelo del SATEDU dentro de 3DS MAX para producir una película de animación que explica la totalidad del proyecto. Por otro, se trabajó desde OpenGL con las piezas ya terminadas, para conseguir, con OpenGL, una interfaz en realidad virtual del modelo físico, con el cual el usuario final del software interactuará directamente.

[17] **Autodesk**, es una compañía dedicada a software y servicios a las industrias de manufactura, infraestructura, construcción, medios y entretenimiento y datos transmitidos vía inalámbrica.

[18] **IGES** es la especificación para intercambio inicial de gráficos (Initial Graphics Exchange Specification) define un formato neutral de datos que permite el intercambio digital de información entre sistemas de diseño asistido por computadora CAD.

[19] El formato de archivo **3DS** se ha convertido en una estándar de la industria para transferir modelos entre aplicaciones de modelado 3D, o para almacenar modelos en catálogos de recursos 3D.

[20] La aplicación de **texturas** es una técnica de computación gráfica, en la cual mediante la correcta aplicación de imágenes (o trozos de imágenes tratadas) preestablecidas, se logra dar un toque de realismo a los modelos de una determinada animación.

Los modelos con formato 3DS resultan manipulables desde una aplicación basada en OpenGL. Para ello se emplean procedimientos de programación que convierten un archivo de tipo 3DS en un formato comprensible para la librería gráfica en cuestión. La programación referida en este caso, se pasa por alto debido a que el software de SATERHUM empleó una librería de programación predefinida, cuya aplicación minimiza el trabajo a unas cuantas llamadas de función, dejando modelos perfectamente manipulables desde el entorno que provee OpenGL.

Finalmente, y luego de haber aplicado las transformaciones trigonométricas adecuadas desde OpenGL, la réplica en realidad virtual del Satélite Educativo que se obtuvo con mayor grado de detalle se muestra en la figura 3.22, lista para su uso en la interfaz del Software de SATERHUM.

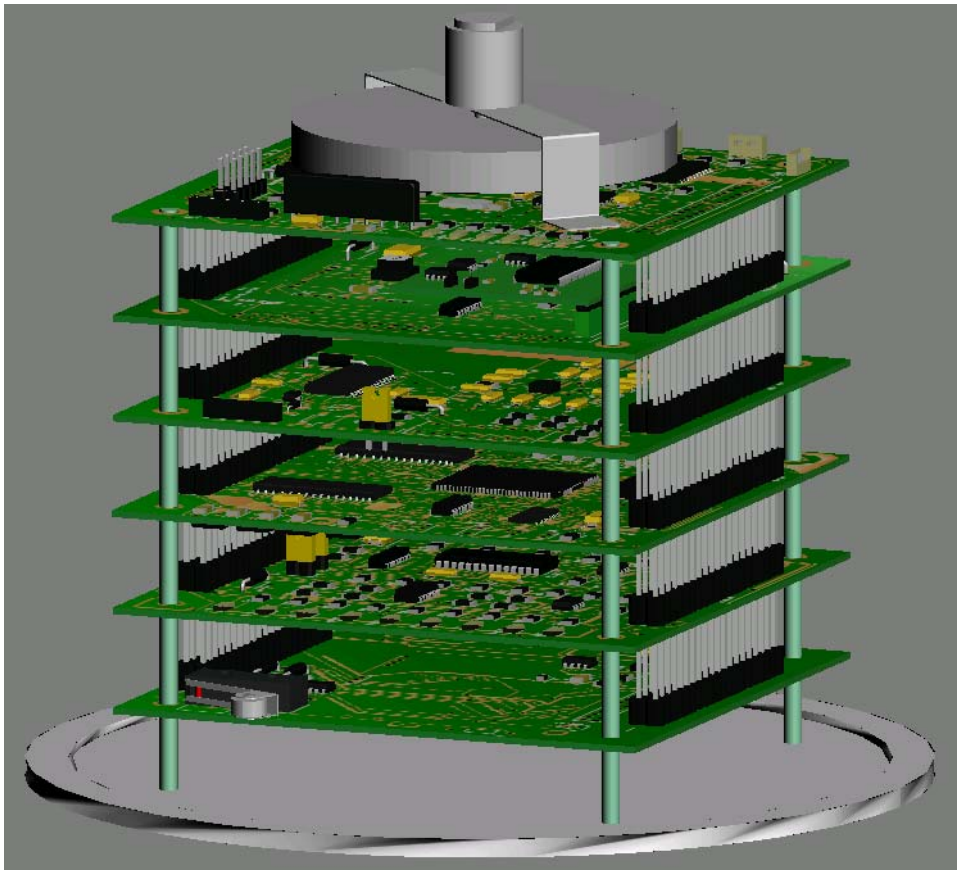


Figura 3.22. Réplica en realidad virtual del Satélite Educativo.

CAPÍTULO 4. INTERFAZ DE USUARIO DEL SOFTWARE DE OPERACIONES Y VISUALIZACIÓN

Dentro del desarrollo del Software tratado en la presente tesis, una de las partes en donde se postula un camino diferente al clásico, es en el rubro de la interfaz que permitirá a los usuarios finales mantener interacción total con el Sistema Satelital para entrenamiento de Recursos Humanos.

La idea del software nació del análisis de planeación profunda que se realizó en los inicios del proyecto completo. De ese modo se persiguió alcanzar una solución que fuese eficiente respecto a los objetivos fundamentales del proyecto, tanto en el terreno didáctico como en el funcional.

Basado en lo anterior, se pensó continuar con una línea enfocada en los gráficos por computadora, aprovechando lo atractivo de su presentación, además de la gama de posibilidades de manejo que pueden aplicarse en la actualidad. Para ello se persiguió diseñar desde sus cimientos, una interfaz que empleara modelos tridimensionales de realidad virtual para concentrar los ingredientes indispensables para una interacción plena y efectiva con el usuario.

A lo largo de los años, la librería gráfica de programación OpenGL ha mantenido una importante independencia con el entorno en el cual se desenvuelve. Permitiendo así, realizar aplicaciones para distintas plataformas[21] con una misma herramienta en común. No obstante, el camino de independencia se convierte en utopía, al considerar que cualquier desarrollo que haga uso de ella, deberá manipular el entorno de trabajo – llámense ventanas, eventos y carga del programa en sí –, con algún elemento de programación que depende (en la mayoría de los casos) del Sistema Operativo empleado.

Para evitar la dependencia de la aplicación que haga uso de OpenGL, con una u otra plataforma, se presenta como opción el uso de una librería de utilidades, cuyo propósito se centra en el apoyo hacia el manejo de aplicaciones, ventanas, y eventos de programa, para creaciones basadas en OpenGL, de nombre GLUT.

GLUT posee una serie de características valiosas, que facilitan los aspectos básicos de cualquier aplicación de entorno gráfico[22], para concentrar esfuerzos en los desarrollos basados en OpenGL; teniendo por ejemplo; manejo de múltiples ventanas, procesamiento de eventos de aplicación, reconocimiento de una gran gama de dispositivos de entrada[23], facilidad en la creación de menús tipo cascada, entre algunas otras.

Si bien, los objetivos del Software para SATERHUM no contemplan la compatibilidad con diversos entornos de trabajo, dicha prestación se añade de forma complaciente a la solución sencilla y eficiente que brinda GLUT en lo que a entornos de trabajo para aplicaciones en

[21] Una **plataforma** en informática, puede referirse como el hardware, software o la combinación de ambas, sobre las cuáles habrá de ejecutarse una aplicación de software determinada.
[22] Las aplicaciones de **entorno gráfico** son aquellas en las que el usuario interactúa con el programa de computadora mediante íconos, ventanas, barras de herramientas, etc.
[23] dispositivos de entrada: aquellos que permiten al usuario introducir datos, comandos y programas en la computadora.

OpenGL se refiere. Esto brindó en su momento, la claridad para pensar en una interfaz basada en modelos tridimensionales, que no representara una tarea sin solución.

4.1 GLUT Y EL DESPLIEGUE EN PANTALLA

El empleo de GLUT se realiza desde las etapas tempranas del desarrollo de aplicaciones basadas en OpenGL, manejando un conjunto fundamental de funciones que se encargan de desplegar de manera correcta los modelos requeridos. Dichas funciones brindan en cualquier programa del mismo tipo, una suerte de pasos a seguir, a través de un proceso previamente establecido. Entre ellas init(), para la inicialización general de los parámetros de la aplicación; reshape(), para el establecimiento de la perspectiva de vista de la escena; y display(), para el establecimiento de los parámetros de dibujo del modelo.

Cada una de las funciones de programación mencionadas se relacionan con diferentes aspectos específicos de la escena general, creada en OpenGL. De tal forma que resulta recomendable procurar un uso congruente del esquema. Así, mientras que dentro de la inicialización de los parámetros de la aplicación deberán fijarse aspectos de color, suavizado, iluminación y prueba de profundidad, por ejemplo. El caso del apartado de reshape() queda reservado para aclarar la manera en que serán visualizados los modelos en pantalla, de acuerdo a las proporciones del área de despliegue. Siendo finalmente en display() donde se acomodarán las funciones matemáticas generadoras del modelo tridimensional, junto con sus respectivas transformaciones.

A partir de la especificación correcta de cada una de las funciones anteriores, el curso natural que exige GLUT, al ser una librería basada en el lenguaje C[24], es la llamada a una función principal – tomando su adjetivo, de su nombre en inglés, “main” – , que concentre a todas aquellas funciones que la aplicación requiera. Desde las que resultan básicas, hasta las que provean medios de interacción usuario-programa, a través de rutinas de control; pasando por aquellas que añadan funcionalidad adicional al programa. De esta forma, GLUT, figura 4.1, provee un medio de trabajo segmentado, con el que es posible hacer un trato directo y diferenciado, sobre las prestaciones que deba tener la aplicación en cuestión.

```
#include <GL/glut.h>  //librería necesaria
......
void init(void)
{
//se definen las variables de materiales e ilumiación

GLfloat mat_specular[] = { 1.0, 1.0, 1.0, 1.0 };
GLfloat light_position[] = { 4.0, 0.0, 3.5, 0.0 };
GLfloat white_light[] = { 1.0, 1.0, 1.0, 1.0 };

......

glClearColor(0.0, 0.0, 0.0, 0.0);
glShadeModel(GL_SMOOTH);

......
//se manda llamar  a las funciones de materiales e ilumiación
```

[24] C es un lenguaje de programación creado para la implementación de sistemas operativos; es apreciado por la eficiencia del código que produce, es el lenguaje más popular para crear software de sistemas.

```
glMaterialfv(GL_FRONT, GL_SPECULAR, mat_specular);
glLightfv(GL_LIGHT0, GL_POSITION, light_position);
glLightfv(GL_LIGHT0, GL_DIFFUSE, white_light);

.....

glEnable(GL_LIGHT0);
glEnable(GL_DEPTH_TEST);

CalculoNormales(); //en init() se calculan las normales de los modelos

}

void display(void)
{
        //se definen los buffers a utilizar
        glClear(GL_COLOR_BUFFER_BIT| GL_DEPTH_BUFFER_BIT);

//color de fondo
        glColor3f(1.0,1.0,1.0);

        glLoadIdentity();

//se define desde donde y cómo se visualizará la escena
        gluLookAt(0.0,-20.0,5.4,0.0,0.0,0.0,0.0,0.0,1.0,0.0);

//Transformaciones y dibujo de modelos

glTranslatef(1.0,0.0,2.0);
DibujaModelo();

    glFlush();      //asegurar que todo se haya dibujado
        glutSwapBuffers();    //intercambio de buffers

}
void reshape(int w, int h)
{
        glViewport(0,0,(GLsizei) w, (GLsizei) h);
        glMatrixMode(GL_PROJECTION);    //se usará matriz de proyección
        glLoadIdentity();

/*****función que define la perspectiva*****/
        glFrustum(-1.0,1.0,-1.0,1.0,1.5,200.0);
        glMatrixMode(GL_MODELVIEW); //se usará matriz de modelo de vista
        glLoadIdentity();
}
/****función para eventos del teclado***/
void keyboard (unsigned char key, int x, int y)
{
 ......

  switch (key)
    {
    case 'i':
    case 'I':

      .....

    case 'q':
    case 'Q':
      exit (0);
      break;
    }
}
/****función principal de la aplicación que hace uso de las anteriores definiciones****/
int main(int argc, char** argv)
{
        glutInit(&argc, argv);
        glutInitDisplayMode(GLUT_SINGLE | GLUT_RGB);
        glutInitWindowSize(1000,800);
        glutInitWindowPosition(0,0);
        glutCreateWindow(argv[0]);
        init();
    glutKeyboardFunc (keyboard);
        glutDisplayFunc(display);
```

```
        glutReshapeFunc(reshape);
        glutMainLoop();
        return 0;
}
```

Figura 4.1. Bosquejo de código de programación de un programa con GLUT

4.2 DISEÑO DE LA INTERFAZ DE USUARIO FINAL

De acuerdo a lo antes expuesto, las funciones que dotan a la aplicación de los medios necesarios para una correcta interacción (por parte de quien haga uso de ella) serán las funciones primarias que acompañen a la librería de utilidades de OpenGL, logrando así una compaginación virtuosa, en aras de combinaciones más completas.

Para diseñar una interfaz de usuario con características de manipulación totalmente expuestas, debe tomarse como punto de partida de los modelos gráficos el desarrollo de rutinas de programación que representen la funcionalidad requerida, al tiempo que se interrelacionen con aquellas preestablecidas para fines de presentación.

El manejo de eventos propuesto para el software del SATERHUM, consistió en dos aspectos básicos. El empleo del teclado para la libre navegación sobre el modelo de realidad virtual, a manera de recorrido. Además de la ayuda del ratón de la computadora en el manejo de menús emergentes, información adicional, características de seguimiento y exploración minuciosa de los módulos de SATEDU. Con ello se pensó cubrir en su totalidad los aspectos funcionales de software, necesarios para SATERHUM.

Por otro lado, la manera en la que la aplicación se presentaría al final, fue diseñada a partir de dos momentos fundamentales de operación. El primero cuando el Satélite Educativo no se encuentra operando, en cuyo caso, el software le ofrece al usuario información del proyecto y de cada uno de sus módulos, partiendo del modelo detallado en realidad virtual de SATEDU. El segundo, cuando SATEDU requiera monitoreo. Así entonces, se pensó en un programa basado en dos ventanas básicas, de las cuales, la principal contendría el modelo detallado, con los medios para iniciar las actividades del prototipo satelital y su respectivo seguimiento.

4.2.1 VENTANA PRINCIPAL

El software del Sistema Satelital para entrenamiento se desarrolló para entrenar personas en el área satelital. Adaptándose de modo sencillo a las circunstancias propias de un proceso de aprendizaje. Si bien cuando el sistema se emplee en un salón o laboratorio debidamente acondicionado, los estudiantes podrían contar con la infraestructura completa que compone al SATERHUM, probablemente les sería complicado, trasladar el sistema completo fuera de los salones de clase. Esta situación se resuelve al usar el sistema junto con una computadora portátil. Por ello, el software de comunicaciones debe contar con un camino de libre tránsito, sin la necesidad de cables.

La ventana primaria que se desarrolló para el software del SATERHUM, figura 4.2, se emplea para una serie de propósitos indispensables. Desde la simple exploración a partir del modelo tridimensional de SATEDU, pasando por el escudriñamiento minucioso de sus características;

hasta la puesta en marcha de las acciones del satélite educativo, con el respectivo seguimiento instantáneo. Todo esto mantendrá un orden, sin que interfieran en ningún momento una función con la otra.

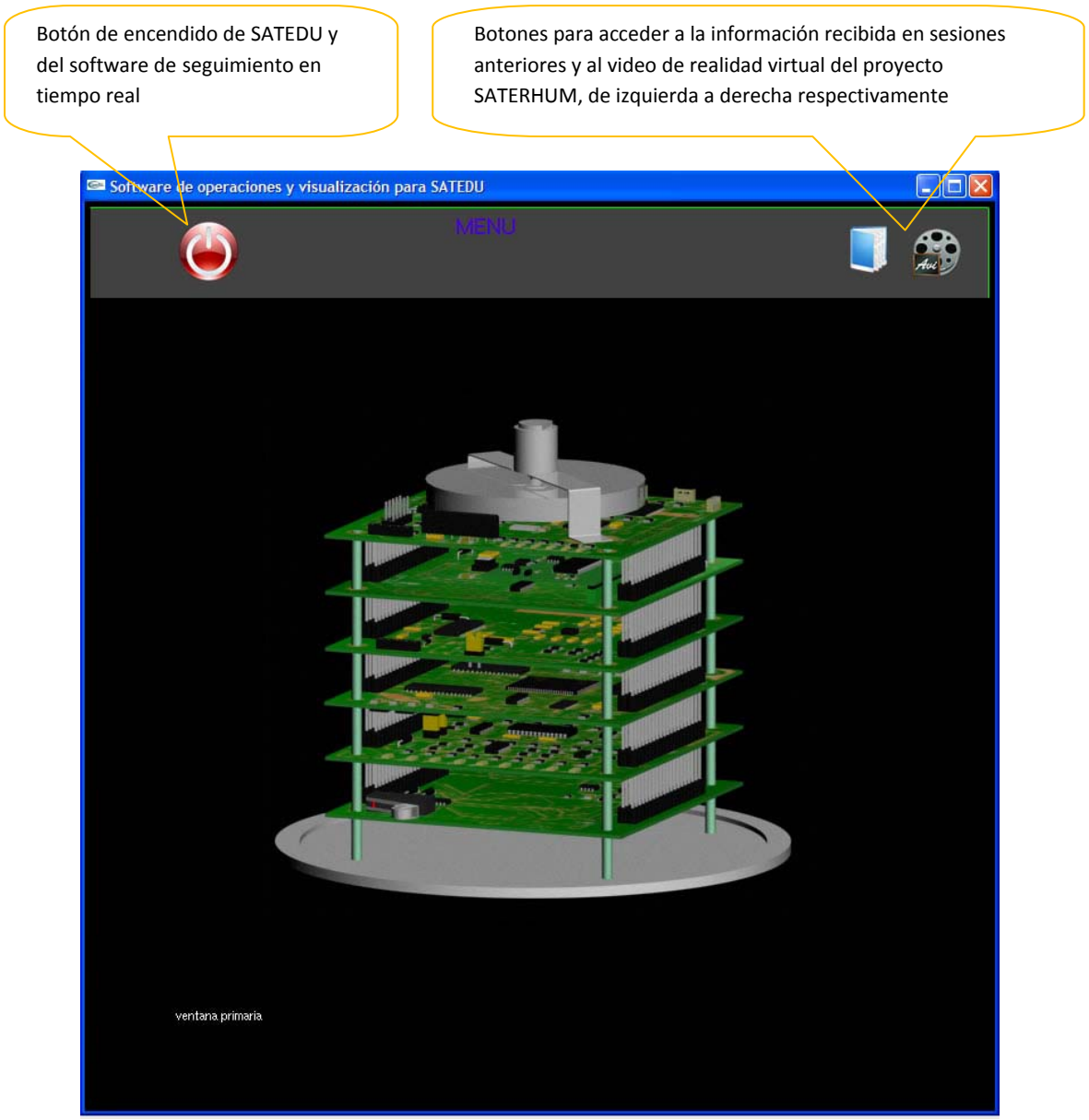


Figura 4.2. Ventana principal del software del SATERHUM

4.2.1.1 VENTANAS ANIDADAS

GLUT maneja dentro de sus características, la posibilidad de crear varias ventanas, las cuales pueden ser o no independientes. Es decir, pueden definirse ventanas, que residan dentro de otra, o que subsistan por sí solas. Así, es posible desplegar elementos distintos, que a pesar de pertenecer al mismo proceso, no interfieran en el espacio predeterminado para los demás. Lo

anterior es valioso, si se pretende desplegar un modelo tridimensional, cuyo movimiento, en él mismo o en la cámara[25] que lo observa, pueda indeseablemente, modificar la visualización de otros elementos. Este caso fue previsto en el diseño del software tratado en esta tesis.

De la ilustración correspondiente a la presentación general de la ventana principal, figura 4.2, se desprende el uso que se hizo del concepto de subventanas; mostrando una división efectiva entre el área en que se presenta el modelo de SATEDU y aquella en la que aparecen algunas opciones del software. Para este módulo de aplicación, la división referida se hace necesaria a partir del momento en que el modelo de realidad virtual del satélite se use para que los usuarios observen a detalle, las piezas y características de los componentes del prototipo satelital; realizando una especie de paseo en primera persona[26] a lo largo de todo el modelo. Esto se logra con el movimiento de la cámara, hecho con el apoyo de OpenGL. Dicho movimiento, provoca la pérdida de visualización de algunos objetos de la escena, tal y como pasaría en la filmación de una película. Por lo tanto, y tomando en cuenta que se trata de opciones que deben ser accesibles, se anida una ventana sobre la que se dibujó el modelo del Satélite Educativo, cuyo foco de visualización no se mueva junto con la cámara de la escena principal.

Con el entorno territorial definido, en cuanto al campo de visualización se refiere, fue posible diseñar a plenitud, una interacción completa y funcional, ofrecida por la ventana principal del programa de visualización y control de SATEDU. En donde el manejo adecuado de los eventos ocurridos en la aplicación brindan las prestaciones requeridas en conjunto con un modelo versátil y puntual del prototipo satelital.

El segmento de ventana o módulo principal de la anidación de las mismas, dentro de la ventana primaria del software del SATERHUM, fue concebida para apoyar las funciones de exploración y reconocimiento de SATEDU. Así, con la ayuda de algunas teclas del teclado, se puede realizar un recorrido minucioso de cada parte del modelo del Satélite Educativo; avanzando, retrocediendo y virando vertical u horizontalmente. Consiguiendo con ello una idea plena y certera de sus componentes, gracias al importante nivel de detalle que presenta cada pieza, con su forma e identificador reales, figura 4.4.

Para este aspecto específico de la interfaz, GLUT provee una sencillez notable en el manejo de los eventos recibidos desde el teclado. Dentro de la definición de una rutina de programación de propósito específico, se marcan las reacciones que habrá de presentar la aplicación, dependiendo de la tecla que se haya especificado, tomando como referencia simbólica el conjunto de caracteres ASCII[27]. Dichas reacciones marcadas, se encuentran referidas en este caso al movimiento que habrá de experimentar la cámara, previamente definida desde OpenGL. Si bien, desde este momento se presentó la necesidad de hacer uso de la animación por computadora, el

[25] En computación gráfica la **cámara** representa, al igual que en una película, el elemento que define lo que estará visible de una determinada escena; sea por el acercamiento, el ángulo de visión o las características de la toma que se maneje.

[26] En aplicaciones de realidad virtual, se hace alusión a vista en **primera persona**, cuando el usuario ve en pantalla los modelos como si él mismo se desplazara dentro de la animación.

[27] El código **ASCII** (acrónimo inglés de **A**merican **S**tandard **C**ode for **I**nformation **I**nterchange — *Código Estadounidense Estándar para el Intercambio de Información*) pronunciado generalmente [áski], es un código de caracteres basado en el alfabeto latino tal como se usa en inglés moderno y en otras lenguas occidentales

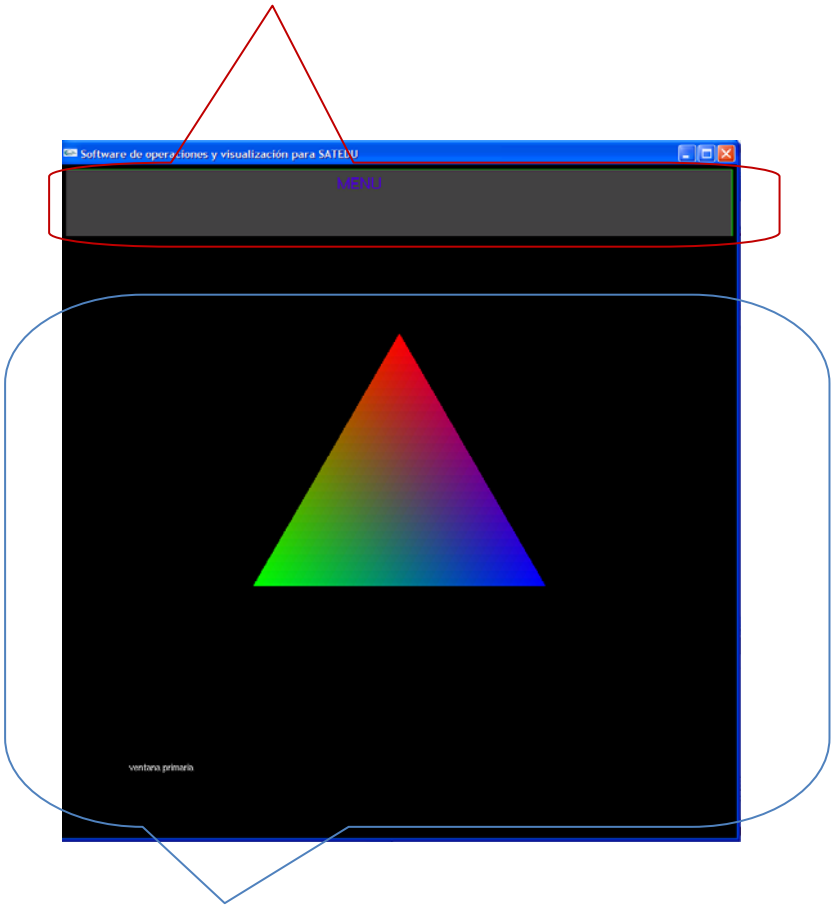
tema de la animación se aborda hasta el siguiente capítulo, por considerar que ello contribuye al entendimiento tanto del tema como de la misma animación.

```
void  subDisplay ()
{

/* Iniciaización de la subventana */
glutSetWindow (winIdSub);
glClearColor (0.25, 0.25, 0.25, 0.0);
glClear (GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);

/* Dibujos que se presentarán en la subventana */

......
}
```




```
void
mainDisplay (void)
{

  /* Clean drawing board */
  glutSetWindow (winIdMain);
  glClear (GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);
  glLoadIdentity ();

  /**Dibujos sobre la ventana principal**/

DibujaTriangulo();

  glutSwapBuffers ();
};
```

Figura 4.3. Subventanas con OpenGL.

Figura 4.4. Acercamiento a un giróscopo, hecho desde el software del SATERHUM

Como parte del reconocimiento completo del satélite y sus partes, se planteó un procedimiento que permitiera brindar información clara y efectiva de las bases operativas de SATEDU. Dicho planteamiento deriva en opciones tanto didácticas como informativas, cuyo funcionamiento tiene como elemento principal de interlocución comunicativa, entre el usuario y el programa, al ratón de la computadora.

Si bien, la base de programación con GLUT no difiere demasiado de lo que se comentó para el teclado, con respecto al manejo del ratón es necesaria la previsión de algunos elementos que intervienen en la interacción que el usuario puede llevar a cabo sobre la aplicación. Así por ejemplo, puede seleccionarse determinado objeto presentado en pantalla, partiendo de las coordenadas en las que se encuentre. O por otro lado, es posible hacer uso de la opción de menú emergente para aumentar las opciones funcionales posibles.

Teniendo lo anterior como punto de partida, se diseño un módulo de interacción basado en acciones ejecutadas por el ratón, en donde se combinan de manera efectiva las prestaciones de selección y de menús emergentes, para ofrecer un ambiente de manipulación dinámica y amigable. Con ello se cubren otros aspectos como: la exploración exhaustiva, a través del modelo de SATEDU; así como el apoyo didáctico por medio de videos explicativos, accesibles desde el menú desplegable. Respecto al ámbito de éste último aspecto, se puede hacer extensivo a lo largo de la aplicación, haciéndolo presente en todo momento, por medio de opciones que precisen ofrecerse al usuario.

La creación del módulo de interacción basado en el ratón de la computadora (destinado al trabajo en la ventana primaria de la anidación) se bifurcó en dos partes principales, que si bien se encuentran íntimamente relacionadas, su planteamiento exigió una prudente separación. Por un lado, un menú cuyas opciones se adaptaran a los cambios presentados en el escenario de trabajo; y por otro, la selección libre de cada parte del modelo del satélite, enfocado a una exploración independiente.

En lo que respecta a la actividad de exploración del prototipo, trató de emplearse un concepto en el cual el usuario manipule el modelo de realidad virtual de SATEDU sin la necesidad de botones o añadidos, con la opción de elegir directamente las partes a su conveniencia. Sin embargo, fue menester crear un medio de adaptación para dicho proceso. En éste, el modelo se

muestra de modo tridimensional y sobre el desenvuelven los eventos que GLUT captura desde el ratón de la computadora; además de que la selección de un determinado elemento requeriría en algunos casos de una búsqueda extenuante.

De manera general, el procedimiento empleado para ofrecer la manipulación directa del modelo del satélite se basó en mover el modelo a una posición tal, que pudieran visualizarse todas las piezas del segmento en donde se encontrara el elemento buscado, aunado a la asignación de coordenadas en pantalla para cada pieza presentada. Así, si por ejemplo se buscara una pieza dentro del subsistema de sensores, el usuario sólo debe elegir la tarjeta correspondiente desde una vista donde se encontraran disponibles todos los subsistemas; tras la selección, ello derivaría en una visualización plena de los elementos de la pieza. En donde cada uno tiene ya sus coordenadas bidimensionales asociadas, posibilitando su correcta referencia.

Con el objeto de completar una exploración efectiva de SATEDU a través de su modelo virtual, se acompañó la interacción con un menú emergente, que se acondiciona de acuerdo a los requerimientos de cada situación; de manera tal, que el usuario cuente con las opciones de manipulación necesarias para cada caso.

Por medio de la librería GLUT se concreta el manejo de menús emergentes, los cuales son una opción excelente para aplicaciones de gráficos por computadora debido a su versatilidad. Estos intervienen en la aplicación sólo cuando es requerido, lo que permite la libre visualización de modelos gráficos. Dichos menús son además de fácil programación y especifican aspectos básicos del funcionamiento del menú como: botón del ratón que dispara la ejecución del menú; árbol derivado de cada una de las opciones – sub-opciones de la primarias por decirlo de algún modo –, y acción relacionada con la elección de alguna de las opciones.

Con éstos, se diseñó un conjunto de menús cuyas opciones responden a lo que se está presentando en pantalla, mostrando en un determinado caso que se esté seleccionando una pieza determinada (al hacer click sobre el botón derecho del ratón) las posibilidades de pasar a: un video explicativo, alternativas de la pieza, hoja de especificaciones, etc.

En las figuras 4.5_1, 4.5_2, 4.5_3,4.5_4, se muestra respetivamente se muestra una ilustración, por cada paso que se debe seguir en el proceso de exploración total del modelo virtual del Satélite Educativo.

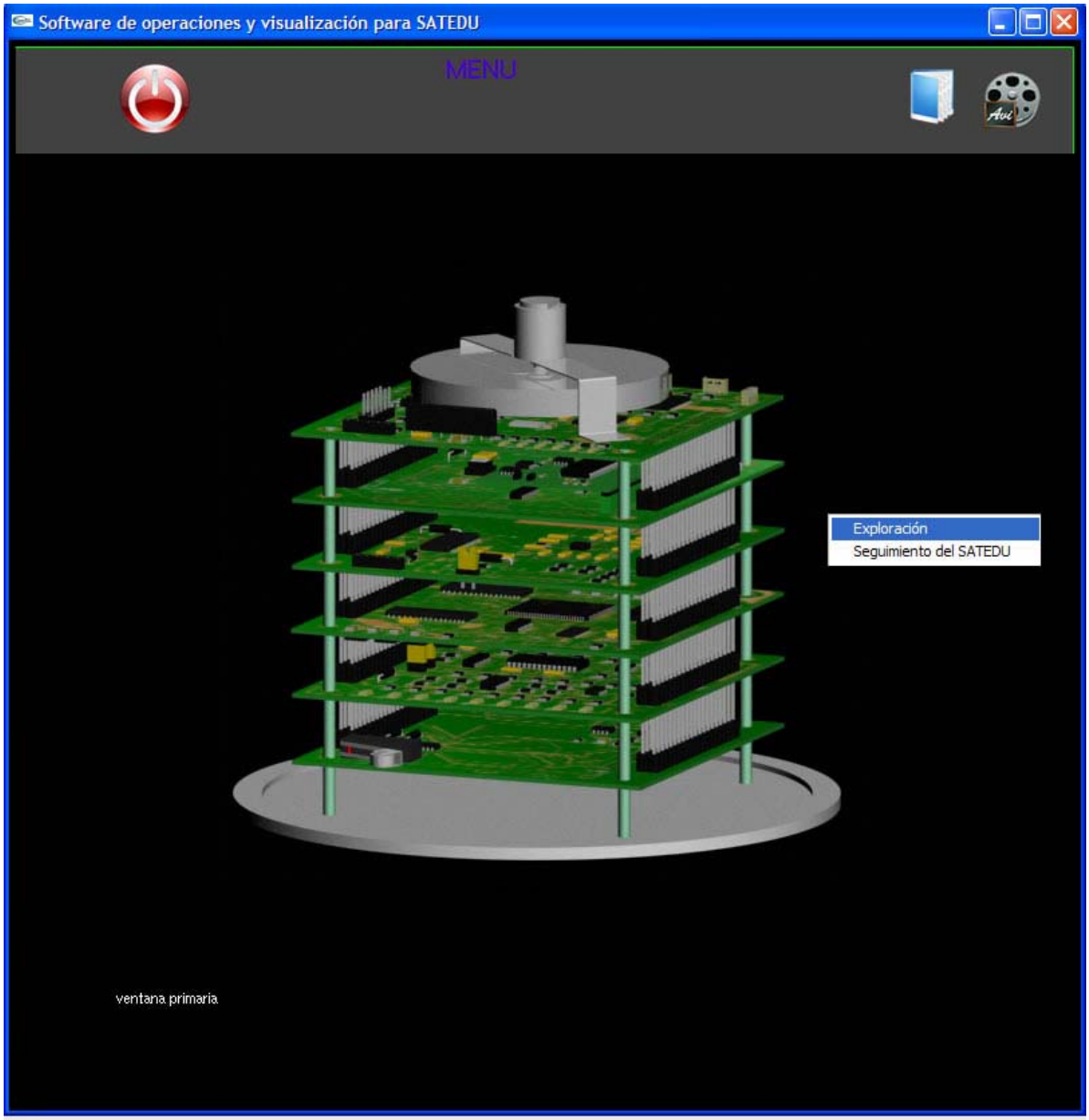


Figura 4.5_1. Primer paso de la exploración del modelo del SATEDU.

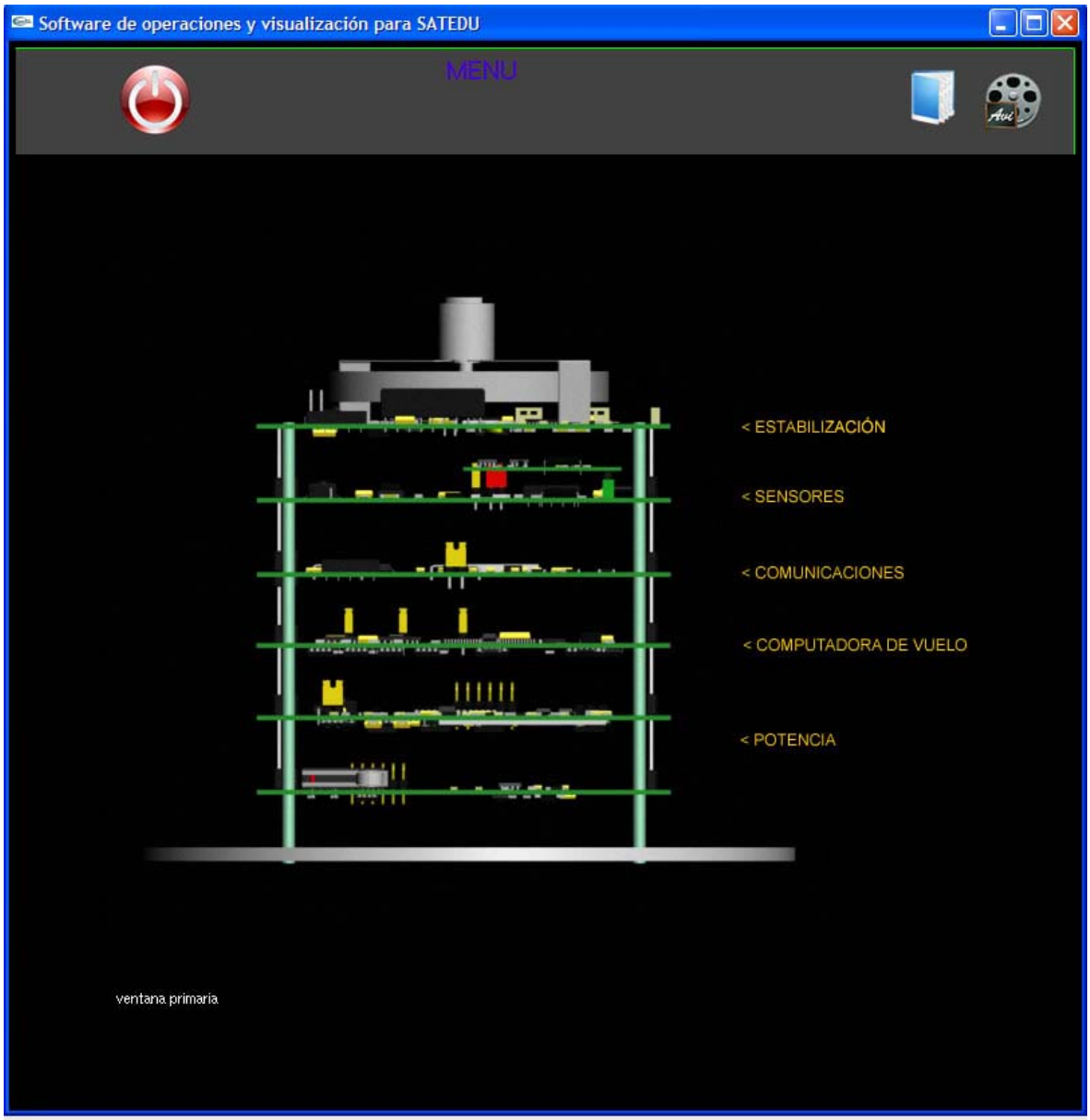


Figura 4.5_2. Segundo paso de la exploración del modelo del SATEDU.

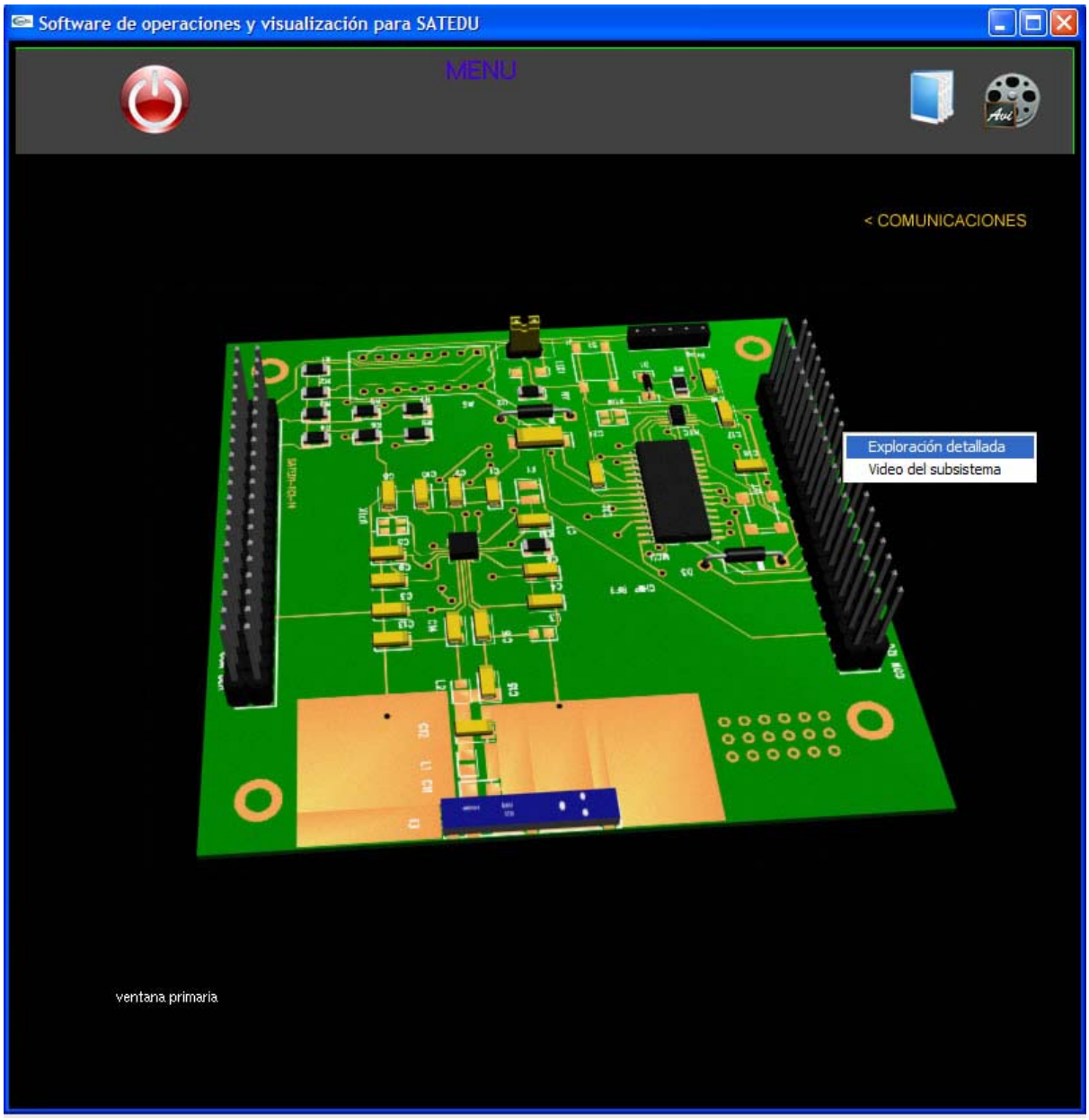


Figura 4.5_3. Tercer paso de la exploración del modelo del SATEDU.

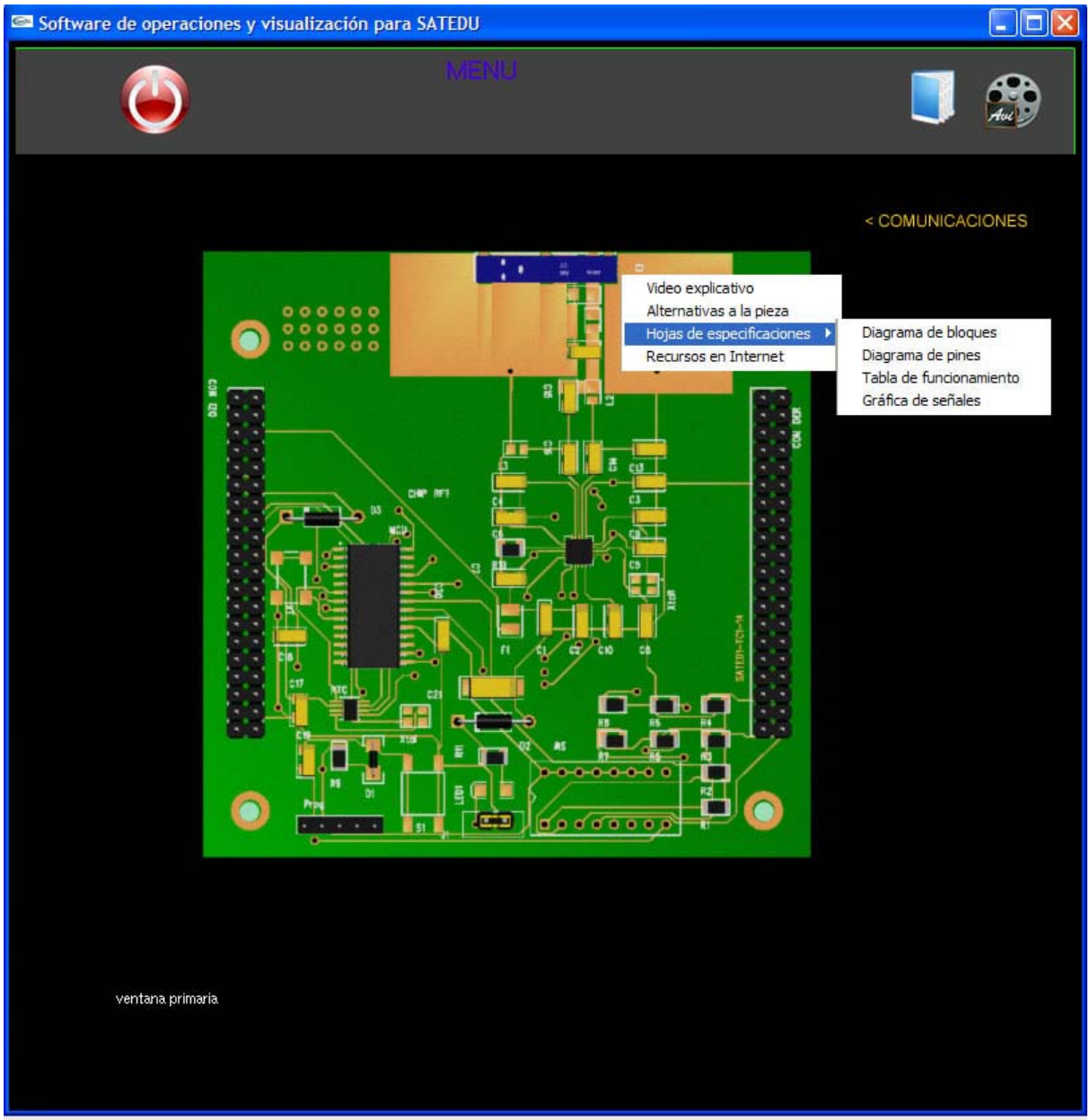


Figura 4.5_4. Cuarto paso de la exploración del modelo del SATEDU.

Una vez planteada la forma de manejar la ventana primaria, perteneciente a la anidación producida en la pantalla principal del software de visualización y control del SATEDU, restaban algunas opciones que debido a su trascendencia requerían de una vista fija a lo largo de la ejecución del programa. Entre ellos por ejemplo: la puesta en marcha del módulo de

seguimiento del satélite, así como el vínculo hacia los registros de telemetría[28] recibida en sesiones de trabajo anteriores. Los cuales requieren estar a la vista del usuario en todo momento, para su fácil análisis. Para este propósito se empleó una sub-ventana, sobre la primera definida, consiguiendo que sus elementos se mostraran sin cambios, ante movimientos presentados en el modelo virtual del satélite.

La ventana anidada a la principal, quedó exenta del uso de menús emergentes, empleando solo reconocimientos espaciales de eventos de ratón para desencadenar el proceso predeterminado de acuerdo a la opción elegida.

4.2.2 VENTANA DE SEGUIMIENTO EN TIEMPO REAL DEL SATEDU

Los mismos conceptos que se utilizaron en el planteamiento de la interfaz de la ventana principal del software de SATERHUM sirvieron para presentar el seguimiento en tiempo real del Satélite Educativo en una ventana. Con la diferencia, por un lado, de que en éste caso no cuenta con las opciones dispuestas en el menú emergente de la primera, y por otro, de hacer uso de una sub-ventana con elementos gráficos que además de generar una determinada actividad, muestra datos en tiempo real, provenientes de la interacción activa con SATEDU, figura 4.6.

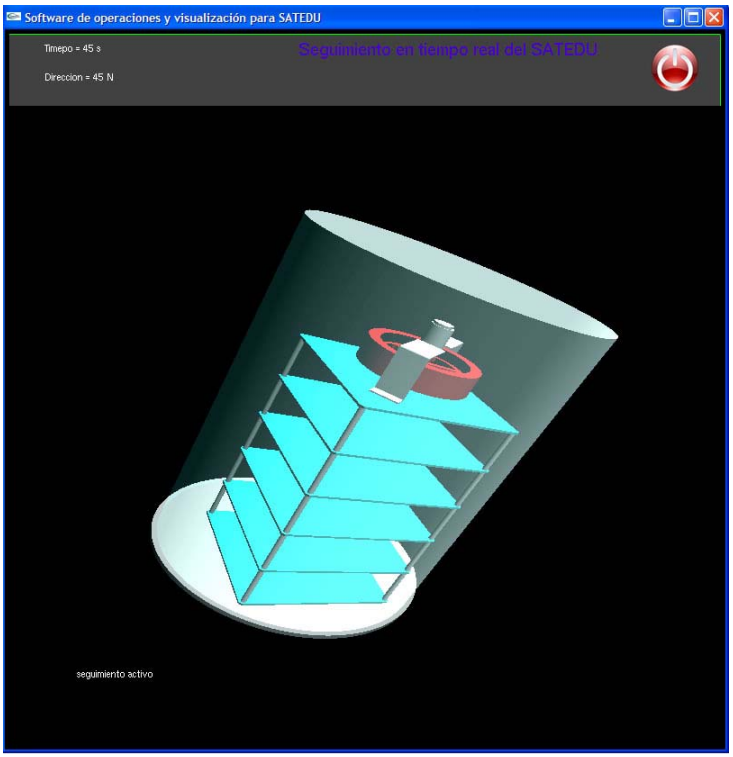


Figura 4.6. Ventana correspondiente al seguimiento en tiempo real del SATEDU

[28] La telemetría se refiere a los datos recibidos desde un puerto serial.

CAPÍTULO 5. SEGUIMIENTO EN TIEMPO REAL DEL SATEDU

Debido a los objetivos que persiguió el desarrollo del software del SATERHUM, se tornó conveniente realizar el monitoreo del estado instantáneo de su prototipo satelital, lo que brindaría las bases para una relación usuario-sistema de interacción plena, y que a su vez, permitiría complementar de manera general, los escenarios de control y visualización para el proyecto en su conjunto.

Así, se diseño un módulo de software especial para el seguimiento en tiempo real del Satélite Educativo, cuyo propósito en una primera etapa se concreta a registrar el movimiento rotatorio del prototipo. Logrando con ello desplegar en pantalla el modelo en realidad virtual del SATEDU dando seguimiento automático a los movimientos que experimente el dispositivo real. Con el desarrollo de dicho módulo, se dispone de un método probado extrapolable al establecimiento de un apartado completo de interacción entre el software y el satélite. Cabe aclarar que la construcción de semejante módulo se realizó sólo con el empleo de la tarjeta de sensores por parte de SATEDU y resta la construcción completa del prototipo satelital. Aparentemente, ello no representa dificultades posteriores para su implantación.

Este capítulo presenta el procedimiento en que fue logrado el objetivo expuesto, haciendo uso de una variedad importante de elementos. Desde el manejo de los dispositivos de lectura de datos de movimiento; la programación necesaria para la recepción, manipulación e interpretación de los datos relacionados; hasta la animación del modelo tridimensional abordado con antelación.

5.1 ESQUEMA BÁSICO DEL SEGUIMIENTO

El seguimiento en tiempo real del movimiento rotacional de SATEDU fue proyectado con la unión entre el software de visualización y el dispositivo físico para hacer posible la comunicación sin interrupciones, en sí mismo. Ello con el objeto de de presentar en la aplicación, una réplica virtual del satélite, cuyo movimiento correspondiera en tiempo y forma con el que en la realidad estuviera sucediendo.

De esta forma, se planteó un procedimiento cuya forma básica puede descomponerse como sigue:

- Se establece la comunicación desde el programa de computadora hacia el satélite, con el envío vía puerto serial de un comando que desencadena un ciclo en el que la tarjeta de sensores de SATEDU estará leyendo sus sensores de navegación (movimiento) transmitiéndolos instantáneamente al software de laptop.
- El software de Laptop recibe los datos y los interpreta a fin de replicar el movimiento ocurrido en SATEDU.
- El proceso continúa hasta que se envíe un nuevo comando desde la aplicación, que detenga todo el proceso.

5.2 SENSADO DE DATOS DE MOVIMIENTO

Una vez que la tarjeta de sensores recibe el comando enviado por la aplicación de computadora, procede a activar el funcionamiento de los sensores para después obtener la información, que luego de interpretarla la representa como indicadores de movimiento. Sin embargo, como se mencionó en el primer capítulo de esta tesis, el subsistema de sensores está compuesto por una serie de sensores diferentes entre sí. No obstante, para registrar el movimiento de rotación sólo se utilizó el giróscopo del plano paralelo al suelo terrestre.

Un giróscopo es un dispositivo que mide velocidad angular en grados por segundo (˚/s), sobre el eje en que se localiza el giróscopo, para registrar datos de posición instantánea. En particular, la instrumentación de la tarjeta de sensores de SATEDU contiene tres giróscopos colocados ortogonalmente para registrar movimiento en 3 dimensiones, figura 5.1.

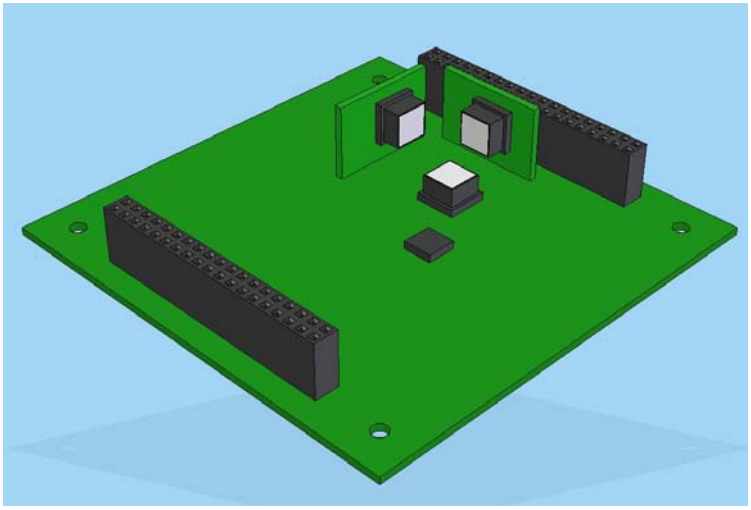


Figura 5.1. Disposición de los giróscopos en la tarjeta de sensores de SATEDU.

Con el arreglo referido basta con interpretar desde el software de la computadora, los datos obtenidos por los giróscopos para extraer la información instantánea de posición requerida. Sin embargo, se requiere resolver el efecto de error en los datos de posición causado por la falta de rigidez en el punto de fijación de los giróscopos. Dicho efecto, conocido como deriva, además de la interpretación respectiva necesaria se resolvió con una rutina sencilla integrada al software de SATERHUM. Estos datos se encuentran disponibles para su utilización en la animación del modelo virtual del Satélite Educativo.

En la figura 5.2 se muestra el esqueleto del código que permite interpretar los datos recibidos desde el subsistema de sensores.

```
-> Recibe cadena de 7 bytes
   -> (CmdRx[1-7]

//se obtiene el dato del giróscopo en decimal

suma = CmdRx[2] * 256 + CmdRx[3]

//mediante el bit de signo se sabe el sentido

si suma > 2047
  entonces
     suma = suma - 4095 - 60 //se corrige y se resta error del giróscopo

  /* se pasa a º/seg y se multiplica por el factor de tiempo
  predispuesto en la tarjeta para obtener sólo los grados*/
posX = posX + (suma/4.1)(0.1)

/*se elimina la deriva*/
posXn = posX - 0.6 * contador
```

Figura 5.2.Procedimiento de programación para procesar los datos de la tarjeta de sensores de SATEDU.

5.3 COMUNICACIÓN SERIAL

La interfaz de comunicaciones entre el software de aplicación y el Sistema Satelital para entrenamiento de recursos humanos, es de tipo serial. Hasta el momento el intercambio de datos entre las partes se efectúa con un puerto serie convencional, sin requerir solucionar el aspecto inalámbrico de la comunicación.

El software del SATERHUM contiene un módulo cuyo propósito se concentrara en establecer una correcta comunicación de ida y vuelta entre ambas entidades. Dicha comunicación se basa en el puerto serie de la computadora, para ello emplea programación sobre C++, compatible mínimamente con Windows y con el entorno empleado en la creación del programa final, GLUT.

La comunicación serial es un protocolo para comunicación entre dispositivos que se incluía de manera estándar en cualquier computadora. Por otro lado, con la aparición de la interfaz USB[29] (basada en principios similares a la comunicación serial pero con un desarrollo tecnológico que permite tazas de transferencia de datos mucho mayores) el puerto serial ya no es tan utilizado en computadoras sobre todo en las portátiles. No obstante, la comunicación serial sigue siendo un

[29] **USB**: El Universal Serial Bus (bus universal en serie) es un puerto que sirve para conectar periféricos a una computadora. El USB permite plug-and-play, con una velocidad notablemente mayor a un puerto serie convencional

protocolo ampliamente utilizado en dispositivos de construcción electrónica y de instrumentación, por su importante sencillez de manejo.

El concepto de comunicación serial no representa mayores complicaciones. El puerto serial envía y recibe bytes de información un bit a la vez. Y aún cuando esto es más lento que la comunicación en paralelo, que permite la transmisión de bytes completos, este método de comunicación es más sencillo y con capacidades mayores de alcance.

Típicamente, la comunicación serial se utiliza para transmitir datos en formato ASCII, caso que de hecho se empleó en la comunicación satélite-software de control. Para realizar la comunicación, se utilizan 3 líneas: tierra o referencia, transmisión y recepción. Debido a que la transmisión es asíncrona, es posible enviar datos por una línea mientras se reciben datos por otra, a menos que se realice handshaking, o intercambio de pulsos de sincronización. Este caso no se tomó en cuenta para el módulo de monitoreo satelital al considerar que la comunicación es expedita, ante cualquier tipo de validación y verificación de datos.

Para que dos puertos seriales se puedan comunicar, es necesario que las características manejadas en cada extremo sean iguales. Dichas características quedan determinadas por:

- La velocidad de transmisión o baud rate, que indica el número de bits por segundo que se transfieren, y cuya unidad empleada es el baud[30].
- Los bits de datos, referidos a la cantidad de bits en la transmisión y prefijados por el protocolo que se esté empleando.
- Los bits de parada, usados para indicar el fin de la comunicación de un solo paquete, además de que brindan un margen de tolerancia ante el comportamiento asíncrono de los relojes en cada extremo.
- La paridad, que representa una forma sencilla de verificar si hay errores en la transmisión.

A partir de lo anterior, la aplicación de computadora del SATERHUM, debía contar con los medios necesarios para el establecimiento de los factores básicos de comunicación comentados. Sin embargo, por la estructura que llevaba el programa hasta este punto – basado en GLUT, OpenGL y C++ puro –, el camino marcaba la necesaria utilización del API Win32[31] para la programación de las comunicaciones.

Si bien el API Win32 posee las herramientas necesarias para la programación completa de un módulo de comunicaciones seriales, la manera en que las implementa no resulta del todo sencilla; convirtiendo en una buena opción, la utilización de una librería de programación, previamente definida y probada.

La librería para comunicaciones seriales[32] elegida, permite su sencilla incorporación, dentro de una aplicación basada en OpenGL y en GLUT, como la que se abordó en esta tesis. Brinda además, la posibilidad de realizar una programación sencilla del puerto serie de la computadora,

[30] **Baud**: unidad informática que se utiliza para cuantificar el número de cambios de estado o eventos de señalización, que se producen cada segundo durante la transferencia de datos

[31] **Win32** es un conjunto de funciones, tipos y mensajes pre-definidos para programar sobre los sistemas operativos de 32 bits de Microsoft.

[32] Se colocan los datos de la referencia en el respectivo apartado de esta tesis.

con todos los elementos mencionados. El código de programación que se utilizó para establecer la comunicación entre el software y el SATEDU, con el apoyo de la citada librería, se muestra en la figura 5.3.

Así pues, se añadió un módulo enfocado en la comunicación a través del puerto serial, en la parte de monitoreo en tiempo real del SATEDU. Este, además de interactuar con la tarjeta de sensores, permite hacerse extensivo hacia cualquier comunicación requerida dentro del SATERHUM. A diferencia del módulo actual, en el que se consideró irrelevante incluir bits de comprobación de datos – considerando difícil la presencia de errores en la transmisión – otras partes de SATEDU podrían requerir una comunicación controlada. De cualquier manera, como ya se comentó, la librería añadida al programa facilita significativamente el establecimiento de los factores de comunicación; haciendo posible un desarrollo incremental del proyecto en este tema.

```
.
.
.
CSerial serial;
        LONG    lLastError = ERROR_SUCCESS;

   lLastError = serial.Open(_T("COM1"),0,0,false); //abre el puerto serial
//establece los parámetros de comunicación
        lLastError = serial.Setup(CSerial::EBaud9600,CSerial::EData8,CSerial::EParNone,CSerial::EStop1);
//establece la máscara para la comunicación
        lLastError = serial.SetMask(CSerial::EEventBreak |
                                                                CSerial::EEventCTS   |
                                                                CSerial::EEventDSR   |
                                                                CSerial::EEventError |
                                                                CSerial::EEventRing  |
                                                                CSerial::EEventRLSD  |
                                                                CSerial::EEventRecv);

        lLastError = serial.SetupReadTimeouts(CSerial::EReadTimeoutNonblocking);
//manda el mensaje que requiere la tarjeta de sensores para responder con los datos requeridos
serial.Write("/¦AAAAAAAAAAAü");


        bool fContinue = true;
        do
        {
                //espera a que se envíen datos
                lLastError = serial.WaitEvent();


                // Salva el evento
                const CSerial::EEvent eEvent = serial.GetEventType();

                // maneja el evento de recepión de datos
                if (eEvent & CSerial::EEventRecv)
                {
                        // Lee datos, hasta que no falte ninguno
                        DWORD dwBytesRead = 0;

                        do
                        {
                                // Lee datos de puerto COM
                                lLastError = serial.Read(szBuffer,sizeof(szBuffer)-1,&dwBytesRead);

                                if (dwBytesRead > 0)
                                {
                                        // Finalize the data, so it is a valid string
                                        szBuffer[dwBytesRead] = '\0';
```

```
if(n_envio == 0)
{

if(szBuffer[0]=='N')
{
  fContinue = false;
  printf("Error en la transmision");

  exit(0);
  n_envio = 0;
}

else if(szBuffer[0] == 'K')
{
  fContinue = true;
  printf("Inicia Transmisión");
  n_envio++;

}
                                                    }
.
.
.
```

Figura 5.3. Código referente a la comunicación serial entre SATEDU y el software del SATERHUM

5.4 ANIMACIÓN DEL MODELO VIRTUAL DEL SATEDU

Una de las cosas más interesantes que se puede hacer en la computación gráfica es dibujar imágenes que se muevan. Ya sea para ver todas las partes de una pieza o dispositivo; para hacer un recorrido por un determinado escenario; o para que un elemento responda con movimiento de acuerdo con estímulos externos. Es claro que la animación es una parte fundamental del tratamiento de gráficos por computadora.

En una película de cine, el movimiento se logra al tomar una secuencia de imágenes y proyectarlas en la pantalla en una secuencia de 24 cuadros cada segundo. Cada cuadro de la película pasa por detrás de la lente, el disparador se abre y se despliega el cuadro. El disparador se cierra momentáneamente mientras la película avanza al siguiente cuadro, luego vuelve a desplegarse otro cuadro y así sucesivamente. A pesar de que el espectador está viendo 24 cuadros diferentes cada segundo, su cerebro mezcla las imágenes en una animación continua. En este sentido, los proyectores han evolucionado desde las películas antiguas que se notaban entrecortadas, trabajando con 16 cuadros por segundo, hasta los proyectores actuales que despliegan dos veces cada imagen, a una velocidad de 48 por segundo para reducir el parpadeo. En computación gráfica, la pantalla típicamente se refresca – redibuja la imagen – aproximadamente 60 o 76 veces cada segundo; llegando algunas veces, a una velocidad de 120 refrescos por segundo. Claramente, 60 por segundo resultará en un mejor suavizado que 30; y 120 será marginalmente mejor que 60; sin embargo, una velocidad mayor a los 120 por segundo, se encuentra más allá de lo que el ojo humano puede percibir con claridad.

La razón clave por la que el movimiento se puede dar en las películas de cine, se debe a que cada cuadro está completamente terminado cuando es proyectado. Cuestión que no ocurre en la pantalla de las computadoras, donde se realiza un barrido de arriba abajo y de izquierda a derecha, a fin de presentar las imágenes en pantalla. Por ello, el trabajar tal y como en el cine, representaría ciertos inconvenientes.

Partiendo por ejemplo, del deseo de realizar una animación en la computadora, compuesta por un millón de cuadros, con el empleo de un programa como el que se muestra a continuación.

```
abre_ventana();

for (i = 0; i < 1000000; i++) {

  limpia_ventana();

  dibuja_cuadro(i);

  espera_hasta_que_termine_un_veintricuatroavo_de_segundo();

}
```

Si se añadiera el tiempo que le toma a la computadora limpiar la pantalla para dibujar un típico cuadro de imágenes, el programa generaría más y más resultados indeseables dependiendo que tan cerca del 1/24 de segundo se encontrara el tiempo para limpiar y dibujar. Suponiendo que el dibujado tomara prácticamente 1/24 de segundo completo; los elementos dibujados primero, estarían visibles durante el completo veinticuatroavo de segundo, mostrando una imagen sólida en la pantalla; mientras que los elementos dibujados hacia el final serían borrados tan pronto se pasara al siguiente cuadro. Con esto, se estaría presentando una imagen indeseable, debido a que en la última parte de la imagen el ojo del espectador estaría viendo un fondo limpio la mayor parte del tiempo, en lugar de observar los elementos correspondientes. El problema entonces es, que el programa de ejemplo, no despliega cuadros completamente dibujados; presentando en su lugar, el proceso de dibujo mientras se desarrolla.

OpenGL ofrece un concepto de doble-buffering, en donde a través de dos buffers[33] de color completos, se realiza el despliegue con uno, mientras en el otro se dibuja. Una vez, que el dibujo del cuadro entero se termina, se intercambian los buffers para presentar la nueva imagen terminada siguiendo con dicho procedimiento a lo largo de toda la animación. Esto es como un proyector con sólo dos cuadros en un ciclo, mientras un cuadro se proyecta en pantalla, un artista borra y redibuja – ¡increíblemente rápido! – sobre el cuadro que no está visible. De este modo, el espectador no notará la diferencia entre este método y aquel en el que todos los cuadros están ya dibujados y el proyector simplemente los va desplegando uno tras otro. Con el doble-buffering, cada cuadro se muestra sólo cuando está completamente dibujado; haciendo que el espectador nunca perciba imágenes parcialmente dibujadas.

Una versión modificada del programa antes presentado, con las modificaciones pertinentes y de acuerdo a lo arriba planteado, podría ser el siguiente:

```
abre_ventana_en_modo_de_doble_buffer();
for (i = 0; i < 1000000; i++) {
  limpia_ventana ();
  dibuja_cuadro(i);
  intercambia_buffers();
}
```

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

OpenGL en sí mismo no tiene un comando intercambia_buffers() o swap_the_buffers, debido a que dicha característica no está disponible en cualquier hardware y sumado al hecho de que es un aspecto altamente dependiente del sistema de ventanas. Por ello, la animación del modelo virtual del SATEDU se hizo en GLUT, que además de contar con la función glutSwapBuffers(void) emplea el intercambio de buffers y brinda un procedimiento muy asequible para efectuar las transformaciones de la animación final respecto al movimiento de los modelos dibujados.

En la realización de animaciones con la librería GLUT con el doble buffer es necesaria la función glutIdleFunc(), que crea un ciclo de atención repetida a los cambios de escena de los elementos por animar. Los cambios o transformaciones se generan a partir de variables que cambian bajo cierta condición, ya sea desde un evento de teclado o ratón, o desde una rutina de programación que modifica su valor de acuerdo con un criterio. Todo esto, debe incidir sobre la función básica de GLUT, display() que aloja la definición del modelo geométrico.

A continuación en la figura 5.4 se presenta el código que de manera general sirve para generar animaciones. Si bien en este caso el movimiento depende de los eventos del ratón, el ejemplo es aplicable con ciertas modificaciones, a cualquier tipo de animación

```
#include <GL/glut.h>
#include <stdlib.h>

/*se define una variable que cambiará a lo largo de la
aplicación generando la animación*/
static GLfloat spin = 0.0;

void init(void)
{
.....
}
void display(void)
{
glClear(GL_COLOR_BUFFER_BIT);
glPushMatrix();
/*se dispone en una función de transformación
la variable spin que cambiará desde un evento del ratón*/
glRotatef(spin, 0.0, 0.0, 1.0);
glColor3f(1.0, 1.0, 1.0);
glRectf(-25.0, -25.0, 25.0, 25.0);
glPopMatrix();
glutSwapBuffers();   //se intercambian los buffers
}
void spinDisplay(void)
{
spin = spin + 2.0;
if (spin > 360.0)
spin = spin - 360.0;
glutPostRedisplay();
}
void reshape(int w, int h)
{
......
}
void mouse(int button, int state, int x, int y)
{
switch (button) {

            /*la continuación de la animación dependerá
            de eventos del ratón en este caso*/
case GLUT_LEFT_BUTTON:
if (state == GLUT_DOWN)

    glutIdleFunc(spinDisplay);
```

```
break;
case GLUT_MIDDLE_BUTTON:
if (state == GLUT_DOWN)

  glutIdleFunc(NULL);

break;
default:
break;
}
}

int main(int argc, char** argv)
{
......

}
```

Figura 5.4. Código necesario para una animación.

5.5 INTEGRACIÓN DEL SEGUIMIENTO EN TIEMPO REAL DEL SATEDU

A partir de los módulos de comunicación y animación se realizó el acoplamiento para generar el seguimiento en tiempo real del movimiento rotacional del Satélite Educativo. Sin embargo, al crear ambos módulos con procesamiento independiente y tomando en cuenta la necesidad de trabajarlos como procesos paralelos, se recurrió a la solución no del todo trivial denominada multi-hilos.

Cuando se ejecutan dos programas en un sistema operativo que ofrece protección de memoria como Windows y UNIX/LINUX, éstos se ejecutan como procesos separados, lo que significa que se les asignan espacios de direcciones separados. Esto quiere decir, que cuando el programa #1 modifica la dirección 0x800A 1234[34] en su espacio de memoria, el programa #2 no ve ningún cambio en el contenido de su memoria en la dirección 0x800A 1234. Con sistemas operativos más simples que no pueden logra esta separación de procesos, un programa defectuoso puede derribar, no sólo el programa mismo y otros que se encuentren en ejecución, sino incluso al sistema operativo.

Por otro lado, un hilo o 'thread', visto desde el procesador, es una secuencia de ejecución y cada hilo tiene su propio registro y pila de contexto. El entorno de tiempo de ejecución ejecuta solamente un hilo a la vez, interrumpiendo dicha ejecución cuando se necesitan recursos que no están disponibles. Por ejemplo en operaciones de entrada y salida, cuando el procesador cambia la ejecución de un proceso a otro, a lo que se conoce como "cambio de contexto". Cuando se bloquea un proceso por medio de la ejecución de un hilo diferente, el sistema permite reducir los tiempos muertos del procesador, lo cual recibe el nombre de "multitarea" y que emplean ampliamente los sistemas operativos ya mencionados.

Cuando se ejecuta un programa se le indica al sistema en que parte del disco puede obtener las instrucciones y la información estática. Al mismo tiempo se le asigna un conjunto de localidades de memoria virtual[31] conocidas como espacio de direcciones y que corresponden a

[34] **0x800A 1234** es un ejemplo hipotético de una dirección de memoria, desde que éstas son referenciadas por números hexadecimales

recursos del sistema. A todo este marco de ejecución se le denomina proceso. Sin embargo, antes de que un proceso pueda realizar cualquier acción, debe tener al menos un hilo de procesamiento. Cuando se crea cada proceso se le da automáticamente un hilo, de nombre “hilo primario”. Mas este hilo no tiene mayor capacidad que otros creados para dicho proceso y sólo recibe tal nombre por haber sido el primero en crearse. Por otro lado, el número de hilos en un proceso puede variar en tiempo de ejecución, en virtud del programa que se encargue de su control.

El trabajo realizado por un proceso puede dividirse en sub-tareas, cada una ejecutada en un hilo diferente, a lo que se le denomina “multihilo” o “multithreading”. Cada hilo en el proceso comparte el mismo espacio de direcciones y los mismos recursos de proceso. En estos casos el proceso termina cuando finaliza su ejecución el último hilo en el proceso.

La idea de tener más de un hilo en un proceso puede ser muy útil en algunos casos determinados. Si un proceso tiene solamente un hilo, se ejecuta en serie, generando pérdidas de tiempo en el procesador cuando el hilo quede bloqueado al no existir otro proceso en espera. Lo anterior puede ser inevitable, cuando las sub-tareas de un proceso deben realizarse en serie, sin embargo, este no es el caso de una buena variedad de procesos; como por ejemplo, el del módulo del software del SATERHUM abordado en el presente capítulo.

Tomando por un lado el proceso referente a la animación y por otro, el proceso de comunicación, se dispuso a cada uno bajo un hilo diferente, a fin de que los datos que sensaran los giróscopos fueran leídos y procesados hasta obtener los valores para las variables de rotación del módulo de animación. Con ello se logró un ciclo en donde la animación refleja instantáneamente los cambios de posición del Satélite Educativo.

La programación de seguimiento en tiempo real se realizó con el apoyo de Win32 API –empleado antes en la comunicación –, que además de que permite implantar multi-hilos de forma sencilla, resulta completamente compatible con la variedad de herramientas usadas en el desarrollo de otros módulos**.** En la figura 5.5 se muestra lo simple que resulta crear una aplicación multi-hilos, cabe la aclaración que en el código de dicha figura no se especificó el necesario uso de la función Sleep(), la cual cambia el foco de la aplicación de uno a otro hilo.

```
/*función principal encargada del hilo primario de la aplicación
en este caso se dispuso para la animcación*/

void main ( int argc, char** argv )
{
        /*se crea el hilo adicional que acompañará al de la apicación principal
        correspondiente a la función main()*/

        _beginthread( comunic, 0, (void*)12 );

 ......

 glutCreateWindow    ( "Seguimiento del SATEDU" ); // Window Title (argv[0] for current directory as title)
 glutFullScreen      ( );          // Put Into Full Screen
 InitGL ();
 glutDisplayFunc     ( display );  // Matching Earlier Functions To Their Counterparts
```

```
  glutReshapeFunc     ( reshape );

  ......
}

/*se definen las tarea que ejecutará el segundo hilo de la aplicación
que en este caso trató del módulo de comunicación e interpretación de datos*/

void  comunic( void *arg )
{
   int n_envio = 0;
              int x_1, x_2;
              int suma;
              float posX = 0.0;

              CSerial serial;
              LONG    lLastError = ERROR_SUCCESS;

......
}
```

Figura 5.5. Código de multi-hilos empleado en el módulo de seguimiento de SATEDU.

CAPÍTULO 6. CONCLUSIONES Y RECOMENDACIONES

Luego de haber realizado los trabajos de la primera etapa de desarrollo del software de visualización y operaciones del Sistema Satelital para entrenamiento de Recursos Humanos, se tiene un producto que ofrece funcionalidad y permite continuar su elaboración hacia la evolución final requerida. De esta forma, es posible realizar una serie de conclusiones y recomendaciones que permitan establecer un punto de partida para futuros trabajos.

6.1 CONCLUSIONES

Se desarrolló exitosamente la primera versión del software del SATERHUM para interactuar con el prototipo satelital educativo que actualmente se termina de integrar en el Instituto de Ingeniería, UNAM, como parte del proyecto CONACYT 7126.

El software de SATERHUM permite interactuar a nivel de funciones básicas con SATEDU, sin embargo ofrece excelentes apoyos de visualización de telemetría y seguimiento de los movimientos que experimenta físicamente SATEDU por medio de animaciones en tiempo real. Adicionalmente ofrece un video explicativo a los usuarios del sistema SATERHUM.

A lo largo del proceso de creación del software del SATERHUM se generaron resultados y conocimientos que podrán utilizarse en beneficio del proyecto final.

El empleo de gráficos por computadora como base de la interfaz de usuario, resultó ser una idea muy recomendable para proyectos en donde no es necesario ingresar datos, ofreciendo un manejo intuitivo y eficaz. Además, el empleo de gráficos no implica tiempos largos de programación al hacer uso de una librería de utilidades como GLUT.

En lo referente a modelos virtuales, el diseño inicial del software consideró a OpenGL como el único camino viable para el monitoreo del movimiento del SATEDU. Sin embargo, conforme fue avanzando el desarrollo del proyecto, se advirtió la posibilidad de utilizar herramientas que favorecen la rápida creación de modelos, y que permiten interactuar con OpenGL. A estas se les empleó en su momento, sólo para el modelado detallado del satélite, sin pensar que tal vez, hubieran podido suplir el trabajo extenuante de modelado con la librería gráfica mencionada.

De las herramientas que se mencionan arriba, particularmente 3DS MAX de Autodesk hizo posible realizar una película de animación que permite mostrar la manera en que opera SATEDU. Con esto queda claro que el modelado tridimensional es una opción muy buena para ejemplificar elementos físicos que aún no estén construidos, o que no se pueda tener a la mano.

Por otro lado, las aplicaciones como el caso de SATERHUM constituyen una opción interesante para fusionar elementos tanto educativos como de experimentación tecnológica. Sin embargo, pueden no representar la mejor opción para trabajos de experimentación pura, en los que sólo se necesite obtener información, sin mayores exigencias en muy poco tiempo

En términos de la evolución que pueda experimentar el software desarrollado en esta tesis, la integración del módulo para seguir en tiempo real los movimientos rotacionales del satélite cubrió todos los temas que son necesarios para que en un futuro el software pueda emular todas

las características de SATEDU. Desde la comunicación entre el dispositivo y el software, hasta la manera de lograr que dicha comunicación se ejecute en paralelo con otro proceso, pudiendo manipular los datos como sea necesario.

6.2 RECOMENDACIONES

Es recomendable que la evolución del software se vea reflejada sólo en lo que se refiere a sus funciones. Evitando por ejemplo, añadir detalles gráficos al modelo empleado para el seguimiento de los parámetros del Satélite Educativo que puedan hacer mella en el rendimiento de la aplicación en ciertos equipos de cómputo.

Sin embargo, es muy recomendable que al requerirse un cambio del modelo de SATEDU, por cuestiones de evolución de la estructura, se realicen desde un programa como 3DS MAX. Obteniendo resultados mucho más rápido, que lo que se lograría si se modela desde cero con OpenGL.

Existen además algunas consideraciones que no se tomaron en cuenta en la realización de esta etapa del software del SATERHUM, que posiblemente dificulten su compatibilidad en algunas computadoras con prestaciones muy limitadas. Por un lado las explicaciones que se hacen por medio del video de animación, pueden llegar a saturar la memoria de video de la máquina en algunos casos aislados, ya que si bien no se emplean videos de alta definición, algunas computadoras pueden presentar problemas con este tipo de procesamiento. Y por otro lado, la ventana principal del programa es la que contiene un modelo detallado, que además se encarga de cargar las películas de animación, por lo cual pudiera generar conflictos en ciertos equipos.

Se recomienda realizar una separación entre el trabajo de la ventana de seguimiento y la de exploración, para que a pesar de que alguna máquina no fuera capaz de procesar correctamente la parte de gráficos muy pesados de la aplicación, sea aun posible interactuar desde el software con el SATEDU por medio de la ventana de seguimiento.

Ahora bien, en lo que respecta a la interfaz de usuario puede ser necesario utilizar en algún momento, elementos de las interfaces gráficas convencionales. Esto puede lograrse empleando librerías de funciones de interfaz como las MFC (Microsoft Foundation Class), que proveen una rápida integración de controles de aplicación, como botones, barras de menús, entre otros. Lo anterior se recomienda partiendo de que es posible la integración de ese tipo de controles en aplicaciones basadas en GLUT y OpenGl, como la del SATERHUM

BIBLIOGRAFÍA

Breton, Rémi et. al.
Learning Autodesk 3ds Max 2008
Focal Press, 2007.

Shreiner, Dave et. al.
OpenGL Programming Guide
Addison-Wesley, 5ta Ed, 2006.

Vince, John
Mathematics for Computer Graphics
Springer, 2nd Ed, 2006.

García, Oscar et. al.
Introducción a la programación Gráfica con OpenGL
La Salle, 2004

J. Kilgar, Mark
The OpenGL Utility Toolkit (GLUT) Programming Interface
Silicon Graphics, Inc., 1996

S. Wright, Richard et. al.
OpenGL SuperBible
Pearson Education, 2nd Ed,1999

Kopplin, John
Multithreading Tutorial
2006, Internet : http://www.codeproject.com/KB/threads/MultithreadingTutorial.aspx

de Klein, Ramon
Serial library for C++
2003, Internet : http://www.codeproject.com/KB/system/serial.aspx

Sepúlveda, Miguel Angel
GLUT : Utilizando menús
1998, Internet : http://www.linuxfocus.org/Castellano/May1998/article47.html

NeHe's OpenGL Tutorials
2004, Internet : http://nehe.gamedev.net/

Internet :

PC Magazine
miniaturized satellite Definition
http://www.pcmag.com/encyclopedia_term/0,2542,t=picosat&i=56855,00.asp

Wikipedia
Giróscopo
http://es.wikipedia.org/wiki/Gir%C3%B3scopo

Wikipedia
Lithium-ion battery
http://en.wikipedia.org/wiki/Lithium-ion_battery

masadelante.com
Definición de Pixel
http://www.masadelante.com/faq-pixel.htm

Wikipedia
Pila (estructura de datos)
http://es.wikipedia.org/wiki/Pila_(estructura_de_datos)

WolframMathWorld
Normal Vector
http://mathworld.wolfram.com/NormalVector.html

Wikipedia
Autodesk
http://es.wikipedia.org/wiki/Autodesk

Wikipedia
IGES
http://es.wikipedia.org/wiki/IGES

Okino Computer Graphics
3D Studio
http://www.okino.com/conv/exp_3ds.htm

Wikipedia
ASCII
http://es.wikipedia.org/wiki/ASCII

National Instruments
Comunicación Serial: Conceptos Generales.
http://digital.ni.com/public.nsf/allkb/039001258CEF8FB686256E0F005888D1

canal visual basic.net
Manual de la API de Windows
http://www.canalvisualbasic.net/temarios/api1.asp (win32)

Win API con Clase
Programación en Windows
http://winapi.conclase.net/curso/index.php