UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO

ESCUELA NACIONAL DE ESTUDIOS PROFESIONALES CAMPUS ARAGÓN

"ANÁLISIS Y DISEÑO DE UN EDITOR DE TEXTURAS PARA MATERIAL 3D"

T E S I S

QUE PARA OBTENER EL TITULO DE:

INGENIERO EN COMPUTACIÓN

P R E S E N T A N:

GUADALUPE RAMÍREZ AMEZCUA

CARLOS OMAR DE LA ROSA DELFIN

DIRECTOR DE TESIS: M. en C. MARCELO PÉREZ MEDEL

MÉXICO, **2003**

Dedicatorias

A mis padres:

Graciela Amezcua Cortes *Juan José Ramirez Flores*

A quienes me han heredado el tesoro más valioso que puede dársele a un hijo Amor. A quienes sin escatimar esfuerzo han sacrificado gran parte de su vida. A quienes la ilusión de sus existencia ha sido verme convertido en una persona de provecho. A quienes nunca podré pagar todos sus desvelos ni con las riquezas más grandes del mundo. A los seres universalmente más queridos sinceramente gracias.

A mis hermanos:

Juan José Ramirez Amezcua *Victor José Ramirez Amezcua*

Por todo su apoyo que me brindaron en los momentos que más lo necesitaba. A ti Juan por haberme impulsado a ser mejor y a ti Victor porque siempre me contagiabas de entusiasmo y de confianza.

A mis Abuelos:

Esther, Benito, Cecilia, Leobardo.

Por todo el cariño, sabiduría y comprensión que me inculcaron desde niña a pesar de las distancias.

A mi amor:

Juan Gabriel Sosa García

Porque sin tu apoyo, comprensión, y confianza en los momentos mas difíciles siempre conmigo, gracias chiquito por todo.

A mi tío y a mi bisabuela

Crecencio y Elodia

Por todo lo que hicieron por mi, el cariño y amor que me brindaron donde quiera que estén mil gracias

Agradecimientos

A Dios:

Por permitirme llegar a este día con salud.

A la UNAM:

Al C.C.H Vallejo y a la E.N.E.P Aragón y sobre todo a mis profesores que me inculcaron conocimientos para poder llegar a cualquier meta y por todas las bases que nos dejaron para poder caminar en cualquier camino.

A la DGSCA:

Principalmente al departamento de Infraestructura y soporte tecnológico al Ing Justino Peñafiel Salinas y la Quim. Laura Mata Montiel sinceramente muchas gracias por permitirme aprender de ustedes y por inculcarme confianza para que este proyecto llegue a su meta.

A Carlos Omar:

Por que sin tu ayuda y apoyo todo esto hubiera sido muy difícil.

A Marcelo:

A mi asesor definitivamente muchas gracias por tu apoyo en todo los momentos en que estuvimos juntos revisando este trabajo, por tu paciencia y consejos que día a día nos inculcabas GRACIAS.

A todos mis amigos del CCH y de Aragón al igual que a mis compañeros por su amistad y confianza Gracias por siempre estar aquí.

Guadalupe

Dedicatorias:

"Está alegre el camino que somos, caminando para hacer más bueno el camino. Somos el camino para que otros caminen de un lado a otro". Al llegar este momento en el que se cumple una etapa importante de mi vida es una satisfacción el poder disfrutarlo con las personas que hicieron esto posible.

A mis padres:

Carlos De La Rosa Ortega

Maricela Delfín Medina

Por el gran esfuerzo realizado durante todos estos años de estudio, sus desvelos, su confianza, su paciencia, apoyo y cariño incondicional que me han brindado. A ustedes va dedicado este trabajo.

A mi abuelo:

Carlos De La Rosa Rosas

Por el gran cariño que me tuvo y la confianza en mi, me hubiera gustado compartir este momento con el, donde quiera que estés gracias!

A mi abuelita:

Sara Ortega Ramos

Por el cariño, confianza y sus rezos hacia mi

A mi hermana:

Maricela De La Rosa Delfín

Por su apoyo en los momentos difíciles, por que solo ella sabe que cuando titubeamos algo nos impulso a seguir adelante.

A mis primos, tíos y amigos que han hecho que esta meta se cumpla.

Agradecimientos

A Dios:

Por brindarme la salud y guiarme por los caminos adecuados para llegar a esta meta.

A la UNAM:

En especial al CCH-SUR y la E.N.E.P Aragón por inculcarme el amor al estudio y la cultura que me han brindado, así como a mis maestros que han hecho esto posible.

A la D.G.S.C.A, :

En especial al departamento de Infraestructura, al Ing. Justino Peñafiel Salinas y la Química. Laura Mata Montiel, así como a mis compañeros becarios por el apoyo y consejos brindados.

A mi compañera Lupita:

Por que sin su ayuda el lograr este objetivo hubiera sido más difícil.

A mi asesor Marcelo:

Por la paciencia, apoyo y consejos brindados para realizar este trabajo.

A Vero:

Por que sin su apoyo, sus consejos y su confianza en mí, este objetivo hubiera sido muy difícil, gracias por la paciencia en los momentos difíciles.

A mis amigos y compañeros de la universidad, Cuauhtemoc, Oscar, Israel y Luis, a todos ellos gracias por la amistad y el apoyo brindado hasta este momento.

Carlos Omar

ÍNDICE

OBJETIVOS

OBJETIVO GENERAL

Diseñar un editor de texturas para "Material 3D" con la herramienta de programación GLADE analizando las características y antecedentes del modelado en 2D y 3D.

OBJETIVOS ESPECÍFICOS

- Interpretar los antecedentes del modelado 2D y 3D, para generar un objeto el cual tenga un efecto real ante la vista.
- Analizar interfaces gráficas para el desarrollo de GUI'S.
- Analizar diversos editores de texturas para el diseño de un editor de texturas para "Material 3D".
- Implementar y evaluar el editor de materiales para obtener resultados de calidad y mejora del software.

DELIMITACION Y JUSTIFICACION DEL TEMA.

EL trabajo de investigación será de tipo exploratorio y descriptivo, ya que se realizará partiendo de un desglose de información teórico respecto a algunos de los puntos de investigación.

PLANTEAMIENTO DEL PROBLEMA

Uno de los problemas fundamentales de los editores de texturas es que no tienen opciones de parámetros físicos lo cual limita la funcionalidad de los mismos, en algunos de ellos se observa las ventajas y desventajas al estar trabajando con ellos, tales como utilización de recursos de la maquina que trae como consecuencia tardanza en la ejecución del programa.

INTRODUCCIÓN

En la actualidad el desarrollo de software en México y en el mundo está dividido en dos grandes bloques que están en constante lucha para obtener el mayor número de usuarios, por un lado el software comercial que surge con el inicio de las computadoras, un software que brinda cierta garantía sobre sus productos pero que su precio es bastante elevado, por otro lado está lo que se le denomina software libre, un proyecto que inicia a principios de los 90's con la creación del Sistema Operativo LINUX por Linus Torvalds, un proyecto que surge con la ideología del desarrollo de software gratuito, liberando el código fuente para su utilización no comercial, este proyecto ha sido apoyando por programadores de todo el mundo, en aproximadamente diez años se ha logrado desarrollar un S.O como UNIX, WINDOWS ó SOLARIS, pero no sólo se crea el sistema operativo, sino que ahora se crean aplicaciones para éste y en general funcionan en cualquier plataforma UNIX e incluso WINDOWS.

En este proyecto se diseña un editor de texturas para un programa de modelado en 3D, que es desarrollado con la filosofía de software libre, la necesidad de que Material 3D tenga su propio editor de texturas, así como el deseo de participar en el desarrollo de aplicaciones gratuitas son los motivos que nos llevan a diseñar esta aplicación, esperando que sea de utilidad no sólo para M3D sino para cualquier aplicación que este interesada en un editor de texturas gratuito para LINUX o su utilización independientemente para generar texturas con extensión .pov.

En el capítulo uno se realiza una revisión de los conceptos básicos empleados en el modelado en 2D y 3D, esto es con el fin de que se tenga información necesaria para una comprensión más clara en la graficación y del modelado de objetos. Uno de los objetivos principales del modelado es el realismo que se puede llegar a obtener utilizando diversas formas geométricas y criterios de iluminación, sombras, texturas y colores entre otras características esenciales.

En el capítulo dos se explica el desarrollo de las interfaces gráficas utilizando GTK+, desde sus inicios, sus creadores y sus funciones, dando como referencia los lenguajes de programación utilizados tales como (C, C++, Perl 5, Python, Ojective C, Scheme y Ada 95), acerca de GTK+ se menciona lo que es X-Windows, que es estructurado desde un nivel muy bajo de programación.

Se menciona un poco la historia de GNOME y de la comunidad Linux ya que con ello la herramienta de GTK+ se libera bajo la licencia de GPL, acerca de la distribución de LINUX se da la referencia de la versión inicial de la misma dando la oportunidad a los desarrolladores para crear aplicaciones en todo el mundo.

Es de importancia mencionar la ergonomía ya que es fundamental en la adecuación de los productos, sistemas y entornos artificiales en las características, limitaciones y necesidades de sus usuarios, con el fin de optimizar su eficiencia, seguridad y confort, para el desarrollo de herramientas, sistemas, software e interfaces gráficas.

Sobre el desarrollo de interfaces gráficas es de gran importancia mencionar el uso de la herramienta de programación para el desarrollo de GUI'S GLADE que esta basado en GTK+, dando un enfoque hacia la utilización de la herramienta, ya que este trabajo esta desarrollado con ella.

El capítulo tres está enfocado hacia el análisis y desarrollo de un editor de materiales, aquí se describen algunos editores de materiales que se encuentran en algunos modeladores de 2D y 3D, entre los más comerciales y de prestigio son 3D STUDIO MAX, AutoCAD, MORAY y AC3D.

Los materiales en este editor son de gran importancia ya que para obtener la representación de un material tal como un mármol, se necesita proyectar una imagen de 2D que sea compatible con la textura que se tenga como referencia, la imagen que se proyecta tiene varios formatos tales como BMP, TGA, etc.

Acerca del diseño del editor de texturas para "Material 3D" se describe el porque de la necesidad de desarrollar el programa y su importancia para el mismo, se diseño pensando en desarrollar una aplicación de software libre que ayude en la creación de materiales para este programa, sin embargo este puede ser utilizado de forma individual o por cualquier aplicación que sea compatible, el código fuente queda liberado con la intención de que sirva a otras personas y pueda ser modificado para otras aplicaciones.

Por último en el capítulo cuatro se implementa y se evalúa un editor de texturas para "Material 3D". En la primera parte se describe paso a paso el desarrollo, la creación y el funcionamiento del mismo; es de gran importancia mencionar la utilización del manual API ya que aquí se encuentran las funciones de los widgets ya que cada uno de ellos tiene un conjunto de funciones las cuales dependiendo estrictamente de la respuesta que se quiere obtener con el widget programado, se mencionan los códigos de cada uno de los widgets utilizado desde que se desarrolla la interfaz hasta la programación.

Al final de este trabajo se evalúa el proyecto habiendo obtenido los resultados que nos planteamos al principio del desarrollo de éste.

CAPÍTULO I
ANTECEDENTES DE MODELADO 3D

Uno de los objetivos esenciales de modelado en 3d es el realismo del objeto en la imagen. Para poder generar un modelo real es necesario tomar formas geométricas o primitivas; estas formas pueden ser polígonos, poliedros, esferas, etc, estas son tomadas como base para generar otros objetos. En el realismo del modelo es necesario tomar en cuenta otros criterios como son: el color, la iluminación, la textura, la sombra etc.

El modelado también lo podemos ver en 2 dimensiones, con base en las coordenadas X y Y del plano cartesiano, esto representa la anchura y la altura del objeto a modelar, el modelado en 2D no llega a ser muy realista en comparación al modelado en 3D, ya que este último utiliza una coordenada más "Z", que es la profundidad del objeto, también se debe tomar en cuenta que se le agrega color, textura, y de esta forma el modelo llega a ser muy realista y agradable a la vista, que por lo general es lo que todos buscamos en una imagen. El grado de realismo toma en cuenta el software y hardware del sistema, pero se debe tener en cuenta que entre más realista sea el objeto, más recursos de la computadora se utilizan.

1.1 CONCEPTOS BÁSICOS

En este capítulo se tratan los conceptos que se utilizan en modelado 2D y 3D, esto es con el fin de que el lector tenga información que le permita una comprensión más clara, respecto al modelado y a la graficación por computadora, ya que algunos de los conceptos no se utilizan cotidianamente.

1.1.1 Píxel

Es la unidad mínima que se puede representar tanto en pantalla como en impresión, píxel es la abreviatura de las palabras inglesas Picture Element, elemento de dibujo que conforman una imagen. Los píxeles de color están formados por la mezcla de 3

Colores, rojo, verde y azul, esta mezcla es la base que genera todos los colores de la pantalla, la representación de un píxel se muestra en la figura 1.1

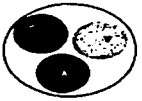


Figura 1.1 Representación de un pixel

1.1.2 Resolución

Es la cantidad de píxeles por unidad de medida, mostrados horizontal y verticalmente, cuando la resolución es mayor, aumenta el detalle o calidad de la imagen, obteniendo para esto un menor tamaño del píxel y una mayor cantidad de los mismos, por ejemplo cuando tenemos una resolución de 800*600 en una pantalla se refiere a que hay 800 columnas y 600 líneas, la resolución puede variar dependiendo del tipo del adaptador del monitor, figura 1.2

Figura 1.2 Resolución de una pantalla

En la tabla 1.1 se muestran algunos tipos de adaptador de monitor, la resolución que alcanzan y su color.

ADAPTADOR	RESOLUCIÓN	COLOR
MCGA	320*200	256
EGA	640*200	16
VGA	640*480	16
SVGA	1024*768, 1280*1024, 1600*1200	256,65536, 16.7 millones

Tabla 1.1 Tipos de adaptadores para monitores

Los adaptadores gráficos se encargan de enviar la imagen al monitor, estos utilizan una memoria denominada memoria gráfica o memoria de video, la memoria se encuentra en la tarjeta y es implementada en circuitos VRAM (video RAM), la cual tiene un doble puerto que le sirve para introducir al mismo tiempo el CPU y el secuenciador de la tarjeta, es más rápida y costosa que la DRAM, la DRAM (RAM dinámica) se caracteriza por que su memoria se tiene que estar refrescando en un determinado tiempo, para que no pierda su información.

1.1.3 Punto gráfico

Es la representación de un punto, este tiene posición y área, normalmente es circular y muy pequeño, el tamaño de este punto depende del diámetro del haz de electrones y del cristal de fósforo de la pantalla, el CRT(tubo de rayos catódicos) se puede utilizar 2 sistemas uno derecho y otro izquierdo, los sistemas de computadora por lo general utilizan el izquierdo, este es formado por el eje X que se sitúa en el extremo superior, el eje Y en la orilla izquierda y el origen en la esquina superior izquierda de la pantalla. Las computadoras para localizar un punto utiliza un mapa de memoria el cual se ve como una serie de bytes o una línea recta.

1.1.4 Sistemas de coordenadas

Un sistema coordenado proporciona un marco para traducir ideas geométricas o expresiones matemáticas. En un plano bidimensional, es posible elegir cualquier punto y señalarlo como un punto de referencia, denominado origen. A través del origen se construyen dos rectas numéricas perpendiculares denominados ejes "X"y

"Y". Una orientación del plano se determina por las posiciones de los lados positivos de los ejes[1].

Un plano tridimensional consta de un punto denominado origen, y de tres rectas perpendiculares que pasan por el origen, en la orientación de los ejes "X","Y",y "Z", estos son denominados orientación a la derecha y a la izquierda, estas se pueden determinar por la regla de la mano derecha, esto se refiere a que los dedos de la mano derecha se alinea con el eje "X" positivo y se gira sobre el ángulo más pequeño en la dirección del eje Y, y el pulgar que apunte en la dirección del eje Z, esto es para la orientación positiva y para la negativa es con la mano izquierda. En las figura 1.3 y 1.4 se representan las coordenadas bidimensionales y tridimensionales.

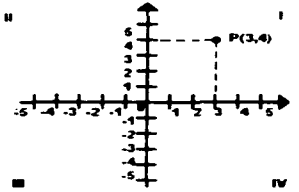


Figura 1.3
Sistema de coordenadas bidimensional

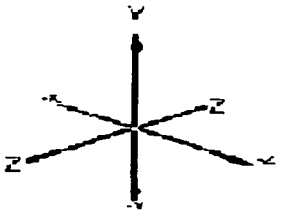


Figura 1.4
Sistema de coordenadas tridimensional

1.1.5 Bit

Es la unidad más pequeña que puede ser representado por un pulso que es enviado a través de un circuito, o un punto en un disco magnético que almacena un dígito binario un 0 o un 1. El sistema binario es el sistema numérico que utilizan las computadoras, un voltaje más alto internamente es representado por un 0 y un

[1] Graficas por computadora autor Roy A Plastock editorial McGraw-Hill pag 265

voltaje bajo lo representa un 1. Cada posición de almacenamiento en la memoria de la computadora se denomina bit.[2]

1.1.6 Byte

Un byte esta formado actualmente por 8 bits, aunque no siempre ha sido así y equivale a un carácter, un número no forzosamente. La cantidad de memoria y de almacenamiento se indica KiloBYTES, MegaBYTES, GigaBYTES. Esto significa que 1k=1024 BYTES, 1m=1.048.576 BYTES o $1024=2^{10}$, etc.

1.1.7 Brillantez

Es la característica de un color, que permite identificarlo con la impresión que es producida por un elemento de la escala de grises, también se refiere a la intensidad de un punto de luz proyectada en un objeto.

1.1.8 Saturación de color

Representa la pureza o la intensidad de un color, por ejemplo los color pastel son los más claros, la saturación esta relacionado muy estrechamente con la brillantez, pureza y la cromacidad, esta última propiedad es utilizada para definir las propiedades del color.

1.1.9 Sombra

La sombra se puede obtener mediante la eliminación de partes ocultas, estas son las superficies que no se pueden observar desde la fuente de luz, en la generación de una sombra se pueden utilizar distintos modelos de sombreado para calcular la intensidad de luz, pero esta, varia dependiendo del modelo de sombreado que se utilice, generando así distintos niveles de iluminación. Para la asignación de los

[2] Graficas por computadora Autor Roy A. Plastock edit. McGraw-Hill pag.276-277.

niveles de intensidad se distribuyen en un intervalo de 0 a 1, donde el 0 significa que una posición del píxel esta apagada, y con el 1 esta en su máxima intensidad.

Las partes que no son visibles y no son iluminadas por una fuente de luz son iluminadas por una ambiental, para esto, las áreas de sombra que proyectan la superficie de un objeto son sombreadas por la luz ambiental.

El modelo de sombreado trata de aparentar el funcionamiento de la luz de un objeto, para generarlo se necesita simular la reflectancia que indica cuanta luz retornara a los ojos, textura, color y transparencia.

1.1.10 Tinte

Los tintes son creados agregando color blanco, esto genera la reducción del color original haciéndolo más claro, y cuando se agrega color negro se genera un color más oscuro.

1.1.11 Tonos

Los tonos son generados agregando blanco y negro a un color, en el caso de medio tono es utilizado en las impresiones, y en las computadoras es para que cuando se despliega una escena.

1.1.12 Luz

Es la radiación electromagnética que se comporta a la vez como onda y partícula, es visible en radiaciones de longitudes de 380 a 740 nanómetros. Este espectro esta conformado desde los rayos Gamma hasta las microondas.

El ojo humano solo puede percibir una porción de la luz, gran parte de esta luz es desviada por la atmósfera. La luz ultravioleta es absorbida por el ozono y la infrarroja

por el vapor del agua. En la figura 1.4 se muestra el espectro electromagnético que permite observar el rango de luz visible para el ojo humano.

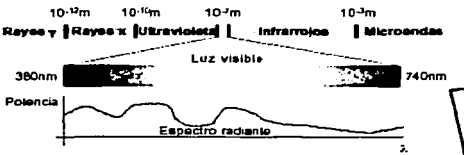


Figura 1.4 Representación de un espectro electromagnético

Los colores que se encuentran en el espectro de la luz visible tienen una gama que van desde los rojos a los violetas, en el intervalo de estos colores encontramos que los colores van desde los rojos a los naranja, y los amarillos a los verdes, siguiendo

los azules y por último a los violeta, las características de la luz son: brillantez, pureza y su frecuencia dominante, estas características definen las propiedades de las fuentes de luz.

1.1.13 Fuente de luz

Es la intensidad de luz que sirve para observar un objeto, encontramos dos tipos de fuentes de luz, las emisoras y las reflectoras, cada una de ellas tienen características especiales, las emisoras son provocadas por la luz solar, la de los focos, etc., y las reflectoras que son generadas por la iluminación de otros objetos, como el reflejo de un espejo.

Para calcular una fuente de luz podemos generarlo con un modelo de sombreado, para producir reflexiones de luz podemos mencionar las reflexiones difusas, que es la luz disipada, en un ejemplo más claro como en los colores pasteles que es muy brillante en todas sus direcciones y la reflexión especular es la que genera un toque de luz, muy brillante en algunos puntos del objeto.

1.1.14 Cubo unitario

El cubo unitario es un paralelepípedo en donde sus caras son cuadros iguales entre si, su volumen esta definido por x=0, x=1, y=0, y=1, z=0, z=1, su valor esta en el intervalo de 0 a 1 y se le conoce como volumen con vista normalizada cuando su volumen es proyectado hacia su plano frontal obteniendo una visión tridimensional.

1.1.15 Textura

La textura es la caracteristica física del elemento, Al añadir una textura al objeto a modelar estamos creando un objeto más real, para esto es necesario tomar en cuenta patrones de color, este describe los colores que encuentra en cada punto de la superficie del objeto, la intensidad que es generada cuando se le agrega un modelo de sombreado modifica la normal de la superficie, esta si es cambiada al azar se puede obtener una textura irregular, para no permitir esto se utilizaria una función de repetición para obtener una superficie regular, esta se modela particionando las superficies en porciones más pequeñas.

1.1.16 Transparencia

La transparencia es cuando el elemento permite la visibilidad de otro que este conjuntamente en una posición en la cual se interponga el elemento que tenga este tipo de textura transparente. Cuando un objeto visible es transparente, se pone en su lugar al objeto al que tapa con el color un poco más débil, para simular la absorción por parte de luz del cuerpo transparente[3]

1.1.17 Bézier

Esta curva fue creada por el Ingeniero francés Bézier, para el diseño de carrocerías de automóvil de la compañía Renault. La curva de Bézier es definida por n+1 puntos de control, con los vectores de $p_k=(X_k, Y_k, Z_k)$, en donde k esta en los valores de 0 a

[3] Programación de gráficos en 3d Autor: Manuel Escribano. Edit Addison_Wesley pag.95

n, desde estos puntos coordenados es calculada la función vectorial P(u), y representa las ecuaciones paramétricas; $P(u)=\sum_{k=0}^{n} P_k B_{k,n}(u)$ [4],donde u esta en el intervalo de 0 a 1, en donde $B_{k,n}(u)=C(n,k)u^k(1-u)^{n-k}$, y C(n,k) es el coeficiente binomial $C(n.k)=\frac{n!}{k!(n-k)!}$.

De la primera ecuación es definida como el conjunto de ecuaciones parametricas para las coordenadas de la curva:

$x(u)\sum_{k=0}^{n} X_k B_{k,n}(u)$ $Y(u)\sum_{k=0}^{n} Y_k B_{k,n}(u)$ $Z(u)\sum_{k=0}^{n} Z_k B_{k,n}(u)$

Una de las características principales de la curva de bézier es que la tangente de uno de sus puntos del extremo esta a lo largo de la línea que une al punto de control adyacente y el extremo, estas curvas no poseen continuidad de segundo orden, Se puede observar una curva de bézier en la figura 1.5

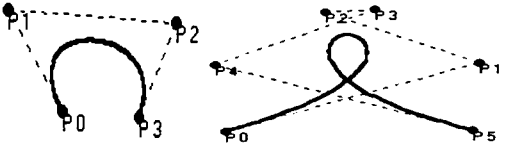


Figura 1.5 Curvas de Bézier

1.1.18 Renderizar

Es la representación de la imagen modelada al término de aplicarle todos lo modelos de iluminación y al utilizar las técnicas de eliminación de partes ocultas.

[4] Ecuación 1

1.1.19 Color

Color es una de las características elementales de los objetos en el cual intervienen la luz para reflejar al objeto. El color es percibido por el sentido de la vista, el cual es uno de los principales sentidos que tiene el ser humano, toda la información que percibimos proviene del sentido de la vista, el ojo humano puede distinguir entre 7 y 10 millones de colores.

La percepción del color se representa como la cantidad de rojo, verde y azul, según Helmholtz, los ojos contienen tres clase de receptores cromáticos, cada uno de los cuales es especialmente sensible a un color concreto, rojo, verde y azul[5].

1.2 MODELOS DE COLOR

Un modelo de color es la base que utilizaremos para crear y definir colores, existen dos modelos muy utilizados: el RGB(rojo, verde y azul) y el CMY(cian, magenta , y amarillo). Estos modelos son utilizados para describir los colores, ya que los diferentes medios manipulan la luz de forma distinta. Cada modelo describe el color de manera diferente, se estableció una norma en 1931 para representar los colores primarios por una comisión Internacional de iluminación (CIE).

La CIE es una organización dedicada a la cooperación internacional y al intercambio de información entre sus miembros, esta organización tiene como objetivos:

1) Proveer un foro internacional para la discusión de todas las materias relacionadas con la ciencia, la tecnología y el arte en los campos de la luz y la iluminación, y para el intercambio de información entre los países en esas materias.

2) Desarrollar patrones básicos y procedimientos metrológicos en el campo de la luz y la iluminación.

[5] Programación de graficos en 3d Autor Manuel Escribano Edit Addsion-Wesley Pag. 82

3) Proveer directivas en la aplicación de principios y procedimientos en el desarrollo de normas nacionales e internacionales en el campo de la iluminación.

4) Preparar y publicar normas, informes y otros textos relativos a las materias propias de la luz y la iluminación.

5) Mantener una conexión y una interacción técnica con otras organizaciones internacionales relacionadas con las materias propias de la ciencia, la tecnología, la normalización y el arte en los campos de la luz y la iluminación.[6]

El modelo CIE se representa en la figura 1.6

Este modelo se determina por la suma de los puntos x+y+z=1 y los puntos que lo rodean por la superficie, es el color de acuerdo a la longitud de onda dada en nanómetros, por ejemplo el punto que se encuentra a la distancia de 400nm es el violeta y el que se encuentra a 700nm es el rojo.

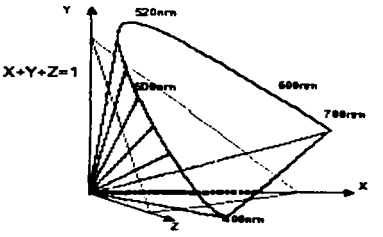


Figura 1.6 Modelo de iluminación CIE

1.2.1 Modelo de color RGB

[6] http://www.ceisp.com/grupos/grupos.html

El modelo de color RGB(rojo, verde, azul), es utilizado por la tecnología de video, como por ejemplo los monitores y escáneres, este modelo es aditivo, es decir forma los colores sumando, empezando en negro y terminando en blanco.

El modelo RGB se representa en un cubo unitario, el origen, que tiene las coordenadas (0, 0, 0) representa el color negro, el punto (1, 0, 0) es el rojo, en el punto (0,1,0) es el verde, el punto (0,0,1) el azul y el blanco tiene las coordenadas (1, 1, 1). En la diagonal entre el blanco y el negro se encuentran la escala de grises, en la figura 1.7 se muestra la representación del cubo unitario.

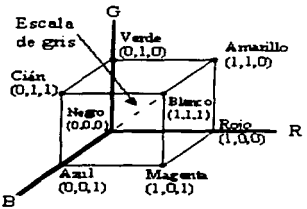


Figura 1.7 cubo unitario del modelo RGB

Los bits de profundidad de color nos ayudan porque indican cuantos bits de información se dispone para poder crear más colores a partir de los primarios (rojo, verde y azul). Se puede decir que a mayor profundidad y cantidad de bits, las imágenes tienen una mejor calidad, pero ocupan más espacio en disco y en memoria, todas las imágenes son de tipo todo color (true color) o color verdadero, se refiere a que por cada píxel se necesita 24 bits o 3 bytes, (8 bits corresponde a 1 byte) para cada canal de color RGB, esto lo podemos apreciar en la tabla 1.2.

BITS	COLOR	NOMBRE
1	2 Tonos	Color
2	4 Tonos	Color
3	8 Tonos	Color
4	16 Tonos	Color
8	256 Tonos	Color
16	65.536 Tonos	Color
24	16.7millones de tonos	Color verdadero
32		Ultra color verdadero

Tabla 1.2 Profundidad de color

Para los valores de 24 bits el color negro es representado por el punto de coordenadas (0, 0, 0), mientras que el blanco se representa en las coordenadas (255, 255, 255), esto se puede observar en la figura 1.8

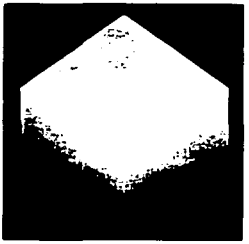


Figura 1.8 Cubo unitario RGB

1.2.2 Modelo de color CMY

El modelo CMY(cian, magenta y amarillo), también es denominado como modelo sustractivo, se refiere a que sustrae o resta el color para formar otros, este modelo es utilizado principalmente en impresoras, a diferencia de los monitores que solo utiliza tres puntos de color, CMY necesita cuatro puntos de color, el cuarto seria el

negro, ya que con la combinación de los otros 3 colores se forman la escala de grises oscuros pero no se obtiene por completo el color negro.

Este modelo se puede representar por medio de un cubo unitario, el cual describe de forma simple el modelo CMY. Dentro de este cubo se representa el negro por el punto de coordenadas (0, 0, 0), el blanco con coordenadas (1, 1, 1). La diagonal entre estos 2 puntos forma la escala de grises., esto se observa en la figura 1.9

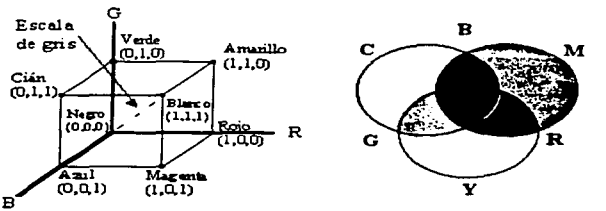


Figura 1.9 Cubo unitario CMY

1.3 REPRESENTACION DE SUPERFICIES

En la representación de superficies se toman en cuenta varias características especiales para modelar objetos tridimensionales, sus características varían dependiendo del modelo que se utilizara para modelar el objeto, estos modelos se representan por medio de mallas poligonales y ecuaciones paramétricas, cada modelo tiene ventajas y desventajas que más delante se mencionan.

1.3.1 Modelado por ecuaciones paramétricas

Este método se basa en ecuaciones matemáticas explícitas o implícitas, la ecuación de una curva determina las coordenadas que la conforman, se puede expresar en 3 coordenadas, un punto de la curva se representa en funciones vectoriales, se puede ver que se generan con 2 parámetros u,v, en los cuales la posición de las

coordenadas de la superficie es representado por la función p(u,v)=(x(u,v),Y(u,v),z(u,v)) los puntos x, y, z son cualquier punto en el plano, los parámetros tienen que estar dentro del intervalo de 0,1 en el cual u es la línea de la longitud constante de la superficie y v es la longitud constante, en la tabla 1.3 se describe las ventajas y desventajas de este modelo.

Ventajas	Desventajas
Ocupan muy poco espacio	Depende por completo del procesador en la creación de imágenes
Su nivel de detalle es muy alto	Es muy difícil de modelar

Tabla 1.3 Ventajas y desventajas del modelado por ecuaciones paramétricas

1.3.2 Modelado de mallas poligonales (marco de alambre)

En este método la imagen se va creando por la concatenación de los polígonos de diferentes o iguales tamaños. Para lograr mayor definición de los modelos, es necesario aumentar el número de partes de los polígonos y disminuir el grueso de estos, este método se utiliza en programas como 3D studio para permitir la creación fácil de los modelos.

Cuando se exhibe un modelo mediante este método se puede mejorar dividiendo la superficie en caras, se organiza la información del objeto en tablas, cada una en vértices, polígonos y aristas, en la tabla de vértices se colocan las coordenadas de los vértices del objeto, en la de polígonos, los polígonos que conforman el objeto y por último el de las aristas, donde se colocan los vértices de ellas.

Si no se tuviera estas tablas se tendrían que enlistar muy estrictamente las coordenadas de cada una de las características del objeto.

En la representación de una red poligonal se enlistan los polígonos, cada vértice es almacenado en $V=(p_0,....p_N)$, el modelo puede desplegarse mediante el uso de los datos de las tablas de las aristas, esto para que se tracen los componentes de las

líneas, si alguna de las tablas faltara se utilizaría las restantes, pero en algunos casos las líneas serían trazadas dos veces.

Los modelos de marco de alambre están compuestos por líneas rectas, para generar modelos principalmente en el sector de la ingeniería, estos modelos son manejables en las transformaciones geométricas, ya que al construirlos es necesario utilizar un gran número de polígonos para que el modelo tenga uniformidad, las principales ventajas y desventajas de este modelo se describen en la tabla 1.4

Ventajas	Desventajas
Son fácil de modelar	Ocupan mucho espacio
Utilizan las tarjetas aceleradoras, para optimizar la creación de triángulos	Sus modelos tiene un bajo nivel de detalle

Tabla 1.4 Ventajas y desventajas del modelado de mallas poligonales

Una de las formas que se utilizan para representar este modelo es el listado explicito de la arista, aquí se obtiene una lista de las aristas el cual es almacenado solo una vez, cada una de ellas apunta a los vértices, en cada una de las caras de los polígonos tienen 2 lados uno interior y el otro hacia fuera a la vista del observador, se puede formar cualquier forma utilizando una serie de poliedros, al formar un objeto tridimensional es muy difícil ya que se necesitan una gran cantidad de datos, para reducir esto es necesario utilizar o partir de una forma primitiva, estas son las formas básicas para formar imagen, en la figura 1.10 se observa el modelo de mallas poligonales.

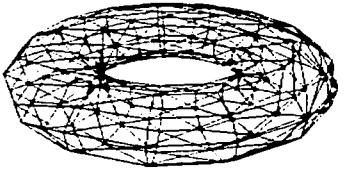


Figura 1.10 Modelo de mallas poligonales

1.4 MODELOS DE ILUMINACIÓN

En esta etapa se estudiarán los modelos de iluminación que contendrán los objetos a modelar, la elección del modelo de iluminación varia dependiendo de las características que serán representadas en cada caso.

El resultado de aplicar un modelo de iluminación especular es el sombreado, para esta reflexión la superficie del modelo es más brillosa refleja la luz de incidencia, produciendo una mancha del mismo color.

En el caso de la reflexión difusa la luz ambiente genera una iluminación uniforme, para la cual en cualquier punto es visible.

1.4.1 Modelo de iluminación constante

Este modelo funciona por medio de la evaluación del producto punto de los vectores, un vector que describe cada uno de los puntos de la superficie de la imagen y el otro es el vector que describe el ángulo en que la luz llega a la imagen.

El modelo de iluminación constante o (Flat), es el modelo más simple para sombrear, flat llena la superficie de un color en el polígono, cada cara se le aplica una iluminación con un valor medio para todo el polígono, esto es para que tenga un solo color y así no permite que sea sombreado de manera lisa., el objeto se sombrea para que sea realista utilizando el modelo de iluminación constante, cuando no se le aplica un color, una textura, sombra etc, este modelo puede ser capaz de modelar el objeto dejándolo muy real, esto es cuando se expone a una luz ambiente, cuando se expone a un punto de luz, esto genera una intensidad de una superficie constante.

EL modelo es el adecuado para los objetos poliédricos ya que los cambios son altos en la inclinación de sus caras, en la orientación de los planos adyacentes cambia

generando un efecto de aspereza, y se puede arreglar variando la intensidad en la superficie, tomando en cuenta la interpolación.

Las superficies curvas son definidas como superficies planas, las cuales se particionan en planos más pequeños, especialmente cuando la luz esta muy alejada del objeto, esto se observa en la figura 1.11

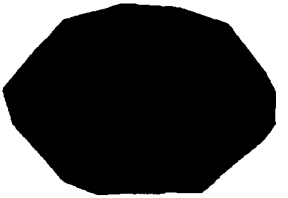


Figura 1.11 Modelo de iluminación constante

En este modelo al aumentar los polígonos y la resolución de los mismo, ocasiona el aumento del efecto de las bandas de Mach, este efecto fue descubierto por el doctor austriaco A. Mach y consiste en que aumenta la claridad o la oscuridad de los colores claros u oscuros, obteniendo un resultado del objeto modelado peor, ocasionando que se definen como formas irregulares ocasionadas por los puntos de luz.

1.4.2 Modelo de iluminación GOURAUD

Este modelo sombrea tomando en cuenta un sistema de puntos de un polígono (un triangulo), le interpola la intensidad de la luz para los demás puntos, utiliza la intensidad de cada cima del triangulo, la interpola linealmente a lo largo de los bordes, la intensidad es tomada usando la iluminación de la normal de la cima. Gouraud puede eliminar casi totalmente las bandas de Mach, la interpolación de la intensidad de luz en la cima aproxima los datos para el resto de los puntos generando gradientes lisos de color, Gouraud sombrea en tiempo real de las

esquinas del polígono de la cima, estas líneas tienen que ser lo más aproximadas a las superficies para poder encontrar la normal.

El modelo también es conocido como interpolación de intensidad, reduce las diferencias de color de las caras más cercanas entre si haciendo que los polígonos parezcan curvas, esto se puede observar mejor en la figura 1.12

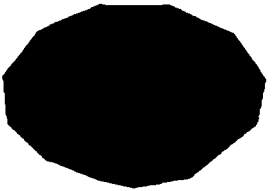


Figura 1.12 Modelo de Gouraud

1.4.3 Modelo de iluminación PHONG

Este modelo fue creado por Phong Bui Toung, este modelo genera la proporcionalidad del coseno$^{n}\phi$, donde n determina la superficie que será observada, este valor será determinante para la intensidad de la luz, entre más brillosa sea, el valor de n será más grande, en caso contrario cuando sea más oscuro, el valor disminuye a 1, este modelo esta relacionado con la reflexión especular, este modelo es más avanzado que los modelos Gouraud y Flat, pero casi no se puede utilizar en tiempo real.

Se necesita encontrar la normal para cada cima, esta se interpola para encontrar la normal, entonces la interpolación se hace lineal y así encuentra la intensidad, en Phong se encuentra toques de luz más exactos de diferentes fuentes de luz, Utiliza

un método que cuando es mayor la energía es más brillante el toque de luz y más pequeño en el material.

El sombrear de phong se refiere a encontrar el valor exacto del color de cada píxel, primero determina un vector normal en cada cima del polígono utilizando Gouraud, después interpola los vectores normales a lo largo de los bordes de la superficie del objeto, interpola los vectores normales para obtener un vector para cada píxel, en este caso utiliza el producto punto del vector en el píxel y del vector ligero para determinar el color.

1.4.4 Modelo de trazo de rayos

Este modelo "dispara" rayos en la escena para calcular las imágenes, esta escena esta formada por cámaras, materiales y fuentes de luz, para cada píxel de la imagen, esto es para calcular el color de la superficie del punto. Se calcula la cantidad de luz que viene de cada fuente, esto para saber si el objeto permanece en la sombra.

El trazado de rayos es una de las técnicas del renderizado, el cual trabaja sobre la geometría de los objetos. Se calcula el punto de observación y se traza el rayo a través de cada píxel, empezando desde el primer punto que se encuentra en la esquina superior izquierda y terminando en el ultimo punto que esta en la esquina inferior derecha, llevando un recorrido línea por línea horizontalmente, este rayo al encontrar el objeto aplica un color del mismo tono al que se esta tomando en la escena, se puede mostrar el trazado de rayos en la figura 1.13.

Figura 1.13 trazado de rayo en una escena

El trazado de rayos es un método en el cual a los cálculos principales se les denomina pruebas de la intersección del rayo llamados rayos reales, el trazo permite gráficos muy hermosos y su algoritmo es de gran alcance, para representar las iluminaciones tales como sombras, reflexión especular y la refracción de la luz que viaja a través de los materiales transparentes.

Podemos encontrar varios métodos de trazado de rayos, uno de ellos es el de trazado de rayos hacia delante, que es cuando se trazan los rayos desde un punto inicial, ósea desde el origen, hasta que se pierdan los rayos que son dirigidos hacia el objeto de la escena, por ejemplo cuando tenemos una fuente de luz sus rayos apuntan hacia el objeto y son rebotados por el en la escena y llegan al observador, los rayos que no alcanzan al objeto se pierden, algunos de los rayos de la fuente de luz son blancos y cuando se intercepta con el objeto transporta entonces el color del objeto modelado de la escena hacia el observador, otro método es el de trazado hacia atrás, en este caso traza los rayos desde el objeto y busca su origen, trata de saber que tipo de color es el rayo de la fuente de luz que sale de la escena a través del rayo, que es trazado desde el ojo, este método se divide en 4 grupos: rayos primarios, rayos reflejados, rayos transmitidos y los rayos de iluminación, los rayos primarios son los que son trazados desde el observador, pasan por un pixel y hasta que lleguen a la escena a modelar, los rayos transmitidos son idénticos a los reflejados, aquí el color será el que la fuente de luz determine, en el caso de los rayos reflejados el color será determinado por la reflexión del rayo primario, esto quiere decir que el rayo secundario toma el lugar del primario buscando el color y

interceptándolo con el objeto, y los rayos de sombra, se traza un rayo primario el cual tiene que tocar la superficie del objeto y determinar si ese punto se encuentra en la sombra o no, eso es para que se encuentre el color que se vera por el observador.

El árbol de rayos traza uno primario, se intercepta en el objeto y a continuación se trazan 2 de sombra en dirección a la fuente de luz, cuando uno de estos se salga de la escena determina el color del rayo primario, este método es muy importante para representar modelos de iluminación y métodos de sombreado.

CAPÍTULO II
DESARROLLO DE INTERFACES GRAFICAS EN GTK+

Actualmente las interfaces gráficas son herramientas indispensables en el desarrollo del software. Los usuarios se han acostumbrado a manejar software con interfaces gráficas y el desarrollo de software para LINUX no puede ser la excepción, así que se empiezan a desarrollar herramientas para hacer mas fácil el desarrollo de interfaces graficas, GTK+ tiene toda una gama de herramientas para el desarrollo de interfaces que ayudan a la construcción de éstas, en este capítulo se mencionan las características de GTK+, la herramienta de programación GLADE, así como los elementos a tomar en cuenta para el diseño y desarrollo de interfaces graficas.

2.1 QUE ES GTK+

Las siglas de GTK+[7] quieren decir "GIMP TOOOLKIT" que significa conjunto de herramientas GIMP[8], GTK+ en sus orígenes fue creado para el desarrollo de GIMP, sus principales autores son: Peter Mattis, Spencer Kimball, Josh MacDonald, el GIMP es un potente programa de manipulación de gráficos del proyecto GNU[9], de ahí su nombre GNU IMAGE MANIPULATION PROGRAM.

GTK+ es una biblioteca del lenguaje de programación C y está orientada a objetos, esta biblioteca está diseñada para el desarrollo de aplicaciones gráficas o mejor dicho para el desarrollo de interfaces gráficas de usuario, tiene la característica de soportar varios lenguajes de programación como son: C, C++, Perl 5, Python, Objective C, Scheme y Ada 95.

[7] http://www.gtk.org/
[8] http://www.gimp.org/
[9] http://www.gnu.org/

2.1.1 Bibliotecas en GTK+

Existe una estructura de diferentes bibliotecas en las que esta basada GTK+, si las empezamos a mencionar desde el nivel mas bajo de programación que existe en interfaces graficas de usuario debemos mencionar lo que es X-WINDOWS, un sistema de ventanas orientado a red, este sistema de ventanas es independiente del hardware y del sistema operativo por eso es que se le considera como la base o estándar para el desarrollo de interfaces graficas de usuario.
Programar en X-WINDOWS es realmente complejo por el nivel de información del programador con el hardware. Otra biblioteca básica es la de GLIB que es una biblioteca de funciones y definiciones para usarse con aplicaciones creadas en GDK y GTK, tiene la propiedad de mejorar la portabilidad de los programas, los tipos que se utilizan en GLIB son muy parecidos a los de C estándar, esta biblioteca es muy útil para el manejo de listas doblemente enlazadas y con enlace simple, temporizadores, manejo de cadenas, un rastreador léxico y funciones de error.

Siguiendo con el orden planteado ahora mencionaremos la biblioteca GDK que significa GIMP drawing kit, esta incluye las funciones de bajo nivel que debe haber para acceder al sistema de ventanas en el que se está programando. En el caso de X-WINDOWS es XLIB, las XLIB son una interfaz de programación en lenguaje C estándar que permite desarrollar aplicaciones para X-WINDOWS, en general se puede decir que son una serie de bibliotecas que utiliza X-WINDOWS.

El último nivel de estas bibliotecas la conforma GTK que ya es la interfaz para la programación de aplicaciones orientadas a objetos (la API), esta también esta escrita en C estándar pero con una idea orientada a objetos haciendo uso de clases, punteros y funciones respuesta, llamadas Callbacks que no son otra cosa más que punteros a funciones.

Todo este conjunto de bibliotecas es el que conforma GTK+, una forma sencilla de ilustrarlo es con la imagen de la figura 2.1. donde se puede observar la dependencia de cada una de las bibliotecas con las demás. Las bibliotecas básicas son X-WINDOWS y GLIB, estando en un nivel superior se encuentra GDK que puede trabajar bajo X-WINDOWS o GLIB y posteriormente se encuentra GDK que a su vez trabaja directamente con X-WINDOWS y también lo puede hacer con GLIB, así las aplicaciones que se desarrollan bajo este entorno pueden estar llamadas directamente en GTK, GDK o GLIB, lo que varia es el nivel de programación, en el caso de GTK cuenta con el soporte del resto de las bibliotecas.

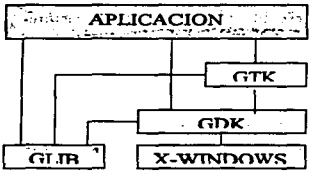


Figura 2.1 Niveles de aplicación de GTK+

2.1.2 Aplicaciones de GTK+

Actualmente GTK+ no sólo ha servido para el desarrollo de GIMP. Uno de los proyectos más importantes en el que se utiliza GTK+ es la interfaz que las aplicaciones de GNOME[10] utilizan para interactuar con el usuario.

Vale la pena mencionar un poco lo que es GNOME, por que su historia ejemplifica la lo que es el software libre y la programación en GTK+ nace con esa idea. Lo primero que tenemos que saber es que quiere decir GNOME, significa GNU NETWORK OBJECT MODEL ENVIRONMENT, entorno de trabajo en red orientado a objetos, en otras palabras es el escritorio GUI del proyecto GNU.

[10] http://www.gnome.org/

Todo comienza cuando uno de los desarrolladores de interfaces gráficas de usuario TrollTech desarrolla una herramienta llamada Qt que es muy buena para el desarrollo de interfaces gráficas de usuario en LINUX pero el tipo de licencia que usa tiene una restricción, esta dice que las aplicaciones desarrolladas bajo esta licencia deben ser gratuitas a menos que adquieran la licencia comercial. Esto impide que muchos programadores utilicen esta herramienta, muchos de los desarrolladores de LINUX no están de acuerdo con este tipo de políticas ya que LINUX se distribuye bajo la licencia GNU que permite que cualquiera que quiera, puede bajar el código fuente modificarlo y posteriormente distribuirlo, así que la comunidad de LINUX desarrolló la herramienta GTK+ que esta liberada bajo la licencia LGPL, una licencia abierta y libre para todos los desarrolladores independientes de la licencia que utilicen para su propio software ya sea comercial o de otra índole.

¿Y todo esto que tiene que ver con GNOME? Pues el escritorio de LINUX KDE[11] está desarrollado con la herramienta Qt, por lo que se tienen restricciones para tener acceso al código fuente de esta interfaz, entonces los desarrolladores de GNOME utilizaron GTK+, para seguir con la política con la que nació LINUX.

Para terminar con lo que es GTK+ hay que destacar que su distribución empieza con la versión 1.2, y en estos momentos se está distribuyendo la versión 2.0 que implementa el manejo de otras bibliotecas como son:

- gdk-pixbuf es la biblioteca que permite el tratamiento de (carga, visualización, grabación) de imágenes gráficas en distintos formatos (png, gif, jpeg, etc).

- Pango es una biblioteca que sirve para que usuarios de diversos países utilicen las aplicaciones realizadas en GTK+ en su propio idioma, puede funcionar a la cabeza de múltiples sistemas de ventanas, facilita la representación y visualización de texto internacional en pantalla. Con pango se puede representar caracteres en distintos alfabetos (hebreo, tailandés

[11] http://www.kde.org/

coreano, árabe, chino, etc), esta biblioteca se encarga de tomar en cuenta detalles de cada tipo de escritura como separación entre caracteres o la dirección de la escritura. Cuando se programa en GTK+, el programador solo utiliza caracteres Unicode que es un estándar para la representación de caracteres en múltiples alfabetos, Unicode proporciona un número único para cada carácter, sin importar la plataforma, sin importar el programa, sin importar el idioma.

- ATK que significa Accessibility Tool Kit es una biblioteca que es desarrollada por la empresa SUN[12], esta biblioteca es un conjunto de clases abstractas integradas en GTK+ cuya finalidad es servir de base para el desarrollo de aplicaciones accesibles para personas con deficiencias físicas, con el uso de esta biblioteca se permite el manejo de aplicaciones desarrolladas en GTK+ para personas discapacitadas.

Al introducir estas nuevas bibliotecas en GTK+ se dan facilidades a los desarrolladores de todo el mundo para crear aplicaciones que sirvan a una gran cantidad de usuarios independientemente del idioma y de alguna discapacidad física que pudieran tener.

2.2 INTERFACES GRAFICAS DE USUARIO (GUI´S)

Cuando se habla de una interfaz grafica de usuario (GUI) se refiere a un ambiente de interacción con la computadora de tipo gráfico y no un ambiente de tipo sólo texto.

El término entró en existencia porque la primera interfaz que une a las computadoras y a los usuarios no era gráfico; eran únicamente texto y se apoyaba la introducción de datos a través del teclado, este tipo de interfaz normalmente consistía en

[12] http://www.sun.com/

órdenes que tenía que recordar y contestaciones de la computadora que eran breves.

La interfaz del sistema operativo de DOS es un ejemplo de una interfaz del usuario típica antes de que GUI´S llegaran.

Un paso intermedio que une al usuario entre la interfaz de línea y la GUI era la interfaz menú no-gráfica que permite que el usuario actúe recíprocamente usando un ratón en lugar de teclear órdenes.

Las GUI'S se definen como el tipo de visualización que permite al usuario elegir comandos, iniciar programas y ver listas de archivos y otras opciones utilizando las representaciones visuales (iconos) y las listas de elementos del menú. Las selecciones pueden activarse bien a través del teclado o con el ratón u otro dispositivo de señalamiento.

Para los autores de aplicaciones, las interfaces gráficas de usuario ofrecen un entorno que se encarga de la comunicación con la computadora. Esto hace que el programador pueda concentrarse en la funcionalidad, ya que no está sujeto a los detalles de la visualización ni a la entrada a través del ratón o del teclado.

También permite a los programadores crear programas que realicen de la misma forma las tareas más frecuentes, como guardar un archivo, porque la interfaz proporciona mecanismos estándar de control como ventanas y cuadros de diálogo. Otra ventaja es que las aplicaciones escritas para una interfaz gráfica de usuario son independientes de los dispositivos: a medida que la interfaz cambia para permitir el uso de nuevos dispositivos de entrada y salida, como un monitor de pantalla grande o un dispositivo óptico de almacenamiento, las aplicaciones pueden utilizarlos sin necesidad de cambios.

Ahora la comunicación entre el usuario y la máquina suele realizarse a través de una interfaz de línea de comandos (terminales de LINUX) ó de una interfaz gráfica de usuario (escritorio GNOME).

El uso de las GUI'S es más sencillo que el de las interfaces de línea de comandos. Sin embargo, la introducción de instrucciones con una GUI es más lenta, por lo que las GUI'S suelen tener la opción de emplear un sistema equivalente al de línea de instrucciones como alternativa rápida para los usuarios más expertos.
La mayoría del GUI'S incluye los rasgos siguientes:

- Un dispositivo apuntador, normalmente un ratón,
- Drop-Down menús operados por el dispositivo apuntador.
- Que una pantalla se divide en las ventanas separadas.
- imágenes Gráficas que permiten al usuario saber lo que la computadora está haciendo, y le permite al usuario decirle a la computadora qué hacer.
- Iconos para representar acciones diferentes.

Los sistemas operativos de hoy proporcionan una interfaz de usuario gráfica. Actualmente los sistemas operativos de Windows y Macintosh son ejemplos de GUI'S. En LINUX el entorno de interfaces gráficas puede ser KDE o GNOME. En la figura 2.2 se muestra el escritorio de KDE y en la figura 2.3 se muestra el de GNOME.

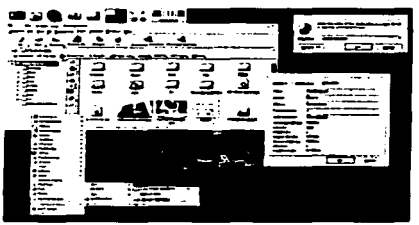


Figura 2.2 escritorio de KDE

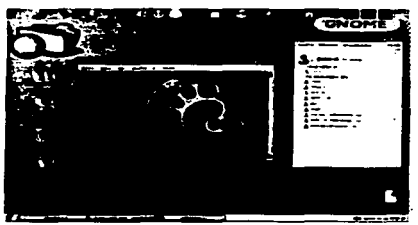


Figura 2.3 escritorio de GNOME

Las aplicaciones usan los elementos del GUI que viene con el sistema operativo y agrega sus propios elementos de interfaz de usuario gráficos. Una GUI a veces usa uno o más metáforas para los objetos familiares en la vida real, como el desktop, la vista a través de una ventana, o el esquema físico en un edificio. Los elementos de un GUI incluyen tales cosas como: ventanas, botones, imágenes del icono, el ratón.

En la figura 2.3 se pueden observar diversos ejemplos de iconos del escritorio de GNOME, aquí se observa como se representa un acceso directo a la pagina de red hat con la imagen que lo representa, o un CD-ROM, un directorio de archivos personales, un disquete de 3½, y una papelería de reciclaje.

Figura 2.4 iconos del escritorio GNOME

La palabra usabilidad se utiliza mucho en el diseño de interfaces graficas así que vamos a dar algunas de sus características:

Efectividad: Que tan rápido se ejecuta un programa
Aprendizaje: El tiempo que necesitan los usuarios para manejar el sistema
Flexibilidad: Mide hasta donde es efectiva la interfaz
Actitud: Prácticamente es como reacciona los usuarios ante la interfaz, si les es fácil o difícil interactuar con ella.

Existen tres tipos de diseño de interfaces:

- Centrado en los datos; que como su nombre lo dice se basa únicamente en los datos.
- Centrado en la tecnología; que considera un mecanismo de entrada y salida para el usuario.

- Centrado en el usuario; este requiere mas esfuerzo que los anteriores ya que se deben interpretar las necesidades frente al prototipo, establecer requerimientos de usabilidad, sus reacciones y hacer modificaciones.

Los pasos básicos recomendables para obtener una buena interfaz son:

- Idea rápida y continua sobre el usuario.
- Diseño integrado.
- Pruebas tempranas y continuas sobre el usuario.
- Diseño interactivo a través de las actividades de diseño, modificación y prueba.

2.2.1 El proceso GUIDE

Este proceso da un marco de referencia para describir la participación del usuario en el diseño de la GUI, se compone de varias actividades, los cuales tienen un objetivo especifico y producen un producto bien definido, éste es el proceso en el cual se diseña y evalúa una interfaz de usuario grafica, buscando aumentar la usabilidad de la interfaz, haciendo otra vez mención de usabilidad nos vamos a referir a la calidad de la interfaz gráfica.

Este proceso utiliza un diseño centrado en el usuario, así que en quien se debe pensar cuando se esta creando una interfaz grafica es en el usuario, adaptando el sistema al tipo de usuario y no al revés que el usuario se tenga que adaptar al sistema.

Las actividades son las siguientes:

1) Definir los usuarios y requerimientos de usabilidad, se basa en las personas que utilizarán el sistema, con esto ayuda a que el diseño GUI se ajuste a los usuarios y a las tareas.

2) Definir el modelo de tarea de usuario, hay que observar como está trabajando actualmente el usuario para entender el ambiente en que se desarrolla y así poder ayudarle.

3) Definir el modelo de objetos de usuario, hay que identificar los objetos que sean utilizados por el usuario.

4) Definir el modelo dinámico, hay que definir el comportamiento dinámico de los objetos de usuario.

5) Definir la guía de estilo, se define el estilo estándar para el diseño de la interfaz, hay que seleccionar un estilo adecuado para los usuarios, esta depende del entorno del Sistema Operativo.

6) Diseño GUI, se diseña la apariencia y el comportamiento visible al usuario.

7) Construcción del prototipo GUI, consiste en tener una herramienta para evaluar el diseño GUI.

8) Evaluación GUI, consiste en evaluar la aplicación cumpliendo con los criterios de usabilidad.

En la figura 2.5 se muestra el diagrama a bloques del proceso GUIDE

2.2.2 Ergonomía

Cuando hablamos de GUI´s y de la calidad que debe existir en ellas tenemos que mencionar lo que es ergonomía, esta palabra se deriva de las palabras griegas ergon", que significa trabajo, y "nomos", leyes, reglas; por lo que literalmente significa "leyes del trabajo"[13].

[13] Oxford, Inglaterra, en 1949 K.F.H Murrel

La asociación española de ergonomía (AEE)[14], constituida en 1964, miembro de la Intenacional Ergonomics Association, plantea una definición que se puede considerar integradora de las diferentes tendencias de la ergonomía y la ingeniería de los factores humanos. "La ciencia aplicada de carácter multidisciplinario que tiene como finalidad la adecuación de los productos, sistemas y entornos artificiales a la características, limitaciones y necesidades de sus usuarios, para optimizar su eficiencia, seguridad y confort".

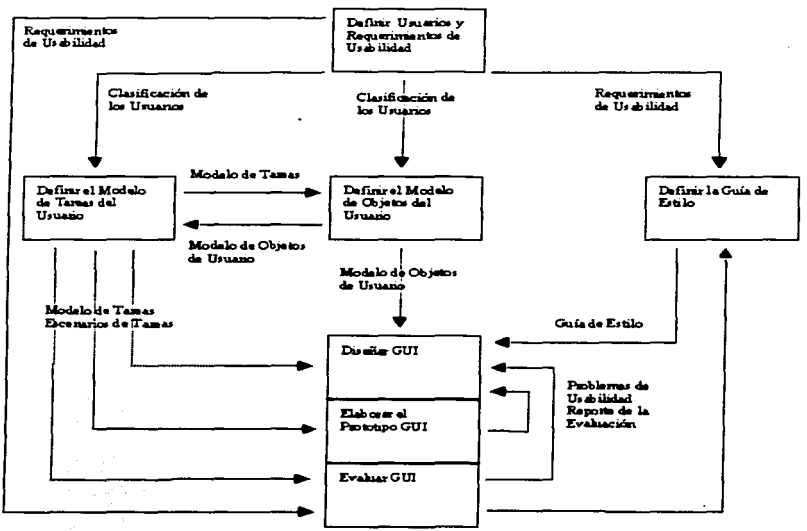


Figura 2.5 Representación gráfica del proceso

[14] http://www.prevencionintegral.com/Default.asp?http://www.prevencionintegral.com/AEE/

Así que cuando se habla de diseño ergonómico se refiere a la aplicación de estos conocimientos en el diseño de herramientas, máquinas, sistemas, tareas, trabajos y ambientes seguros, confortables y de uso humano efectivo, incluyendo software e interfaces graficas de usuario

La historia de la ergonomía como disciplina nace al final de la segunda guerra mundial, cuando las necesidades obligan a los ingenieros, que diseñaban cada vez más complejos equipos, a tomar en cuenta, de una forma explícita y sistemática, las leyes fisiológicas y psicológicas del comportamiento humano.

La primera sociedad de ergonomía "La Ergonomics Research Society"[15] fue fundada en 1949 y fue promovida por Murrell, junto con otros ingenieros, fisiólogos y sociólogos, con el objeto de adaptar el trabajo a las personas. En 1964, se funda la Sociedad Ergonómica de Investigación Científica Japonesa.

Así una vez finalizada la guerra se produjo un renovado interés por las condiciones en que el ser humano desarrolla su trabajo, pero desde un nuevo enfoque, al considerar que la relación hombre - máquina - ambiente, es una relación interactiva en la que los tres elementos han de ser vistos como componentes de un mismo sistema.

Y como dice, Castillo y Prieto, "La novedad de este enfoque va ha dar lugar aún neologismo capas de expresarlo: Ergonomía".

Hoy en día la ergonomía está comprendida dentro de varias profesiones y carreras académicas como son la ingeniería, higiene industrial, terapia física, terapeutas ocupacionales, enfermeras, quiroprácticos, médicos, pero la ergonomía se puede aplicar al estudio de cualquier actividad sea laboral o no, y de las personas que realizan o desarrollan cualquier tipo de tarea.

[15] http://www.ergonomics.org.uk/society/history/bibliography.htm

Existen varias clasificaciones en las que se puede dividir a la ergonomía, aquí se mencionaran dos, una que se refiere a la ergonomía industrial; la cual se encarga de los aspectos físicos del trabajo y ciertas capacidades humanas como son fuerza y postura, la otra se refiere a los factores humanos, así que esta orientada a los aspectos psicológicos del trabajo como la carga mental y la toma de decisiones.

2.2.3 Ergonomía cognitiva

Dentro de las diferentes clasificaciones de la ergonomía nos encontramos con la ergonomía cognitiva que es la que nos interesa en el diseño de GUI'S, porque se refiere al proceso que existe en la recepción de señales y de información, así como la habilidad que se debe tener para procesar y dar una respuesta a esas señales o información.

En la creación de GUI'S se debe tomar en cuenta que hay una interacción entre el usuario y la computadora específicamente el programa que se este utilizando y este programa va a depender del intercambio de información que se da en ambas partes, tanto del usuario como del programa ya que el usuario controla las acciones del programa o de la computadora por medio de la información que introduce y las acciones que realiza sobre este, pero también es necesario considerar que el programa alimenta de cierta información al usuario por medio de señales, para indicar el estado del proceso o las condiciones en que se encuentra el programa.

Esta área de la ergonomía tiene gran aplicación en el diseño y evaluación de software, tableros de control y material didáctico.

2.2.4 Elementos a usar en una interfaz en GTK+

En una interfaz desarrollada en GTK+ el elemento más importante es lo que denominamos un "widget" éste es un elemento grafico con el que el usuario puede interactuar, algunos ejemplos de widgets son los botones, ventanas, áreas de texto, cajas combo, etc., los widgets que se crean van a responder a señales o eventos.

esta es una de las partes fundamentales de la programación de interfaces graficas con GTK+ ya que cada widget debe responder a una señal, el ejemplo más claro de esto es cuando se hace clic en un botón, el hacer clic sobre el botón se activa una señal y esta va a responder a través de una retrollamada a una función que realice la acción que se esta pidiendo, así es como funcionan los widgets por medio de señales y retrollamadas, cuando se tratan las señales hay que registrar las retrollamadas en GTK+ y asociarla con un widget, una gran ventaja es que una misma retollamada se puede registran en varios widgets.

En la figura 2.6 se muestra la relación entre la aplicación GTK+ y las retrollamadas, ambos botones definen retrollamadas que serán invocadas cuando se realice la señal correspondiente, un widget puede definir las retrollamadas y señales que sean necesarias para su ejecución, así un solo widget puede responder a varias señales, como pulsaciones del ratón, tecleos de entre, movimientos del ratón, etc.

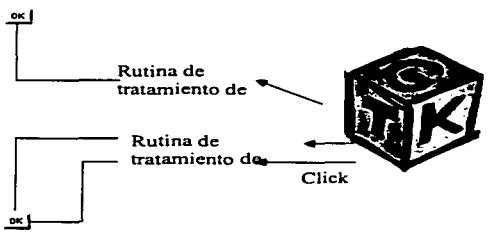


Figura 2.6 Ejecución de señales en GTK+

2.3 PROGRAMACION CON GTK+

Programar en GTK+ es muy parecido a programar en lenguaje C, sin embargo el método de programación que se utiliza debe ser orientado a objetos utilizando punteros a funciones, herencia e incluso polimorfismo, parece increíble que las características que hacen a C++ un lenguaje poderoso las utilice GTK+ como una de sus características de programación basada completamente en C y no C++, pero se

utiliza C porque cuando surge GTK+ el lenguaje C++ no era completamente estable, además de que C sigue siendo el lenguaje utilizado por los programadores y desarrolladores de aplicaciones en LINUX.

Programar en GTK+ es utilizar objetos a los cuales se les va a aplicar una señal, estos objetos van a estar en espera de recibir una señal, los objetos que se manejan en GTK+ no son otra cosa mas que estructuras en C, si manejamos objetos debemos tener herencia y esta se logra incluyendo una estructura padre como primer elemento de una estructura hija.

Para manejar el modelo de objetos en GTK+ se utilizan tres tipos de métodos:

- uno es una función en C que esta esperando un puntero a un objeto como primer argumento, cuando se utiliza este método es él más rápido pero no existe ninguna forma de modificar su funcionamiento.

- Otro es un método virtual que consiste en reemplazar las funciones en C por funciones propias derivadas en clases, esto es a lo que se le llama polimorfismo.

- El tercer método esta basado en señales, este es parecido al método virtual, en estos dos métodos se puede modificar su funcionamiento, este método se realiza conectando manejadores que se ejecutan antes o después del método.

2.3.1 Manejo del ciclo principal

Una vez definidos los elementos básicos a usar en una interfaz gráfica desarrollada en GTK+, es importante mencionar el bucle de sucesos o bucle de escucha bajo el cual funcionan los programas desarrollados en GTK+, todo el modelo de programación de GTK+ esta basado en la programación por eventos a llamadas a funciones, estos eventos marcan un cambio de estado en el objeto, para ejemplificar

esto podemos imaginar un botón (que seria un objeto) en el cual se hace un clic (la señal o evento) y responde a una acción por ejemplo cerrar una ventana (retrollamada a la función que cierra la ventana).

Para que este sistema funcione se usa lo que se denomina el bucle de escucha en el cual entran todas las aplicaciones desarrolladas en GTK+, este es un bucle interno de GTK+ en el que entran las aplicaciones una vez que se hace la llamada a la función gtk_main, así la aplicación entra en un ciclo de espera de una señal o señales registrando estas y haciendo la llamada a las funciones, es importante mencionar que este bucle no termina hasta que se hace una llamada a gtk_main_quit que es la función que nos permite salir del bucle de escucha, si nunca se hace la llamada a esta función la aplicación sigue estando en espera de un evento por lo que nunca termina.

Este bucle tiene sus orígenes en el bucle de ejecución de la biblioteca GLIB con modificaciones para hacerlo funcionar en interfaces graficas.

Otra de las cosas que vale la pena mencionar son los tipos de widgets que hay, los widgets en GTK+ tienen la propiedad de que se pueden derivar de otros widgets de nivel superior, esta propiedad permite la reutilización de código ya que si un widget realiza parte del trabajo de otro widget éste puede heredar las propiedades de otro widget. Todas las funciones de creación de widgets devuelven un puntero a un tipo de dato GtkWidget, que es un puntero a un widget genérico, este puede ser convertido para determinadas funciones.

2.3.2 Compilación de un programa en GTK+

La forma de guardar los programas desarrollados en GTK+ suponiendo que se está programando en lenguaje C es con la extensión.c, pero depende del lenguaje en el que sé este programando, para poder compilar un programa desarrollado en GTK+ es necesario enlazar la aplicación con algunas bibliotecas, los desarrolladores de GTK+ han incluido el programa gtk-config, que permite enlazar automáticamente la

aplicación con las bibliotecas necesarias, así con este programa y el compilador GNU de C (gcc) se puede compilar las aplicaciones desarrolladas en GTK+, la instrucción que se utiliza es la siguiente:

```
gcc –Wall –g ejemplo.c –o ejemplo 'gtk-config –cflags' 'gtk-config –libs'
```

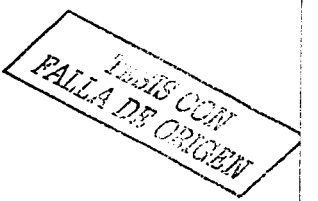
Así el compilados gcc y el programa gtk-config se encargan de compilar el programa (ejemplo.c), si el compilador no marca ningún error se debe crear un archivo ejecutable en este caso con el nombre ejemplo.

2.3.3 Como programar en GTK+

Lo primero que se debe hacer cuando se escribe un programa en GTK+ es incluir el archivo gtk/gtk.h que es el archivo de inclusión de GTK+, después se debe inicializar la biblioteca GTK+ con una llamada a la función gtk init, a dicha función se le deben pasar los argumentos &argc y &argv, de la siguiente forma:

```
gtk_init (&argc, &argv);
```

El elemento principal cuando se está programando en GTK+ son los widgets, estos se definen como punteros a una estructura de tipo GtkWidget, esta estructura es de un tipo de dato genérico y se utiliza por todos los widgets, por ejemplo si queremos declarar una ventana la instrucción queda de la siguiente manera:




```
GtkWidget *ventana;
```

ó si queremos declarar un botón la instrucción es la siguiente:

```
GtkWidget *boton;
```

Otra de las características de GTK+, es que está basada en contenedores, un widget puede ser contenedor de uno o más widgets. El ejemplo más sencillo es un botón. El botón puede ser contenedor de otros widgets, por ejemplo una etiqueta, o un icono

gráfico, pero a su vez el botón podría estar contenido en una ventana principal en GTK+, a la ventana principal se le denomina ventana de nivel superior y estas no están contenidas dentro de ningún otro widget. A esta ventana se le dice que es la ventana padre, en GTK+ existe una relación entre widgets denominada Padre-Hijo donde el widget padre es el contenedor y el widget hijo es el que está dentro del contenedor, un widget hijo puede ser padre de otros contenedores.

Cuando se crean aplicaciones en GTK+ se debe crear una ventana de nivel superior que sea el contenedor de los demás widgets que se van a utilizar en la aplicación, para crear un widget primero se utiliza una función que lo pueda crear y luego una función que sea capaz de mostrarlo en el escritorio, por ejemplo cuando creamos una ventana de nivel superior se utiliza la función gtk_window_new y se le pasa como parámetro GTK_WINDOW_TOPLEVEL esta función devuelve un puntero a un dato de tipo widget, en este momento se crea la ventana pero no es visible para el usuario así que con la función gtk_widget_show se hace visible la ventana pasando como parámetro el puntero de tipo GtkWidget, el fragmento de código para crear una ventana y luego hacerla visible quedaría de la siguiente manera:

```
// llamada al archivo de inclusión de GTK+
#include (gtk/gtk.h)
int main(int argc, char *argv [ ])
 {
  // crear una ventana de tipo GtkWidget
  GtkWidget *ventana;
  // inicializar la biblioteca GTK+
  gtk_init (&argc, &argv);
  // crear la ventana de nivel superior
  ventana=gtk_widget_new(GTK_WINDOW_TOPLEVEL);
  // hacer visible la ventana
  gtk_widget_show(ventana);
```

Después de que se inicializa GTK+, se crean y muestran los widgets, se debe hacer la llamada al bucle de sucesos de GTK+ con la función gtk_main, así se cede el control a GTK+ para que la aplicación quede en espera de algún suceso como un

enter o un clic sobre un botón, una vez que se entra al bucle de sucesos no se puede salir de este hasta que se hace una llamada a la función grtk_main_quit, así que antes de hacer la llamada a gtk_main se debe crear una señal que haga una llamada a gtk_main_quit, que no es otra cosa mas que una retrollamada.

Es importante mencionar la jerarquía que existe en los tipos de datos de GTK+, ya se mencionó que todos los datos van a ser de tipo GtkWidget pero existe una jerarquía que se debe tomar en cuenta, por ejemplo en el nivel mas alto se encuentra un objeto de GTK+ denominado GtkObject de éste se deriva un widget genérico (GtkWidget) y de éste se puede derivar un widget contenedor (GtkContainer) y dentro de éste pueden existir otros tipos de widgets como son los de botones (GtkButton).

Cuando creamos un widget de cualquier tipo las funciones devuelven un puntero a un widget genérico (GtkWidget), esto en ocasiones tiene que ser convertido al tipo de widget adecuado, por ejemplo si una función de creación de botón devuelve un GtkWidget este puede ser convertido a un GtkButton, para convertir tipos de widgets se utilizan macros con el nombre en mayúsculas del widget que quiera ser convertido separado por un guión bajo, en este caso GTK_BUTTON.

Esta conversión se puede utilizar solo si el widget fue creado por una función de creación de botón o algún tipo de widget derivado de el, por ejemplo un contenedor (GtkContainer) utilizando la macro GTK_CONTAINER por que de hecho el botón deriva de un contenedor, aquí se debe tener mucho cuidado porque no se puede convertir widgets que no se deriven de otros widgets, por ejemplo un contenedor no puede pasar como un botón porque no todos los contenedores son botones.

Esto parece un poco complicado pero solo hay que fijarse que un widget derive de otro para poder convertirlo, el tener un widget genérico que podamos convertir a otro tipo de dato es muy útil porque las funciones genéricas que utiliza GTK+ se pueden utilizar para cualquier tipo de dato solo cambiando este, un ejemplo se puede observar con la función gtk_widget_show que se utiliza para cualquier tipo de widget.

2.4 DESARROLLO DE INTERFACES GRAFICAS CON "GLADE"

GLADE es una herramienta de programación para el desarrollo de GUI'S, su creador es Damon Chaplin, GLADE es desarrollado con las mismas normas del proyecto GNU bajo la licencia GPL, lo cual significa que es un software que se distribuye libremente.

GLADE esta basado en GTK+, de hecho está desarrollado con GTK+, cuando se genera una aplicación con GLADE este brinda la opción de generar código (ligado a GTK+) para lenguaje C, C++, Ada95, Perl y Eiffel.

Cuando desarrollamos aplicaciones con GLADE, este genera el código de la interfaz como si se estuviera programando en GTK+, dejando la aplicación lista para programar los eventos que necesitemos, esto nos permite ahorrar tiempo y reutilizar código.

2.4.1 Elementos de "GLADE"

Se puede iniciar GLADE desde el desktop de GNOME en la opción de desarrollo en el escritorio de KDE también se puede encontrar si se instala GNOME, en el desk-up, programas GNOME y desarrollo. También se puede ejecutar desde una terminal escribiendo GLADE y enter.

Cuando se inicia la aplicación se abren tres ventanas, la ventana principal, una ventana de propiedades y una paleta donde se encuentran los widgets que se tienen a disposición para el desarrollo de nuestras interfaces.

En la figura 2.7 se muestra la imagen de la ventana principal, esta contiene los widgets de ventanas que se han agregado, por ejemplo en la figura 2.7 se observa que se ha agregado una ventana GNOME y una ventana de dialogo, aquí también se muestra el nombre del proyecto.

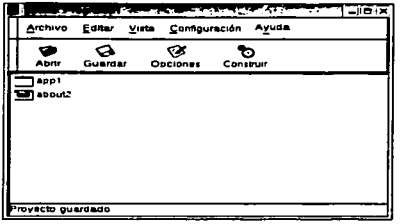


Figura 2.7 Ventana principal de GLADE

En la paleta de widgets existen 3 menús, el de GTK+ Basic, en este se encuentran los widgets de ventanas, barra de menú, barra de herramientas, etiquetas, entradas de texto, botones, un dialogo de selector de color, etc. El menú de GTK+ Additional contiene widgets como el de escala horizontal, vertical, un calendario, barras de desplazamiento vertical y horizontal y por último se tienen los widgets del modo GNOME, aquí encontramos ventanas de aplicación ya hechas, en la figura 2.8 se muestra las tres paletas de widgets que contiene GLADE y en la tabla 2.1 se muestran todos los tipos de widgets existentes en GLADE.

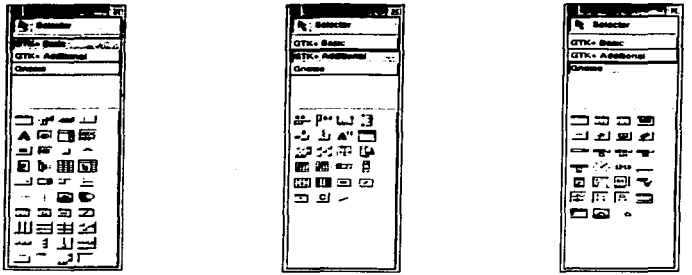


Figura 2.8 Paletas de widgets

La otra ventana que muestra GLADE es la de propiedades, en esta se pueden observar las propiedades del widget seleccionado, en la parte superior aparece el nombre del widget y contiene cuatro pestañas, la de widget, posición, básico y señales, el contenido de estas pestañas varia dependiendo del widget seleccionado, algunos widgets como el de botón en la pestaña de widget contiene el nombre, el ancho del borde, una opción llamada stock button en la que se puede seleccionar una etiqueta para el botón, por ejemplo OK y si la opción esta en none se puede escribir la etiqueta que uno quiera (figura 2.9), en la pestaña de comunes las opciones son básicamente para ver donde va situado el widget (figura 2.10).

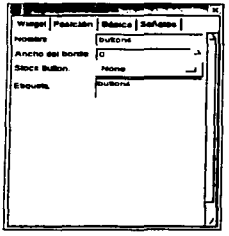


Figura 2.9 Paleta de propiedades (Widget)

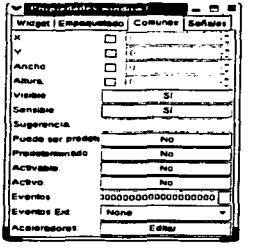


Figura 2.10 Paleta de propiedades (Comunes)

Icono	Descripción	Icono	Descripción
	Ventana		Barra de Menús
	Barra de Herramientas		Caja Flotante
	Etiqueta		Entrada de Texto
	Caja Combo		Caja de Texto
	Boton		Boton de Dos Estados
	Boton de Chequeo		Boton de Radio
	Lista		Arbol
	Lista en Columnas		Arbol en Columna
	Menu de Opciones		Boton de Incremento
	Barra de Progreso		Barra de Estado
	Separador Horizontal		Separador Vertical
	Mapa de Bits		Zona de Dibujo
	Dialogo		Dialogo de Seleccion de Archivo
	Dialogo de Seleccion de Color		Dialogo de Seleccion de Fuente
	Caja Horizontal		Caja Vertical
	Tabla		Posiciones Estaticas
	Caja de Botones Horizontal		Caja de Botones Vertical
	Panel Horizontal		Panel Vertical
	Libreta		Marco
	Ventana de Desplazamiento		Vista

Tabla 2.1 Widgets Basicos

En la pestaña de básico se encuentra su tamaño y donde esta ubicado, existe una opción llamada texto de ayuda que nos permite colocar una referencia al botón, que se puede observar cuando se coloque el puntero del ratón sobre este (figura 2.11), y la pestaña de señales nos sirve para agregar señales que utilizaremos en nuestro widget como por ejemplo un clic, o un enter estas señales también dependen del tipo de widget que estemos utilizando ya que no se pueden aplicar las mismas señales a todos los widgets (figura 2.12).

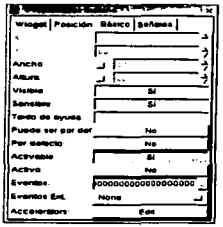


Figura 2.11 Paleta de propiedades (Básico)

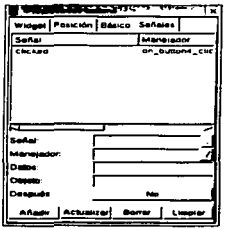


Figura 2.12 Paleta de propiedades (Señales)

Una vez que se crea una aplicación en GLADE se debe guardar y posteriormente construir el código de la interfaz, cuando se crea el código, GLADE genera varios archivos unos que podemos modificar y programar y otros que no deben ser modificados, se genera un directorio con el nombre del proyecto donde se tiene la aplicación, y dentro de este se crea otro llamado src en el que se encuentran los archivos de la interfaz, aquí se pueden encontrar los archivos callbacks.c y callbacks.h. Para compilar se debe presionar la opción construir y después de establecer los parámetros requeridos se debe abrir una terminal y colocarnos en el directorio del proyecto ahí debemos ejecutar el comando./autogensh, que es la instrucción para compilar nuestro programa, después debemos teclear make para terminar de compilar y generar el programa ejecutable, si no detecta errores se creara un archivo ejecutable dentro del directorio src con el nombre que se le da a la aplicación, este programa se puede ejecutar desde ese mismo directorio con la instrucción./nombredelaaplicacion.

CAPÍTULO III
ANALISIS Y DISEÑO DEL EDITOR DE MATERIALES.

En este capítulo se describen diferentes tipos de editores de materiales, algunos incluidos en programas de modelado en 2D y 3D, otros que funcionan de forma independiente, con el fin de tener una idea clara de que es lo que se necesita en un editor de materiales, en la actualidad existen muchos programas modeladores de 3D y cada uno de ellos tiene su propio editor de materiales o al menos así sucede con la mayoría. Se mencionan algunos editores de programas comerciales de gran prestigio como 3D STUDIO MAX, AutoCAD, MORAY, AC3D, algunos se distribuyen de forma libre o no comercial y funcionan tanto en plataforma LINUX como WINDOWS, además se describirán sus ventajas y desventajas.

3.1 EDITOR DE TEXTURAS DE AutoCAD.

AutoCAD es un programa que fue creado para el dibujo en 2D, pero actualmente las necesidades de crear formas reales han obligado a posibilitar la creación de objetos en 3D, así que ahora cuenta con herramientas para crear objetos en 3D, y por supuesto su propio editor de texturas para dar mayor realidad a sus objetos.

Mediante las órdenes vinculadas al RENDER[16] de AutoCAD se pueden representar imágenes de modelos 3D, con un aspecto casi real, teniendo gran control respecto a los materiales, la situación de los puntos de luz y el tipo de iluminación.

Cuando dibujamos en papel o cualquier otro material empleamos diferentes técnicas de color como lápices de color, acuarelas, aerógrafo, plumones, etc., todo esto con el objetivo de darle al objeto que dibujamos una mayor realidad, exactamente éste es el propósito de render en AutoCAD, darle mayor realidad a los objetos que se dibujen ayudándose de los materiales, sombras, luces, etc.

[16] Es la conversión de un trazo de líneas en una imagen tridimensional a través de modelos matemáticos, como introducir variaciones de color, sombreado que se producen por cambios específicos de luces sobre un objeto. Webster's new world. Diccionario of computers. Eighth edition. Autor: Bryan Pfaffenberger. P.P 452 y 462

3.1.1 Tipos de renderizado.

En AutoCAD se utilizan tres tipos de renderizado que son:

Render básico.- Corresponde al único renderizado que existía en versiones anteriores a la 14, es el más rápido de todos, pero presentando una calidad bastante inferior a las dos restantes.

Render fotorrealistico.- emplea la técnica de líneas de barrido, permite la utilización de materiales y transparencias así como generar sombras volumétricas.

Render raytracing.- es el que da mejores resultados al emplear la trayectoria del rayo para calcular las reflexiones y las sombras.

Para acceder de forma rápida y fácil al render es recomendable activar la barra de herramientas, se activa desde cualquier barra, y seleccionando con botón derecho, así se tiene acceso rápido a las diferentes opciones, como librería de materiales, luces, editor de materiales, etc., en la figura 3.1 se muestra la barra de herramientas de render.

Figura 3.1 Barra de herramientas de render

En AutoCAD Existen tres formas de ejecutar los comandos, la más sencilla es hacer clic en el icono correspondiente, la otra es buscar la opción en el menú y la otra es escribir el comando en la parte inferior izquierda de la ventana principal.

3.1.2 Colores en AutoCAD

AutoCAD utiliza el modelo de color RGB en forma conjunta con HLS que determinan la pureza, luminosidad y la saturación del color, el color es una de las propiedades más importantes de los materiales para lograr el realismo en los objetos, por ejemplo, una bola de billar de color azul, si se observa detalladamente se pueden distinguir variaciones o diferentes matices del color azul como consecuencia de las sombras y de la luz que se le aplique así como de la geometría del objeto.

En AutoCAD los colores se distribuyen de la siguiente forma:

- ***Color***.- corresponde al color principal del objeto.
- ***Ambiente*** o ***Ambient***.- es el color de la luz ambiente y que afecta a las zonas en que no incide ninguna luz de forma directa.
- ***Reflexión*** o ***Reflection***.- es el punto brillante en que incide de forma directa una luz, cuyo tamaño depende de la aspereza.
- ***Aspereza*** o ***Roughness***.- es la deformación de algunas partes en la textura del objeto.

Se puede utilizar 8 bits y se tendrán 256 colores o 24 bits obteniendo 16.8 millones de colores.

El color contiene tres barras de desplazamiento con los colores primarios, con valores desde 0 (negro) hasta 1 (blanco), si se selecciona la opción de ***Seleccionar color personalizado*** o ***Select Custom Color*** aparecerá el selector de color mostrado en la figura 3.2 donde se puede escoger un color básico o un color predeterminado de la zona de color y el rango de colores a la derecha con las opciones de los colores primarios y su matiz, saturación y luminosidad, en la parte inferior derecha tiene una opción para agregar el color seleccionado a los personalizados.

Con la opción de ***Seleccionar desde ACI*** o ***Select from ACI*** (figura 3.3) , se puede establecer el color a partir del índice de colores de AutoCAD "ACI" que significa "AutoCAD Color Index" (Índice de colores de AutoCAD).

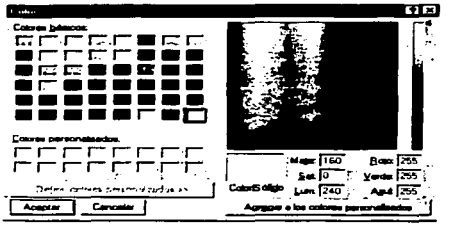


Figura 3.2 Selector de colores

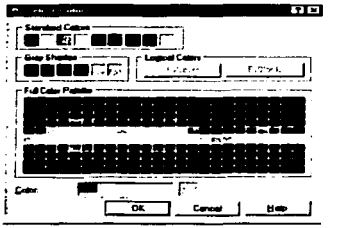


Figura 3.3 Selector ACI

3.1.3 Materiales en AutoCAD

Para asignar un material a varios objetos de un dibujo se deben agrupar en un bloque o separarlos en una capa aparte, o si se desea aplicar un material a una entidad en especial del objeto se deberá primero descomponer el mismo con el comando ***DESCOMP*** o ***EXPLODE***, si se insertan bloques con materiales propios procedentes de otros dibujos también se deben importar los materiales vinculados. AutoCAD presenta tres tipos de materiales especiales o definidos que son el mármol, granito y madera.

Para obtener la representación de un material como el mármol sobre una superficie concreta se debe proyectar una imagen 2D que corresponda con la textura sobre la superficie en cuestión, a este método se le denomina mapeado, la imagen que se proyecta deberá tener un formato de tipo BMP, TGA, TIFF o JPEG, este método utiliza sus propias coordenadas U y V que son independientes de las coordenadas de AutoCAD.

Para trabajar con materiales se puede teclear el comando ***MATERIALR*** o ***RMAT***, o desde la barra de render el icono de la figura 3.4, a continuación se muestra una ventana como la de la figura 3.5, en la opción de materiales aparecen los materiales que se han añadido, si no se han añadido aparece global.

Para crear un nuevo material hay que pulsar la opción de ***Nuevo*** o ***New***, aquí se puede escoger el tipo Estándar, Granito, Mármol y Madera, por ejemplo si se escoge estándar se muestra la ventana de la figura 3.6, en la que se le debe dar un nombre y contiene los atributos que son Color, Ambiente, Reflexión, Aspereza, Transparencia, Refracción y Relieve a los cuales se les puede modificar sus valores y mostrar una vista preeliminar escogiendo la forma en que se quiere representar ya sea cubo o esfera y al darle ***OK*** o ***Aceptar*** se agregará a la lista de materiales de la ventana principal.

Figura 3.4 Icono de materiales

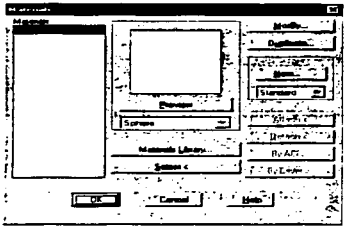


Figura 3.5 Ventana de materiales

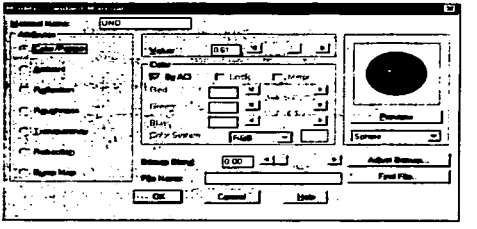


Figura 3.6 Ventana de materiales estándar

La opción ***enlazar*** o ***Attach*** de la ventana principal de materiales enlaza el material con las entidades designadas y ***desenlazar*** o ***Detach*** realiza lo contrario.

Con la opción ***Biblioteca de materiales*** o ***Materials Library*** se accede a los materiales que ya tiene creado AutoCAD y se presenta la ventana de la figura 3.7.

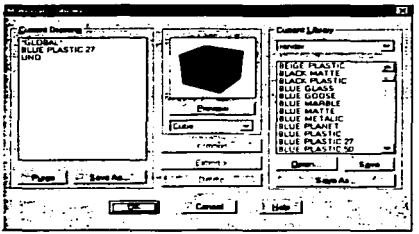


Figura 3.7 Ventana de biblioteca de materiales

También se puede acceder a esta opción desde la barra de render con el icono de la figura 3.8 o con el comando ***BIBLIOMAT*** o ***MATLIB***, en está ventana existen dos bibliotecas de donde puedes escoger los materiales Render y Mini, se pueden visualizar e importar a la lista de materiales.

Figura 3.8 Icono de la Biblioteca de materiales

3.1.4 Tipos de mapeado

Mapa de Textura.- Es el más básico y el que más se utiliza y su función es únicamente proyectar la imagen asociada a la superficie que se designa.

Mapa de reflexión.- Simula una escena reflejada, con éste se consigue efectos de espejo o de reflexión pero el material deberá tener poca aspereza.

Mapa de opacidad.- es para determinar áreas opacas y transparentes, los píxeles de color negro serán transparentes y los de color blanco serán opacos.

Mapa de relieve.- Simula un efecto de relieve al contrastar los píxeles del fondo de la imagen.

El mapa de reflexión es el único que no necesita definir un mapa de coordenadas, todos los demás es necesario definir su mapa y una vez que se aplique a un objeto, éste quedará vinculado.

Existen cuatro tipos de mapeado y estos se escogen dependiendo del tipo de objeto al aplicar.

- Plana.- Asigna la textura al objeto de forma directa simplemente proyectándola.
- Cilíndrica.- Asigna la textura de forma que los bordes horizontales queden cerrados, pero no los bordes inferiores y superiores.
- Esférica.- Gira la textura de forma horizontal y vertical, de manera que el borde superior del mapa se comprima hasta el punto superior de la esfera y el borde inferior hasta el punto inferior de la esfera.
- Sólida.- Los materiales sólidos son tridimensionales y por lo tanto podrán aplicarse desde cualquier ángulo.

El comando de ***MAPEADO*** o ***SETUV*** tiene como función la de especificar la forma y el tamaño de las texturas que deberá presentarse en los objetos seleccionados, para poder utilizar el mapeado previamente se tuvo que enlazar el material con el objeto u objetos y después se debe seleccionar el objeto, también se puede acceder a este comando con el icono de la figura 3.9 y se muestra una ventana como la de la figura 3.10 en la cual se escoge la proyección que puede ser; Plana, Cilíndrica, Esférica o Sólida y con la opción de ***ajustar coordenadas*** o ***Adjust Coordinates*** se especificará el área a ser mapeada, la opción ***Adquirir de*** o ***Acquire from*** es por si se quiere asignar al objeto un mapeado que ya tenga otro objeto y la de ***Copiar a*** o

Copy to si se quiere aplicar los parámetros a otros objetos, todos los cambios se pueden observar con la opción de ***vista preeliminar*** o ***Preview.***

Figura 3.9 Icono de mapeado

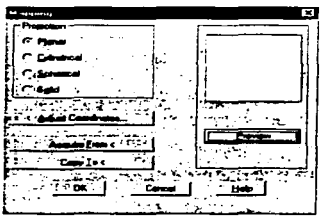


Figura 3.10 Ventana de mapeado

AutoCAD utiliza una gran variedad de utilidades para dar realidad a sus objetos y eso hace que su manejo sea un tanto complejo, pero es algo necesario ya que los objetos que se deben crear muchas veces deben ser lo más realista posibles, otra desventaja que tal vez no se le tome mucha importancia ya que muchas veces depende del tipo de ordenador que se utiliza es que AutoCAD consume muchos recursos tanto del ordenador como de la memoria, se podría pensar que en la actualidad con ordenadores tan rápidos no es un serio inconveniente pero cuando se tienen programas como AutoCAD, 3D-MAX, PHOTOSHOP, u otros corriendo al mismo tiempo, los ordenadores no son tan rápidos ni tan eficientes y sobre todo si se está trabajando bajo una plataforma Windows.

3.2 EDITOR DE TEXTURAS MORAY

Moray es un modelador de marco de alambre que ayuda a POVRAY, este programa se encuentra en la página: http://www.strmuc.com/moray, esta página proporciona el software de tipo shareware que significa que es de prueba, este programa sólo te permite guardar un archivo 30 veces, este software corre en la plataforma PC, se tienen disponibles todas las versiones para poder probarlo sin ningún problema, al instalarlo manda un mensaje indicando que se está utilizando una versión no registrada, algunas de sus versiones tienen modificaciones en sus características principales como la presentación de las primitivas, sus fuentes de luz, las cámaras, etc., en la versión más actual las colocan en una de las barras de herramientas. En la tabla 3.1 se muestran las versiones con los sistemas operativos que corre Moray sin ningún problema.

PLATAFORMA	VERSIÓN DE MORAY
Win3.11, win95,win98,winnt,win2000	Ver 3.3, Ver 3.2, Ver 3.1, Ver 3.01,
Dos	Ver 2.5

Tabla 3.1 Plataformas y versiones de Moray

Al ejecutar el programa muestra 4 vistas de trabajo, estas vistas son muy semejantes a los de otros programas de modelado en 3D, proporcionan unas barras de menús, herramientas, etc., estas utilerías son modificadas dependiendo de la versión del programa, el programa genera un archivo con extensión .POV que es utilizado por el programa POVRAY, este software trabaja bajo el modelo de trazado de rayos, y la extensión .MDL es generada sólo para Moray.

Al descargar este programa de la página proporcionan un tutorial que contiene algunos ejemplos y un manual (figura 3.11).

Figura 3.11 Programa Moray y manual

En esta sección se estudió el editor de materiales que Moray ofrece para agregar diferentes texturas en Povray, en estos materiales se combinan las propiedades de la superficie y del interior de cada uno de los objetos, cada una de éstas describe o representa la refracción de la intensidad de la luz.

Este editor de materiales tiene varias secciones que son:

- Una sección en la que el usuario puede determinar o modificar las propiedades del color, está dividido en tres secciones que tiene 8 barras que se deslizan con 2 botones de incremento y decremento respectivamente, éstas tienen un intervalo con valores de 0 a 1, como colores primarios se

  tiene el rojo, verde y azul que representan el modelo de color RGB, en las siguientes barras está el filtro y transparencia los cuales controlan los ajustes del color, en ésta parte se puede observar que si el color es negro el objeto es opaco y en caso contrario si el color es blanco el objeto es transparente, en la parte de la gama muestra un diálogo que contiene los controles en los cuales se determinan los valores que permitirán los límites de la transparencia de la filtración del modelo de color RGB, estos valores están en un rango de –10 a 10 para ambos casos, el botón de color despliega un diálogo que muestra colores básicos y colores personalizados.

- Mapa de color; se utiliza la misma ventana de color sólo que en ésta hay una modificación, aquí se proporciona un campo de color que está compuesto por subcampos de color que se puede combinar dentro de un objeto, aquí se puede observar la gama del color desde el más oscuro hasta el más claro, las flechas que se encuentran debajo de este campo se utilizan para conectarse con la regla que está debajo del mismo, para poder utilizar estas flechas es necesario presionar el botón del ratón, las flechas que no se pueden mover son las que están en los extremos de la regla del mapa, por último debajo de las flechas encontramos dos campos los cuales señalan los extremos de la regla, el primero representa los valores (0,0) y el último los valores (1,0), estos dos campos representan el mapa de color real, se puede agregar más campos de color utilizando un botón de agregar y se le puede modificar el color del campo, las flechas que generan estos se desplazan de un extremo a otro.

- La siguiente característica de el editor es el ***blockpattern*** este diálogo contiene una etiqueta que tiene tres características que modifica el objeto, una de ellas es el ***checker***, este es un patrón de 3D que contiene 2 entradas las cuales son los pigmentos y las texturas, el siguiente es el ***hexagono*** el cual contiene 3 entradas, la primera es el centro del patrón, las siguientes entradas son los parámetros que definen el hexágono en el plano **X,Y**, por último se tiene el ***brick*** aquí se simula los ladrillos de una pared, su primera entrada representa los ***brick***, su segunda entrada es el ***mortar*** entre ellos, este parámetro es controlado por otros parámetros como el ***bricksize***, y el ***mortar***.

- El ***bricksize*** es el parámetro que define el tamaño del ***brick*** tomando en cuenta los ejes ***x***,***y*** y ***z*** que por defecto representa la longitud, la altura y la profundidad del componente del objeto, ***mortar*** define el espacio entre los ***brick***.

- ***Noise Pattern*** esta ventana da la oportunidad de modificar todo el escenario de trabajo, la sección de ***pattern type*** proporciona una lista de patrones para buscarlos en POVRAY y en los siguientes botones se modifican los efectos de textura.

- ***Noise Modifiers*** para modificar el ruido se emplea la fuerza y el comportamiento de la turbulencia, también en este cuadro de diálogo se encuentran los parámetros de: ***octaves***, ***omega*** y ***lamda*** los cuales son utilizados para modificar los parámetros de color dependiendo de la textura en la cual se está trabajando en ese momento, estos parámetros trabajan utilizando puntos de la superficie del objeto.

- ***Finish Parameters*** en esta sección se modifican los parámetros de ***ambiente***, ***diffuse***, ***brillantez***, ***phong***, ***phong size***, ***Specular***, ***Roughness***, ***refleccion*** entre otros los cuales básicamente se encargan de la luminosidad de la escena en el objeto tomando en cuenta los modelos de iluminación para generar los puntos o fuentes de luz, en la figura 3.12 se muestra el editor de materiales.

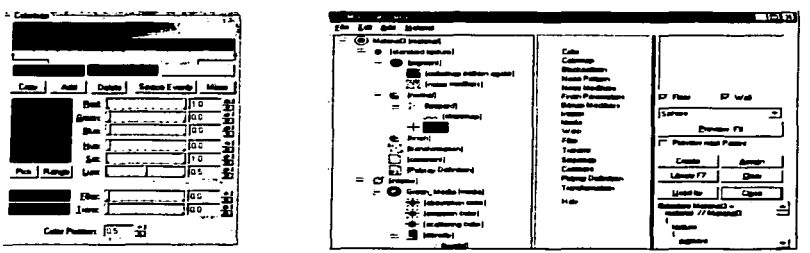


Figura 3.12 editor de materiales

Para crear materiales existen dos formas, una es utilizar un botón de ***Create*** que se encuentra en el editor de materiales, con éste se muestra otra ventana la cual contiene varias opciones para generar los tipos de texturas que son las siguientes:

- Textura Estándar; en esta opción el editor tiene tres acciones las cuales son; pigmento, normal y acabado.

- Mapa de textura; aquí se mezclan diferentes texturas utilizando parámetros que van desde 0 hasta 1, ésta tiene varias entradas las cuales representan la posición dentro del mapa de la textura.

- Mapa de Material; se basa en una imagen que depende del archivo en el que se esté trabajando, esta opción soporta hasta 256 colores con un componente máximo de 8 bits.

- Bloque de patrones de texturas; esta opción dispone de 2 a 3 texturas entre una forma geométrica.

Al material generado se le puede agregar diversas opciones que el editor proporciona, algunos ya se mencionaron anteriormente, estas opciones también se pueden generar utilizando las que están en la barra de herramientas, al crear un material nuevo da la opción de ponerle un nombre al material o se puede dejar el que el editor proporciona por defecto, en la figura 3.13 se muestra el diálogo para crear nuevos materiales.

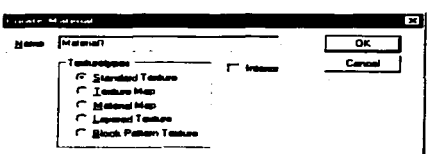
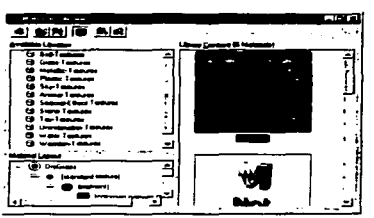


Figura 3.13 Diálogo para crear nuevos materiales

El editor de materiales tiene una opción que despliega una ventana, ésta contiene una variedad de texturas que proporciona Moray a partir de sus bibliotecas, esta opción está compuesta por varias secciones en una de ellas se encuentra un listado de diferentes texturas de madera, metal, plástico, agua etc. Al seleccionar una de estas opciones despliega un subdirectorio que está representado con objetos, estos tienen una variedad del mismo tipo de textura tal como el de madera que tiene una diversidad de maderas, por ejemplo dentro de los metales tiene: aluminio, plata, bronce, etc., en las maderas se tiene: corcho, pino, etc.

Para observar la textura hay que renderizar el objeto de la textura para esto hay un opción en la que se despliega otra venta que pregunta si se quiere renderizar una sola textura o todas de un sólo tipo, el renderizar significa aplicar o actualizar el objeto más reciente, en la otra sección se presenta las partes con la que está conformada la textura seleccionada, estas partes son los pigmentos, color, efectos, etc. Al seleccionar la textura el siguiente paso seria copiarla al objeto utilizando un botón que se encuentra en la barra de herramientas de dicha ventana (figura 3.14).



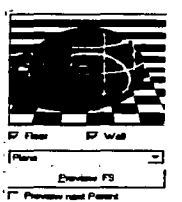


Figura 3.14 Libreria de materiales de Moray

Al utilizar el botón ***used by*** despliega una ventana de diálogo que permite tener una lista de materiales o texturas que se estén utilizando actualmente en la escena de trabajo, permitiendo modificar, actualizar o reemplazar el material de un objeto.
Los últimos botones son de borrado de asignación, con éste se aplica la textura de trabajo al objeto y se cierra la ventana.

La sección que está debajo de estos botones es una sección que muestra la textura con un lenguaje que es compatible con POVRAY aquí se puede exportar el código a este programa para utilizarlo en el modelado de otras escenas.

Moray permite que se observe el material dentro del editor antes de aplicar o asignar este mismo al objeto dentro de la escena, para esto contiene una sección de previsualización aquí se tienen diferentes formas de visualizar el objeto como esfera, cilindro, plano, cubo , etc. Esta opción trabaja junto con el programa Povray que renderiza el objeto aplicando la textura o creando en forma automática la imagen.

Los tipos de pigmentos contienen un subdirectorio de color sólido, que únicamente genera y aplica un color para la textura que se trabaja en ese momento.

Moray es un programa muy sencillo a la hora de modelar objetos, ya que su editor de texturas proporciona una biblioteca de materiales las cuales son sencillas para aplicarlas, su presentación es un poco tosca ya que despliega una ventana en la cual están ordenadas las texturas por tipo, pero una de sus desventajas es que hay que renderizar la textura. Al seleccionar este paso despliega otra ventana y tenemos que esperar a que mande llamar al programa de Povray, en esta parte es un poco lento ya que para visualizar todas se tiene que realizar este paso y seleccionar la opción de rederizar todas, es uno de los programas más completos y sencillos que tiene mucho futuro para el modelado sólo tiene que ser más conocido por la gente que se empieza a diseñar objetos o escenas en 3D.

3.3 EDITOR DE TEXTURAS 3D STUDIO MAX

El software es empleado para modelar objetos 3D y 2D, al empezar a utilizarlo proporciona un vista que muestra el objeto, estos visores se pueden ver en cualquier perspectiva, para poder tener acceso a todas las herramientas del programa, se puede realizar desde el panel de comandos.

El software proporciona un editor de materiales, éste se encuentra en la barra de herramientas, al hacer un clic en ella despliega una ventana que contiene varios carácteristicas las cuales son las siguientes:

Al aplicar un material solamente se tiene que asignar el material utilizando el botón que se encuentra en el editor, estos materiales están clasificados en 10 tipos estándar de los cuales a partir de ellos se puede generar más tipos de materiales, dentro de estos materiales estándar se encuentran:

- ***Blend***; Estos son generados al mezclar diferentes en un sólo lugar del objeto.
- ***Composite***; Combina 10 tipos de materiales, éstos se mezclan utilizando los controles de cada material.
- ***Double Sided***; Aplica 2 tipos de materiales uno en cada lado, tanto en el interior como en el exterior del objeto.
- ***Estándar***; Es la opción en la cual se puede trabajar con todos los materiales que 3D max tiene.
- ***Matte/Shadow***; El efecto hace que los objetos tengan sombras y reflexiones haciendo que las imágenes parezcan muy reales.
- ***Morpher***; El efecto se utiliza para generar animaciones ya que este es un material de mezcla, haciendo que los modelos cambien.
- ***Multi/subobjeto***; Se utiliza cuando es necesario utilizar varios materiales en un objeto, para esto enumera los conjuntos de caras para asignar los diferentes materiales.
- ***Raytrace***; Es utilizado para representar la reflexión y la refracción , este es un poco inconveniente ya que necesita muchos recurso de la computadora por utilizar el trazo de reyos.
- ***Shellac***; Controla la luminosidad del segundo material, haciendo que el objeto tenga mucho más brillo, este material es una modificación del material de ***blend***.
- ***Top/Bottom***; El material para poder trabajar necesita de las normales del objeto, les aplica diferentes tipos de materiales dependiendo de la posición de las caras del objeto.

El editor proporciona un nombre predeterminado el cual se puede modificar por el usuario, estos materiales tienen propiedades muy específicas, el color y el resplandor del material.

El color del material está compuesto por la difusión y el color ambiente, generando estos dos a un color básico, la difusión es generada con la luz natural provocando que un color se observe con mayor facilidad, el color ambiente es provocado por la iluminación tenue el cual se observa de la misma manera como si fuera iluminado por la luz del sol, caso contrario si se ilumina con una luz débil.

3D MAX controla la especularidad mediante el material tomando en cuenta algunas propiedades que son; el color especular, que controla el color de resalte del material, el nivel especular, éste controla la intensidad de la especularidad de un material[17], ésta opción se encuentra en los parámetros básicos del editor de materiales.

Otra de las opciones de editor es la autoiluminación que se encuentra en los parámetros básicos, este efecto provoca la iluminación de la superficie del material haciendo que el color se intensifique, remplazando el color de las sombras.

La opacidad se encuentra en los parámetros básicos, ésta maneja la transparencia del material el cual puede utilizar mapas de bits, hay algunos materiales como la madera, metales que tienen un 100% de opacidad, caso contrario los plásticos y vidrios tienen un intervalo mas bajo de opacidad, ya que éstos son transparentes.

Las sombras se pueden crear utilizando la opción que se encuentra en los parámetros básicos de sombreador, éste es determinado por el tipo de material que se utilizará en el objeto, para esto hay varios tipos de materiales que son determinantes para generar otros.

Para generar un reflejo hay que tomar en cuenta la rapidez, esta opción es muy importante para que el objeto sea lo más realista posible.

[17] 3D Studio Max 3.1 Autor: David J. Kalwick Editorial: Pretince Hall. Pag. 362

La creación de mapas se habilita accionando la opción de mapas del editor de materiales que se encuentra junto a los parámetros de Maps, estos mapas son utilizados para que el material del objeto tenga una apariencia de desgaste.
El editor cuenta con varias herramientas que son necesarias para modificar los materiales, se enlistan a continuación:

• Material y navegador de mapa; Sirve para explorar materiales es muy útil cuando tenemos niveles en los materiales.

• Get material; Esta herramienta abre el visor de materiales, recuperándolos de las escenas.

• Go to Sibling; Sirve para explorar entre niveles tomando en cuenta el nivel de igualdad de los materiales.

• Put Material; sólo funciona cuando se utiliza el mismo nombre del material en una misma escena.

• Go to Parent; Explora entre los niveles de los materiales ascendiendo en el nivel de los materiales.

• Put to library; Sirve para guardar el material con extensión mat.

•Make material copy; La opción genera una copia idéntica del material, para modificar sin alterar originales.

• Reset Map/Material; Elimina los atributos del material, poniendo los valores de inicio.

• Assign Material; Funciona poniendo el material a todos los objetos de la escena cuando todavía no se selecciona uno en especial.

• Select by Material; Esta opción genera una lista de los materiales que se están utilizando en una escena.

• Background; Esta opción genera un fondo de apariencia de un tablero de ajedrez.

En la figura 3.15 se muestra el editor de materiales con todas sus herramientas en la creación o modificación de materiales.

El selector de color se puede abrir cuando queremos cambiar de color, esta aplicación se encuentra al utilizar el color diffuse, al seleccionar el color aparecerá en la muestra de color del cuadro de diálogo, este cuadro contiene botones deslizables los cuales muestran si el color es demasiado claro u oscuro, aquí se puede utilizar uno de los modelos de color RGB, HSV, etc., por lo regular cada uno representa diferente el aspecto de los colores.

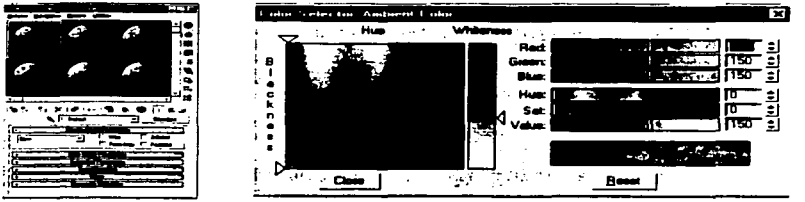


Figura 3.15 Editor de materiales y selector de color

Para poder dominar el programa hay que dedicarle mucho tiempo para entenderle, ya que tiene muchas opciones, respecto a su editor de texturas o de materiales es muy enredado tienes que realizar muchos pasos para seleccionar una textura, en si el programa en general tarda demasiado en abrirse.

Al contrario de otros programas para aplicar la textura al objeto no es necesario utilizar programas adicionales para ver la textura que se utilizara, una de sus ventajas es que tiene autoiluminación.

3.4 EDITOR DE TEXTURAS DE AC3D

AC3D [18] es un programa de distribución libre para LINUX y una versión no comercial para WINDOWS, pero si se quieren cargar todas sus aplicaciones es necesario adquirir la versión comercial, AC3D es un modelador de objetos en 3D y también trae su propio editor de texturas, tal vez no sea tan completo como el editor de AutoCAD o 3D MAX pero cumple con la función principal de un editor de texturas que es la de aproximar los objetos a formas reales, así que a pesar de no ser un programa con grandes características se pueden lograr la creación de objetos bastante reales, un ejemplo de lo que se puede lograr con este programa y su editor de texturas se muestra en la figura 3.16

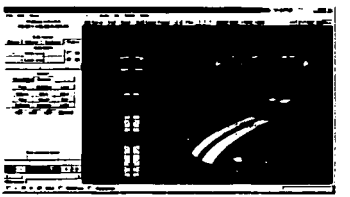


Figura 3.16 Ejemplo de un modelado en AC3D.

3.4.1 Colores en AC3D.

El modelo de color que utiliza AC3D es el RGB, y lo utiliza de cuatro formas diferentes que son:

- ***Difuse Red***, ***Difuse Green*** y ***Difuse Blue.***

[18] http://www.ac3d.org/

- ***Ambient Red***, ***Ambient Green*** y ***Ambient Blue***.
- ***Emissive Red***, ***Emissive Green*** y ***Emissive Blue***.
- ***Specular Red***, ***Specular Green*** y ***Specular Blue***.

Cada opción tiene una barra de desplazamiento que incluye valores de 0.000 a 1.000, con estas barras se puede variar el tipo de color que se va a aplicar en el objeto, combinando estas cuatro opciones de color se pueden hacer texturas bastante creativas y reales, y para dar mayor calidad a las texturas se tiene la opción de ***Shininess*** con la que se da brillo a los objetos, esta opción tiene una barra de desplazamiento con un intervalo de 0 a 128, la última opción que incluye el editor de texturas es la de ***Transparency*** que tiene una barra de desplazamiento con intervalos de 0.000 a 1.000, tomando en cuenta que al llegar a uno la transparencia será total.

En la parte inferior derecha se encuentra una casilla que se puede activar con la opción de ***Realtime update***, esta opción permite aplicar los cambios inmediatamente después de mover alguna de las barras de desplazamiento si no está activada no se hacen los cambios hasta que se cierra la ventana con la opción de ***Close***, en la figura 3.17 se muestra la ventana.

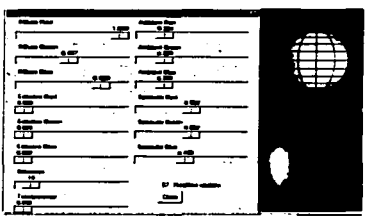


Figura 3.17 Ventana de colores de AC3D

Se puede tener acceso a esta ventana dando clic derecho del ratón en alguno de los colores que da por defecto el programa (figura 3.18) y la opción ***Edit...***

Figura 3.18 Barra de colores de AC3D

Esta barra de colores (***Set surface type***) muestra 14 colores por defecto que con sólo hacer clic sobre ellos se aplica al objeto seleccionado, desde aquí se pueden editar nuevos colores con ***Edit...***y agregarlos a la barra con ***New entry***.

3.4.2 Texturas en AC3D

Para agregar texturas a los objetos se debe seleccionar en el menú principal, ***Object***, ***Texture*** y ***Load Texture*** o ***Ctrl + T*** como se muestra en la figura 3.19, con esta opción aparecerá una ventana (figura 3.20) en la que se puede seleccionar la textura, AC3D no cuenta con una gran cantidad de texturas predeterminadas, en la versión no comercial sólo cuenta con una, pero permite agregar texturas con formato bmp, rgb, rgba y gif, si se selecciona abrir se cargará de forma automática la textura en el objeto seleccionado, es importante seleccionar primero el objeto ya que de lo contrario no estará activada la opción de ***Texture***.

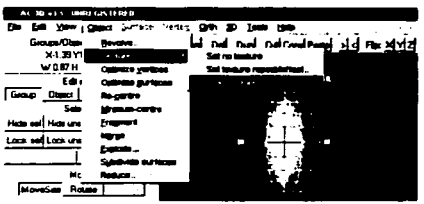


Figura 3.19 Cargar texturas en AC3D

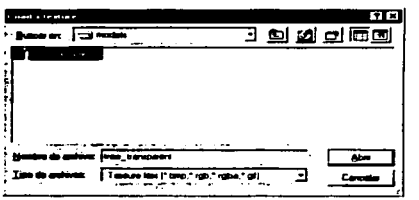


Figura 3.20 Ventana de texturas de AC3D

Con la opción ***Object***, ***Texture*** y ***Set texture repeat/offset*** se muestra una ventana (figura 3.21) con la opción de ***Texture repeat X*** y ***Texture repeat Y*** con cajas de texto y flechas para cambiar sus valores, este valor indicará el número de veces que

se repetirá la textura en el eje ***X*** o en el eje ***Y*** según corresponda, el número que se puede poner es infinito, pero llega el momento en que no se aprecia el cambio, también tiene la opción de ***Texture offset X*** y ***Texture offset Y***, que sirve para poner la posición de la textura en el objeto ya sea en el eje de las ***X*** o el eje de las ***Y***, aquí aparece una barra de desplazamiento y una caja de texto con valores de –1.00 a 1.00, para apreciar los cambios cuando se muevan las barras debe estar activada la opción de ***Realtime update***, también ofrece las opciones de cerrar la ventana (***Close***) o aplicar los cambios (***Apply***), ha esta textura una vez aplicada en el objeto se le puede cambiar el color simplemente seleccionándolo de la barra de colores personalizados o creando uno nuevo.

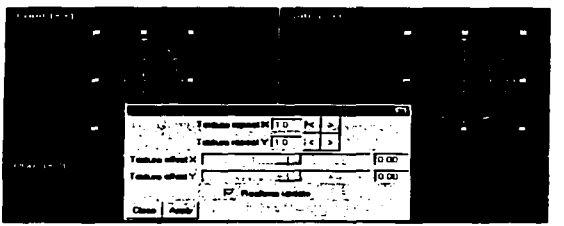


Figura 3.21 Opciones de la texturas de AC3D

Si se quiere quitar la textura simplemente se debe seleccionar la opción ***Object***, ***Texture*** y ***Set no texture***.

La principal característica de AC3D es que su utilización no es muy compleja, las opciones son bastantes básicas y no ocupa muchos recursos del sistema ni en plataforma Windows ni en Linux, pero le hacen falta algunas opciones, principalmente la de poder cancelar la aplicación de una textura porque una vez realizados los cambios no se puede deshacer estos.

3.5 VENTAJAS Y DESVENTAJAS DE LOS EDITORES DE TEXTURA ANALIZADOS

Es importante mencionar algunas ventajas y desventajas de los editores de textura que se han analizado hasta el momento, todos tienen características semejantes como el color, poder insertar imágenes como una textura o alguna textura propia, pero también presentan sus propias características, éstas muchas veces dependen para que programa está hecho el editor de materiales, como AutoCAD que es un programa en el que sus simulaciones necesitan efectos de luces muy reales por ejemplo en la simulación de casas las luces que se aplican en las texturas son muy importantes para resaltar focos o luz ambiental en una ventana, por eso es que AutoCAD cuenta con muchas opciones de luces y sombras que se aplican a los diferentes tipos de materiales, en la tabla 3.2 se simplifican algunas ventajas y desventajas de los editores de textura estudiados.

Editor de texturas	Ventajas	Desventajas
AutoCAD	Permite crear modelos con muy buena calidad.	Si se quiere mejor calidad se ocupan más recursos de la máquina
	Se tienen tres formas de aplicar sus opciones (comandos, iconos y barra de menús).	Existen muchos comandos y no son fáciles de recordar
	La cantidad de texturas que tiene es muy grande	sólo tiene tres formas de representar las textura (madera, granito y mármol)
	Se pueden observar los cambios antes de ser aplicados al objeto	No es fácil manejar todas las opciones del render, ya que son bastantes, y cada una con parámetros diferentes.
3D MAX	Asigna la textura sin renderizar	Tarda en abrir el programa
	Sólo tiene 10 conjuntos de diferentes tipos de texturas estandar. Tiene autoiluminación	Es un poco confuso para asignar una textura
MORAY	Abre rápido el programa	Necesita del Programa de Povray para renderizar la textura.
	Asigna la textura o el material rápido	Se necesita renderizar todas las texturas y materiales que se utilicen, de la librería de materiales

Editor de texturas	Ventajas	Desventajas
	Es sencillo al trabajar	Se tienen que realizar varias ejecuciones antes de que se aplique la textura
	Te genera el código para exportarlo a otro programa	
AC3D	Ocupa pocos recursos del sistema	Sólo tiene una textura predeterminada
	Es fácil de manejar	Si escoges una textura o color, no tienes opción de cancelarla
	Puedes cambiar colores de forma muy sencilla	La gama de colores no es muy visible.

Tabla 3.2 Ventajas y Desventajas de los editores de textura

3.6 DISEÑO DEL EDITOR DE TEXTURAS PARA "MATERIAL 3D"

El editor de texturas para "Material 3D" se diseñó pensando en desarrollar una aplicación de software libre que ayude en la creación de materiales para este programa, sin embargo puede ser utilizado de forma individual o por cualquier aplicación que sea compatible, el código fuente queda liberado con la intención de que sirva a otras personas y pueda ser modificado para otras aplicaciones.

Es una aplicación que funciona bajo una plataforma LINUX, se desarrolló en un sistema operativo Red Hat 8.0 [19] y aprovechando las nuevas herramientas de programación que ofrece LINUX para el desarrollo de interfaces gráficas principalmente GTK+ 2.0 la aplicación es creada con la herramienta de programación GLADE 2 versión 1.1.6 que está incluida en el sistema operativo.

3.6.1 Ingenieria de software

Es importante hacer una mención de lo que es la ingeniería de software y como se aplica, "la ingeniería de software es una actividad de modelado, una actividad para la solución de problemas, se usan los modelos para buscar una solución aceptable, es una actividad dirigida por la fundamentación "[20]

[19] http://www.redhat.com
[20] Ingeniería de Software Orientada a Objetos, Bernd Bruegge pg 6

La ingeniería de software surge con la necesidad de crear software de calidad, existe mucho software en el ámbito computacional ya sea comercial o libre sin embargo la mayoría del software que existe no cumple con los criterios de calidad que se requieren, es muy poco el software que se desarrolla como debe ser, la mayoría no tiene una buena planeación o no se piensa en el usuario final, el desarrollador de software se enfrenta a varios problemas, uno de ellos es entender las necesidades de los usuarios, la mayoría de las veces los programas lo solicitan personas ajenas al manejo del software, asi que es importante entender directamente las necesidades de los usuarios, otro problema con el que se enfrentan es la rapidez con que avanza la tecnología, nuevo hardware, sistemas operativos, etc. Como experiencia personal se puede citar que al desarrollar el editor de materiales se liberó una nueva versión de la herramienta de programación con la que construimos la interfaz del editor, evaluando la nueva versión y observando que los cambios que se habían realizado sobre ésta eran bastante significativos se optó por rediseñar el editor en la nueva versión del programa, este tipo de cosas no se pueden prevenir pero es importante tener presente que los cambios tecnológicos pueden afectar el desarrollo y diseño de un proyecto.

La forma más sencilla de desarrollar software es mediante ensayo y error tomando en cuenta el tiempo para el desarrollo y el costo del mismo, pero el método de ingeniería más simple incluye cinco pasos que son:

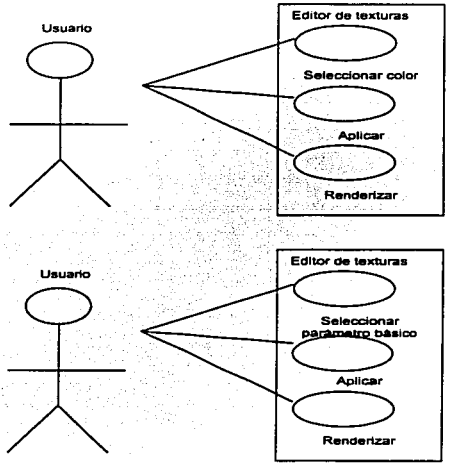
1. Formular el problema
2. Analizar el problema
3. Buscar soluciones
4. Decidir cual es la solución adecuada
5. Especificar la solución

Con esto se puede definir que el desarrollo de software "es el conjunto de actividades necesarias para identificar y describir un sistema como un conjunto de modelos que abordan el problema del usuario final"[21].

[21] Ingeniería de Software Orientada a Objetos, Bernd Bruegge pg. 7

El modelo que utilizamos para el desarrollo del editor es el modelado con UML (Lenguaje de modelado unificado), esta técnica surge de la unificación de la técnica de modelado de objetos e ingeniería de software orientada a objetos, UML se puede adaptar a una gran variedad de aplicaciones, pero es necesario diferenciar los tres modelos que se pueden aplicar con UML.

- Modelo funcional: que describe la funcionalidad del sistema desde el punto de vista del usuario, este modelo se representa en UML con diagramas de caso de uso, estos se usan durante la obtención de requerimientos y el análisis de un sistema y su principal objetivo es representar el comportamiento del sistema desde el punto de vista de los usuarios o factores externos al sistema, ejemplo figura 3.22.



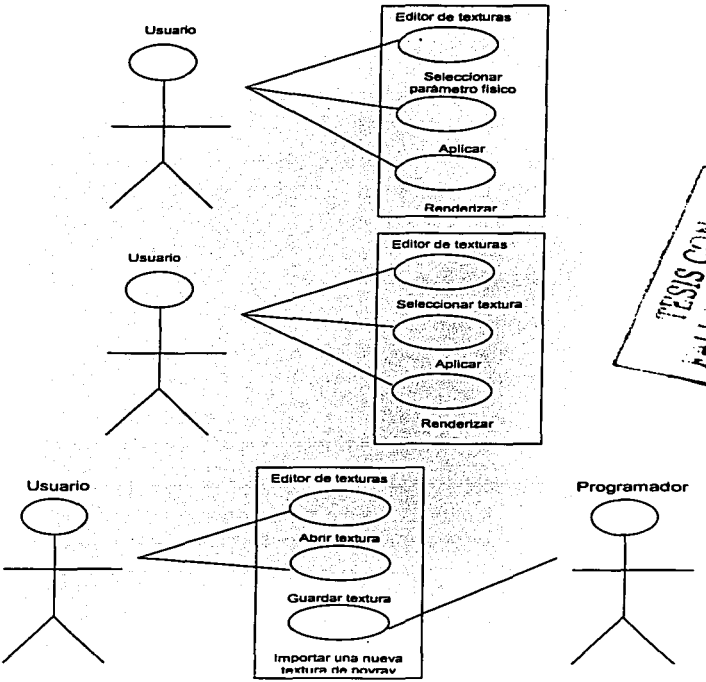


Figura 3.22 Diagramas de caso de uso

- Modelo de objetos: como su nombre lo indica describe un sistema desde el punto de vista de objetos, identificando sus partes como objetos y en UML se representa con diagramas de clase, estos representan la estructura del sistema, se describen los objetos, sus atributos, propiedades y aplicaciones así como su relación con otros objetos, ejemplo figura 3.23.

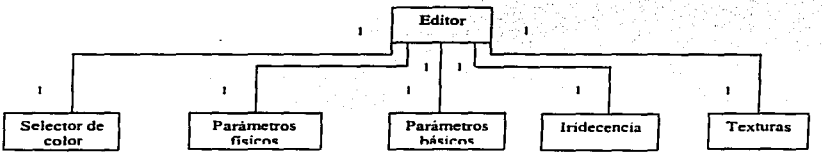


Figura 3.23 Diagrama de clase

- Modelo dinámico: describe el comportamiento interno del sistema y se representa con diagramas de secuencia que se usan para ver el comportamiento del sistema y la comunicación entre procesos, también incluye diagramas de gráfica de estado, que describen el comportamiento de un objeto de forma individual y diagramas de actividad, que describen al sistema de acuerdo a sus actividades, ejemplo figura 3.24 y 3.25.

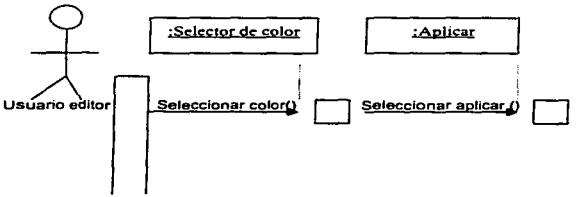


Figura 3.24 Secuencia UML

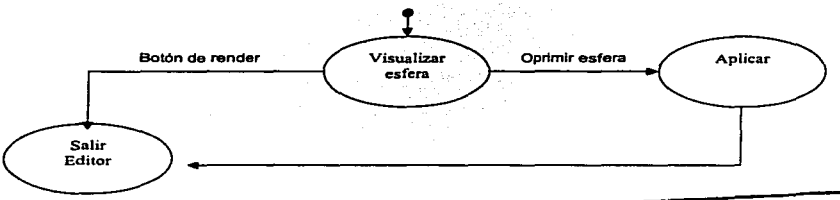


Figura 3.25 gráfica UML de estado

3.6.2 Selección de colores

Después de analizar distintos editores de materiales se han seleccionado las características más importantes para la construcción de un editor que cumpla con su función, una de las partes importantes en la creación de materiales son los colores, todos los editores que se han mencionado tienen su propia forma de seleccionar colores que serán agregados a las texturas o a los objetos, en el editor de texturas para Material 3D se ha escogido un selector de color que viene incluido como un widget básico en GLADE, este selector utiliza el modelo de color RGB (Rojo, Verde y Azul), estos colores se encuentran distribuidos en un anillo, así como la gama de colores que se pueden formar con estos tres colores, en el centro del anillo hay un triángulo en el que se puede variar la oscuridad o luminosidad de algún color seleccionado, se pueden seleccionar los colores con el ratón directamente del anillo y triángulo que se visualizan o de forma manual con las siguientes opciones:

- Hue: Es la posición del anillo de colores, tiene un intervalo de 0 a 360, girando alrededor del circulo para escoger cualquiera de los colores que se presentan.

- Saturación: Se selecciona la profundidad del color, esta varia de 0 a 255, en 0 el valor mas alejado del color y en 255 el más cercano, se puede observar como varia esta opción en el triángulo interno.

- Valor: Permite controlar el brillo del color, su variación también se puede observar en el triángulo interno con un intervalo de 0 a 255, pudiéndose observar en 0 mayor oscuridad y en 255 el mayor brillo.

- Rojo, Verde y Azul : También se pueden seleccionar los colores principales de forma manual, cada uno tiene un intervalo de 0 a 255, observándose en 0 la ausencia del color seleccionado y en 255 el color exacto.

La combinación de estas opciones ya sea de forma manual o gráfica en el anillo y su triángulo interno, dan una gran variedad de combinaciones de colores, si se aprende a hacer un buen uso de este selector se puede obtener gran provecho de su variedad de colores.

Además de las herramientas de color, presenta una opción en la que se puede introducir el nombre del color en formato hexadecimal de forma muy parecida a como se representan los colores en HTML[22], el valor debe ir antecedido por un #, los dos primeros caracteres representan el color rojo, los dos siguientes el verde y los últimos el azul, cada uno con un intervalo de 0 a 255 en decimal y de 0 a FF en hexadecimal, teniendo un intervalo por los tres colores de 000000 a FFFFFF, por ejemplo si se introduce #FFFFFF que es la combinación de todos los colores se representa el color blanco, si se quisiera el color negro que es la ausencia de todos los colores se tendría que introducir el valor #000000, para representar el color azul en su totalidad la opción es #0000FF, este valor puede varia de forma automática cuando se selecciona un color del anillo y del triángulo de colores así como de los parámetros que se presentan.

El selector incluye una casilla en donde se visualiza el color seleccionado. Se ha incluido una opción que puede ser de gran utilidad, se llama ***eyedropper*** y se utiliza para igualar algún color que este en la pantalla u otra aplicación, así si se quiere un color pero no se sabe exactamente que color es, sólo se debe seleccionar el icono

[22] Metalenguaje de Hipertexto

de ***eyedropper*** y se hace un clic en el objeto que tiene el color que se desea y el color aparecerá de forma automática en el selector de colores.

También tiene una opción para variar la opacidad del color con un intervalo de 0 a 255 y una paleta de colores con algunos colores predeterminados, a esta paleta se le puede agregar colores personalizados solo arrastrándolos con el ratón.

Es un selector sencillo de utilizar con las opciones básicas que brindan la mayoría de los selectores de color y cumple con su función que es la de poder añadir color a las texturas u objetos.

3.6.3 Selección de Texturas

El editor cuenta con una opción que aparece por default cuando se abre la aplicación en la que se especifica que se va a trabajar sin una textura, si está seleccionada esta opción se pueden trabajar los colores con sus diferentes opciones, además el editor cuenta con opciones de parámetros básicos en donde se pueden escoger las opciones de: ***Ambiente, Rediosidad, Reflexión, Especular, Refracción, Rougnness,*** con un intervalo de cero a uno, otros de los parámetros que se pueden modificar son los de ***Iridecencia*** en el que se pueden modificar la ***Anchura***, ***Cantidad*** y ***turbulencia*** también con un intervalo de cero a uno, el editor tiene la opción de escoger texturas que se han insertado que representan ***Metales***, ***Plásticos***, ***Maderas***, ***Piedras***, ***Cielo***, ***Océano*** y ***Animales***.

3.6.4 Funcionamiento del editor de texturas

Como ya se mencionó el editor puede funcionar con el programa Material 3D o de forma independiente, si se quiere que funcione de forma independiente se tiene que instalar en el directorio principal tecleando en el promt **$make install** y se puede ejecutar desde el directorio src con **$./editor**

Su funcionamiento es muy sencillo, se ha observado que en editores como el de AutoCAD y 3D-MAX su aplicación es algo confusa, sobre todo porque tienen muchos elementos, así que en este editor se utilizan parámetros básicos muy fáciles de aplicar, en la figura 3.23 se muestra la imagen del editor de texturas.

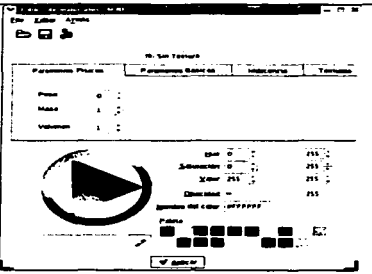


Figura 3.23 Editor de Materiales

En la presentación principal se puede escoger los parámetros físicos de peso, masa y volumen que ayudarán para hacer cálculos de los objetos, el intervalo es variado dependiendo de la opción, se puede seleccionar cualquiera de las pestañas de parámetros básicos, iridiscencia y texturas, se pueden seleccionar los colores desde la ventana principal, el editor tiene opciones para abrir archivos los cuales deben tener extensión .pov, la textura se renderiza automáticamente en una esfera y se debe hacer clic en la esfera para cerrar ésta y regresar a la aplicación, también se pueden guardar las texturas creadas, éstas también deben tener la extensión .pov., todos los cambios que se hagan cuando se seleccionan los colores o algunos de sus parámetros se deben aplicar con el botón que se encuentra en la parte inferior y para visualizar se debe renderizar con el icono de la parte superior que dice renderizar.

CAPÍTULO IV
IMPLEMENTACIÓN Y EVALUACIÓN

4.1. IMPLEMENTACIÓN

El Editor está dividido en seis widgets principales cada uno con sus propios elementos. Se utiliza el Widget ventana que se encuentra en la paleta de widgets básicos (Gtk+ basic), figura 4.1.

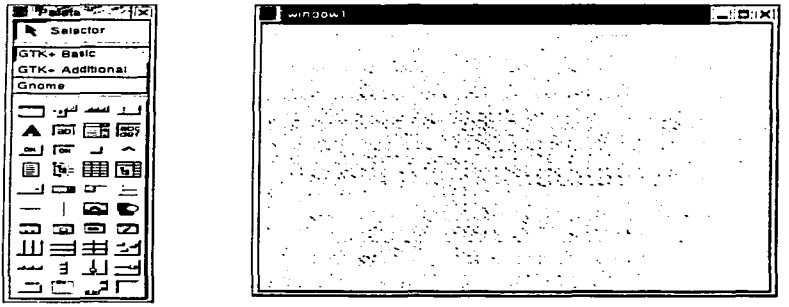


Figura 4.1 Paleta de (Widgets Gtk+ Basic) Widget ventana

La ventana principal se dividió utilizando el widget de caja vertical, en la primera caja seleccionamos e insertamos la barra de menú, en ésta se editaron las opciones de ***Archivo*** que contiene las siguientes opciones (***nuevo, abrir, guardar, guardar como*** y ***salir*** del programa), ***Edición*** contiene lo siguiente (***pegar, copiar, cortar***) y por último ***Ayuda*** que contiene el ***acerca_de*** esta opción despliega una ventana la cual da información del Editor.

A cada una de estas herramientas se les asoció una señal ***clicked*** cada vez que se presione alguna de las opciones mostrará a otros widgets que conforman al editor, el código para las opciones de la barra de menús está en la siguiente tabla 4.1

Opción Nuevo	Opción Abrir

GtkWidget *archivoopen;
archivoopen=create_fileselection1();
gtk_widget_show(archivoopen);
	GtkWidget *archivoopen;
	archivoopen=create_fileselection1();
	gtk_widget_show(archivoopen);

Opción Guardar	**Opción Guardar como**

GtkWidget *archivosave;
archivosave=create_fileselection2();
gtk_widget_show(archivosave);
	GtkWidget *archivosave;
	archivosave=create_fileselection2();
	gtk_widget_show(archivosave);

Opción Salir	**Cerrar ventana principal**

GtkWidget *dialogo;
if(salvado)
gtk_main_quit();
else
{
dialogo=create_dialog1();
gtk_widget_show(dialogo);
}
	GtkWidget *dialogo;
	if(salvado)
	gtk_main_quit();
	else
	{
	dialogo=create_dialog1();
	gtk_widget_show(dialogo);
	}

Tabla 4.1 Código de la barra de menú Opción Archivo

Para generar el código para esta aplicación se utilizó como referencia las API de la página de GTK www.gtk.org, gtk+ son librerías para generar interfaces gráficas se utilizó el GtkWidget, éste es la base para crear otros widgets y está conformado por los siguientes:

```
void    gtk_widget_unparent      (GtkWidget *widget);
void    gtk_widget_show          (GtkWidget *widget);
void    gtk_widget_show_now      (GtkWidget *widget);
void    gtk_widget_hide          (GtkWidget *widget);
void    gtk_widget_show_all      (GtkWidget *widget);
void    gtk_widget_hide_all      (GtkWidget *widget);
void    gtk_widget_map           (GtkWidget *widget);
void    gtk_widget_unmap         (GtkWidget *widget);
```

```
void    gtk_widget_realize      (GtkWidget *widget);
void    gtk_widget_unrealize    (GtkWidget *widget);
void    gtk_widget_queue_draw   (GtkWidget *widget);
void    gtk_widget_queue_resize (GtkWidget *widget);
```

Se tiene que declarar un apuntador de tipo GtkWidget y hacer el apuntador igual al selector de archivos con la función create_fileselection1(), es importante hacer visible el selector de archivos con la función gtk_main_show() pasando como argumento el apuntador. Tabla 4.2.

GtkWidget *archivoopen;	Widget es cualquier elemento de la interfaz, el cual apuntará a una variable que en este caso será archivoopen
archivoopen=create_fileselection1();	La variable archivoopen será el valor a crear el widget fileselection1 sin devolver ningún valor
gtk_widget_show(archivoopen);	Gtk es la biblioteca donde esta el objeto y la funcion, widget es el objeto sobre el que se aplica la accion, show es la acción la cual mostrara otro widget el cual es el archivoopen

Tabla 4.2 funcionamiento del codigo

La opción de ayuda está asociada con una señal de ***clicked*** que contiene el código para mostrar la ventana de diálogo Acerca _ dé que contiene información de la aplicación y de los autores, se debe crear el apuntador y luego la función create_gialog2() y luego mostrarla.

```
GtkWidget *dialogo;
dialogo=create_dialog2();
gtk_widget_show(dialog2);
```


En la otra caja horizontal de la ventana principal se insertó una barra de herramientas las cuales contienen los botones de ***Abrir, Guardar***, y ***Renderizar***, con los iconos en cada uno de los cuadritos que se generaron para esto se utilizó el widget de botón figura 4.2

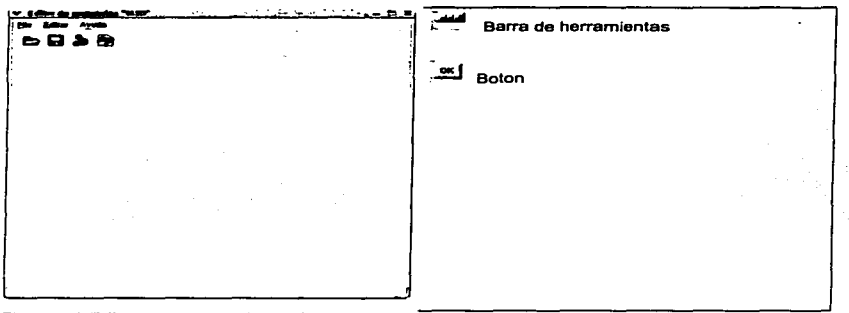


Figura 4.2 Editor con menús de archivos y Barra de herramientas con botones

A cada uno de los botones se les asoció la señal de ***clicked*** para que cuando se arrastre el puntero sobre él muestre el diálogo, la programación de estos botones es idéntica a las opciones de la barra de menú, se debe colocar el código correspondiente en cada señal de ***clicked***.

Dentro de la opción de Ayuda está "***Acerca_de***". Para generar esta ventana se utilizó el widget de la caja de diálogo, esta caja se usa para construir un Acerca_de, cuando se selecciona despliega un diálogo del cual se muestra que botón tendrá la ventana, ésta se divide en dos secciones en una se coloca una imagen y dos etiquetas en las que se coloca el nombre de la aplicación, los autores y correos electrónicos, en la otra sección se coloca un botón de OK el cual tiene asociada una señal de ***clicked***, la función de este botón es la de cerrar la ventana de Acerca_de, para programar este botón también se debe declarar un apuntador de tipo GtkWidget e igualarlo con la función lookup_widget() pasando como argumentos el botón y la caja de diálogo que se va a cerrar.

```
void    gtk_widget_unref        (GtkWidget *widget);
void    gtk_widget_destroy      (GtkWidget *widget);
void    gtk_widget_destroyed    (GtkWidget *widget,
```

```
                                GtkWidget **widget_pointer);
void    gtk_widget_set          (GtkWidget *widget,
                                const gchar *first_property_name);
void    gtk_widget_unparent     (GtkWidget *widget);
void    gtk_widget_show         (GtkWidget *widget);
void    gtk_widget_show_now     (GtkWidget *widget);
void    gtk_widget_hide         (GtkWidget *widget);
void    gtk_widget_show_all     (GtkWidget *widget);
void    gtk_widget_hide_all     (GtkWidget *widget);
void    gtk_widget_map          (GtkWidget *widget);
void    gtk_widget_unmap        (GtkWidget *widget);
void    gtk_widget_realize      (GtkWidget *widget);
void    gtk_widget_unrealize    (GtkWidget *widget);
```

Para este widget se utilizó la libreria de gtk_widget_destroy();, ya que requeriamos que cuando se despliegue la información de ***acerca_de*** se destruya sólo esa ventana (figura 4.3).

```
GtkWidget *cancelar;
cancelar=lookup_widget(button,
"dialog2");
gtk_widget_destroy(cancelar);
```

Figura 4.3 Widget Acerca_de

Para la opción ***desea salvar?*** se utilizó el widget de caja de diálogo que se encuentra en la paleta de widget Gtk+ basic, éste widget tiene tres botones (***Salir, Cancelar, Ok***) y en el área que deja para modificarla se utilizó una caja vertical con dos divisiones en una de ellas se insertó una imagen y en la otra una etiqueta, este funcionará cada vez que se cierre el editor sin salvar los cambios, cada uno de estos botones se asoció con una señal de ***clicked***, el botón de cancelar que tiene la función de salir sin salvar los cambios debe llamar a la libreria gtk_main_quit() esto se debe hacer en el archivo callbacks.c, los botones de Aplicar y OK que tienen como función salvar los cambios deben llamar al selector de archivos para indicar el

nombre con el cual se quiere guardar el archivo, la forma de programar estos dos botones una vez agregadas sus señales es declarar un apuntador de tipo GtkWidget y hacer el apuntador igual al selector de archivos con la función create_fileselection2(), es importante hacer visible el selector de archivos con la función gtk_main_show() pasando como argumento el apuntador. Figura 4.4

Opción Salir

```
gtk_main_quit();
```

Opción Cancelar

```
GtkWidget *salvar;
 salvar=create_window1();
 gtk_widget_show(salvar);
 gtk_widget_hide (gtk_widget_get_toplevel
(GTK_WIDGET (button)));
```

Opción Ok

```
GtkWidget *salvar;
 salvar=create_fileselection2();
 gtk_widget_show(salvar);
```

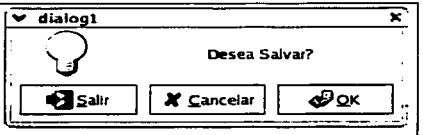
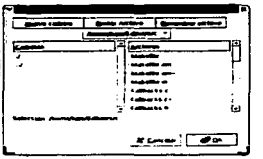


Figura 4.4 Widget Desea salvar?

Otros de los Widgets principales son 2 selectores de archivos que se utilizan para salvar y abrir archivos cada uno tiene 2 botones, uno de OK y otro de Cancelar, el botón de OK tiene distintas funciones ya sea para guardar los cambios hechos en la textura o para abrir una nueva textura dependiendo del selector de archivos, el botón de Cancelar cerrara este widget.

Glade ofrece este selector de archivos en su paleta de widgets de Gtk, con este tipo de opciones se ahorra mucho trabajo de programación ya que el widget ya viene programado con las opciones para crear nuevos directorios, renombrar archivos y desplazarse por los diferentes directorios de LINUX. Figura 4.5

Opción OK



Opción Cancelar

```
gtk_widget_hide (gtk_widget_get_toplevel
(GTK_WIDGET (button)));
```

Figura 4.5 Widget Abrir y salvar archivo.

Para nuestro selector de color utilizamos el widget que se encuentra en la opción que está en la paleta de Gtk+ adicional, para esto en nuestra ventana principal del Editor y en la caja vertical que creamos al principio en esta parte utiliza una posición estática, este widget se selecciona en la paleta de Gtk+ basic, después de esto insertamos el selector de color. Para que funcione esta parte insertamos también un botón de OK, funciones para el widget son las siguientes. Figura 4.6

```
gboolean    gtk_color_selection_palette_from_string  (const gchar *str, GdkColor **colors,
                                                      gint *n_colors);
gchar*      gtk_color_selection_palette_to_string    (const GdkColor *colors, gint n_colors);
GtkColorSelectionChangePaletteWithScreenFunc
gtk_color_selection_set_change_palette_with_screen_hook
                                (GtkColorSelectionChangePaletteWithScreenFunc func);
void        (*GtkColorSelectionChangePaletteWithScreenFunc) (GdkScreen *screen,
                                                             const GdkColor *colors,
                                                             gint n_colors);
void        gtk_color_selection_set_color  (GtkColorSelection *colorsel, gdouble *color);
                    void    gtk_color_selection_get_color  (GtkColorSelection *colorsel);
```

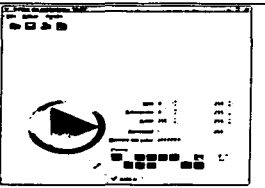


Figura 4.6 Selector de color

Para el selector de color se utilizó la función de gtk_color_selection_get_color y su código es el siguiente tabla 4.3.

```
1 GtkColorSelection *colorsel;
2 gdouble color[4];
3 gdouble r,g,b,o;
4 Colorsel=lookup_widget(GTK_WIDGET(button),"ColorSelection1");
5 gtk_color_selection_get_color(GTK_COLOR_SELECTION(selector),color);
6  r=color; g=color[1];  b=color[2];  o=color[3];
```

1 Gtk es la biblioteca donde está el objeto y la funcion, ColorSelection es el elemento de la interfaz, el cual apuntara a una variable que en este caso será colorsel.

2 y 3 Las variables son de tipo gdouble.

4 El valor de lavariable es buscar el widget al presionar el botón hasta encontrar el widget colorselection1.

5 Tomara el valor de color y regresara un valor al color.

Tabla 4.3 funcionamiento del codigo

En la última caja vertical se colocó una posición estática y posteriormente un widget de libretas el cual contiene ***los paramatros básicos***, ***físicos***, ***texturas*** etc, en cada uno de estos se colocó una posición estática en el cual se utilizó los widgets de hscale horizontal estos se encuentran en la paleta de Gtk+ adicional y se utilizó etiquetas para indicar el nombre de la barra, en el caso de las texturas se insertó un conjunto de libretas las cuales están dividiendo los tipos de texturas que contiene el

editor en éste caso se utilizó el radiobutton para seleccionar cada una de las texturas Figura 4.7

En el caso de los widgets Hscale se utilizó las funciones de las API de tipo GtkRange ya que entre las que contiene el widget utilizado no se encontraba la que era más recomendable para el funcionamiento del mismo. De acuerdo al árbol de herencia del widget, la sinopsis de este widget son los siguientes. Tabla 4.4

```
Gobject
  +----GtkObject
      +----GtkWidget
          +----GtkRange
              +----GtkScale
                  +----GtkHScale
```

Árbol de herencia de Widget

Funciones del widget

```
struct    GtkRange;
GtkAdjustment* gtk_range_get_adjustment        (GtkRange *range);
void      gtk_range_set_update_policy          (GtkRange *range, GtkUpdateType policy);
void      gtk_range_set_adjustment             (GtkRange *range, GtkAdjustment *adjustment);
gboolean  gtk_range_get_inverted               (GtkRange *range);
void      gtk_range_set_inverted               (GtkRange *range, gboolean setting);
GtkUpdateType gtk_range_get_update_policy      (GtkRange *range);
gdouble   gtk_range_get_value                  (GtkRange *range);
void      gtk_range_set_increments             (GtkRange *range, gdouble step, gdouble page);
void      gtk_range_set_range                  (GtkRange *range, gdouble min, gdouble max);
void      gtk_range_set_value                  (GtkRange *range, gdouble value);
```

Tabla 4.4 Funciones del Widget

Para que este widget aplique el valor de los parámetros que se estarán cambiando de acuerdo a la selección de los colores tales como ***difusión***, ***brillantez***, ***ambiente***, ***refracción***, ***especularidad***, ***turbulencia***, etc del mismo, se utilizó la función gtk_range_get_value, ésta es la más conveniente ya que se necesitaba tomar el

valor para modificar los parámetros de color de cualquier aplicación, el código de este widget es el siguiente:

```
GtkWidget *range;
gdouble ambiente, refle, radio, spec, refra, roug, anchura, canti, turbu
range=lookup_widget(GTK_WIDGET(button),"hscale8");
ambiente=gtk_range_get_value(GTK_RANGE(range));
range=lookup_widget(GTK_WIDGET(button),"hscale9");
radio=gtk_range_get_value(GTK_RANGE(range));
range=lookup_widget(GTK_WIDGET(button),"hscale10");
refle=gtk_range_get_value(GTK_RANGE(range));
range=lookup_widget(GTK_WIDGET(button),"hscale14");
spec=gtk_range_get_value(GTK_RANGE(range));
range=lookup_widget(GTK_WIDGET(button),"hscale15");
```

El widget utilizado apuntará a una variable range, las variables son de tipo gdouble, el apuntador será igual a buscar el widget desde Gtk_widget tomando como referencia al botón con el nombre de hscale8 este nombre estará cambiando de acuerdo al widget seleccionado, las variables serán igual a la biblioteca donde esta el objeto y la función el cual aplicará la acción al objeto utilizado, el cual tomará el valor del widget al que apunta el mismo.

En el caso del widget radiobutton el cual se seleccionará el tipo de textura que se aplicará en el objeto tiene el siguiente árbol de herencia Tabla 4.6

GObject +—GtkObject +—GtkWidget +—GtkContainer +—GtkBin +—GtkButton +—GtkToggleButton +—GtkCheckButton +—GtkRadioButton	En este caso la funcion que se utilizó para el funcionamiento del radio button se encontraba dentro del árbol de herencia en el GtkToggleButton

Tabla 4.6 Árbol de herencia del widget

Las funciones que contiene este widget (GtkToggleButton) en las cuales funciona de acuerdo a lo que se requiere en el caso correspondiente, para este caso se utilizó el de gboolean gtk_toggle_button_get_active (GtkToggleButton *toggle_button);.

```
struct     GtkToggleButton;
GtkWidget*  gtk_toggle_button_new                    (void);
GtkWidget*  gtk_toggle_button_new_with_label         (const gchar *label);
GtkWidget*  gtk_toggle_button_new_with_mnemonic      (const gchar *label);
gboolean    gtk_toggle_button_get_mode               (GtkToggleButton *toggle_button);
#define     gtk_toggle_button_set_state
void        gtk_toggle_button_toggled                (GtkToggleButton *toggle_button);
gboolean    gtk_toggle_button_get_active             (GtkToggleButton *toggle_button);
void        gtk_toggle_button_set_active             (GtkToggleButton *toggle_button,
                                                      gboolean is_active);
gboolean    gtk_toggle_button_get_inconsistent       (GtkToggleButton *toggle_button);
void        gtk_toggle_button_set_inconsistent       (GtkToggleButton *toggle_button,
                                                      gboolean setting);
```

Esta función se utlizó ya que se necesita tomar el valor al activarlo, para aplicar la textura correspondiente el cual el widget tendrá un apuntador con el nombre toogle_button con variables de tipo bolean y estas serán desde la textura 1 hasta la 43, el apuntador será igual a buscar el widget desde un GtkWidget tomando como referencia a un botton con el nombre de toggle_button y su código es el siguiente:

```
GtkWidget  *toogle_button;
gboolean tex1, tex2, tex3, tex4.....tex43;
toogle_button=lookup_windet(GTK_WIDGET(button),"radiobutton1");
tex1=gtk_toggle_button_get_active(GTK_CHECK_BUTTON(toggle_button));
```

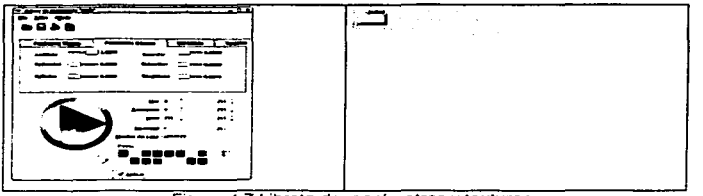


Figura 4.7 Libreta de parámetros y texturas

Se debe crear un archivo en el directorio src que se llamó globales.h en el que se utilizó para crear una variable (salvado) de tipo entero, es importante mencionar que se debe incluir este archivo tanto en main.c como en callbacks.c además de que en main.c se debe inicializar la variable a cero para todas las opciones.

Este mismo código se utilizó para programar la ventana principal, pero primero hay que agregar una señal de destroy. Es importante realizar esto porque de lo contrario la aplicación jamás terminaría, ya que al ejecutarse entra en el bucle principal esperando señales y si no se le da la señal de gtk_main_quit nunca termina la aplicación.

Dentro del mismo directorio se crearon archivos (***Temp***, ***tempuno***, ***uno.pov y textura***), estos archivos son necesarios para que el programa funcione correctamente. En el archivo temp escribieron los valores que seleccionan del editor este archivo está rescribiendo cada vez que se actualicen los valores, en el archivo uno.pov se escribió el código necesario para ser ejecutado por Povray este archivo consta de 13 líneas constantes en las que se especifica la esfera, la fuente de luz y la cámara para el resto del código se tomará del archivo temp que contiene los valores seleccionados del editor, es necesario asegurarse de contar con las primeras 13 líneas del archivo uno.pov para no sobrescribir código, es por eso que se utiliza el archivo tempuno en donde se almacena de forma temporal el contenido

de estas 13 líneas, en los archivos con el nombre de textura contienen el código necesario para que tenga apariencia de las diversas texturas más conocidas.

Temp	Tempuno	Uno.pov

Inicia en blanco y se estará rescribiendo	#include "colors.inc"
	background { color White }
	camera {
	location <0, 2, -3>
	look_at <0, 1, 2>
	
	}
	light_source { <2, 4, -3> color White}
	
	sphere {
	<0, 1, 2>, 2
	texture {
	pigment {
		#include "colors.inc"
		background { color White }
		camera {
		location <0, 2, -3>
		look_at <0, 1, 2>
		
		}
		light_source { <2, 4, -3> color White}
		
		sphere {
		<0, 1, 2>, 2
		texture {
		pigment {
		codigo que se escriba en Temp
		}

Tabla4.1 archivo necesarios para renderizar

Estos archivos son necesarios para renderizar, para poder hacer estos utilizamos el programa Povray que se puede descargar gratuitamente desde la página de htt://www.povray.org para Linux.

En el archivo de Callbacks.c escribimos el código para que funcione el selector de color y las barras desplazamiento que se encuentran en las libretas de parámetros físicos, básicos e iridiscencia, se escribe en el botón de aplicar, donde también se declara el nombre del archivo de temp para su uso posterior.

```
{
  FILE *fp;
  gchar archivo[10]="temp";
  GtkWidget *range;
  GtkColorSelection *colorsel;
  GtkWidget *toggle_button;
  gdouble ambiente, refle, radio, spec, refra, roug, anchura, canti, turbu;
  gdouble color[4];
  gdouble r,g,b,o;
  gboolean tex1, tex2, tex3, tex4, tex5, tex6, tex7, tex8, tex9, tex10,
 tex11, tex12, tex13, tex14, tex15, tex16, tex17, tex18, tex19, tex20,
 tex21, tex22, tex23, tex24, tex25, tex26, tex27, tex28, tex29, tex30,
```

```
tex31, tex32, tex33, tex34, tex35, tex36, tex37, tex38, tex39, tex40,
tex41, tex42, tex43;
toggle_button=lookup_widget(GTK_WIDGET(button),"radiobutton1");
tex1=gtk_toggle_button_get_active(GTK_CHECK_BUTTON(toggle_button));
toggle_button=lookup_widget(GTK_WIDGET(button),"radiobutton2");
tex2=gtk_toggle_button_get_active(GTK_CHECK_BUTTON(toggle_button));
toggle_button=lookup_widget(GTK_WIDGET(button),"radiobutton3");
tex3=gtk_toggle_button_get_active(GTK_CHECK_BUTTON(toggle_button));
toggle_button=lookup_widget(GTK_WIDGET(button),"radiobutton4");
tex4=gtk_toggle_button_get_active(GTK_CHECK_BUTTON(toggle_button));
toggle_button=lookup_widget(GTK_WIDGET(button),"radiobutton5");
tex5=gtk_toggle_button_get_active(GTK_CHECK_BUTTON(toggle_button));
toggle_button=lookup_widget(GTK_WIDGET(button),"radiobutton6");
tex6=gtk_toggle_button_get_active(GTK_CHECK_BUTTON(toggle_button));
toggle_button=lookup_widget(GTK_WIDGET(button),"radiobutton7");
tex7=gtk_toggle_button_get_active(GTK_CHECK_BUTTON(toggle_button));
toggle_button=lookup_widget(GTK_WIDGET(button),"radiobutton8");
tex8=gtk_toggle_button_get_active(GTK_CHECK_BUTTON(toggle_button));
toggle_button=lookup_widget(GTK_WIDGET(button),"radiobutton9");
tex9=gtk_toggle_button_get_active(GTK_CHECK_BUTTON(toggle_button));
toggle_button=lookup_widget(GTK_WIDGET(button),"radiobutton10");
tex10=gtk_toggle_button_get_active(GTK_CHECK_BUTTON(toggle_button));
toggle_button=lookup_widget(GTK_WIDGET(button),"radiobutton11");
tex11=gtk_toggle_button_get_active(GTK_CHECK_BUTTON(toggle_button));
toggle_button=lookup_widget(GTK_WIDGET(button),"radiobutton12");
tex12=gtk_toggle_button_get_active(GTK_CHECK_BUTTON(toggle_button));
toggle_button=lookup_widget(GTK_WIDGET(button),"radiobutton13");
tex13=gtk_toggle_button_get_active(GTK_CHECK_BUTTON(toggle_button));
toggle_button=lookup_widget(GTK_WIDGET(button),"radiobutton14");
tex14=gtk_toggle_button_get_active(GTK_CHECK_BUTTON(toggle_button));
toggle_button=lookup_widget(GTK_WIDGET(button),"radiobutton15");
tex15=gtk_toggle_button_get_active(GTK_CHECK_BUTTON(toggle_button));
toggle_button=lookup_widget(GTK_WIDGET(button),"radiobutton16");
tex16=gtk_toggle_button_get_active(GTK_CHECK_BUTTON(toggle_button));
toggle_button=lookup_widget(GTK_WIDGET(button),"radiobutton17");
tex17=gtk_toggle_button_get_active(GTK_CHECK_BUTTON(toggle_button));
toggle_button=lookup_widget(GTK_WIDGET(button),"radiobutton18");
tex18=gtk_toggle_button_get_active(GTK_CHECK_BUTTON(toggle_button));
toggle_button=lookup_widget(GTK_WIDGET(button),"radiobutton19");
tex19=gtk_toggle_button_get_active(GTK_CHECK_BUTTON(toggle_button));
toggle_button=lookup_widget(GTK_WIDGET(button),"radiobutton20");
tex20=gtk_toggle_button_get_active(GTK_CHECK_BUTTON(toggle_button));
toggle_button=lookup_widget(GTK_WIDGET(button),"radiobutton21");
tex21=gtk_toggle_button_get_active(GTK_CHECK_BUTTON(toggle_button));
toggle_button=lookup_widget(GTK_WIDGET(button),"radiobutton22");
tex22=gtk_toggle_button_get_active(GTK_CHECK_BUTTON(toggle_button));
toggle_button=lookup_widget(GTK_WIDGET(button),"radiobutton23");
tex23=gtk_toggle_button_get_active(GTK_CHECK_BUTTON(toggle_button));
toggle_button=lookup_widget(GTK_WIDGET(button),"radiobutton24");
tex24=gtk_toggle_button_get_active(GTK_CHECK_BUTTON(toggle_button));
toggle_button=lookup_widget(GTK_WIDGET(button),"radiobutton25");
tex25=gtk_toggle_button_get_active(GTK_CHECK_BUTTON(toggle_button));
toggle_button=lookup_widget(GTK_WIDGET(button),"radiobutton26");
tex26=gtk_toggle_button_get_active(GTK_CHECK_BUTTON(toggle_button));
toggle_button=lookup_widget(GTK_WIDGET(button),"radiobutton27");
tex27=gtk_toggle_button_get_active(GTK_CHECK_BUTTON(toggle_button));
toggle_button=lookup_widget(GTK_WIDGET(button),"radiobutton28");
tex28=gtk_toggle_button_get_active(GTK_CHECK_BUTTON(toggle_button));
toggle_button=lookup_widget(GTK_WIDGET(button),"radiobutton29");
```

```
tex29=gtk_toggle_button_get_active(GTK_CHECK_BUTTON(toggle_button));
toggle_button=lookup_widget(GTK_WIDGET(button),"radiobutton30");
tex30=gtk_toggle_button_get_active(GTK_CHECK_BUTTON(toggle_button));
toggle_button=lookup_widget(GTK_WIDGET(button),"radiobutton31");
tex31=gtk_toggle_button_get_active(GTK_CHECK_BUTTON(toggle_button));
toggle_button=lookup_widget(GTK_WIDGET(button),"radiobutton32");
tex32=gtk_toggle_button_get_active(GTK_CHECK_BUTTON(toggle_button));
toggle_button=lookup_widget(GTK_WIDGET(button),"radiobutton33");
tex33=gtk_toggle_button_get_active(GTK_CHECK_BUTTON(toggle_button));
toggle_button=lookup_widget(GTK_WIDGET(button),"radiobutton34");
tex34=gtk_toggle_button_get_active(GTK_CHECK_BUTTON(toggle_button));
toggle_button=lookup_widget(GTK_WIDGET(button),"radiobutton35");
tex35=gtk_toggle_button_get_active(GTK_CHECK_BUTTON(toggle_button));
toggle_button=lookup_widget(GTK_WIDGET(button),"radiobutton36");
tex36=gtk_toggle_button_get_active(GTK_CHECK_BUTTON(toggle_button));
toggle_button=lookup_widget(GTK_WIDGET(button),"radiobutton37");
tex37=gtk_toggle_button_get_active(GTK_CHECK_BUTTON(toggle_button));
toggle_button=lookup_widget(GTK_WIDGET(button),"radiobutton38");
tex38=gtk_toggle_button_get_active(GTK_CHECK_BUTTON(toggle_button));
toggle_button=lookup_widget(GTK_WIDGET(button),"radiobutton39");
tex39=gtk_toggle_button_get_active(GTK_CHECK_BUTTON(toggle_button));
toggle_button=lookup_widget(GTK_WIDGET(button),"radiobutton40");
tex40=gtk_toggle_button_get_active(GTK_CHECK_BUTTON(toggle_button));
toggle_button=lookup_widget(GTK_WIDGET(button),"radiobutton41");
tex41=gtk_toggle_button_get_active(GTK_CHECK_BUTTON(toggle_button));
toggle_button=lookup_widget(GTK_WIDGET(button),"radiobutton42");
tex42=gtk_toggle_button_get_active(GTK_CHECK_BUTTON(toggle_button));
toggle_button=lookup_widget(GTK_WIDGET(button),"radiobutton43");
tex43=gtk_toggle_button_get_active(GTK_CHECK_BUTTON(toggle_button));
range=lookup_widget(GTK_WIDGET(button),"hscale8");
ambiente=gtk_range_get_value(GTK_RANGE(range));
range=lookup_widget(GTK_WIDGET(button),"hscale9");
radio=gtk_range_get_value(GTK_RANGE(range));
range=lookup_widget(GTK_WIDGET(button),"hscale10");
refle=gtk_range_get_value(GTK_RANGE(range));
range=lookup_widget(GTK_WIDGET(button),"hscale14");
spec=gtk_range_get_value(GTK_RANGE(range));
range=lookup_widget(GTK_WIDGET(button),"hscale15");
refra=gtk_range_get_value(GTK_RANGE(range));
range=lookup_widget(GTK_WIDGET(button),"hscale16");
roug=gtk_range_get_value(GTK_RANGE(range));
range=lookup_widget( GTK_WIDGET(button),"hscale11");
anchura=gtk_range_get_value(GTK_RANGE(range));
range=lookup_widget(GTK_WIDGET(button),"hscale12");
canti=gtk_range_get_value(GTK_RANGE(range));
range=lookup_widget(GTK_WIDGET(button),"hscale13");
turbu=gtk_range_get_value(GTK_RANGE(range));
colorsel=lookup_widget(GTK_WIDGET(button),"colorselection1");
gtk_color_selection_get_color(GTK_COLOR_SELECTION(colorsel),color);
r=color[0];
g=color[1];
b=color[2];
o=color[3];
//  g_print("\n\t LA textura es:%d \n", textura1);
if (tex43==1)
{
// g_print("\n\t LA textura es:%d \n", tex43);
  if ((fp = fopen(archivo,"w")) == NULL)
{
```

```
    printf(" no se puede abrir el archivo\n");
    exit(1);
  }
  else
    {
       fprintf(fp,"color rgb < %f, %f, %f > } finish { ambient %f diffuse
%f reflection %f specular %f refraction %f  roughness %f irid { %f
thickness %f turbulence %f }} }}",r,g,b,ambiente,radio,refle,spec,refra,
roug);
  }
   fclose(fp);
}
   if (tex1==1)

      system("cp metal1 temp");
 if (tex2==1)
    system("cp metal2 temp");
 if (tex3==1)
    system("cp metal3 temp");
 if (tex4==1)
    system("cp metal4 temp");
 if (tex5==1)
    system("cp metal5 temp");
 if (tex6==1)
    system("cp metal6 temp");
 if (tex7==1)
    system("cp glass1 temp");
 if (tex8==1)
    system("cp glass2 temp");
 if (tex9==1)
    system("cp glass3 temp");
 if (tex10==1)
    system("cp plastico1 temp");
 if (tex11==1)
    system("cp plastico2 temp");
 if (tex12==1)
    system("cp plastico3 temp");
 if (tex13==1)
    system("cp wood1 temp");
 if (tex14==1)
    system("cp wood2 temp");
 if (tex15==1)
    system("cp wood3 temp");
 if (tex16==1)
    system("cp wood4 temp");
 if (tex17==1)
    system("cp wood5 temp");
 if (tex18==1)
    system("cp wood6 temp");
 if (tex19==1)
    system("cp stone1 temp");
 if (tex20==1)
    system("cp stone2 temp");
 if (tex21==1)
    system("cp stone3 temp");
 if (tex22==1)
    system("cp stone4 temp");
 if (tex23==1)
    system("cp stone5 temp");
```

```
 if (tex24==1)
      system("cp stone6 temp");
  if (tex25==1)
     system("cp skies1 temp");
  if (tex26==1)
     system("cp skies2 temp");
  if (tex27==1)
     system("cp skies3 temp");
  if (tex28==1)
     system("cp skies4 temp");
  if (tex29==1)
     system("cp skies5 temp");
  if (tex30==1)
     system("cp skies6 temp");
  if (tex31==1)
     system("cp oceano1 temp");
  if (tex32==1)
     system("cp oceano2 temp");
 if (tex33==1)
     system("cp oceano3 temp");
  if (tex34==1)
     system("cp oceano4 temp");
  if (tex35==1)
     system("cp oceano5 temp");
  if (tex36==1)
     system("cp oceano6 temp");
  if (tex37==1)
     system("cp animal1 temp");
  if (tex38==1)
     system("cp animal2 temp");
  if (tex39==1)
     system("cp animal3 temp");
  if (tex40==1)
     system("cp animal4 temp");
  if (tex41==1)
     system("cp animal5 temp");
}
```

Para renderizar se utiliza el botón creado para este fin con el siguiente código.


```
FILE *fp, *fpt;
char temporal[10]="temp";
char archivo[10]="uno.pov";
int bytes_leidos;
char buffer[2000];
system("cat uno.pov|head -13 >tempuno");
system("cp tempuno uno.pov");
if ((fp = fopen(temporal,"r")) == NULL)
        {
                printf(" no se puede abrir el archivo\n");
                   exit(1);
                    }
if ((fpt = fopen(archivo,"a")) == NULL)
        {
                printf(" no se puede abrir el archivo\n");
                   exit(1);
```

```
                }
while(bytes_leidos=fread(buffer,1,2000,fp))}
fwrite(buffer,1,bytes_leidos,fpt);}
fclose(fp);
fclose(fpt);
system("/home/lupe/povary/povray-3.5/povray +w160 +h120 +p +x +d0 -v -
iuno.pov");
```

En este código se crean dos apuntadores los cuales son para abrir archivos temp y uno.pov, en el system cat cortará las 13 primera líneas del código de uno.pov a tempuno ya que no se puede rescribir en el mismo archivo y es necesario escribirlo en otro ya que linux no permite eso, en la siguiente línea de system cp pega el código de tempuno a uno.pov, en la línea del while realiza un ciclo desde el inicio del archivo de temp hasta el final del mismo, el fwrite escribe en el archivo uno.pov, con las líneas fclose cerramos los archivos y con la línea de system("/home/lupe/povary/povray-3.5/povray +w160 +h120 +p +x +d0 -v -iuno.pov") mandamos llamar con los parámetros a povray para renderizar figura 4.8.

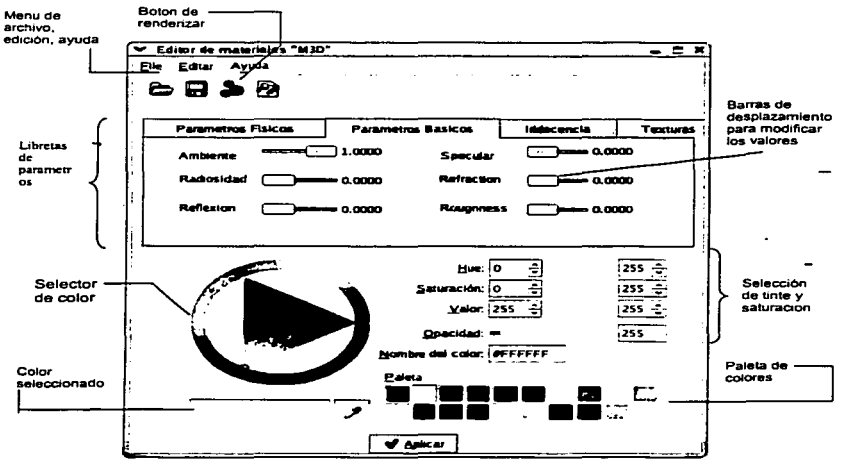


Figura 4.8 Editor de materiales

4.2 EVALUACIÓN

Para la evaluación del Editor se les dio a las personas que están enfocadas en las áreas del desarrollo y graficación, al igual para aquellas que no tiene ninguna relación hacia el área. Se llevaron acabo las siguientes características de calidad, tales como: pruebas, análisis de problemas, revisiones, entre otras.

Tomando en cuenta algunas de las herramientas que son necesarias para evaluar como son: capacidad, dominio y entrenamiento para utilizar el editor.

Dentro de la evaluación encontramos la operacionabilidad ésta mostrará la manera de cómo funciona el editor así como su facilidad, la velocidad de su proceso para insertar una textura. Dentro de su desarrollo funcional se evalúa el funcionamiento y las áreas en las cuales se puede utilizar.

Hoy en día nos encontramos con una gran diversidad de software que está enfocado en la graficación y en la modelación de objetos, la necesidad de diseñar un software diferente nos lleva contemplar algunas normas de calidad, ya que el crecimiento de software hace la diferencia en la selección, considerando métodos de evaluación algunos de estos criterios es el análisis de ventajas y desventajas de los mismos (por ejemplo):

- Acceso fácil.
- Integración con otro software.
- Requerimientos mínimos de instalación.
- Manejabilidad con otros sistemas.
- Costo.
- Velocidad de respuesta.

El editor de texturas para "Material 3D" fue diseñado pensando en el desarrollo de aplicaciones de software libre para la creación de materiales, es necesario instalar en el sistema operativo la versión de povray para la generación de imagen, una de

las partes fundamentales del editor es la creación de los colores y la selección de las texturas.

El funcionamiento del editor es más amigable a diferencia de otros tales como el AutoCad, 3DMáx ya que la aplicación de otros programas es algo confusa en su funcionamiento debido a que tiene muchos elementos, a diferencia del editor que utiliza parámetros básicos muy fáciles de aplicar.

Cabe mencionar que para que el editor pueda renderizar, es necesario cambiar en el archivo de callbacks.c en la linea de system la nueva ubicación de donde se encuentra ahora povray system ("/home/lupe/povray/povray-3.5/povray +w160 +h120 +p +x +d0 –v –iuno.pov").

Respecto a los criterios de calidad el acceso es muy fácil, la integración con otro software siempre y cuando sea compatible, su código fuente queda libre con la intención de que sirva a otras personas y sea modificado para otras aplicaciones.

Dentro de la evaluación se llevaron acabo algunas preguntas a los evaluadores del editor y son las siguientes en donde el 90% de los evaluadores encontraron una buena calidad del editor.

¿La funcionalidad del editor fue? Muy buena.
¿El acceso al editor fue? Muy buena.
¿El tiempo para aprender a utilizar el editor fue? Rápida.
¿La calidad de los materiales fue? Buena.
El tiempo de respuesta del editor fue? Rápida.
¿La opinión al respecto del editor en forma general fue? Buena.

Alguna de las desventajas que se encontró en la evaluación fue que a las personas que no están en el área de la graficación les incomodo el estar instalando povray y el modificar la línea de system.

las partes fundamentales del editor es la creación de los colores y la selección de las texturas.

El funcionamiento del editor es más amigable a diferencia de otros tales como el AutoCad, 3DMáx ya que la aplicación de otros programas es algo confusa en su funcionamiento debido a que tiene muchos elementos, a diferencia del editor que utiliza parámetros básicos muy fáciles de aplicar.

Cabe mencionar que para que el editor pueda renderizar, es necesario cambiar en el archivo de callbacks.c en la linea de system la nueva ubicación de donde se encuentra ahora povray system ("/home/lupe/povray/povray-3.5/povray +w160 +h120 +p +x +d0 –v –iuno.pov").

Respecto a los criterios de calidad el acceso es muy fácil, la integración con otro software siempre y cuando sea compatible, su código fuente queda libre con la intención de que sirva a otras personas y sea modificado para otras aplicaciones.

Dentro de la evaluación se llevaron acabo algunas preguntas a los evaluadores del editor y son las siguientes en donde el 90% de los evaluadores encontraron una buena calidad del editor.

¿La funcionalidad del editor fue? Muy buena.
¿El acceso al editor fue? Muy buena.
¿El tiempo para aprender a utilizar el editor fue? Rápida.
¿La calidad de los materiales fue? Buena.
El tiempo de respuesta del editor fue? Rápida.
¿La opinión al respecto del editor en forma general fue? Buena.

Alguna de las desventajas que se encontró en la evaluación fue que a las personas que no están en el área de la graficación les incomodo el estar instalando povray y el modificar la linea de system.

CONCLUSIONES

Fue de gran importancia dar como referencia algunos de los conceptos básicos de modelado de 3D y 2D ya que son muy utilizados en la graficación y en el modelado de objetos, cabe mencionar algunos de los modelos de color tales como el RGB y el CMY ya que estos son los más utilizados en el modelado, y en la representación de las superficies, consideramos que el modelado por ecuaciones parámetricas y las mallas poligonales tienen sus ventajas y sus desventajas las cuales son el espacio que utilizan, sin son fáciles de modelar entre otras, gracias a eso podemos tomar una decisión en escoger que tipo de representación de la superficie utilizará el modelo.

Cabe mencionar la importancia que tiene el utilizar algún modelo de iluminación ya que con ello el objeto a modelar llega a tener el realismo que se necesita, cada uno de estos modelos que se mencionaron en esta primera parte no tiene un criterio único para implementar la iluminación necesaria para generar efectos especiales en el modelado del objeto.

El desarrollo de GUI's en la actualidad juega un papel importante en el diseño de nuevo software, el software que se diseña actualmente no podría ser competitivo sin una interfaz gráfica agradable para el usuario, es importante mencionar que esto no significa que se pierda el objetivo y la eficiencia del programa, existen muchas aplicaciones de software que buscan hacer tan interactivo el programa con imágenes y animaciones, que la función del programa pasa a segundo plano.

GTK+ brinda potentes herramientas de programación para crear interfaces gráficas y con la ayuda de GLADE se realiza de forma sencilla y reutilizando código, esto ofrece al programador una gran ayuda al no tener que preocuparse por programar desde cero una interfaz gráfica, dejando que se concentre en la parte de programación la funcionalidad del programa.

El editor de texturas para M3D es creado pensando en el usuario, con una interfaz gráfica que sea fácil de manejar, pero un código interno eficiente que no genere errores.

GLADE con la ayuda de la biblioteca GTK+ es una eficiente herramienta en el desarrollo de GUI's para LINUX, ayuda a la reutilización de código en la construcción de las interfaces graficas, dejando lista la interfaz para ser programada un algún lenguaje compatible.

Después de analizar distintos tipos de editores, el diseño del editor de texturas para M3D está basado en las necesidades de los programas para generar modelados en tercera dimensión, cuenta con las características de cualquier editor de texturas como son: color, brillo radiosidad, etc., y se implementan nuevas características físicas como el volumen peso, etc., que ayudarán a realizar cálculos físicos sobre los objetos, su utilización es sencilla tratando de no confundir al usuario con muchas opciones, pero poniendo las necesarias para crear textura que puedan dar a los objetos mayor realismo.

Es importante el uso de una metodología para el desarrollo de software, UML brinda herramientas que facilitan el diseño y desarrollo de software de calidad, ayudando al programador a llevar un orden así como la plantación del desarrollo del software.

Es importante destacar que se adquieren conocimientos que permiten ampliar la visión de lo que se refiere a los modeladores de 3D y 2D. Se puede destacar el trabajo que realiza Moray que es un modelador de marco de alambre el cual utiliza a Povray para su renderizado, otro ejemplo de esto es el editor de texturas de 3D Studio Max estos modeladores contienen su editor de texturas que tienen semejanzas afines tales como el selector de colores, biblioteca de texturas, sombras, iluminaciones etc. Cada uno de estos programas tienen características que los diferencian de uno con el otro, teniendo en común el crear objetos lo más realista, utilizando muy pocos recursos de la computadora, excepto 3D máx.

Con lo expuesto en esta parte sirve como base para realizar cualquier objeto a modelar y para encontrar sus ventajas y desventajas de estos modeladores, en el transcurso del estudio se identifican el funcionamiento de cada uno de ellos.

Consideramos que al término de este trabajo hemos contribuido al desarrollo del software libre, el éxito que está teniendo el desarrollo del software libre en México y en el mundo se debe a la creación de pequeñas aplicaciones que al conjuntarse con otras, crean grandes aplicaciones que compiten con el software comercial.

Las pruebas de evaluación que se han realizado con el Editor muestra que se ha cumplido el objetivo que nos planteamos al desarrollar un Editor que sirva para crear texturas que puedan darle una mayor realidad a los objetos, creemos que el código se puede optimizar mas si se programa todo con lenguaje C, y se pueden modificar las texturas seleccionadas por default para poder interactuar con ellas, claro que esto implicaría una gran cantidad de código, pero mejoraría la funcionalidad del programa.

BIBLIOGRAFÍA

- Graficas por computadora.
  Donald Hearn.
  Prentice-Hall Hispanoamericana, S.A.

- Graficas por computadora.
  Roy A. Plastock.
  Mc-Graw-Hill.

- Gráficos animados
  David Fox y Mitchell Waite.
  Byte Books/Mc-Graw-Hill.

- Técnicas de iluminación en 3D max4
  Juan Carlos Jiménez Vadillo
  Anaya Multimedia.

- 3D Studio max 3 manual practico
  Catell Cebolla
  RA-MA

- 3D Studio max 3.1
  David J. Kalwick.
  Prentice Hall

- Ingenieria de software orientado a objetos
  Bernd Bruegge allen h. Dutoit
  Prentice hall

- Autocad 2000 practico
  Jordi Crosi Ferradiz
  Inforbook's S.L.

- Desarrollo de aplicaciones Linux con GTK+ y GDK
  Eric Harlow
  Prentice Hall

- http://tigre.aragon.unam.mx/m3d/glade/glade_manual.html
- http://www.gtk.org/tutorial/
- http://www.gtk.org/
- http://glade.gnome.org/
- http://www.povray.org/