UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO

POSGRADO DE CIENCIA E INGENIERÍA DE LA COMPUTACIÓN

ALGORITMOS DE EXPANSIÓN DE DATOS APLICADOS A LA CONTAMINACIÓN ATMOSFÉRICA DE LA CIUDAD DE MÉXICO

T E S I S

QUE PARA OBTENER EL GRADO DE:

MAESTRO EN CIENCIAS

P R E S E N T A:

FRANCISCO HIRAM CALVO CASTRO

DIRECTOR DE TESIS:
DR VLADIMIR TCHIJOV

Cuautitlán Izcalli, Edo. De Mex 2002

A mis padres
A mi escuela FES-Cuautitlán

Con cariño

Objetivos.

Para el desarrollo de esta tesis, se plantean los siguientes objetivos:

1 Seleccionar de los Algoritmos de Expansión de Datos utilizados en problemas similares al de contaminación atmosférica, 3 candidatos a implementar para evaluar su desempeño

2 Implementar los Algoritmos de Expansión de Datos seleccionados en lenguaje C++, de manera modular y reutilizable, con una interfaz portable

3. Visualizar las mallas generadas en tiempo real utilizando OpenGL integrado a los Algoritmos de Expansión de Datos implementados

4. Determinar la eficacia de cada uno de los Algoritmos de Expansión de Datos, probados con funciones muestra y con datos reales obtenidos de las estaciones de medición de contaminantes de la Ciudad de México

Introducción

En el mundo real, es imposible obtener valores exhaustivos de datos en cada punto deseado debido a las restricciones prácticas Es por esto que la interpolación es importante y fundamental para comprender, analizar y graficar los datos bidimensionales [DEVILLERS 1998]

Debido a que la interpolación de un conjunto de datos dispersos no tiene una única solución, es importante que los cálculos se realicen cuidadosamente, produciendo superficies resultantes realistas físicamente El campo interpolado es crítico para cálculos como la divergencia de campo de viento, reducción, contorno de valores de datos, e inicialización de cálculos de transporte de contaminantes [WATSON 1982]

En este trabajo se presenta inicialmente un panorama acerca de la situación actual del software que permite realizar interpolación de datos Algunos de ellos especializados en aplicaciones geológicas. Posteriormente, centraremos nuestra atención en tres algoritmos de expansión de datos: Cressman, Voronoi y *Kriging* Se describen los fundamentos matemáticos de cada uno de ellos, y algunos detalles acerca de su implementación

Una vez que tenemos las herramientas teóricas matemáticas necesarias, investigamos las herramientas que nos permitirán visualizar de manera interactiva las superficies de los resultados Después de analizar diversas opciones para realizar esta representación gráfica tridimensional, se decidió trabajar utilizando OpenGL, conjuntamente con la librería de utilidades GLUT (Graphics Library Utility Toolkit) Entre las razones para haber elegido estas librerías, se encuentran: portabilidad, documentación extensa, gratuidad y efectividad

En el CAPÍTULO 3 se muestra el software que se desarrolló, incluyendo una interfaz gráfica programada en μ-UI (micro- User Interface) que preserva las características convenientes de portabilidad mencionadas anteriormente, pues trabaja interrelacionadamente con GLUT

Por último, habiendo conseguido datos reales de contaminación atmosférica, que incluyen posición de las estaciones de medición de contaminantes en nuestra ciudad (Ciudad de México), y valores tomados de los diversos contaminantes, se procede a alimentar el programa, para obtener interpolaciones numéricas que serán comparadas con los valores originales de las lecturas, en las regiones formadas por la malla donde se encuentren los puntos con el valor conocido De esta forma, puede determinarse el algoritmo que difiere en menor grado de los valores reales, teniendo una base para comparar los diversos algoritmos

Como apéndices se incluyen las traducciones de diversos manuales de referencia y guía del usuario de OpenGL y GLUT, con el fin de poner al alcance de la comunidad de habla hispana estas herramientas de visualización gráfica

1.1 Estado del Arte

1.1.1 Software Geológico de Modelación.

Actualmente existen diversos paquetes que incluyen algoritmos de expansión de datos Esta es una tarea requerida frecuentemente en aplicaciones GIS (Geological Information Systems). La mayoría de estos paquetes son comerciales, y su costo es bastante elevado

Podemos clasificar este software, basándonos primero en su precisión y exactitud [STROUSTRUP 1997], después en su generalización, y por último, considerando las capacidades de estos programas Para esto último, tenemos las clasificaciones de software para agrimensura, de delineado, de modelaje estratiforme, de eventos locales o historia, e interacciones de placas o tectonostratigrafía

A continuación se muestra la taxonomía de algunos de los paquetes existentes de Software Geológico de Modelación

<table>
<tr><td colspan="4">← Precisión / Exactitud</td></tr>
<tr><td colspan="4">Generalización →</td></tr>
<tr><td rowspan="2">Geometría</td><td colspan="2">local / mining</td><td>regional / exploración</td></tr>
<tr><td colspan="2">Para agrimensura:
• 2D: GIS
• Vulcan
• Datamine
• CAD</td><td>Delineado
• 2D: GIS, GeoSec
• 3D: REGI
• 4D: Knoxel Ed</td></tr>
<tr><td></td><td colspan="2">Modelaje Estratiforme
• earthVision
• goCAD
• GeoSec3D</td></tr>
<tr><td>Tiempo (Geología / escenarios)</td><td colspan="2">Eventos locales / Historia

Escenarios estructurales / dibujos
• Noddy</td><td>Interacciones de placas / Tectonostratigrafía

Sistemas petrolíferos / de minerales
• PREDICT</td></tr>
</table>

Como podemos apreciar, no existe un software especializado en temas de contaminación atmosférica, aunque los algoritmos estudiados son utilizados con otros propósitos, como es el caso de GMS, un sistema para modelaje de tierras submarinas, que utiliza el algoritmo de *Kriging*

1.1.1.1 GMS- Sistema de Modelaje de Tierras Submarinas

Entre otras características, este sistema tiene la opción de realizar interpolación de superficie utilizando el algoritmo de *Kriging*

***Kriging* Zonal**

Puede definirse un variograma para cada zona estratográfica Las zonas se definen por los identificadores de material asociados con las celdas de una malla tridimensional que rodea a los puntos diseminados Cuando se interpola una celda en una zona, sólo los puntos diseminados en la misma zona que la malla se usan para interpolar la celda

Simulación de Indicador

El *Kriging* de indicador es una forma de *Kriging* que se utiliza para interpolar información integral o zonal en vez de valores escalares A cada punto diseminado se le asigna un identificador de material y los identificadores de material se interpolan a las celdas de la malla. Esto hace posible que se establezcan zonas estratográficas para un modelo de tierra submarina vía *Kriging*

Por otra parte, existen programas comerciales que pueden realizar, con un propósito más general, y no enfocándose exclusivamente al Software de Modelación Geológico, utilizando mallas Podemos considerar nuestro problema de ajustar valores de datos dispersos a una patrón regular, como un problema de ajuste de mallas

1.2 Programas comerciales para manejo de mallas.

Existe una gran cantidad de programas para manejo de mallas, debido a que muchas aplicaciones se apoyan en el uso de ellas A continuación se muestra un resumen de algunos de estos programas

1.2.1 Algor

Un sistema de modelaje de superficie basado en NURBS[1] tridimensionales El sistema permite crear paramétricamente y manipular la malla sin cambiar la geometría

[1] NURBs Significa, en Inglés: Non Uniform Rational B-Spilines Son una técnica matemática para definir un objeto tridimensional Debido a que los NURBs están compuestos de curvas, son ideales para crear objetos "orgánicos" [página de Unrealized]

original La densidad de la malla puede ser controlada sobre el modelo completo o en áreas designadas.

Referencia: http://www.algor.com/products.htm
Plataformas: Unix, DOS, SGI, Hp, Sun

1.2.2 ARGUS MeshMaker

Generador de mallas bidimensionales de GIS y CAD. Los Entornos Numéricos Argus (ANE por sus siglas en inglés) [http://www.slope-analysis.com/inane.htm] son una familia de procesadores gráficos previos y posteriores para el modelador numérico. Representan un nuevo enfoque de combinar GIS y modelaje numérico.

Referencia: http://www.argusint.com/
Plataformas: Macintosh, PowerMac, PC (Windows 3.x, Windows 95, Windows NT), IBM RS 6000, IBM PowerPC, versiones para HP, Sun y estaciones de trabajo DEC a ser lanzadas

1.2.3 EarthVision

Modelaje basado en retículas de superficies bidimensionales y propiedades tridimensionales para Ciencias de la Tierra. Usa diversos métodos de interpolación de retícula para calcular las distribuciones de propiedades de materiales. Diversos métodos de visualización interactiva e interfaces a SURFER, AutoCAD y ARC/INFO.

Referencia: http://www.dgi.com/ev.html
Plataforma: UNIX, SGI y Sun

1.2.4 FEMAP

Un pre- y post-procesador de propósito general para Análisis de Elemento Finito en Ingeniería. El Módulo de Mallas Avanzado puede importar y convertir a mallas modelos sólidos de AutoCAD, AutoCAD Designer, MicroStationModeler, o cualquier otro paquete CAD basado en ACIS. El Módulo de Mallas Avanzado contiene además una interfaz para archivos de estereolitografía para convertir a mallas desde otros sistemas CAD.

Referencia: http://www.entsoft.com/
Plataformas: Unix, Windows, Windows 95, Windows NT

1.2.5 Finite-Octree

Generador automático de mallas tridimensionales capaz de generar mallas de complejidad arbitraria

Referencia: http://www.scorec.rpi.edu/
Plataformas: Unix, compilador f77

1.2.6 GOCAD9.3

Diseñado para guiar la construcción de superficies tridimensionales realistas. Varias restricciones como puntos, líneas, inclinación, distancia, se utilizan para controlar las superficies trianguladas. El espacio tridimensional puede ser discretizado usando retículas regulares, irregulares, o tetraedros. Las propiedades discretas pueden ser interpoladas utilizando el propio interpolador de GOCAD (DSI) o usando métodos geoestadísticos (*Kriging* y simulación)

Referencia: http://www.ensg.u-nancy.fr/GOCAD/index.html
Plataformas: Unix, SGI, Hp, Sun, Dec

1.2.7 GRIDGEN

Es software interactivo orientado gráficamente para la generación de retículas tridimensionales estructuradas de múltiples bloques. El sistema consiste de dos códigos: Gridgen interactivo y Gridgen3D de procesamiento por lotes. Estos dos códigos pueden crear celdas cuadriláteras y hexaédricas, mallas mapeadas, bloques sencillos o múltiples, interfaces abultantes o con bloques traslapados y bloques bi- y tridimensionales.

Referencia: http://www.pointwise.com/
Plataformas: Unix, SGI, Hp

1.2.8 Hypermesh

Crea modelos de elemento finito y diferencia finita bi- y tridimensionales para simulación ingenieril y análisis. Soporta un amplio espectro de códigos de análisis en un sistema integrado. Uso directo de geometría CAD, así como elementos existentes de modelo finito.

Referencia: http://www.altair.com/Comp/Products/home_page.html
Plataformas: Unix, DOS

1.2.9 ICEM CFD

Crea retículas de multibloque estructuradas, retículas no estructuradas tetraédricas y superficies triangulares, retículas cartesianas ajustadas al cuerpo, retículas-H[2] refinadas de geometría CAD tridimensional existente Usa formatos abiertos estándar como IGES, VDA/FS, SET, DXF

Referencia: http://icemcfd.com/icemcfd.html
Plataformas: Unix

1.2.10 ICEM CFD Hexa

Es un módulo de creación de mallas hexaédricas semiautomatizado en ICEM CFD que provee generación rápida de mallas de volumen de multibloque estructuradas o sin estructura

Referencia: http://icemcfd.com/hexa.html
Plataformas: Unix

1.2.11 Patran

Es un entorno de propósito general para MCAE[3] tridimensional Se usa principalmente en estaciones de trabajo para la creación de modelos de elemento finito usados en ABAQUS Patran puede también desplegar los resultados de ABAQUS

Referencia: http://www.macsch.com/products/patran/patran.html
Plataformas: Unix

1.2.12 STRATAMODEL

Un sistema de modelaje tridimensional con un enfoque celular Usa métodos basados en retículas bi- y tridimensionales y permite una visualización interactiva, así como diversas funciones de mapeo/ploteo Usa diversos algoritmos de interpolación de retícula para calcular distribuciones de material

---

[2] Las Retículas H son retículas donde los elementos no concuerdan necesariamente 1 a 1, sino que pueden concordar de 1 a n

[3] MCAE: Mechanical Computer Aided Engineering Ingeniería Mecánica Asistida por Computadora En contraposición con el MCAD, Diseño Mecánico Asistido por Computadora que se enfoca en el diseño y modelaje del producto, MCAE prueba la fuerza de estos diseños por simulación Cuando estas dos disciplinas trabajan en conjunción, el producto final logra excelencia tanto en funcionalidad como en calidad en tanto que se reducen los ciclos de desarrollo [http://wwwwswest sun com/technical-computing/mcad html]

Referencia: http://www.lgc.com/Product/StrataModel/StrataModel.html
Plataforma: UNIX, Hp, SGI y Sun

1.2.13 TrueGrid

Secciona un modelo geométrico en elementos ladrillo hexaédricos y elementos de concha cuadrilateral. TrueGrid soporta mallas altamente estructuradas, de multimalla. Cada bloque está compuesto de elementos hexaédricos tridimensionales, cuadrilaterales bidimensionales, y elementos lineares o cuadráticos unidimensionales arreglados en renglones, columnas y capas. Usa la interfaz IGES estándar para la mayoría de los sistemas CAD que hacen una importación de geometría.

Referencia: http://www.xyzsa.com/home.html
Plataformas: UNIX, DOS

Al ser todos estos programas de uso general para manejo de mallas, de nuevo nos encontramos con la falta de un paquete específico para expansión de datos, orientado a problemas de contaminación atmosférica. Además, muchos de estos paquetes son de uso comercial, con precios que oscilan entre los cientos miles e dólares. Como puede verse, estos paquetes utilizan muchos algoritmos distintos. Nos centraremos en tres de ellos, que han sido probados en la solución problemas similares al que intentamos resolver, con resultados satisfactorios [MOCHIMA]

CAPÍTULO 2. Algoritmos de expansión de datos

Como hemos mencionado anteriormente, existen diversos algoritmos para realizar expansión de datos Tres de los más utilizados frecuentemente en el ámbito de análisis de partículas suspendidas o en movimiento en la atmósfera, partiendo de datos dispersos, son los los algoritmos propuestos por Cressman [CRESSMAN 1959], el llamado *Kriging* [OLIVER 1990], y un algoritmo basado en la triangulación de Voronoi [EPPSTEIN, AURENHAMMER 2000, 1991]

A continuación presentaremos los fundamentos teóricos en los que se basan cada uno de estos algoritmos

2.1 Algoritmo de Cressman

Un enfoque común para la interpolación de datos dispersos hacia una retícula regular es asumir que el valor de la retícula es un promedio ponderado de los valores de datos circundantes, es decir:

$$C_{ij} = \frac{\sum_{k=1}^{n} C_k W_k(r)}{\sum_{k=1}^{n} W_k(r)} \qquad (1)$$

donde C_k es el valor medido en la estación de medición k-ésima, $W_k(r)$ la función ponderante y r la distancia desde el punto de la retícula a la estación

Cressman propuso un procedimiento para utilización en análisis de altura de superficies en el cual usó el siguiente factor de ponderación [CRESSMAN 1959]:

$$W(r) = \frac{R^2 - r^2}{R^2 + r^2} \qquad (2)$$

donde R es la distancia a la cual el factor de ponderación tiende a cero; es decir, el "radio de influencia". Esta técnica de ponderación ayudó al procedimiento de interpolación en áreas de datos dispersos. Valores decrecientes de R fueron usados en pruebas sucesivas para analizar un espectro de escala. Los valores obtenidos de cada prueba se promediaron para producir el campo final

Endlich y Mancuso [ENDLICH 1968] combinaron ajuste polinomial y peso de distancias en su técnica de interpolación Un ajuste de mínimos cuadrados a un polinomio de primer orden se realizó usando cinco de las estaciones más cercanas, de acuerdo con:

$$W(r) = \frac{a}{(r + r^*)^2 + a} \tag{3}$$

donde a es una constante, r la distancia a la estación, y r^* un factor de distancia ($0 \leq r^* \leq r$) que indica si la observación es de arriba hacia abajo ($r^*=r$) o cruzada ($r^* = 0$) desde el punto de la retícula

2.2 *Algoritmo Kriging*

La técnica de *Kriging* es también llamada "predicción óptima" Es un método de interpolación que predice valores desconocidos de datos observados en ciertas localidades Este método usa un variograma para expresar la variación espacial, y minimiza el error de los valores predichos que se estiman por la distribución espacial de los valores predichos

2.2.1 Teoría general

En el *Kriging* ordinario, que estima el valor desconocido usando combinaciones lineares ponderadas de la muestra disponible [SETHIAN 1996]:

$$\hat{v} = \sum_{j=1}^{n} w_j * v \qquad \sum_{i=1}^{n} w_i = 1 \tag{4}$$

El error del estimado i-ésimo, ri, es la diferencia del valor estimado y el verdadero valor en la misma locación

$$r_i = \hat{v} - v_i \tag{5}$$

El error promedio de un conjunto de k estimados es:

$$m_r = \frac{1}{k}\sum_{i=1}^{k} r_i = \frac{1}{k}\sum_{i=1}^{k} \hat{v}_i - v_i \tag{6}$$

La varianza del error es:

$$\delta_R^2 = \frac{1}{k}\sum_{i=1}^{k}(r_i - m_R)^2 = \frac{1}{k}\left[\sum_{1}^{k} \hat{v}_i - v_i - \frac{1}{k}\sum_{i=1}^{k}(\hat{v}_i - v_i)\right]^2 \qquad (7)$$

Desafortunadamente, no podemos utilizar la ecuación porque no conocemos el valor real $V_1, \ldots, V_k$. Con objeto de resolver este problema, se aplica una función al azar estacionaria que contiene diversas variables aleatorias, $V(x_i)$ x_i es la locación de los datos observados para $i > 0$ e $i <= n$ (n es el número total de datos observados). El valor desconocido en la localidad x_0 que estamos tratando de estimar es $V(x_0)$ El valor estimado representado por la función aleatoria es:

$$\begin{aligned} \hat{V}(x_0) &= \sum_{i=1}^{n} w_i * V(x_i) \\ R(x_0) &= \hat{V}(x_0) - V(x_0) \end{aligned} \qquad (8)$$

La varianza del error es:

$$\delta_R^2 = \delta'^2 + \sum_{i=1}^{n}\sum_{j=1}^{n} w_i w_j \hat{C}_{ij} - 2\sum_{i=1}^{n} w_i \hat{C}_{i0} + 2\mu\left(\sum_{i=1}^{n} w_i - 1\right) \qquad (9)$$

δ'^2 es la covarianza de la variable aleatoria $V(X_0)$ con sí misma y se asume que todas las variables aleatorias tienen la misma varianza

μ es el parámetro de Lagrange [SETHIAN 1996]

Con objeto de obtener el error mínimo de varianza, calculamos las primeras derivadas parciales de la ecuación (9) para cada w y se coloca el resultado a 0 A continuación se presenta un ejemplo de diferenciación con respecto a w_1:

$$\frac{\partial(\delta_R^2)}{\partial w_1} = 2\sum_{j=1}^{n} w_j \hat{C}_{1j} - 2\hat{C}_{10} + 2\mu = 0 \qquad \sum_{j=1}^{n} w_j \hat{C}_{1j} + \mu = \hat{C}_{10} \qquad (10)$$

Todo el peso de W_i puede ser representado como:

$$\sum_{j=1}^{n} w_j \hat{C}_{ij} + \mu = \hat{C}_{i0} \text{ Para cada i, } 1 <= i <= n \ (8) \qquad (11)$$

Podemos obtener cada w_i a través de la ecuación (11) Después de obtener el valor, podemos estimar el valor localizado en x_0

2.2.2 Consideraciones particulares

Se utilizó un variograma en vez de una covarianza para calcular cada peso de la ecuación (11) El variograma es:

$$\gamma_{ij} = \delta^2 - \hat{C}_{ij} \qquad (12)$$

La estimación de la varianza minimizada es:

$$\delta_R^2 = \sum_{i=1}^{n} w_i \gamma_{i0} + \mu \qquad (13)$$

Se incluyen dos modelos de variograma:

1 esférico

$$\gamma(h) = \begin{cases} C_0 + C_1 \left(1.5\frac{h}{a} - 0.5\left(\frac{h}{a}\right)^3 \right) & \text{si } |h| \le 0 \\ C_0 + C_1 & \text{si } |h| > 0 \end{cases} \qquad (14)$$

2 exponencial

$$\gamma(h) = \begin{cases} 0 & \text{si } |h| = 0 \\ C_0 + C_1 \left(1 - \exp\left(\frac{-3|h|}{a} \right) \right) & \text{si } |h| > 0 \end{cases} \qquad (15)$$

Efecto de pepita (c_0) :

Aunque el valor del variograma para $h = 0$ es estrictamente 0, diversos factores, como el error de muestreo y la variabilidad de escala corta, pueden causar que los valores muestra separados por distancias extremadamente pequeñas sean poco disimilares. Esto causa una discontinuidad en el origen del variograma. El salto vertical del valor de cero en el origen del valor del variograma a distancias de separación extremadamente pequeñas es llamado el efecto pepita [SETHIAN 1996]

Rango (a) :

A medida que la distancia de dos pares se incrementa, el variograma de estos dos pares también se incrementa. Eventualmente, el incremento de la distancia puede causar que el variograma no se incremente. La distancia que causará que el variograma alcance a la meseta se llama rango

Techo (sill) ($C_0 + C_1$) :

El valor máximo del variograma que es la meseta de la figura siguiente:

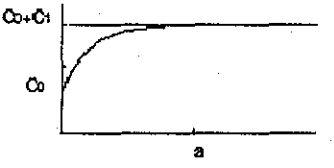


Figura 1. Un ejemplo de un modelo de variograma exponencial

Distancia (h):
La distancia entre la locación estimada y la locación observada

La ecuación (11) puede ser escrita en notación matricial como:
V * W = D

donde

V: es la matriz $(n+1) \times (n+1)$ que contiene el variograma de cada dato conocido. Los componentes de la última columna y renglón son 1 y el último componente de la matriz es 0;

W: es la matriz $(n+1)$ que contiene el peso correspondiente a cada locación. El último componente de la matriz es el Parámetro de Lagrange;

D: es la matriz $(n+1)$ que contiene el variograma de los datos conocidos y los datos estimados. El último componente de la matriz es 1

Como **V** y **D** son conocidos, podemos obtener la matriz desconocida *W*:

$$\mathbf{W = V - 1 * D}$$

donde 1 es la matriz identidad

Aplicando la ecuación (8) podemos obtener el valor estimado en una locación específica. También podemos obtener la varianza del error de la raíz cuadrada de la ecuación (13)

2.2.3 Comportamiento

Muchas propiedades de la superficie de la tierra varían de una manera aparentemente aleatoria, aunque espacialmente correlacionada Usar *Kriging* para interpolar nos permite estimar la confianza en cualquier valor interpolado en una mejor forma que otros métodos [MELKEMI 1997]

Kriging es también el método que se asocia con el acrónimo B L U E (mejor estimador lineal sin inclinación – best linear unbiased estimator) Es linear puesto que los valores estimados son combinaciones ponderadas lineales de los datos disponibles Es sin inclinación porque la media del error es 0 Es "mejor" porque tiende a minimizar la varianza de los errores La diferencia entre *Kriging* y otros métodos de estimación es su tendencia a minimizar la varianza del error

2.3 *Triangulación de Voronoi*

Existen diversas formas de triangular, dado un conjunto de puntos

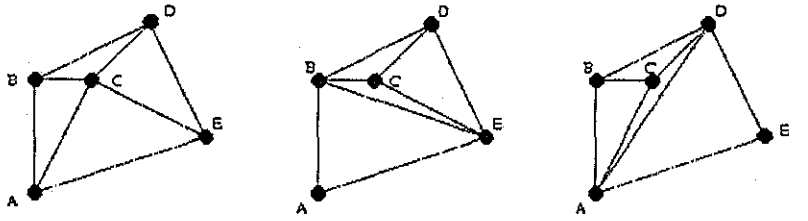


Figura 2

Algunas veces es necesaria una triangulación de los puntos con ciertas propiedades. Una de las triangulaciones más comunes y útiles es la triangulación de *Delaunay* [TERJE 1993], Nombrada por el matemático ruso, Boris Delaunay (originalmente escrito Delone, y posteriormente cambiado a la escritura más afrancesada *Delaunay*). La triangulación de Delaunay de un conjunto dado de puntos está definida por la siguiente propiedad:

AB es una arista de la triangulación e Delaunay, si y solo si existe un círculo que pasa a través de A y B de tal forma que todos los demás puntos en el conjunto de puntos, C, donde C son distintos de A y B, se encuentran fuera del círculo.

Equivalentemente, todos los triángulos en la triangulación de *Delaunay* para un conjunto de puntos tendrán círculos circunscritos vacíos. Esto es, no existen puntos dentro del interior de la circunferencia de cualquier triángulo

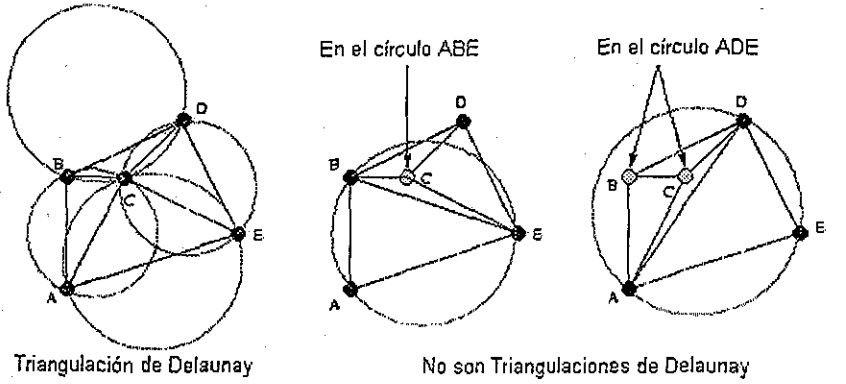


Figura 3

Podemos ver inmediatamente que la primera triangulación es *Delaunay*, puesto que todos sus círculos circunscritos están vacíos

Existe una triangulación de Delaunay para cualquier conjunto de puntos en dos dimensiones. Siempre es única en tanto que no ocurra que cuatro puntos en el conjunto de puntos sean co-circulares. Debido a que minimiza los ángulos pequeños y los círculos circunscritos, la triangulación de *Delaunay* es geométricamente conveniente, y en general, agradable a la vista.

2.3.1 El algoritmo

Para generar la triangulación de *Delaunay*, se eligió implementar un algoritmo de "divide y vencerás" presentado por Guibas y Stolfi en [EDELSBRUNNER 1994] Existen diversas formas de implementar la triangulación de Delaunay: algoritmo jerárquico modificado, de barrido radial [FORTUNE 1987], partición recursiva, dividir y vencer [GUIBAS 1985], algoritmo paso a paso, y el algoritmo incremental [LAWSON 1977]

En el uso práctico, los modelos digitales de terrenos requieren la opción de manejar conjuntos de datos muy grandes. Consecuentemente, la complejidad y la eficiencia de almacenamiento son de gran importancia. El algoritmo de dividir y vencer tiene un tiempo favorable de ejecución. El proceso de dividir también hace al algoritmo apropiado para arquitecturas paralelas. El procesado en paralelo puede ser usado también para los pasos

iniciales de los algoritmos de partición recursiva y de barrido radial Desafortunadamente, el paso final de estos algoritmos es el que más tiempo consume.

El algoritmo jerárquico modificado tiene las posibilidades de una estructura de almacenamiento jerárquica en el paso inicial Cada triángulo se particiona en tres nuevos triángulos Debido a esto, un árbol de triángulos con tres ramas puede almacenar los triángulos. Sin embargo, la estructura del árbol es difícil de mantener durante el intercambio de bordes

Para la generalización de la superficie, al elegir los puntos calificados durante la triangulación, los algoritmos incrementales no tienen rival. El algoritmo modificado jerárquico también ofrece cierto tipo de selección de puntos durante el paso inicial, pero esta selección se basa en triángulos inconvenientes (largos y delgados) Para los otros algoritmos, el filtrado de puntos, antes y después de la triangulación, debe proveerse Esto incrementa el consumo e tiempo [LEE 1991]

La selección de puntos para la generalización de una red triangular irregular, es de valor considerable en el modelaje de superficies Al usar el algoritmo incremental, los puntos redundantes pueden ser evitados en conjuntos de datos grandes. Además, el número de computadoras basadas en arquitectura paralela se está incrementando [TERJE 1993]

El algoritmo de dividir y vencer calcula la triangulación para el límite convexo del conjunto de puntos El primer paso es ordenar todos lo puntos en orden creciente según la coordenada x (cuando dos puntos tienen la misma coordenada en x, su orden se define por su coordenada y)

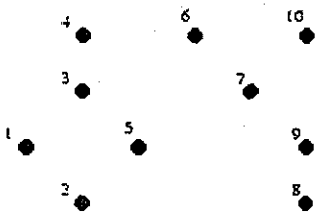


Figura 4

Una vez que se han ordenado los puntos, el conjunto ordenado se divide sucesivamente en mitades, hasta que quedan subconjuntos que contienen a lo más 3 puntos Estos subconjuntos pueden ser triangulados instantáneamente como un segmento en el caso de dos puntos y como un triángulo en el caso de tres puntos

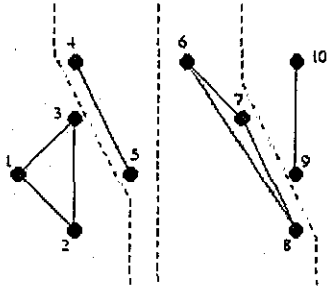


Figura 5. El conjunto ha sido "dividido" en subconjuntos de 2 ó 3 puntos y los subconjuntos han sido triangulados individualmente

Después, los subconjuntos triangulados se funden recursivamente con sus otras mitades anteriores. El producto final de cualquier fusión será una triangulación consistente en aristas LL (aristas presentes previamente en la triangulación izquierda, que tienen puntos de terminación del subconjunto izquierdo), aristas RR (aristas previamente presentes en la triangulación derecha, que tienen puntos de terminación del subconjunto derecho), y aristas LR (nuevas aristas que corren entre las triangulaciones izquierda y derechas, que tienen un punto de terminación el subconjunto izquierdo y un punto de terminación del subconjunto derecho). Para mantener las *delaunaycidad* sobre los subconjuntos de puntos fundidos, puede ser necesario borrar aristas LL y RR. Sin embargo, nunca creamos nuevas aristas LL o RR cuando fusionamos

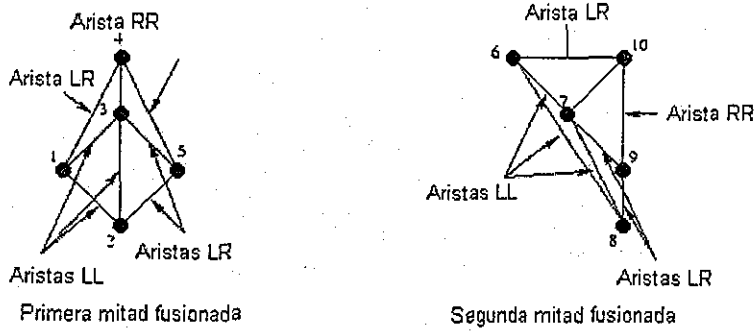


Figura 6. Las primeras fusiones se muestran aquí. Son algo simples, sin eliminación de ninguna arista LL o LR en ambos casos. La siguiente fusión será un poco más complicada y demostrativa del algoritmo.

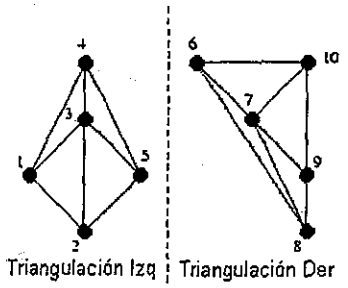


Figura 7. Hasta aquí va nuestra triangulación hasta ahora, con una fusión más restante. Todas las aristas en la triangulación izquierda son aristas LL y todas las aristas en la triangulación derecha son aristas RR

El primer paso para fundir las dos mitades es insertar la arista LR base. La arista LR base es la arista inferior (la de más abajo) que no intersecta a ninguna arista LL o RR.

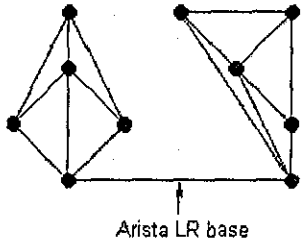


Figura 8. La arista LR base entre las triangulaciones izquierda y derecha.

Partiendo de esto, necesitamos determinar la siguiente arista LR a ser añadida justamente sobre la arista LR. Claramente, dicha arista tendrá como un punto de terminación, el punto izquierdo o bien el derecho de la arista LR base. El otro punto de terminación, entonces, vendrá ya sea del subconjunto de puntos izquierdo o derecho. Naturalmente, hacemos más estrecha nuestra decisión al seleccionar dos puntos candidatos: uno del subconjunto izquierdo y otro del subconjunto derecho.

Comenzaremos con el lado derecho. El primer candidato potencial es el punto conectado al punto derecho de la arista LR base por la arista RR que define el ángulo en el sentido de las manecillas del reloj más pequeño desde la arista LR base. Similarmente, el siguiente candidato potencial define el ángulo siguiente más pequeño en el sentido de las manecillas del reloj, de la arista LR base.

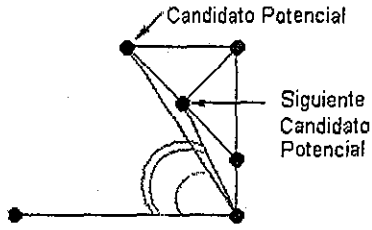


Figura 9

De acuerdo con la propiedad de la triangulación de Delaunay de un conjunto dado de puntos enunciada en la página 21, se verifica que el candidato potencial cumpla con los siguientes dos criterios:

1 El ángulo en el sentido de las manecillas del reloj de la arista LR base al candidato potencial debe ser menor a 180 grados
2 La circunferencia circunscrita definida por los dos puntos terminales de la arista LR base y el candidato potencial no debe contener al siguiente candidato potencial en su interior

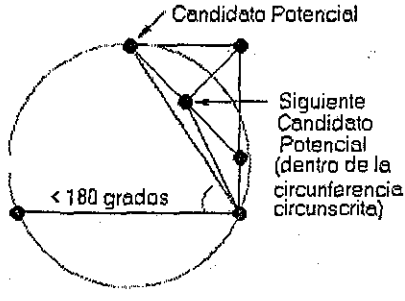


Figura 10 Aquí el candidato potencial satisface el primer criterio, pero no el segundo

Si se satisfacen ambos criterios, el candidato potencial se convierte en el candidato final para el lado derecho Si el primer criterio no aplica, entonces no se elige candidato para el lado derecho Si el primer criterio aplica, pero el segundo no, entonces la arista RR del candidato potencial al punto terminal derecho de la arista LR base es eliminado El proceso se repite entonces con el siguiente candidato potencial como el candidato potencial hasta que un candidato derecho final es elegido o se determina que no se elegirá ningún candidato

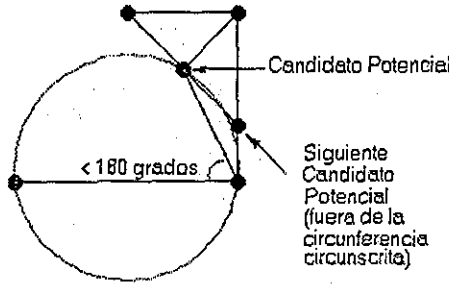


Figura 11. Después de eliminar a la "arista mala", se encontró un candidato derecho adecuado

Para la selección de candidato izquierdo, el proceso es justamente la imagen espejo de aquél para el derecho

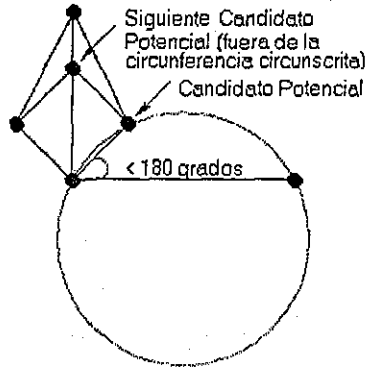


Figura 12. Del lado izquierdo, el candidato potencial inicial satisface ambos criterios.

Cuando no se obtiene candidato izquierdo ni derecho, la fusión está completa Si sólo se obtiene un candidato, automáticamente define a la arista LR a ser añadida En el caso de que ambos candidatos sean obtenidos, la arista LR apropiada se decide por medio de una prueba simple: si el candidato izquierdo no está contenido en el interior del círculo definido por los dos puntos terminales de la arista LR base y el candidato izquierdo, entonces el candidato izquierdo define a la arista LR y viceversa Por la existencia garantizada de la triangulación de *Delaunay* (aplicada aquí a sólo 4 puntos), al menos uno de os candidatos satisfará lo siguiente; por la unicidad de la triangulación de *Delaunay*, sólo un candidato satisfará esto (excepto en el caso donde lo cuatro puntos son co-circulares)

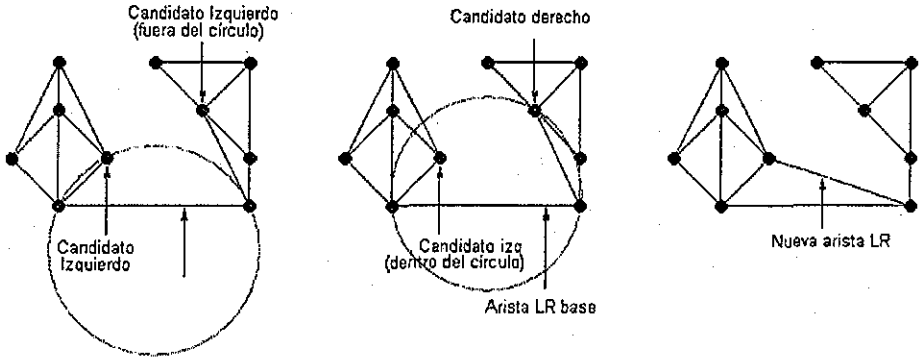


Figura 13. En este ejemplo, sólo el candidato izquierdo satisface la condición; de esta forma define la nueva arista LR.

Una vez que la nueva arista LR se añade, el proceso entero se repite con la nueva arista LR como la arista LR base

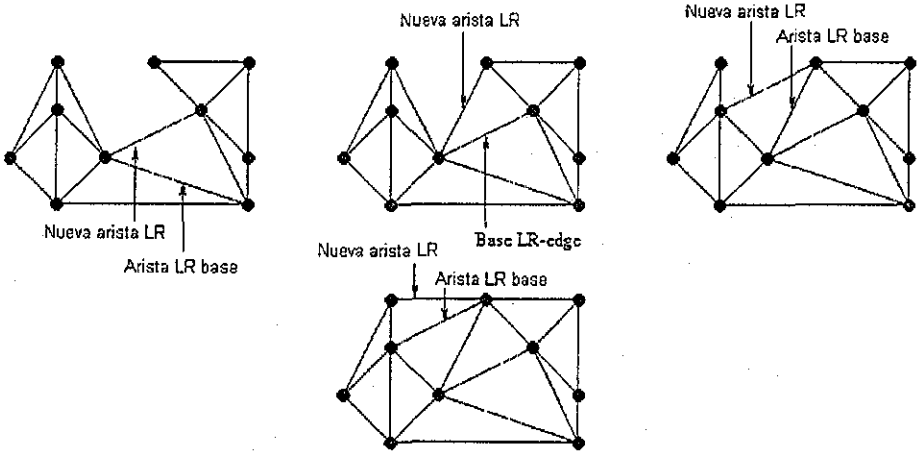


Figura 14. Se añaden aristas LR hasta que la fusión se completa (es decir, hasta que no quedan candidatos izquierdos o derechos)

Finalmente, una vez que las últimas dos mitades (aquellas que resultaron de la primera división del conjunto de puntos) se fusionan, la triangulación de *Delaunay* está completa

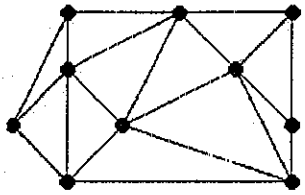


Figura 15. La triangulación de *Delaunay*

A continuación, presentaremos tres ejemplos de triangulaciones de *Delaunay*, con 10 puntos, con 100 puntos y con 1000 puntos

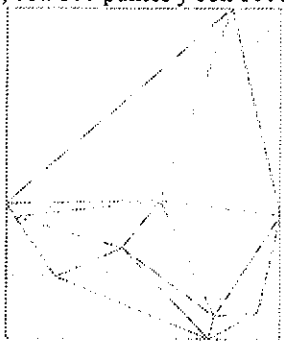


Figura 16. Triangulación de *Delaunay* con 10 puntos

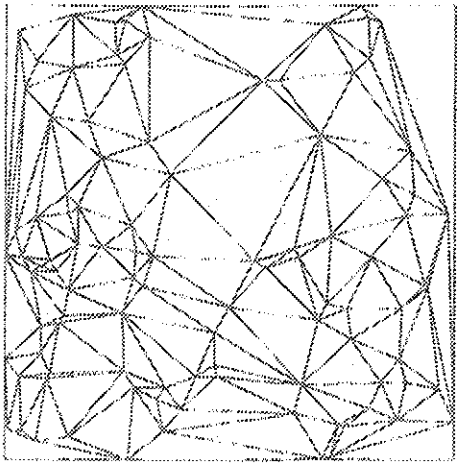


Figura 17. Triangulación de *Delaunay* con 100 puntos

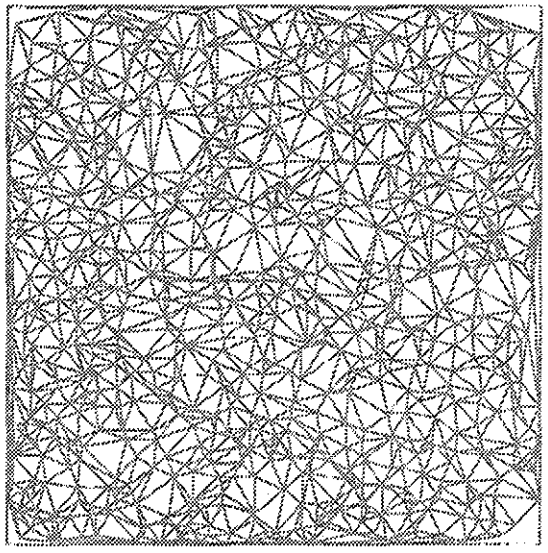


Figura 18. Triangulación de Voronoi con 1,000 puntos.

2.3.2 Diagramas de Voronoi.

Los Diagramas de Voronoi se llamados así por el matemático ruso Georges Voronoi. Los diagramas de Voronoi también son conocidos como polígonos de Thiessen y la Transformada de Eje Medial de Blum. A las celdas se les llama Regiones de Dirichlet, politopos de Thiessen o polígonos de Voronoi

El diagrama de Voronoi para un conjunto de puntos, S, en dos dimensiones (asumiendo que no hay tres puntos colineares o cuatro co-circulares) es una división del plano en polígonos. Cada punto en S está en el interior de algún polígono y cada polígono contiene exactamente un punto en su interior. Cada polígono corta la región que está más cerca de su punto contenido que cualquier otro punto en S

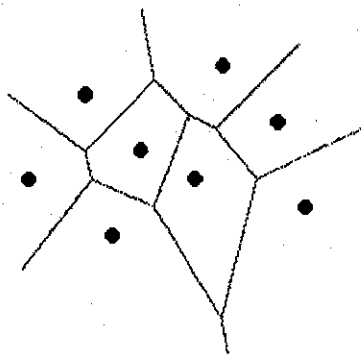


Figura 19

El diagrama de Voronoi es el dual lineal de la triangulación de Delaunay Esto quiere decir que podemos ir de el diagrama de Voronoi a la triangulación de Delaunay dibujando las aristas que son perpendiculares a los límites de la región y viceversa

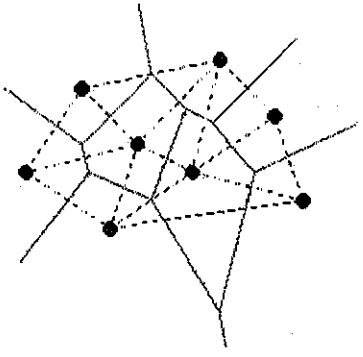


Figura 20

Ahora que tenemos la triangulación única de Voronoi, nos enfrentamos al problema de saber si un punto se encuentra contenido en un triángulo, para que de esta manera, utilizando ecuaciones que describen a un plano en el espacio, podamos obtener los valores en los puntos de la retícula regular que se encuentran sobrepuestos a los triángulos generados

2.3.3 Prueba de la inclusión de un punto en un triángulo

Dado un triángulo (v1,v2,v3) y un punto *p*, la prueba de la inclusión de *p* en el triángulo es la siguiente:

Si recorremos los puntos v1, v2, v3 y el punto está dentro, siempre veremos al punto del mismo lado del segmento que estamos visitando Si v1,v2,v3 están arreglados en un sentido en contra de las manecillas del reloj, los puntos dentro de él están siempre a la izquierda de los segmentos. Si el punto está fuera, al menos uno de los segmentos del punto estará a la derecha (ver figura) Si los vértices están arreglados en el sentido de las manecillas del reloj, el razonamiento es idéntico, excepto que un punto que está dentro del triángulo estará siempre a la derecha del segmento que estamos visitando

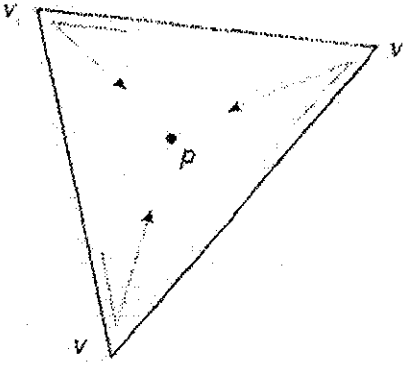


Figura 21

De esta forma, para determinar si el punto *p* está dentro del triángulo (v1,v2,v2), necesitamos obtener las direcciones de rotación a lo largo de las tripletas (v1,v2,p), (v2,v3,p) y (v3,v1,p) El punto está dentro si y sólo si las tres direcciones son iguales

Ahora que tenemos una forma eficiente [BERNARDINI 1997] de saber si un punto está dentro de un triángulo, obtenemos el valor de z utilizando ecuaciones conocidas de geometría analítica que describen al plano

A continuación presentamos la aplicación los tres métodos de expansión de datos expuestos anteriormente, a datos reales de contaminación atmosférica

CAPÍTULO 3. Implementación de los algoritmos seleccionados

Una vez habiendo cubierto el objetivo 1, contamos con 3 algoritmos de expansión de datos seleccionados para ser implementados. Como los objetivos 2 y 3 nos indica,n crearemos un paquete que nos permita obtener datos distribuidos en una malla regular preseleccionada, a partir de datos de entrada dispersos, y además, que nos permita visualizarlos de manera interactiva. Los detalles se presentan a continuación

La implementación del programa se realizó en lenguaje C++, utilizando la librería OpenGL para la graficación en pantalla, y la librería μ-UI (Micro User Interface) para la interfaz del usuario. μ-UI aprovecha las características de dibujo bidimensional de OpenGL. De esta forma, la aplicación puede ser fácilmente portada a otros sistemas operativos que tengan compilador C++

La ventana principal del programa permite elegir el Algoritmo de Expansión de Datos deseado; algunos parámetros particulares de cada uno de los algoritmos implementados, el tamaño y espaciado de la malla que se desea obtener, y el archivo de datos de entrada

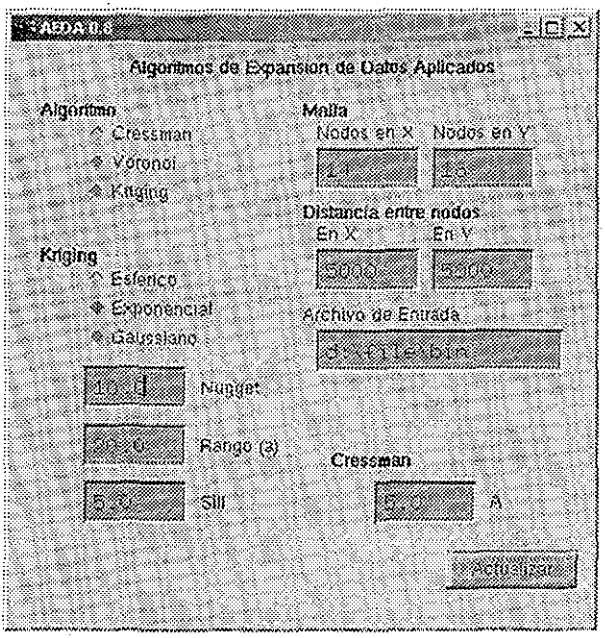


Figura 22

Al oprimir el botón Actualizar, aparecerá una ventana con la renderización de la malla generada Para demostración, cuando el archivo de entrada no es válido, o no existe, se utilizan las funciones originales por omisión:

$$x^2 + y^2 \tag{16}$$

o bien,

$$x^2 - y^2 \tag{17}$$

de las cuales se toman 30 muestras en puntos dispuestos al azar. y posteriormente, como se hace con los datos de entrada normales, se reconstruye la malla usando el algoritmo de expansión de datos elegido

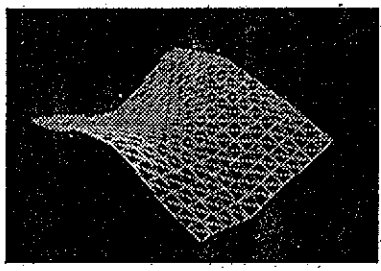


Figura 23

En esta imagen se muestra la malla generada con el algoritmo de Cressman, con parámetro A = 5 0 En color blanco, pueden verse los puntos en los que se tomó la muestra de altura, para después reconstruir la superficie

Mientras está mostrándose la ventana de graficación, pueden utilizarse las siguientes teclas:

W: Cambia de modo de malla de alambre, a modo de superficie sólida

F: Cambia la función de muestra ($x^2 + y^2$ por $x^2 - y^2$ y viceversa)

P: Muestra u oculta los puntos de muestra (puntos blanco)

8: Rota el objeto en sentido de las manecillas del reloj sobre el eje X

6: Rota el objeto en el sentido de las manecillas del reloj sobre el eje Z

2: Rota el objeto en sentido contrario de las manecillas del reloj sobre el eje X

4: Rota el objeto en el sentido contrario de las manecillas del reloj sobre el eje Z

S: Exporta los datos de la malla a un archivo llamado mesh out

+: Incrementa el factor de escala en el eje Z (altura)

–: Disminuye el factor de escala en el eje Z

A: Incrementa el factor A de Cressman en 1 (hasta 15)[4]

Z: Decrementa el factor A de Cressman en 1 (hasta 1)

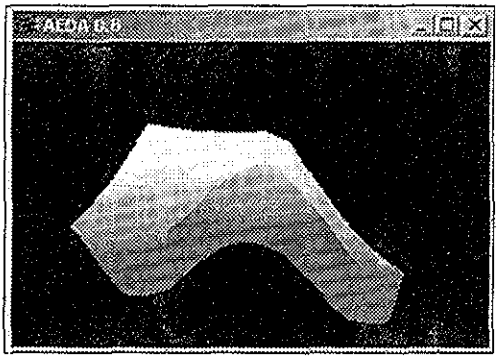


Figura 24. Aquí se muestra una imagen de la función de muestra reconstruida con los puntos de muestra ocultos en modo de superficie sólida.

Las convenciones de ejes han sido tomadas de la siguiente forma:

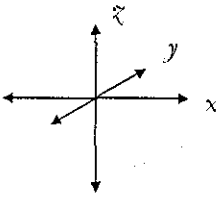


Figura 25

[4] Aunque se haya seleccionado otra algoritmo de expansión de datos, pasará automáticamente a Cressman

Los movimientos de rotación dados por el teclado, también pueden ser logrados con el ratón, haciendo clic sobre algún punto de la ventana, y arrastrando el apuntador

Los archivos de entrada deben estar en formato texto (ASCII), indicando la posición de la estación de medición en su coordenada X en primer lugar, y después, separando por espacios, la coordenada en Y Por último, el valor de la medición Los primeros dos valores serán enteros, y el último puede ser de punto flotante

Ejemplo de archivo:

```
   28660      41020      0.101
   24540      43990      0.171
   45800      50030      0.136
   28521      49233          0
   42000      49090      0.251
   37475      37620      0.292
   28553      26673      0.013
   42150      27800      0.091
   41200      37370      0.143
   33260      31870          0
31711.8       62477          0
   23440      54620          0
39516.6       62932          0
   28950      31340          0
```

Figura 26. Ejemplo de archivo de entrada

El número de mediciones (renglones) no debe ser mayor a 65535

El formato de salida mesh.out está dado en el mismo formato

```
15000      35000      0.161997
15000      40000      0.166417
15000      45000      0.154865
15000      50000      0.129265
15000      55000      0.084798
15000      60000      0.034853
15000      65000     -0.018312
15000      70000     -0.053776
15000      75000     -0.074067
20000      0         -0.202291
20000      5000      -0.200391
20000      10000     -0.190037
20000      15000     -0.140032
20000      20000     -0.058771
20000      25000      0.011562
20000      30000      0.060174
```

Figura 27. Fragmento de un archivo de salida mesh.out

CAPÍTULO 4. Comparación de los algoritmos implementados

Para determinar qué tan buenos son los Algoritmos de Expansión de Datos expuestos en capítulos anteriores, realizaremos dos tipos de pruebas Para realizar la primera de ellas, renderizaremos funciones matemáticas conocidas, de las cuales podemos obtener su valor en cualquier parte y posteriormente realizaremos la reconstrucción de éstas con cada uno de los algoritmos para finalmente poder medir cuantitativamente las diferencias con respecto a los valores originales conocidos en los puntos reconstruidos

A simple vista puede tratar de decirse cuál es el mejor Algoritmo de Expansión de Datos examinando la exactitud con la que se reconstruye la función original, de la cual conocemos ya su forma:

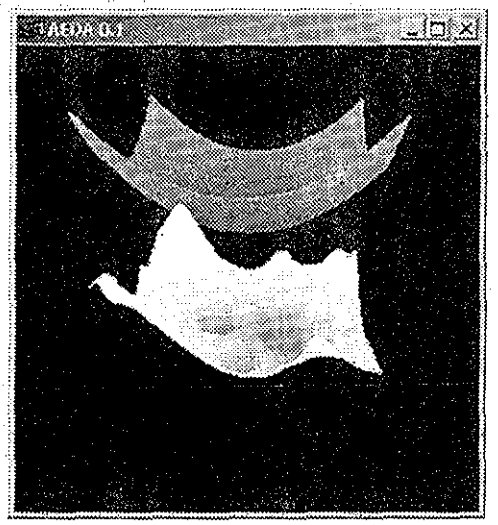


Figura 28 Cressman

En el caso de Cressman, pueden verse los radios de influencia de los puntos de medición Esto produce una superficie irregular, pues este algoritmo trabaja de una manera local

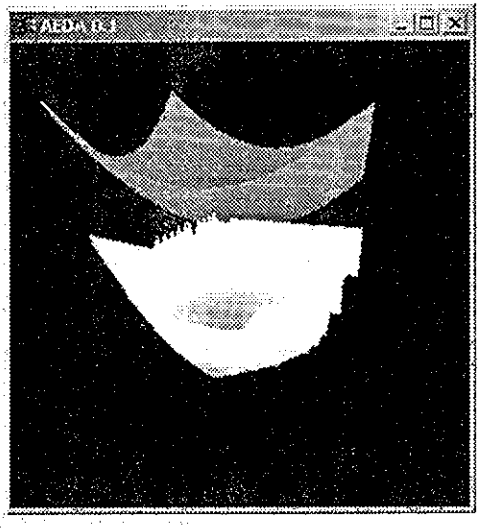


Figura 29 Voronoi

Voronoi produce una superficie más uniforme, aunque en las orillas se presentan espacios, que al no quedar dentro de la triangulación, quedan indefinidos En la siguiente imagen puede verse la triangulación sobrepuesta en color verde

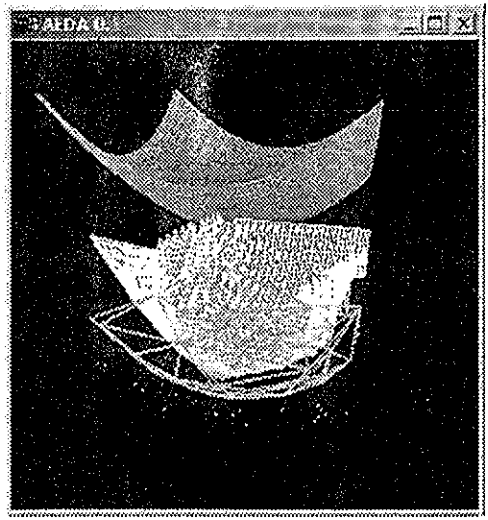


Figura 30 Voronoi en malla de alambre con triangulación sobrepuesta

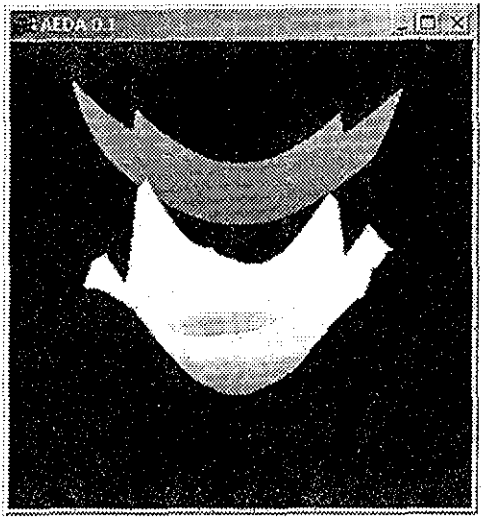


Figura 31. *Kriging*

Por ultimo, puede verse que *Kriging* tiende a ajustarse más a la función original

Ahora veamos cuantitativamente con datos reales de mediciones de contaminación atmosférica, cuál es el mejor AED Los contaminantes medidos son:

NO
NO_2
Ozono
$RCHO_1$
$RCHO_2$
$RCHO_3$
RH_1
RH_2
RH_3

Para obtener una medida de la desviación que presentaban los datos de la malla, se compararon los datos originales que quedaban dentro de la celda definida por 4 nodos de la malla, con el valor del nodo más cercano de la malla reconstruida

A continuación se presentan las tablas obtenidas de este procedimiento, para cada uno de los agentes contaminantes Los Algoritmos de Expansión de Datos están indicados de la siguiente manera:

Cr – Cressman
Vo – Voronoi
Kr – *Kriging*

X y Y indican la ubicación (en metros) del punto medido
Orig indica el valor que se tenía originalmente para ese punto
Dif. cr, dif. kr y dif. vo indican la diferencia entre el valor original y el valor reconstruido con cada uno de los algoritmos (cr, vo y kr)
En el ultimo renglón se indican las sumas absolutas de las diferencias

NO

X	Y	orig	cr	dif. cr	kr	dif. kr	vo	dif. vo
25000	45000	0.171	0.142107	0.028893	0.140619	0.030381	0.143938	0.027062
25000	55000	0	0.015224	0.015224	-0.001427	0.001427	0	0
30000	25000	0.013	0.030708	0.017708	0.011916	0.001084	0	0.013
30000	30000	0	0.041853	0.041853	0.01233	0.01233	0.019735	0.019735
30000	40000	0.101	0.107589	0.006589	0.116832	0.015832	0.116345	0.015345
30000	50000	0	0.029953	0.029953	0.024242	0.024242	0.024039	0.024039
30000	60000	0	0.02213	0.02213	0.002371	0.002371	0	0
35000	30000	0	0.050877	0.050877	0.029264	0.029264	0.026744	0.026744
35000	40000	0.292	0.175527	0.116473	0.2143	0.0777	0.223375	0.068625
40000	30000	0.091	0.102978	0.011978	0.091072	7.2E-05	0.108386	0.017386
40000	35000	0.143	0.163704	0.020704	0.157535	0.014535	0.187391	0.044391
40000	50000	0.251	0.198104	0.052896	0.205752	0.045248	0.20978	0.04122
40000	60000	0	0.033404	0.033404	0.037241	0.037241	0.053122	0.053122
45000	50000	0.136	0.153719	0.017719	0.157546	0.021546	0.156423	0.020423
				0.466401		0.313273		0.371092

NO_2

X	Y	orig	cr	dif. cr	kr	dif. kr	vo	dif. vo
25000	45000	0.051	0.050935	6.5E-05	0.05035	0.00065	0.050151	0.000849
25000	55000	0.048	0.047444	0.000556	0.045676	0.002324	0.045352	0.002648
30000	25000	0.041	0.044646	0.003646	0.040251	0.000749	0	0.041
30000	30000	0	0.050873	0.050873	0.048436	0.048436	0.04715	0.04715
30000	40000	0.054	0.055005	0.001005	0.05918	0.00518	0.05856	0.00456
30000	50000	0.045	0.046607	0.001607	0.045945	0.000945	0.044727	0.000273
30000	60000	0.029	0.035269	0.006269	0.034318	0.005318	0.033654	0.004654
35000	30000	0.056	0.055704	0.000296	0.052554	0.003446	0.049168	0.006832
35000	40000	0.092	0.06895	0.02305	0.07663	0.01537	0.074912	0.017088
40000	30000	0.039	0.050084	0.011084	0.04979	0.01079	0.050086	0.011086
40000	35000	0.067	0.068369	0.001369	0.068737	0.001737	0.069479	0.002479
40000	50000	0.052	0.051053	0.000947	0.050802	0.001198	0.049779	0.002221
40000	60000	0.026	0.032759	0.006759	0.030769	0.004769	0.031519	0.005519
45000	50000	0.045	0.046895	0.001895	0.046169	0.001169	0.046119	0.001119
				0.109421		0.102081		0.147478

Ozono

X	Y	orig	cr	dif. cr	kr	dif. kr	vo	dif. vo
20000	5000	0.017	0.017294	0.000294	0.017323	0.000323	0.017754	0.000754
25000	45000	0.022	0.021259	0.000741	0.021701	0.000299	0	0.022
30000	20000	0.018	0.018802	0.000802	0.019478	0.001478	0.01998	0.00198
30000	25000	0.027	0.023037	0.003963	0.023724	0.003276	0.023694	0.003306
30000	30000	0.018	0.019931	0.001931	0.02034	0.00234	0.021409	0.003409
30000	40000	0.016	0.018296	0.002296	0.016744	0.000744	0.016631	0.000631
30000	45000	0.024	0.021954	0.002046	0.022717	0.001283	0.023868	0.000132
30000	50000	0.017	0.018539	0.001539	0.01803	0.00103	0	0.017
35000	30000	0.02	0.018949	0.001051	0.018476	0.001524	0.017743	0.002257
35000	40000	0.018	0.018657	0.000657	0.018044	4.4E-05	0.017935	6.5E-05
40000	30000	0.017	0.019953	0.002953	0.019742	0.002742	0.021462	0.004462
40000	35000	0.021	0.020466	0.000534	0.021416	0.000416	0.022048	0.001048
40000	50000	0.023	0.022387	0.000613	0.022202	0.000798	0	0.023
45000	30000	0.025	0.021232	0.003768	0.022354	0.002646	0.021462	0.003538
45000	50000	0.025	0.024127	0.000873	0.024589	0.000411	0	0.025
50000	20000	0.02	0.019837	0.000163	0.017725	0.002275	0.017544	0.002456
60000	50000	0.021	0.021058	5.8E-05	0.021509	0.000509	0	0.021
				0.024482		0.022133		0.132038

$RCHO_1$

X	Y	orig	cr	dif. cr	kr	dif. kr	vo	dif. vo
25000	45000	0.037	0.03795	0.00095	0.021701	0.015299	0	0.037
25000	55000	0.008	0.016067	0.008067	0.018601	0.010601	0	0.008
30000	25000	0.009	0.013224	0.004224	0.023724	0.014724	0.023694	0.014694
30000	40000	0.0258	0.02992	0.00412	0.016744	0.009056	0.016631	0.009169
30000	50000	0.075	0.061649	0.013351	0.01803	0.05697	0	0.075
35000	30000	0.0093	0.018668	0.009368	0.018476	0.009176	0.017743	0.008443
35000	40000	0.064	0.044202	0.019798	0.018044	0.045956	0.017935	0.046065
35000	60000	0.0048	0.018454	0.013654	0.020347	0.015547	0	0.0048
40000	30000	0.0217	0.026498	0.004798	0.019742	0.001958	0.021462	0.000238
40000	40000	0.035	0.043052	0.008052	0.020609	0.014391	0.020602	0.014398
40000	50000	0.0505	0.04502	0.00548	0.022202	0.028298	0	0.0505
40000	60000	0.0043	0.029552	0.025252	0.021551	0.017251	0	0.0043
45000	50000	0.0302	0.034373	0.004173	0.024589	0.005611	0	0.0302
				0.121287		0.244838		0.302807

$RHCO_2$

X	Y	orig	cr	dif. cr	kr	dif. kr	vo	dif. vo
25000	45000	0.185	0.161585	0.023415	0.159041	0.025959	0.161741	0.023259
25000	55000	0.04	0.052662	0.012662	0.036777	0.003223	0.037798	0.002202
30000	25000	0.045	0.062947	0.017947	0.0435	0.0015	0	0.045
30000	40000	0.1292	0.136183	0.006983	0.146601	0.017401	0.145784	0.016584
30000	50000	0.0375	0.064419	0.026919	0.059155	0.021655	0.057307	0.019807
30000	60000	0.0242	0.049877	0.025677	0.031138	0.006938	0.02807	0.00387
35000	30000	0.0467	0.089154	0.042454	0.068158	0.021458	0.06327	0.01657
35000	40000	0.32	0.205694	0.114306	0.242156	0.077844	0.248584	0.071416
40000	30000	0.1083	0.128158	0.019858	0.117318	0.009018	0.132042	0.023742
40000	40000	0.175	0.208197	0.033197	0.241067	0.066067	0.245236	0.070236
40000	50000	0.2525	0.212938	0.039562	0.216553	0.035947	0.216301	0.036199
40000	60000	0.0217	0.118748	0.097048	0.09157	0.06987	0.02807	0.00637
45000	50000	0.1508	0.168852	0.018052	0.170245	0.019445	0.166903	0.016103
				0.47808		0.376325		0.351358

$RCHO_3$

X	Y	orig	cr	dif. cr	kr	dif. kr	vo	dif. vo
25000	45000	0.74	0.646359	0.093641	0.636177	0.103823	0.646963	0.093037
25000	55000	0.16	0.210658	0.050658	0.147129	0.012871	0.151215	0.008785
30000	25000	0.18	0.251804	0.071804	0.173994	0.006006	0	0.18
30000	40000	0.517	0.544874	0.027874	0.586592	0.069592	0.583313	0.066313
30000	50000	0.15	0.257689	0.107689	0.236635	0.086635	0.22924	0.07924
30000	60000	0.097	0.199645	0.102645	0.124692	0.027692	0.112429	0.015429
35000	30000	0.187	0.356721	0.169721	0.272716	0.085716	0.253143	0.066143
40000	30000	0.433	0.512552	0.079552	0.469193	0.036193	0.528064	0.095064
40000	35000	0.7	0.777385	0.077385	0.754231	0.054231	0.856149	0.156149
40000	50000	1.01	0.851733	0.158267	0.866231	0.143769	0.865219	0.144781
40000	60000	0.087	0.475027	0.388027	0.366334	0.279334	0.112429	0.025429
45000	50000	0.603	0.675262	0.072262	0.680822	0.077822	0.66745	0.06445
				1.399525		0.983684		0.99482

RH_1

X	Y	orig	cr	dif. cr	kr	dif. kr	vo	dif. vo
25000	45000	0.037	0.03795	0.00095	0.041302	0.004302	0.040772	0.003772
25000	55000	0.008	0.016067	0.008067	0.016517	0.008517	0.014836	0.006836
30000	25000	0.009	0.013224	0.004224	0.008506	0.000494	0	0.009
30000	40000	0.0258	0.02992	0.00412	0.03066	0.00486	0.029121	0.003321
30000	50000	0.075	0.061649	0.013351	0.067591	0.007409	0.068516	0.006484
30000	60000	0.0048	0.014343	0.009543	0.012813	0.008013	0.011209	0.006409
35000	30000	0.0093	0.018668	0.009368	0.013659	0.004359	0.012642	0.003342
35000	40000	0.064	0.044202	0.019798	0.053689	0.010311	0.049703	0.014297
40000	30000	0.0217	0.026498	0.004798	0.023321	0.001621	0.026429	0.004729
40000	35000	0.035	0.039736	0.004736	0.037233	0.002233	0.04282	0.00782
40000	50000	0.0505	0.04502	0.00548	0.051176	0.000676	0.049825	0.000675
40000	60000	0.0043	0.029552	0.025252	0.022593	0.018293	0.011209	0.006909
45000	50000	0.0302	0.034373	0.004173	0.0341	0.0039	0.033413	0.003213
				0.11386		0.074988		0.076807

RH_2

X	Y	orig	cr	dif. cr	kr	dif. kr	vo	dif. vo
25000	45000	0.185	0.161616	0.023384	0.159102	0.025898	0.161803	0.023197
25000	55000	0.04	0.0527	0.0127	0.036831	0.003169	0.037827	0.002173
30000	25000	0.045	0.062969	0.017969	0.043485	0.001515	0	0.045
30000	40000	0.129	0.136086	0.007086	0.146493	0.017493	0.145641	0.016641
30000	50000	0.038	0.064784	0.026784	0.059595	0.021595	0.057766	0.019766
30000	60000	0.024	0.04979	0.02579	0.031065	0.007065	0.027963	0.003963
35000	30000	0.047	0.08931	0.04231	0.068292	0.021292	0.063363	0.016363
35000	40000	0.32	0.205731	0.114269	0.242216	0.077784	0.248571	0.071429
40000	30000	0.108	0.128047	0.020047	0.117192	0.009192	0.131885	0.023885
40000	35000	0.175	0.19436	0.01936	0.188535	0.013535	0.213973	0.038973
40000	50000	0.253	0.213306	0.039694	0.216993	0.036007	0.216755	0.036245
40000	60000	0.022	0.118846	0.096846	0.091645	0.069645	0.027963	0.005963
45000	50000	0.151	0.169097	0.018097	0.170504	0.019504	0.16715	0.01615
				0.464336		0.323694		0.319748

RH_3

X	Y	orig	cr	dif. cr	kr	dif. kr	vo	dif. vo
25000	45000	0.74	0.646359	0.093641	0.636177	0.103823	0.646963	0.093037
25000	55000	0.16	0.210658	0.050658	0.147129	0.012871	0.151215	0.008785
30000	25000	0.18	0.251804	0.071804	0.173994	0.006006	0	0.18
30000	40000	0.517	0.544874	0.027874	0.586592	0.069592	0.583313	0.066313
30000	50000	0.15	0.257689	0.107689	0.236635	0.086635	0.22924	0.07924
30000	60000	0.097	0.199645	0.102645	0.124692	0.027692	0.112429	0.015429
35000	30000	0.187	0.356721	0.169721	0.272716	0.085716	0.253143	0.066143
35000	40000	1.28	0.82281	0.45719	0.968706	0.311294	0.994405	0.285595
40000	30000	0.433	0.512552	0.079552	0.469193	0.036193	0.528064	0.095064
40000	35000	0.7	0.777385	0.077385	0.754231	0.054231	0.856149	0.156149
40000	50000	1.01	0.851733	0.158267	0.866231	0.143769	0.865219	0.144781
40000	60000	0.087	0.475027	0.388027	0.366334	0.279334	0.112429	0.025429
45000	50000	0.603	0.675262	0.072262	0.680822	0.077822	0.66745	0.06445
				1 856715		1 294978		1 280415

Resumimos todas las sumas de las diferencias:

	suma de las diferencias		
	cressman	*Kriging*	voronoi
NO	0.466401	0.313273	0.371092
NO_2	0.109421	0.102081	0.147478
Ozono	0.024482	0.022138	0.132038
$RHCO_1$	0.121287	0.244838	0.302807
$RHCO_2$	0.47808	0.376325	0.351358
$RHCO_3$	1.399525	0.983684	0.99482
RH_1	0.11386	0.074988	0.076807
RH_2	0.464336	0.323694	0.319748
RH_3	1 856715	1 294978	1 280415

Podemos ver que en la mayoría de los casos, *Kriging* es quien tiene los números menores de diferencias, seguido de Voronoi. En algunos casos, Voronoi presenta una mejor aproximación, como es el caso del RH_2 y RH_3

Para tener una idea más concreta, se presentan los valores normalizados, y así obtenemos un porcentaje que nos puede decir en términos numéricos, la aptitud de un Algoritmo de Expansión de Datos

	Cressman	*Kriging*	Voronoi
NO	76.68%	84.34%	81.45%
NO_2	94.53%	94.90%	92.63%
Ozono	98.78%	98.89%	93.40%
$RHCO_1$	93.94%	87.76%	84.86%
$RHCO_2$	76.10%	81.18%	82.43%
$RHCO_3$	30.02%	50.82%	50.26%
RH_1	94.31%	96.25%	96.16%
RH_2	76.78%	83.82%	84.01%
RH_3	7.16%	35.25%	35.98%
	72 03%	79 24%	77 91%

En este caso la presentación intuitiva de que *Kriging* era el AED que mejor se adaptaba se ve confirmada por la evaluación numérica.

CAPÍTULO 5. Conclusiones

Hemos estudiado tres algoritmos de expansión de datos, aplicados específicamente a datos de contaminación atmosférica de la Ciudad de México. Estos algoritmos son: el algoritmo de expansión de datos propuesto por Cressman, el algoritmo de expansión de datos basado en la triangulación de Voronoi y el algoritmo de expansión de datos llamado Kriging. Por brevedad, nos referiremos a ellos como Cressman, Voronoi y Kriging.

Kriging es un interpolador basado en una estructura estimada de covarianza. Es importante no ignorar el efecto de la incerteza en la estructura de la covarianza en predicciones subsecuentes. Kriging es el mejor estimador lineal, según han demostrado las pruebas realizadas con el algoritmo de Kriging, sin embargo, el ajuste de sus parámetros es muy importante para un buen desempeño. En el desarrollo de los experimentos, se determinó que los parámetros más adecuados para la aplicación en particular del estudio de los datos de contaminación atmosférica de la Ciudad de México son los siguientes:

Kriging esférico, con *nugget* de 10 0, *Rango* (a) de 20 0 y *Sill* de 5 0.

Para encontrar estos parámetros se efectuó múltiples veces el experimento con un evaluador de diferencia entre el valor esperado y el obtenido. Los parámetros fueron variados hasta encontrar un mejor desempeño.

Siguiendo un procedimiento similar, se determinó el parámetro a adecuado para Cressman, que es de 5 0 unidades.

Kriging es un interpolador que genera una superficie de interpolación de los datos. El *nugget* juega un papel importante en el tipo de superficie que será generada. Si uno elige un modelo de variograma sin *nugget*, entonces la superficie interpolada será suave. Esto es, para localidades cercanas a la estación de medición, los valores del dato estimado tenderán hacia el valor de la variable de la estación de medición. Con un *nugget* diferente de cero, esto no sucede: kriging funciona aún como interpolador, pero hace saltos discontinuos a los valores de los datos en las localidades donde hay datos. Entre más *nugget* haya en el *sill* total, el mapa tenderá más hacia una media de la superficie de los datos, como un plano sin variaciones (plano) con altura igual a la media.

Debido a que Kriging es polinómico, su tiempo de respuesta es más grande con respecto a aquél de los demás algoritmos. Si se requiere procesar datos en tiempo real, y no se cuenta con un equipo de cómputo poderoso, puede optarse por el algoritmo de interpolación de datos basado en la triangulación de Voronoi. Además, otra ventaja que ofrece el algoritmo de interpolación de datos basado en la triangulación de Voronoi es que no requiere parámetros de ajuste.

Voronoi tiene una respuesta a alteraciones locales muy pequeña. Tiende a suavizar la forma de la superficie. Kriging y Cressman, por el contrario, pueden llegar a responder

activamente a pequeñas alteraciones locales. En el caso de la difusión de gases en la atmósfera, la presencia de alteraciones locales disminuye conforme se incrementa la altura. En el caso de que nos interesara obtener fuentes locales de contaminación atmosférica, como podría ser la identificación de una fábrica con fuertes emisiones de contaminantes, puede ser útil la información provista por estos algoritmos. De aquí que la recomendación sobre qué algoritmo utilizar esté influenciada por la aplicación.

Existe un solo diagrama de Voronoi para un conjunto dado de puntos. Los polígonos de Voronoi son mutuamente exclusivos y forman regiones con polígonos convexos. Particionar un espacio en regiones significativas es a menudo un prerrequisito para realizar análisis cuantitativos. La ventaja de las celdas de Voronoi, es que cada una contiene una sola medición.

Un punto importante aquí es que los puntos aislados requieren más atención que los datos agrupados. Pueden existir áreas que no han sido muestreadas lo suficiente. El uso de un mismo parámetro para describir la complejidad de una red de muestreo es limitante. El uso de variogramas en conjunción con los diagramas de Voronoi podría proveer información interesante acerca del impacto de cada muestra durante un problema de estimación. Pueden obtenerse histogramas de los polígonos que ayuden a describir cuantitativamente la homogeneidad de la red de muestreo: al conocer el tamaño promedio de la celda, que representa el tamaño de la celda que se tendría en caso de una red homogénea, se puede evaluar el impacto de los valores extremos en la red de muestreo. Puede obtenerse un tamaño ideal de la celda, que es el total de la superficie dividido por el número de muestras, de tal manera que puedan determinarse aquellos polígonos que tengan áreas extremadamente grandes y polígonos pequeños resaltando la presencia de datos agrupados.

Como resultado de las investigaciones realizadas durante la elaboración de esta tesis, se obtuvo un software especializado para la expansión de datos mediante los algoritmos Cressman, Kriging y Voronoi, permitiendo visualizar interactivamente desde diversos ángulos las mallas reconstruidas a partir de los puntos de medición proporcionados. Pueden cambiarse los parámetros de los algoritmos, la densidad de la malla, y pueden guardarse los datos en un formato de texto para su procesamiento posterior. La última versión de este software está disponible en:
http://fesc.cuautitlan2.unam.mx/~fcalvo/aeda

Por último, recomendamos al algoritmo de Kriging como el más adecuado para aplicaciones de expansión de datos de contaminación atmosférica en la Ciudad de México.

Como parte de este trabajo, se incluyen dos apéndices con información útil para el usuario que desee adentrarse en la programación con las librerías gráficas OpenGL y GLUT. Para la graficación de superficies se recomienda el método de definición de una malla mediante puntos de triángulos usando el modo `GL_TRIANGLE_STRIP` y la primitiva `glvertex`. Para la creación de interfaces portables utilizando OpenGL con la librería GLUI, se recomienda utilizar la herramienta de creación de interfaces µU-I (micro user interface).

APÉNDICES.

En los algoritmos arriba desarrollados se trata de construir una superficie en espacio de tres dimensiones. De las diversas opciones que aparecen a la hora de elegir una forma de visualizar resultados en tres dimensiones, existe una que cuenta con la mayoría de las ventajas

Se eligió OpenGL para realizar la implementación de la interfaz gráfica debido a que es una librería gráfica gratuita, eficiente, disponible para todas las plataformas que tengan un compilador de lenguaje C, con un rango amplio de usuarios

A continuación presentamos una introducción a OpenGL que nos permitirá conocer la forma en que es utilizado, así como un manual completo de referencia del GLUT Es importante destacar que para la comunidad de habla hispana en Internet no existe un manual completo de OpenGL y GLUT.

APÉNDICE A. OpenGL

OpenGL es una especificación de librerías gráficas a bajo nivel OpenGL da al programador un pequeño conjunto de objetos geométricos primitivos: puntos, líneas, polígonos, imágenes y mapas de bits OpenGL proporciona un conjunto de comandos que permiten la especificación de objetos geométricos en dos o tres dimensiones, usando las primitivas provistos, junto con comandos que controlan cómo se renderizan[1] estos objetos

La API[2] de OpenGL fue diseñada para ser utilizada con lenguajes de programación C y C++, pero también hay versiones disponibles para otros lenguajes de programación como Java, Tcl, Ada y FORTRAN

La especificación de OpenGL es mantenida y actualizada por Silicon Graphics A pesar de que estas librerías surgen como una aplicación de uso comercial, existen implementaciones gratuitas como Mesa[3]

La especificación de OpenGL es independiente del sistema operativo y del sistema que administra la parte gráfica Utiliza, sin embargo, a este último para el manejo de las ventanas, manejo de eventos, operaciones de mapeo[4] de color, etc

---

[1] Renderizar: Aplicar una serie de cálculos y procedimientos a un conjunto de datos origen para obtener un resultado Generalmente se utiliza en cuestiones gráficas. para denotar que se obtiene una imagen visual a partir de una descripción de los objetos que la componen y su interrelación

[2] API: Application's Programmer Interface: Interfaz de aplicación para el programador La parte exterior de un paquete de librerías que permite que el programador utilice dichas librerías sin necesidad de conocer su funcionamiento interno

[3] http://www.ssec.wisc.edu/~brianp/Mesa.html

Los programas que utilizan OpenGL usualmente establecen un bucle En cuanto se inicia el programa, cierto código de inicialización debe ser ejecutado Posteriormente, el programa cae en un bucle infinito, aceptando y manejando los eventos Los eventos incluyen operaciones como opresión de una tecla, movimiento del ratón, opresión de algún botón del ratón, liberación de un botón del ratón, cambio de forma de una ventana, exposición de la ventana y despliegue Un evento de cambio de forma de una ventana ocurre cada vez que la ventana del programa es cambiada de tamaño Un evento de exposición ocurre cuando la ventana del programa se despliega inicialmente, y también cada vez que es traída al frente Un evento de despliegue ocurre después de que uno o más de los demás eventos son manejados

Los comandos de OpenGL utilizan el prefijo **gl** y letras iniciales mayúsculas para cada palabra hasta formar el nombre comando, como `glBegin()` Similarmente, las constantes definidas comienzan con **GL_**, están escritas completamente con mayúsculas y utilizan guiones bajos para separar palabras, como `GL_COLOR_BUFFER_BIT`

Algunos nombres de comandos de OpenGL tienen una, dos o tres letras al final para denotar el número y el tipo de parámetros al comando El primer caracter indica el número de valores del tipo indicado que deben ser presentados al programa El segundo caracter o par de caracteres indican el tipo específico de los argumentos: entero de 8 bits, entero de 16 bits, entero de 32 bits, punto flotante de precisión sencilla, o punto flotante de precisión doble El caracter final, de estar presente, es *v*, indicando que el comando toma un apuntador a un arreglo (un *v*ector) de valores, en vez de una serie de argumentos individuales.

Por ejemplo, en el comando `glVertex3fv()`, '3' es utilizado para indicar tres argumentos, 'f' se usa para indicar que los argumentos son de punto flotante y 'v' indica que los argumentos están en formato de vector Las letras y números usados en los sufijos para la implementación en ANSI C se describen a continuación:

- Número de argumentos: 2, 3 ó 4
- Tipos de datos:
  - 'f' flotante
  - 'd' flotante doble
  - 's' entero corto con signo
  - 'i' entero con signo
  - 'b' caracter
  - 'ub' caracter sin signo
  - 'us' entero corto sin signo
  - 'iu' entero sin signo

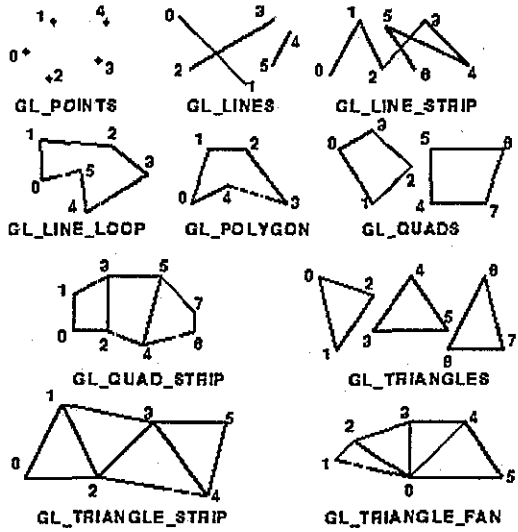
[4] Mapear: Establecer una relación de elementos de un conjunto a elementos de otro conjunto Todos los elementos del conjunto origen deben tener un elemento destino Distintos elementos del conjunto origen pueden tener el mismo elemento destino El número de elementos del conjunto origen y el destino son completamente independientes entre si

- Formatos:
  - 'v' indica formato de vector
  - la ausencia de 'v' indica formato escalar

En algunas ocasiones, se hace referencia a los comandos de OpenGL por sus nombres base solamente, y se añade un asterisco para indicar que puede haber un sufijo al nombre del comando como `glColor*()`

A.1 Primitivas de OpenGL

El programador dispone de las siguientes primitivas para su utilización en la construcción de objetos geométricos



Cada objeto geométrico está descrito por un conjunto de vértices y el tipo de la primitiva a ser dibujada Cómo están conectados los vértices está determinado por el tipo de primitiva

Con OpenGL todos los objetos geométricos se describen ultimadamente como un conjunto ordenado de vértices El comando `glVertex*()` se utiliza para especificar un vértice. A continuación se presentan algunos ejemplos de la utilización de `glVertex*()`

```
glVertex2s(1,2);
glVertex3d(0.0, 0.0, 3.1415926535898);
glVertex4f(1.3,2.0,-4.2,1.0);

GLdouble vector[3] = {3.0, 10.0, 2033.0};
```

```
glVertex3dv(vector);
```

Todas las llamadas a `glVertex*()` deben ocurrir entre un par `glBegin()` y un par `glEnd()` Se dibuja cada vez que se alcanza un par `glBegin()/glEnd()` a excepción de las listas de despliegue que veremos posteriormente

Es importante hacer notar que los comandos OpenGL no se ejecutan necesariamente tan pronto como son solicitados Es necesario llamar al comando `glFlush()` para asegurar que todos los comandos previamente solicitados se ejecuten `glFlush()` se llama generalmente al final de una secuencia de comandos de dibujo para asegurar que todos los objetos en una escena son dibujados

Es importante recalcar que el orden en que los vértices se declaran es importante También que algunas primitivas, cuando se les proporciona un número incorrecto de vértices, ignorarán cualesquiera vértices adicionales Por ejemplo, `GL_TRIANGLES` sólo dibuja al triángulo que corresponde a los vértices 1, 2 y 3 Los vértices 4 y 5 se ignoran

A.2 Cambios de estado

En tanto que una primitiva geométrica se dibuja, cada uno de sus vértices se afecta por las variables de estado actual de OpenGL Estas variables de estado especifican información como ancho de línea, tramado de línea, color, método de sombreado, niebla, selección de polígonos, etc

Algunas variables de estado hacen referencia a las capacidades de OpenGL que pueden estar desactivadas (con el valor `GL_FALSE`) o activadas (con `GL_TRUE`) Otras variables de estado hacen referencia a cierto modo (con el valor del tipo `GLenum`) elegido de un conjunto fijo de modos Finalmente, existen variables de estado con cierto valor asignado (`GLfloat`, `GLint`, etc)

Cada variable de estado tiene un estado por omisión Los valores de las variables de estado, ya sea que sean asignados por omisión o por el programador, permanecen activas hasta que sean cambiadas

A.2.1 Variables de Estado "activadas" o "desactivadas"

Las variables de estado que hacen referencia a las capacidades de OpenGL pueden ser activadas o desactivadas con los comandos `glEnable(GLenum` *opción*`)` y `glDisable(GLenum` *opción*`)` respectivamente, donde *opción* representa una constante simbólica que indica una característica de OpenGL

Su valor actual puede ser obtenido usando el comando `glIsEnabled(GLenum` *opción*`)` que regresa `GL_TRUE` o `GL_FALSE`

Algunas opciones de OpenGL incluyen:

GL_POINT_SMOOTH	Si está activada, dibuja puntos con filtrado apropiado. De *otra forma, dibuja puntos suavizados*
GL_LINE_SMOOTH	Si está activada, dibuja líneas con filtrado correcto. De otra forma, líneas suavizadas.
GL_LINE_STIPPLE	Si está activada, usa el entramado de línea actual cuando dibuja líneas.

A.2.2 Variables de estado con modo

Las variables de estado con modo requieren comandos específicos a la variable de estado siendo accesada con objeto de cambiar su valor. Un ejemplo de esto es el comando para seleccionar la variable de estado de sombreado, `GL_SHADE_MODEL`.

Una línea o una primitiva de polígono lleno puede ser dibujado con un solo color (sombreado plano) o con muchos diferentes colores (sombreado suave, también llamado *Sombreado de Gouraoud*). La técnica deseada de sombreado se puede especificar con el comando `glShadeModel(GLenum` *modo*`)` donde *modo* es ya sea `GL_SMOOTH` para sombreado suave o `GL_FLAT` para el sombreado plano, que es el valor por omisión.

A.2.3 Variables de estado con valor

Las variables de estado con valor requieren comandos específicos a la variable de estado siendo accesada con objeto de cambiar su valor.

En cualquier punto, el programador puede preguntarle al sistema el valor actual de cualquier variable. Típicamente, uno de los cuatro siguientes comandos se usan para hacer eso dependiendo del tipo deseado de la respuesta: `glGetBooleanv()`, `glGetDoublev()`, `glGetFloatv()` o `glGetIntegerv()`.

A.2.4 Color

La variable más comúnmente usada del comando para establecer los colores RGBA[5] es `glColor3f(GLfloat` *rojo*`, GLfloat` *verde*`, GLfloat` *azul*`)` donde *rojo*, *verde* y *azul* son valores entre 0.0 y 1.0, inclusive. El valor 0.0 corresponde a la cantidad mínima de color en tanto que 1.0 corresponde a la máxima cantidad de color.

[5] RGBA: Red Green Blue Alpha: Rojo Verde Azul y Transparencia

Colores RGB comúnmente utilizados

`glColor3f(0.0, 0.0, 0.0);`	**Negro**
`glColor3f(1.0, 0.0, 0.0);`	**Rojo**
`glColor3f(0.0, 1.0, 0.0);`	**Verde**
`glColor3f(0.0, 0.0, 1.0);`	**Azul**
`glColor3f(1.0, 1.0, 0.0);`	**Amarillo**
`glColor3f(0.0, 1.0, 1.0);`	**Azul celeste**
`glColor3f(1.0, 0.0, 1.0);`	**Magenta**
`glColor3f(1.0, 1.0, 1.0);`	**Blanco**

A.2.5 Tamaño del punto.

Para controlar el tamaño de un punto renderizado, se usa el comando `glPointSize(GLfloat` *tamaño*`)` donde *tamaño* es el ancho deseado en pixeles. *tamaño* debe ser mayor que 0.0 y por omisión es 1.0

A.2.6 Ancho de línea

Para controlar el ancho de las líneas, se usa el comando `glLineWidth(GLfloat` *ancho*`)` donde *ancho* es el ancho deseado en pixeles para las líneas renderizadas. *ancho* debe ser mayor que 0.0 y por omisión es 1.0

A.2.7 Entramado de líneas

Para hacer líneas con entramado (punteadas o segmentadas), el patrón de enramado debe ser definido usando el comando `glLineStipple(GLint` *factor*`, GLushort` *patrón*`)`, donde el argumento *patrón* es una serie de 16 bits con ceros y unos, y se repite cuantas veces sea necesario para entramar una línea dada. Un 1 indica que se dibuja y un cero, que no se hace, en una base de píxel por píxel. El patrón puede ser estirado usando *factor*, el cual multiplica cada subserie de unos y ceros consecutivos. *factor* puede tener cualquier valor entre 0 y 255, inclusive. La variable de opción `GL_LINE_STIPPLE` debe ser activada con la llamada a `glEnable(GL_LINE_STIPPLE)` para que ocurra el entramado.

A.3 Dibujo en 3 dimensiones

En una escena tridimensional, los objetos más cercanos al punto de vista pueden obstruir a los objetos más allá del punto de vista. Para dibujar una escena realista, deben eliminarse las superficies ocultas, de tal forma que se asegure que las partes ocultas de los objetos no estén visibles. La eliminación de las superficies ocultas se logra a través del uso de un búfer de profundidad y pruebas de profundidad.

Cuando realiza el dibujo, una profundidad, o un valor de *z* se asocia con cada píxel y se almacena en el búfer de profundidad. Una prueba de profundidad se realiza para cada vértice en la escena. Durante una prueba de profundidad, el búfer de profundidad se verifica para ver si otro vértice ha sido previamente dibujado que podría ocultar al vértice actual. De ser así, el vértice no es dibujado.

La técnica de búfer de profundidad debe ser activada con el comando `glEnable(GL_DEPTH_TEST)`. Antes de dibujar una escena, el búfer de profundidad debe ser borrado con el comando `glClear(GL_DEPTH_BUFFER_BIT)`.

Por omisión, OpenGL dibuja polígonos en modo rellenado llenando todos los pixeles incluidos dentro de los límites de éste, pero los polígonos también pueden ser dibujados en modo de línea dibujando solamente el contorno, o en modo de punto como puntos en los vértices. En OpenGL un polígono tiene un lado frontal y uno trasero. Cada lado puede ser dibujado del mismo modo o en modos distintos para permitir vistas cortadas de objetos sólidos, por ejemplo.

El comando `glPolygonMode(GLenum` *cara*`, GLenum` *modo*`)` controla el modo de dibujo para las caras frontal y trasera de un polígono. El parámetro *cara* puede ser `GL_FRONT`, `GL_BACK` o `GL_FRONT_AND_BACK`; *modo* puede ser `GL_POINT`, `GL_LINE` o `GL_FILL` para indicar si el polígono será dibujado como puntos, contorno, o relleno. Por omisión, tanto la cara frontal como la trasera son dibujadas.

Un polígono rellenado puede ser llenado sólidamente o con un entramado. El patrón de entramado debe estar definido usando el comando `glPolygonStipple(const GLubyte *`*máscara*`)` donde el argumento *máscara* es un apuntador a un mapa de bits de 32x32 que se interpreta como una máscara de ceros y unos. Un 1 indica que el píxel correspondiente en el polígono es dibujado, y 0, que no lo es. La variable de opción `GL_POLYGON_STIPLE` debe estar activada por la llamada `glEnable(GL_POLYGON_STIPPLE)` para que el entramado ocurra.

El comando `glCullFace(GLenum` *modo*`)` indica cuáles polígonos deberán ser descartados antes de que sean convertidos a coordenadas en pantalla. El parámetro *modo* es `GL_FRONT`, `GL_BACK`, o `GL_FRONT_AND_BACK` para indicar todos aquellos que están de cara frente, todos aquellos que están de cara hacia atrás o todos los polígonos. Para que tome efecto la variable de estado para selección ,`GL_CULL_FACE` debe estar activada.

A.4 Transformaciones de vista y de modelo

La transformación de vista es análoga a posicionar y enfocar una cámara. La transformación del modelo es análoga a posicionar y orientar el modelo u objetos a ser dibujados. Las transformaciones de vista deben preceder a las transformaciones de modelo en OpenGL.

Estas transformaciones se realizan a través del uso de matrices. En cualquier punto en un programa OpenGL una de las matrices es modificable: la matriz de vista de modelo, la matriz de proyección o la matriz de textura. Los usos de la matriz de textura serán descritos posteriormente. La variable de estado `GL_MATRIX_MODE` especifica qué matriz es actualmente modificable. La matriz modificable por omisión es la matriz de vista de modelo.

El comando `glMatrixMode(GLenum` *modo*`)`, se utiliza para cambiar el valor de la variable de estado `GL_MATRIZ_MODE`. Los valores posibles para el argumento de *modo* son: `GL_MODELVIEW`, `GL_PROJECTION` y `GL_TEXTURE`. Cuando se llama a `glMatrixMode()`, todos los comandos subsecuentes de transformación afectan a la matriz especificada.

El comando `glLoadIdentity()` se usa para que la matriz actualmente modificable sea igual a la matriz identidad.

El comando `glTranslate(TYPE` *x*`, TYPE` *y*`, TYPE` *z*`)` multiplica la matriz actualmente modificable por una matriz que mueve a un objeto según los valores *x*, *y* y *z* dados (o mueve el sistema local de coordenadas en dichas cantidades).

El comando `glTranslate(TYPE` *ángulo*`, TYPE` *x*`, TYPE` *y*`, TYPE` *z*`)` multiplica la matriz actualmente modificable por una matriz que rota a un objeto (o el sistema local de coordenadas) en un sentido contrario a las manecillas del reloj en torno al rayo que va del origen hasta el punto (*x*, *y*, *z*). El parámetro de *ángulo* especifica el ángulo de rotación en grados.

El comando `glScale(TYPE` *x*`, TYPE` *y*`, TYPE` *z*`)` multiplica la matriz actualmente modificable por una matriz que estira, encoge o refleja un objeto a lo largo de los ejes. Cada coordenada *x*, *y* y *z* de todo punto en el objeto es multiplicada por el argumento correspondiente *x*, *y* o *z*. Desde el punto de vista de el sistema de coordenadas local, los ejes de coordenadas locales se estiran por los factores *x*, *y* y *z*, y el objeto asociado es alargado junto con ellos.

A.5 Transformación del área de visión

La transformación del área de visión es análoga a elegir el tamaño de la fotografía revelada. Por omisión, OpenGL establece que el área de visión sea el rectángulo de pixeles

completo de la ventana original del programa El comando `glViewport (GLint x, GLint y, GLsizei ancho, GLsizei altura)` se usa para cambiar el tamaño de la región de dibujo Los parámetros *x* y *y* especifican la esquina superior izquierda del área de visión y *ancho* y *altura* son el tamaño del rectángulo del área de visión Este comando a menudo se usa para manejar los eventos de cambio de forma (tamaño) de la ventana

A.6 Proyección de perspectiva

En las proyecciones de perspectiva, entre más lejos esté un objeto de la 'cámara', más pequeño aparece en la imagen final Esto se logra a través del uso de un volumen de visión con forma de cono

El comando para crear un volumen de proyección de perspectiva es `glFrustum(GLdouble izquierda, GLdouble derecha, GLdouble abajo, GLdouble arriba, GLdouble cerca, GLdouble lejos)` Los argumentos *izquierda* y *derecha* especifican las coordenadas para los planos de recorte Los argumentos *abajo* y *arriba* especifican las coordenadas para los planos horizontales de recorte superior e inferior Los argumentos *cerca* y *lejos* especifican las distancias de los planos de recorte de profundidad Ambas distancias deben ser positivas

A.6.1 Proyección ortográfica

Con una proyección ortográfica, el volumen de proyección tiene forma de una caja A diferencia de la proyección de perspectiva, el tamaño del volumen de proyección no cambia de un cabo a otro Por tanto, la distancia desde la 'cámara' no afecta qué tan grande se ve un objeto

El comando para crear un volumen de visión ortográfico es `glOrtho(GLdouble izquierda, GLdouble derecha, GLdouble abajo, GLdouble arriba, GLdouble cerca, GLdouble, lejos)` Los argumentos *izquierda* y *derecha* especifican las coordenadas para los planos verticales de recorte izquierdo y derecho Los argumentos *abajo* y *arriba* especifican las coordenadas para los planos horizontales de recorte superior e inferior. Los argumentos *cerca* y *lejos* especifican las distancias a los planos de recorte de profundidad más cercano y más lejano Estas distancias son negativas si se desea que el plano esté atrás del espectador

A.7 Iluminación

OpenGL provee dos tipos de fuentes de luz: direccional y posicional Una fuente de luz direccional se considera como si estuviera alejada a una distancia infinita de los objetos en la escena De esta forma, sus rayos de luz se consideran paralelos al momento de alcanzar al objeto Una luz posicional está cerca o dentro de la escena y la dirección e sus rayos se

toma en cuenta para cálculos de iluminación. Las luces posicionales tienen un mejor desempeño que las luces direccionales debido a estos cálculos adicionales

El comando `glLightfv()` se usa para especificar la posición de la luz y su tipo: direccional o posicional. Se usa también para especificar los valores de los componentes de color de la fuente de luz como color de ambiente, color de difusión, color especular, color de emisión y brillantez.

Una vez que las fuentes de luz se definen, los vectores normales y las propiedades materiales de los objetos en la escena deben también definirse. Los vectores normales de un objeto definen su orientación relativa a las fuentes de luz. Los vectores normales pueden especificarse para cada vértice y/o ser compartidos por vértice. El comando `glNormal()` se usa para hacer esta asignación

Las componentes de color especificadas para las luces tienen distintos significados que aquellos especificados para los materiales. Para las luces, los números corresponden al porcentaje de intensidad de cada color. Así, el valor de luz blanca más brillante tiene valores de RGB de (1.0, 1.0, 1.0). Para los materiales, los números corresponden a las proporciones reflejadas de dichos colores. El comando `glMaterialfv()` se usa para definir las propiedades materiales al especificar los valores de las componentes de color de los materiales

El comando `glColorMaterial()` se usa para minimizar los costos de desempeño asociados con el cambio de las propiedades del material. Este comando debe ser usado siempre que una propiedad material, como color difuso, deba ser cambiada para la mayoría de los vértices en la escena. Cualquier cambio hecho al color actual por una llamada a *glColor() inmediatamente actualiza la propiedad de material especificada* por `glColorMaterial()`. La variable de opción `GL_COLOR_MAIERIAL` debe ser activada por la llamada `glEnable(GL_COLOR_MATERIAL)`

Por último, la iluminación debe ser activada por una llamada a `glEnable(GL_LIGHTING)` y cada luz en la escena debe ser activada con llamadas a `glEnable(GL_LIGHTi)` donde `GL_LIGHTi` es el nombre simbólico de la luz

APÉNDICE B. GLUT

B.1 Introducción

GLUT es una conjunto de utilidades diseñado para OpenGL. GLUT Significa: OpenGL Utility Toolkit Es una interfaz de programación con enlaces para ANSI C y FORTRAN para escribir programas en OpenGL independientes del sistema de ventanas que se esté utilizando El conjunto de herramientas proporciona la siguiente funcionalidad:

- Múltiples ventanas para la renderización en OpenGL
- Procesamiento de eventos controlado por llamadas
- Dispositivos de entrada sofisticado
- Una rutina de inactividad y temporizadores
- Facilidades para un menú emergente de cascada sencillo
- Rutinas de utilidad para generar diversos objetos sólidos y de malla de alambre
- Soporte para fuentes de mapa de bits
- Funciones diversas de manejo de ventanas y capas

Existen implementaciones en ANSI C de GLUT para el Sistema X Window, Windows NT y OS/2

B.1.1 Preliminares

Uno de los mayores logros en la especificación de OpenGL fue el aislamiento del modelo de renderización de OpenGL con respecto a la dependencia del sistema de ventanas El resultado es que OpenGL es independiente del sistema de ventanas

Las operaciones del sistema de ventanas como la creación de una ventana de renderización y el manejo de los eventos del sistema de ventanas son definidos por el sistema de ventanas nativo Las interacciones necesarias entre OpenGL y el sistema de ventanas como crear y enlazar un contexto OpenGL a una ventana se describen separadamente de la especificación OpenGL en una especificación dependiente del sistema de ventanas Por ejemplo, la especificación GLX describe el estándar a través del cual OpenGL interactúa con el sistema de ventanas X Window

El predecesor a OpenGL es IRIS GL A diferencia de OpenGL, IRIS GL especifica cómo se crean y manipulan las ventanas de renderización La interfaz para manejar las ventanas de IRIS GL es razonablemente popular, en gran parte debido a que es fácil de usar Los programadores de IRIS GL pueden preocuparse sobre la programación gráfica sin necesidad de ser expertos en programación del sistema nativo de ventanas La experiencia demostró también que la interfaz de ventaneo de IRIS GL tenía el suficientemente alto nivel de tal forma que podía ser redistribuida a sistemas de ventaneo diferentes Silicon

Graphics migró de NeWS al Sistema X Window sin grandes cambios a la interfaz básica de ventaneo de IRIS GL

El remover las operaciones de sistema de ventanas de OpenGL es buena decisión porque permite que el sistema gráfico OpenGL se redistribuya a varios sistemas incluyendo tanto estaciones gráficas poderosas pero caras como sistemas gráficos de producción en masa como videojuegos, aparatos para televisión interactiva y PCs.

Desafortunadamente, la falta de una interfaz de sistema de ventanas es un hueco en la utilidad de OpenGL. Aprender las APIs del sistema de ventanas nativo como Xlib para X Window, o Motif, puede ser muy complicado. Incluso aquellos que tienen familiaridad con las APIs del sistema nativo necesitan entender la interfaz que conecta a OpenGL con el sistema de ventanas. Y cuando un programa de OpenGL se escribe usando la interfaz nativa de ventanas, a pesar la portabilidad del código de renderización de OpenGL, el programa será en sí dependiente del sistema de ventanas.

Esto condujo al desarrollo de los conjuntos de herramientas `tk` y `aux`. El conjunto de herramientas `aux` se usa en los ejemplos encontrados en la *Guía de Programación de OpenGL*. Desafortunadamente, `aux` tiene numerosas limitaciones y su utilidad está fuertemente limitada a programas pequeños. La librería `tk` tiene más funcionalidad que `aux`, pero fue desarrollada en una forma específica, y aún carece de funcionalidad importante que los programadores de IRIS GL esperarían, como menús emergentes y capas.

GLUT está diseñado para llenar la necesidad de una interfaz independiente del sistema de ventanas para programas de OpenGL. La interfaz ha sido diseñada para ser simple, pero aún cumple las necesidades de programas OpenGL útiles. Se han incluido características de las interfaces IRIS GL, `aux` y `tk` para que a quienes estén acostumbrados a usar estas interfaces, les resulte fácil desarrollar programas para GLUT.

B.1.2 Filosofía de Diseño.

GLUT simplifica la implementación de programas que utilicen renderización OpenGL. La API De GLUT requiere pocas rutinas para desplegar una escena gráfica renderizada usando OpenGL. La API de GLUT (como la de OpenGL) es estacional. La mayor parte del estado inicial está ya definido, y el estadio inicial es razonable para programas simples.

Las rutinas de GLUT también toman, relativamente, pocos parámetros. No se regresan apuntadores. Los únicos apuntadores pasados a GLUT son apuntadores a cadenas de caracteres (todas las cadenas pasadas a GLUT son copiadas, no referenciadas) y a asas (manijas) de fuentes opacas.

La API de GLUT es (tanto como sea razonable) independiente del sistema de ventanas. Por esta razón, GLUT no regresa *ninguna* manija del sistema nativo de ventanas,

apuntadores, u otras estructuras de datos Se evitan dependencias más sutiles hacía el sistema de ventanas, como la de las fuentes dependientes del sistema de ventanas, proporcionando fuentes propias de GLUT, aunque limitadas

Para facilitar la programación, GLUT provee una sub-API de menús simples Aunque el soporte de menús está diseñado para que éstos sean implementados como menús emergentes, GLUT deja abierta la posibilidad al sistema de ventanas de apoyar la funcionalidad de menús en otra forma (menús de persiana, por ejemplo)

Dos de las piezas más importantes del estado de GLUT son *ventana actual* y *menú actual* La mayoría de las rutinas de ventana y menú afectan a la *ventana actual* o al *menú actual,* respectivamente La mayoría de las llamadas implícitamente cambian a *ventana* y *menú actuales* a la ventana o menú responsable de la llamada GLUT está diseñado de tal forma que un programa con una sola ventana y/o menú no necesita seguirle la pista a ningún identificador de ventana o menú Esto simplifica enormemente los programas muy simples escritos con GLUT

GLUT está diseñado para programas que van desde simples a moderados enfocados en renderización con OpenGL. GLUT implementa su propio bucle de eventos Por esta razón, el mezclar a GLUT con otras APIs que requieren su propia estructura de manejo de eventos puede ser difícil La ventaja de tener un bucle de atención a eventos ya incluido es la simplicidad

GLUT contiene rutinas para renderizar fuentes y objetos geométricos. Sin embargo GLUT no toma parte del espacio de nombres de la lista de despliegue de OpenGL Por esta razón, ninguna de las rutinas de renderización de GLUT usa listas de despliegue de OpenGL. Depende del programador en GLUT, compilar la salida de las rutinas de renderización de GLUT para convertirlas en listas de despliegue, si esto es lo que se desea

Las rutinas de GLUT se organizan lógicamente en diversas sub-APIs dependiendo de su funcionalidad Las sub-APIs son:

Inicialización El proceso de la línea de comandos, la inicialización del sistema de ventanas y el estado para la creación de la ventana inicial se controlan desde esta rutina

Inicio de Procesamiento de eventos Esta es la rutina de procesamiento de eventos de GLUT Esta rutina nunca regresa, y continuamente hace llamadas a GLUT cuantas veces sea necesario

Manejo de Ventanas Estas rutinas crean y controlan ventanas

Manejo de Capas Estas rutinas establecen y manejan capas para ventanas

Manejo de Menús Estas rutinas crean y controlan menús emergentes

Registro de Llamadas Estas rutinas registran llamadas a ser efectuadas por el bucle de procesamiento de eventos de GLUT

Manejo de Mapeo de Colores Estas rutinas permiten la manipulación de los mapas de colores de Windows

Obtención de Estado Estas rutinas permiten que los programas obtengan el estado de GLUT

Renderización de Fuentes Estas rutinas permiten la renderización de fuentes de mapas de bits

Renderización de Formas Geométricas Estas rutinas permiten la renderización de objetos geométricos tridimensionales, incluyendo esferas, conos, icosaedros y teteras

B.1.3 API Versión 2

En respuesta a la retroalimentación de la versión original de GLUT, se desarrolló la versión 2 de la API de GLUT Se añadieron a la API original de GLUT:

- Soporte para requisición de ventanas estéreo y de multimuestra
- Nuevas rutinas para verificar y proveer el soporte de ciertas llamadas, para dispositivos de entrada sofisticados: spaceball, tableta, y caja con dial y botón
- Nueva rutina para registrar una llamada para una función de teclado y teclas direccionales En la versión 1, sólo podían generarse caracteres ASCII
- Nuevas rutinas para verificación de capacidades estéreo, de multimuestra y tiempo transcurrido
- Nueva rutina para facilitar la verificación de soporte de extensiones OpenGL

La versión 2 de la API de GLUT es completamente compatible con la versión 1 de la API

B.1.4 API Versión 3

Más retroalimentación condujo al desarrollo de la API de GLUT versión 3 Las mejoras con respecto a la versión dos son:

- La función `GLUIMenuStateFunc` fue desaprobada, a favor de `glutMenuStatusFunc`
- `glutFullScreen` requiere ventanas de pantalla completa de nivel superior
- Tres fuentes *Helvética* de mapa de bits adicionales
- Las implementaciones deberán hacer cumplir que no se permita ninguna modificación a los menús cuando éstos están en uso

- `glutBitmapWidth` y `GLUTStrokeBitmap` regresan los anchos de los caracteres individuales
- `GLUTGetModifiers` llamada durante una llamada de teclado, ratón o una especial, regresa los modificadores (`Shift`, `Ctrl` , `Alt`) oprimidos cuando el evento de ratón o teclado fue generado
- Acceso a las capas transparentes por ventana cuando se tiene hardware para soporte de capas Las rutinas añadidas son: `glutEstablishOverlay`, `glutRemoveOverlay`, `glutShowOverlay`, `GLUTHideOverlay`, `glutUseOverlay`, `glutLayerGet` y `glutPostOverlayRedisplay`
- Un nuevo modo de despliegue llamado `GLUT_LUMINANCE` que usa el modelo de color `RGBA` de OpenGL, pero no tiene componentes azules o verdes El componente rojo se convierte en un índice y se busca en un mapa de colores escribible para determinar los colores desplegados La función: `glutInitDisplayMode`

La versión 3 de la API de GLUT deberá ser grandemente compatible con la versión 2, sin embargo, los programas que solían modificar a los menús (a través de alguna temporización fortuita) mientras los menús estaban en uso, encontrarán errores fatales cuando hagan esto en la versión 3

Otro cambio en GLUT 3 0 que podría requerir modificación a los programas escritos con GLUT con versiones anteriores a la 3 0 es que éste ya no permite que una ventana sea mostrada sin registrar una llamada de despliegue Este cambio asegura que las ventanas no se desplieguen en pantalla sin que GLUT les proporcione una forma de ser renderizada

B.1.5 Convenciones

La ventana de GLUT y las coordenadas en pantalla se expresan en pixeles. La esquina superior izquierda de la pantalla o una ventana es `(0,0)` Las coordenadas en `X` se incrementan de izquierda a derecha Las coordenadas en `Y` se incrementan de arriba hacia abajo Nota: Esto es inconsistente con el esquema de coordenadas de OpenGL que generalmente considera que la coordenada izquierda inferior de una ventana sea `(0,0)`, pero es consistente con los sistemas de ventanas más populares

Los identificadores enteros en GLUT comienzan en uno, no cero También los identificadores de ventana, de menú y los índices de elementos de menú están basados en uno, no en cero

En la construcción con ANSI C, para la mayoría de las rutinas se usan tipos básicos como parámetros `(int, char *)` En las rutinas donde los parámetros se pasan directamente a las rutinas OpenGL, se usan tipos de OpenGL (como `GLfloat`)

El archivo de encabezado de GLUT deberá ser incluido en los programas GLUT con la siguiente directiva:

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

Debido a que un vendedor de sistemas de ventanas (que permanecerá sin ser nombrado) tiene una incapacidad aparente de apreciar que la API de OpenGL es independiente de su sistema de ventanas, los programas portables GLUT ANSI C deberán confiar en que `<GL/glut.h>` incluirá los archivos de encabezado necesarios de OpenGL y GLU, en vez de incluir directamente `<GL/gl.h>` ó `<GL/glu.h>`.

El archivo de librería GLUT ANSI C se llama usualmente `libglut.a` en los sistemas UNIX. Los programas GLUT necesitan enlazarse con las librerías del sistema OpenGL y GLUT (y cuelesquiera otras librerías de las cuales éstas dependan). Un conjunto de librerías dependientes del sistema de ventanas podría ser necesario también para el enlace de los programas GLUT. Por ejemplo, los programas que utilizan la implementación X11 de GLUT necesitan típicamente ser enlacados con Xlib, la librería de extensión de X, posiblemente la librería de extensión de entrada X, la librería de utilidades diversas X, y la librería matemática. Una línea de ejemplo de compilación X11/Unix se vería:

```
cc -o foo foo.c -lGLUT -lGLU -lGL -lXmu -lXi -lXext -lX11 -lm
```

B.1.6 Terminología

Cierto número de términos se usan en una manera específica a GLUT a lo largo de este documento. El significado de GLUT de estos términos es independiente del sistema de ventanas en el cual se utilice GLUT. He aquí los significados específicos para los siguientes términos específicos a GLUT:

Llamada Un rutina especificada por el programador que puede ser registrada con GLUT para ser, precisamente llamada, en respuesta a un tipo específico de evento. También se usa para hacer referencia a una rutina de llamada siendo llamada.

Mapa de Color. Un mapeo de los valores de píxel a valores de color de RGB. Usado por ventanas de índice de color.

Diales y caja de botones Un dispositivo de entrada sofisticado que consiste en un conjunto de botones y un arreglo de diales rotativos, a menudo utilizado por programas de diseño asistido por computadora.

Modo de despliegue. Un conjunto de capacidades de búfer de cuadro de OpenGL que pueden ser atribuibles a una ventana.

Inactivo. Un estado en el cual no se reciben eventos del sistema de ventanas para ser procesados como llamadas. Si la llamada inactiva está registrada, es llamada.

Capa en uso. El plano normal o el revestimiento. Este estado de ventana define cuál capa de búfer de cuadro es afectada por los comandos de OpenGL.

Entrada de Menú. Un elemento de menú que el usuario puede seleccionar para activar la llamada de menú para este valor de entrada de menú.

Elemento de Menú. Ya sea una entrada de menú, o un activador de sub-menú

Modificadores. Las teclas `Shift`, `Ctrl` y `Alt` que pueden ser oprimidas simultáneamente con una tecla o el botón del ratón al ser oprimido o soltado.

Multimuestreo. Una técnica para suavizado por hardware disponible sólo en caro hardware gráfico de 3D. Cada píxel está compuesto de un número de muestras (cada una conteniendo información e color y profundidad). Las muestras se promedian para determinar el valor del color del píxel desplegado. El multimuestreo se soporta como una extensión a OpenGL.

Plano normal. La capa de búfer de cuadro por omisión donde reside el estado de ventana de GLUT, en oposición al *revestimiento*

Revestimiento. Un búfer de cuadro que puede ser desplegado preferentemente al *plano normal* y soporta la transparencia a través del *plano normal*. Los revestimientos son útiles para efectos de banda de hule, anotación de textos y otras operaciones, para evitar dañar el estado normal del búfer de cuadro plano. Los revestimientos requieren soporte de hardware, el cual no está siempre disponible en todos los sistemas

Sacar. El acto de forzar una ventana al frente del orden de acomodación para ventanas hermanas

Menú emergente. Un menú que puede aparecer cuando se oprime un botón especificado del ratón. Un menú emergente corresponde de múltiples elementos de menú

Meter. El acto de forzar una ventana hasta atrás del orden de acomodación para ventanas hermanas

Cambiar la forma. El acto de cambiar el tamaño o la forma de una ventana

Spaceball. Un dispositivo sofisticado de entrada en 3D que provee seis grados de libertad, tres ejes de rotación y tres ejes de traslación. Además, soporta varios botones. El dispositivo es una pelota del tamaño de una mano unida a una base. Al cubrir a la pelota con una mano, y al aplicar fuerza de torsión o direccional, se generan rotaciones y traslaciones

Estéreo. Una opción de búfer de cuadro que provee búferes de color para el lado izquierdo y derecho para crear renderizaciones estereoscópicas. Usualmente el usuario usa lentes con obturador de LCD sincronizados con el despliegue alternante en la pantalla de los búferes de color izquierdo y derecho

Sub-menú Un menú en cascada desde algún activador de sub-menú +

Activador de Sub-menú Un elemento de menú que el usuario puede introducir para activar un sub-menú

Sub-ventana Un tipo de ventana que es la ventaja hija de una ventana del nivel superior, u otra sub-ventana La región visible y la de dibujo de una sub-ventana están limitadas por su ventana ascendiente

Tableta Un dispositivo preciso de entrada en 2D Como un ratón, se envían coordenadas bidimensionales La posición absoluta de la opresión en la tableta es enviada Las tabletas también soportan varios botones

Temporizador Una llamada que puede ser calendarizada para que sea llamada en un lapso de tiempo específico

Ventana Un área rectangular para renderización con OpenGL

Estado de la ventana de despliegue Uno de los siguientes: mostrada, escondida o iconizada Una ventana mostrada es visible potencialmente en la pantalla (puede estar obstruida por otras ventanas y no estar realmente visible) Una ventana escondida nunca estará visible Una ventana iconizada no es visible, pero puede hacerse visible en respuesta a alguna acción del usuario como hacer clic en el ícono correspondiente a la ventana

Sistema de Ventanas Una noción amplia se refiere tanto al mecanismo y a la política del sistema de ventanas Por ejemplo, en el sistema de ventanas X Window, tanto el controlador de ventanas y el servidor X están integrados en lo que GLUT considera el sistema de ventanas

B.2 Inicialización

Las rutinas que comienzan con el prefijo `glutInit-` se utilizan para inicializar el estado de GLUT La primera rutina de inicialización es `glutInit` que deberá ser llamada sólo una vez en un programa de GLUT No deberán llamarse rutinas de GLUT u OpenGL que no tengan el prefijo `glutInit-` antes de llamar a `glutInit`

La razón por la que hay otras rutinas `glutInit-` que pueden ser llamadas antes de `glutInit` es que estas pueden ser usadas para establecer el estado por omisión de inicialización de ventanas que podrá ser modificado por el comando de procesamientos hecho en `glutInit` Por ejemplo, `glutInitWindowSize(400,400)` puede ser llamado antes de `glutInit` para indicar que 400 por 400 es el valor por omisión de tamaño de ventana del programa El establecer el *tamaño inicial de ventana* o *posición* antes de `glutInit` permite al usuario del programa GLUT especificar el tamaño inicial o posición usando argumentos en la línea de comandos

B.2.1 glutInit

`glutInit` se usa para inicializar a la librería GLUT.

uso

```
void glutInit (int *argcp, char **argv);
```

GLUT apuntador a la variable del programa *no modificada* `argc` de `main`. Como regreso, el valor apuntado por `argcp` se actualizar, porque `glutInit` extrae cualesquiera opciones de la línea de comandos que se pretendió que se pasaran a la librería GLUT

GLUT variable *no modificada* `argv` de `main`. Como `argcp`, los datos para `argv` serán actualizados porque `glutInit` extrae cualesquiera opciones en la línea de comandos comprendidos por la librería GLUT

descripción

`glutInit` inicializará la librería GLUT y negociará una sesión con el sistema de ventanas. Durante este proceso, `glutInit` puede causar la terminación del programa GLUT con un mensaje de error al usuario si GLUT no puede ser apropiadamente inicializado. Ejemplos de esta situación incluyen la falla de conexión al sistema de ventanas, la falta de soporte de sistema de ventanas para OpenGL y opciones inválidas en la línea de comandos.

`glutInit` también procesa opciones de la línea de comandos, pero las opciones específicas son dependientes del sistema.

notas de la implementación X

El sistema de ventanas X Window define opciones analizadas por `glutInit` como se describe a continuación:

`-display` *DISPLAY* Especifica el servidor X al cual conectarse. Si no se especifica, se utiliza el valor de la variable de entorno `DISPLAY`.

`-geometry` *WxH+X+Y* Determina dónde deberá ser creada la ventana en la pantalla. El parámetro que sigue a `-geometry` deberá tener el formato de la especificación geométrica estándar X. El efecto de usar esta opción es cambiar el *tamaño inicial* y la *posición inicial* de GLUT como si `GLUTInitWindowSize` o `GLUTInitWindowPosition` se llamaran directamente.

`-iconic` Solicita que todas las ventanas de nivel superior sean creadas en un estado icónico.

`-indirect` Forzar el uso de contextos de renderización de OpenGL indirectos.

`-direct` Fuerza el uso de contextos de renderización de OpenGL directos (no todas las implementaciones de GLX soportan contextos directos de renderización) Se genera un error fatal si la renderización directa no es soportada por la implementación OpenGL

Si no se usan `-indirect` o `-direct` para forzar una situación en particular, GLUT tratará de usar renderización directa de ser posible, y si no, renderización indirecta

`-gldebug` Después de procesar las llamadas y/o eventos, verificar si hay errores llamando a `glGetError` Si se reporta un error, imprimir una advertencia buscando el código de error con `gluErrorString` El usar esta opción es útil para detectar errores de OpenGL en tiempo e ejecución

`-sync` Activar transacciones síncronas del protocolo X Esta opción hace más fácil seguirles la pista a errores de protocolo X potenciales

B.2.2 glutInitWindowPosition, glutInitWindowSize

`glutInitWindowPosition` y `glutInitWindowSize` establecen la posición inicial de ventana y tamaño, respectivamente

uso

```
void glutInitWindowSize(int width, int height);
void glutInitWindowPosition(int x, int y);
```

`width`	Ancho en pixeles
`height`	Altura en pixeles
`x`	Posición en X de la ventana en pixeles
`y`	Posición en Y de la ventana en pixeles

descripción

Se requerirá que las ventanas creadas por `glutCreateWindow` sean creadas con la *posición inicial de ventana* y *tamaño*.

El valor inicial del estado de GLUT de la *posición inicial de ventana* es –1 y –1 Si cualquiera de los componentes X o Y es negativo, la posición actual de la ventana es determinada por el sistema de ventanas El valor inicial del *tamaño inicial de ventana* es 300 por 300 Los componentes del *tamaño inicial de ventana* deben ser mayores que cero

El propósito de los valores de *posición inicial de ventana* y *tamaño* es proveer una sugerencia al sistema de ventanas acerca del tamaño inicial y posición de una ventana El sistema de ventanas no tiene obligación de usar esta información Por esto, los programas GLUT no deberán asumir que la ventana fue creada con el tamaño y posición especificados

Un programa GLUT deberá usar la llamada de ventana de cambio de forma para determinar el tamaño real de la ventana

B.2.3 glutInitDisplayMode

`glutInitDisplayMode` establece el *modo inicial de despliegue*.

uso

```
void glutInitDisplayMode (unsigned int mode);
```

`mode`	Modo de despliegue, normalmente la operación lógica de unión bit a bit de las máscaras de modo de despliegue de GLUT Los valores son:
`GLUT_RGBA`	Máscara de bit para seleccionar una ventana en modo RGBA Este es el valor por omisión si no se especifican `GLUT_RGBA` ni `GLUT_INDEX`
`GLUT_RGB`	Un alias para `GLUT_RGBA`
`GLUT_INDEX`	Máscara de bit para seleccionar una ventana en modo índice de color Esto descarta a `GLUT_RGBA` si también fue especificado
`GLUI_SINGLE`	Máscara de bit para seleccionar una ventana con un solo búfer Este es el valor por omisión si no se especifican `GLUT_DOUBLE` ni `GLUI_SINGLE`
`GLUT_DOUBLE`	Máscara de bit para seleccionar una ventana con doble búfer Esto descarta a `GLUT_SINGLE` si también fue especificado
`GLUI_ACCUM`	Máscara de bit para seleccionar una ventana con un búfer de acumulación
`GLUI_ALPHA`	Máscara de bit para seleccionar una ventana con un componente alpha en los búferes de color
`GLUT_DEPIH`	Máscara de bit para seleccionar una ventana con un búfer de profundidad
`GLUT_SIENCIL`	Máscara de bit para seleccionar una ventana con un búfer de esténcil
`GLUT_MULIISAMPLE`	Máscara de bit para seleccionar una ventana con soporte de multimuestreo Si no está disponible el multimuestreo, se creará una ventana sin multimuestreo automáticamente Nota: Tanto las implementaciones del cliente de OpenGL como las del servidor deben soportar la extensión `GIX_SAMPLE_SGIS` para que esté disponible el multimuestreo
`GLUT_STEREO`	Máscara de bit para seleccionar una ventana estéreo
`GLUT_LUMINANCE`	Máscara de bit para seleccionar una ventana con un modelo de color de "luminancia" Este modelo provee la funcionalidad del modelo de color RGBA de OpenGL pero los componentes de color azul y verde no se mantienen en el búfer marco En vez de esto, cada componente rojo de un píxel se convierte en un índice entre cero y `glutGet(GLUT_WINDOW_COLORMAP_SIZE) - 1` y son buscados

en un mapa de color por cada ventana para determinar el color de los pixeles dentro de la ventana El mapa de color inicial de las ventanas con GLUT_LUMINANCE se inicializa como una rampa gris linear, pero puede ser modificada con las rutinas de color de GLUT

descripción

El *modo de despliegue inicial* se utiliza cuando se crean ventanas de nivel superior, sub-ventanas y capas para determinar el modo de despliegue de OpenGL para la ventana o capa a ser creados.

Note que GLUT_RGBA selecciona el modo de color de RGBA, pero no solicita que ningún bit de alpha sea alojado (algunas veces llamado un búfer de alpha o alpha de destino). Para solicitar alpha, especifique GLUT_ALPHA Lo mismo se aplica para GLUT_LUMINANCE

Notas de implementación de GLUT_LUMINANCE

GLUI_LUMINANCE no es soportado en la mayoría de las plataformas OpenGL

B.3 Inicio del procesamiento de eventos

Después de que un programa GLUT ha hecho configuración inicial como crear ventanas y menús, los programas GLUT entran en el bucle de procesamiento de eventos GLUT llamado glutMainLoop.

B.3.1 glutMainLoop

glutMainLoop entra en el bucle de procesamientos de GLUT

uso

```
void glutMainLoop(void);
```

descripción

glutMainLoop hace que GLUT entre en el bucle de procesamiento de eventos Esta rutina deberá ser llamada al menos una vez en un programa GLUT Una vez llamada, esta rutina nunca regresará Llamará cuantas llamadas sean necesarias, que hayan sido registradas

B.4 Manejo de ventanas.

GLUT soporta dos tipos de ventanas: Ventanas de nivel superior y sub-ventanas Ambos tipos soportan renderización GLUT y llamadas GLUT. Existe un solo espacio de identificadores para ambos tipos de ventanas

B.4.1 glutCreateWindow

`glutCreateWindow` crea una ventana de nivel superior.

uso

```
int glutCreateWindow(char *name);
```

`name` Cadena ASCII a ser usada como nombre de ventana

descripción

`glutCreateWindow` crea una ventana de nivel superior `name` será provisto al sistema de ventanas como el nombre de la ventana

Implícitamente, la *ventana actual* se establece como la nueva ventana creada

Cada ventana creada tiene un contexto OpenGL asociado único Cambios al estado de un contexto OpenGL asociado a una ventana pueden ser hechos inmediatamente después de que la ventana es creada.

El *estado de despliegue* de una ventana es inicialmente que la ventana se muestre Pero el *estado de despliegue* de la ventana no se ejecuta realmente hasta que se entra en `glutMainLoop` Esto significa que hasta que se llame a `glutMainLoop`, renderizar a una ventana recién creada no es efectivo porque la ventana aún no puede ser desplegada

El valor regresado es un identificador entero pequeño para la ventana El rango de identificadores alojados comienza en uno Este identificador de ventana puede ser usado cuando se llame a `glutSetWindow`.

Notas de Implementación en X

Las propiedades de nivel superior apropiadas son establecidas, según el Manual de Convenciones de Comunicaciones Inter-Clientes X (ICCCM). La propiedad WM_COMMAND que lista la línea de comandos usada para invocar al programa GLUT se establece sólo para la primera ventana creada

B.4.2 glutCreateSubWindow

`glutCreateSubWindow` crea una sub-ventana

uso

```
int glutCreateSubWindow(int win, int x, int y, int width, int
                                    height);
```

`win`	Identificador de la ventana padre de la sub-ventana
`x`	Posición en pixeles en X de la ventana relativa al origen de la ventana padre
`y`	Posición en pixeles en Y de la ventana relativa al origen de la ventana padre
`width`	Ancho en pixeles
`height`	Altura en pixeles

descripción

`glutCreateSubWindow` crea una sub-ventana de la ventana identificada por `win` de ancho `width` y altura `height` en la ubicación `x` y `y` dentro de la *ventana actual* Implícitamente, la *ventana actual* se establece como la nueva sub-ventana creada

Cada ventana creada tiene un contexto OpenGL asociado Los cambios de estado a un contexto OpenGL asociado pueden ser hechos inmediatamente después de que la ventana es creada

El *estado de despliegue* de una ventana es inicialmente que la ventana sea mostrada, pero el *estado de despliegue* de la ventana no es ejecutado hasta que se entra en `glutMainLoop` Esto significa que hasta que se llame a `glutMainLoop`, renderizar a una ventana recién creada no es efectivo porque la ventana aún no puede ser desplegada Las sub-ventanas no pueden ser iconificadas

Las sub-ventanas pueden ser anidadas con profundidad arbitraria

El valor regresado es un identificador entero pequeño para la ventana El rango de identificadores alojados comienza en uno

B.4.3 glutSetWindow, glutGetWindow

`glutSetWindow` establece la *ventana actual*; `glutGetWindow` regresa el identificador de la *ventana actual*

uso

```
void glutSetWindow(int win);
```

```
int glutGetWindow(void);
```

`win` Identificador de la ventana GLUT que será la *ventana actual*.

descripción

`glutSetWindow` establece la *ventana actual*; `glutGetWindow` regresa el identificador de la *ventana actual* Si no existen ventanas, o la *ventana actual* previa fue destruida, `glutGetWindow` regresa cero `glutSetWindow` no cambia la capa en uso para la ventana Eso se realiza usando `glutUseLayer`

B.4.4 glutDestroyWindow

`glutDestroyWindow` destruye a la ventana especificada

uso

```
void glutDestroyWindow(int win);
```

`win` Identificador de la ventana GLUT a destruir

descripción

`glutDestroyWindow` destruye la ventana especificada por `win` y el contexto OpenGL asociado a la ventana, mapa de colores lógico (si la ventana es índice de color) y la capa y estado relacionado (si una capa ha sido establecida) Cualesquiera sub-ventanas de ventanas destruidas también son destruidas por `glutDestroyWindow` Si `win` era la *ventana actual*, la *ventana actual* es inválida (`glutGetWindow` regresará cero)

B.4.5 glutPostRedisplay

`glutPostRedisplay` marca a la *ventana actual* como "necesita ser redesplegada"

uso

```
void glutPostRedisplay(void);
```

descripción

Marca el plano normal de la *ventana actual* como "necesita volverse a desplegar" En la siguiente iteración a través de `glutMainLoop`, la llamada de despliegue de la ventana será llamada para volver a desplegar el plano normal de la ventana Múltiples llamadas a `glutPostRedisplay` antes de la siguiente oportunidad de llamada de

despliegue genera una sola llamada de redespliegue `glutPostRedisplay` puede ser llamado dentro de una llamada de despliegue de la ventana o capa para volver a marcar a la ventana para ser vuelta a desplegar.

Localmente, la notificación de daño al plano normal de una ventana es tratado como un `glutPostRedisplay` en la ventana dañada. A diferencia del daño reportado por el sistema de ventanas, `glutPostRedisplay` no establecerá como verdadero el estado de daño del plano normal (regresado por `glutLayerGet(GLUT_NORMAL_DAMAGED)`.

B.4.6 glutSwapBuffers

`glutSwapBuffers` intercambia los búfers de la *ventana actual* si ésta tiene doble búfer.

uso

```
void glutSwapBuffers(void);
```

descripción

Realiza un intercambio de búferes en la *capa en uso* para la *ventana actual*. Específicamente, `glutSwapBuffers` promueve el contenido del búfer trasero de la *capa en uso* de la *ventana actual* para que se convierta en el contenido del búfer frontal. El contenido del búfer trasero, se vuelve indefinido. La actualización típicamente toma lugar durante el retrazado vertical del monitor, en vez de tomar lugar inmediatamente después de que `glutSwapBuffers` es llamado.

Un `glFlush` implícito es hecho por `glutSwapBuffers` antes de que regrese. Comandos subsecuentes de OpenGL pueden ser solicitados inmediatamente después de llamar a `glutSwapBuffers`, pero no son ejecutados hasta que el intercambio de búferes sea completado.

Si la *capa en uso* no tiene doble búfer, `glutSwapBuffers` no tiene efecto.

B.4.7 glutPositionWindow

`glutPositionWindow` solicita un cambio de posición de la *ventana actual*.

uso

```
void glutPositionWindow(int x, inty);
```

x Nueva posición de la ventana en X en pixeles

y Nueva posición de la ventana en Y en pixeles

descripción

`glutPositionWindow` solicita un cambio en la posición de la *ventana actual*. Para ventanas de nivel superior, los parámetros x y y son desplazamientos de pixel desde el origen de la pantalla. Para sub-ventanas, los parámetros x y y son desplazamientos de pixel desde el origen de la ventana padre.

Las solicitudes hechas por `glutPositionWindow` no son procesadas inmediatamente. La solicitud se ejecuta después de regresar al bucle principal de eventos. Esto permite que múltiples solicitudes de `glutPositionWindow`, `glutReshapeWindow` y `glutFullScreen` a la misma ventana se incorporen.

En el caso de ventanas de nivel superior, una llamada a `glutPositionWindow` es considerada sólo como una solicitud de posicionar la ventana. El sistema de ventanas es libre de aplicar sus propias políticas para la ubicación de la ventana de nivel superior. El designio es que las ventana de nivel superior deban ser reposicionadas de acuerdo con los parámetros de `glutPositionWindow`.

`glutPositionWindow` desactiva el estado de pantalla completa de una ventana si estaba previamente activado.

B.4.8 glutReshapeWindow

`glutReshapeWindow` solicita un cambio al tamaño de la *ventana actual*.

uso

```
void glutReshapeWindow(int width, int height);
```

`width`	Nuevo ancho de la ventana en pixeles
`height`	Nueva altura de la ventana en pixeles

descripción

`glutReshapeWindow` solicita un cambio en el tamaño de la *ventana actual*. Los parámetros de `ancho` y `alto` son extensiones de tamaño en pixeles. El `ancho` y el `alto` deben tener valores positivos.

Las solicitudes de `glutReshapeWindow` no son procesadas inmediatamente. La solicitud se ejecuta después de regresar al bucle principal de eventos. Esto permite que solicitudes múltiples de `glutReshapeWindow`, `glutPostionWindow` y `glutFullScreen` a la misma ventana, se incorporen.

En el caso de ventanas de nivel superior, una llamada a `glutReshapeWindow` es considerada sólo como una solicitud para cambiar el tamaño de la ventana El sistema de ventanas es libre de aplicar sus propias políticas para el cambio de tamaño de ventanas El propósito es que las ventanas de nivel superior sean cambiadas de tamaño de acuerdo con los parámetros de `glutReshapeWindow` En caso de que un cambio de tamaño realmente surja efecto, las nuevas dimensiones son reportadas al programa por una llamada de cambio de forma

`glutReshapeWindow` desactiva el estado de pantalla completa de una ventana, si éste estaba activo

B.4.9 glutFullScreen

`glutFullScreen` solicita que la *ventana actual* sea de pantalla completa

uso

```
void glutFullScreen(void);
```

descripción

`glutFullScreen` solicita que la *ventana actual* sea de pantalla completa La semántica exacta de qué significa pantalla completa puede variar entre sistemas de ventanas. El propósito es hacer la ventana tan grande como sea posible y desactivar cualesquiera decoraciones o bordes añadidos a la ventana por el sistema de ventanas No se garantiza que el ancho y el alto de la ventana sean iguales que el ancho y el alto de la pantalla, pero es el intento de hacer una ventana de pantalla completa

`glutFullScreen` ha sido definido para funcionar sólo en ventanas de nivel superior

Las solicitudes de `glutFullScreen` no son procesadas inmediatamente La solicitud es ejecutada después de regresar al bucle de eventos principal Esto permite que múltiples solicitudes de `glutReshapeWindow`, `glutPositionWindow` y `glutFullScreen` sean incorporadas

Solicitudes subsecuentes de `glutReshapeWindow` y `glutPositionWindow` a la ventana desactivarán el estado de pantalla completa de la ventana

Notas de Implementación en X

En la implementación en X de GLUT, la pantalla completa se implementa al cambiar el tamaño y posicionar la ventana de tal forma que cubra la pantalla completa, y se coloca la propiedad `_MOTIF_WM_HINTS` de tal forma que no se solicite absolutamente ninguna

decoración Otros manejadores de ventanas distintos, o no compatibles con Motif pueden no responder a `_MOTIF_WM_HINIS`

B.4.10 glutPopWindow, glutPushWindow

`glutPopWindow` y `glutPushWindow` cambian el orden de la *ventana actual* relativo a sus hermanas

uso

```
void glutPopWindow(void);
void glutPushWindow(void);
```

descripción

`glutPopWindow` y `glutPushWindow` trabajan tanto en ventanas de nivel superior como sub-ventanas El efecto de meter y sacar ventanas no toma lugar inmediatamente En vez de esto, el meter o sacar es guardado para ejecución cuando se regrese al evento bucle de GLUT Solicitudes subsecuentes de meter o sacar en una ventana remplazan las solicitudes para esa ventana El efecto de meter y sacar ventanas de nivel superior está sujeto a la política del sistema para reacomodar ventanas

B.4.11 glutShowWindow, glutHideWindow, glutIconifyWindow

`glutShowWindow` `glutHideWindow` y `glutIconifyWindow` cambian el estado de despliegue de la *ventana actual*

uso

```
void glutShowWindow(void);
void glutHideWindow(void);
void glutIconifyWindow(void);
```

descripción

`glutShowWindow` mostrará la *ventana actual* (aunque podría permanecer aún no visible si está obstruida por otras ventanas siendo mostradas) `glutHideWindow` esconderá a la *ventana actual* `glutIconifyWindow` iconizará a una ventana de nivel superior, pero GLUT prohíbe la iconificación e una sub-ventana El efecto de mostrar, esconder e iconizar ventanas no toma lugar inmediatamente En vez de esto, las solicitudes son guardadas para ser ejecutadas cuando se regrese al bucle de eventos de GLUT Solicitudes subsecuentes de mostrar, esconder o iconizar una ventana remplazan las solicitudes anteriormente guardadas para dicha ventana. El efecto de esconder, mostrar o iconizar ventanas de nivel superior está sujeto a las políticas del sistema para desplegar ventanas

B.4.12 glutSetWindowTitle, glutSetIconTitle

`glutSetWindowTitle` y `glutSetIconTitle` cambian el título de la ventana o del ícono respectivamente de la ventana actual de nivel superior

uso

```
void glutSetWindowTitle (char *nombre);
void glutSetIconTitle (char *nombre);
```

`nombre` es una cadena de caracteres ASCII para el nombre de la ventana o el ícono

descripción

Estas rutinas deberán ser llamadas sólo cuando la *ventana actual* sea una ventana de nivel superior Con la creación de una ventana de nivel superior, los nombre de ícono y ventana se determinan con el parámetro `nombre` en `glutCreateWindow`. Una vez creada, `glutSetWindowTitle` y `glutSetIconTitle` pueden cambiar los nombres de ventana o de ícono respectivamente de las ventanas de nivel superior Cada llamada le solicita al sistema de ventanas que cambie el título apropiadamente Los requerimientos no se almacenan en el buffer ni son fusionados La política según la cual el nombre de la ventana o el ícono son desplegados, es dependiente del sistema

B.4.13 glutSetCursor

`glutSetCursor` cambia la imagen del cursor para la *ventana actual*.

uso

```
void glutSetCursor (int cursor);
```

`cursor` nombre de la imagen del cursor a cambiar.

GLUT_CURSOR_RIGHT_ARROW	Flecha apuntando hacia arriba y a la derecha.
GLUT_CURSOR_LEFT_ARROW	Flecha apuntando hacia arriba y a la izquierda
GLUT_CURSOR_INFO	Mano que apunta
GLUT_CURSOR_DESTROY	Calavera y huesos en cruz
GLUT_CURSOR_HELP	Signo de interrogación
GLUT_CURSOR_CYCLE	Flechas rotando en un círculo
GLUT_CURSOR_SPRAY	Spray
GLUT_CURSOR_WAIT	Reloj de pulsera
GLUT_CURSOR_TEXT	Punto de inserción para el texto
GLUT_CURSOR_CROSSHAIR	Cruz simple

GLUT_CURSOR_UP_DOWN	Bidireccional apuntando hacia arriba y hacia abajo
GLUT_CURSOR_LEFT_RIGHT	Bidireccional apuntando hacia la izq y hacia la der.
GLUT_CURSOR_TOP_SIDE	Flecha apuntando al lado superior
GLUT_CURSOR_BOTTOM_SIDE	Flecha apuntando al lado inferior
GLUT_CURSOR_LEFT_SIDE	Flecha apuntando al lado izquierdo
GLUT_CURSOR_RIGHT_SIDE	Flecha apuntando al lado derecho
GLUT_CURSOR_TOP_LEFT_CORNER	Flecha apuntando a la esquina superior izquierda
GLUT_CURSOR_TOP_RIGHT_CORNER	Flecha apuntando a la esquina superior derecha
GLUT_CURSOR_BOTTOM_LEFT_CORNER	Flecha apuntando a la esquina inferior izquierda
GLUT_CURSOR_BOTTOM_RIGHT_CORNER	Flecha apuntando a la esquina inferior derecha
GLUT_CURSOR_FULL_CROSSHAIR	Cursor de cruz de pantalla completa (de ser posible, si no, GLUT_CURSOR_CROSSHAIR)
GLUT_CURSOS_NONE	Cursor invisible
GLUT_CURSOR_INHERIT	Usar el cursor de la ventana padre.

descripción

glutSetCursor cambia la imagen del cursor de la ventana actual Cada llamada le pide al sistema de ventanas que cambie el cursor apropiadamente La imagen del cursor cuando se crea una ventana es GLUT_CURSOR_INHERIT

Las imágenes exactas de los cursores utilizados son dependientes de la implementación La intención es que la imagen coincida con el significado del nombre del cursor Para una ventana de nivel superior, GLUT_CURSOR_INHERIT utiliza el cursor por omisión del sistema de ventanas

B.5 Manejo de revestimientos

Cuando se dispone de hardware de manejo de revestimientos, GLUT provee un conjunto de rutinas para establecer, usar y remover una capa para las ventanas GLUT. Cuando se establece un revestimiento, se genera un contexto separado de OpenGL El estado OpenGL del revestimiento de la ventana es diferente del estado OpenGL de los planos normales

B.5.1 glutEstablishOverlay

`glutEstablishOverlay` establece un revestimiento (de ser posible) para la *ventana actual*

uso

```
void glutEstablishOverlay(void);
```

descripción

`glutEstablishOverlay` establece un revestimiento para la *ventana actual.* El modo requerido de despliegue para el revestimiento se determina por el m*odo inicial de despliegue*

`glutLayerGet(GLUT_OVERLAY_POSSIBLE)` puede ser llamado para determinar si un revestimiento es posible para la *ventana actual* con el *modo actual de despliegue inicial* No debe intentarse establecer un revestimiento cuando no es posible; GLUT terminará al programa

Si `glutEstablishOverlay` es llamado cuando un revestimiento ya existe, el revestimiento existente primero es removido, y entonces uno nuevo es establecido El estado del contexto OpenGL del revestimiento anterior se descarta y se muestra el estado inicial de despliegue de un revestimiento, aunque éste sólo es mostrado si está siendo mostrada la ventana revestida

Implícitamente, la *capa de la ventana en uso* se cambia al revestimiento inmediatamente después de que éste se establece

B.5.2 glutUseLayer

`glutUseLayer` cambia la capa en uso para la ventana actual

uso

```
void glutUseLayer(GLenum layer);
```

`layer` puede ser ya sea `GLUT_NORMAL` o `GLUT_OVERLAY`, seleccionando el plano normal, o el revestimiento, respectivamente.

descripción

`glutUseLayer` cambia la *capa de la ventana en uso* para la *ventana actual* seleccionando ya sea el plano normal o el revestimiento Sólo deberá especificarse revestimiento si existe uno Sin embargo, las ventanas sin un revestimiento pueden llamar a `glutUseLayer(GLUT_NORMAL)` Los comandos OpenGL para la ventana son dirigidos a la capa actual en uso

Para inquirir la capa en uso, llámese a `glutLayerGet(GLUT_LAYER_IN_USE)`

B.5.3 glutRemoveOverlay

`glutRemoveOverlay` remueve el revestimiento (si existe) de la *ventana actual*

uso

```
void glutRemoveOverlay(void);
```

descripción

`glutRemoveOverlay` remueve el revestimiento (de existir éste) Es seguro llamar a `glutRemoveOverlay` aún si no se ha establecido ningún revestimiento (no hace nada en este caso) Implícitamente, la *capa de la ventana en uso* cambia al plano normal una vez que se ha removido el revestimiento

Si el programa requiere reestablecer después el revestimiento, es típicamente más rápido y consume menos recursos utilizar `glutHideOverlay` y `glutShowOverlay` para simplemente cambiar el estado de despliegue del revestimiento

B.5.4 glutPostOverlayRedisplay

`glutPostOverlayRedisplay` marca el revestimiento de la *ventana actual* para que se vuelva a desplegar

uso

```
void glutPostOverlayRedisplay(void);
```

descripción

Marca el revestimiento de la *ventana actual* para que se vuelva a desplegar La siguiente iteración a través de `glutMainLoop`, la llamada del revestimiento de la ventana (o simplemente la llamada de despliegue si no se ha registrado ninguna llamada de despliegue del revestimiento) será llamada para volver a desplegar el plano del revestimiento de la ventana Llamadas múltiples a `glutPostPOverlayRedisplay` antes de la siguiente oportunidad de llamada a despliegue (llamada a despliegue de revestimiento, si hay alguna registrada) generan un solo redespliegue `glutPostOverlayRedisplay` puede ser llamado dentro de una llamada a despliegue de ventana o llamada a despliegue de revestimiento para volver a marcar esa ventana para redespliegue

Lógicamente, la notificación del daño de revestimiento par una ventana es tratado como un `glutPostOverlayRedisplay` en la ventana dañada A diferencia del daño reportado por el sistema de ventanas, `glutPostOverlayRedisplay` no establecerá como verdadero el estado de daño del revestimiento (regresado por `glutLayerGet(GLUT_OVERLAY_DAMAGED)`

Ver además, `glutPostRedisplay`

B.5.5 glutShowOverlay, glutHideOverlay

`glutShowOverlay` muestra el revestimiento de la ventana actual; `glutHideOverlay` esconde el revestimiento

uso

```
void glutShowOverlay(void);
void glutHideOverlay(void);
```

descripción

`glutShowOverlay` muestra el revestimiento de la *ventana actual*; `glutHideOverlay` esconde el revestimiento El efecto de mostrar o esconder el revestimiento toma lugar inmediatamente. Note que `glutShowOverlay` no desplegará el revestimiento a menos que la ventana esté también siendo mostrada (e incluso una ventana siendo mostrada puede ser obstruida por otras ventanas, por lo tanto obstruyendo el revestimiento) Es típicamente más rápido y consume menos recursos utilizar estas rutinas para controlar el estado de despliegue de un revestimiento en oposición a remover y reestablecer el revestimiento

B.6 Manejo de Menús

GLUT soporta menús simples emergentes de cascada Están designados para permitir al usuario seleccionar diversos modos dentro de un programa La funcionalidad es simple y

minimalista, y fue diseñada de esta forma No debe malinterpretarse la funcionalidad de GLUT de menús emergentes con un intento de crear una interfaz con todas las características

Es ilegal crear o destruir menús, o cambiar, añadir o remover elementos de menú mientras un menú (y cualesquiera submenús en cascada) están siendo utilizados (es decir, emergidos)

B.6.1 glutCreateMenu

`glutCreateMenu` crea un nuevo menú emergente

uso

```
int glutCreateMenu(void (*func)(int value));
```

`func` la llamada de retorno para el menú que es llamado cuando una entrada del menú es seleccionada. El valor pasado a la llamada de retorno está determinada por el valor para la entrada del menú seleccionada

descripción

`glutCreateMenu` crea un nuevo menú emergente y regresa un identificador entero pequeño único El rango de identificadores alojados comienza en uno El rango de identificadores de menú está separado del rango de identificadores de ventana Implícitamente, el *menú actual* se convierte en el menú recién creado Este identificador de menú puede ser utilizado cuando se llame a `glutSetMenu`

Cuando la llamada de retorno del menú es llamada debido a que se selecciona una entrada del menú, el menú actual será establecido implícitamente al menú con la entrada seleccionada, antes de que la llamada de retorno sea hecha

B.6.2 glutSetMenu, glutGetMenu

`glutSetMenu` establece el menú actual; `glutGetMenu` regresa el identificador del *menú actual*

uso

```
void glutSetMenu(int menu);
int glutGetMenu(void);
```

`menu` el identificador del menú que será el *menú actual*

descripción

`glutSetMenu` establece el *menú actual*; `glutGetMenu` regresa el identificador del *menú actual*. Si no existen menús, o el contenido previo del menú actual fue destruido, `glutGetMenu` regresa cero.

B.6.3 glutDestroyMenu

`glutDestroyMenu` destruye el menú especificado.

uso

```
void glutDestroyMenu(int menu);
```

GLUT El identificador del menú a dstruir

descripción

`glutDestroyMenu` destruye el menú especificado por `menu`. Si `menu` era el menú actual, el *menú actual* se invalida y `glutGetMenu` regresará cero.

Cuando se destruye un menú, no tiene efecto en cualesquiera submenús para los cuales el menú destruido tiene detonadores (*triggers*).

Los detonadores de submenú se utilizan por nombre, no por referencia.

B.6.4 glutAddMenuEntry

`glutAddMenuEntry` añade una entrada al final del menú actual.

uso

```
void glutAddMenuEntry(char *name, int value);
```

`name` Cadena de caracteres ASCII para desplegar la entrada del menú.
`value` Valor a enviar para la llamada de retorno si se selecciona la entrada del menú.

descripción

`glutAddMenuEntry` añade una entrada al final del *menú actual*. La cadena de caracteres será desplegada para la recién añadida entrada del menú. Si la entrada del menú es seleccionada por el usuario, la llamada de retorno del menú será llamada pasando `value` como el parámetro de la llamada de retorno.

B.6.5 glutAddSubMenu

`glutAddSubMenu` añade un detonador de submenú al final del menú actual

uso

```
void glutAddSubMenu(char *name, int menu);
```

`name` Cadena de caracteres ASCII que se mostrará en el elemento de menú para el cual aparecerá en cascada el submenú
`menu` Identificador del menú a aparecer en cascada desde este elemento del submenú

descripción

`glutAddSubMenu` añade un detonador de submenú al final del menú actual El nombre de la cadena se desplegará para el detonador de submenú recién añadido Si el detonador de submenú es activado, el submenú aparecerá en cascada, permitiendo que los elementos del submenú sean seleccionados

B.6.6 glutChangeToMenuEntry

`glutChangeToMenuEntry` cambia el elemento de menú especificado en el menú actual a una entrada de menú

uso

```
void glutChangeToMenuEntry(int entry, char *name, int value);
```

`entry` Índice dentro de los elementos de menú del menú actual (1 es el elemento superior de menú)
`name` Cadena de caracteres ASCII que será mostrada en la entrada de menú
`value` Valor a enviar para la función de la llamada de retorno si se selecciona la entrada de menú

descripción

`glutChangeToMenuEntry` cambia el elemento de menú especificado a una entrada de menú El parámetro `entry` determina qué elemento de menú deberá ser cambiado, siendo 1 el elemento superior `entry` debe estar entre 1 y `gluGet(GLUT_MENU_NUM_ITEMS)` inclusive El elemento de menú a cambiar no deberá ser una entrada de menú El nombre de la cadena será mostrado para la entrada de menú recién creada El valor será regresado a la llamada de retorno del menú si esta entrada de menú es seleccionada

B.6.7 glutChangeToSubMenu

`glutChangeToSubMenu` cambia el elemento especificado del menú en el *menú actual* a un detonador de submenú

uso

```
void GLUIChangeToSubMenu(int entry, char *name, int menu);
```

`entry` Índice dentro de los elementos de menú del *menú actual* (1 es el elemento superior)
`name` Cadena de caracteres ASCII que se mostrará en el elemento de menú desde el cual aparecerá en cascada el submenú
`menu` identificador del menú que aparecerá en cascada desde este elemento de menú

descripción

`glutChangeToSubMenu` cambia el elemento de menú especificado en el *menú actual* a un detonador de submenú El parámetro `entry` determina qué elemento de menú será cambiado, con 1 siendo el elemento superior `entry` debe estar entre 1 y `glutGet(GLUT_MENU_NUM_ITEMS)` inclusive El elemento de menú a cambiar no tiene que ser ya un detonador de menú La cadena `name` será mostrada para el detonador de submenú recientemente creado El identificador `menu` nombra al submenú que aparecerá en cascada desde el recién añadido detonador de menú

B.6.8 glutRemoveMenuItem

`glutremoveMenuItem` remueve el elemento de menú especificado

uso

```
void glutremoveMenuItem(int entry);
```

`entry` Índice dentro de los elementos de menú del menú actual (1 es el elemento superior de menú)

descripción

`glutRemoveMenuItem` remueve el elemento de menú sin considerar si se trata de una entrada de menú o un detonador de submenú `entry` debe estar entre 1 y `gluGet(GLUT_MENU_NUM_ITEMS)` inclusive Los elementos de menú por debajo del menú removido se renumeran

B.6.9 glutAttachMenu, glutDetachMenu

`glutAttachMenu` acopla un botón del ratón para la ventana actual al identificador del menú actual;

`glutDetachMenu` desacopla un botón del ratón acoplado para la ventana actual

uso

```
void glutAttachMenu(int button);
void glutDetachMenu(int button);
```

`button` El botón al cual acoplar o desacoplar un menú

descripción

`glutAttachMenu` acopla un botón del ratón para la *ventana actual* al identificador del *menú actual*;

`glutDetachMenu` desacopla un botón del ratón acoplado de la *ventana actual* Al acoplar un identificador de menú a un botón, el menú nombrado emergerá cuando el usuario presione el botón especificado `button` deberá ser alguno de `GLUT_LEFT_BUTTON`, `GLUI_MIDDLE_BUTTON`, y `GLUT_RIGHT_BUITON` Note que el menú es acoplado al botón por identificador, no por referencia

B.7 Registro de Llamadas de retorno

GLUT soporta diversas llamadas de retorno (*callbacks*) para responder a eventos Existen tres tipos de llamadas de retorno: de ventana, de menú y globales Las llamadas de retorno de ventanas indican cuándo volver a desplegar o redimensionar una ventana, cuándo cambia la visibilidad de la ventana y cuándo está disponible la entrada de datos a la ventana La llamada de retorno de menú se establece por medio de la llamada `glutCreateMenu` descrita anteriormente. Las llamadas de retorno globales manejan el paso del tiempo y la utilización de menús El orden de llamado de las llamadas de retorno entre distintas ventanas no está definido

Las llamadas de retorno para los eventos de entrada deberán ser entregadas a la ventana en la cual ocurre el evento Los eventos no deberán propagarse a ventanas padre

B.7.1 glutDisplayFunc

`glutDisplayFunc` establece la llamada de despliegue para la ventana actual

uso

```
void glutdisplayFUnc(void (*func)(void));
```
`func` La nueva función de llamada de retorno de despliegue.

descripción

`glutDisplayFunc` establece la llamada de retorno de despliegue para la *ventana actual* Cuando GLUT determina que el plano normal para la ventana necesita volver a ser desplegado, se llama a la llamada de retorno de despliegue para la ventana Antes de la llamada de retorno, la *ventana actual* se establece como la ventana que necesita ser redesplegada y (si no existe llamada de retorno de despliegue de revestimiento registrada) el plano normal se convierte en la *capa en uso* La llamada de retorno de despliegue es llamada sin parámetros. La región entera del plano normal deberá ser desplegada de nuevo en respuesta a la llamada de retorno (esto incluye búfers auxiliares si el programa depende de su estado)

GLUT determina cuándo deberá ser detonada la llamada de retorno de despliegue basada en el estado de redespliegue de la ventana El estado de redespliegue para una ventana puede ser establecido ya sea explícitamente al llamar `glutPostRedisplay` o implícitamente como el resultado de daño a la ventana reportado por el sistema de ventanas. Los redespliegues múltiples anunciados para una ventana se conjuntan por GLUT para minimizar el número de llamadas de retorno realizadas

Cuando un revestimiento se establece para una ventana, pero no hay llamada de retorno de despliegue de revestimiento, la llamada de retorno de despliegue se utiliza para volver a desplegar tanto el revestimiento como el plan normal Esto es, será llamada para el estado de redespliegue o para el estado de redespliegue del revestimiento En este caso, la capa en uso no se cambia implícitamente al entrar a la llamada de retorno de despliegue

Vea la función `glutOverlayDisplayFunc` para entender cómo pueden establecerse distintas llamadas para un revestimiento y un plano normal.

cuando una ventana se crea, no existe llamada de despliegue para esta ventana Es responsabilidad del programador instalar una llamada de despliegue para la ventana antes de que se muestre la ventana Una llamada de retorno de despliegue debe registrarse para cada ventana que se muestra Si una ventana se despliega sin una llamada de retorno de despliegue que haya sido previamente registrada, ocurre un error fatal. El pasarle NULL a `glutDisplayFunc` es ilegal a partir de GLUT 3.0; no hay forma de "desregistrar" una llamada de despliegue (aunque siempre puede ser registrada otra rutina de llamada de retorno) Cuando se regresa de la llamada de retorno de despliegue, el estado de daño de la ventana (regresado al llamar `glutLayerGet(GLUI_NORMAL_DAMAGED)` es borrado Si no hay llamada de retorno de despliegue de revestimiento, el estado de daño de la ventana (regresado al llamar `glutLayerGet(GLUT_OVERLAY_DAMAGED)` también es borrado

B.7.2 glutOverlayDisplayFunc

`glutOverlaydisplayFunc` establece la llamada de retorno de despliegue para la *ventana actual*.

uso

```
void glutOverlaydisplayfunc(void (*func)(void));
```
`func` es la nueva función de llamada de despliegue de revestimiento.

descripción

`glutDisplayFunc` establece la llamada de retorno de despliegue para la *ventana actual*. La llamada de retorno de despliegue de revestimiento es funcionalmente la misma que la llamada de retorno de despliegue excepto que la llamada de retorno de despliegue de revestimiento se utiliza para volver a desplegar el revestimiento de la ventana

Cuando GLUT determina que el plano de revestimiento para la ventana necesita ser redesplegado, se llama a la llamada de retorno de despliegue de revestimiento de la ventana. Antes de la llamada de retorno, la *ventana actual* se establece como la ventana que necesita ser vuelta a desplegar y el revestimiento se convierte en la *capa en uso*. La llamada de despliegue de revestimiento se llama sin parámetros. La región de revestimiento completa debe ser vuelta a desplegar en respuesta a la llamada de retorno (esto incluye a los búferes auxiliares si el programa depende de su estado)

GLUT determina cuándo debe ser detonada la llamada de retorno de despliegue de revestimiento basándose en el estado de redespliegue de revestimiento de la ventana. Éste puede ser establecido explícitamente al llamar `glutPostOverlayRedisplay` o implícitamente como el resultado de daño de la ventana reportado por el sistema de ventanas. Solicitudes de redespliegue de revestimiento múltiples para una ventana se reúnen en una sola por GLUT para minimizar el número de llamadas de retorno de despliegue de revestimiento.

Cuando se regresa de la llamada de retorno de despliegue del revestimiento, el estado de daño de la ventana (regresado al llamar `glutLayerGet(GLUI_OVERLAY_DAMAGED)`) se borra.

La llamada de retorno de despliegue de revestimiento puede ser desregistrada al llamar a `glutOverlaydisplayFunc` con parámetro `NULL`. La llamada de retorno de despliegue de revestimiento se establece como `NULL` cuando se crea un revestimiento. Véase `glutDisplayFunc` para entender cómo se usa la llamada de retorno de despliegue de manera única si no existe una llamada de retorno de despliegue de revestimiento registrada.

B.7.3 glutReshapeFunc

`glutReshapeFunc` establece la llamada de retorno de cambio de tamaño para la ventana actual

uso

```
void glutReshapeFunc(void (*func)(int width, int height));
```

`func` La nueva función de llamada de retorno de cambio de tamaño

descripción

`glutReshapeFunc` establece la llamada de retorno de cambio de tamaño para la *ventana actual* La llamada de retorno de cambio de tamaño se detona cuando una ventana cambia de tamaño Una llamada de retorno de cambio de tamaño también se detona inmediatamente antes de la primera llamada de retorno de despliegue de una ventana después de que se crea una ventana o cada vez que se establece un revestimiento para la ventana Los parámetros de `width` y `height` de la llamada de retorno especifican el nuevo tamaño en pixeles Antes de la llamada de retorno, la ventana actual se establece a la ventana que ha cambiado de tamaño

Si la llamada de retorno de cambio de tamaño no está registrada para una ventana o se pasa `NULL` a `glutReshapeFunc` (para desregistrar una llamada de retorno previamente registrada), la llamada de retorno de cambio de tamaño por omisión es utilizada Ésta simplemente llamará a `glViewport(0,0,width,height)` en el plano normal (y en el revestimiento, de existir éste)

Si un revestimiento se establece para la ventana, sólo se genera una llamada de retorno de cambio de tamaño Es responsabilidad de la llamada de retorno actualizar tanto el plano normal y el revestimiento para la ventana (cambiando la capa en uso como sea necesario) Cuando una ventana de nivel superior cambia de forma, las sub-ventanas no cambian de forma Depende del programa GLUT el mantener el tamaño y las posiciones de las sub-ventanas dentro de la ventana de nivel superior Aún así, las llamadas de retorno de cambio de tamaño se detonan para las sub-ventanas cuando su tamaño se cambia usando `glutReshapeWindow`

B.7.4 glutKeyboardFunc

GLUTKeyboardFunc establece la llamada de retorno del teclado para la ventana actual

uso

```
void glutKeyboardFunc(void (*func)(unsigned char key,
                      int x, int y));
```

`func` la nueva llamada de retorno de teclado

descripción

`glutKeyboardFunc` establece la llamada de retorno para la *ventana actual* Cuando un usuario escribe en la ventana, cada tecla presionada que genere un caracter ASCII generará una llamada de retorno de teclado. El parámetro `key` de la llamada de retorno es el caracter ASCII generado El estado de las teclas modificadoras como `Shift` no puede ser determinado directamente; su único efecto será en los datos ASCII regresados Los parámetros `x` y `y` indican la localización del ratón en las coordenadas relativas a la ventana cuando se presionó la tecla Cuando se crea una nueva ventana, no se registra una llamada de retorno de teclado inicial, y las teclas presionadas ASCII se ignoran `glutKeyboardFunc` con parámetro NULL desactiva la generación de las llamadas de teclado

Durante una llamada retorno de teclado, `glutGetModifiers` puede ser llamada para determinar el estado de las teclas modificadoras cuando se presionó la tecla que originó la llamada de retorno

B.7.5 glutMouseFunc

`glutMouseFunc` establece la llamada de retorno del ratón para la ventana actual

uso

```
void glutMouseFunc(void (*func(int button, iont state,
                        int x int y));
```

`func` La nueva llamada de retorno del ratón

descripción

`glutMouseFunc` establece la llamada de retorno del ratón para la ventana actual. Cuando un usuario presiona y suelta los botones del ratón en la ventana, cada presión y liberación genera una llamada de retorno del ratón El parámetro `button` es ya sea `GLUT_LEFT_BUTTON`, `GLUT_MIDDLE_BUTTON`, o `GLUI_RIGHT_BUIION` Para sistemas con sólo dos botones, podría no ser posible generar la llamada de retorno con `GLUT_MIDDLE_BUTION` Para sistemas con un solo botón, puede ser que sólo se genere una llamada de retorno con `GLUT_LEFT_BUTTON` El parámetro estado es ya sea `GLUI_UP` o `GLUT_DOWN` indicando si la llamada se debe a una presión o liberación del botón, respectivamente Los parámetros `x` y `y` de la llamada de retorno indican las coordenadas relativas a la ventana cuando el estado del botón cambió Si una llamada de retorno con `GLUT_DOWN` para un botón específico se detona, el programa puede asumir que una llamada de retorno con `GLUT_UP` será generada (asumiendo que la ventana todavía tiene una llamada de retorno del ratón registrada) cuando el botón del ratón se libera aún si el ratón se ha movido fuera de la ventana

Si se adjunta un menú a un botón para una ventana, no se generarán llamadas de retorno del ratón para este botón durante la llamada de retorno del ratón, puede llamarse `glutGetModifiers` para determinar el estado de las teclas modificadoras cuando ocurrió un evento del ratón que generó la llamada de retorno

El pasarle `NULL` a `glutMouseFunc` desactiva la generación de llamadas de retorno del ratón.

B.7.6 glutMotionFunc, glutPassiveMotionFunc

`glutMotionFunc` y `glutPassiveMotionFunc` establecen las llamadas de retorno de movimiento y movimiento pasivo, respectivamente, para la ventana actual

uso

```
void glutMotionFunc(void(*func)(int x, int y));
void glutPassiveMotionFunc(void(*func)(int x, int y));
```

`func` la nueva llamada de retorno de movimiento o movimiento pasivo

descripción

`glutMotionFunc` y `glutPassiveMotionFunc` establecen las llamadas de retorno de movimiento y movimiento pasivo respectivamente, para la ventana actual La llamada de retorno de movimiento para una ventana se llama cuando el ratón se mueve dentro de la ventana mientras uno o más botones del ratón están presionados La llamada de retorno de movimiento pasivo se llama cuando el ratón se mueve dentro de la ventana mientras no está presionado ningún botón del ratón

Los parámetros x y y indican la localización del ratón en coordenadas relativas El pasarle `NULL` a `glutMotionFunc` o `glutPassiveMotionFunc` inhabilita la generación de la llamada de retorno de movimiento o de movimiento pasivo

B.7.7 glutVisibilityFunc

`glutvisibilityFunc` establece la llamada de retorno de visibilidad para la ventana actual

uso

```
void glutVisibilityFunc(void(*func)(int state));
```
`func` La nueva función de llamada de retorno de visibilidad

descripción

`glutVisibilityFunc` establece la llamada de retorno de visibilidad para la ventana actual La llamada de retorno de visibilidad para una ventana es llamada cuando la visibilidad de la ventana cambia El parámetro `state` de la llamada de retorno es ya sea `GLUT_NOT_VISIBLE` o `GLUT_VISIBLE`, dependiendo de la visibilidad actual de la ventana `GLUT_VISIBLE` no distingue una ventana siendo totalmente visible de una que lo es sólo parcialmente `GLUT_NOT_VISIBLE` significa que ninguna parte de la ventana es visible Hasta que la visibilidad de la ventana cambia, toda renderización de la ventana es descartada

GLUT considera una ventana como visible si cualquier pixel de la ventana es visible o cualquier pixel de cualquier ventana descendente es visible en la pantalla

El pasarle NULL a `glutVisibilityFunc` inhabilita la generación de la llamada de retorno de visibilidad Si la llamada de retorno de visibilidad para una ventana se inhabilita y posteriormente se vuelve a habilitar, el estado de visibilidad de la ventana no se encontrará definido; cualquier cambio en la visibilidad de la ventana será reportado, de tal forma que si se deshabilita una llamada de retorno y posteriormente vuelve a habilitarse, se garantiza que el siguiente cambio de visibilidad será reportado

B.7.8 glutEntryFunc

`glutEntryFunc` establece la llamada de retorno de entrada / salida del ratón para la *ventana actual*

uso

```
void GLUTEntryFunc(void (*func(int state));
```

`func` La nueva función de llamada de retorno de entrada

descripción

`glutEntryFunc` establece la llamada de retorno de entrada / salida del ratón para la ventana actual El parámetro de estado de la llamada de retorno es ya sea `GLUT_LEFT` o `GLUT_ENTERED` dependiendo si el apuntador del ratón ha salido o entrado a la ventana El pasarle `NULL` a `glutEntryFunc` deshabilita la generación de la llamada de retorno de entrada / salida del ratón Es posible que algunos sistemas de ventanas no generen llamadas de retorno de entrada / salida precisas

B.7.9 glutSpecialFunc

`glutSpecialFunc` establece la llamada de retorno especial de teclado para la ventana actual

uso

```
void glutSpecialFunc(void (*func)(int key, int x, int y));
```

`func` La nueva llamada de retorno especial

descripción

`glutSpecialFunc` establece la llamada de retorno especial de teclado para la ventana actual. Ésta es detonada cuando se presionan las teclas de función o las teclas direccionales El parámetro `key` de la llamada de retorno es una constante `GLUT_KEY_*` para la tecla especial presionada Los parámetros x y y de la llamada de retorno indican las coordenadas relativas a la ventana del ratón cuando se presionó la tecla. Cuando se crea una nueva ventana, no se registra ninguna llamada de retorno especial y las teclas especiales presionadas en la ventana son ignoradas. El pasarle `NULL` a `glutSpecialFunc` deshabilita la generación de llamadas de retorno especiales

Durante una llamada de retorno especial, `glutgetModifiers` puede ser llamado para determinar el estado de las teclas modificadoras cuando al presionar una tecla se genera la llamada de retorno

Una implementación deberá hacer lo mejor para proveer formas de generar todas las teclas especiales `GLUT_KEY_*`. Los valores disponibles de `GLUT_KEY_*` son:

`GLUT_KEY_F1`	La tecla de función F1
`GLUT_KEY_F2`	La tecla de función F2
`GLUT_KEY_F3`	La tecla de función F3
`GLUT_KEY_F4`	La tecla de función F4
`GLUT_KEY_F5`	La tecla de función F5
`GLUT_KEY_F6`	La tecla de función F6
`GLUT_KEY_F7`	La tecla de función F7
`GLUT_KEY_F8`	La tecla de función F8
`GLUT_KEY_F9`	La tecla de función F9
`GLUT_KEY_F10`	La tecla de función F10
`GLUT_KEY_F11`	La tecla de función F11
`GLUT_KEY_F12`	La tecla de función F12
`GLUT_KEY_LEFT`	Tecla direccional izquierda
`GLUT_KEY_UP`	Tecla direccional hacia arriba
`GLUT_KEY_RIGHT`	Tecla direccional derecha
`GLUT_KEY_DOWN`	Tecla direccional hacia abajo
`GLUT_KEY_PAGE UP`	Tecla direccional de regresar página
`GLUT_KEY_PAGE DOWN`	Tecla direccional de avanzar página
`GLUT_KEY_HOME`	Tecla direccional de inicio
`GLUT_KEY_END`	Tecla direccional de fin

GLUT_KEY_INSERT Tecla direccional de inserción

Note que las teclas de escape, retroceso y suprimir son generadas como un caracter ASCII

B.7.10 glutSpaceballMotionFunc

establece la llamada de retorno de movimiento de spaceball para la ventana actual

uso

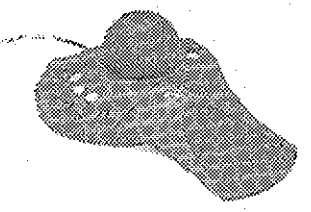
```
void glutSpaceballMotionFunc(void (*func)(int x,
                                          int y, int z));
```

func La nueva función de llamada de retorno de movimiento de spaceball

descripción

glutSpaceballMotionFunc establece la llamada de retorno de movimiento de spaceball para la ventana actual Ésta es llamada cuando la ventana tiene foco de entrada del Spaceball (normalmente, cuando el ratón está en la ventana) y el usuario genera traslaciones de spaceball Los parámetros *x*, *y* y *z* de la llamada de retorno indican las traslaciones a lo largo de los ejes X, Y y Z Los parámetros de la llamada de retorno se normalizan para que se encuentren dentro del rango de –1000 a 1000, inclusive

El registrar una llamada de retorno de spaceball cuando no está disponible un dispositivo de spaceball no tiene efecto y no es un error En este caso, no se generan llamadas de retorno de movimiento de spaceball. El pasarle NULL a glutSpaceballMotionFunc deshabilita la generación de llamadas de retorno de spaceball Cuando una nueva ventana se crea, no se registra inicialmente ninguna llamada de retorno de movimiento de spaceball



[spaceball]

B.7.11 glutSpaceballRotateFunc

`glutSpaceballRotateFunc` establece la llamada de retorno de rotación para la *ventana actual*

uso

```
void glutSpaceballRotateFunc(void (*func)(int x,
                                          int y, int z));
```

`func` La nueva función de llamada de retorno de rotación de spaceball

descripción

`glutSpaceballRotateFunc` establece la llamada de retorno de rotación de spaceball para la ventana actual Ésta es llamada cuando la ventana tiene foco de entrada del spaceball (normalmente, cuando el ratón está dentro de la ventana) y el usuario genera rotaciones con el spaceball Los parámetros de la función de retorno de llamada *x*, *y* y *z* indican la rotación a lo largo de los ejes `X`, `Y` y `Z`. Los parámetros de la llamada de retorno se normalizan dentro del rango de –1800 a 1800 inclusive

Si se registra una llamada de retorno de rotación cuando no está disponible un dispositivo de spaceball, ésta no tendrá efecto y no será un error En este caso, no se generarán llamadas de retorno de rotación

El pasarle `NULL` a `glutSpaceballRotateFunc` desactiva la generación de llamadas de retorno de rotación de spaceball Cuando se crea una nueva ventana, no se registra inicialmente ninguna llamada de retorno de rotación de spaceball

B.7.12 glutSpaceballButtonFunc

`glutSpceballButtonFunc` establece la llamada de retorno de botón del spaceball para la ventana actual Ésta es llamada cuando la ventana tiene el enfoque de entrada (normalmente cuando el ratón está en la ventana) y el usuario genera opresiones de botón del spaceball El parámetro `button` será el número del botón, comenzando en 1 El número de botones disponibles del spaceball puede ser determinado con `glutDeviceGet(GLUI_NUM_SPACEBALL_BUTTONS)` `state` es ya sea `GLUT_UP` o `GLUT_DOWN` indicando si la llamada se debió a una liberación o a una opresión de botón, respectivamente

El registrar una llamada de retorno de botón de spaceball cuando no está disponible un dispositivo de spaceball no tiene efecto y no es un error En este caso no se generarán llamadas de retorno de botón de spaceball

El pasarle NULL a glutSpaceballButtonFunc inhabilita la generación de llamadas de retorno de botón de spaceball Cuando una nueva ventana se crea, no se registra ninguna llamada de retorno de botón de spaceball

B.7.13 glutButtonBoxFunc

glutButtonBoxFunc establece la llamada de retorno de caja de dial y botón para la ventana actual

uso

```
void glutButtonBoxFunc(void (*func)(int button, int state));
```
func la nueva función de llamada de retorno de botón de caja de botón y dial

descripción

glutButtonBoxFunc establece la llamada de retorno de botón de caja de dial y botón para la ventana actual La llamada de retorno de botón de caja de dial y botón para una ventana es llamada cuando la ventana tiene enfoque de entrada para la caja de dial y botón (normalmente, cuando el ratón se encuentra dentro de la ventana) y el usuario genera opresiones de botón en la caja de dial y botón El parámetro botón será el número de botón (comenzando en uno) El numero de botones disponibles en la caja de dial y botón puede determinarse con glutDeviceGet(GLUT_NUM_BUTTON_BOX_BUTTONS) El estado es ya sea GLUT_UP o GLUT_DOWN indicando si la llamada se debió a una liberación o a una opresión, respectivamente

El registrar una llamada de retorno de botón de caja de dial y botón cuando no está disponible este dispositivo no tiene efecto alguno y no es un error En este caso, no se generarán llamadas de retorno de botón de caja de dial y botón El pasarle NULL a glutButtonBoxFunc desactiva la generación de llamadas de retorno de botón de caja de dial y botón Cuando se crea una nueva ventana, no se registra inicialmente ninguna llamada de retorno de botón de caja de dial y botón

B.7.14 glutDialsFunc

glutDialsFunc establece la llamada de retorno de dial de caja de dial y botón para la llamada actual

uso

```
void glutdialsFunc (void (*func)(int dial, int value));
```
func La nueva función de llamada de retorno del dial

descripción

`glutDialsFunc` establece la llamada de retorno de dial de caja de dial y botón para la ventana actual La llamada de retorno de dial de caja de dial y botón para una ventana es llamada cuando la ventana tiene enfoque de entrada de caja de dial y botón (normalmente, cuando el ratón se encuentra en la ventana) y el usuario genera cambios en el dial de la caja de dial y botón El parámetro dial será el número de dial (comenzando en uno) El número de diales disponibles en la caja de dial y botón puede ser determinada con `glutDeviceGet(GLUT_NUM_DIALS)` `value` mide la rotación absoluta en grados Los valores de dial no se enrollan con cada rotación completa, sino que continúan acumulando grados (hasta que el valor de dial se desborda)

El registrar una llamada de retorno de dial de caja de botón y dial cuando no está disponible este dispositivo no tiene efecto y no es un error En este caso, no se generarán llamadas de retorno de dial de caja de botón y dial El pasarle `NULL` a `glutDialsFunc` desactiva la generación de llamadas de retorno de dial de caja de botón y dial Cuando se crea una nueva ventana, no se registra inicialmente ninguna llamada de retorno de dial de caja de botón y dial

B.7.15 glutTabletMotionFunc

`glutTabletMotionFunc` establece la llamada de retorno de movimiento en la tableta digitalizadora para la ventana actual

uso

```
void glutTabletMotionFunc(void (*func)(int x, int y));
```
`func` La nueva función de llamada de retorno de movimiento de tableta

descripción

`glutTabletMotionFunc` establece la llamada de retorno de movimiento de tableta digitalizadora para la ventana actual La llamada de retorno de movimiento de tableta digitalizadora es llamada cuando la ventana tiene enfoque de entrada (normalmente cuando el ratón está en la ventana) y el usuario genera movimiento en la tableta Los parámetros x y y de la llamada de retorno indican la posición absoluta de la opresión ("puc") en la tableta Los parámetros de la llamada de retorno se normalizan de tal forma que se encuentren dentro del rango de 0 a 2000 inclusive

El registrar una llamada de retorno de movimiento de tableta digitalizadora cuando no está disponible dicho dispositivo no tiene efecto y no es un error En este caso, no se generarán llamadas de retorno de movimiento de tableta digitalizadora

el pasarle `NULL` a `glutTabletMotionFunc` inhabilita la generación de llamadas de retorno de movimiento de tableta digitalizadora Cuando se crea una nueva ventana, no

se registra inicialmente ninguna llamada de retorno de movimiento de tableta digitalizadora

B.7.16 glutTabletButtonFunc

`glutTabletButtonFunc` establece la llamada de retorno de botón de tableta digitalizadora para la ventana actual

uso

```
void glutTabletButtonFunc(void (*func)(int button,
                                       int state, int x, int y));
```

`func` la nueva función de llamada de retorno de botón de tableta digitalizadora

descripción

`glutTabletButtonFunc` establece la llamada de retorno de botón de tableta digitalizadora para la *ventana actual* Esta llamada de retorno para una ventana es llamada cuando la ventana tiene enfoque de entrada de la tableta (normalmente, cuando el botón está en la ventana) y el usuario genera opresiones de botón de la tableta digitalizadora El parámetro `button` será el número de botón (comenzando en uno) El número de botones disponibles en la tableta puede ser determinado con `glutDeviceGet(GLUT_NUM_TABLEI_BUTTONS)`. `state` es ya sea `GLUT_UP` o `GLUT_DOWN` indicando si la llamada se debe a una liberación u opresión respectivamente Los parámetros *x* y *y* de la llamada de retorno indican las coordenadas relativas de la ventana cuando cambió el estado del botón

El registrar una llamada de retorno de botón de tableta cuando no está disponible el dispositivo de tableta digitalizadora no tiene efecto alguno y no es un error En este caso, no se generarán llamadas de retorno de botón de tableta digitalizadora.

El pasarle `NULL` a `GLUTTabletButtonFunc` desactiva la generación de llamadas de retorno de botón de tableta digitalizadora Cuando se crea una nueva ventana, no se registra inicialmente ninguna llamada de retorno de botón de tableta digitalizadora

B.7.17 glutMenuStatusFunc

`glutMenuStatusFunc` establece la llamada de retorno de estado de menú global

uso

```
void glutMenuStatusFunc(void (*func)(int status,
                                     int x, int y));
```

```
void glutMenuStateFunc(void (*func)(int status));
```

`func` La nueva función de llamada de retorno de estado de menú nuevo

descripción

`glutMenuStatusFunc` establece la llamada de retorno de estado de menú global de tal forma que un programa GLUT puede determinar si está en uso un menú o no lo está. Una vez que se registra una llamada de retorno de estado de menú, será llamada con el valor `GLUT_MENU_IN_USE` como su parámetro status cuando están en uso por el usuario los menús emergentes, y será llamada con el valor `GLUT_MENU_NOT_IN_USE` cuando ya no lo estén Los parámetros *x* y *y* indican la localización en coordenadas de la ventana de la opresión del botón que causó que el menú fuera usado, o la localización donde el menú fue liberado (puede ser fuera de la ventana) El parámetro `func` nombra a la función de llamada de retorno Otras llamadas continúan operando (excepto llamadas de retorno de movimiento de ratón) cuando los menús emergentes están en uso, de tal forma que la llamada de retorno de estado de menú le permite a u programa detener la animación u otras tareas cuando se están utilizando los menús Los submenús en cascada de un menú emergente inicial no generan llamadas de retorno de estado de menú Hay una llamada de retorno de estado de menú única en GLUT

Cuando se llama a la llamada de retorno de estado de menú, el menú actual será el menú emergente inicial tanto en los casos de `GLUT_MENU_IN_USE` como en el de `GLUT_MENU_NOT_IN_USE` La *ventana actual* será la ventana desde la cual el menú inicial emergió, también en ambos casos

El pasarle `NULL` a `glutMenuStatusFunc` desactiva la generación de llamada de retorno de estado de menú `glutMenuStateFunc` es una versión ya no utilizada de `glutMenuStatusFunc` La única diferencia es que el prototipo de llamada de retorno para `glutMenuStateFunc` no proporciona las coordenadas *x* y *y* adicionales

B.7.18 glutIdleFunc

`glutIdleFunc` establece la llamada de retorno global de inactividad

uso

```
void glutIdleFunc(void (*func)(void));
```

`func` La nueva función de llamada de retorno de inactividad.

descripción

`glutIdleFunc` establece la llamada de retorno global de inactividad como `func`, de tal manera que un programa GLUT puede realizar tareas de procesamiento en segundo

plano, o animación continua cuando no se reciben eventos del sistema en la ventana Si se habilita, la llamada de retorno de inactividad es llamada continuamente cuando no se reciban eventos Esta rutina de llamada de retorno no tiene parámetros La ventana actual y el menú actual no se cambiarán antes de la llamada de retorno de inactividad Los programas con ventanas múltiples y/o menús deberán establecer explícitamente la ventana actual y/o el menú actual y no basarse en su valor actual

La cantidad de cálculos y renderizaje realizado en una llamada de retorno de inactividad deberán ser minimizados para evitar el afectar la respuesta interactiva del programa En general, no deberá renderizarse más de un solo cuadro durante una llamada de retorno de inactividad

El pasarle `NULL` a `glutIdleFunc` desactiva la generación de la llamada de retorno de inactividad

B.7.19 glutTimerFunc

`glutTimerFunc` registra una llamada de retorno de temporizador de tal forma que se detone en un número específico de milisegundos

uso

```
void GLUTTimerFunc(unsigned int msecs,
                   void (*func)(int value),value);
```

`msecs` Número de milisegundos que transcurrirán antes de llamar a la llamada de retorno

`func` La función de llamada de retorno del temporizador

`value` El valor entero a pasar a la llamada de retorno de temporizador

descripción

`glutTimerFunc` registra la llamada de retorno de temporizador `func` de tal forma que se detone en al menos `msec` milisegundos El parámetro `value` de la llamada de retorno de temporizador será el valor del parámetro `value` pasado a `glutTimerFunc` Llamadas de retorno de temporizador múltiples al mismo tiempo o con tiempos que difieren pueden ser registradas simultáneamente

El número de milisegundos es un límite inferior del tiempo que transcurre antes de que se genere la llamada de retorno GLUT intenta proporcionar la llamada de retorno de temporizador tan pronto como sea posible antes de la expiración del intervalo de tiempo de la llamada de retorno No hay soporte para cancelar una llamada de retorno registrada En vez de esto, debe ignorarse la llamada de retorno basada en su parámetro `value` cuando es detonada

B.8 Manejo del índice del mapa de colores.

OpenGL soporta renderizaje tanto RGBA como por índice de color. El modo RGBA es generalmente preferible al de índice de color porque están disponibles más capacidades de renderizaje OpenGL y el índice de color requiere la carga de entradas de un mapa de colores.

Las rutinas de GLUT para manejo de índice de color se utilizan para escribir y leer entradas en un mapa de colores de una ventana. Cada ventana de índice de color GLUT tiene su mapa de colores lógico. El tamaño de un mapa de colores de una ventana puede ser determinado al llamar `glutGet(GLUI_WINDOW_COLORMAP_SIZE)`.

Las ventanas de índice de color GLUT dentro de un programa pueden intentar compartir recursos de mapa de colores al copiar un solo mapa de colores a múltiples ventanas utilizando `glutCopyColormap`. De ser posible, GLUT intentará compartir el mapa de colores actual. El copiar mapas de colores usando `glutCopyColormap` puede permitir potencialmente compartir recursos del mapa de colores físico. Lógicamente, cada ventana tiene su propio mapa de colores, de tal forma que al cambiar un mapa de color copiado de una ventana forzará a la duplicación del mapa de colores. Por esta razón, los programas que manejen índice de colores deberán generalmente cargar un solo mapa de color, copiarlo a todas las ventanas de índice de color dentro del programa, y después no modificar ninguna celda del mapa de colores.

La utilización de múltiples mapas de colores resulta a menudo en problemas con la instalación de mapa de colores donde algunas ventanas se despliegan con un mapa de colores incorrecto debido a limitaciones en los recursos del mapa de colores.

B.8.1 glutSetColor

`glutSetColor` establece el color de una entrada del mapa de colores en la capa en uso de la ventana actual.

uso

```
void glutSetColor (int cell, GLfloat red,
                   GLfloat green, GLfloat blue);
```

`cell`	índice de la celda de color (comenzando en cero)
`red`	intensidad de rojo (entre 0.0 y 1.0 inclusive)
`green`	intensidad de verde (entre 0.0 y 1.0 inclusive)
`blue`	intensidad de azul (entre 0.0 y 1.0 inclusive)

descripción

Establece la entrada de la celda de índice de color en el mapa de colores para el mapa de colores actual lógico de la ventana actual para la capa en uso con el color

especificado por `red`, `green` y `blue` La capa en uso de la ventana actual deberá ser una ventana de índice de color `cell` deberá ser mayor o igual que cero y menor que el número total de entradas del mapa de color para la ventana. Si el mapa de color de la capa en uso fue copiado por referencia, una llamada a `glutSetColor` forzará la duplicación del mapa de colores No debe intentarse establecer el color de un índice transparente de revestimiento

B.8.2 glutGetColor

`glutGetColor` recupera un componente rojo, verde o azul para una entrada dada de índice de mapa de colores para el mapa de colores lógico de la capa en uso para la ventana actual

uso

```
GLfloat glutGetColor (int cell, int component);
```

`cell` índice de la celda de color (comenzando en cero)
`component` `GLUT_RED`, `GLUT_GREEN` o bien `GLUT_BLUE`

descripción

`glutGetColor` recupera un componente rojo, verde o azul para una entrada del mapa de colores dada para el mapa de colores lógico de la ventana actual. La ventana actual deberá ser una ventana de índice de color `cell` deberá ser mayor o igual que cero y menor que el número total de entradas del mapa de colores para la ventana Para índices de color válidos, el valor regresado es un valor de punto flotante entre 0.0 y 1 0 inclusive `glutGetcolor` regresará –1 0 si el índice de color especificado es un índice transparente de revestimiento, menor que cero, o mayor o igual al valor regresado por `glutGet(GLUT_WINDOW_COLORMAP_SIZE)`, si el índice de color es transparente o está fuera del rango válido de índices de color

B.8.3 glutCopyColormap

`glutCopyColormap` copia el mapa de colores lógico para la capa en uso desde una ventana especifica para la *ventana actual*

uso

```
void glutcopyColormap(int win);
```
`win` El identificador de la ventana desde la cual se copiará el mapa de colores lógico

descripción

`glutCopyColormap` copia (flojamente, de ser posible, para promover la compartición) el mapa de colores de una ventana especificada a la capa en uso de la ventana actual. La copia será de plano normal a plano normal, o bien de revestimiento a revestimiento (nunca entre capas distintas) Una vez que se ha copiado el mapa de colores, debe evitarse establecer celdas en el mapa de colores con `glutSetColor` puesto que esto forzará a que se haga una copia real del mapa de colores si es que se había copiado previamente por referencia `glutCopyColormap` deberá ser llamado solamente cuando tanto la ventana actual como la ventana `win` son ventanas de índice de color

B.9 Obtención del estado

GLUT mantiene una cantidad considerable de estado visible al programador Parte de este estado puede ser directamente obtenido

glutGet

`glutGet` obtiene el estado simple de GLUT representado por enteros

uso

```
int glutGet(Glenum state);
```

`state` Nombre del estado a obtener

`GLUT_WINDOW_X`	Coordenada X de localización en pixeles de la ventana actual relativa al origen de la pantalla
`GLUT_WINDOW_Y`	Coordenada Y de localización en pixeles de la ventana actual, relativa al origen de la pantalla
`GLUT_WINDOW_WIDTH`	Ancho en pixeles de la ventana actual
`GLUT_WINDOW_HEIGHT`	Alto en pixeles de la ventana actual
`GLUT_WINDOW_BUFFER_SIZE`	Número total de bits para el búfer de color de la ventana actual Para una ventana RGBA, es la suma de `GLUT_WINDOW_RED_SIZE`, `GLUT_WINDOW_GREEN_SIZE`, `GLUT_WINDOW_BULE_SIZE` y `GLUT_WINDOW_ALPHA_SIZE` Para ventanas de índice de color, es el número de bits para índices de color
`GLUT_WINDOW_STENCIL_SIZE`	Número de bits en el búfer de esténcil de la ventana actual
`GLUT_WINDOW_DEPTH_SIZE`	Número de bits en el búfer de profundidad

	de la ventana actual
GLUT_WINDOW_RED_SIZE	Número de bits de rojo almacenados en el búfer de color de la ventana actual Cero si la ventana es de índice de color
GLUT_WINDOW_GREEN_SIZE	Número de bits de verde almacenados en el búfer de color de la ventana actual Cero si la ventana es de índice de color
GLUI_WINDOW_BLUE_SIZE	Número de bits de azul almacenados en el búfer de color de la ventana actual Cero si la ventana es de índice de color.
GLUT_WINDOW_ALPHA_SIZE	Número de bits de alpha almacenados en el búfer de color de la ventana actual Cero si la ventana es de índice de color.
GLUT_WINDOW_ACCUM_RED_SIZE	Número de bits de rojo almacenados en el búfer de acumulación de la ventana actual Cero si la ventana es de índice de color
GLUI_WINDOW_ACCUM_GREEN_SIZE	Número de bits de verde almacenados en el búfer de acumulación de la ventana actual Cero si la ventana es de índice de color
GLUI_WINDOW_ACCUM_BLUE_SIZE	Número de bits de azul almacenados en el búfer de acumulación de la ventana actual Cero si la ventana es de índice de color
GLUT_WINDOW_ACCUM_ALPHA_SIZE	Número de bits de alpha almacenados en el búfer de acumulación de la ventana actual Cero si la ventana es de índice de color.
GLUT_WINDOW_DOUBLEBUFFER	Uno si la ventana actual tiene doble búfer Cero de otra forma
GLUT_WINDOW_RGBA	Uno si la ventana está en el modo RGBA, cero de otra forma (es decir, índice de color)
GLUT_WINDOW_PARENT	El número de ventana del ancestro de la ventana Cero si la ventana es una ventana de nivel superior
GLUT_WINDOW_NUM_CHILDREN	El número de sub-ventanas que tiene la ventana (sin contar hijas de hijas)
GLUI_WINDOW_COLORMAP_SIZE	Tamaño del mapa de colores de la ventana de índice de color Cero para ventanas RGBA
GLUT_WINDOW_NUM_SAMPLES	Número de muestras para multimuestreo de la ventana actual
GLUI_WINDOW_STEREO	Uno si la ventana actual es estéreo, cero de otra forma
GLUI_WINDOW_CURSOR	Cursor actual para la ventana actual.
GLUT_SCREEN_WIDTH	Anchura de la pantalla en pixeles Cero indica que la anchura es desconocida o no está disponible
GLUT_SCREEN_HEIGHT	Altura de la pantalla en pixeles Cero indica que la altura es desconocida o no está

	disponible
GLUT_SCREEN_WIDTH_MM	Ancho de la pantalla en milímetros Cero indica si la anchura es desconocida o no está disponible
GLUT_SCREEN_HEIGHT_MM_	Altura de la pantalla en milímetros Cero indica si la altura es desconocida o no está disponible
GLUT_MENU_NUM_ITEMS	Número de elementos de menú en el menú actual
GLUT_DISPLAY_MODE_POSSIBLE	Indica si el modo de despliegue actual es soportado o no lo es
GLUT_INIT_DISPLAY_MODE	La máscara de bits del modo de despliegue inicial
GLUT_INIT_WINDOW_X	El valor X de la posición inicial de la ventana
GLUT_INIT_WINDOW_Y	El valor Y de la posición inicial de la ventana
GLUT_INIT_WINDOW_WIDTH	El valor de anchura del tamaño inicial de la ventana
GLUT_INIT_WINDOW_HEIGHT	El valor de altura del tamaño inicial de la ventana
GLUT_ELAPSED_TIME	Número de milisegundos desde que se llamó a `glutInit` (o primera llamada a `glutGet(GLUT_ELAPSED_TIME)`)

descripción

`glutget` obtiene el estado simple de GLUT representado por enteros El parámetro `state` determina qué tipo de estado regresar Los estados de capacidades de la ventana regresan valores correspondientes a la capa en uso Los nombres de estado de GLUT que comienzan con `GLUT_WINDOW` regresan el estado para la ventana actual Los nombres de estado *que comienzan con `GLUT_MENU` regresan el estado para el menú actual* Otros nombres de estado de GLUT regresan el estado global El requerir el estado para un nombre de estado de GLUT inválido regresa –1

B.9.1 glutLayerGet

`glutLayerGet` obtiene el estado de GLUT concerniente a las capas de la ventana actual

uso

```
int glutLayerGet(GLenum info);
```

`info` Nombre del dispositivo del cual se obtendrá información

GLUT_OVERLAY_POSSIBLE	Indica si un revestimiento puede ser establecido para la ventana actual dado el actual modo de despliegue inicial Si es falso, `glutEstablishOverlay` caerá con un error fatal si es llamado
GLUT_LAYER_IN_USE	Es ya sea GLUT_NORMAL o GLUT_OVERLAY dependiendo de si la capa en uso es el plano normal o el revestimiento
GLUT_HAS_OVERLAY	Si la ventana actual tiene un revestimiento establecido
GLUI_TRANSPARENT_INDEX	Es el índice de color transparente del revestimiento de la *ventana actual Se regresa un número negativo si no hay* revestimiento en uso
GLUT_NORMAL_DAMAGED	Verdadero si el plano normal de la ventana actual ha sido dañado (por actividad del sistema de ventanas) desde que se detonó la última llamada de retorno de despliegue El llamar a `GLUTPostRedisplay` no establecerá a GLUT_NORMAL_DAMAGED como verdadero
GLUT_OVERLAY_DAMAGED	Verdadero si el plano de revestimiento de la ventana actual ha sido dañado (por actividad del sistema de ventanas) desde que se detonó la última llamada de retorno de despliegue El llamar a `glutPostRedisplay` no establecerá a GLUT_NORMAL_DAMAGED como verdadero Se regresará un valor negativo si no hay revestimiento en uso

descripción

`glutLayerGet` obtiene la información de capa de GLUT para la ventana actual representada por enteros El parámetro `info` determina qué información de capa se regresará

B.9.2 glutDeviceGet

`glutDeviceGet` obtiene información de dispositivos de GLUT representada por enteros

uso

```
int glutDeviceGet(GLenum info);
```

`info` Nombre de la información del dispositivo a obtener

GLUI_HAS_KEYBOARD	Es distinto de cero si está disponible un teclado *Cero si no está disponible Para la mayoría de las* implementaciones de GLUT puede asumirse que se dispone de un teclado.
GLUT_HAS_MOUSE	Distinto de cero si está disponible un ratón Cero

	si no está disponible. Para la mayoría de las implementaciones de GLUT, puede asumirse que se dispone de un ratón
GLUT_HAS_SPACEBALL	Distinto de cero si está disponible un spaceball. Cero si no está disponible
GLUT_HAS_DIAL_AND_BUTTON_BOX	Distinto de cero si está disponible una caja de botón y dial Cero si no está disponible
GLUI_HAS_TABLET	Distinto de cero si está disponible una tableta Cero si no está disponible
GLUT_NUM_MOUSE_BUTTONS	Número de botones soportado por el ratón Si no está soportado el ratón, se regresa cero
GLUT_NUM_SPACEBALL_BUTTONS	Número de botones soportado por el spaceball Si no está soportado el spaceball, se regresa cero
GLUI_NUM_BUTTON_BOX_BUTIONS	Número de botones soportado por el dispositivo de dial y botón Si no está soportado este dispositivo, se regresa cero
GLUT_NUM_DIALS	Número de diales soportado por el dispositivo de dial y botón Si no está soportado este dispositivo, se regresa cero
GLUT_NUM_TABLET_BUTTONS	Número de botones soportado por la tableta digitalizadora Si no está soportado este dispositivo, se regresa cero

descripción

`glutDeviceGet` obtiene información de dispositivos de GLUT representada por enteros. El parámetro `info` determina qué información de dispositivo se regresará. El requerir información de dispositivo para un nombre de información de dispositivo de GLUT regresa un número negativo

B.9.3 glutGetModifiers

`glutGetModifiers` regresa el estado de las teclas modificadoras cuando ciertas llamadas de retorno se generan

uso

```
int glutGetModifiers(void);
```

GLUT_ACTIVE_SHIFT	Establecida si el modificador de mayúsculas o bloqueo de mayúsculas está activo.
GLUT_ATIVE_CTRL.	Establecida si el modificador `Ctrl` está activo.
GLUT_ACTIVE_ALT	Establecida si el modificador `Alt` está activo.

descripción

`glutGetModifiers` regresa el estado de las teclas modificadoras cuando un evento de entrada para una llamada de retorno de teclado, especial o de ratón es generada Esta rutina deberá ser sólo llamada mientras una llamada de retorno de teclado, especial o de ratón está siendo atendida Al sistema de ventanas se le permite interceptar opresiones de teclas modificadoras definidas por el sistema o botones de ratón, en cuyo caso, no se generará llamada de retorno GLUT Esta intercepción será independiente del uso de `glutGetModifiers`

B.9.4 glutExtensionSupported

`glutExtensionSupported` ayuda a determinar fácilmente si tiene soporte una extensión dada de OpenGL

uso

```
int glutExtensionSupported (char * extension);
```

`extension` Nombre de la extensión de OpenGL

descripción

`glutExtensionSupported` ayuda a determinar fácilmente si una extensión dada de OpenGL está soportada o no El parámetro extensión nombra la extensión a verificar Las extensiones soportadas se pueden determinar también con `glGetString(GL_EXTENSIONS)`, pero `glutExtensionSupported` examina correctamente la cadena que se regresa.

`glutExtensionSupported` regresa un valor distinto de cero si la extensión está soportada, cero si no lo está Debe haber una ventana actual válida para llamar a `glutExtensionSupported`.

`glutExtensionSupported` sólo regresa información sobre extensiones OpenGL. Esto quiere decir que extensiones dependientes del sistema (por ejemplo, extensiones GLX) no son reportadas por `glutExtensionSupported`

B.10 Renderizaje de fuentes.

GLUT soporta dos tipos de renderizaje de fuentes: fuentes de trazado, lo cual significa que cada caracter se renderiza como un conjunto de segmentos de línea; y fuentes de mapa de bits, donde cada caracter es un mapa de bits generado con `glBitmap` Las fuentes de trazado tienen la ventaja de que debido a que son objetos geométricos, pueden ser escalados y renderizados arbitrariamente. Las fuentes de mapas de bits son menos flexibles puesto que se renderizan como mapas de bits, pero son usualmente más rápidas que las fuentes de trazado

B.10.1 glutBitmapCharacter

`glutBitmapCharacter` renderiza un caracter de mapa de bits usando OpenGL

uso

```
void glutBitmapCharacter (void *font, int character);
```

`font` Fuente de mapa de bits a utilizar
`character` Caracter a renderizar (no está confinado a 8 bits)

descripción

Sin utilizar ninguna lista de despliegue, `glutBitmapCharacter` renderiza el carácter en la fuente de mapa de bits nombrada Las fuentes disponibles son:

fuente	descripción	estándar de mapas de bits de glifos X para la fuente X llamada:
GLUT_BITMAP_8_BY_13	fuente de ancho fijo con cada caracter dentro de un rectángulo de 8 x 13.	-misc-fixed-medium-r-normal--13-120-75-75-C-80-iso8859-1
GLUT_BITMAP_9_BY_15	fuente de ancho fijo con cada caracter dentro de un rectángulo de 9 x 15.	-misc-fixed-medium-r-normal--15-140-75-75-C-90-iso8859-1
GLUT_BITMAP_TIMES_ROMAN_10	Una fuente proporcionalmente espaciada de 10 puntos tipo Times Roman.	adobe-times-medium-r-normal--10-100-75-75-p-54-iso8859-1
GLUT_BITMAP_TIMES_ROMAN_24	Una fuente proporcionalmente espaciada de 24 puntos tipo Times Roman.	-adobe-times-medium-r-normal--24-240-75-75-p-124-iso8859-1
GLUT_BITMAP_HELVETICA_10	Una fuente proporcionalmente espaciada de 10 puntos tipo Helvética.	-adobe-helvetica-medium-r-normal--10-100-75-75-p-56-iso8859-1
GLUT_BITMAP_HELVETICA_12	Una fuente proporcionalmente espaciada de 12 puntos tipo Helvética	-adobe-helvetica-medium-r-normal--12-120-75-75-p-67-iso8859-1
GLUT_BITMAP_HELVETICA_18	Una fuente proporcionalmente espaciada de 18 puntos tipo Helvética	-adobe-helvetica-medium-r-normal--18-180-75-75-p-98-iso8859-1

El renderizar un caracter no existente no tiene efecto `glutBitmapCharacter` automáticamente establece los modos de desempacado de almacenamiento de OpenGL que requiere apropiadamente y almacena y restaura los modos previos antes de regresar La llamada a `glBitmap` generada ajustará la posición actual del trazador basada en el ancho del caracter.

B.10.2 glutBitmapWidth

`glutBitmapWidht` regresa el ancho de un caracter de mapa de bits

uso

```
int glutBitmapWidth(GLUTbitmapFont font,int character);
```

`font` mapa de bits a utilizar
`character` caracter del cual se regresará en ancho (no confinado a 8 bits)

descripción

`glutBitmapWidth` regresa el ancho en pixeles de un caracter de mapa de bits en una fuente soportada de mapa de bits En tanto que el ancho de caracteres en una fuente puede variar (aunque las fuentes de ancho fijo no varían), las características de altura máximas de una fuente en particular son fijas

B.10.3 glutStrokeCharacter

`glutStrokeCharacter` renderiza un caracter de trazado usando OpenGL

uso

```
void glutStrokeCharacter(void *font, int character);
```
`font` Fuente de trazado a utilizar
`character` Caracter a renderizar (no confinado a 8 bits)

descripción

Sin utilizar ninguna lista de despliegue, `glutStrokeCharacter` renderiza el caracter en la fuente de trazado nombrada Las fuentes disponibles son:

GLUT_STROKE_ROMAN	Una fuente proporcionalmente espaciada Roman Simples para caracteres ASCII del 32 al 127. El tope superior de la fuente es 119 05 unidades; el inferior desciende 33 33 unidades.
GLUT_STROKE_MONO_ROMAN	Una fuente monoespaciada Roman Simplex (con las mismas características que GLUT_STROKE_ROMAN) para caracteres ASCII del 32 al 127 El tope superior de caracter en la fuente es de 119 05 unidades; el inferior desciende 33 33 unidades Cada caracter tiene 104 76 unidades de ancho.

El renderizar una fuente no existente no tiene efecto Se utiliza `glTranslatef` para trasladar la matriz de visión de modelo para adelantar un espacio del ancho del caracter

B.10.4 glutStrokeWidth

`glutStrokeWidth` regresa el ancho en pixeles de un caracter de trazado en una fuente soportada de trazado En tanto que el ancho de los caracteres en una fuente puede variar (aunque las fuentes de ancho fijo no varían), las características de altura máxima para una fuente en particular son fijas

B.11 Renderización de objetos geométricos

GLUT incluye diversas rutinas para generar objetos geométricos tridimensionales fácilmente reconocibles Estas rutinas reflejan la funcionalidad disponible en el conjunto de herramientas descrito en la Guía del Programador de OpenGL y se incluyen en GLUT para permitir la construcción de programas simples GLUT que renderizan objetos reconocibles Estas rutinas pueden ser implementadas como rutinas de renderizaje puramente OpenGL Las rutinas no generan listas de despliegue para los objetos que crean

Las rutinas generan normales apropiadas para iluminación pero no generan coordenadas de textura (excepto por la tetera)

B.11.1 glutSolidSphere, glutWireSphere

`glutSolidSphere` y `glutWireSphere` renderizan una esfera sólida o en malla de alambre, respectivamente

uso

```
void glutSolidSphere (GLdouble radius,
                      GLint slices, GLint stacks);
void glutWireSphere (GLdouble radius,
                     GLint slices, GLint stacks);
```

`radius`	El radio de la esfera
`slices`	El número de subdivisiones alrededor del eje Z (similar a las líneas de longitud)
`stacks`	El número de subdivisiones a lo largo del eje Z (similar a las líneas de latitud)

descripción

Renderiza una esfera centrada en las coordenadas origen de modelado del radio especificado La esfera se subdivide alrededor del eje z en rebanadas y a lo largo del eje z en pilas

B.11.2 glutSolidCube, glutWireCube

`glutSolidCube` y `glutWireCube` renderizan un cubo sólido o de malla de alambre respectivamente

uso

```
void glutSolidCube(GLdouble size);
void glutWireCube(GLdouble size);
```

`size` longitud de cada arista

descripción

`glutSolidCube` y `glutWireCube` renderizan un cubo sólido o de malla de alambre respectivamente El cubo se centra en las coordenada origen de modelaje con lados de longitud `size`

B.11.3 glutSolidCone, glutWireCone

`glutSolidCone` y `glutWireCone` renderizan un cono sólido o de malla de alambre respectivamente

uso

```
void glutSolidCone (GLdouble base, GLdouble height,
                    GLint slices, GLint stacks);
void glutWireCone(GLdouble base, GLdouble height,
                    GLint slices, GLint stacks);
```

`base`	el radio de la base del cono
`height`	la altura del cono
`slices`	el número de subdivisiones alrededor del eje z
`stacks`	el número de subdivisiones a lo largo del eje z

descripción

`glutSolidCone` y `glutWireCone` renderizan un cono sólido o de malla de alambre respectivamente orientado a lo largo del eje z La base del cono se encuentra en z

= 0 y la parte superior en z = height el cono se subdivide alrededor del eje z en rebanadas y a lo largo del eje z en pilas

B.11.4 glutsolidTorus, glutWireTorus

glutsolidTorus y glutWireTorus renderizan un toroide (dona) sólido o de malla de alambres respectivamente

uso

```
void glutSolidTorus (GLdouble innerRadius,
          GLdouble outerRadius, GLint nsides, GLint rings);
void glutWireTorus (GLdouble innerRadius,
          GLdouble outerRadius, GLint nsides, GLint rings);
```

innerRadius	radio interno del toroide
outerRadius	radio externo del toroide
nsides	número de lados para cada sección radial
rings	número de divisiones radiales para el toroide

descripción

glutSolidTorus y glutWireTorus renderizan un toroide (dona) sólido o de malla de alambre respectivamente, centrado en las coordenadas de origen de modelaje cuyo eje está alineado con el eje z

B.11.5 glutSolidDodecahedron, glutWireDodecahedron

glutSolidDodecahedron y glutWireDodecahedron renderizan un dodecaedro (poliedro regular de 12 lados) sólido o de malla de alambre respectivamente

uso

```
void glutSolidDodecahedron(void);
void glutWireDodecahedron(void);
```

descripción

glutSolidDodecahedron y glutWireDodecahedron renderizan un dodecaedro sólido o de malla de alambre respectivamente centrado en las coordenadas de origen de modelaje con un radio de $\sqrt{3}$

B.11.6 glutSolidOctahedron, glutWireOctahedron

`glutSolidOctahedron` y `glutWireOctahedron` renderizan un octaedro (poliedro regular de 8 lados) sólido o de malla de alambra respectivamente

uso

```
void glutSolidOctahedron(void);
void glutWireOctahedron(void);
```

descripción

`glutSolidOctahedron` y `glutWireOctahedron` renderizan un octaedro sólido o de malla de alambre respectivamente centrado en las coordenadas origen de modelaje con un radio de 1 0

B.11.7 glutSolidTetrahedron, glutWireTetrahedron

`glutSolidTetrahedron` y `glutWireTetrahedron` renderizan un tetraedro (poliedro regular de 4 lados) sólido o de malla de alambre respectivamente

uso

```
void glutSolidTetrahedron(void);
void glutWireTetrahedron(void);
```

descripción

`glutSolidTetrahedron` y `glutWireTetrahedron` renderizan un tetraedro sólido o de malla de alambre respectivamente, centrado en las coordenadas origen de modelaje con un radio de $\sqrt{3}$

B.11.8 glutSolidTeapot, glutWireTeapot

`glutsolidTeapot` y `glutWireTeapot` renderizan una tetera[6] sólida o de malla de alambre, respectivamente

uso

```
void glutSolidTeapot(GLdouble size);
void glutWireTeapot(GLdouble size);
```

[6] La tetera *clásica* de gráficas por computadora modelada por Martin Newell en 1975

`size` tamaño relativo de la tetera

descripción

`glutSolidTeapot` y `glutWireTeapot` renderizan una tetera sólida o de malla de alambre, respectivamente Se generan tanto normales de superficie como coordenadas de textura para la tetera La tetera se genera usando evaluadores de OpenGL

tetera
[http://www cs washington edu/homes/csk/tile/graphics/escherization/teapot html]

B.12 Consejos de utilización

Existen diversos puntos a tener en cuenta cuando se escriben programas GLUT Algunos de estos son recomendaciones fuertes, otros simplemente consejos y sugerencias

- No cambie el estado que afectará la forma en la cual una ventana será dibujada en la llamada de retorno de despliegue de la ventana Sus llamadas de retorno de despliegue deberán ser idempotentes
- Si necesita volver a desplegar una ventana, en vez de renderizar en cualquier llamada de retorno en la que se encuentre, llame a `glutPostResdiplay` (o `glutPostOverlayRedisplay` para revestimientos) Como regla general, el único código que renderiza directamente a la pantalla deberá ser llamado desde llamadas de retorno de despliegue; cualquier otro tipo de llamadas de retorno no deberán renderizar a la pantalla
- Si utiliza una llamada de retorno de inactividad para controlar su animación, use llamadas de retorno de visibilidad para determinar cuándo la ventana está completamente oscurecida o iconizada para determinar cuándo no desperdiciar tiempo de procesador en renderización

- Ni GLUT ni el sistema de ventanas redimensionan automáticamente sub-ventanas Si las sub-ventanas deben ser redimensionadas para reflejar un redimensionamiento de la ventana de nivel superior, el programa GLUT es responsable de hacer esto
- Evite usar el modo de índice de color lo más posible El modelo de colores RGBA es más funcional y tiende a producir efectos de intercambio de mapa de colores
- No llame a ninguna rutina GLUT que afecta a la ventana actual o al menú actual si no existe ventana actual o menú actual definido Este puede ser el caso en tiempo de inicialización (antes de que cualquier ventana o menú haya sido creado) o si usted destruye la ventana actual o el menú actual Las implementaciones de GLUT no están obligadas a generar una advertencia debido a que esto disminuiría la velocidad de ejecución de cada una de estas rutinas para asegurarse de que existe una ventana o un menú actual
- Para la mayoría de las llamadas de retorno, la ventana actual y/o el menú actual se establece apropiadamente al tiempo de la llamada de retorno Las llamadas de retorno de temporizador y de inactividad son excepciones Si su aplicación usa diversas ventanas o menús, asegúrese de que se establezcan explícitamente la ventana actual o el menú actual usando `glutSetWindow` o `glutSetMenu` en las llamadas de retorno de inactividad y de temporizador
- Si registra una sola función como una rutina de llamada de retorno para múltiples ventanas, puede llamar a `glutGetWindow` dentro de la llamada de retorno para determinar qué ventana generó la llamada de retorno De igual manera, `glutGetMenu` puede ser llamado para determinar qué menú
- Por omisión, las llamadas de retorno de temporizador y las llamadas de retorno de inactividad pueden ser llamadas mientras un menú emergente está activo En máquinas lentas, una renderización lenta en una llamada de retorno inactiva puede comprometer el desempeño del menú Asimismo, puede ser deseable que se detenga el movimiento inmediatamente cuando un menú es detonado En este caso, use el valor de llamada de retorno de entrada o salida de menú con `glutMenuStateFunc` para observar el uso de menús emergentes
- No seleccione más llamadas de retorno de las que necesita realmente Por ejemplo, sin no desea llamadas de retorno de movimiento o de movimiento pasivo, desactívelas pasándole `NULL` a sus funciones de registro de llamada de retorno. El desactivar las llamadas de retorno de entrada permite a la implementación GLUT limitar los eventos de entrada del sistema de ventanas que deberán ser procesados
- No todas las implementaciones GLUT soportan el mismo rango de capacidades de búfer de cuadro, aunque existen requerimientos mínimos para las capacidades de búfer de cuadro Si se llama a `glutCreateWindow` o a `glutCreateSubWindow` con un modo inicial de despliegue no soportado por la implementación OpenGL, se generará un error fatal con un mensaje explicativo Para evitar esto, `glutGet(GLUT_DISPLAY_MODE_POSIBLE)` deberá ser llamado para determinar si el modo inicial de despliegue es soportado por la implementación de OpenGL

- Las teclas `Retroceso`, `Suprimir` y `Escape` generan caracteres ASCII, de tal forma que se pueden detectar opresiones de estas teclas con la llamada de retorno `glutKeyboardFunc`, no con la llamada de retorno `glutSpecialFunc`
- Mantenga en mente que cuando se daña una ventana, deberá asumirse que todos los búferes auxiliares están dañados y deben ser dibujados de nuevo
- Mantenga en mente que después de realizar un `glutSwapBuffers`, debe asumir que el estado de el búfer de atrás es indefinido
- Si no está usando `glutSwapBuffers` para animación de doble búfer, recuerde usar `glFlush` para asegurarse que requisiciones de renderización son dirigidas al búfer de cuadro Algunas implementaciones de OpenGL pueden descargar los comandos pendientes, pero esto no es obligatorio específicamente
- Recuerde que es ilegal crear o destruir menús, o cambiar, añadir o remover elementos de menú mientras un menú (y cualesquier submenús en cascada) están en uso (esto es "emergidos") Use la llamada de retorno de estado para saber cuándo evitar manipulación de menús
- Es más eficiente usar `glutHideOverlay` y `glutShowOverlay` para controlar el estado de un revestimiento de una ventana en vez de eliminar y reestablecer un revestimiento cada vez que se requiere un revestimiento
- Pocas estaciones de trabajo tienen soporte para mapas de color de revestimientos instalados simultáneamente Por esta razón, si un revestimiento es borrado, o de otra forma no utilizado, es mejor esconderlo usando `glutHideOverlay` para evitar que otras ventanas con revestimientos activos sean desplegadas con el mapa de colores equivocado Si su aplicación usa múltiples revestimientos, use `glutCopyColormap` para promover la compartición de mapas de colores
- Si encuentra advertencias o errores fatales en sus programas, pruebe colocar un punto de ruptura en `glutWarning` o `glutFatalError` (aunque estos nombres son potencialmente dependientes de la implementación) para determinar en qué parte del programa ocurrió el error
- GLUT no tiene rutina especial para salir del programa Los programas en GLUT deberán usar la rutina de salida de ANSI C Si un programa requiere realizar operaciones especiales antes de salir del programa, use la rutina de ANSI C `onexit` para registrar llamadas de retorno de salida GLUT terminará el programa unilateralmente cuando ocurran errores fatales o cuando el sistema de ventanas requiera que termine el programa Por esta razón, evite llamar cualquier rutina de GLUT dentro de una llamada de retorno de salida
- Definitivamente, use la opción `-gldebug` para encontrar errores de OpenGL cuando la renderización de OpenGL aparentemente no funciona apropiadamente Los errores de OpenGL se reportan explícitamente sólo si se les requiere

B.13 Cuestiones de Implementación.

En tanto que se pretende describir la Interfaz API de GLUT y no su implementación, esta sección describe cuestiones de implementación que pueden ayudar tanto a quienes hacen implementaciones de GLUT para que lo hagan apropiadamente, como a proveer a los programadores de GLUT la información necesaria para utilizar mejor GLUT

B.13.1 Convenciones de Nombres es Espacio

La implementación GLUT deberá tener un espacio de nombres bien definido tanto para símbolos exportados como para símbolos exportados visibles pero no de propósito determinado Todas las funciones exportadas tienen el prefijo GLUT Todas las definiciones macro tienen el prefijo GLUT_ No se exportan símbolos de datos Todos los símbolos externos que podrían ser visibles al usuario pero no que no se pretenden exportar deben tener el prefijo GLUT Usuarios de la API de GLUT no deberán usar ningún símbolo con prefijo GLUT

B.13.2 Implementación modular.

A menudo ocurre que las librerías de ventanas resultan en programa grandes, de bulto, por la gran cantidad de código "dinámicamente muerto" que se enlaza en los programas porque no puede determinarse en el tiempo de enlace que el programa nunca requerirá (es decir, ejecutará) el código Una consideración (aunque no primaria) en el diseño de la API de GLUT es hacer a la API lo suficientemente modular de tal forma que los programas que utilizan un subconjunto limitado de la API de GLUT pueden minimizar la porción de la implementación de la librería GLUT requerida Esto asume que la implementación de GLUT está estructurada para tomar ventaja de la modularidad de la API

Una buena implementación puede ser estructurada de tal forma que porciones significativas de código para el manejo de índice de mapa de colores, soporte de dispositivos no estándar (como spaceball caja de dial y botones, y tableta), manejo de revestimientos, menús emergentes, rutinas de manejo de ventanas diversas (pop, push, mostrar, esconder, pantalla completa, iconizar), renderización de figuras geométricas y renderización de fuentes sólo requieran se jaladas dentro de los programas GLUT cuando la interfaz a estas funcionalidades se use explícitamente por el programa GLUT

B.13.3 Verificación de errores y reportes.

La forma en que los errores y advertencias sobre uso impropio de GLUT se reportan a los programas GLUT es dependiente de la implementación La conducta recomendada en el caso de un error es mostrar un mensaje y salir En el caso de una advertencia, la conducta recomendada es mostrar un mensaje y continuar Todos los usos impropios de la interfaz de

GLUT no necesariamente deben ser atrapados o reportados Definir qué condiciones son atrapadas o reportadas deberá estar basado en qué tan cara es la condición que deberá verificarse. Por ejemplo, una implementación puede no verificar cada llamada a `glutSetWindow` para determinar si el identificador de llamada es válido

La carga en tiempo de ejecución para verificación de errores para una operación muy común, podría sobrepasar los beneficios de un reporte de errores limpio Se deja al implementador decidir este intercambio El implementador deberá considerar también la dificultad de diagnosticar el uso impropio sin que se obtenga ningún mensaje Por ejemplo, si un programa GLUT intenta crear un menú mientras un menú está en uso (uso impropio), esto garantiza un mensaje porque este uso impropio puede ser a menudo benigno, permitiendo que el error se quede sin ser notado

B.13.4 Evite restricciones no especificadas de uso de GLUT.

Las implementaciones de GLUT deberán tener cuidado de no limitar las condiciones bajo las cuales deben ser llamadas las rutinas GLUT Se espera que las implementaciones GLUT sean flexibles cuando los programas GLUT llamen a rutinas GLUT con una conducta definida en tiempos "inesperados" Por ejemplo, se le deberá permitir a un programa destruir la ventana actual desde una llamada de retorno de despliegue (asumiendo que el usuario no llama a rutinas GLUT que requieran una ventana actual). Esto quiere decir que antes de atender a las llamadas de retorno, una implementación GLUT debe ser "defensiva" con respecto a la manera como el programa podría haber usado el estado de GLUT manipulado durante la llamada de retorno

Referencias

[AMENTA 1998] N. Amenta, M Bern, y M. Kamvysselis. A new voronoi-based surface reconstruction algorithm SIGGRAPH'98, Computer Graphics Proceedings, Annual Conference Series, pages 415-412, July 1988

[AMENTA 1999] N. Amenta y M.Bern Surface reconstruction by Voronoy filtering Discrete Comput Geom , 22(4): 481-504, 1999

[ATTALI 1998] D Attali. r-regular shape reconstruction from unorganized poings Comput Geom Theory Appl , 10:239-247, 1998

[AURENHAMMER 1991] Aurenhammer, F , 1991, Voronoi diagrams – a survey of a fundamental geometric data structure: ACM Computing Surveys, 23(3), p 345-401

[AURENHAMMER 2000] Aurenhammer, F. and Klein, R "Voronoi Diagrams " Ch 5 in Handbook of Computational Geometry (Ed J -R. Sack and J Urrutia) Amsterdam, Netherlands: North-Holland, pp 201-290, 2000

[BAJAJ 1995] C Bajaj, F Bernardini and G. Xu Automatic reconstruction of surface and scalar fields from 3D scans Comput Graph, 109-118, 1995 SIGGRAOH 95

[BERG 2000] Berg, M ; van Kreveld, M ; Overmans, M ; and Schwarzkopf, O. "Voronoi Diagrams: The Post Office Problem " Ch. 7 in Computational Geometry: Algorithms and Applications, 2nd rev ed Berlin: Springer-Verlag, pp 147-163, 2000

[BERNARDINI 1997] F Bernardini y C L. Bajaj Sampling and reconstructiong manifods using alpha-shapes 9th Canad Conf Computing Geom, páginas 193-198, 1997

[BERNARDINI 1997] F Bernardini, C. Bajaj, J Chen y D Schikore. Triangulation – based object reconstruction methods ACM Sympos Comput Geom , páginas 481-484, 1997

[BLOOMENTHAL 1997] J. Bloomenthal, editor Introduction to Implicit Surfaces, vol 391 MorganKaufmann, 1997

[BOISSONNAT 1984] J D Boissonnat Geometric structures for three-dimensional shape representation ACM Trans Graph, 3(4): 266-286, 1984

[BOISSONNAT 2000] J D Boissonnat y F Cazals Natural coordinates on a surface Rapport de recherche, INRIA, 2000

[BRUCE 1992] J W Bruce y P J. Giblin Curves and Singularities (2a ed) Cambridge University Press, New York, 1992

[CHEW 1993] L P Chew Guaranteed-quality mesh generation for curved surfaces 9th Ann ACM Sympos Comput Geom , pg 274-280, 1993

[CRESSIE 1991] Noel A.C Cressie "Statistics for Spatial Data", A Wiley-Interscience publication, 1991

[CRESSMAN 1959] Cressman, G P An operational objective analysis system Mon Wes Rev., 87, 367-374, 1959

[CURLESS 1996] B. Curless y M Levoy A volumetric method for building complex models from range images SIGGRAOH 96, pg 303-312, 1996

[DELAUNAY] Información sobre Triangulaciones de Delaunay:
http://www ifor.math ethz.ch/ifor/staff/fukuda/polyfaq/node13 html#SECTI ON00040000000000000000
http://tamfana.informatik uni-dortmund de/RVS/MA/Stud/misch/applets/Delaunay/Delaunay html
http://www gris uni-tuebingen de/gris/proj/dt/dteng html

[DEVILLERS 1998] O Devillers Improved incremental randomized Delaunay triangulation 14th Ann ACM Sympos Comput Geom, pg 106-115, 1998

[DEY 1999] T K Dey, K Mehlhorn y E R Ramos. Curve reconstruction: Connecting dots with good reason 15th Ann ACM Sympos Comput Geom , pg 197-206, 1999

[EDELSBRUNNER 1983] H. Edelsbrunner, D.G. Kirkpatrick y R Seidel. On the shape of a set of points in the plane IEEE Trans Inform Theory, IT-29:551-559, 1983

[EDELSBRUNNER 1994] H. Edelsbrunner y E P Mücke. Three-dimensional alpha shapes ACM Trans Graph , 13(1):43-72, Ene 1994

[ENDLICH 1968] Endlich, R M , y R L Mancuso, 1968: Objective analysis of environmental conditions associated with severe thunderstorms and tornadoes Mon Wea Rev , 96, 342-350, 1968

[EPPSTEIN] eppstein, D "Nearest Neighbors and Voronoi Diagrams."
http://www ics uci edu/~eppstein/junkyard/nn html

[FOMENKO 1997] Fomenko y T Krunii Topological Modeling for Visualization Springer, 1997

[FRANCO 1985] Franco Preparata, Michael I. Shamos Computational Geometry, An Introduction Springer-Verlag, 1985

[GIESEN 1999] J Giesen Curve reconstruction, the reaveling salesman problen and menger's theorem on length 15th Annu ACM Sympos Comput Geom , pg 207-216, 1999

[GLUT] http://reality sgi com/OpenGL/GLUT3/GLUT3 html (conjunto de utilidades GLUT)

[GOODIN 1979] W R. Goodin, G J. McRae y J H Seinfeld, A comparison of Interpolation Methods for Sparse Data: Application to Wind and Concentration Fields, American Meteorological Society, 1979

[GUIBAS 1985] Guibas, L. and Stolfi, J , "Primitives for the Manipulation of General Subdivisions and the Computation of Voronoi Diagrams", ACM

Transactions on Graphics, Vol 4, No 2, April 1985, pages 74-123 (http://www cl cam ac uk/users/pjcb2/Publications/Delaunaysurvey/node9 ht ml)

[HOPPE 1992] H Hoppe, T DeRose, T Duchamp, J McDonald y W. Stuetzle. Surface reconstruction from unorganized points Comput. Graphics, 26(2):71-78, 1992, SIGGRAPH '92

[HOPPE 1994] H. Hoppe, T. DeRose, T Duchamp, M Halstead, H Jin, J McDonald, J Schweitzer y W Stuetzle Piecewise smooth surface reconstruction SIGGRAOH 94 Pg 295-302, 1994

[ISAAKS 1989] Isaaks and Srivastava "An Introduction to Applied Geostatistics",Oxford University Press 1989

[JOURNEL 1981] G. Journel and CH J Huijbregts " Mining Geostatistics", Academic Press 1981

[KLEE 1980] Klee, V "On the Complexity of d-Dimensional Voronoi Diagrams " Archiv Math 34, 75-80, 1980

[LEE 91] Jay Lee Comparison of existing methods for building triangular irregular network of terrain from grid digital elevation models *International Journal of Geographical Information Systems*, 5(3):267-285, 1991

[LAWSON 1977] C L Lawson, *Software for C^1 Surface Interpolation*, in Mathematical Software III, John R Rice, editor, Academic Press, New York, pp 161-194, 1977

[LEVIN 1999] D Levin Mesh-independent surface interpolation. In 4th International Conference on Curves and Surfaces, pg 46, 1999 Saint Malo, France

[MELKEMI 1997] M Melkemi A-shapes and their derivatives 13th Ann ACM Sympos Comput Geom , pg 367-369, 1997

[MELKMAN 1987] Avraham A Melkman On-line Construction of the Convex Hull of a Simple Polyline
Information Processing Letters 25, 1987

[MESA3D] www mesa3d org (OpenGL para Linux)

[MOCHIMA] Efficient 2-D geometric operations
http://www mochima com/articles

[OKABE 1992] Okabe, A , Boots, B y Sugihara, K , 1992, Spatial tesselations: concepts and applications of Voronoi diagrams: Wiley & Sons, New York, 532 p

[OKABE 1992] Okabe, A ; Boots, B ; and Sugihara, K Spatial Tessellations: Concepts and Applications of Voronoi Diagrams New York: Wiley, 1992

[OLIVER 1990] M A Oliver and R Webster "Kriging: a method of interpolation for geographical information system", INT. J Geographical Information Systems, 1990, VOL 4, No 3, 313-332

[OWEN 1992] S. Owen. An implementation of natural neighbor interpolation in three dimension. Tesis de Maestría, 1992.

[PREPARATA 1985] Preparata, F. R. and Shamos, M. I. Computational Geometry: An Introduction. New York: Springer-Verlag, 1985.

[ROGERS 1964] Rogers, C. A., 1964, Packing and covering: Cambridge tracts in mathematics and mathematical physics no. 54, Cambridge University Press, 111p.

[SETHIAN 1996] J. A. Sethian. Level Set Methods. Cambridge, University Press, 1996.

[SIBSON 1980] R. Sibson. A vector identity for the Dirichlet tesselation. Math. Proc. Camb. Phil. Soc., 87:151-155, 1980.

[SKIENA 1997] Skiena, S. S. "Voronoi Diagrams." §8.6.4 in The Algorithm Design Manual. New York: Springer-Verlag, pp. 358-360, 1997.

[STROUSTRUP 1997] Bjarne Stroustrup. The C++ Programming Language, 3rd Edition. Addison-Wesley, 1997.

[TERJE 1993] Terje Midtbø. Spatial Modelling by Delaunay Networks of Two and Three Dimensions. Tesis doctoral, Image Processing Group. University of Zagreb, Febrero 1993.

[WATSON 1982] Watson, D. F., 1992, CONTOURING: A guide to the analysis and display of spatial data: Pergamon Press, 321p.

[WATSON 1985] Watson, D. F., 1985, Natural neighbor sorting: The Australian Computer J., 17(4), p. 189-193.

[WATSON 1992] D. F. Watson. Contouring: A guide to the Analysis and Display of Spatial Data. Pergamon, 1992.

[WOO] Mason Woo, Jackie Nieder, Tom Davis, Dave Shreiner. OpenGL Programming Guide. Third Edition, The Official Guide to Learning OpenGL, version 1.2, OpenGL Architecture Review Board. http://www.agcrc.csiro.au/projects/3046CO/modelling taxonomy.html

[WRIGHT 1997] Richard S. Wright Jr., Michael Sweet: Programación en OpenGL. Ediciones Anaya Multimedia, S.A. 1997.

[ZHAO 1998] H.-K. Zhao, S. Osher, B. Merriman y M. Kang. Implicit, nonparametric shape reconstruction from unorganized points useing a variational level set method. Cam report 98-7, UCLA, http://www.math.ucla.edu/applied/cam/index.html, 1998.