26

UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO

ESCUELA NACIONAL DE ESTUDIOS PROFESIONALES.

CAMPUS ARAGÓN

"SIMULADOR DE TRAYECTORIAS DE FLUJOS VOLCÁNICOS INTERACTIVOS 3D (STFVI 3D) "

T E S I S

QUE PARA OBTENER EL TÍTULO DE

INGENIERO EN COMPUTACIÓN

P R E S E N T A :

CRISTIAN ULISES PÉREZ FERNÁDEZ

MÉXICO, 2000

Agradecimientos.

Quiero agradecer toda la paciencia, enseñanzas, dirección y confianza que me brindó mi amigo y director de tesis **Elio Vega Munguía.**

De igual forma, agradezco los consejos y ayuda de **José Luis Villareal** y **Lizbeth Heras.** También agradezco al personal del **Centro Nacional de Prevención de Desastres, Gerardo Juárez** y **Alex Onar** por depositar en mi su confianza para el desarrollo de este proyecto.

Asimismo, quiero agradecer a mis revisores, al **M. en C. Jesús Díaz**, al **Ing. Marcelo Pérez**, al **Ing. Amilcar Monterrosa** y al **Ing. Juan Gastaldi** por sus consejos y la aprobación de este trabajo de tesis.

También quiero agradecer en particular a mi escuela la **Escuela Nacional de Estudios Profesionales Aragón** y al **Departamento de Visualización de la Dirección General de Servicios de Computo Académico** por todo lo que me enseñaron y por el apoyo que me brindaron para realizar este trabajo.

De manera muy especial quiero agradecer a la **Universidad Nacional Autónoma de México** por darme el privilegio de ser un orgulloso miembro de esta, la Máxima Casa de Estudios de México.

Este trabajo esta dedicado a mis padres **Melania Fernández** y **Nicolás Pérez**, a mis hermanos **Joel, Guevar, Tania** y **Diana** por su cariño y por todos los buenos momentos que hemos pasado juntos.

Este trabajo esta dedicado de manera muy especial al amor de mi vida, **Berenice Cervantes**, por todo su cariño, confianza, compañía, alegría, amor y porque a su lado descubrí que la vida es algo maravilloso que guarda muchas cosas hermosas.

También dedico este trabajo a todos mis compañeros, amigos y entrenador del **Equipo de Tae Kwon Do de la ENEP Aragón** por todo lo que me enseñaron y por todos los momentos de alegría, coraje deportivo y triunfo que pasamos juntos.

Índice General

Capítulo IV

Índice de Figuras

Introducción.

En el estudio de los fenómenos naturales, una de las herramientas más utilizadas son los simuladores computacionales. Estos simuladores permiten recrear artificialmente a estos fenómenos en un ambiente aislado, donde es posible el control de parámetros involucrados en el modelo físico que implementan, el cual trata de reproducir el comportamiento del fenómeno. Así se pueden reconstruir escenarios o imaginar otros nuevos en la forma que más convenga a los investigadores y de esta manera evitar ir directamente al lugar donde se desarrolla el fenómeno o esperar a que este suceda, lo cual daría como consecuencia una investigación lenta, costosa y en algunas casos peligrosa.

Tal sería el caso de la simulación de un volcán activo o con cierta probabilidad de activarse. Un sistema que simule a un volcán activo puede resultar muy complejo si se desean involucrar a todos los eventos que se derivan de este fenómeno, por ejemplo: actividad sísmica, sobrecalentamiento de la tierra en los alrededores del volcán, alteración de la topografía causados por deslaves y por reblandeciminetos de tierra, emanaciones de gases y cenizas, expulsión y fragmentación de rocas, flujos de *lava* y *lahares*. Estos últimos son corrientes que fluye cuesta abajo compuestas de materiales volcánicos como: rocas, tierra, cenizas asentadas y agua, que mezclados forman una masa muy consistente capaz de arrasar con pueblos y ciudades enteras.

Como se puede observar, la simulación de un volcán activo es muy compleja y proporciona una enorme cantidad de datos a recolectar, almacenar, procesar y evaluar. Una forma eficiente de administrar estos datos es insertando al simulador volcánico dentro de un *Sistema de Información Geográfico* (*SIG*) el cual le dará un procesamiento más adecuado a la información, traduciendola en productos finales que ayuden a la toma de decisiones por parte de autoridades, organismos ecológicos y personal de protección civil; uno de estos productos son los *Mapas de Riesgo Volcánico,* los cuales muestran gráficamente las zonas de mayor riesgo y son utilizados para evaluar los daños sobre el ambiente, ecosistemas o poblaciones consideradas de alto riesgo por su cercanía al volcán.

México es un país con alto riesgo volcánico, ya que cuenta con más de 14 volcanes activos, los cuales son monitoreados y estudiados por un gran número de personas que tienen la necesidad de utilizar herramientas como los simuladores de volcanes activos.

Debido a lo complejo que puede resultar estudiar conjuntamente a todos los eventos derivados de un volcán activo, es preferible estudiar cada evento de forma separada para obtener mejores resultados en tiempos más cortos, por lo que es conveniente crear simuladores especializados para cada evento en particular. Por ejemplo, un *Simulador de Trayectorias de Flujos Volcánicos* puede enfocarse solo en predecir las posibles rutas de los flujos emanados de un volcán.

El presente trabajo describe el desarrollo *del Simulador de Trayectorias de Flujos Volcánicos Interactivo en 3 Dimensiones* (*STFVI 3D*) que tiene como finalidad ser una herramienta tanto para los investigadores de fenómenos volcánicos como para personal encargado de crear planes de contingencia en casos de desastres de origen volcánico.

El *STFVI 3D* es un simulador gráfico computacional, interactivo y tridimensional que permite la visualización de datos topográficos de volcanes activos o con posibilidades de activarse, pero el *STFVI 3D* no es tan sólo un simulador sino que es un sistema más completo que preprocesa, administra, manipula, analiza y despliega información de tipo geográfica, por lo que se le considera como un *SIG* altamente especializado en la predicción de trayectorias de flujos volcánicos.

La topografía se representa con gráficos 3D de mapas de *Curvas de Nivel* o de *Modelo Digital de Elevaciones* (*MDE*) que son formatos de mapas comúnmente utilizados en cartografía y en la mayoría de los *SIGs*. Para simular las posibles trayectorias de flujos sobre la topografía el sistema utiliza un *Modelo Físico Gravitacional* que calcula las pendientes de los planos que conforman la malla topográfica, hace uso de parámetros de gravedad, velocidad y coeficientes de fricción, estos últimos para simular diferentes viscosidades en el material de los flujos. El *STFVI 3D* también puede variar el número de flujos generados en cada evento volcánico para simular su magnitud.

A continuación se describirán brevemente los elementos del ***Diagrama de Bloques del Funcionamiento del STFVI 3D*** (*ver Figura 0.1*).

El sistema inicia cargando un conjunto de archivos de entrada en formato vector y en formato *raster* que contienen los datos de la topografía del volcán en forma de *Curvas de Nivel* y de *MDE* en una malla regular y rectangular, además carga un archivo de configuración inicial con algunos datos particulares para cada volcán.

Con los datos del *MDE* se realizan cálculos de *derivadas parciales* para obtener los *vectores gradientes* de cada plano del *MDE*. Estos valores son almacenados en memoria y opcionalmente pueden guardarse en un par de archivos con el fin de utilizarlos en simulaciones posteriores. Los *vectores gradientes* son necesarios para determinar las posibles trayectorias de los flujos a generar.

Posteriormente se extraen los valores de las *alturas máxima y mínima* del archivo con el *MDE* para crear un mapa de colores asociado a una *escala de elevaciones* del volcán. Con ayuda de este mapa de colores se aplica una *interpolación de colores* en mapas sin iluminación. En los mapas con iluminación se aplica una interpolación pero de materiales y además se calculan los vectores normales de cada plano para poder aplicarles una iluminación.

Para optimizar el tiempo de despliegue (*tiempo de rendering*)de los mapas, se aplican comandos de encapsulación de *OpenGL* conocidos como *Listas de Despliegue* sobre las rutinas de que definen las geometrías. Estos comandos permiten realizar un preprocesamiento de algunas transformaciones geométricas en las rutinas donde se definen los mapas antes de que estos sean desplegados.

Con ayuda del menú interactivo creado con las librerías de *GLUT* y con la interfaz creada con la librerías *MUI* se pueden realizar diferentes acciones como la selección de un formato de mapas, el número de flujos a simular, un coeficiente de fricción asociado a los flujos, sobreponer una malla rectangular sobre uno de los mapas, o aplicar un método de suavizamiento (*smooth*) en la interpolación de los colores y materiales. A través del teclado también es posible aplicar otras operaciones como mostrar un sistema de ejes y señalamientos de orientación, cambio del color de fondo, mostrar una barra de escala de elevaciones o retornar los mapas a su posición y escala iniciales.

Con la ayuda del *ratón* y en combinación con el teclado es posible aplicar operaciones de rotación, translación y escalamiento de los mapas. También es posible seleccionar un punto de emisión de los flujos desde cualquier posición y orientación de los mapas. Después de que se ha seleccionado un punto de emisión, ajustado un valor de coeficiente de fricción e indicado el número de flujos, se realizan la simulación de las trayectorias de los flujos. La escena resultante muestra al mapa en el formato activo con las trayectorias de los flujos sobrepuestas sobre este.

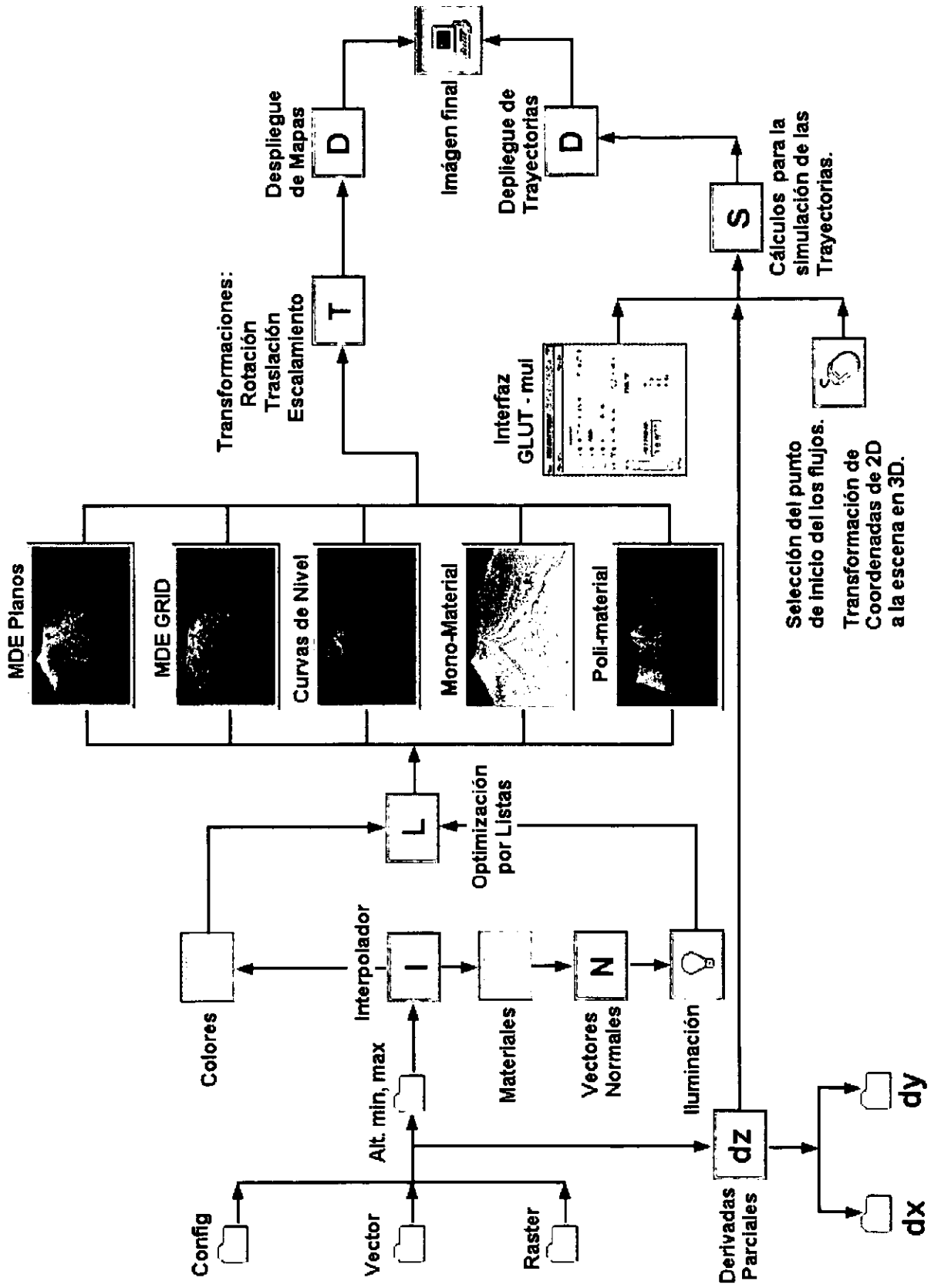


Figura 0.1 Diagrama de Bloques del Funcionamiento del *STFVI 3D*.

Cabe mencionar que el *STFVI 3D* viene a renovar a otro sistema más limitado: el "*RVOL*" por "R*iesgo* VOL*cánico* desarrollado por el *Centro Nacional de Prevención de Desastres* (*CENAPRED*) que utiliza el mismo modelo físico pero está muy limitado en sus capacidades gráficas y de portabilidad entre diferentes plataformas.

Las ventajas que presenta el *STFVI 3D* sobre el sistema *RVOL* del *CENAPRED* son:

- Es un sistema multiplataforma (trabaja sobre ambientes *UNIX*, *LINUX* y *Windows*).
- Se desarrolla utilizando bibliotecas gráficas de alto rendimiento y calidad en gráficos (*OpenGL*, *GLUT*, *MUI*).
- Altamente interactivo, permite realizar transformaciones geométricas de rotación, translación, escalamiento y cambio del modelo de proyección de los diferentes mapas y también permite la selección de puntos de inicio de las trayectorias de los flujos.
- Utilizando técnicas de optimización para el rendimiento del tiempo de procesamiento, despliegue de gráficos y almacenamiento de datos.
- Visualiza varios modelos de mapas topográficos como son el mapa de *Curvas de Nivel*, el mapa del *Modelo Digital de Elevaciones MDE* de malla con polígonos sólidos, con interpolación de materiales, etc.
- Maneja un *Mapa de Colores* para la *Interpolación de Colores* y una *Interpolación de Materiales* con *Iluminación*.

El presente trabajo describe tanto el modelo físico como sus capacidades gráficas y computacionales esperando que sean aprovechados para futuras aplicaciones.

Capítulo I

Introducción a los *Sistemas de Información Geográficos (SIG).*

1.1 Descripción general de los *Sistemas de Información Geográficos.*

Un *Sistema de Información Geográfico (SIG)* se deriva de una clasificación de sistemas más general que es la de los *sistemas de información. Un Sistema de Información* tiene como principal función proveer productos derivados de un procesamiento previo de la información para apoyar al soporte a toma de decisiones.

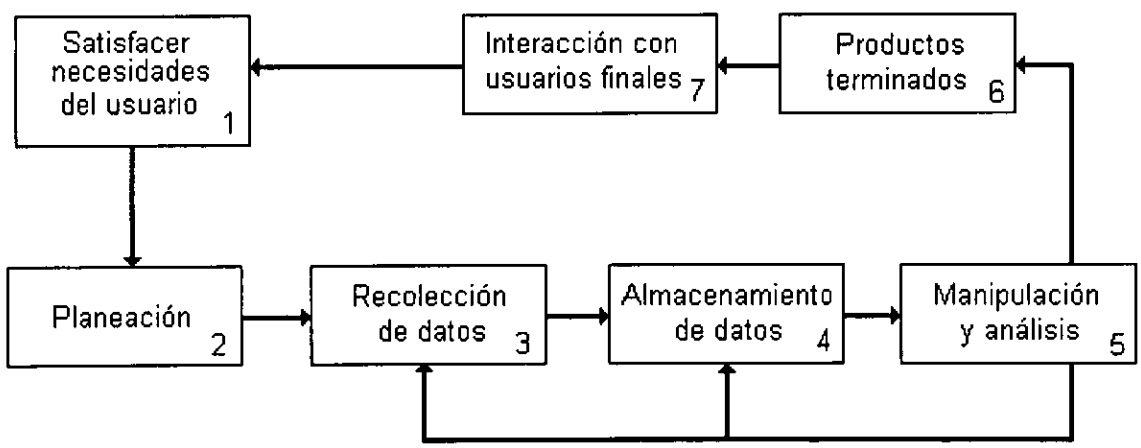


Figura 1.1 Diagrama de bloques de los elementos involucrados en un *Sistema de Información.*

Un *Sistema de Información* es una cadena de operaciones a partir de una planeación o planteamiento de un problema. A partir de este planteamiento se realiza una recolección, almacenamiento, manipulación y análisis de datos, para crear productos finales que integren todos estos datos. Estos productos son proporcionados a los usuarios finales los cuales los utilizarán para sus necesidades de soporte a toma de decisiones, las cuales generarán nuevos planteamientos para resolver nuevos problemas más específicos *(ver Figura 1.1).*

Esto nos trae a un importante concepto: *un mapa es en cierta forma un tipo básico de Sistema deInformación*. Un mapa es una colección de datos almacenados y analizados, y la información derivada de esta colección es usada en soporte a toma de decisiones.

Un *SIG* es un *Sistema de Información* que está diseñado para trabajar con datos geográficos relacionados. En otras palabras, un *SIG* es tanto un sistema de base de datos con capacidades específicas para datos espacialmente georeferenciados como un conjunto de operaciones que trabajan con estos datos.

Es fundamentalmente un sistema cíclico, que trabaja con flujos de datos de sus fuentes primarias para la generación de nueva información para un uso final, pero esta nueva información puede ser reutilizada para resolver nuevos problemas más complejos y generar nuevos productos terminales. Los *Sistemas de Información Geográficos* son diseñados para manipular información respecto a localidades espaciales.

Los *SIGs* modernos son altamente especializados y se adaptan a las necesidades específicas de los usuarios, así podemos tener *SIGs* que presenten productos finales como: mapas de caminos, mapas meteorológicos, mapas de vegetación o mapas de riesgo volcánico.

Una forma de entender un *SIG* es mirando sus aplicaciones. Existimos dentro de un mundo espacial y temporal y tenemos una necesidad de información que tiene dimensiones espaciales y temporales. El ambiente es un sistema dinámico que es mostrado, monitoreado, modelado y simulado. Las futuras decisiones que nos afectarán, como la planeación de nuevos caminos o ciudades, formulación de planes y estrategias agrícolas, localización de sitios de extracción mineral y petrolera, o la creación de planes de contingencia en casos de desastres naturales con la ayuda de mapas de riesgo, dependen en gran medida de la adecuada recolección, manejo, análisis y presentación de la información espacial y temporal.

Un *SIG* especializado en el registro de actividad volcánica utiliza una gran cantidad de información, ya que no solo utiliza los datos que actualmente se están recolectando sino que utiliza los datos registrados en eventos pasados para compararlos con los actuales y poder hacer predicciones más exactas. El registro histórico del vulcanismo aporta una serie de datos muy valiosos para la interpretación correcta de procesos geológicos importantes.

Se ha profundizado en el estudio de este registro histórico, el cual muestra gráficamente como el número de volcanes activos conocidos se ha incrementado paralelamente al crecimiento de la población y a una serie de factores humanos como son los descubrimientos y ocupación de nuevos territorios, los progresos en las comunicaciones, el interés despertado por grandes catástrofes volcánicas, etc. asimismo, el aparente descenso de actividad volcánica en determinados períodos. Con ayuda de los registros históricos de eventos volcánicos y con la información de los nuevos asentamientos humanos es posible crear un sistema de información que ayude a la toma de decisiones a partir de mapas de riesgo y así crear planes eficaces de contingencia ante un futuro evento volcánico.

Hay cinco elementos esenciales que un SIG debe contener: adquisición de datos, preprocesamiento, administración de datos, manipulación y análisis, y generación de productos. En un *Sistema de Información Geográfico*, es importante ver estos elementos como un proceso contínuo.

La Adquisición de Datos es el proceso de identificación y colección de datos requeridos para alguna aplicación. Esto típicamente involucra un número de procedimientos. Un procedimiento puede ser coleccionar nuevos datos para la preparación de mapas a gran escala de vegetación natural en campos de observación, o para la colección de fotografías aéreas para la identificación de áreas urbanas. El muestreo de una topografía para su representación en mapas cartográficos, como los mapas de *Modelo Digital de Elevaciones* (*MDE*) o los de *Curvas de Nivel*.

El Preprocesamiento es el tratamiento previo de los datos para poder ser utilizados por el *SIG* de una forma adecuada. Una de las principales tareas del preprocesamiento es la conversión de formatos originales como fotografías, mapas u otros medios impresos a formatos que se puedan almacenar en bases de datos computacionales adecuadas para su procesamiento y tratamiento.

La Administración de los Datos es un conjunto de operaciones que controlan la creación, permisos de acceso, actualización, eliminación de datos y nuevas capturas en las bases de datos.

La Manipulación y Análisis son frecuentemente el foco de atención para un usuario del sistema. Muchos usuarios creen, incorrectamente, que este módulo es todo lo que constituye un *Sistema de Información Geográfico*. En esta porción del sistema se encuentran los operadores analíticos o algoritmos que trabajan con los contenidos de las bases de datos para generar nueva información.

La Generación de productos es la fase donde son creados los productos finales del *SIG*. Estos productos pueden incluir reportes estáticos, gráficas, mapas, etc. Estos productos pueden ser impresos en papel o desplegados en un monitor de televisión o computadora.

1.2 Ubicación del *STFVI 3D* en los *Sistemas de Información Geográficos.*

Para ubicar correctamente al *STFVI 3D*, es necesario observar sus objetivos y que tipo de necesidades cubre. El principal objetivo del *STFVI 3D* es ser una herramienta de apoyo en la creación de *Mapas de Riesgo Volcánico.*

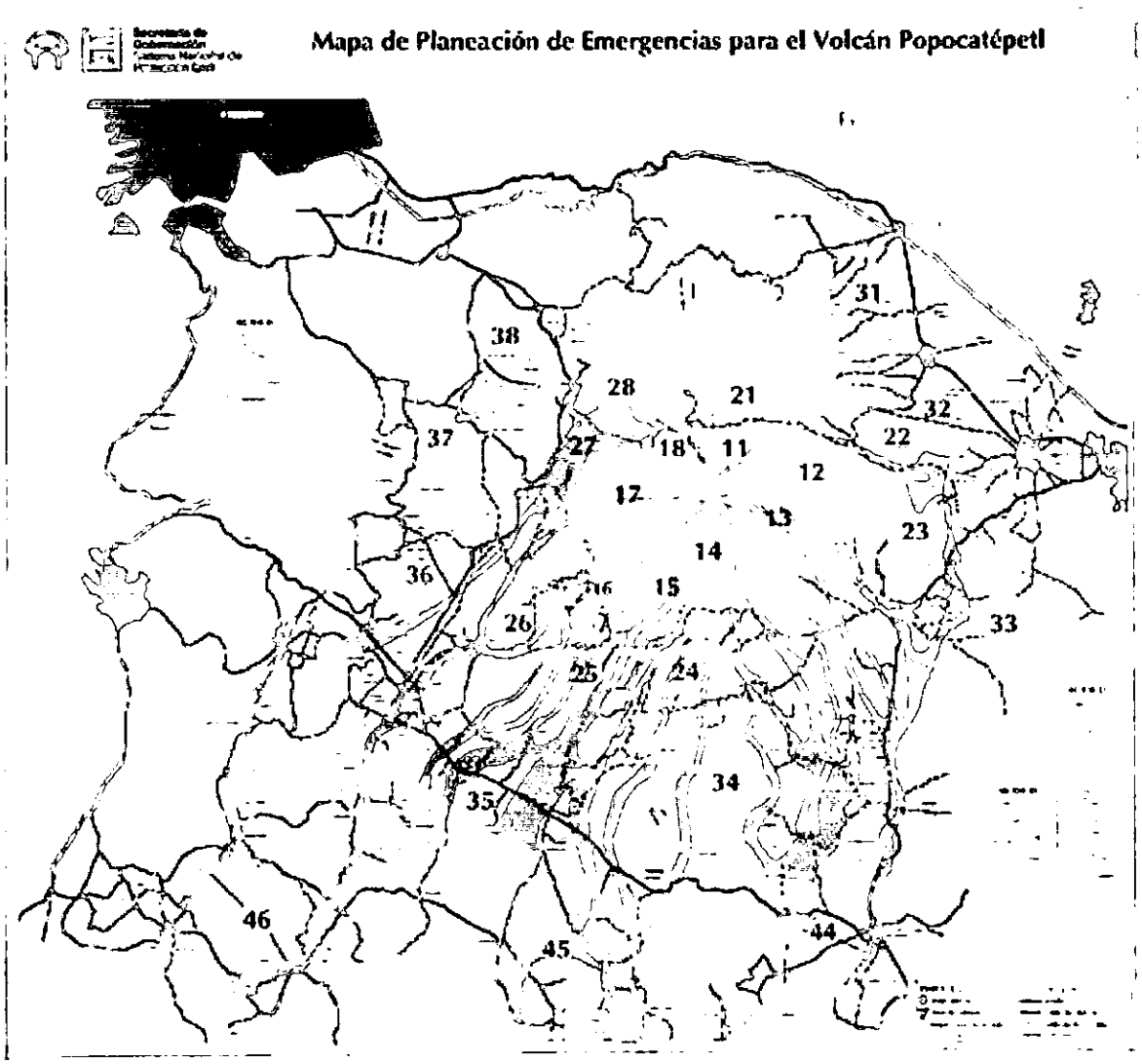


Figura 1.2 ***Mapa de Riesgo Volcánico*** **del Volcán Popocatépetl.**

Los *Mapas de Riesgo Volcánico* tienen como finalidad mostrar a grandes rasgos la magnitud de los daños que puede causar un evento volcánico a un nivel regional. En un nivel local, estos *Mapas de Riesgo Volcánico* pueden ayudar en el diseño de planes de contingencia, estableciendo una escala de prioridades entre localidades, poblaciones y recursos afectados (*ver Figura 1.2*). Por lo mismo es conveniente que estos mapas estén al alcance inmediato del personal encargado de actuar durante emergencias de origen volcánico, como son el personal de Protección Civil, autoridades estatales y municipales, Cuerpo de Bomberos, Cruz Roja, Fuerzas Armadas, etc.

Para ver la importancia de crear herramientas más eficientes en la creación de *Mapas de Riesgo Volcánico* es conveniente hacer una pequeña revisión de otros métodos alternativos que se siguen usando tradicionalmente.

Existen varias formas de crear *Mapas de Riesgo Volcánico*, pero la más común es la forma que tradicionalmente se ha empleado, que es el estudio de la evolución geológica: a través de una interpretación de los materiales arrojados en la historia de un volcán en observación, como se pueden apreciar en un mapa geológico o directamente en observaciones de campo, se genera un *Mapa de Riego Volcánico*. Este tipo de Mapa de Riesgo basa su corrección en la hipótesis de que los lugares más probables para ser afectados por nuevos episodios eruptivos son precisamente aquellos que ya lo han sido previamente. Si bien existen razones para confiar en la factibilidad de esta hipótesis, también tiene limitaciones importantes, la más evidente, es que después de una erupción, la actividad puede cambiar su magnitud, dirección, tipo de actividad y hasta trasladarse a otro punto. Este tipo de mapa es además, estático y presenta únicamente la medida del riesgo, lo cual es incompatible con las expectativas actuales de cantidad de información y capacidad de análisis.

Además de los *Mapas de Riesgo Volcánico* elaborados a partir de interpretación geológica, existe una gran variedad de modelaciones por computadora de distintos aspectos de la erupción de un volcán. Muchos de estos modelos sin embargo, no están orientados estrictamente a la evaluación del riesgo sino al estudio de otros aspectos del fenómeno volcánico. Por ejemplo, algunos presentan un modelo con el que se simula el comportamiento de un flujo, dirigido principalmente a un estudio detallado de la dinámica del material arrojado por un volcán. Este es el caso de muchos modelos unidimensionales, en los que se aprecian principalmente los tiempos de arribo del material a lo largo de un perfil y tienen un valor demostrativo, pero sin permitir una medida del riesgo en la región (que es bidimensional) que circunda al volcán.

También se han propuesto modelos energéticos bidimensionales que permiten conocer principalmente el alcance y la distribución potencial del material. En mayor o menor medida estos modelos solamente recrean cierto aspecto de una erupción. Esta restricción obedece al hecho de que una simulación "más completa" está fuera del alcance de computadoras personales, e incluso, dado cierto nivel de detalle, fuera del alcance de cualquier computadora actual, pues constituye aún un tema de investigación sin soluciones matemáticas satisfactorias.

Recientemente se ha propuesto la creación de *Mapas de Riesgo Volcánico* mediante la modelación de algunos de los aspectos de la actividad volcánica, esperando así superar las limitaciones de los mapas basados únicamente en la geología. Una buena disponibilidad de computadoras dentro de centros científicos y gubernamentales hace posible seguir este camino, con la ventaja adicional de poder utilizar un despliegue dinámico de la información en monitores de computadora, además de la elaboración de mapas finales de tipo estático.

El *STFVI 3D* es un simulador computacional que predice las trayectorias más probables de los flujos con distinta viscosidad de material, dirigidos por la topografía y pretende ser instalado en diversos centros de prevención para realizar una estimación rápida de los riesgos potenciales, permitiendo la evaluación de diversos escenarios de emergencia.

El *STFVI 3D* también sirve directamente como un elemento útil dentro de la investigación y el monitoreo volcánico, permitiendo la confirmación de depósitos ya determinados geológicamente, el pronóstico de depósitos futuros, la elección de sitios para la ubicación de equipo de observación de un volcán, etc. Este método puede sugerir nuevas zonas de riesgo, hasta ahora no afectadas, pero que podrían estar en peligro debido a la acumulación de material durante erupciones previas, derrumbes o la migración de la actividad volcánica. Con este sistema es posible elegir el sitio donde se origina la erupción. Esto agrega la flexibilidad de, por ejemplo, considerar una erupción lateral en un volcán o la migración de la actividad. El usuario puede generar simulaciones de trayectorias de flujos desde distintos puntos, formando lo que denominamos un escenario de riesgo. La distribución de las trayectorias acumuladas durante un episodio eruptivo señala las zonas de mayor riesgo, junto con las que potencialmente serían dañadas en caso de presentarse diversos tipos de actividad.

El *STFVI 3D* no es sólo un simulador, sino que se le considera un *Sistema de Información Geográfico* altamente especializado en la predicción de trayectorias de flujos volcánicos. Se ubica dentro de la clasificación de *SIG* porque es un sistema que reúne prácticamente a todos los elementos esenciales para serlo, es decir, es un sistema que almacena, preprocesa, administra, manipula, analiza y despliega información de tipo geográfica en forma de mapas con gráficos por computadora.

El *STFVI 3D* utiliza bases de datos de *Modelo Digital de Elevaciones* (*MDE*), de *Curvas de Nivel* y otros archivos de datos de configuración que son propiedad del *Centro Nacional de Prevención de Desastres* (*CENAPRED*). Este las adquirió de discos compactos comerciales que publico el *Instituto Nacional de Estadística, Geografía e Informática* (*INEGI*), pero cabe mencionar que el *CENAPRED* hizo algunas pequeñas modificaciones a la base de datos para su conveniencia. Estas bases de datos están contenidas en un conjunto de archivos en formatos *texto*.

El sistema realiza un preprocesamiento de datos al realizar una conversión de formato de los archivos de texto a formato *binario*, esto con la finalidad de ahorrar tiempo de acceso y de espacio de almacenamiento ya que el formato *binario* es de menor tamaño. También se crean nuevos archivos de datos necesarios para cálculos intermedios que se utilizan para el correcto despliegue de los mapas. La actualización de alguna base es muy esporádica y se realiza de forma manual por parte del personal del *CENAPRED*.

En los módulos de manipulación y análisis de datos, el sistema realiza los cálculos matemáticos de la simulación que consisten en cálculos de derivadas parciales para determinar las pendientes de la topografía del volcán en estudio, estas pendientes son necesarias para predecir las trayectorias de los flujos volcánicos.

En el módulo de generación de productos, el sistema muestra un conjunto de mapas en diferentes formatos como son:

- *Mapa de Curvas de Nivel con Interpolación de Colores.*
- *Mapa de MDE tipo Malla de Alambre con Interpolación de Colores.*
- *Mapa de MDE sólido con Interpolación de Colores.*
- *Mapa de MDE sólido Mono-Mateial y Luces.*
- *Mapa de MDE sólido con Interpolación de Materiales y Luces.*

Todos estos mapas pueden ser mostrados en diferentes plataformas de trabajo y se tiene la posibilidad de interactuar con estos, rotándolos, trasladándolos o escalándolos, además de que en todos estos mapas se realizan las simulaciones en tiempo real de las trayectorias de los flujos que también se grafican sobre los mapas. Otros productos creados por el simulador son los archivos con los valores de las derivadas parciales en los ejes "X" y "Y" en archivos separados y que se pueden utilizar para futuras simulaciones.

Capítulo II

Descripción de las bases de datos y del modelo físico.

En el presente capítulo se describirán las de bases de datos que conforman los archivos de entrada y salida que utiliza y genera el *STFVI 3D,* así como el modelo físico utilizado para la simulación de las trayectorias de los flujos. Los archivos contienen datos de coordenadas y alturas necesarios para la graficación de los mapas en diferentes formatos utilizados en sistemas de información geográficos como son el formato *"vector"* o el formato *"raster"*, también hay archivos que contiene información de parámetros necesarios para cálculos intermedios y para generar los mapas de colores asignados a cada topografía. Este capítulo también describe los algoritmos para calcular las derivadas parciales aplicadas a cada punto o vértice de la matriz del *MDE* para obtener los vectores gradientes asociados a cada plano de las topografías, también se describen otros parámetros como: viscosidad, gravedad y velocidad.

2.1 Archivos de *Curvas de Nivel (polígonos)*, archivo *POL.*

Los mapas basados en el *Modelo de Curvas de Nivel* o *Modelo de Contornos* están formado por polilíneas, cada polilínea es un vector de (n) pares de coordenadas (x, y) que describe la trayectoria de una curva a un mismo nivel de altura (z). El número de elementos de cada vector es variable dependiendo de los puntos muestreados para el contorno que representa ese vector o curva de nivel en particular.

El uso directo del *Modelo de Contornos* es poco útil para la simulación de los flujos pero se incluye porque casi todos los sistemas de información geográficos tienen herramientas para incorporarlos y transformarlos a otras estructuras y formatos.

El *STFVI 3D* lee archivos con el sufijo *POL* (*por Polilíneas*) el cual tiene una ordenación de datos en formato vector, sirve para graficar los *Mapas de Curvas de Nivel* (*ver Figura 2.1*), estos archivos también contienen los datos para graficar la red carretera, vías de ferrocarril y los contornos de los límites geográficos de poblaciones aledañas al volcán en estudio.

Los archivos *POL* deben estar en formato "*texto*" (*con extensión .txt*) si se van a utilizar por primera vez, ya que si utiliza formatos "binarios" de una plataforma a otra se tendrá problemas de lectura de estos archivos. El *STFVI 3D* generará automáticamente los archivos "*binarios*" (*con extensión .bin*) para el tipo de plataforma que lo esta ejecutando después de la primera corrida. Los archivos *POL* deben ser nombrados con el siguiente criterio:

*nombre_del_volcan*POL.txt	Para el archivo de "texto"
*nombre_del_volcan*POL.bin	Para el archivo "binario"

Por ejemplo si tenemos los datos del volcán de Colima, *nombre_del_volcan* sería *colima* y el archivo se llamaría :

*colima*POL.txt	Para el archivo de "texto"
*colima*POL.bin	Para el archivo "binario" (este archivo es generado automáticamente después de la primera corrida con el archivo de "texto").

Figura 2.1 Mapa de *Curvas de Nivel* del Volcán Popocatépetl visto con el *STFVI 3D*.

Los archivos *POL* distribuyen sus datos de la siguiente forma:

Al inicio del archivo hay cuatro datos de punto flotante que son los valores máximos y mínimos para las coordenadas (x, y), el orden de estos cuatro valores es el siguiente:

Xmin	Valor máximo para una coordenada x.
Xmax	Valor mínimo para una coordenada x.
Ymin	Valor máximo para una coordenada y.
Ymax	Valor mínimo para una coordenada y.
	A partir de aquí, el archivo contendrá sólo conjuntos de datos que están organizados de la siguiente forma:
Altura	Valor para la componente (z) del siguiente bloque de datos o vector de datos.

LongVector	La longitud del vector, es decir, el número de pares de coordenadas (*x, y*) que contiene el vector. Este dato es necesario para que el programa sepa cuantos datos debe leer para cada contorno.
N puntos (*x, y*)	Un conjunto de (*N*) pares de datos que son las coordenadas (*x, y*) para un cierto contorno, donde *N* es igual a la longitud del vector actual. Si (*Altura*) es igual a (*cero*), el contorno es un segmento de la red carretera o el límite geográfico de alguna población.

Después de leer el primer conjunto de datos compuestos por: una Altura, una Longitud de Vector y un Número puntos (*x, y*), seguirán otros con la misma estructura hasta llegar al final del archivo.

A continuación se lista un fragmento enumerado de un archivo *POL*:

1	18.805579	Valor mínimo para una coordenada *X*.
2	19.148591	Valor máximo para una coordenada *X*.
3	98.332136	Valor mínimo para una coordenada *Y*.
4	98.904913	Valor máximo para una coordenada *Y*.
5	2500.000000	Altura para el vector actual, la componente *Z*.
6	125.000000	Longitud del vector, número de coordenadas (*x, y*).
7	19.120142 98.487602	Inicio del primer bloque de coordenadas (*x, y*).
8	19.120909 98.492012	
---	----------	
---	----------	
---	----------	
130	19.146650 98.497620	
131	19.147308 98.498688	Fin del primer bloque de coordenadas (*x, y*).
132	2600.000000	Altura para el siguiente vector.
133	97.000000	Longitud del siguiente vector.
134	19.120340 98.729912	Inicio del siguiente bloque de coordenadas (*x, y*).
134	19.120413 98.729675	
---	----------	
---	----------	
---	----------	
EOF (*END OF FILE*).		Final del archivo.

Los datos de las coordenadas son valores en distancias angulares, o también conocidas como coordenadas geográficas, estas distancias son determinadas tomando como referencia el primer meridiano que en este caso es el meridiano de *Greenwich.*

A continuación se muestra una parte del código del sistema que realiza la lectura de los archivos *POL*

Las siguientes son estructuras en *lenguaje C*, las cuales están compuestas de variables que contendrán los valores de los datos del archivo *POL.*

```
/* Estructura para leer el formato y las dimensiones                    */
/* del archivo de curvas de nivel POL                                   */

struct HEADER{

     char formato[16];      /* Formato del archivo                      */
                            /* NO UTILIZADO TODAVIA                     */

     float x1, x2;          /* Coordenadas máximas y mínimas del mapa */
     float y1, y2;

     float dimensionX;      /* Dimensiones absolutas del mapa           */
     float dimensionY;

}hdr;


struct LEE              /* Estructura que lee los datos      */
{                       /* del mapa de curvas de nivel       */

     int seg;           /* Numero del Segmento a leer        */
     float *altura;     /* Altura en ese segmento            */
     float *np;         /* Numero de puntos en ese segmento  */

     float *x;          /* Coordenadas (X,Y) del segmento    */
     float *y;

}datos;
```

La siguiente función abre un flujo de escritura en formato texto para el archivo *POL* y después abre otro flujo de escritura para escribir el archivo *POL* en formato binario. La lectura de los datos se lleva acabo de acuerdo a la ordenación de los datos anteriormente descrita.

```
void LeeDatosPolTxtToBin(char *file)
{
   FILE *fpoltxt;
   FILE *fpolbin;
   int i,
   int npp;                          /* npp numero de puntos totales */
   char fileOrigen[80],
   char fileDestino[80];

   strcpy(fileOrigen,file);              strcpy(fileDestino,fileOrigen);
   strcat(fileOrigen, "POL.txt");        strcat(fileDestino, "POL.bin");
   fpoltxt=fopen(fileOrigen,"rt");       fpolbin=fopen(fileDestino,"wb");

   datos.seg=0;
   npp=0;
   fscanf(fpoltxt,"%f",&hdr.y1);
   fwrite(&hdr.y1, sizeof(float), 1, fpolbin);
   fscanf(fpoltxt,"%f",&hdr.y2);
   fwrite(&hdr.y2, sizeof(float), 1, fpolbin);
   fscanf(fpoltxt,"%f",&hdr.x1);
   fwrite(&hdr.x1, sizeof(float), 1, fpolbin);
   fscanf(fpoltxt,"%f",&hdr.x2);
   fwrite(&hdr.x2, sizeof(float), 1, fpolbin);

   while(fscanf(fpoltxt,"%f",&datos.altura[datos.seg])==1)
   {
      fwrite(&datos.altura[datos.seg], sizeof(float), 1, fpolbin);
      fscanf(fpoltxt,"%f",&datos.np[datos.seg]);
      fwrite(&datos.np[datos.seg], sizeof(float), 1, fpolbin);
      for(i=0; i<datos.np[datos.seg]; i++)
      {
         fscanf(fpoltxt,"%f %f",&datos.y[npp+i],&datos.x[npp+i]);
         fwrite(&datos.y[npp+i], sizeof(float), 1, fpolbin);
         fwrite(&datos.x[npp+i], sizeof(float), 1, fpolbin);
      }
      npp+=datos.np[datos.seg];
      datos.seg++;
   }
   fclose(fpoltxt);
   fclose(fpolbin);
}
```

2.2 Archivos del *Modelo Digital de Elevaciones MDE* (*raster*) archivo *GRD*.

El Archivo *GRD* almacena los datos de la topografía de un volcán en forma de una malla regular que contiene la información de las altitudes de este volcán, a este tipo de formato se le conoce como *Modelo Digital de Elevaciones* (*MDE*). Para comprender mejor el formato de este tipo de archivos, a continuación se dará una pequeña explicación de que es el *MDE*.

El *MDE* es una estructura numérica de datos que representa la distribución espacial de las alturas sobre el nivel del mar de la superficie de un terreno.

El *MDE* es frecuentemente utilizado en cartografía para la construcción de mapas topográficos. El modelo se basa en la representación del muestreo de elevaciones de una zona geográfica ordenadas equiespaciadamente tanto en latitud como en longitud (*ver Figura 2.2*).

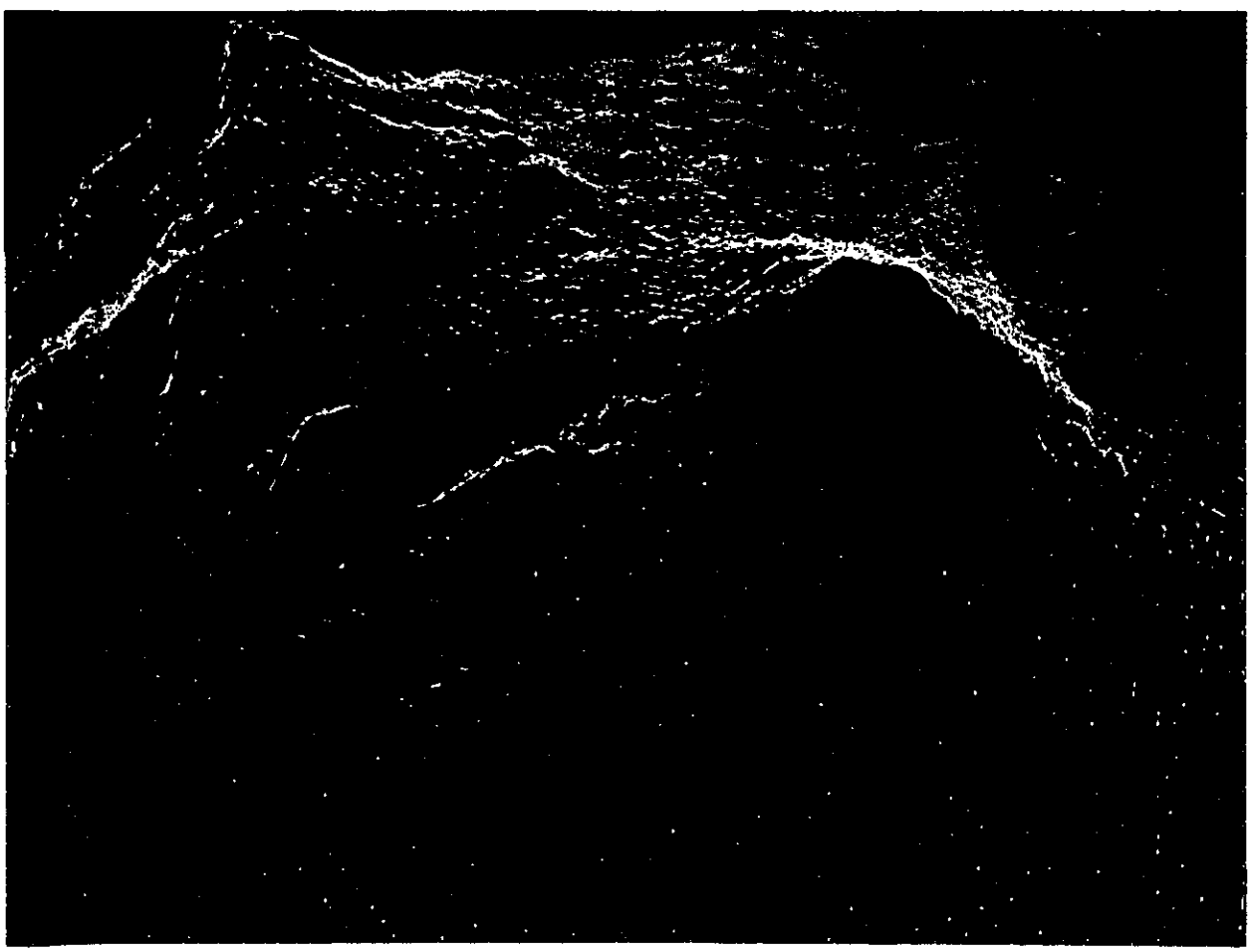


Figura 2.2 *MDE* del volcán Popocatépetl visto en el *STFVI 3D*.

De forma general, la unidad básica de información en un *MDE* es un punto acotado definido como una terna compuesta por un valor de altitud, (z), al que acompañan los valores correspondientes de (x, y) estos últimos no son coordenadas muestreadas sino que se refieren a los índices de la malla utilizada para muestrear la componente (z).

La matriz regular es la estructura más utilizada para construir los *MDE* debido a su cómodo manejo informático y simplicidad estructural.

Las matrices regulares se construyen superponiendo una retícula sobre el terreno y extrayendo la altitud media de cada celda o la latitud de cada nodo de la retícula. Normalmente, la retícula es una red regular de malla cuadrada. La localización espacial de cada dato está determinada de forma implícita por su ubicación en la matriz, definidos el origen y el valor del intervalo entre filas y columnas.

Los archivos *GRD* deben estar en formato "*texto*" (*con extensión .txt*) si se van a utilizar por primera vez, ya que si utilizan formatos "*binarios*" de una plataforma a otra se tendrá problemas de lectura de estos archivos. *El STFVI 3D* generará automáticamente los archivos "*binarios*" (*con extensión .bin*) para la plataforma que lo está ejecutando después de la primera corrida. Los archivos por deben ser nombrados con el siguiente criterio:

*nombre_del_volcan*GRD.txt	Para el archivo de "*texto*"
*nombre_del_volcan*GRD.bin	Para el archivo "*binario*"

Por ejemplo si el tenemos los datos del volcán de Colima, *nombre_del_volcan* sería *colima* y el archivo se llamaría:

*colima*GRD.txt	Para el archivo de "*texto*"
*colima*GRD.bin	Para el archivo "*binario*" (este archivo es generado automáticamente después de la primera corrida con el archivo de "*texto*").

Para utilizar los archivos binarios ya generados se debe añadir el parámetro -b en la línea de comandos.

La siguiente es la estructura en *lenguaje C* que recibe los valores de los datos del archivo *GRD* con el *MDE*.

```
struct GRID                    /* Estructura para leer el mapa GRID    */
{

   float *z[MAXDATOS];         /* Matriz de alturas del MDE            */
   float *dx[MAXDATOS];        /* Matrices de las derivadas en X y Y   */
   float *dy[MAXDATOS];

}grd;
```

La siguiente función abre un flujo de escritura en formato "*texto*" para el archivo *GRD* y después abre otro flujo de escritura para escribir el archivo *GRD* en formato "*binario*". La lectura de los datos se lleva acabo de acuerdo a la ordenación de los datos que se muestra en la *Figura 2.3*.

```
int LeeDatosGrdTxtToBin(char *file)
{

   FILE *fp;
   FILE *fp1;
   int i,j;
   char fileOrigen[80], char fileDestino[80];

   strcpy(fileOrigen,file);
   strcpy(fileDestino,fileOrigen);
   strcat(fileOrigen, "GRD.txt");
   strcat(fileDestino, "GRD.bin");
   fp = fopen(fileOrigen, "rt");
   fp1 = fopen(fileDestino, "wb");

   for(i=0;i<def.ren;i++)
   {
      for(j=0;j<def.col;j++)
      {
         fscanf(fp,"%f",&grd.z[i][j]);
         fwrite(&grd.z[i][j], sizeof(float), 1, fp1);
          }
   }

   fclose(fp);
   fclose(fp1);
   return 0;

}
```

2.3 Archivos de coeficientes de fricción, de alturas máximas y mínimas y de dimensiones de la malla del *MDE*.

El archivo *DEF* es un archivo de "*texto*" que contiene datos adicionales como constantes de coeficientes de fricción, dimensiones absolutas en metros de la longitud del mapa *MDE* medido de este a oeste y el valor máximo de las derivadas (este valor es registrado de alguna simulación previa) necesarios para los cálculos de las derivadas parciales y para determinar las trayectorias de los flujos, este archivo se configura para cada mapa *MDE* y trabaja de forma conjunta con el archivo *GRD*.

A continuación se muestra como ejemplo el contenido de un archivo *DEF*. Los datos de este archivo deben respetar el siguiente orden :

200.000000	Número de renglones de la matriz del *MDE*
200.000000	Número de columnas de la matriz del *MDE*
1.086516	Máximo valor que puede alcanzar una derivada
60001.000000	Distancia en metros del mapa de este a oeste.
0.040000	primer coeficiente de fricción.
0.125000	segundo coeficiente de fricción.
0.310000	tercer coeficiente de fricción.

En las siguientes secciones se describen más detalladamente el significado de estos valores y que papel juegan en el sistema.

La siguiente es la estructura en *lenguaje C* que recibe los datos del archivo de configuración *DEF*.

```
struct DEF
{

   float ren, col;          /* renglones y columnas del matriz del MDE */
   float dvmax, metros;     /* máxima derivada y la distancia del mapa */
                            /* en dirección de este a oeste            */
   float friccion[3];       /* arreglo con los valores de              */
                            /* de fricción los coeficientes            */
   float friccionActual;    /* valor de la fricción actual             */

}def;
```

La siguiente función abre un flujo de lectura en formato "*texto*" para el archivo *DEF* y asigna los datos de este archivo a los elementos de la *estructura DEF*.

```
void LeeDef(char *file)
{

   FILE *fdef;
   char fileDef[80];

   strcpy(fileDef,file);
   strcat(fileDef,".def");

   fdef = fopen(fileDef, "rt");

   fscanf(fdef,"%f", &def.ren);
   fscanf(fdef,"%f", &def.col);

   fscanf(fdef,"%f", &def.dvmax);

   fscanf(fdef,"%f", &def.metros);

   fscanf(fdef,"%f", &def.friccion[0]);
   fscanf(fdef,"%f", &def.friccion[1]);
   fscanf(fdef,"%f", &def.friccion[2]);

   fclose(fdef);

}
```

2.3.1 Dimensiones de las columnas y los renglones de la matriz del *MDE*.

Los primeros dos datos del archivo *DEF* son las dimensiones de la matriz de datos muestrados que definen al *MDE*, aquí se manejan como el número de *renglones* y *columnas* respectivamente de la matriz de datos del archivo *GRD*, hay que poner mucho cuidado de no equivocarse en el orden y los valores en estos dos números porque de estos depende la lectura correcta del archivo *GRD* y de este último archivo dependen todos los demás cálculos internos para resolver las ecuaciones de las trayectorias de los flujos.

El orden en que se recorren los índices de la matriz de (m) renglones por (n) columnas es el que se muestra en la *Figura 2.3*. Este ordenamiento de datos se debe a que se quiso mantener viejos formatos de archivos utilizados por el *CENAPRED* que manejan los datos de las matrices de ese modo.

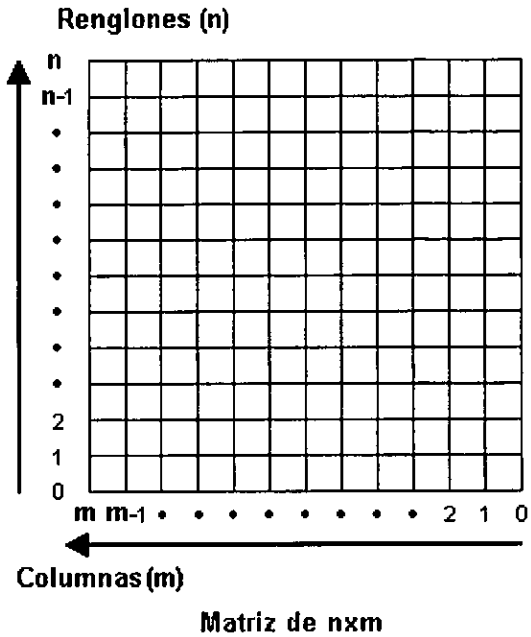


Figura 2.3 Dirección de la ordenación de los datos en una malla de (*nxm*).

2.3.2 Distancias y pendientes máximas del *MDE*.

Un número real es el que define la distancia en metros que mide el mapa en la dirección este - oeste (*E-W*). Este dato se utiliza para calcular la distancia absoluta entre cada celda en dirección abajo - arriba (aquí realmente debería ser de izquierda a derecha).

distanciaCelda = distanciaEsteOste / númeroRenglones.

El siguiente dato es un valor previamente calculado, aunque el sistema lo puede calcular si es necesario. Este valor se refiere a la inclinación máxima o pendiente máxima que puede alcanzar un plano, es decir el máximo valor que puede tener una derivada.

Para obtener este valor se realiza la siguiente operación:

$$dvmax = \sqrt{dxmax^2 + dymax^2}$$

Donde:

dxmax = máxima derivada registrada en x.
dymax = máxima derivada registrada en y.

La forma de obtener los valores *dxmax* y *dymax* se discutirá más adelante.
El valor *dvmax* se utiliza también para modificar el tamaño de la flecha de la derivada.

2.3.3 Coeficientes de fricción.

Los tres últimos valores del archivo *DEF* son tres constantes de coeficientes de fricción y son calculados o asignados empíricamente para cada volcán. Estos valores están entre 0.0 y 1.0 y valores ayudan a simular diferentes viscosidades en el material de los flujos.

Los coeficientes de fricción son un parámetro importante para determinar el comportamiento de un flujo ya que la viscosidad del material limita el alcance del flujo. La inclinación de la topografía aumenta la velocidad de los flujos por la acción de la fuerza de gravedad pero esta velocidad se ve disminuida por diversas razones como: la constante disminución de la inclinación de la topografía, accidentes geográficos como cuencas o muros naturales o la simple fricción del flujo con la superficie, por lo que podemos decir que con un mayor coeficiente de fricción, el flujo se detendrá más rápidamente y su alcance será menor en comparación con otro flujo generado desde la misma posición pero con un menor coeficiente de fricción.

La variación del coeficiente de fricción simula distintas viscosidades del material emitido. Aunque debe admitirse que ésta es una simplificación muy importante de la dinámica de un flujo, a una escala macroscópica su comportamiento se aproxima en efecto a las trayectorias que se obtienen con este modelo. En efecto, este modelo ha reproducido exitosamente las trayectorias de flujos ocurridos en erupciones pasadas, para la mayoría de las simulaciones hechas en diversos volcanes con estos valores de fricción, según lo han comprobado personal del *CENAPRED*.

Para asignar alguno de los valores de coeficientes de ficción se puede usar alguno de los siguientes dos métodos:

1.- Utilizar el menú general que se activa presionando el botón derecho del ratón y escoger entre uno de los tres valores indicados como: *Viscosidad Baja, Viscosidad Media o Viscosidad Alta* (*ver Figura 2.4a*).

2.- Utilizar la *"Barra de Scroll"* de la interfaz *MUI* (*Micro User Interface*), la cual no se limita sólo a tres valores, sino que hace una interpolación de valores entre los datos de mayor y menor coeficientes de fricción (*ver Figura 2.4b*).

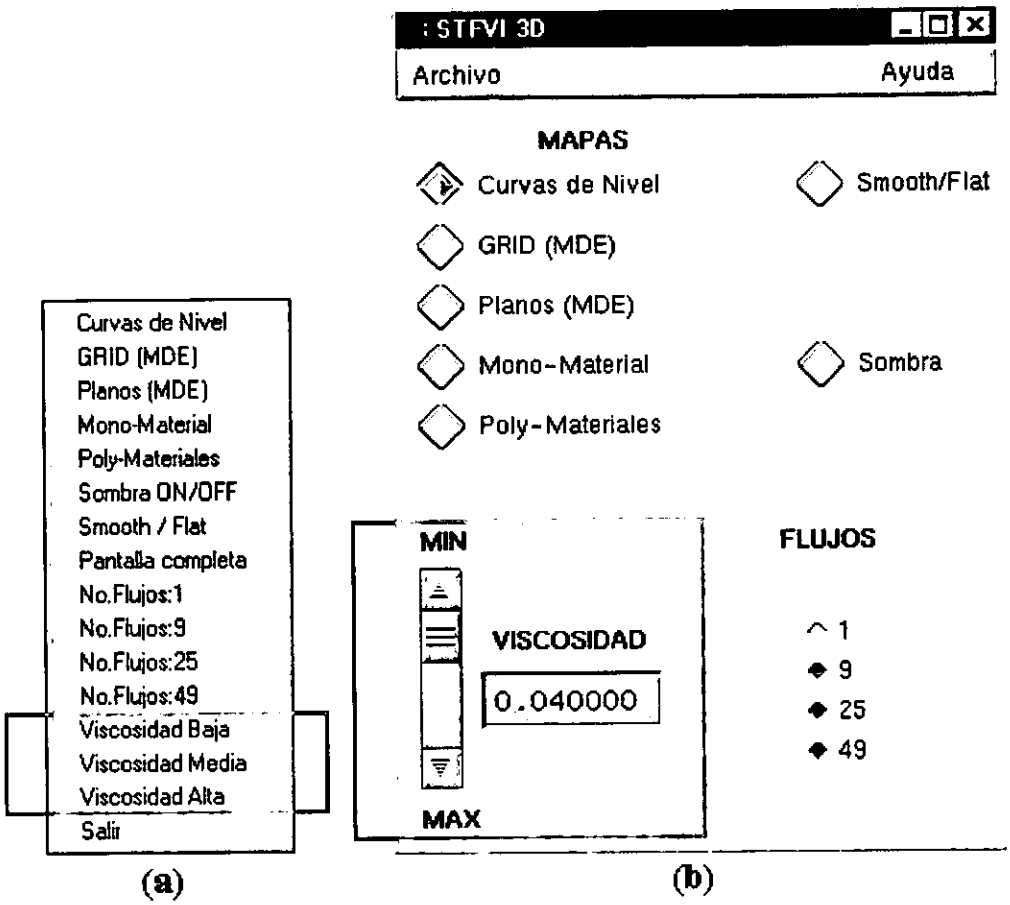


Figura 2.4 Opciones para el control del parámetro de viscosidad del fluido.

2.3.4 Búsqueda de alturas máximas y mínimas en el *MDE* para definir la *Interpolación del Mapa de Colores.*

El archivo *COL* es un archivo de "*texto*" generado por el sistema después de la primera corrida.

El programa internamente hace una búsqueda de la altura máxima y mínima en el archivo *GRD*, estas alturas son datos necesarios para generación de mapas de colores y sus interpolaciones que serán aplicados en las geometrías de los mapas de los volcanes. Cada volcán tiene un mapa de colores propio que se genera a partir de los datos de las alturas máxima y mínima del archivo *GRD*. El procedimiento para generar estos mapas de colores y sus interpolaciones se discutirán en las secciones 3.2.5 y3.2.6 del Capítulo 3.

La siguiente función hace la búsqueda de las alturas máxima y mínima en la matriz de datos del *MDE*.

```
void buscaMaxMin(void)
{
   int i, j;
   float tmp = 0;
   alturaMax = grd.z[0][0];
   alturaMin = grd.z[0][0];
   for(i=0; i<def.ren; i++)
   {
      for(j=0; j<def.col; j++)
      {
         tmp = grd.z[i][j];
         if(tmp > alturaMax)
            alturaMax = tmp;
         if(tmp < alturaMin)
            alturaMin = tmp;
      }
   }
}
```

2.4 Descripción del modelo Físico.

El modelo físico aquí descrito es un modelo gravitacional que parte de los datos del *Modelo Digital de Elevaciones* (*MDE*) contenidos en el archivo *GRD*, a partir del *MDE* se calcula el gradiente, obteniendo sus dos componentes, usando para su aproximación numérica las fórmulas de diferencias de orden $O(h^2)$ [16]. El resultado de este proceso son dos mallas *DX* y *DY* de iguales dimensiones que el *MDE*, con cada una de las componentes del vector gradiente.

Las trayectorias de los flujos se determinan suponiendo que una partícula es depositada sobre una superficie (la topografía) con un coeficiente de fricción y con velocidad inicial cero. A partir de ese momento la partícula es impulsada por el efecto de la gravedad y dirigida por la pendiente variando su velocidad en dirección y en magnitud, hasta salir del área considerada por el mapa o hasta que se detiene por efecto de la fricción.

Como se cuenta con una malla discreta, el *MDE* está formado realmente por pequeñas caras planas. Los valores *Dxij* y *Dyij* son las componentes del gradiente correspondientes al punto *Zij* del *MDE*. Con estos valores se calcula el ángulo de la cara [*i* ,*j*] con un plano horizontal y con este ángulo se resuelven las ecuaciones de movimiento en dos dimensiones para una partícula que se desliza con fricción sobre un plano inclinado.

Las *Figuras 2.5 y 2.6* muestran gráficamente el cálculo de las pendientes para cada plano (*línea morada*), los vectores de aceleración "*ax*", "*ay*" y los vectores de fricción "*frx*", "*fry*".

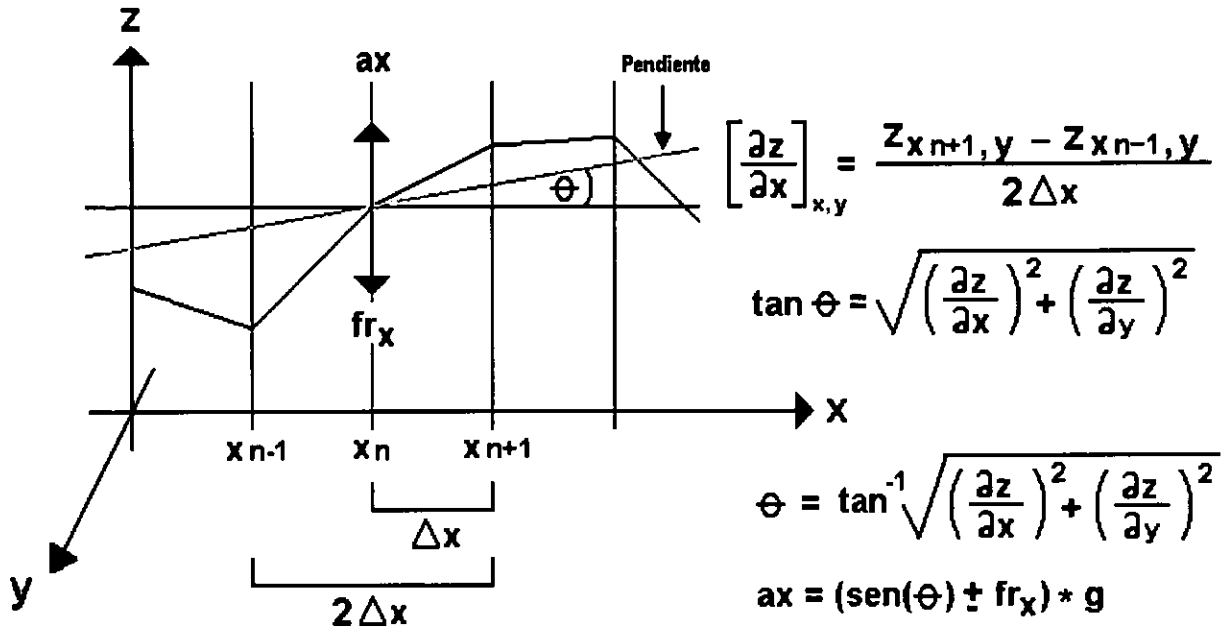


Figura 2.5 Cálculo de las derivadas parciales en "X" y cálculo de la aceleración y fricción de flujos en el plano ZX.

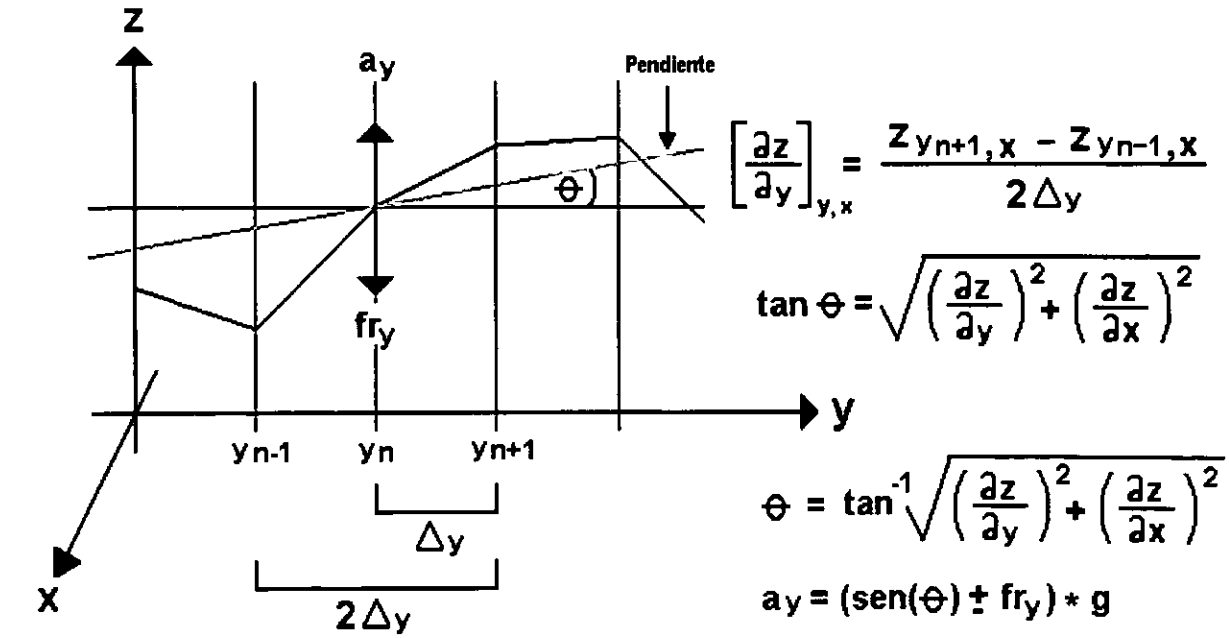


Figura 2.6 Cálculo de las derivadas parciales en "Y" y cálculo de la aceleración y fricción de flujos en el plano ZY.

La siguiente función realiza los cálculos de las derivadas parciales.

```
void derivadas(void)
{
   int i, j, r, c;
   double dxmax=0;
   double dxmay=0;
   r=def.ren;          /* Para hacer mas facil la lectura */
   c=def.col;          /* Para hacer mas facil la lectura */


   grd.dx[0][0]=(-3*grd.z[0][0]+4*grd.z[0][1]-grd.z[0][2])/dosh;

   for(i=1;i<c-1;i++) grd.dx[0][i]=(grd.z[0][i+1]-grd.z[0][i-1])/dosh;

   grd.dx[0][c-1]=(grd.z[0][c-3]-4*grd.z[0][c-2]+3*grd.z[0][c-1])/dosh;

   for(i=0; i<c; i++)
      grd.dy[0][i]=(-3*grd.z[0][i]+4*grd.z[1][i]-grd.z[2][i])/dosh;

   for(i=1;i<r-1;i++)
   {
      grd.dx[i][0]=((-3)*grd.z[i][0]+4*grd.z[i][1]-grd.z[i][2])/dosh;

      for(j=1;j<c-1;j++) grd.dx[i][j]=(grd.z[i][j+1]-grd.z[i][j-1])/dosh;

      grd.dx[i][c-1]=(grd.z[i][c-3]-4*grd.z[i][c-2]+3*grd.z[i][c-1])/dosh;

      for(j=0;j<c;j++) grd.dy[i][j]=(grd.z[i+1][j]-grd.z[i-1][j])/dosh;
   }

   grd.dx[r-1][0]=((-3)*grd.z[r-1][0]+4*grd.z[r-1][1]-grd.z[r-1][2])/dosh;

   for(i=1;i<c-1;i++)
      grd.dx[r-1][i]=(grd.z[r-1][i+1]-grd.z[r-1][i-1])/dosh;

   grd.dx[r-1][c-1]=(grd.z[r-1][c-3]-4*grd.z[r-1][c-2]+ \
                     3*grd.z[r-1][c-1])/dosh;
   for (i=0;i<c; i++)
      grd.dy[r-1][i]=(3*grd.z[r-1][i]-4*grd.z[r-2][i]+ grd.z[r-3][i])/dosh;

   for(i=0;i<r;i++)
      for(j=0;j<c;j++)
      {
         if(dxmax<grd.dx[i][j]) dxmax=grd.dx[i][j];
         if(dymax<grd.dy[i][j]) dymax=grd.dy[i][j];
      }
   def.dvmax = sqrt(POW2(dxmax)+POW2(dymax));
}
```

La simulación de la trayectoria de un flujo es muy sencilla. Se inicia en una posición (*x0, y0*); se consulta el valor de las componentes del gradiente en ese punto; con estos valores se calcula el ángulo y finalmente, la aceleración y la velocidad de la partícula después de un tiempo t; la partícula se mueve a una nueva posición, de acuerdo a su velocidad actual y esta posición se utiliza para repetir todo el proceso.

Al evaluar el gradiente, si (*x0, y0*) coincide con un nodo, los valores de las derivadas parciales son los de ese nodo precisamente. En general no es así y (*x0, y0*) cae en algún punto del interior de una celda. En este caso, se consultan los valores de los cuatro nodos circundantes y la derivada en (*x0, y0*) es el promedio de estos valores pesados por la distancia al punto.

Mientras que la topografía en general aumenta la velocidad del flujo por la acción de la fuerza de gravedad, el coeficiente de fricción de la superficie mencionado previamente pone un límite a este aumento y puede inclusive detener el flujo. Cada avance en la trayectoria de la partícula es graficado con una línea. La simulación se detiene cuando se han cumplido un número determinado de pasos o si el flujo se sale del mapa.

2.5 Control del número de flujos.

El número de flujos puede controlarse de dos formas: con los botones excluyentes de la interfaz *MUI* en la opción de "*FLUJOS*" con la posibilidad de escoger entre los valores de 1, 9, 25 y 49. La otra forma es utilizar las opciones del menú "*pop-up*" con la posibilidad de escoger los mismos valores que brinda la interfaz *MUI* (*ver Figura 2.7*).

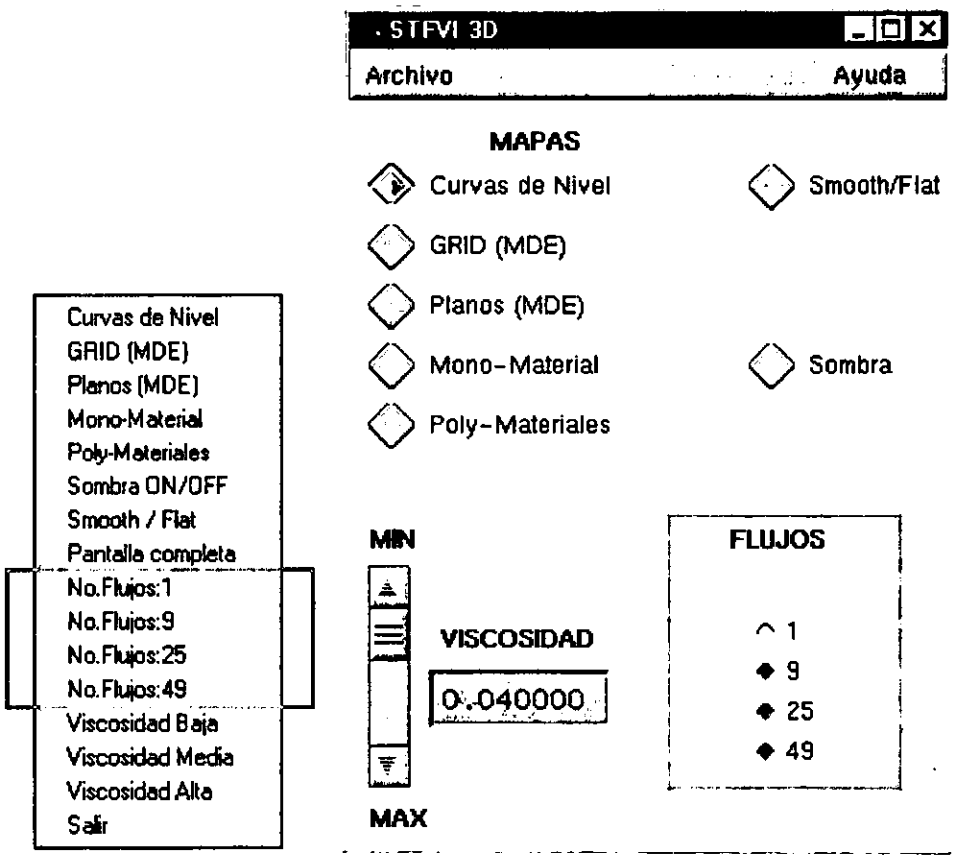


Figura 2.7 Opciones para el control del número de flujos de la simulación.

La variación del número de flujos ayuda a simular la magnitud de un evento volcánico. Cuando se quiere simular un solo flujo, el flujo iniciará su recorrido desde el vértice más cercano al punto de selección con la flecha señaladora.

Cuando se simulan más de un flujo, los flujos se generan con el siguiente criterio: el primer flujo iniciará su recorrido a partir del vértice más cercano al punto de selección, los demás flujos partirán de los vértices vecinos al primero quedando este en el centro de los demás (*ver Figura 2.8*).

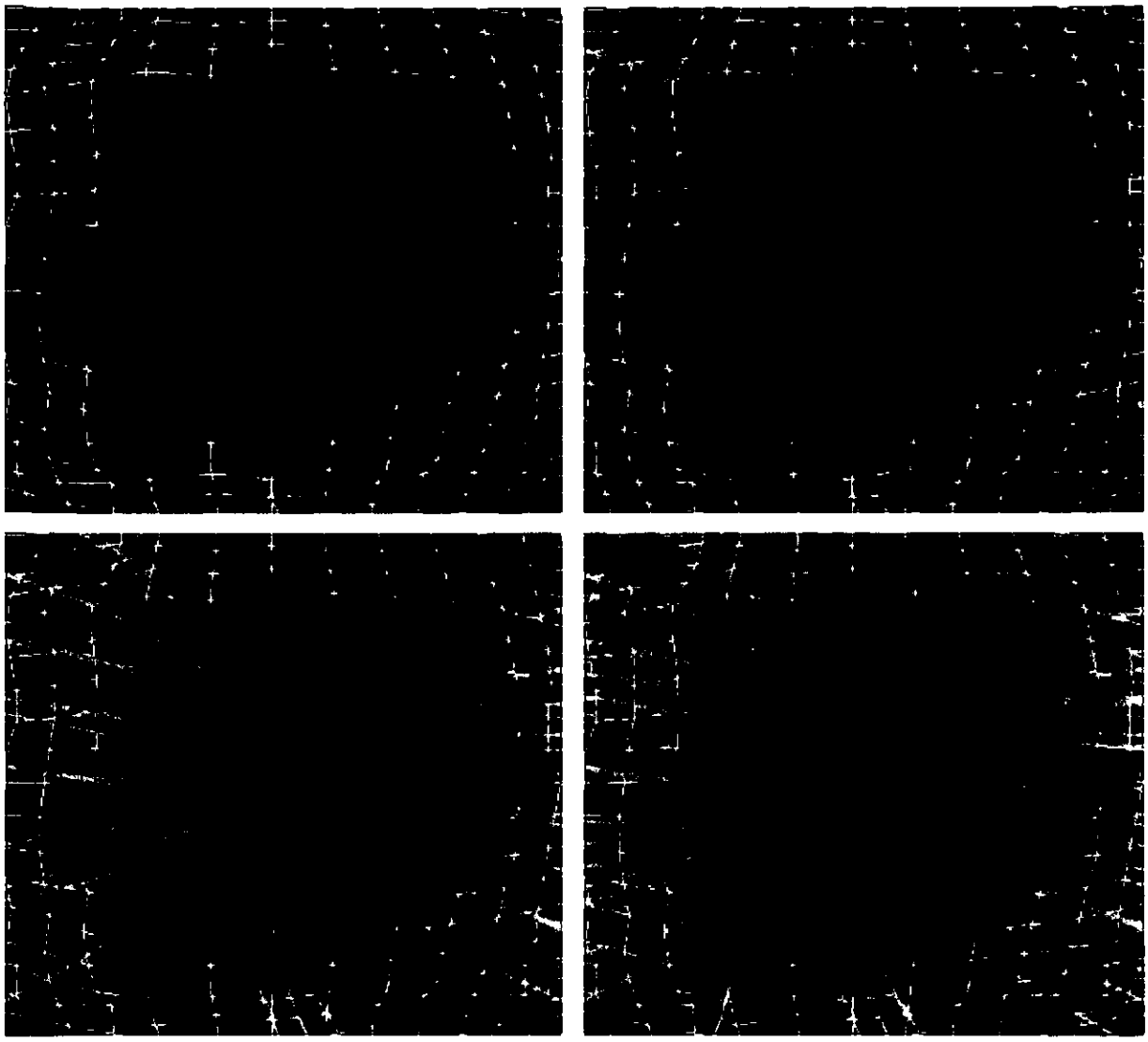


Figura 2.8 Simulaciones de 1, 9, 25 y 49 flujos al variar los controles del número de flujos.

Capítulo III

Modelo gráfico.

3.1 Justificación y descripción general de las Bibliotecas Gráficas utilizadas.

En la presente sección se justificará el uso de las Bibliotecas Gráficas de *OpenGL*, las Bibliotecas para el manejo de ventanas *GLUT* y su extensión *MUI* para la creación de una pequeña interfaz de usuario en el desarrollo del *STFVI 3D*. También se explicarán algunos conceptos utilizados en Gráficos por Computadora y cómo estos son implementados por *OpenGL* en el *STFVI 3D*.

3.1.1 Bibliotecas Gráficas *OpenGL*.

Figura 3.1 Logotipo de las Bibliotecas Gráficas de *OpenGL*.

Para cubrir las necesidades de que el *STFVI 3D* fuera un sistema mutiplataforma, interactivo y que manejara gráficas de alta calidad se decidió utilizar Bibliotecas Gráficas *OpenGL* como el software base para el desarrollo del sistema. *OpenGL* se concidera como una de las mejor opciones en gráficos interactivos de alta calidad en el mercado actual de software de desarrollo gráfico.

Las Bibliotecas Gráficas *OpenGL* (*Open Graphics Libraries*) son unas de las más estándares y populares en el mercado de software gráfico debido a su flexibilidad y portabilidad entre plataformas, creando así un importante ambiente para desarrollo de aplicaciones gráficas e interactivas en 2 y 3 dimensiones.

Las Bibliotecas *OpenGL* fueron lanzadas en 1992 y desde entonces han llegado a ser una de las *Interface de Programación de Aplicaciones* (*API, Aplication Programming Interface*) gráficas en 2D y 3D con gran aceptación por parte de desarrolladores de aplicaciones gráficas.

Existe una gran cantidad de aplicaciones gráficas con fines científicos y de entretenimiento basadas en *OpenGL* que trabajan en una variedad amplia de plataformas de computadoras.

El principal propósito de *OpenGL* es generar un *rendering* para objetos en 2 y 3 dimensiones. Estos objetos son descritos en secuencia de vértices (que definen objetos geométricos) o pixeles (que definen imágenes).

OpenGL es confiable y portable porque las aplicaciones producen resultados de despliegue visual consistente sobre cualquier hardware sin importar el sistema operativo ó sistema de ventanas que posea.

Las Bibliotecas de *OpenGL* tienen las siguientes cualidades:

- Fomenta la innovación y el desarrollo de aplicaciones veloces por incorporar un amplio conjunto de funciones de *rendering* (rendering es el proceso de dibujar o desplegar en una pantalla de computadora a todos los elementos gráficos involucrados en una escena de gráficos por computadora).

- Manejo de colores basado en el Modelo de Color *RGB* y de Indexación de Mapa de Colores.

- Modelo de Iluminación.

- Manejo de Materiales.

- Rutinas para Transformaciones Geométricas.

- Mapeo de Texturas, efectos especiales y otras funciones de visualización.

- Los desarrolladores pueden aprovechar las cualidades de *OpenGL* en todas las plataformas de escritorio (*PCs*) y estaciones de trabajo (*workstation*).

- No contiene comandos para el manejo de ventanas dejando el control a cada plataforma para alcanzar portabilidad entre diferentes sistemas.

- Un conjunto de Bibliotecas Gráficas independiente del *hardware*.

- Un *API* que consta de alrededor de 250 distintos comandos (*para la versión 1.2*), para la creación y manipulación de objetos en 2 y 3 dimensiones.

- Las aplicaciones resultantes son altamente interactivas.

- Es un estándar con un sólido soporte por parte de las más importantes compañías de software y hardware, las cuales conforman el *Comité de Revisión de la Arquitectura de OpenGL* (*ARB*, "*OpenGL Arquitecture Review Board*") para crear una guía única de especificación para *OpenGL*.

- Las bibliotecas son estables porque las futuras contribuciones y mejoras a la especificación original son bien controladas y las actualizaciones propuestas se anuncian a tiempo para que los desarrolladores adopten los cambios necesarios.

- Los requerimientos de compatibilidad hacia versiones anteriores aseguran que existan aplicaciones que no lleguen a ser obsoletas.

- Es un software evolutivo gracias a su "mecanismo de extensión" que permite accesibilidad a nuevas características de hardware y software.

- Es fácil de aprender y utilizar porque esta bien estructurado con un diseño intuitivo y con comandos lógicos, y cuenta con una sintaxis y nomenclatura lógica de los comandos.
- Cuenta con un amplia, buena y detallada documentación (*ver bibliografía*).

La página oficial de *OpenGL* es la siguiente:

http://www.opengl.org

Las bibliotecas precompiladas para la mayoría de las plataformas y documentación de las mismas se pueden obtener en las siguientes direcciones.

http://www.opengl.org/Downloads/Downloads.html

http://www.opengl.org/Downloads/DevDownloads.html

Una característica de *OpenGL* que es importante mencionar es que *OpenGL* es una máquina de estados, es decir, se pueden alterar un conjunto de estados (o modos) los cuales mantienen un cierto efecto en la escena hasta que estos estados cambian su valor.

Por ejemplo el color actual de dibujo es una variable de estado, inicialmente se puede asignar un color determinado a esta variable, los objetos que se crean después de esta asignación serán coloreados con este color, pero si se altera el estado o valor del la variable de color los objetos posteriores serán coloreados con este nuevo color. Es decir los objetos de una escena se alteran o se comportan de cierta forma por la intervención de las variables de estado de *OpenGL*. El valor del color actual es solo uno de muchas variables de estado que maneja *OpenGL*.

La siguiente es una lista de algunas las variables de estados de *OpenGL* que se utilizaron en el desarrollo del *STFVI 3D*:

- La matriz de vista (*view matrix*).
- La matriz de proyección (*projection matrix*).
- El estilo o apariencia de los polígonos.
- El color actual de dibujo.
- Las posiciones de las luces y las características de la iluminación.
- Las propiedades de los materiales de los objetos.

La mayoría de estas variables estado se habilitan o deshabilitan usando los comandos:

```
glEnable();
glDisable();
```

En las siguientes secciones se observará como el *STFVI 3D* usa estas variables de estado.

3.1.2 Bibliotecas *GLUT* para el manejo del sistema de ventanas.

El *OpenGL Utility ToolKit* (*GLUT*) es una interfaz de programación en *ANSI C* y *FORTRAN* que comunica a programas de *OpenGL* con algún sistema de ventanas independiente de alguna plataforma en particular. Este conjunto básico de herramientas soporta las siguientes funcionalidades:

- Manejo de múltiples ventanas para el "*Rendering*" de *OpenGL*.
- Rutinas de "*CallBacks*" (*manejo de eventos*).
- Soporte para sofisticados dispositivos de entrada.
- Un identificador para medir tiempo de ejecución "*Timers*".
- Menúes "*pop-up*" en cascada simples y fáciles de usar (*ver Figura 3.2*).
- Utilerías con rutinas para generar varios objetos sólidos o en malla.
- Soporte para lectura de mapa de bits.
- Asignación de texturas y uso de diferentes "*fonts*" (*fuentes de letras*).
- Una amplia variedad de funciones para el manejo de ventanas, incluyendo "*overlays*".

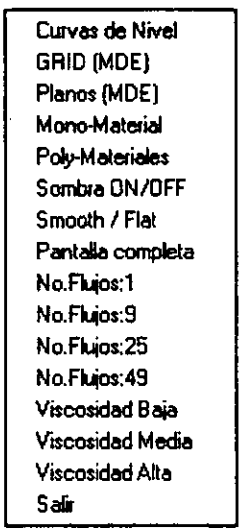


Figura 3.2 Menú pop-up creado con *GLUT* utilizado en el *STFVI 3D*.

El creador de estas bibliotecas es *Mark J. Kilgard* de *Silicon Graphics Inc.* y las ha implementado para versiones en *ANSI C* para los sistemas de ventanas X ("*X Window System*") así como para "*Windows NT*" y "*OS/2*".

Existe una documentación que sirve como especificación de las bibliotecas y como guía de programación. Esta información está disponible en la siguiente dirección:

http://reality.sgi.com/mjk/glut3/glut3.html

Las *Bibliotecas GLUT* son un conjunto básico de herramientas para el manejo de los sistemas de ventanas para programas que utilizan las bibliotecas gráficas de *OpenGL*. Estas implementan un simple *API* (*Application Programing Interface*) para *OpenGL*. *GLUT* hace considerablemente más fácil de aprender y explorar la programación con *OpenGL*.

GLUT provee un *API* portable para escribir programas con *OpenGL* que trabajen tanto en plataformas de *PC* como en estaciones de trabajo con sistemas de ventanas *X11*. *GLUT* está diseñado para construir aplicaciones pequeñas y medianas con *OpenGL*. *GLUT* puede ser aprovechado para aprender *OpenGL* y desarrollar aplicaciones simples de *OpenGL*.

GLUT no cuenta con muchas capacidades para desarrollar aplicaciones muy grandes y que requieren interfaces de usuario muy sofisticadas, para aplicaciones de este tipo es recomendable usar programación de los sistemas de ventanas nativos, usando toolkits como *Motif* o *Athena*. *GLUT* es simple, fácil de entender, usar, y ocupa muy poco espacio por ser muy pequeño.

3.1.3 Bibliotecas para la Interfaz Gráfica de Usuario MUI.

Las Bibliotecas *MUI* (*Micro User Interface*) fueron escritas por *Tom Davis* de *Silicon Graphics Inc.* y vienen como una extensión de las bibliotecas *GLUT* a partir de la versión 3.6.

Figura 3.3 Logotipo de la extensión *MUI* de las Bibliotecas *GLUT*.

El principal inconveniente de *MUI* es que tiene una documentación pobre, el único lugar donde se puede obtener algo de información es la siguiente:

http://reality.sgi.com/mjk/tips/mui/mui.html.

Lo significativo de *MUI* es que está escrito enteramente en el nivel de programación más alto de *GLUT* y *OpenGL* y por lo tanto también es completamente independiente del sistema operativo y del sistema de ventanas.

Para el usuario, *MUI* tiene una apariencia (*look-and-feel*) similar a *X-Motif* (*X-Motif es uno de los manejadores de ventanas más popurales para plataformas X*). Todas las funciones son nombradas utilizando la convención de *SGI* (*Silicon Graphics Inc*) con un "*mui*" como prefijo para cada función.

La *Interfaz de Programación de Aplicaciones* de *MUI* es fuertemente orientada a objetos pero usa la sintaxis de *lenguaje C*.

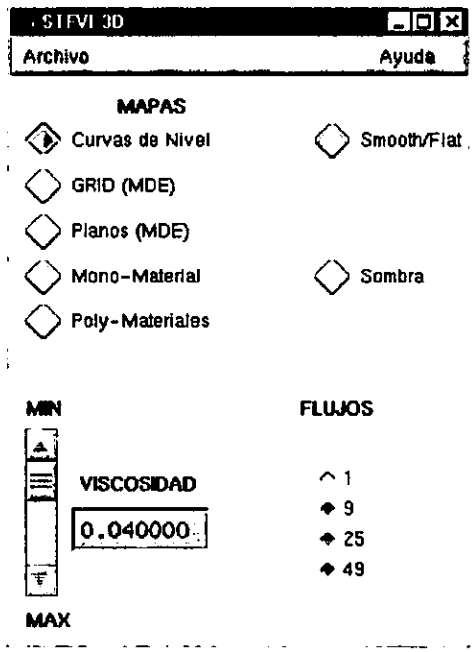


Figura 3.4 Algunos de los componentes gráficos de *MUI* en la interfaz del *STFVI 3D*.

Se decidió utilizar estas bibliotecas complementarias porque contienen un conjunto de componentes gráficos (*widgets*) (*ver Figura 3.4*) como son:

- Barra de menú (*Menu Bar*)
- Opciones de menú (*Menu Items*)
- Botones con texto (*Bottons*)

- Grupos de botones de estado de tamaños grandes y pequeños con texto (*Radio Botton*)
- Etiquetas de texto (*Text Label*)
- Cajas de diálogo (*TextBox*)
- Listas de texto (*Text Lists*)
- Barras de desplazamiento horizontal y vertical (*Vertical and Horizontal Sliders*)

3.2 Fundamentos de Gráficos por Computadora aplicados al *STFVI 3D*.

En esta sección se explicarán brevemente algunos conceptos básicos utilizados en Gráficos por Computadora como son el sistema de coordenadas que se usa comúnmente en gráficos en 3D, las transformaciones geométricas aplicadas a los elementos gráficos de una escena, el concepto de "*Cámara*" utilizado para ubicar objetos en un espacio 3D y para determinar la forma y tamaño de la imagen final, los modelos de proyecciones para definir la forma del volumen de visión el cual contiene a los elementos de una escena, definiciones de primitivas básicas, percepción humana del color y modelos de color, cálculo de vectores normales y su importancia para la iluminación de una escena. Todos estos fundamentos se aplican al *STFVI 3D* a través de *OpenGL*.

3.2.1 Sistema de coordenadas tridimensionales.

El *Sistema de Coordenadas Cartesianas* de tres dimensiones es uno de los más comúnmente utilizados en la representación de gráficos por computadora. Las coordenadas cartesianas son tercias de números que especifican un punto en el espacio cartesiano, estas coordenadas comúnmente se representan con las letras (*x, y, z*).

La coordenada (*x*) representa una posición en el *eje horizontal*, mientras que la coordenada (*y*) representa una posición en el *eje vertical* y la coordenada (*z*) representa una posición en un eje perpendicular tanto al *eje horizontal* (*eje x*) como al *eje vertical* (*eje y*), que en otros términos sería una posición de profundidad.

En la *Figura 3.5* se observa un *Sistema de Coordenadas Cartesianas* donde se muestra la posición del punto (*-4, 5, 4*) en este espacio coordenado. Para observar el punto desde un mejor ángulo, el *eje x* y el *eje y* están rotados.

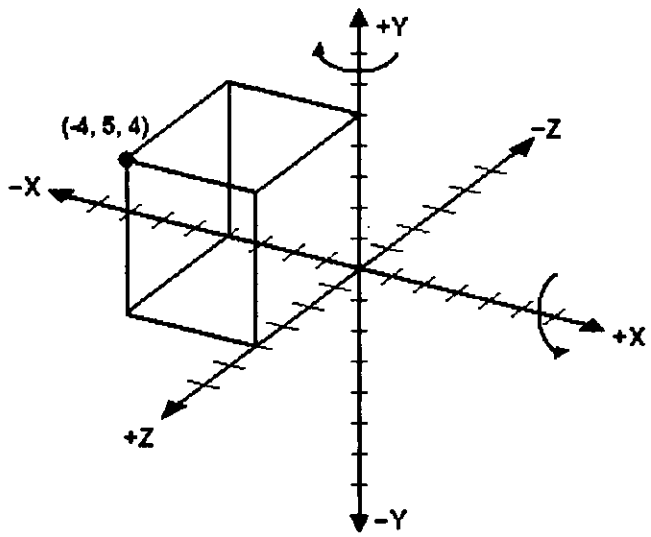


Figura 3.5 Localización de un punto en un espacio de coordenadas cartesianas.

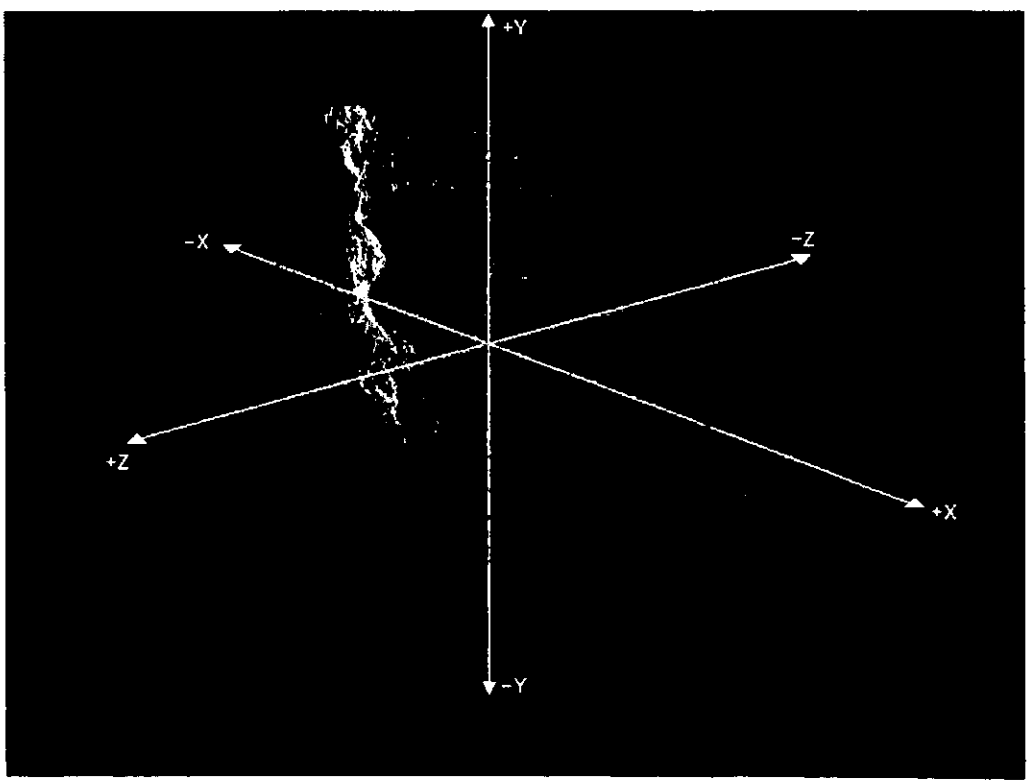


Figura 3.6 El *STFVI 3D* trabaja sobre un espacio de coordenadas cartesianas.

El *STFVI 3D* trabaja sobre un *Sistema de Coordenadas Cartesianas* como se puede observar en la *Figura 3.6*. Los datos de las alturas de los mapas corresponden a las coordenadas (z) en el sistema cartesiano.

Cuando se trabaja con gráficas 2D es fácil mapear coordenadas (x, y) de un plano bidimensional a las coordenadas de puntos o pixeles en una pantalla plana 2D de computadora y de otros dispositivos de despliegue 2D, esto se debe a que existe una correspondencia natural entre las coordenadas del plano gráfico y las coordenadas del dispositivo físico.

Pero para representar gráficas por computadora en tres dimensiones en una pantalla plana de computadora que solo puede representar un sistema coordenado en 2D, hay que crear un efecto de fondo o una ilusión óptica que haga pensar en una tercera dimensión. La complejidad que se añade en las representaciones de objetos tridimensionales es causada por la adición de esta ilusión de dimensión extra. Esta ilusión puede crearse por la proyección de los objetos que se pretenden dibujar en la escena.

Los ambientes tridimensionales pueden verse muy complejos a primera vista, pero en realidad son el resultado de una serie de pasos simples de entender. Esta serie de pasos y la forma de proyectar objetos se discuten en las siguientes secciones.

3.2.2 Analogía de la "*Cámara Fotográfica*" con la creación de escenas 3D.

Una analogía útil para la creación y la observación de una escena en 3D es el concepto de la manipulación de una Cámara Fotográfica. La *Figura 3.7* muestra los pasos a seguir en el proceso para obtener imágenes 3D con una cámara fotográfica y con una computadora.

A continuación se describen los pasos mostrados en *la Figura 3.7:*

- Ubicar y dirigir la Cámara en la escena. Transformación de vista (*viewing transformation*).
- Colocar dentro de la escena a los objetos a fotografiar. Transformación de modelado (*modeling transformation*).
- Escoger los lentes de la cámara o ajustar el zoom. Transformación de proyección (*projection transformation*).
- Determinar el tamaño de la fotografía. Transformación de *viewport* (*viewport transformation*).

Después de ejecutar todos estos pasos la fotografía puede ser tomada en el caso de la Cámara o dibujada en el caso de la Computadora.

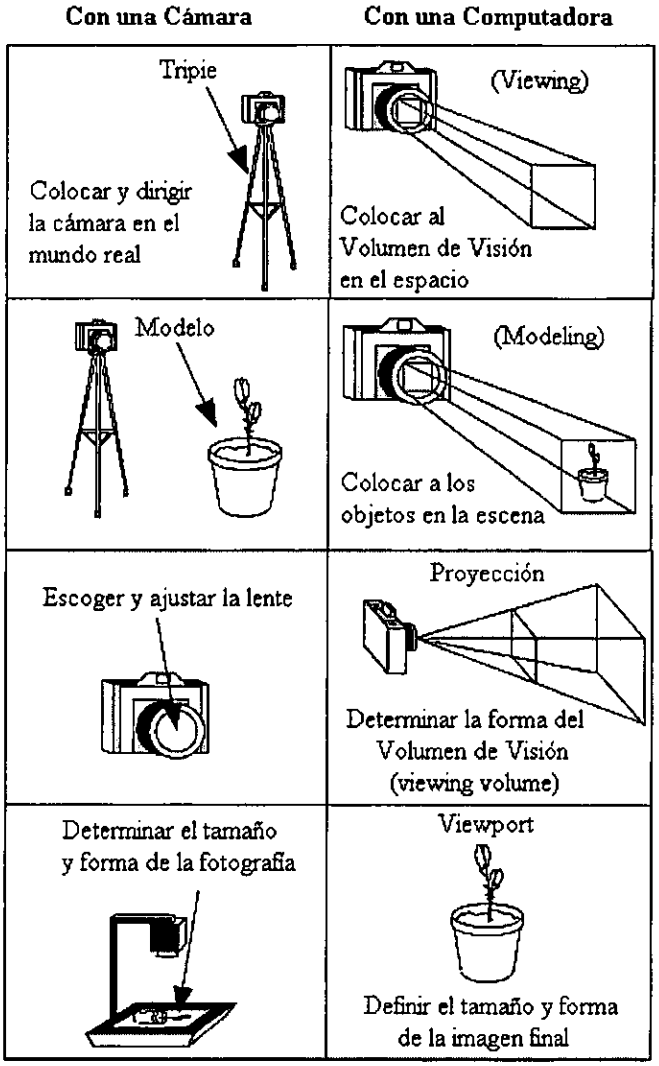


Figura 3.7 Pasos del proceso en la obtención de una escena 3D con una *Cámara Fotográfica* y con una Computadora.

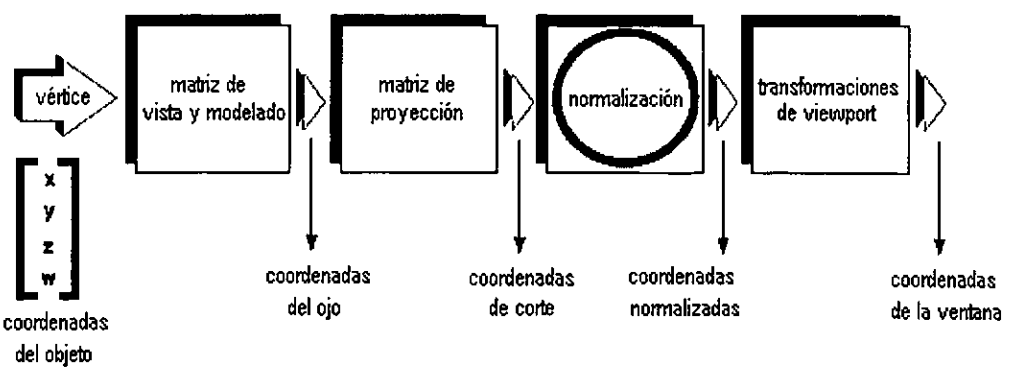


Figura 3.8 Orden de las operaciones matriciales para obtener una escena 3D.

Hay que hacer notar que estos pasos pueden no seguir este orden de manera estricta, ya que estos pasos son operaciones con matrices y estas operaciones pueden obtener los mismos resultados si se aplican en otro orden.

La única restricción es que la transformación de vista debe preceder a la transformación de modelado, pero se pueden especificar transformaciones de proyección y de viewport en cualquier punto antes del proceso de dibujo. La *Figura 3.8* muestra el orden en que estas operaciones ocurren en la computadora.

Para especificar las transformaciones de vista, modelado y proyección, se construye una matriz M de 4×4; que se multiplica por las coordenadas de cada vértice v de los objetos en la escena para completar la transformación: $v' = Mv$.

Las transformaciones de vista y modelado se combinan para formar la matriz de vista-modelado (*model-view*) que se aplica a los vértices de los objetos. Posteriormente se aplican *Planos de Corte* (*clipping planes*) para eliminar objetos o recortar parte de estos que salen del volumen definido por estos planos.

Después de esto *OpenGL* aplica la matriz de proyección a las coordenadas de los objetos resultantes del paso anterior. Estas transformaciones definen un volumen de visión (*viewing volume*); los objetos que se encuentran fuera de este volumen se recortan de modo que no se dibujen en la escena final.

Después de este punto, se realizan operaciones de normalización, para producir las coordenadas finales para el dispositivo físico.

Finalmente las coordenadas transformadas se convierten a coordenadas de ventana a través del proceso de aplicar transformaciones de *viewport*. Se pueden manipular las dimensiones del *viewport* para producir la imagen final, la cual puede estar alargada o encogida. Todas estas transformaciones se discuten más detalladamente en la siguiente sección.

3.2.3 Transformaciones Geométricas.

Las Transformaciones Geométricas, en Gráficos por Computadora, se refieren a operaciones con matrices. Hay transformaciones de vista (*viewing*), transformaciones de modelado (*modeling*), transformaciones de proyección (*projection*) y transformaciones de *viewport*. Las operaciones con matrices resultan en rotaciones, translaciones, escalamientos, proyecciones ortogonales o en perspectiva sobre los objetos de una escena y determinan el área disponible de la pantalla para el despliegue de la escena.

Transformación de Vista.

La transformación de vista cambia la posición y orientación del punto de observación. En la analogía con la Cámara Fotográfica, la transformación de vista equivale a colocar la cámara en un tripie y dirigirla al modelo a fotografiar. En *el STFVI 3D* las coordenadas de la cámara (del observador) están transladadas positivamente sobre el eje (z).

Transformación de Modelado (rotación, translación y escalamiento).

Las tres rutinas de *OpenGL* para transformaciones de modelado son las de Translación Rotación y Escalamiento. Como se puede suponer, estas rutinas afectan a un objeto (o a un sistema de coordenadas, si se piensa al sistema como un objeto) moviéndolo, rotándolo y escalándolo. *OpenGL* cuenta con tres comandos que producir traslaciones, rotaciones y escalamientos apropiados a través de operaciones con matrices. Los comandos `glTranslate()`, `glRotate()` y `glScale()`de *OpenGL* calculan automáticamente las operaciones con matrices. El *STFVI 3D* utiliza estos comandos y otros de control del ratón para rotar, transladar y escalar los mapas en la escena.

Transformación de Proyecciones.

Las transformaciones encargadas de determinar como se proyectan los objetos dentro de una escena. *OpenGL* cuenta con dos tipos básicos de proyecciones. Un tipo es la proyección en perspectiva que es la forma en la que vemos las cosas en la vida diaria. La proyección en perspectiva hace que los objetos que están más lejanos se vean más pequeños; por ejemplo, si un observador se coloca entre las vías del tren y las ve a lo largo, estas parecen que convergen en la distancia. El otro tipo de proyección es la ortogonal, la cual mapea directamente las coordenadas de un objeto dentro de la pantalla sin afectar sus tamaños relativos. Las proyecciones ortogonales se usan frecuentemente en arquitectura y en aplicaciones *CAD*, donde las imágenes finales necesitan reflejar las medidas originales del objeto que se desea representar.

Los comandos de *OpenGL* que se utilizan para especificar el tipo de proyección a utilizar son el `gluPerspective()` (*ver Figura 3.9*) y el `glOrtho()` (*ver Figura 3.10*) los cuales le dan forma al volumen de visión que contiene a los objetos de la escena. El *STFVI 3D* cuenta con la opción de cambiar entre estos dos tipos de proyecciones.

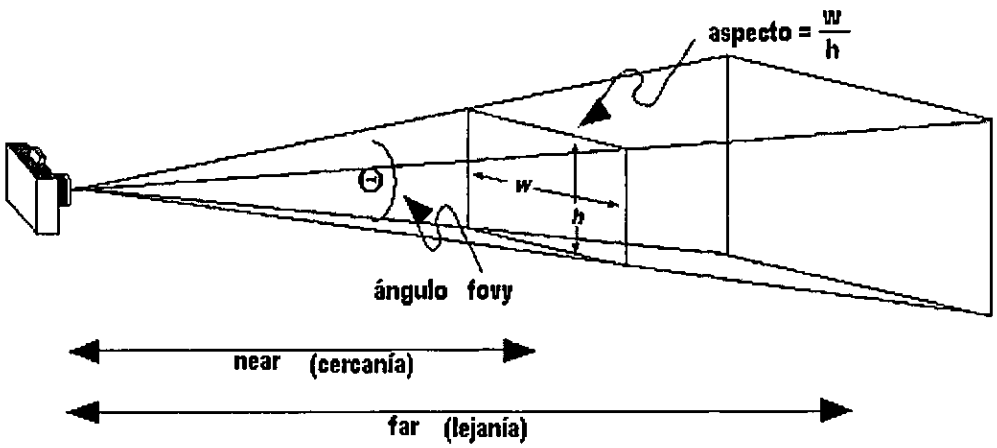


Figura 3.9 Proyección en Perspectiva utilizando el comando *gluPerspective()*.

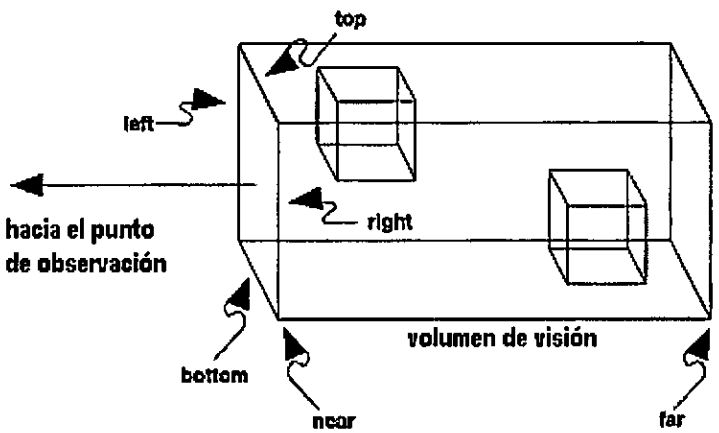


Figura 3.10 Proyección Ortogonal utilizando el comando *glOrtho()*.

Transformación de *viewport*.

Recordando a la analogía de *Cámara Fotográfica*, las transformaciones de *viewport* corresponden al paso donde se decide el tamaño final de la fotografía que equivale a definir un área de la pantalla donde la imagen será dibujada. Hay que tomar en cuenta que para este paso ya todos los vértices han sufrido transformaciones de vista, de modelado y de proyección, y que los objetos que salen del volumen de visión han sido recortados y eliminados.

El sistema de ventanas, y no *OpenGL*, es el responsable del manejo de ventanas en la pantalla. Pero el *viewport* define un área dentro de una ventana para dibujar una escena. El comando `glViewport()` de *OpenGL* define el tamaño de un área dentro de una ventana.

3.2.4 Construcción de primitivas geométricas.

Los objetos geométricos complejos que genera *OpenGL* se crean a partir de primitivas básicas, estas primitivas están definidas por un conjunto de uno o más vértices. Un vértice define un punto, el punto final de un segmento de línea o alguna esquina de un polígono. Todas estas primitivas gráficas están descritas en términos de las coordenadas de sus vértices.

A continuación se dan algunas definiciones para cada una de las primitivas de *OpenGL.*

Puntos: Un punto se representa por un conjunto de coordenadas, dicho conjunto define a un vértice, los vértices pueden definir puntos de dos o tres dimensiones según el número de parámetros que contengan. *OpenGL* siempre trabaja en 3D y cuando se requiere graficar algo en 2D *OpenGL* asigna a la componente (z) un valor igual a cero.

Líneas: En *OpenGL* se trabaja con *segmentos de línea*, ya que en términos matemáticos una línea es aquella que se extiende hasta el infinito en ambas direcciones. Una línea siempre necesitará especificar las coordenadas de sus dos extremos o puntos finales. También se pueden especificar series de líneas que forman Figuras abiertas o cerradas.

Polígonos: Los polígonos son áreas encerradas dentro de un conjunto cerrado de segmentos de líneas, donde los segmentos de líneas son especificados por sus vértices en los extremos de cada segmento. Los polígonos son típicamente dibujados como superficies sólidas, es decir los pixeles en el interior del polígono son coloreados para dar una apariencia de estar relleno, pero también puede ser representados sólo dibujando sus contornos.

OpenGL hace dos restricciones importantes para definir a un polígono. La primer restricción es que los polígonos de *OpenGL* no pueden intersectarse en sus aristas. La segunda es que los polígonos deben ser convexos, esto significa que no pueden estar dentados, para comprobar que el polígono sea convexo, se unen con un segmento de línea a dos vértices cualesquiera del polígono, si el segmento esta totalmente en el interior, el polígono es convexo (*ver Figura 3.11*).

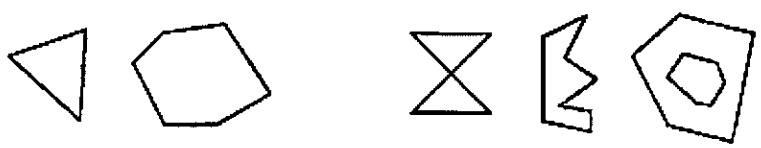


Figura 3.11 Polígonos validos y no validos por restricciones de *OpenGL*.

Curvas: Las curvas en *OpenGL* son series de pequeños segmentos de líneas que se aproximan a la curva deseada (*ver Figura 3.12*).

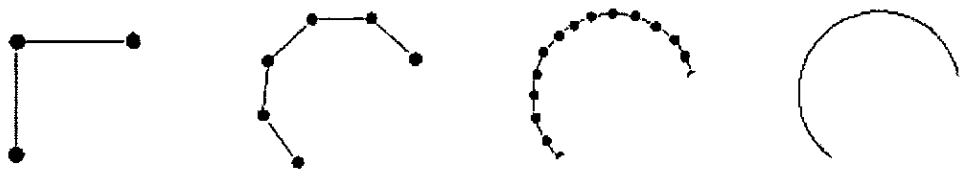


Figura 3.12 Las curvas en *OpenGL* se representan con series de segmentos de líneas.

Especificación de los vértices: En *OpenGL*, todos los objetos geométricos son descritos como un conjunto ordenado de vértices. Para especificar un vértice en *OpenGL* se usa el comando:

```
glVertex*();
```

En forma general se especifica así:

```
void glVertex{234}{sifd}[v](TYPEcoord);
```

Las primeras llaves indican que se deben especificar el número de coordenadas o vértices que utilizaremos. Las siguientes llaves indican que se debe especificar el tipo de datos de las coordenadas, es decir, (flotante doble "*double*", flotante "*float*", entero "*int*", entero corto "*short*"). La [v] indica que las coordenadas están contenidas dentro de un vector o arreglo.

La siguiente lista muestra todas las posibles formas de especificar un vértice en *OpenGL*:

```
void glVertex2d( GLdouble x, GLdouble y );
void glVertex2f( GLfloat x, GLfloat y );
void glVertex2i( GLint x, GLint y );
void glVertex2s( GLshort x, GLshort y );
void glVertex3d( GLdouble x, GLdouble y, GLdouble z );
void glVertex3f( GLfloat x, GLfloat y, GLfloat z );
void glVertex3i( GLint x, GLint y,GLint z );
void glVertex3s( GLshort x, GLshort y,GLshort z );
void glVertex4d( GLdouble x, GLdouble y, GLdouble z, GLdouble w );
void glVertex4f( GLfloat x, GLfloat y, GLfloat z, GLfloat w );
void glVertex4i( GLint x, GLint y, GLint z, GLint w );
void glVertex4s( GLshort x, GLshort y, GLshort z, GLshort w );
```

```
void glVertex2dv( const GLdouble *v );
void glVertex2fv( const GLfloat *v );
void glVertex2iv( const GLint *v );
void glVertex2sv( const GLshort *v );
void glVertex3dv( const GLdouble *v );
void glVertex3fv( const GLfloat *v );
void glVertex3iv( const GLint *v );
void glVertex3sv( const GLshort *v );
void glVertex4dv( const GLdouble *v );
void glVertex4fv( const GLfloat *v );
void glVertex4iv( const GLint *v );
void glVertex4sv( const GLshort *v );
```

Después de conocer como se especifican los vértices, es necesario saber como crear primitivas a partir de esos vértices. En *OpenGL* existen dos comandos: `glBegin()` y `glEnd()` los cuales delimitan donde empieza y termina una o varias primitivas, es decir entre estas dos funciones se deben especificar todos los vértices de esta o estas primitivas a través de llamadas a `glVertex()`. La función `glBegin()` tiene un argumento, el cual sirve para determinar el tipo de primitivas y el orden en las primitivas serán construidas.

La declaración de la función `glBegin()` es la siguiente:

```
void glBegin(Glenum mode);
```

Donde*mode* especifica el tipo de primitivas que serán creadas (para darse una mejor idea, ver la *Figura 3.13).* Los valores que puede tener el parámetro *mode* son los siguientes:

Valor	**Tipo de Primitiva**
GL_POINTS	Puntos individuales.
GL_LINES	Pares de vértices que crean segmentos de líneas individuales.
GL_LINE_STRIP	Serie de segmentos de líneas conectadas entre sí, formando una figura abierta.

GL_LINE_LOOP	Serie de segmentos de líneas conectadas entre sí, formando una figura cerrada, uniendo el primer vértice con el último.
GL_TRIANGLES	Conjuntos de tres vértices que forman triángulos individuales.
GL_TRIANGLE_STRIP	Serie de vértices que forman un conjunto de triángulos conectados entre sí.
GL_TRIANGLE_FAN	Serie de vértices que forman un conjunto de triángulos conectados entre sí y que tienen como al primer vértice especificado como un vértice común a todos los triángulos.
GL_QUADS	Serie de vértices que forman cuadros individuales.
GL_QUAD_STRIP	Serie de vértices que forman un conjunto de cuadros conectados entre sí.
GL_POLIGON	Conjuntos vértices que forman un solo polígono convexo uniendo el primer vértice con el último.

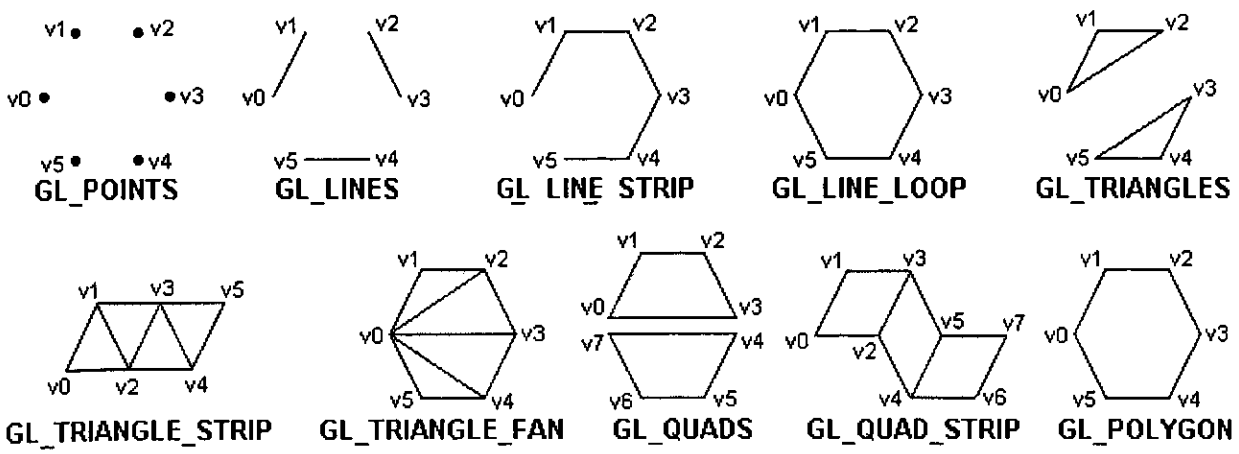


Figura 3.13 Tipos de primitivas en *OpenGL*.

La información más importante de los vértices son sus coordenadas, que como ya se vio son especificadas por el comando `glVertex()`. Se pueden añadir también otros datos relacionados a los vértices como son: color, normales, coordenadas de texturas, etc. Para especificar estos atributos, existen comandos validos para contenerse entre `glBegin()` y `glEnd()`.

La siguiente es una lista de los comandos más importantes:

Comandos	Propósito
`glVertex()`	Especifica las coordenadas de un vértice.
`glColor()`	Especifica las componentes de *RGB* para establecer el color actual.
`glIndex()`	Especifica el color actual a partir de una paleta de colores.
`glNomal()`	Especifica las coordenadas de una normal.
`glCallList()`	Invoca a los comandos contenidos en una lista.
`glTexCoord()`	Especifica las coordenadas de textura.
`glMaterial()`	Especifica las propiedades de material .

Otros comandos que determinan otras propiedades asociadas a las primitivas son:

```
void glPointSize(GLfloat size);
```

Este comando especifica el diámetro de los puntos.

```
void glLineWidth(GLfloat width);
```

Este comando especifica el grueso de los segmentos de línea.

Un polígono tiene dos lados o caras: la de frente y la trasera, y puede presentarse de diferentes formas dependiendo de que lado se esté observando. Esto permite ver al mismo objeto de dos diferentes formas. Por convención el frente y la parte trasera son dibujadas de la misma forma, pero esto se puede controlar utilizando el comando:

```
void glPolygonMode(GLenum face, Glenum mode);
```

Donde *face* especifica la cara o las caras a modificar, los valores que puede tomar son:

Valor	Propósito
GL_FRONT	Modificar sólo a la cara del frente.
GL_BACK	Modificar sólo a la cara trasera.
GL_FRONT_AND_GL_BACK	Modificar ambas caras.

Donde *mode* especifica la forma en que el polígono se presentará, es decir como una serie de puntos, como una malla hueca, o como un polígono sólido. Los valores que puede tomar *mode* son:

Valor	**Propósito**
GL_POINT	Serie de puntos.
GL_LINE	Malla hueca.
GL_FILL	Polígono sólido.

El *STFVI 3D* utiliza la mayoría de los comandos de *OpenGL* anteriormente descritos para generar los mapas de *Curvas de Nivel* y de *MDE* en sus formatos diferentes. Por ejemplo el mapa de *Curvas de Nivel* se genera usando el modo GL_LINE_LOOP del comando `glBegin()` para indicar que los vértices se unirán formando una serie de segmentos de líneas conectadas entre sí, formando una figura cerrada, uniendo el primer vértice con el último. El resultado se puede observar en la *Figura 3.14*.

Figura 3.14 Mapa de *Curvas de Nivel* utilizando el modo GL_LINE_LOOP.

El mapa del *MDE* se genera usando el modo GL_QUADS del comando `glBegin()` el cual indica que los vértices formarán cuadros independientes. La malla resultante se puede ver como una malla hueca utilizando GL_LINE (*ver Figura 3.15*) o de polígonos rellenos utilizando GL_FILL (*ver Figura 3.16*) como parámetro del comando `glPolygonMode()`.

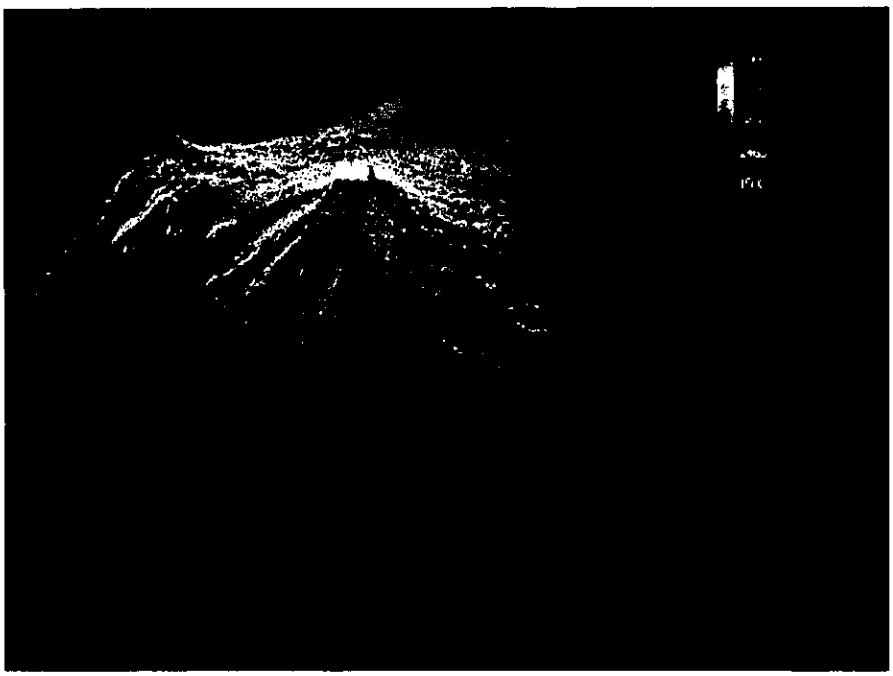


Figura 3.15 Mapa de *Curvas de Nivel* utilizando el modo GL_QUADS y GL_LINE.

Figura 3.16 Mapa del *MDE* utilizando el modo GL_QUADS y GL_FILL.

3.2.5 Percepción del Color, Modelo de Color *RGB* e Interpolación de Colores.

Percepción del Color.

La completa oscuridad es raramente experimentada por alguien. De hecho, la mayor parte del tiempo el ambiente en el cual vivimos se encuentra en algún grado constantemente bañado con luz y se ha vuelto algo tan común que raramente uno se pregunta. ¿Qué es la luz?

La luz se puede entender o interpretar de dos formas diferentes, como una onda, o como un flujo de partículas. La última interpretación ve a la luz como una corriente de partículas llamadas fotones sin masa, que se mueven a una velocidad constante. El atributo más importante de un fotón es su frecuencia, ya que esta es proporcional a la energía del fotón. En física los fotones son útiles para explicar una multitud de fenómenos (tales como la interacción entre materia y luz). En gráficas por computadora, la propiedad más importante de la luz es su color, y el color se determina por la longitud de onda, por esto es que usamos la interpretación de que consideramos a la luz como una onda.

Las ondas en la luz son parte de un campo eléctrico y otro magnético. Cuando una región del espacio se baña con luz, estos campos cambian periódicamente cuando nos movemos de un punto a otro en este espacio. Si permanecemos es un punto, los campos también cambian periódicamente con el tiempo. La luz visible es así una pequeña parte del espectro electromagnético (*ver Figura 3.17*) que incluye ondas de radio, rayos ultravioleta, rayos infrarojos, rayos X, y otros tipos de radiación.

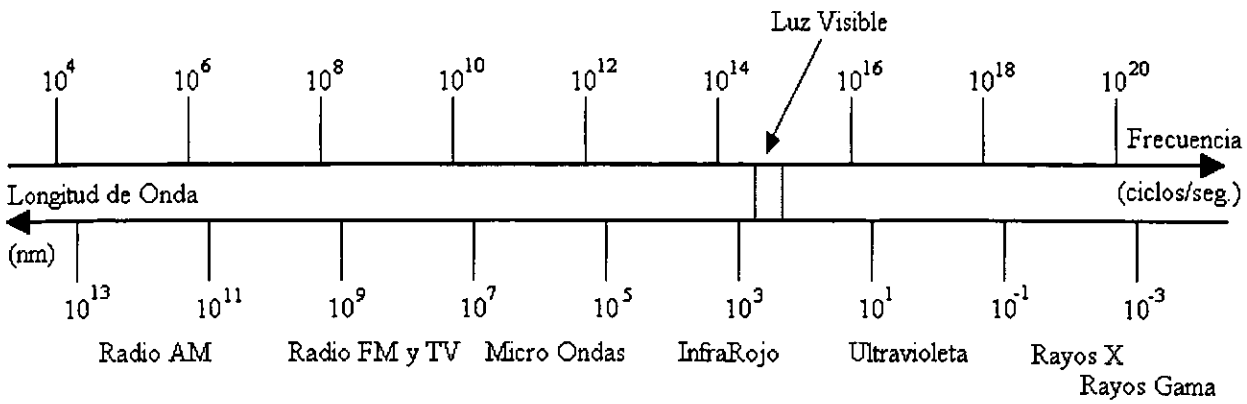


Figura 3.17 Espectro Electromagnético.

Las propiedades más importantes de una onda son su frecuencia *f*, su longitud de onda λ, y su velocidad. La luz viaja, obviamente, a la velocidad de la luz (en el vacío, esta es de $c \approx 3x10^{10} cm/s$). Las tres cantidades son relacionadas en la ecuación: $f\lambda = c$.

Es importante mencionar que la velocidad de la luz depende del medio en que esta viaja. Por ejemplo cuando la luz viaja del vacío al aire y después viaja a través de un cristal, la velocidad de la luz disminuye (en el cristal, la velocidad de la luz es de alrededor de $0.65c$). Su longitud de onda también disminuye, pero su frecuencia se mantiene constante. Sin embargo, se acostumbra relacionar a los colores con la longitud de onda y no con la frecuencia. Debido a que la luz visible tiene longitudes de onda muy pequeñas, la unidad conveniente para referirse a estas son los nanómetros ($1\,nm = 10^{-9}\,m$).

Los rangos de la luz visible van de los 400 nm a los 700 nm aproximadamente y el se determina el color por la longitud de onda. Una longitud de onda de 420nm, por ejemplo, corresponde al color violeta puro, mientras que 620mn se percibe por el ojo humano como un rojo puro. Usando láseres especiales, es posible crear luces monocromáticas puras que consisten de un rayo de luz con una sola longitud de onda. La mayoría de las fuentes de luz, sin embargo, emanan una luz que es una mezcla de varias (o muchas) longitudes de onda, normalmente con una longitud de onda dominante.

Los colores del espectro que son más visibles para el ojo humano son el violeta (390-430), el azul (460-480), el cyan, el verde (490-530), el amarillo (550-580), el anaranjado (590-640), y el rojo (650-800). La luz blanca es una mezcla de todas las longitudes de onda. Pero esta luz blanca también puede regularse con otros importantes atributos como la intensidad, así podemos crear una luz gris a partir de una mezcla de todas las longitudes de onda pero a una baja intensidad.

Como se ha visto, la luz esta compuesta por fotones. Un fotón se caracteriza por su posición, dirección, frecuencia, longitud de onda y energía. Los fotones con longitudes de onda entre 400 nanómetros (violeta) y 720 nanómetros (rojo) cubren los colores del espectro visible, formando los colores del arco iris (violeta, índigo, azul, verde, amarillo anaranjado, rojo). Sin embargo el ojo percibe muchos más colores de los que se encuentran en el arco iris, esto es posible porque el ojo percibe una mezcla de fotones de diferente longitudes de onda. Las fuentes de luz se caracterizan por la distribución de longitudes de onda de los fotones emitidos.

El ojo humano percibe la luz que entra en él y cae en la retina, porque hay dos tipos de celdas fotosensibles. Estas celdas contienen pigmentos que absorben la luz visible y por esto nos dan la sensación de visión. El tipo de celdas rods (bastón), que son numerosas y se esparcen sobre toda la retina responden solo a la luz y la oscuridad. Estas son muy sensitivas ya que pueden responder a un solo fotón de luz. Hay entre 110,000,000 y 125,000,000 de este tipo de celdas en el ojo. El otro tipo son las celdas cono, que se localizan en una pequeña área de la retina (la fovea). Estas se aproximan a unas 6,400,000 celdas y son sensibles al color pero requieren de más intensidad de luz, en el orden de cientos de fotones.

Las celdas cono son muy sensibles al rojo, al verde y al azul (*ver Figura 3.18*). Cuando una mezcla de fotones entra por el ojo, las celdas cono en la retina registran grados diferentes de excitación dependiendo de los tipos de longitudes de onda que se encuentran en la mezcla de fotones. El ojo registra cada color por el nivel de excitación producido por la entrada de fotones en las celdas cono, el ojo puede percibir colores que no se encuentran en el espectro que produce un prisma o el arco iris. Por ejemplo, si una mezcla de fotones en rojo y azul excitan las celdas cono de la retina, el ojo registra el color magenta , que no se encuentra dentro del espectro del arco iris.

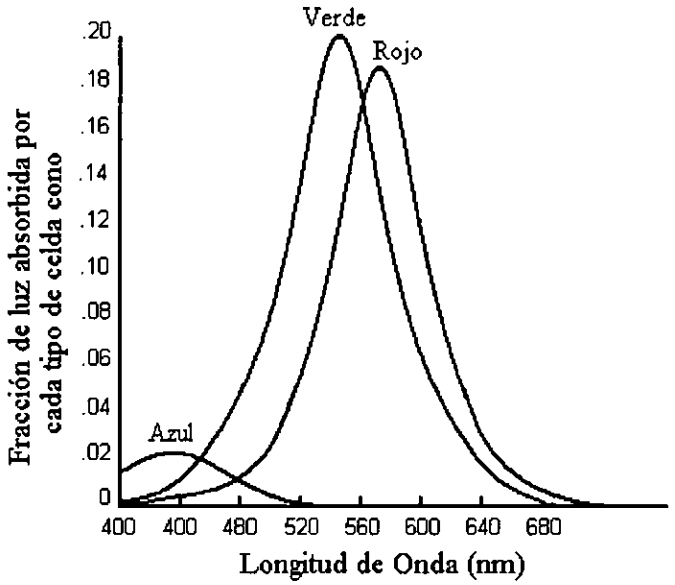


Figura 3.18 Sensitividad de las celdas cono.

Un monitor de computadora emula la composición de colores visibles por el ojo humano con la iluminación de pixeles, cada pixel de divide en componentes de rojo, verde y azul que excitan a las celdas cono sensibles al rojo, verde y azul.

El Modelo de Color *RGB*.

Existe una amplia variedad de Modelos de Color como el *CMYK* que se usa comúnmente en la industria de impresiones, los Modelos *HLS* y *HVS* son usados por muchos artistas ó el Modelo *RGB* (*RGB* por las siglas en ingles de *Red*, *Green* y *Blue*), que es el más popular en gráficos por computadora.

El Modelo de Color *RGB* es usado en la mayoría de los monitores de color con *CTR* (*tubo de rayos catódicos*) y gráficos *raster* en color, emplea un sistema de coordenadas cartesiano. El Modelo *RGB* se basa en la composición de tonos de colores a partir de mezclar los colores primarios: rojo, verde y azul con diferentes proporciones.

Un color primario es aquel que no puede ser formado a partir de los otros colores contenidos en el modelo de color. Los colores primarios sirven como una base para mezclar y crear a todos los otros colores en el modelo de color. Un color creado a partir de la mezcla de dos colores primarios en un modelo de color es un color secundario en este modelo.

En el modelo de color *RGB*, los tres colores primarios son el rojo, verde y el azul. Se combinan dos de estos para crear colores secundarios. Por ejemplo el magenta se forma a partir de la combinación (*R*+*B*), el cyan a partir de (*B*+*G*) y el amarillo a partir de (*R*+*G*).

La *Figura 3.19* muestra un cubo *RGB*, que contiene en cada vértice una combinación extrema en sus componentes primarios rojo, verde y azul, a partir de estas combinaciones se generan los colores amarillo, magenta, cyan, blanco y negro, la diagonal que va del vértice que representa al color negro hasta el color blanco contiene todos los tonos de gris.

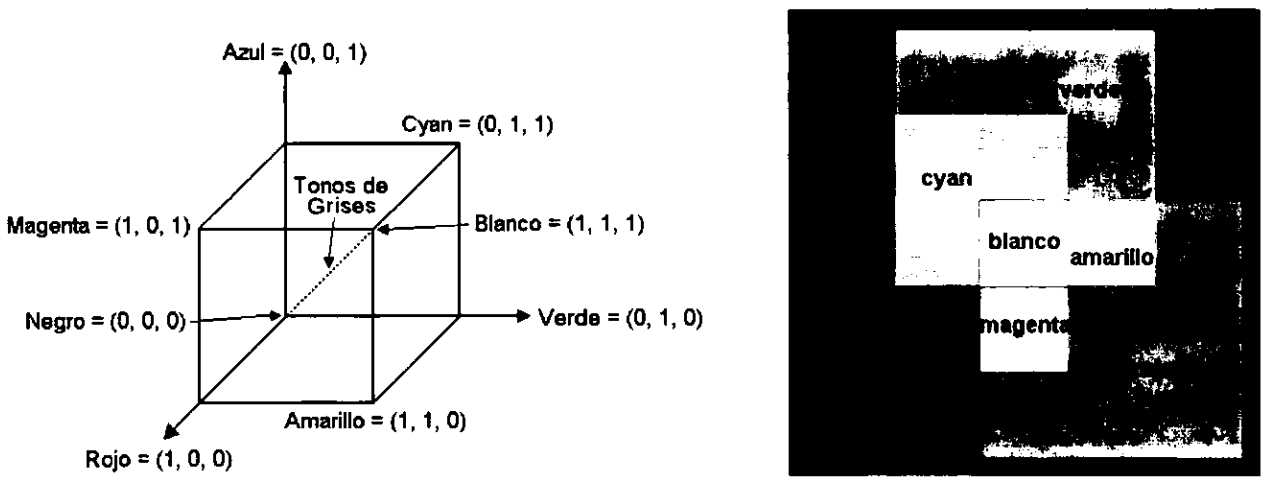


Figura 3.19 Cubo *RGB* y representación de colores secundarios a partir de colores primarios en el modelo *RGB*.

El modelo de color más importante que utiliza *OpenGL* es el *RGB*. Para indicar un color se utiliza el comando `glColor()`, el cual tiene tres parámetros para indicar el tono de las componentes de Rojo, Verde y Azul.

En el *STFVI 3D* se implementó un mapa de colores el cual esta relacionado con la escala de elevaciones o altimetría de las topografías de los volcanes. Este mapa de colores se crea a partir de tres ecuaciones las cuales defines los segmentos de una parábola y de dos rectas. Estas ecuaciones sirven para determinar las cantidades de *Rojo*, *Verde* y *Azul* que se deben aplicar a cada punto en la topografía.

En la *Figura 3.20* se muestran las curvas y rectas que definen los valores para las componentes de *Rojo*, *Verde* y *Azul* con respecto a la altura de un punto en la topografía.

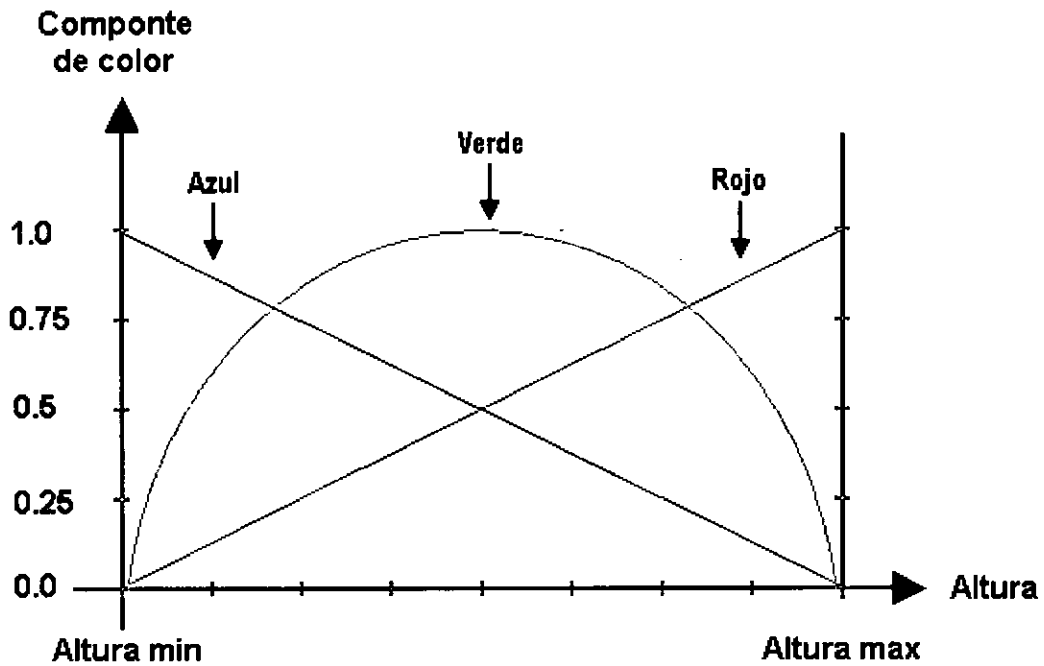


Figura 3.20 Parábola y rectas que definen los valores para las componentes *RGB* para los puntos en los mapas.

El *STFVI 3D* muestra una *Barra de Escala de Elevaciones* (*ver Figura 3.21*) la cual muestra una relación entre la altura y una combinación *RGB*.

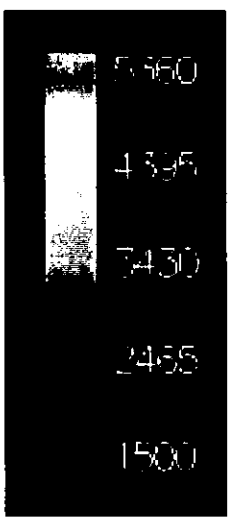


Figura 3.21 *Barra de Escala de Elevaciones* del *STFVI 3D*.

3.2.6 Cálculo del *Vector Normal* en polígonos y su relación con la iluminación de escenas 3D.

Vector Normal.

Un *Vector Normal* o simplemente normal es un vector que apunta en una dirección que es perpendicular a una superficie. Para una superficie plana, una dirección perpendicular es la misma para cada punto en la superficie, pero para una superficie curva en general, la dirección de la normal puede ser diferente para cada uno de los puntos en la superficie.

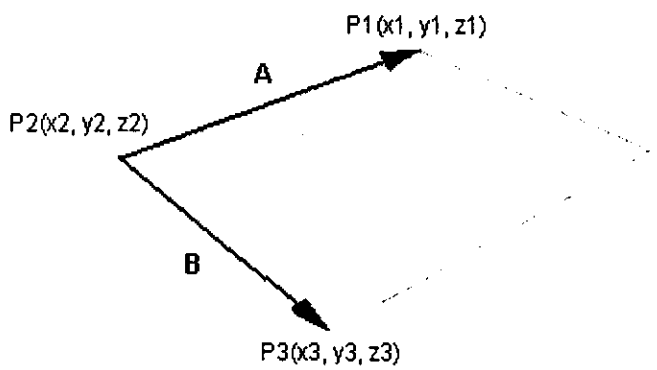
Con *OpenGL*, se puede especificar una normal para cada uno de los polígonos o para cada uno de los vértices. Los vértices de un mismo polígono pueden compartir la misma normal (para una superficie plana) o tener diferentes normales (para una superficie curva). Pero no se pueden asignar normales en otro lugar que no sea un vértice.

Los *Vectores Normales* de un objeto geométrico definen la orientación de su superficie en el espacio, en particular, su orientación relativa a las fuentes de iluminación. OpenGL usa estos vectores para determinar la cantidad de luz que recibe el objeto en cada uno de sus vértices.

Como se mencionó en la sección anterior el cálculo de los *Vectores Normales* de un polígono o vértice es muy importante para el uso adecuado de luces y texturas en un objeto gráfico tridimensional.

El procedimiento a seguir, para obtener el *Vector Normal* de un vértice de un polígono es el siguiente:

1.- Encontrar tres vértices vecinos.



2.- A partir de estos vértices calcular los vectores A y B.

$$A = [ax \quad ay \quad az]$$
$$B = [bx \quad by \quad bz]$$

$$ax = x1 - x2$$
$$ay = y1 - y2$$
$$az = z1 - z2$$

$$bx = x3 - x2$$
$$by = y3 - y2$$
$$bz = z3 - z2$$

3.- Calcular el producto cruz (*AXB*).

$$(AXB) = [ax \quad ay \quad az] \quad X \quad [bx \quad by \quad bz]$$

$$[ax\ ay\ az]\ X\ [bx\ by\ bz] = (ay\ bz - az\ by)(az\ bx - ax\ bz)(ax\ by - ay\ bx)$$

$$nx = ay\ bz - az\ by$$
$$ny = az\ bx - ax\ bz$$
$$nz = ax\ by - ay\ bx$$

La siguiente función realiza los pasos 2, 3 anteriormente descritos, es decir calcular los vectores A y B a partir de los puntos *P1*, *P2* y *P3*, y calcular el producto cruz de estos vectores (*AXB*).

```
void ProductoCruz(void)
{
   GLfloat A[3], B[3];
   int i;
   for(i=0; i<3; i++)
   {
      A[i]=P1[i]-P2[i];
      B[i]=P3[i]-P2[i];
   }
   Normal[0]=A[1]*B[2]-A[2]*B[1];
   Normal[1]=A[2]*B[0]-A[0]*B[2];
   Normal[2]=A[0]*B[1]-A[1]*B[0];
}
```

4.- Normalizar el *Vector Normal* resultante en el paso 3 en un *Vector Normal Unitario.*

$$long = \sqrt{nx^2 + ny^2 + nz^2}$$

$$Nx = \frac{nx}{long} \qquad Ny = \frac{ny}{long} \qquad Nz = \frac{nz}{long}$$

OpenGL realiza esta normalización automáticamente cuando se utiliza el siguiente comando:

```
glEnable(GL_NORMALIZE);
```

5.- Especificar el *Vector Normalizado* para un vértice común ó para todo el polígono.

```
glBegin(GL_POLIGON)

   glNormal3f(Nx, Ny, Nz);
   glVertex3f(x1, y1, z1);
   glVertex3f(x2, y2, z2);
   glVertex3f(x3, y3, z3);

glEnd();


glBegin(GL_POLIGON)

   glNormal3f(Nx1, Ny1, Nz1);
   glVertex3f(x1, y1, z1);

   glNormal3f(Nx2, Ny2, Nz2);
   glVertex3f(x2, y2, z2);

   glNormal3f(Nx3, Ny3, Nz3);
   glVertex3f(x3, y3, z3);

glEnd();
```

El *Mapa con Iluminación* al cual se le aplica el cálculo de los *Vectores Normales* de sus vértices se puede observar en la *Figura 3.22*.

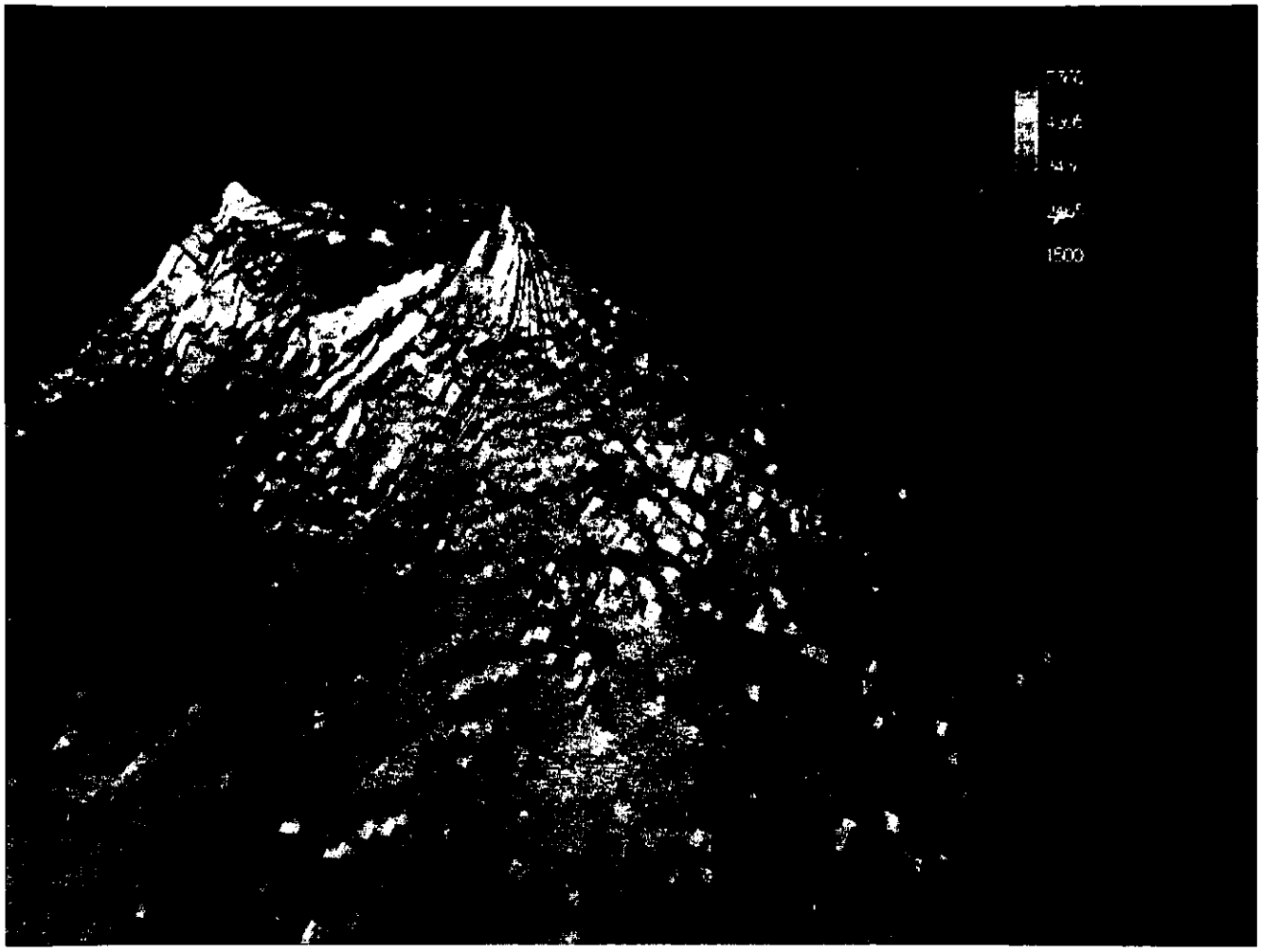


Figura 3.22 ***Mapa del MDE con Iluminación.***

3.3 Uso de "*Listas de Despliegue*" de *OpenGL* para optimizar el tiempo de dibujo.

Una lista es simplemente un conjunto de comandos de *OpenGL* que han sido almacenados o encapsulados para su subsecuente ejecución. El comando `glNewList()` inicia la creación de una lista de despliegue, y el comando `glEndList()` termina esta. Con pocas excepciones casi todos los comandos de *OpenGL* pueden contenerse dentro de una lista. Para llamar o invocar a los comandos de una lista, se ejecuta el comando `glCallList()` que tiene como uno de sus parámetros el nombre de la lista invocada.

Debido al gran número de cálculos realizados en la creación de los objetos geométricos y sus parámetros asociados como color, materiales, estados de iluminación, vectores normales y transformaciones geométricas iniciales de rotación, translación y escalamiento en un escenario tridimensional, es necesario implementar métodos para optimizar dichos cálculos.

El *STFVI 3D* utiliza el método de *"Listas de Despliegue" (Display List)* de *OpenGL* para optimizar cálculos de transformaciones, iluminación y otros estados constantes de la escena. Este método es uno de los dos con los que cuenta *OpenGL* para la creación y despliegue de los objetos geométricos, el otro método es conocido como "*Modo inmediato*" que también es utilizado en el desarrollo de este sistema. La decisión de utilizar uno u otro método dependerá de algunas razones que a continuación se explicarán más detalladamente.

El método de "*Listas de Despliegue*" se utiliza preferentemente cuando se van a crear objetos geométricos que no cambiarán en su forma o estructura ni en algunos estados asociados a sus vértices como color o material asignado, estado de iluminación o vectores normales, es decir las coordenadas geométricas de cada vector de los objetos permanecerán constantes así como algunos estados asociados a estos como normales, color o material, si se aplicará iluminación o no a ese vértice o el vector normal asociado a este durante toda la ejecución del sistema, también se pueden añadir transformaciones como rotación, translación y escalamiento a un polígono para darle una orientación, posición y un tamaño inicial con este método pero no así a cada vértice.

Una de las características de las listas es que estas una vez creadas ya no pueden ser alteradas (la filosofía de las listas es optimizar cálculos) ya que el estado resultante de una lista será permanente durante la ejecución del programa.

La ventaja de utilizar este modelo radica en que el cálculo de los cambios de estados o transformaciones hacia un objeto dentro de una lista son calculados una sola vez y el estado final de estos cálculos nunca cambia y solo son llamados cuando se necesiten a lo largo de la ejecución del programa. Esto ahorra una gran cantidad de recursos de cómputo y da una respuesta más rápida en el *rendering* y en la respuesta de interactividad con el usuario.

El método de "*Modo Inmediato*" se requiere cuando se desea realizar un cambio en la forma interna de alguna geometría, alterando la posición de alguno de sus vértices, también cuando se realizan transformaciones de rotación, translación escalamiento de uno o todos los objetos en la escena.

El *STFVI 3D* hace uso de las listas debido a que los mapas que maneja no cambian las coordenadas de sus vértices ni la composición de la topografía que representan ni tampoco la interpolación de colores asociada a la altimetría de los mapas. Es por esto que las listas resultan de gran ayuda para la optimización de cálculos, sino se utilizara la técnica de listas los cálculos para el despliegue de los mapas se recalcularían cada vez que se redibujara la escena. Pero también hace uso del método de "*Modo Inmediato*" cuando aplica transformaciones de rotación, translación, escalamiento o cambio de tipo de proyección a todos la topografía.

Capítulo IV

Comparación del *STFVI 3D* con otros sistemas similares.

4.1 Comparación con *RVOL* desarrollado por el *Centro Nacional de Prevención de Desastres* (*CENAPRED*).

Como ya se ha mencionado en la introducción y en el capítulo I de este trabajo, *el STFVI 3D* se crea a partir de la necesidad de sustituir al antiguo sistema *RVOL* por "*Riesgo VOLcánico*" (*ver Figura 4.1*) desarrollado por el *CENAPRED*. El *RVOL* se convirtió en un sistema obsoleto ya que presentaba muchas limitaciones de tipo gráfico y de portabilidad entre sistemas.

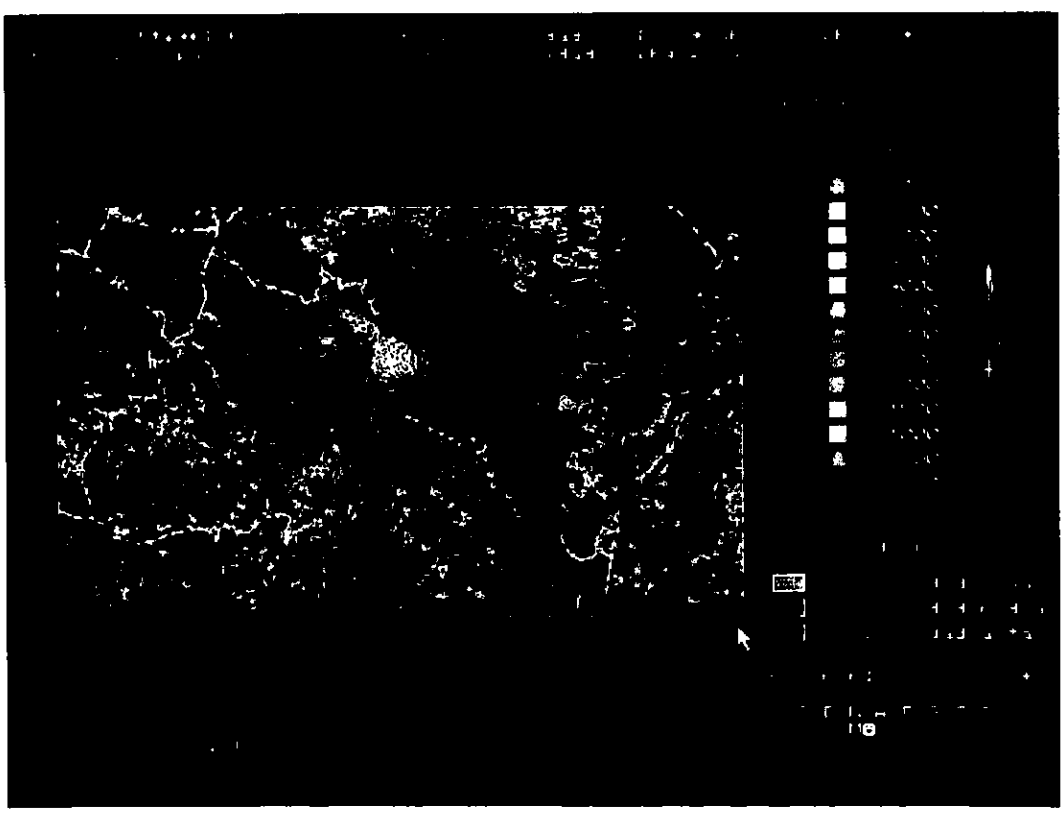


Figura 4.1 Vista de *Curvas de Nivel* en el sistema *RVOL*.

Realizar actualizaciones del *RVOL* ya no resultaba práctico porque sus módulos gráficos fueron desarrollados con las funciones básicas de gráficos propias del lenguaje C, en particular las funciones de gráficos compatibles con los compiladores de *Borland C++ 3.0* y *Turbo C 2.0*, las cuales son muy limitadas y solo son compatibles para máquinas *PC* con *MS-DOS* o *Windows*.

El *STFVI 3D* solo rescata el Modelo Físico, pero adecua los datos de salida para que puedan ser graficados en tres dimensiones.

Las ventajas que presenta el *STFVI 3D* sobre el *RVOL* se listan a continuación:

- Es un sistema multiplataforma (trabaja sobre ambientes *UNIX*, *LINUX* y *Windows*).
- Se desarrolló utilizando Bibliotecas Gráficas de alto rendimiento y calidad en gráficos (*OpenGL*, *GLUT*, *MUI*).
- Es altamente interactivo, permite realizar Transformaciones Geométricas de Rotación, Translación, Escalamiento y cambio del Modelo de Proyección de los diferentes mapas y también permite la selección de puntos de inicio de las trayectorias de los flujos.
- Utiliza técnicas de optimización para el rendimiento del tiempo de procesamiento, tiempo de despliegue de gráficos y almacenamiento de datos.
- Visualiza varios modelos de mapas topográficos como son el mapa *de Curvas de Nivel*, el mapa del *Modelo Digital de Elevaciones MDE* de malla con polígonos sólidos, con interpolación de materiales, etc.
- Maneja un *Mapa de Colores* para la *Interpolación de Colores* y de *Interpolación de Materiales con Iluminación*.

4.2 Comparación con *FLOW 3D* desarrollado en *la Universidad Estatal de Búfalo, Nueva York.*

El sistema *Flow3D* se desarrolló en el *Departamento de Geología de la Universidad de Búfalo en Nueva York*, por *Thomas P. Kover* bajo la dirección del Dr. *Michael F. Sheridan* (la página del *Dr. Sheridan* se puede consultar en la siguiente dirección de Internet: http://www.eng.buffalo.edu/~mfs/) durante 1994-1995. *Flow3d* (*ver Figura 4.2*) es un sistema que simula flujos volcánicos en tercera dimensión de manera interactiva.

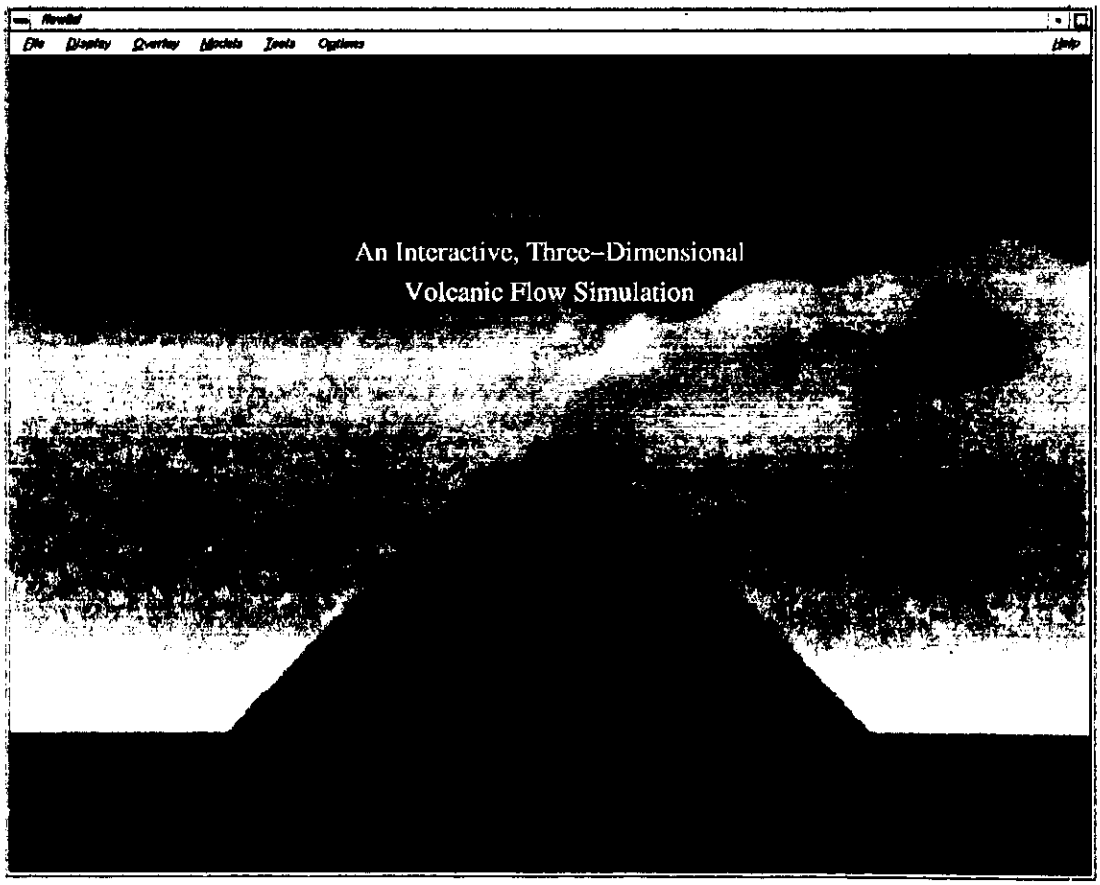


Figura 4.2 Pantalla de presentación de *Flow3D*.

El *Flow3D* es un sistema que al igual que el *STFVI 3D* utiliza un modelo físico gravitacional pero sus simulaciones no parten de los datos de un *Modelo Digital de Elevaciones* definido en una malla rectangular uniforme equiespaciada, en su lugar toma como base un mapa de triángulos irregulares o *TIN* (*Triangulated Irregular Network) (ver Figura 4.3*) como se le conoce en el lenguaje de los *SIGs*.

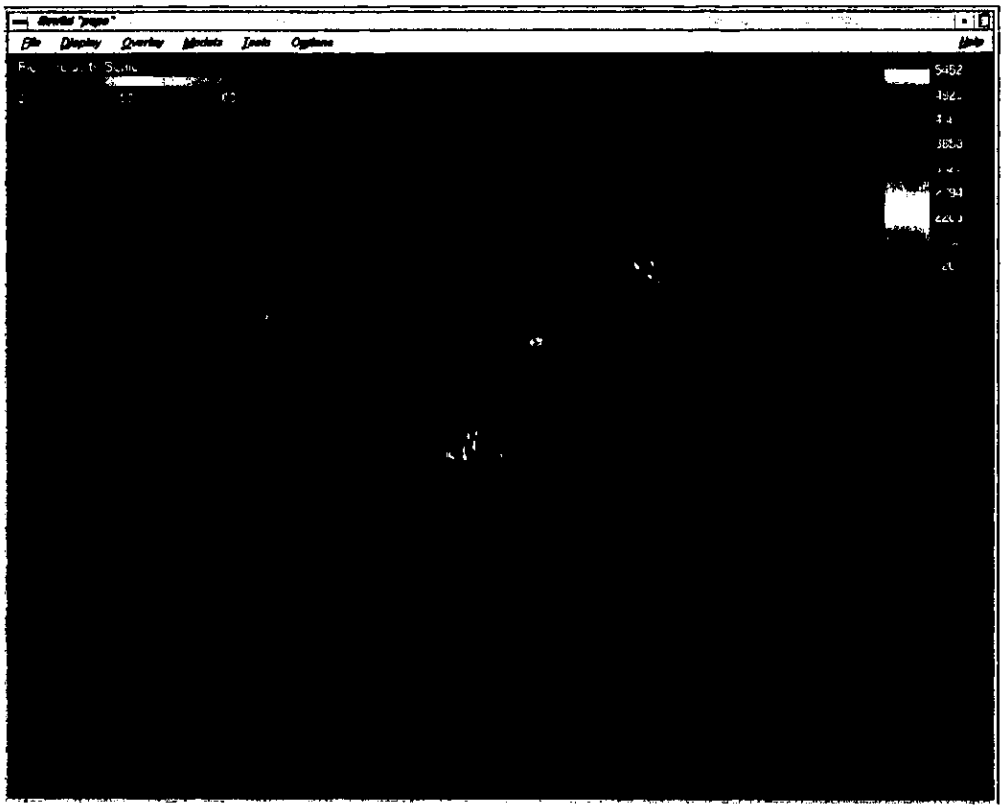


Figura 4.3 Mapa *TIN* del Volcán Popocatepelt visto con *Flow3D*.

Flow3D genera sus propios mapas *TIN* y realiza los siguientes pasos para obtenerlos: Primero lee un archivo con formato *dxf* (los archivos *dxf* se generan con aplicaciones como *AutoCad*) el cual contiene los datos de las curvas de nivel de un volcán. Una vez que se carga el archivo *dxf Flow3D* lo convierte en un archivo de tipo vector con valores de las coordenadas (x, y, z) de cada punto. Posteriormente aplicando algoritmos de triangulación (triangulación por *Delauney*) de a los puntos del archivo vector se genera el archivo *TIN* en modo binario. La aproximación del *TIN* a la topografía real del volcán depende en gran medida de la calidad del muestreo realizado en el mapa de curvas de nivel, es decir la cantidad y distribución de puntos en cada curva. Las mejores digitalizaciones son las que consideran un mayor número de puntos cuando las curvas son más pronunciadas, y así obtener una mejor resolución de las zonas que presentan más cambios en la topografía.

La ventaja de utilizar una malla rectangular uniforme en lugar de un *TIN* es que es que es una estructura más sencilla de manejar y de almacenar. Un *TIN* tiene que almacenar tanto las coordenadas de cada uno de los puntos así como un conjunto de índices de vértices para formar los triángulos. Una malla rectangular solo tiene que almacenar en una matriz bidimensional los datos de las alturas de cada punto muestreado sobre la topografía. Por otro lado el proceso de simulación de trayectorias de flujos es más lento en un *TIN* que en una malla rectangular porque el acceso a los datos de los vértices de los planos triangulares contiguos es más complejo por tratarse de una malla irregular.

Flow3D dispone de una completa interfaz de usuario pero que está desarrollada para trabajar solo sobre estaciones de trabajo. Su barra de menúes se muestra en la *Figura 4.4*.

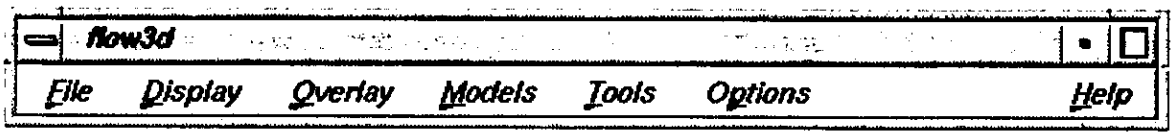


Figura 4.4 Barra de menúes de *Flow3D*.

Esta barra contiene un conjunto de menúes con diversas funciones las cuales a continuación se explican más detalladamente:

Menú: File

- ***Open*** — Abre archivos tipo *dxf*, *xyz* y *tin*.
- ***Edit text*** — Abre un archivo de texto con el editor de texto *JOT*.
- ***Delete*** — Borra un archivo.
- ***Import overlay image*** — Inserta un archivo de imagen para sobreposición.
- ***Save image as*** — Guarda la vista generada como un archivo de imagen.
- ***Print*** — Manda a imprime la vista generada.
- ***TIN info*** — Proporciona información acerca de archivos tipo *TIN*.
- ***Exit*** — Sale del sistema.

Como se puede observar el menú *File* es un menú típico para el manejo de archivos con operaciones para abrir, editar, eliminar, insertar, mostrar información y guardar archivos. También cuenta con la opción de mandar a imprimir vistas de simulaciones y mapas y de salir del sistema. Muchas de estas funciones no están disponibles en la interfaz del *STFVI 3D* pero en su lugar algunas de estas funciones se ejecutan desde la línea de comandos o de forma automática. Por ejemplo, la apertura de los archivos es realizada desde la línea de comandos, donde se especifica solo un prefijo común (generalmente el nombre de un volcán) de los archivos de *Curvas de Nivel*, del *MDE*, de configuración y el archivo opcional de alturas máximas y mínimas.

El *STFVI 3D* realiza la función de guardar archivos de forma automática. Por ejemplo al cargar archivos en formato texto, el sistema genera la versión binaria de estos archivos para optimizar el espacio de almacenamiento en futuras simulaciones. Los archivos con los datos de las derivadas parciales también son generados y guardados automáticamente por el sistema al igual que el archivo de registro de alturas máximas y mínimas necesarios para la generación de un mapa de colores para la escala de elevaciones.

El *STFVI 3D* no cuenta con un editor de archivos, pero hay que mencionar que el editor *JOT* que utiliza *Flow3D* no es un editor propio del sistema sino que es una de las aplicaciones que trae el Sistema Operativo *IRIX* para las estaciones del trabajo de *SGI*.

Es de esta forma que el *STFVI 3D* compensa algunas de las funciones del menú del *File* del *Flow3D*.

Menú: Display

- ***Slope map*** Mapa *TIN* con interpolación de colores por inclinación o por elevaciones.
- ***Network*** Muestra al mapa *TIN* hueco o sólido.
- ***Shading*** Habilita o deshabilita iluminación y materiales en el *TIN*.
- ***Elevation scale*** Muestra una escala de elevaciones en un mapa de colores.
- ***Flow traces*** Inicia el recorrido de los flujos.
- ***Flows status*** Muestra información acerca del estado de los flujos.
- ***Flow velocity scale*** Muestra escala de velocidad de flujos en un mapa de colores.
- ***Reset*** Retorna variables a estados iniciales (posición, escala, etc.).

El menú *Display* del *Flow3D* realiza funciones de habilitar o deshabilitar estados de despliegue del mapa *TIN*. Si se habilita el estado ***Slope map*** el mapa *TIN* se muestra con una interpolación de colores relacionada con el grado de inclinaciones en la topografía, si esta opción esta deshabilitada el mapa *TIN* se muestra con un mapa de colores relacionado con las elevaciones del mismo. El estado ***Network*** sirve para mostrar al mapa *TIN* como un malla de triángulos huecos o de triángulos sólidos y el estado ***Shading*** habilita o deshabilita la aplicación de iluminación y materiales en el *TIN*.

Como se vio en los capítulos anteriores el *STFVI 3D* realiza la selección de sus diferentes mapas con la ayuda de la interfaz *MUI* y del menú *pop-up* de *GLUT*. A diferencia del *Flow3D* el *STFVI 3D* cuenta con un número mayor de formatos de despliegue de mapas, por ejemplo el *Flow3D* no visualiza mapas de *Curvas de Nivel*, ni sobrepone una cuadricula sobre el mapa de polígonos sólidos para dar una apariencia de sombra y tampoco muestra un mapa con un solo material con iluminación.

La opción ***Elevation scale*** habilita una *Barra de Colores* la cual representa una relación entre la *Escala de Elevaciones* con el *Mapa de Colores* de la barra (*ver Figura 4.5*).

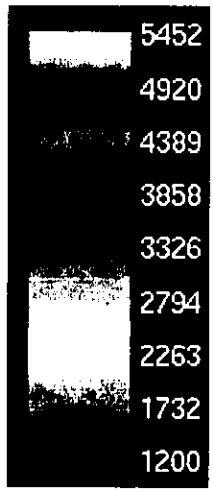


Figura 4.5 ***Barra de Escala de Elevaciones del Flow3D.***

El *STFVI 3D* también cuenta con una *Barra de Escala de Elevaciones* pero con un mapa de colores diferente, ya que su interpolación va del rojo al azul (*ver Figura 4.6*).

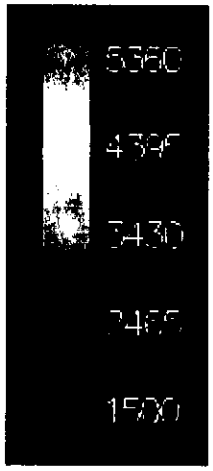


Figura 4.6 ***Barra de Escala de Elevaciones del STFVI 3D.***

La opción ***Flow velocity scale*** habilita una *Barra de Colores* la cual representa una relación entre la *Escala de Velocidad* de los flujos con el *Mapa de Colores* de la barra (*ver Figura 4.7*).

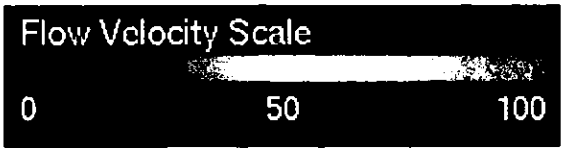


Figura 4.7 ***Escala de Velocidad*** **de flujos en** ***Flow3D.***

La opción ***Flows traces*** inicia la simulación y el despliegue de los flujos(*ver Figura 4.8*). Las características de la simulación de los flujos se deben inicializar previamente con la opción ***Flow*** del menú ***Model.***

Figura 4.8 Simulación de flujos en el Volcán Popocatepelt con ***Flow3D.***

Menú: Overlay

- ***Select*** Abre una ventana para sobreposición de archivo de imagen *ovl.*
- ***Clear all*** Elimina las sobreposiciones en los mapas.

La opción ***Select*** del menú ***Overlay*** muestra una ventana para seleccionar los archivos de imagen y determinar el modo de sobreponerlos en el mapa *TIN* (*ver Figura 4.9*). La opcion ***Clear all*** elimina todos las imágenes sobrepuestas y también elimina la interpolación de colores del *TIN* dejando en tonos de gises.

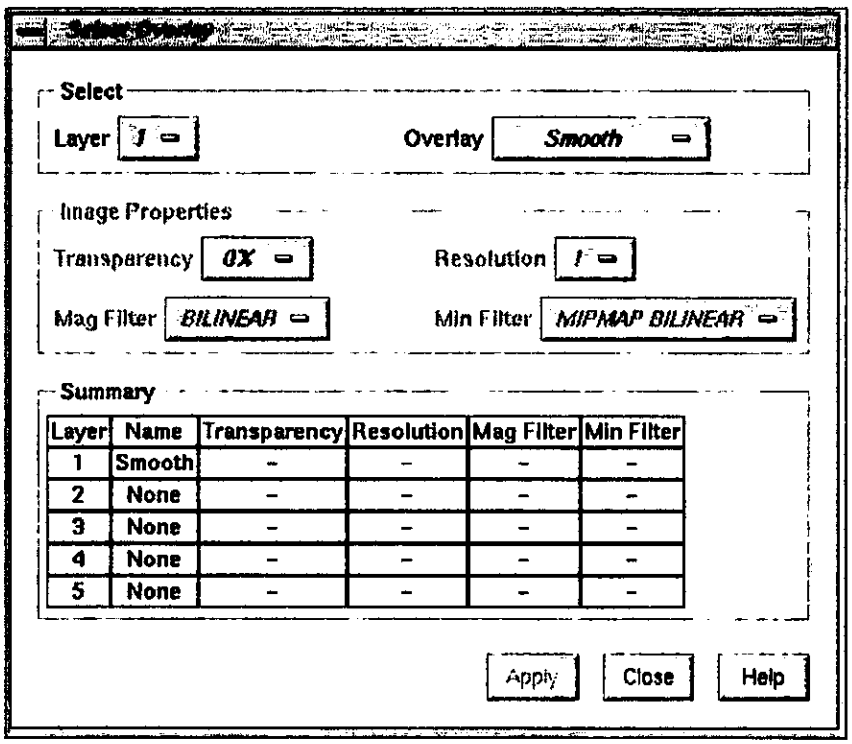


Figura 4.9 Ventana para la sobre posición de imágenes en el *TIN* del *Flow3D*.

Menú: Model

- ***Flow*** Abre interfaz para dar valores iniciales, características y punto de emisión de los flujos.

Este menú es muy importante porque abre una ventana de inicialización (*ver Figura 4.10*) de parámetros de coeficiente de fricción, coeficiente de viscosidad, coeficiente turbulente, velocidad inicial, número de flujos y un formato de área de emisión de los flujos.

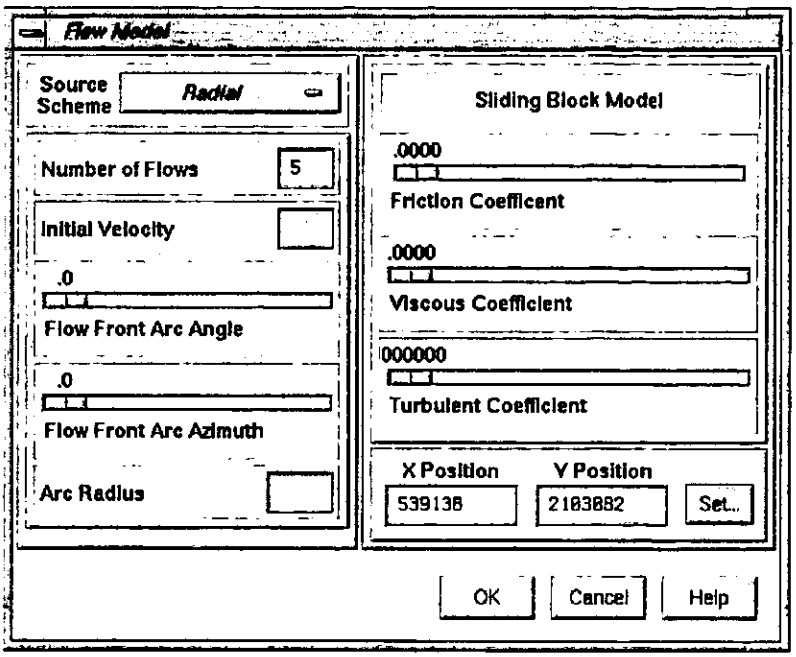


Figura 4.10 Ventana para inicialización de parámetros de los flujos en *Flow3D*.

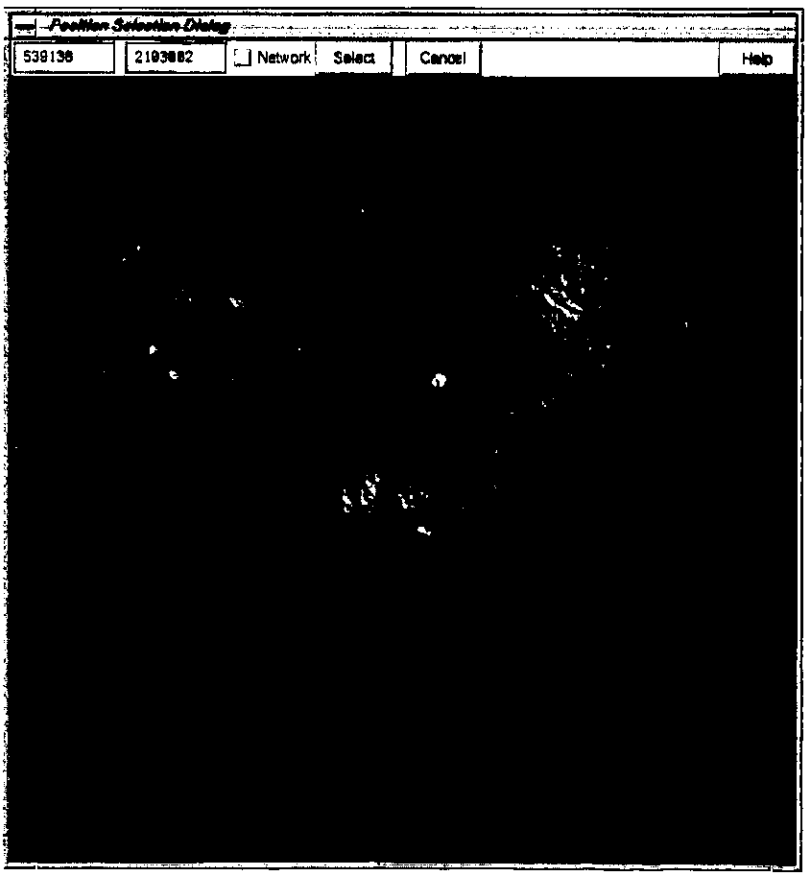


Figura 4.11 Ventana de selección de punto de emisión de flujos en *Flow3D*.

En la parte inferior derecha de la ventana de inicialización de parámetros de los flujos hay una sección para la selección de las coordenadas del punto o centro de emisión, estas coordenadas pueden darse manualmente o abriendo otra ventana para la selección de la posición en modo gráfico (*ver Figura 4.11*) en la cual muestra al *TIN* desde una vista aérea y con el ratón seleccionar un punto sobre el mapa. Es importante mencionar que este método limita la selección a un plano 2D, porque no es posible seleccionar un punto desde cualquier posición. El *STFVI 3D* si cuenta con esta posibilidad porque la selección del punto de emisión se realiza desde cualquier posición que tengan los mapas.

La *Figura 4.12* muestra información acerca del tiempo, distancia y velocidad máximas registradas por los flujos en las simulaciones de *Flow3D*.

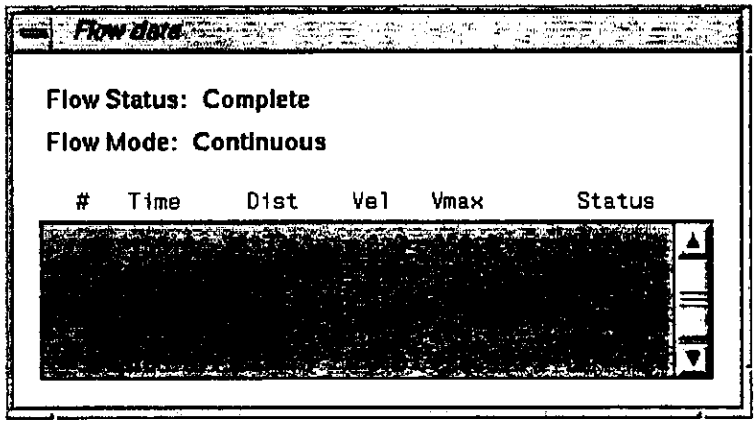


Figura 4.12 Venta que muestra información acerca de los flujos en *Flow3D*.

Conclusiones.

Después del desarrollo del presente trabajo se puede concluir que se ha cumplido satisfactoriamente el objetivo inicialmente planteado, el cual era crear un sistema que remplazara al antiguo sistema *RVOL* desarrollado por el *CENAPRED* superando las limitaciones que este presentaba.

El *STFVI 3D* cumple con los requerimientos de portabilidad entre plataformas, alta calidad y rendimiento en el despliegue de gráficos tridimensionales, alta interactividad y con un fácil mantenimiento para futuras modificaciones y versiones. Este sistema se ha instalado y probado satisfactoriamente en máquinas de tipo *WorkStation* con *UNIX* y de *PC* con Windows.

Las ventajas del *STFVI 3D* sobre el *RVOL* son evidentemente muy grandes y se puntualizan a continuación. Es un sistema multiplataforma que puede trabajar sobre ambientes *UNIX*, *LINUX* y *Windows*. Se desarrolló utilizando Bibliotecas Gráficas de *OpenGL*, *GLUT* y *MUI* que son bibliotecas de alto rendimiento y calidad en gráficos. Es un sistema altamente interactivo y permite realizar Transformaciones Geométricas de Rotación, Translación, Escalamiento, cambio del Modelo de Proyección de los diferentes mapas y también permite la selección de puntos de inicio de las trayectorias de los flujos. Utiliza técnicas de optimización para el rendimiento del tiempo de procesamiento, timpo de despliegue de gráficos y en el almacenamiento de datos. Visualiza varios formatos de mapas topográficos como son el mapa de *Curvas de Nivel*, el mapa *del Modelo Digital de Elevaciones MDE* de malla con polígonos sólidos, con interpolación de materiales, etc. Maneja un *Mapa de Colores* para la *Interpolación de Colores* y para la *Interpolación de Materiales con Iluminación.*

También se cumplió el objetivo de crear un sistema para servir como una herramienta de apoyo para es estudio y minitoreo de volcanes activos ya que este sistema ya ha sido instalado y probado exitosamente en los equipos del *CENAPRED.*

Este sistema esta siendo aprovechado para implementarse como parte de un Sistema de Información Geográfico conocido como *GisLabvis* que se desarrolla en el Departamento de Visualización de la Dirección General de Servicios de Computo Académico (*DGSCA*) de la *UNAM*. El Modelo Gráfico esta siendo aprovechado para crear un Simulador de Trayectorias de Balísticos Volcánicos

Trabajo a futuro:

- Sustituir la actual interfaz *GLUT* con otra en *JAVA* utilizando las clases de *4GL* y *JNI* que comunican a *JAVA* con *OpenGL*.
- Mejorar el modelo físico añadiendo nuevos parámetros ó sustituirlo con un Modelo de Dinámica de Fluidos.
- Adecuar al sistema para que trabaje en ambientes colaborativos remotos.
- Sobreponer imágenes de satélite como textura para los mapas.
- Incorporar al *STFVI 3D* como una herramienta del desarrollo de *GisLabvis*.

El *STFVI 3D* se ha presentado en los eventos de la **X Reunión Nacional *Selper*** (Sociedad de Especialistas Latinoamericanos en Percepción Remota y Sistemas de Información Espacial) en noviembre de 1999 en la ciudad de Guanajuato Gto. México y en la **VII Reunión Internacional "Volcán de Colima"** en marzo del 2000 en la ciudad de Colima Col. México.

Bibliografía

[1] Neider, Jackie. Davis,Tom. Wood, Mason / OpenGL Architecture Review Board.
OpenGL Programming Guide: The Official Guide to Learning OpenGL, Release 1.
Addison-Wesley
1993
ISBN 0-201-63274-8

[2] Neider, Jackie. Davis,Tom. Wood, Mason. Shreiner, Dave. OpenGL Architecture Review Board.
OpenGL Programming Guide: The Official Guide to Learning OpenGL, Release 1.2 Third Edition
Addison-Wesley
1997
ISBN 0-201-60458-2

[3] Kilgard, Mark J.
OpenGL Programming for the X Window System.
Addison-Wesley
1996
ISBN 0-201-48359-9

[4] Fosner, Ron.
OpenGL Programming for Windows 95 and Windows NT.
Addison-Wesley
1996
ISBN 0-201-40709-4

[5] OpenGL Architecture Review Board.
OpenGL Reference Manual: The Official Reference Document for OpenGL, Release 1.
Addison-Wesley
1992
ISBN 0-201-63276-4

[6] Kilgard, Mark J.
The OpenGL Utility Toolkit (GLUT) Programming Interface, API Version 3.
http://reality.sgi.com/mjk/glut3/glut3.html
1996.

[7] Baker, Steve. Davis,Tom.
A Brief MUI User Guide.
http://reality.sgi.com/mjk/tips/mui/mui.html

[8] Angel, Edward.
Interactive Computer Graphics: A Top-Down Approach whit OpenGL. 2nd ed.
Addison-Wesley
2000
ISBN 0-201-38597-X

[9] Wright, Richard S. Jr. and Sweet, Michael.
OpenGL Superbible: The Complete Guide to OpenGL Programming for Windows NT and Windows 95.
The Waite Group Press, Inc.
1996
ISBN 1-57169-073-5

[10] Foley, James D. van Dam, Andries. Ferner, Steven K. Hughes, John F.
Computer Graphics Principies and Practice, Second Edition.
Addison-Wesley
1992
ISBN 0-201-12110-7

[11] Salomon, David.
Computer Graphics & Geometric Modeling.
Springer-Verlag New York, Inc.
1999
ISBN 0-387-98682-0

[12] Watt, Alan H.
Fundamental of Tree-Dimensional Computer Graphics.
Addison-Wesley
1989
ISBN 0-201-15442-0

[13] Worboys, Michael F.
GIS: A Computing Perspective.
Taylor & Francis Publishers
1995
ISBN 0-7484-0064-8

[14] Star, Jeffrey. Estes, John.
Geographics Information Systems An introduccion.
Prentice Hall
1990
ISBN 0-13-351123-5

[15] Araña Saavedra, Vicente. Ortiz Ramiz, Ramón.
Volcanología.
Editorial Rueda Porto Cristo
1984
ISBN 84-00-05833-X CSIC
ISBN 84-7207-037-9 Editorial Rueda

[16] Mathews, Jhon H.
Numerical Methods for Computer Science Engineering and Mathematics.
Prentice Hall
1987
ISBN 0-13-62665 6-8

[17] Vega Munguía, Elio.
Sistema de visualización de datos topográficos del territorio nacional.
Tesis de Licenciatura de Matemáticas Aplicadas y Computación.
ENEP Acatlán, UNAM, México.
1999

[18] Avalos Torres, Hugo.
Peligros y riesgos asociados a lahares en el volcán Popocatépetl.
Tesis de Licenciatura de Ingeniería Geológica.
Escuela Superior de Ingeniería y Arquitectura, Unidad Ticomán IPN, México.
1998

[19] Alencastre Miranda, Moises.
OpenGL como herramienta para simulación en robótica.
Tesis de Licenciatura de Ingeniería en Cibernética y en Sistemas Computacionales.
Escuela de Ingeniería, Universidad La Salle, México.
2000