Capítulo III

UDLASIG

En este capítulo se describe el contexto bajo el cual se desarrolló la presente tesis, en el se describe la aplicación UDLASIG (Sistema de Información Geográfica de la Universidad de Las Américas Puebla) y el SIGOO (Sistema de Información Geográfica Orientada a Objetos) que también forma parte del UDLASIG.

3. UDLASIG

El Sistema de Información Geográfica de la Universidad de las Américas Puebla consta de dos partes, la interfaz desarrollada por Juan Luis Briones [Briones,98] y el modelado de la base de datos realizada por Nidia Posada [Posada,99].

3.1 Interfaz

Es una interfaz gráfica capaz de desplegar información espacial en un applet contenida en una base de datos relacional, y permite editar información e incluso agregar elementos nuevos que pudieran actualizar la base de datos[Briones,98].

Existen cuatro subsistemas en un SIG (ver figura 3.1), el subsistema de entrada de datos, el subsistema de almacenamiento y recuperación de datos, el subsistema de análisis y manipulación de datos y por último el subsistema de salida y despliegue de resultados. La interfaz gráfica del UDLASIG que pertenece al subsistema de salida y despliegue de datos de un SIG, tiene funcionalidades del subsistema de entrada y parte del subsistema de almacenamiento y recuperación de datos.

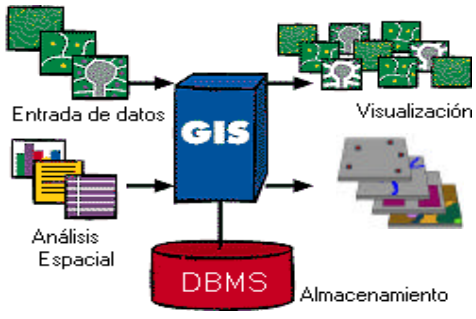

Figura 3.1 Subsistemas de un SIG

3.1.2 Interfaz gráfica de usuario

Esta interfaz fue implementada usando el JDK1.8 de Java más sus correspondientes librerias de Abstract Windowing Toolkit (AWT) y de Java Swing. La interfaz cuenta con botones, checkboxes, ventanas internas y en uno de los cuales se pueden desplegar los elementos básicos de un SIG (líneas, Puntos y Polígonos). También cuenta con algoritmos de transformación de coordenadas de dispositvo a coordenadas del mundo y viceversa. Los elementos básicos de un SIG se encuentran en el programa como objetos.

El formato propietario del UDLASIG esta compuesto por los elementos básicos de un SIG Punto, Línea y Polígono, los cuales son en realidad extensiones de la clase "Polygon" de java.awt, lo que en realidad se tiene son clases que heredan del "Polygon" todos sus atributos y métodos. La solución propuesta por Briones [Briones, 98] es práctica, ya que únicamente se dibujan Polígonos en la interfaz, el punto es un pequeño cuadrado de 2 x 2 pixels, la línea es un rectángulo de 2 pixels de ancho por la longitud de la línea, el polígono está conformado por un conjunto de líneas que forman un anillo.

3.1.3 Limitantes del formato

Solamente se guardan los puntos de cada tipo de elemento en la base de datos, siendo todas las coordenadas tanto de dispositivo y como de atributos de tipo entero, lo cual es una limitante muy grande ya que la mayoría de los sistemas de coordenadas emplean números de punto flotante de doble precisión. La otra limitante es que el sistema solamente puede leer y manipular los datos que se crean con la aplicación misma.

3.1.4 Funcionamiento del applet

Como se puede apreciar en la figura 32 la interfaz cuenta con dos filas de botones, en la primer fila se encuentran las opciones para manejar las vistas y las capas (botones del 1 al 12). Estas opciones incluyen la creación, edición y almacenamiento de vistas y capas.

La segunda fila de botones (del 13 al 21) se emplean para la manipulación de los elementos de las vistas.

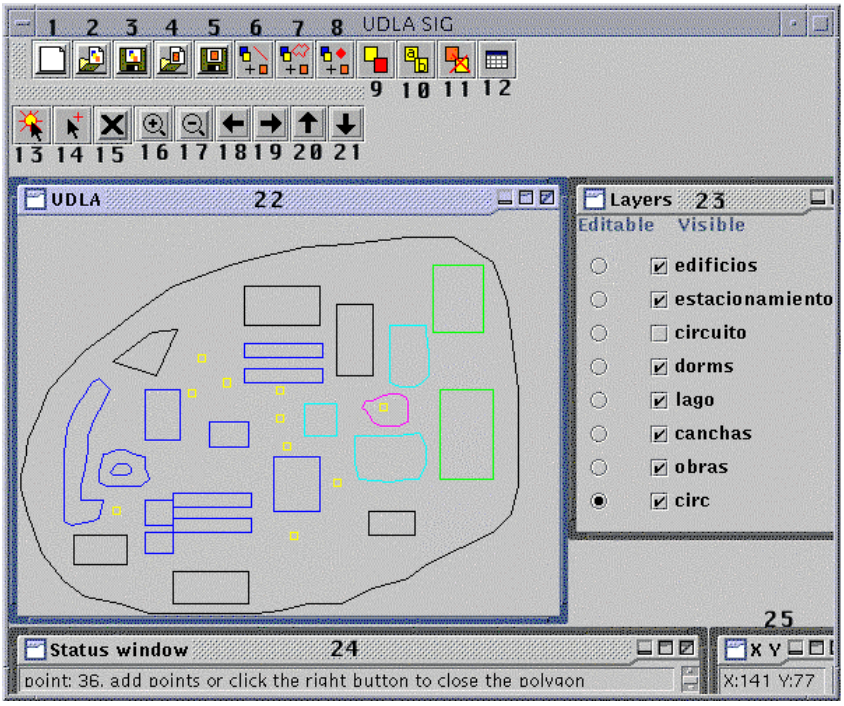

Figura No 3.2 Interfaz de UDLASIG

Las funciones de los botones podemos resumirlas en la siguiente lista:

1. Crear nuevas capas o vistas
2. Abrir una vista
3. Guardar los cambios realizados en la vista
4. Abrir una capa

5. Guardar los cambios realizados en una capa
6. Capa de líneas en la vista actual
7. Capa de polígonos en la vista actual
8. Capa de puntos en la vista actual
9. Cambiar de color los elementos de una capa
10. Cambiar de nombre de una capa
11. Borrar una capa
12. Consultar la base de datos

Aparte de las barras de botones también se tienen una ventana para la Vista, otra para las capas (layers, número 23), una ventana de estatus ("status window" número 24) y una ventana de coordenadas (número 25).

En la ventana de la Vista se visualizan las capas con sus respectivos elementos, en la ventana de Layers se muestran las capas que pertenecen a la vista con dos opciones, la primera opción es para poner la capa en estado de edición y la segunda es un checkbox para hacer la capa visible o invisible.

En la ventana de Estatus se muestran los mensajes del sistema como lo es la apertura de la conexión de la base de datos, instrucciones para capturar puntos y errores.

La aplicación funciona como editor, ya que solamente se puede visualizar la información que con ella se crea a través de los botones , la información que se crea es almacenada en la base de datos Illustra bajo un modelo relacional, la cual solamente es consultada cuando se necesita guardar o extraer información. Todas las operaciones de manipulación de la información como son los acercamientos y paneos se realizan en el cliente.

Podemos resumir las características más importantes en la siguiente tabla:

Características del UDLASIG
• Interfaz en java que emplea las JFC (Swing y AWT) • Procesamiento en el cliente • Realiza las operaciones básicas de un SIG (Acercamientos, alejamientos y paneos) • Formato propietario, no compatible con ningún otro visualizador de un SIG • Aplicación extendible • Manejo de coordenadas con números enteros • El modelado de la base de datos está bajo un modelo relacional

Tabla 3.1 Características del UDLASIG

3.2 SIGOO (Sistema de Información Geográfica Orientado a Objetos)

Este sistema pertenece al UDLASIG, el cual es tomado como base para incrementar su funcionalidad.

3.2.1 Nuevas características y clases principales

Se le agregó un nuevo tipo de elemento básico, la Polilínea, quedando los elementos gráficos de la siguiente forma: punto, línea, polígono y polilínea. También se enriqueció con un algoritmo que permite conocer las intersecciones que existen entre las capas de la vista. La aportación más importante fue el modelado que se realizo en la base de datos, pasando de un modelo relacional a un objeto-relacional.

Para la implementación del modelo objeto-relacional se cambio de DBMS de Illustra a Informix Universal Server (IUS), a través del cual se crearon los tipos de datos necesarios como lo son el punto, la línea, el polígono y la polilínea. También se crearon los tipos vista y capa, quedando los tipos de datos con los mismos atributos que las clases que se programaron en java. Cabe resaltar que no se realizó ninguna programación en el IUS, solamente la definición de tipos de datos. La estructura de las clases quedo de la siguiente forma.

Clase Vista **(Métodos)**
• Seleccionar • Salvar • Recuperar • Crear • Intersección

Clase Vista
(Métodos)
• Borrar
• Cambiar color
• Cambiar nombre

Tabla 3.2 Métodos de la clase vista

Clase Capa	
Atributos	**Métodos**
• Color	• Creación de la clase Capa
• Nombre	• Pintado de Línea
• Tipo	• Pintado de Punto
	• Pintado de Polilínea
	• Pintado de Polígono

Tabla 3.3 Atributos y métodos de la clase Capa

3.2.2 Subclases de Capa

La clase principal es vista y una vista puede tener una o más capas y cada capa puede estar conformada por solamente un tipo de objetos geográficos, es decir:

- Punto
- Línea
- Polilínea
- Polígono

3.2.3 Atributos de los elementos básicos

Llamamos elementos básicos a las clases que constituyen las partes de una vista. Las tablas siguientes describen los objetos geográficos de base.

Atributos de la clase Punto
X // coordenada en el eje x Y // coordenada en el eje y

Tabla 3.4 Atributos de la clase Punto

Atributos de la clase Línea
• X1 // coordenada x del punto 1 • Y1 // coordenada y del punto 1 • X2 // coordenada x del punto 2 • Y2 // coordenada y del punto 1

Tabla 3.5 Atributos de la clase Línea

Clase Polilínea	
Atributos	**Métodos**
• XP[] // coordenada X del iesimo punto (arreglo de puntos) • YP[] // coordenada Y del iesimo punto (arreglo de puntos) • NP //número de puntos con los cuales cuenta cada arreglo	• Pintado parcial • Dibujado de polilínea

Tabla 3.6 Clase Polilínea

Atributos de la clase Polígono
• XP[] // coordenada X del iesimo punto (arreglo de puntos) • YP[] // coordenada Y del iesimo punto (arreglo de puntos) • NP //número de puntos con los cuales cuenta cada arreglo

Tabla 3.7 Atributos de la clase Polígono

Nota: todas las clases con excepción de la Polilínea heredan los atributos de la clase Polygon de java.awt. Esto implica que todos los valores de las coordenadas estén en números enteros y que hereden sus métodos.

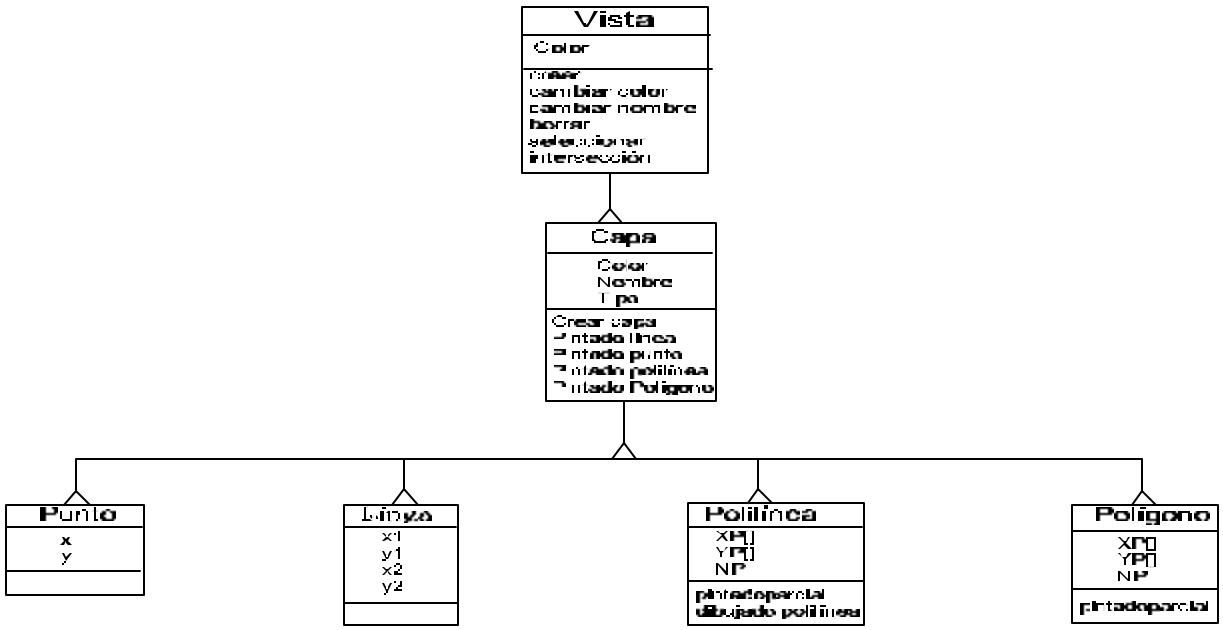

Figura 3.3 Diagrama de la jerarquía de clases

En la figura 3.3 se aprecia plenamente la estructura de los tipos de datos, la que es igual al de las clases de java. De aquí desprendemos la idea, de que una vista esta formada por capas y las capas se forman ya sea de puntos, líneas, polilíneas o polígonos.

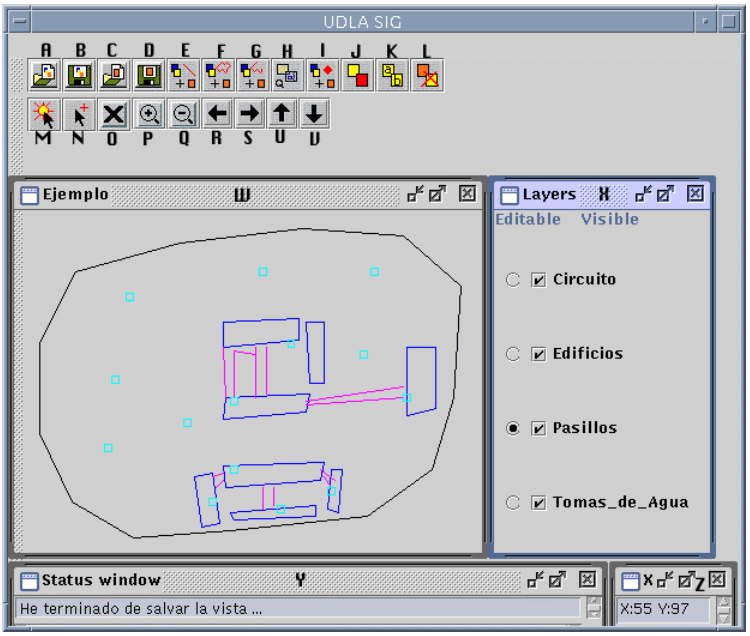

Fig 3.4 Interfaz del SIGOO

Como se muestra en la figura 3.4 la Interfaz del SIGOO es la misma que la del GISUDLA, con excepción de que en la barra de botones aparecen dos nuevos, el botón G de la pimera barra que se emplea para agregar las polilíneas y el botón H sirve para hacer consultas de intersección de capas. Solamente se eliminó de la interfaz anterior el botón que realizaba consultas a la base de datos.

En el SIGOO la información que se visualiza y maneja unicamente es la que se crea con el propio editor de la interfaz, es decir, no maneja información que haya sido procesada de alguna foto ærea o de algún plano, es decir falta el empleo de un formato estándar que nos permita visualizar la información de otras aplicaciones.

Características del SIGOO
• Interfaz en java que emplea las JFC (Swing y AWT) • Procesamiento en el cliente • Realiza las operaciones básicas de un SIG (Acercamientos, alejamientos y paneos)

Características del SIGOO
• Formato propietario • Aplicación extendible • Manejo de coordenadas con número enteros • El modelado de la base de datos está bajo un modelo objeto-relacional • Manejo de puntos, líneas, polígono y polilíneas • Queries de intersección entre capas

Tabla 3.8 Características del SIGOO

3.3 Línea de ensamble entre ArcView, GISUDLA y OpenGIS

La falta de interoperabilidad de aplicaciones de GISUDLA hace necesario la creación de un módulo que nos permita exportar datos de otros SIG, visualizarlos y emplear un formato estándar como OpenGis para su almacenamiento.

Tomamos en cuenta OpenGis como estándar debido a la gran disponibilidad de información, también pensamos en ArcView como SIG del cual se van a tomar los datos para la exportación debido a su alto volumen de usuarios y a que se dispone del mismo en la Universidad. Entre las ventajas con las que contaría nuestra aplicación con respecto a ArcView Inernet Map Server (AIMS) se reflejarían en el formato de almcenamiento ya que se contaría con un formato estándar en lugar de uno propietario, el procesamiento de imagenes se llevará acabo en el cliente y solamente se consultará al servidor cuando haya que agregar otra capa o salvar, contra AIMS el cual lleva acabo todo el procesamiento en el servidor.

3.4 Conclusión

GISUDLA se tomó como base para agregarle y cambiarle características: La primer característica que requería cambiar es el despliegue de coordenadas de números enteros a números de punto flotante de doble precisión, lo cual nos llevo a realizar una investigación sobre el prototipo debido a que la documentación sobre el código fuente fue muy escasa, ya entendiendo el funcionamiento nos dimos cuenta que se empleaba una clase de java.awt.Poligon la cual no existe como tal en java2D y por lo mismo fue necesario hacer de nueva cuenta todos los métodos de lectura, almacenamiento y graficación con los tipos de datos de java2D. Ya teniendo el applet graficando números de punto flotante de doble precisión comenzamos a meternos más a fondo a la especificación del formato propietario de ArcView, el cual se describe en el siguiente capítulo.