INTERSECTION AND UNION OVERLAYS FOR GIS

Major Project

Submitted in partial fulfillment of the requirements

For the degree of

Master of Technology in Computer Engineering

By

Keyur Patel
(04MCE009)

DEPARTMENT OF COMPUTER SCIENCE AND TECHNOLOGY
Ahmedabad 382481
May 2006

This is to certify that the Dissertation entitled

Intersection and Union Overlays for GIS

Presented by

Keyur Patel

has been accepted toward fulfillment of the requirement
for the degree of
Master of technology in Computer Science & Engineering

Professor In Charge **Head of The Department**

Date

ABSTRACT

Intersection and Union are the most important Overlay functions in the GIS. It mainly concerns with the algorithms which should be optimized and made efficient in terms of memory requirement and processor utilization. Memory can be saved by using the simple data structure and processor utilization can be improved by using algorithms which minimizes the use of more iterative loops.

This thesis explains efficient algorithms related to the Intersection and Union overlay and is based on the vector images mainly stored as shape file format. Intersection and Union algorithms mainly concerns with the Point in Polygon strategies, Line-Line intersection, Line-Polygon intersection, Polygon-Polygon intersection, Polygon-Polygon Union and their integration with each other. By using this algorithms Intersection and Union Overlays can be developed using different programming languages. Here it is developed using VC++ and VB.

Keywords: Point in Polygon strategies, Line-Line intersection, Line-Polygon intersection, Polygon-Polygon intersection, Polygon-Polygon Union, Intersection, Union

ACKNOWLEDGEMENT

I take this opportunity to express my immense gratitude to my guide **Mr. Shashikant Sharma**, Senior Scientist, BISAG Gandhinagar, for accepting me to work under his guidance for my Training. I am grateful to him for his prolonged interest in my work and excellent guidance. He has been a constant source of motivation to me. His uncompromising demand for quality and instance for meeting the deadlines motivated me a lot to achieve excellence in whatever I did. He induced me to learn and improve all the time.

I would also like to thank **Mr. T.P.Singh**, Director, BISAG , who always motivate students to do something extraordinary. He provided very friendly environment in the institute so that I have been able to work at extreme level of mine. I also thank him for sparing his valuable time for us. I would like to thank him for providing me unlimited resources, which greatly helped me in making me learn and do the things faster.

I am also thankful to **Dr. A.R. Dasgupta**, Senior Scientist, BISAG, for sparing his valuable time for giving us training about the GIS system and its Applications to make our view batter to understand the GIS systems. I also offer my thanks to all the employees of BISAG, for helping me, for providing all the resources I required.

I am highly indebted to **Dr. S.N. Pradhan** (Course Co-ordinator, MTech -CSE) and **Dr. D.J. Patel** (H.O.D-Computer Department), Nirma Institute of Technology, Ahmedabad, for allowing me to join BISAG, for my research and theses work and also for their encouragement and guidance given to me throughout my training period.

I am also thankful to my family members and friends who motivated me throughout this course, I am thankful to all of them.

CONTENTS

LIST OF FIGURES

LIST OF TABLES

Chapter 1

INTRODUCTION

- Motivation
- Client and targeted users of project
- Literature Survey
- Existing Systems

1.1 Motivation

About BISAG, The BISAG is scientific society registered under the societies act, Govt. of Gujarat state. All office bearers are executive senior government officials. It has a singular purpose to identify and satisfy the needs of Gujarat state by using remote sensing, GIS technology and satellite communication. It has mandate to apply state of the art technology to create the GIS based decision support system for various infrastructure and resource utilities in Gujarat and promote, train and help users to establish their own GIS facility using Satellite communication. To provide linkage between state capital, Districts and taluka level for distant interactive training, education and extension. To do all such activities BISAG is armed with Scientists and Engineers.

Need of BISAG in Gujarat

Space technology offers synoptic and repetitive information, which could be of immense value in providing scientific database and reliable maps. It also is a means to provide distant interactive learning by a way of satellite communication.

Gujarat is an industrially advanced and progressive state. The state has characteristic problem of natural diversity of highly uneven rainfall and drought prone areas whereas few areas are prone to floods. Uncertain, erratic and uneven rainfall causes droughts and stresses on water for drinking and irrigation purposes. This has led to over exploitation of ground water resources and salinity ingress in coastal areas. Soil fertility varies from the central fertile plains to low fertility and hilly areas.

Saurashtra and Kutch are starved of soil moisture and are denuded whereas the coastal areas are waterlogged and saline. Due to terrain conditions large agricultural areas are facing soil degradation affection productivity. Forest areas facing degradation require immediate attention. Similarly wasteland reclamation program also requires to be taken up on large scale. The state has about 1600 KMs long coast.

Motivation for Work

So it is a great opportunity to have research work to be carried out at the BISG. And we also get here challenging work to do at a time immense moral support and guidance from the organization to achieve Excellency in our work. For that all the resources are given by the organization.

As I mentioned BISAG is mainly doing work related to the maps taken from the satellite. So it is used for the survey of the soil, forest area, rainy area, polutated area etc.

So many time it is required to get combined information from two different different maps. Like we want to get the view of the cities which are populated more than 1,00,000 and also having local train network. So here we need to intersect two maps one is related to the cities having population attribute and other describing the cities having local train network attribute.

Likewise if we want the area which is either forest area or area having the pollution more than some specific value than we have to do the Union of the two maps one having the attribute forest and other with pollution values.

So my work is to do this thing in most efficient way to utilize memory and processor efficiently and for that defining new algorithms and also implementing it using Visual Basic and Visual C++. For that I need to learn the Shape file as all the maps are stored in shape file format.

So I find this thing very challenging and inspiring as it requires immense study on memory optimization, algorithms and also on the processor utilization.

1.2 Client and targeted users of project

All the organization requiring the information related to the particular area on the earth based on its map.

Government sectors requiring information regarding villages and doing regular surveys for this villages and cities can be benefited a lot by using this.

Any business firm or individual wanting to develop plant at any place can use this system to see that all the requirements can be easily fulfilled or not at that place and what is the economic value of that particular place.

1.3 Literature Survey

1.3.1 Geographic Information System

Introduction

Geography is the science of spatial relationships. Maps from a major constitute of geography, as they are a means of representing very large spatial relationships in a physically handle-able size. The need for environmentally being and socially accountable development has put a heavy demand on the capabilities of planners. Therefore planning and execution now require more accurate, reliable and timely information and better tools for the management of such information. This require not only a variety of maps but a large amount of spatial information, commonly known as statistics, a means to handle this tools to selectively extract information relevant to the planning task. In sort, an information system for geographical data is needed.

Any information system has four major components. There is an input module which accepts data, a data base module which organized and stores the data, an analysis module which selectively retrieves and manipulates the data and an output module which presents the analyzed information.

A Geographic Information System (GIS) is different in the sense that it handles both spatial and non-spatial data; consequently the corresponding module becomes more complex.

Information system incorporates apart from database management functions, a set of analysis modules which selectively retrieve and manipulate the data and output module which represents analyzed information in the context of given planning needs.

Input to an Information System is data, which is raw and does not convey any meaning unless analyzed and converted into meaningful information. Output from any Information System is the information, which conveys a meaning and prompts a set of action or decisions.

GIS (Geographical Information System) plays major role by providing linkage between the information domain and the technologies available for management and development planning. Geographical Information System is a particular from of information system is a set of process, executed on a raw data, to product information, which will be useful indecision making. Therefore an information system must have a full range of functions to achieve its purpose, including observation, measurement, description, explanation, forecasting and decision-making.

Functionality of GIS

GIS, basically refers to science and technology dealing with the character and the structure of spatial information, its method of capture, organization, classification, analysis, measurement, display and dissemination as will as the infrastructure necessary for the optimal use of the information. With the increase in volume and dimensionality of data it becomes essential to use automated GIS. Use of an automated system has become necessary as the data are maintain in a physically compact format (i.e. magnetic media), data can be retrieved with the greater speed, various computerized tools allow a variety of manipulation. Hence, GIS is defining as:

"AN AUOTOMATED TOOL USEFUL TO CAPTURE, STORAGE, RETRIVALAND AND MANIPULATION, DISPLAY AND QUERYING BOTH SPATIAL AND NON-SPATIAL DATA TO GENERATE VARIOUS PLANNING SCENARIOUS FOR DECISION MAKING."

Alternatively, we could describe "Geographic Information System (GIS) as a system which provide a computerized mechanism for integrating various geo-reference datasets analyzing them in order to geo –reference datasets analyzing them in order to generate information relevant to planning needs in a given context."

Thus GIS is a tool able to answer the location, condition, trends and modeling. For example, the location of feature is answered with the help of geographic reference

Latitude and longitude. The second question the converse of the first one and requires spatial analysis to answer. Instead of identifying what exists at a given location, one wants to find location where certain conditions are satisfied. The third one involves both of the above to answer and seeks to find the difference with an area through time. The fourth one determines what types of spatial pattern exists when the last one would be *"what if…"*question is paused to determine what happens. To answer this integrated to require combining spatial and non- spatial information.

A Geographical Information System (GIS) is a particular from of information system applied to geographical data. It is powerful tool to input, store, manipulate analysis, model, and output spatial and attribute information. We could describe Geographical Information System various geo referenced data sets and analyzing in order to generate information relevant to planning needs in a given context. The ultimate goal of a GIS is to support decision-making.

Like any other information System, GIS is also primarily an input output system and include Database Management System (DBMS) function. However, it differs from conventional DBMS and information systems in the sense that every pieces of data element in GIS has to be directly as co –ordinates with respect to predefined Co-ordinate system.

The data handling in GIS is not limited to only data with explicit coordinate reference. Coordinate could express indirectly, e.g. demography associated with the

village and village being expressed in form of a polygon or point coordinate. In general GIS facilities handling of variety of data both spatial as well as non-spatial.

By organizing spatial information into from, GIS allows us to manipulate and display geographical knowledge in new and exciting ways.

Requirements of GIS

The environment in which a GIS operates is defined by hardware (the machinery including a hot computer), a digitizer or a scanner for converting the input data, a plotter for presentation processed outputs and video display unit for commanding the system by a user, the software (programs that tell the computer what to do) and the data. In this context GIS can be seen as a system of hardware, software, and procedures design to support the capture, management data for solving, analyzing, modeling and display spatially-referenced data for solving complex planning and management problems. Although many other computer programs can use spatial data (e.g. Auto CAD and statistics packages), GIS include the additional ability to perform spatial operations.

Major Component of GIS

The major components of GIS are:

1. The end use or management
2. Data Acquisition
3. Data Input
4. Data Storage and retrieval
5. Analysis
6. Information presentation

There are two components of Geographic data i.e. spatial data and attribute data. On the maps, symbols and text convey descriptive information. The map then becomes a powerful tool for referencing geographic information. The same concept applied to spatial data model. Therefore the powerful capability of GIS lies in the

link between the spatial and the tabular data, which illustrate the relationship between and attribute data.

Types of GIS

GIS may be categorizing into three basic type:

- Map Based
- Integrated
- Linkage Based

Information Storage

Because graphic information is very different from tabular information, GIS uses to database to graphic and tabular information that must manage: a spatial and an attribute database.

Spatial Database

A spatial database is used to manage graphic information for several reasons.

A spatial database makes searches for graphic extremely fast because the spatial database has been optimize to manage graphic information rather than tabular information.

A spatial database can store and manipulate complex entities as discrete objects. For example, a polygon can be selected by referring a point anywhere on or within the polygon.

A spatial database allow the use of double precision numbers consequently, a spatial database can store very large coordinates values and is able to manage very large amount of graphic information without significant decline in performance.

A spatial database allows entities to be retrieved and manipulated independently of the graphic engine and attribute database.

The spatial database stores two forms of graphic information

Object Geometric

Object relationships (Topology)

Objects

All graphical entities are store in the spatial data as spatial objects. Any entity for which you wish to maintain, store, and retrieve the geographical location may be considered an object. In GIS, first creating a graphic representation of the object in Cad and then loading this representation of the object into the spatial database create objects.

Object Types

The GIS spatial data is able to handle four types of objects. This objects can be categorized as simple objects and complex objects. Simple objects include point and line objects are defined known number of coordinate pairs. Complex objects are defined by undetermined of coordinates pair and include line, string and polygon objects.

- Point Objects
- Line Objects
- String Objects
- Polygon Objects

Connecting Features and Attributes

The importance of GIS lies in its link between the graphic (spatial) and the tabular (non-spatial or a spatial) data. Basically there are three characteristics of this connection. They are

- ✓ There is one to one relationship between features on digital map and the records in the feature attribute table. The link between the feature and

its records is maintained through a unique numerical identifier assigned to each feature (label points).

- ✓ The unique identifier is physically in two places i.e. in the file that content X, Y coordinates and with the corresponding records in the feature attribute table.
- ✓ So, once this connection is established one can query the digital map to display attribute information or create a map based on the attributes stored in the feature attribute tables.

Data Manipulation

GIS allows a variety of manipulation such as map measurement, map overlay analysis transformation from one coordinate projection to another coordinate projection, graphic design and manipulation. In addition to this graphics and non-graphics data can be merged and manipulated simultaneously in related manner.

Apart from this system must also facilities other typical related DBMS function like interactive query language, basic file creation / update file management, basic search retrieval and report generation, database content and high level language interface.

Spatial Database Management in GIS

The scope database in GIS is quite varied as compared conventional DBMS .In the conventional DBMS on the deals with one-dimensional data (non-spatial) in form of hierarchical, network or relation model. In GIS the data is handled in the form of Geo-Relational model, which ha two components viz., Entity location data in form topological structures handled in conventional file mode and attributes in form of a RDBMS, which is embedded within GIS.

The first component concerned with the graphic data domain, where in the location specific data for various geographic entities are stored, is system specific and a user is not given access for making any modifications in the file structure in the file structures. In the second domain pertaining to the non-location attributes of the spatial entities, users can attempt variety of modification by adding, removing or changing the specifications of specific set of attribute for the spatial entities.

Graphics and non-graphics domains are link through the relation tables. Such an organization gives sample scope for building up an integrated date base involving spatial and non-spatial data elements. This opens up many exciting possibilities with regards to the end use of the database. Looking at the ends of such an integrated database in a perspective, the Geo-relational nature of the data organizations makes it possible to:

- Visualize the GIS database as integrated database providing a common platform for spatial as well as non-spatial data from variety of sources.
- Make integrated queries on the underlying database looking at any combination of data elements together.
- General planning views based on multiple analyses.
- General planning views based on analyses of multiple parameters.
- And simulate "what – if…… Scenarios in an interactive and iterative manner providing flexible decision supports in spatial context approaching towards paperless planning.

Relation between Spatial and non-Spatial Data

With spatial data if proper LINKAGE BETWEEN SPATIAL AND NON-SPATIAL data Non-spatial data are an important part of GIS database. User can use these data along is defined. There linkages and inter relationship are an import element of the GIS database organizations as they define the user relations or user views that can be created.

Spatial and non-Spatial Data Linkage

All the spatial data sets have as associated attribute table where the detail attributes table of each feature is recorded. There are two major linkage aspects involved:

- For all the spatial data sets other than administrative maps, the linkage is achieved through the data dictionary feature code.
- For administrative maps, villages and Taluka maps, the linkage is archived on a one-to one relation based to the on a unique code for each

village or the Taluka. This link code also is related to the census village number on a one-to –one basis. Thus the internal organization of the spatial village, Taluka

- boundaries if flexible to relate to the non-spatial data set son a one-to-one basis further, because of the co-relation to census village numbers it is also possible to abstract village data to Taluka data.

There are varying approaches foe design of data models and structure as described below.

Two basic types of data models have evolved for storing the spatial data in digital form.

1. **Vector**
2. **Tessellation (grid) models.**

In the vector type of data model, the basic logical unit in a map is line or vector. A series of x-y point's location along the line are recorded as the components of a single data record. The points can be represented as lines of zero length (i.e. one x-y location).

In the grid model the basic logical unit for storage is a single cell or unit of space in the mesh. This class of data models encompasses much more than the approaches based on a rectangular of square grid. These may include May identify repeatable pattern of a regular polygon (two dimensional) or polyhedron (three dimensional) vector data mode also has its variation in the way they describe the entity interrelationships.

Whereas the two classes of spatial data models are in common practice today, there also exists a third type of model –the hybrid type. This class of the data model is a recent development with an attempt to possess the characteristics of both the vector and models.

Since a GIS attempts to model the real world as a set of digital number and features, these models should be amenable to structuring, querying and processing the problems. The way a spatial data processed creates the problems.

The way a spatial data processing algorithm works, is not necessary the way a human operator will solve a problem in the real world. The data structure must therefore account for the quirks of the algorithms. This requirement leads to the adoption of a functional structure based on primitives.

The data structure decides the physical and logical storage, retrieval, query and analysis capabilities. It allows for the linking of spatial entitles and spatial attributes. It also impacts the efficiency of storage (space) and access (time). A spatial data is organizing their inter–relationships. In a GIS this is essential. This relationship is the topology.

The oldest and most popular model has been the vector data model that deals which an ordered set of x and y locations representing the points, line and polygons. The ordering takes into account the spatial inter-relationship.

With the advent of remote sensing as well as automatic scanners another structure, which has gained popularly, is the raster structure (one amongst the family of grid models). Other structure have evolved form the need to handle both ector and raster data. One such example is the quid-tree data structure.

Vector data model and associated structures

The vector data model is based on representation of objects by an array of coordinates. As mentioned above, the vector mode of data representation has its variations.

The spaghetti structure

The most simplest and elementary vector data model is a direct line-for-line translation of the paper map. This is called sapphire mode. Associated data structure for this model is very simple and easy to understand. Each geographic entity becomes one logical record in the digital file, and is defined as strings of X-Y coordinates.

This model has severe limitations since there are no spatial relationships explicitly defined amongst various entitles. Moreover, for adjacent polygon data, the model results in recording of X-Y coordinate of the shared boundary segment twice.

The model is very inefficient for most types of spatial data analysis, since any relationship required has to be derived through computation. However the lack of explicitly stored relationships, which are extraneous to the plotting process, makes this model efficient for reproducing the original graphic input. The model is thus used primarily for applications that are limited to map inputs and outputs.

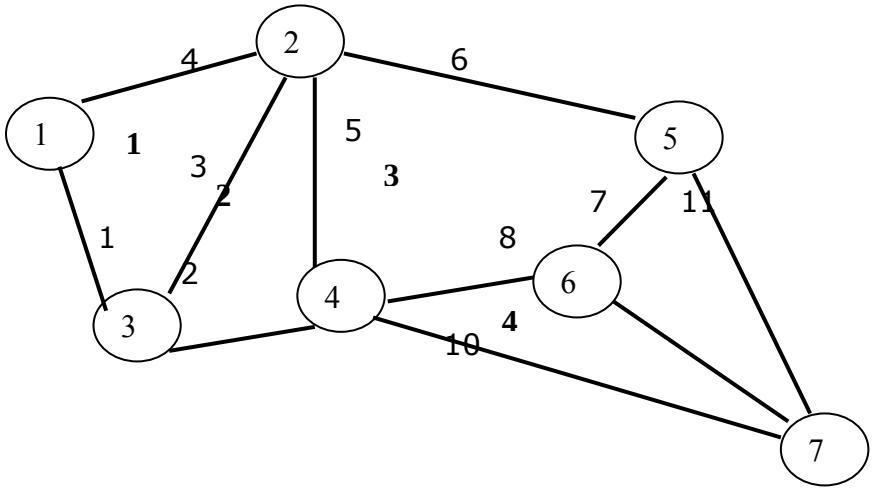


Figure 1.1 The Topological model

Table 1.1 Topologically coded Network and Polygon File

Link	Right Polygon	Left Polygon	Node 1	Node 2
1	1	0	3	1
2	2	0	4	3
3	2	1	3	2
4	1	0	1	2
5	3	2	4	2
6	3	0	2	5
7	3	3	5	6
8	4	3	6	4
9	5	4	7	6
10	4	0	7	4
11	5	0	5	7

Node	X Coordinate	Y Coordinate
1	23	8
2	17	17
3	29	15
4	26	21
5	8	26
6	22	30
7	24	38

The second vector model is the topological model where in the spatial relationships amongst the entities are explicitly recorded. The basic logical entities in this structure are points or nodes, arcs which are lines joining nodes and polygons which are areas completely enclosed by arcs. The attributes or descriptors are lined to each of these entitles.

Such an organization has an advantage over Spaghetti Model for spatial analysis as it minimizes geometric computation overheads for inferring the relationships amongst the spatial object. Moreover, it records the coordinates of the common polygon boundaries only once thereby reducing the redundancy and facilities consistency and integrity of data storage.

GIS-Core of the Database

The Geographic Information System (GIS) package is the core of the data sets, as both spatial and non-spatial databases have to be handled. The GIS package offers efficient utilities for handling both of these data sets and also allows for the spatial database organization; non-spatial data sets organization-mainly as attributes of the spatial elements; analysis and transformation for obtaining the required information; obtaining information in specific format (cartographic quality outputs and reports), organization of a user friendly query-system.

In the present study ARC/INFO GIS package has been used as the core of the spatial database. ARC/INFO is a modular, vector based package and is for making cartographic quality out puts in the form of maps and generation of model while the non-spatial data is organized using topological data model while non-spatial attribute data is stored using a database management package.

Three main aspects of any modern technology are: the instrumentation, the software and the manpower. Even though GIS does not demand any special Instrumentation, with the rapidly expanding use of computer technology in India has been a great boon for GIS. Most public sector organizations today are increasingly using GIS technology for the effective utilization of their resources, and its use in private sector is also catching up.

In GIS-related software development, even though the contribution of Indian GIS professionals is significant, a lot more needs to be done in developing

indigenous software systems for global market. In manpower development also, many academic and professional organizations in India are today offering comprehensive-training courses related to GIS.

However, introduction of GIS as a major field of study in Indian engineering and technological institutes and Universities is still a dream, and more efforts are required to be directed in this direction. The GIS-based publications, with their high quality technical contents, are also contributing significantly in spreading GIS awareness.

GIS is very useful to the following persons:

- ✓ Urban planner – might like to find out about the urban fringe growth in her/his city, and quantify the population growth that some suburbs are witnessing.
- ✓ Mining engineer could be interested in determining which prospect copper mines are best fit for future exploration, taking into account parameters such as extent, depth and quality of the one body, amongst others.
- ✓ Natural hazard analyst might like to identify the high-risk areas of annual monsoon-related flooding by looking at rainfall patterns and terrain characteristics.
- ✓ Land surveyor working on land development projects for urban expansion.
- ✓ Cadastral engineer working on the provision of up-to-date information.
- ✓ Regarding real estate and land property.
- ✓ Food security officer - working on monitoring systems of crop production.
- ✓ Land use suitability analyst working on assessments regarding production of certain crops in semi-arid areas.
- ✓ Wildlife manager - working to ensure that wildlife (e.g., elephant) has freedom of movement without damaging human settlements.

An important distinction between GIS application is whether the geographic phenomena studied are man-made or natural. There are many different uses of GIS, as may have become clear from our list of professionals as shown above who deal with geo information. An important distinction between GIS for urban planning purposes involves a study of man-made things: the roads, sidewalks, and at larger scale, suburbs and transportation routes are man-made. These entities are assumed to have clear-cut boundaries.

On the other hand, geomorphologies, ecologists and soil scientists often have natural phenomena as their study objects. They may be looking at rock formations, plate tectonics, and distribution of natural vegetation or soil units.

About Maps

The best-known models of the real world are maps. Maps have been used for thousands of years to represent information about the real world. Their conception and design has developed into a science with a high degree of sophistication. Maps have proven to be extremely useful for many applications in various domains.

A disadvantage of maps is that they are restricted to two-dimensional static representations, and that they always are displayed in a given scale. The map scale determines the spatial resolution of the graphic feature representation. The smaller the scale, the less detail a map can show. The accuracy of the base data, on the other hand puts limits to the scale in which a map can be sensibly drawn. The selection of a proper map scale is one of the first and most important steps in map design.

Cartography as the science and art of map making functions as an interpreter of translation real world phenomena into correct, clear and understandable representations for our use.

1.3.2 Shape file

This document defines the shape file (.shp) spatial data format and describes why shape files are important. It lists the tools available in Environmental Systems Research Institute, Inc. (ESRI), software for creating shape files directly or converting data into shape files from the formats. This document also provides all the technical information necessary for writing a computer program to create shape files without the use of ESRI ® software for organizations that want to write their own data translators.

Why Shape files?

A shape file stores no topological geometry and attributes information for the spatial features in a data set. The geometry for a feature is stored as a comprising a set of vector coordinates.

Because shape files do not have the processing overhead of a topological data structure, they have advantages over other data sources such as faster drawing speed and edit ability. Shape files handle single features that overlap or that is noncontiguous. They also typically require less disk space and are easier to read and write.

Shape files can support point, line, and area features. Area features are represented as closed loop, double-digitized polygons. Attributes are held in a dBase ® format file. Each attribute record has a one-to-one relationship with the associated shape record.

How Shape files can be created

Shape files can be created with the following four general methods:

- ✓ Export -- Shape files can be created by exporting any data source to a shape file using ARC/INFO ®, PC ARC/INFO ®, Spatial Database Engine ™ (SDE ™), Arc View ® GIS, or Business MAP ™ software.

- ✓ Digitize -- Shape files can be created directly by digitizing shapes using ARCVIEWGIS feature creation tools.
- ✓ Programming -- Using Avenue ™ (Arc View GIS), MapObjects ™, ARC Macro Language (AML ™) (ARC/INFO), or Simple Macro Language (SML ™)(PC ARC/INFO) software, you can create shape files within your programs.
- ✓ Write directly to the shape file specifications by creating a program.

SDE, ARC/INFO, PC ARC/INFO, Data Automation Kit (DAK ™), and Arc CAD ® software provide shape-to-coverage data translators, and ARC/INFO also provides a coverage-to-shape translator. For exchange with other data formats, the shape file specifications are published in this paper. Other data streams, such as those from global positioning system (GPS) receivers, can also be stored as shape files or X, Y event tables.

Shape File Technical Description

Computer programs can be created to read or write shape files using the technical specification in this section.

An ESRI shape file consists of a main file, an index file, and a dBASE table. The main file is a direct access, variable-record-length file in which each record describes a shape with a list of its vertices. In the index file, each record contains the offset of the corresponding main file record from the beginning of the main file. The dBASE table contains feature attributes with one record per feature. The one-to-one relationship between geometry and attributes is based on record number. Attribute records in the dBASE file must be in the same order as records in the main file.

Naming Conventions

All file names adhere to the 8.3 naming convention. The main file, the index file, and the dBASE file have the same prefix. The prefix must start with an alphanumeric character (a–Z, 0–9), followed by zero or up to seven characters (a–Z, 0–9, _, -). The suffix for the main file is .shp. The suffix for the index file is .shx.

The suffix for the dBASE table is .dbf. All letters in a file name are in lower case on operating systems with case-sensitive file names

Examples

- ✓ Main file: countries.shp
- ✓ Index file: countries.shx
- ✓ DBase table: countries.dbf

Numeric Types

A shape file stores integer and double-precision numbers. The remainder of this document will refer to the following types:

- Integer: Signed 32-bit integer (4 bytes)
- Double: Signed 64-bit IEEE double-precision floating point number (8 bytes)

Floating point numbers must be numeric values. Positive infinity, negative infinity, and Not-a-Number (NaN) values are not allowed in shape files. Nevertheless, shape files support the concept of "no data" values, but they are currently used only for measures. Any floating point number smaller than –10 38 is considered by a shape file reader to represent a "no data" value.

The first section below describes the general structure and organization of the shape file. The second section describes the record contents for each type of shape supported in the shape file.

Organization of the Main File

The main file (.shp) contains a fixed-length file header followed by variable-length records. Each variable-length record is made up of a fixed-length record header followed by variable-length record contents. Figure 1 illustrates the main file organization.

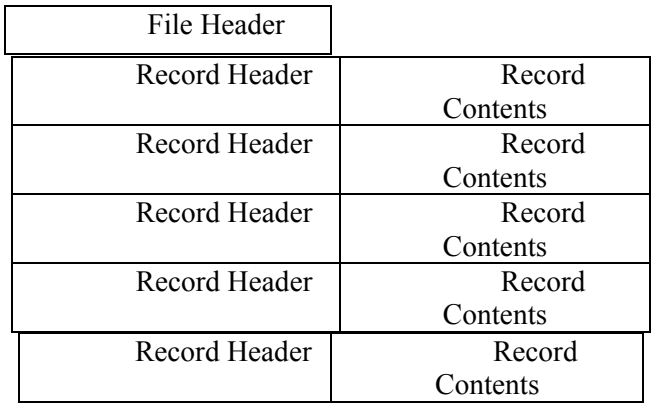


Figure 1 .2 Organization of the Main File

Byte Order

All the contents in a shape file can be divided into two categories:

Data related

· Main file record contents

· Main file header's data description fields (Shape Type, Bounding Box, etc.)

File management related

· File and record lengths

· Record offsets, and so on

The integers and double-precision integers that make up the data description fields in the file header (identified below) and record contents in the main file are in little endian (PC or Intel ®) byte order. The integers and double-precision floating point numbers that make up the rest of the file and file management are in big endian (Sun ® or Motorola ®) byte order. The Main File Header The main file header is 100 bytes long.

Table 1 shows the fields in the file header with their byte position, value, type, and byte order. In the table, position is with respect to the start of the file.

Table 1.2 Description of the Main File Header

Position	Field	Value	Type	Byte Order
Byte 0	File Code	9994	Integer	Big
Byte 4	Unused	0	Integer	Big
Byte 8	Unused	0	Integer	Big
Byte 12	Unused	0	Integer	Big
Byte 16	Unused	0	Integer	Big
Byte 20	Unused	0	Integer	Big
Byte 24	File Length	File Length	Integer	Big
Byte 28	Version	1000	Integer	Little
Byte 32	Shape Type	Shape Type	Integer	Little
Byte 36	Bounding Box	Xmin	Double	Little
Byte 44	Bounding Box	Ymin	Double	Little
Byte 52	Bounding Box	Xmax	Double	Little
Byte 60	Bounding Box	Ymax	Double	Little
Byte 68*	Bounding Box	Zmin	Double	Little
Byte 76*	Bounding Box	Zmax	Double	Little
Byte 84*	Bounding Box	Mmin	Double	Little
Byte 92*	Bounding Box	Mmax	Double	Little

* Unused, with value 0.0, if not Measured or Z type

The value for file length is the total length of the file in 16-bit words (including the fifty 16-bit words that make up the header).

All the non-Null shapes in a shape file are required to be of the same shape type. The values for shape type are as follows:

Value Shape Type

0 Null Shapes
1 Point
3 PolyLine
5 Polygons
8 Multipoints
11 PointZ
13 PolyLineZ
15 PolygonZ
18 MultiPointZ
21 PointM
23 PolyLineM
25 PolygonM
28 MultiPointM
31 MultiPatch

Shape types not specified above (2, 4, 6, etc., and up to 33) are reserved for future use. Currently, shape files are restricted to contain the same type of shape as specified above. In the future, shape files may be allowed to contain more than one shape type. If mixed shape types are implemented, the shape type field in the header will flag the file as such.

The Bounding Box in the main file header stores the actual extent of the shapes in the file:

The minimum-bounding rectangle orthogonal to the X and Y (and potentially the M and Z) axes that contains all shapes. If the shapefile is empty (that is, has no records), the values for Xmin, Ymin, Xmax, and Ymax are unspecified. Mmin and Mmax can contain "no data" values (see Numeric Types on page 2) for shape files of measured shape types that contain no measures.

Record Headers

The header for each record stores the record number and content length for the record. Record headers have a fixed length of 8 bytes. Table 2 shows the fields in the file header with their byte position, value, type, and byte order. In the table, position is with respect to the start of the record.

Table 1.3 Description of Main File Record Headers

Byte	Position Field	Value Type	Order
Byte 0	Record Number	Record Number	Integer Big
Byte 4	Content Length	Content Length	Integer Big

Record numbers begin at 1.

The content length for a record is the length of the record contents section measured in 16-bit words. Each record, therefore, contributes (4 + content length) 16-bit words toward the total length of the file, as stored at Byte 24 in the file header.

Point

A point consists of a pair of double-precision coordinates in the order X, Y.

```
Point
{
Double X // X coordinate
Double Y // Y coordinate
}
```

Table 1.4 Point Record Contents

Position	Field	Value	Type	Number	Byte Order
Byte 0	Shape Type	1	Integer	1	Little
Byte 4	X	X	Double	1	Little
Byte 12	Y	Y	Double	1	Little

PolyLine

A PolyLine is an ordered set of vertices that consists of one or more parts. A part is a connected sequence of two or more points. Parts may or may not be connected to one another. Parts may or may not intersect one another. Because this specification does not forbid consecutive points with identical coordinates, shapefile readers must handle such cases. On the other hand, the degenerate, zero length parts that might result are not allowed.

```
PolyLine
{
Double [4] Box // Bounding Box
Integer NumParts // Number of Parts
Integer NumPoints // Total Number of Points
Integer[NumParts] Parts // Index to First Point in Part
Point[NumPoints] Points // Points for All Parts
}
```

Table 1.5 PolyLine Record Contents

Position	Field	Value	Type	Number	Byte Order
Byte 0	Shape Type	3	Integer	1	Little
Byte 4	Box	Box	Double	4	Little
Byte 36	NumParts	NumParts	Integer	1	Little
Byte 40	NumPoints	NumPoints	Integer	1	Little
Byte 44	Parts	Parts	Integer	NumParts	Little
Byte X	Points	Points	Point	NumPoints	Little

Note: X = 44 + 4 * NumParts

Polygon

A polygon consists of one or more rings. A ring is a connected sequence of four or more points that form a closed, non-self-intersecting loop. A polygon may contain multiple outer rings. The order of vertices or orientation for a ring indicates which side of the ring is the interior of the polygon. The neighborhood to the right of an observer walking along the ring in vertex order is the neighborhood inside the polygon.

Vertices of rings defining holes in polygons are in a counterclockwise direction. Vertices for a single, ringed polygon are, therefore, always in clockwise order. The rings of a polygon are referred to as its parts. Because this specification does not forbid consecutive points with identical coordinates, shapefile readers must handle such cases. On the other hand, the degenerate, zero length or zero area parts that might result are not allowed.

The Polygon structure is identical to the PolyLine structure, as follows:

```
Polygon
{
Double[4] Box                    // Bounding Box
Integer NumParts                 // Number of Parts
Integer NumPoints                // Total Number of Points
Integer[NumParts] Parts          // Index to First Point in Part
Point[NumPoints] Points          // Points for All Parts
}
```

Table 1.6 Polygon Record Contents

Position	Field	Value	Type	Number	Byte Order
Byte 0	Shape Type	5	Integer	1	Little
Byte 4	Box	Box	Double	4	Little
Byte 36	NumParts	NumParts	Integer	1	Little
Byte 40	NumPoints	NumPoints	Integer	1	Little
Byte 44	Parts	Parts	Integer	NumParts	Little
Byte X	Points	Points	Point	NumPoints	Little

Note: X = 44 + 4 * NumParts

Organization of the Index File

File Header
Record
Record
……
Record

The Index File Header: The index file header is identical in organization to the main file header described above.

The file length stored in the index file header is the total length of the index file in 16-bit words (the fifty 16-bit words of the header plus 4 times the number of records).

Index Records The I'th record in the index file stores the offset and content length for the I'th record in the main file. Table shows the fields in the file header with their byte position, value, type, and byte order. In the table, position is with respect to the start of the index file record.

Table 1.7 Description of Index Records

Position Order	Field	Value	Type	Byte
Byte 0	Offset	Offset	Integer	Big
Byte 4	Content Length	Content Length	Integer	Big

The offset of a record in the main file is the number of 16-bit words from the start of the main file to the first byte of the record header for the record. Thus, the offset for the first record in the main file is 50, given the 100-byte header.

The content length stored in the index record is the same as the value stored in the main file record header.

Organization of the dBase File

The dBASE file (.dbf) contains any desired feature attributes or attribute keys to which other tables can be joined. Its format is a standard DBF file used by many table-based applications in Windows ™ and DOS. Any set of fields can be present in the table.

There are three requirements, as follows:

- The file name must have the same prefix as the shape and index file. Its suffix must be .dbf. (See the example on page 2, in Naming Conventions.)
- The table must contain one record per shape feature.
- The record order must be the same as the order of shape features in the main (*.shp) file.
- The year value in the dBASE header must be the year since 1900.

A shape file stores nontopological geometry and attributes information for the spatial features in a data set. The geometry for a feature is stored as a shape comprising a set of vector coordinates.

Because shape files do not have the processing overhead of a topological data structure, they have advantages over other data sources such as faster drawing speed and editability. Shape files handle single features that overlap or

that is noncontiguous. They also typically require less disk space and are easier to read and write.

Shape files can support point, line, and area features. Area features are represented as closed loop, double-digitized polygons. Attributes are held in a dBASE® format file. Each attribute record has a one-to-one relationship with the associated shape record.

An ESRI shape file consists of a main file, an index file, and a dBASE table. The main file is a direct access, variable-record-length file in which each record describes a shape with a list of its vertices. In the index file, each record contains the offset of the corresponding main file record from the beginning of the main file. The dBASE table contains feature attributes with one record per feature. The one-to-one relationship between geometry and attributes is based on record number. Attribute records in the dBASE file must be in the same order as records in the main file.

The main file (.shp) contains a fixed-length file header followed by variable-length records. Each variable-length record is made up of a fixed-length record header followed by variable-length record contents. The index file (.shx) contains a 100-byte header followed by 8-byte, fixed-length records.

The dBASE file (.dbf) contains any desired feature attributes or attributes keys to which other tables can be joined. Its format is a standard DBF file used by many table-based applications in Windows™ and DOS. The file name must have the same prefix as the shape and index file. Its suffix must be .dbf.

1.4 Existing systems

1.4.1 ArcView GIS 3.0

It’s a powerful, easy-to-use tool that brings geographic information to the desktop. ArcView gives one the power to visualize, explore, query and analyze data spatially.

ArcView is made by Environmental Systems Research Institute (ESRI), the makers of ARC/INFO, the leading geographic information system(GIS) software. It has been helping people solve spatial problems with computers for over 20 years.

One doesn't need to know how to create geographic data in order to use ArcView. ArcView comes with a useful set of ready-to-use data. Additional geographic data sets are available from ESRI and from various third parties to suit almost any requirement one might have. Plus, if one's organization uses ARC/INFO data, one will immediately be able to use ArcView to access all these resources, including vector coverage, map libraries, grids, images and event data.

Working spatially

ArcView can be used by anyone who wants to work spatially. A key feature of ArcView is that it's easy to load tabular data, such as dBASE files and data from database servers, into ArcView so that one can display, query, summarize, and organize this data geographically.

In no time one will be working with the data in a completely new way, seeing patterns not seen before, understanding geographic relationships that were previously hidden, gaining new insights… and achieving new results for business.

Views

With ArcView one works with geographic data in interactive maps called views. Every view features ArcView's unique geographic "Table of Contents", making it easy to understand and control what's displayed. GIS has never been simpler!

Tables

Working with tabular data in ArcView's tables puts one in control. Click on features on a view, and their records highlight in the table showing their attributes. Select records in the table and the features they represent highlight on the view. ArcView's tablesslao have a full range of features for obtaining summary statistics, sorting and querying.

Charts

ArcView's charts offer a powerful business graphics and data visualization capability that is fully integrated into ArcView's geographic environment. One can simply click on features on a view to add them to the chart. ArcView lets you work simultaneously with geographic, tabular and chart representations of your data.

Layouts

ArcView's layouts lets one create high quality, full color maps by first arranging the various graphic elements on-screen the way one wants them. One will get great looking results on a wide range of printers and plotters. Layouts are smart because they have a live link to the data they represent. When one prints a layout, any changes to the data are automatically included, so one knows everything on the map will be up-to-date.

Scripts

ArcView scripts are macros written in Avenue, Arc View's programming languages and development environment. With Svenue one can customize almost every aspect of ArcView, from adding a new button to run a script one writes, for creating an entire custom application that can distribute.

Projects

All the components of the ArcView session: views, tables, charts, layouts, and scripts are conveniently stored in one file called a project. ArcView's Project window shows the contents of the project and makes it easy to manage all the work.

1.4.2 ArcMap

ArcMap is also a software for desktop GIS and mapping. It provides better functionality for editing as compared to ArcView.

ArcMap provides a rich set of tools that perform many common network analysis tasks on geometric network. Some of the common network analysis that can be performed using ArcMap are:

Trouble Call analysis: determines the likely cause of a problem based on the location of customers with service problem.

1.4.3 Pragati GIS

Pragati is the project which is being developed at the Bisag. Its main aim is to support the features of the GIS. It is necessary to study and understand ArcView and ArcMap because it aims to include all the functionality of the ArcView and the ArcMap.

Pragati is software which is built at BISAG and which is made on the lines of ArcView S/W. Actually ArcView costs a lot of rupees and hence all the government organization who wants to use ArcView has to purchase it by paying a huge amount of money. So BISAG decided to build a S/W which is identical to ArcView and hence the result was Pragati S/W. Till now various facilities has been added to Pragati and still other modules are being added to built it just like ArcView. Our project that is to build and an ActiveX control that implements various Pen and Brush style is also a module to be added to this S/W.

Pragati S/W is build using Visual Basic 6.0 as it provides an excellent Graphical user interface. While the whole code is written in Visual c++ as it is very good at graphic application with a rich set of graphic functions. Again for graphic application time to run the application is very critical and VC++ is faster at run time and hence it is the most suitable platform to write the whole logic.

It contains various features like:

- Read and Display Shape file
- Allow the user to identify the attribute on map
- Display all types of labels on layers and all types of texts on controls itself
- Have hierarchy map facility i.e. the user can navigate from one layer to another layer
- Select by theme
- Printing and print preview

Chapter 2

THESIS PROFILE

- 0BJECTIVE
- SCOPE
- THESIS PLATFORM
- PROJECT SUMMARY

2.1 0BJECTIVE

Since 1998 BISAG (Gandhinagar) has been working on the task of creating GIS applications, which can be distributed independently throughout the various Governmental Departments of the Gujarat State. They all require GIS software for simplifying their work. There are various GIS softwares available in the market like Arc/Info, Arc View, MapInfo etc, but these softwares are very costly .So BISAG decided to develop its own GIS software (Pragati GIS) that can meet all the needs of government. Arc View includes the facility of Geoprocessing functionality of overlays mainly Intersection and Union of the maps which has not been included in Pragati GIS and since these features helps in solving many real world problems I decided to work on it.

2.2 SCOPE

Thesis Definition

To develop Overlays for the GIS mainly Intersection and Union of the maps using Microsoft Visual C++ and Microsoft Visual Basic 6.0.

Feature

Intersection helps in finding the common area of the two maps with different features and combines all attributes of both the maps in that area and Union helps in finding the common area having combine features of both the maps as well as the separate areas of the two maps having their own features only.

So many times it is required to get combined information from two different maps. Like we want to get the view of the cities which are populated more than 1,00,000 and also having local train network. So here we need to intersect two maps one is related to the cities having population attribute and other describing the cities having local train network attribute.

Likewise if we want the area which is either forest area or area having the pollution more than some specific value than we have to do the Union of the two maps one having the attribute forest and other with pollution values.

So my work is to do this thing in most efficient way to utilize memory and processor efficiently and for that defining new algorithms and also implementing it using Visual Basic and Visual C++.

2.3 THESIS PLATFORM

2.3.1 Platform Details

Operating System:- Windows 2000

Programming Languages:- Visual C++ and Visual Basic

To implement, test and validate the efficiency of the algorithms concerning the Intersection and Union of the maps we need to develop language skill in

-Visual Basic

-Visual Studio

We need to implement our research work using two different languages because we need to do two things,

- GUI is to be done in **Visual Basic** which has to be integrated in Pragati S/W
- Coding is to be done in **Visual C++** which has to be integrated in GGIS control

As we are using VB to develop User Interface, the user is provided with an excellent Graphical User Interface (GUI) and project is very user friendly. So output or result will be displayed as an image to the user and it is stored in the shape file format. This graphical display of the output is stored in one of the three feature types point, line, and polygon in file with the format .shp.

As it has attributes related to each feature point, line, polygon stored in the database file, user can have the required attribute values of the resultant map in the database file stored with the .dbf format.

So we have used "Microsoft Visual Studio" for the implementation of research work in Visual Basic and Visual C++.

2.3.2 Motives for Platform

Windows 2000

Windows 2000 Professional is suitable operating system for geographic system development and also for the use of the software related to the geographic system. Also it gives more update facilities as Microsoft's new Web-based resource site, automates driver and system file updates, and provides up-to-date technical support.

Windows 2000 can review device drivers and system software on your computer, compare those findings with a master database on the Web, and then recommend and install updates specific to your computer. You can also revert to a previous device driver or system file using the uninstall option.

Visual C++

The main reason for using Microsoft VC++ as the development tool is that it provides a variety of new and improved features for managing projects and subprojects, editing code and resources, managing classes, code reuse and code generation and also we need to integrate the module with the existing system.

Visual C++ 6.0 includes the comprehensive Microsoft foundation classes, which simplify and speed development of window applications. It includes sophisticated resource editor to design the complex dialog boxes, menu, toolbars, images, and many other elements of modern window application. There is an excellent integrated development environment called developer studio that present graphical views of your application's structure as you develop it. Totally integrated debugging tool lets you examine in minute detail every aspect of your program as it runs.

Visual Basic

Visual Basic 6.0 has emerged as one of the standard windows programming language and it has become must for all software people for developing applications in visual environment, rather than writing numerous lines of code to describe the appearance(GUI) and location of interface elements, VB provides method to simply add objects and place them on screen also it has rich set of APIs to support easy and quick development.

2.3.3 Features of the languages

Visual C++

The Project Workspace

The Project Workspace window offers three views: in Class View project classes, member variables and functions are shown; in File View files comprising a project are visible and in Resource View resource file components can be manipulated.

Editor

The editor offers improved compatibility with other popular editors.

The Class Wizard and the Wizard Bar

The Class Wizard offers support for OLE control development. Many Class Wizard features can easily be accessed in the Wizard Bar.

Component Gallery

Classes that are created through Class Wizard or entire projects created through AppWizard can be added to the Component Gallery and later inserted to other applications. The Component Gallery comes with many useful tools including OLE controls etc.

Debugging

The integrated debugger has redesigned windows. It also offers a new feature data tips i.e. positioning the mouse cursor over a symbol in an editor window during debugging causes a tool tip style window to appear, displaying the current value of the symbol.

Integration with other Tools

The Developer Studio offers a high level of integration with Microsoft development tools such as the Microsoft Developer Library.

Visual Basic

Visual Basic can serve as an ideal front end tool for the client to interact. It has got connectivity mechanism for all types of database situated far and wide in a network and so it can gratify to the needs of large body of clients. Using the latest ActiveX technologies it can integrate the functionalities provided by other applications. The final application is a true EXE file and so can be freely distributed.

2.4 THESIS SUMMARY

Thesis Title

"To develop Overlays for the GIS mainly Intersection and Union"

Front End Tools

Microsoft Visual Basic

GUI is to be done in Visual Basic which has to be integrated in Pragati S/W

Back End Tools

Microsoft Visual C++

Coding is to be done in **Visual C++** which has to be integrated in GGIS control

Thesis Guide

Shri Shashikant Sharma

Organization

Bhaskaracharya Institute for space Application and GeoInformatics,
Govt. of Gujarat (Science & Technology),
Gandhinagar.

Submitted By

Keyur Patel

Submitted To

Institute Of Technology,
Nirma University
Ahmedabad.

Chapter 3

SYSTEM ANALYSIS

- Problem Definition
- Fact Finding Techniques
- Requirement Analysis
- Feasibility Analysis

3.1 PROBLEM DEFINITION

To develop Overlays for the GIS mainly Intersection and Union of the maps using Microsoft Visual C++ and Microsoft Visual Basic 6.0.

3.2 FACT FINDING TECHNIQUES

Fact Finding Techniques

During requirement determination phase, the system analyst has to find out how the current system works and what is expected from a new system, for that it is required to spend considerable time talking to users and gathering all relevant information for the project.

Information Sources

- Users of the system.
- Forms and documents used in the organization.
- Procedure manuals and rulebooks which specify various activities are carried out in the organization.
- Various reports used in the organization.
- Computer programs of existing system.

3.3 REQUIREMENT ANALYSIS

The description of the service and the constraints are the requirements of the system and the processes involved in the requirement engineering are:

- Finding out
- Analyzing
- Documenting and
- Checking these services and constraints

The process activities are:

Domain Understanding:

Analyst must develop understanding of the application domain and therefore I spent some initial time for domain understanding like using maps, becoming aware of map manipulation software etc.

I took help from software like Arc View which helped me to understand what to build and what are the advantages and disadvantages of that. I Considered user as a point of view and developed a user friendly system.

Requirement Classification:

Requirements are classified as follows:

The main requirements are

- User requirements
- System requirements

Above Requirements can be further classified as:

- Functional requirements
- Non-functional requirements
- Domain requirements

User Requirements

There are mainly following types of users:-

- Visitors and tourists

  Visitors and tourists are mainly interested in beautiful locations, emergency services like Banks, police station, hotels etc.
- Citizens

  Citizens want to get information about emergency services like Banks, police station, hotels all together at one place so we need to do intersection of maps showing the banks information, police station information, hotels information in the city etc.

- Govt. Officials

  Different departments of any govt. can use the system to acquire information about any place which combines different features like the area which is having rain up to some value and also having population up to some value to do the survey and to apply some schemes like schemes regarding farmers may be based on the type of the soil and the rain in that area so intersection and union feature can be used to see the combining and separate effects of these two features soil type and the rain.
- Education Departments

  To see which area is having Higher secondary, secondary and primary school facilities as well as good transportation in that area to provide facility to the students living far away from these schools.
- Health Departments

  To see how many private hospital, govt. hospital are covered in the particular area which might be having another feature like area having populatin more than 1,00,000 people. This can be helpful to develop new hospitals in the more populated area.

System Requirements

System Requirements state which kind of services, functions and facility should be given to users.

Users are allowed to retrieve information of places.

Functional Requirements

Since this project uses database and control, this needs the retrieval of information from the database. It needs the interaction between Visual Basic Container and Visual C++ control.

Non Functional Requirements

Product Requirements

- Efficiency

The system should provide easy and fast access without consuming more cost.

- Reliability

User should never be surprised by the behavior of the system and it should also provide meaningful feedback when errors occur so that user can recover from errors.

Conflict Resolution:

Here, requirement conflicts are handled so that users can distinguish themselves. Like some facility may not be used by other user, there must not be any objection from other users.

Requirement Validation:

Requirement validation is concerned with showing that requirements actually define the system. If this validation is inadequate errors in the requirements will be propagated to the system design and implementation.

3.4 FEASIBILITY ANALYSIS

A feasibility study is a short, focused study which aims to answer a number of questions:

- Does the system contribute to the overall objectives of the organization?
- Can the system be implemented using current technology and given cost and schedule constraints?
- Can the system be integrated with systems which are already in place?

Operational Feasibility

Operational Feasibility measures how well the solution will work in the organization and how will end-user and management feels about system. Proposed system is helpful for tourists, citizen and visitors. It will help them to get appropriate and adequate information.

On studying the operational feasibility of the project the following conclusions were derived:

- Developed system will provide the adequate throughput and all necessary information to end users.
- It will provide advantageous and reliable services.

Thus, it is operationally feasible to develop the proposed system.

Technical Feasibility

Technical Feasibility tries to answer the following questions to make the software feasible to develop.

- The software or tools necessary for building or running the application are easily available or not?
- Compatibility amongst software exists or not?
- Are developers aware of these technologies?
- What about the alternative of these chosen technologies?

Factors Considered

- Here we have to consider those tools which will be required for developing the project?
- The tools that are available and tools that will be required are taken into account.
- I have to work on Visual Basic and Visual C++ for GIS. The company's former versions are developed in VC++, So the company already has the licensed version of Visual Studio with MSDN.
- Various Shapefiles are desired as maps are to be displayed with the help of shapefiles. It is desired to have as much as shapefiles possible. BISAG is

Govt. of Gujarat enterprise .So, it has all official maps, DBF files as data source and index files are also available.

- Well known softwares like Arc View, Arc Map are available.

Considering all the above points it is technically feasible to carry on the project.

Economical feasibility

Economical feasibility addresses to the following issues:

- Is the organization having suitable budget to develop the proposed system?
- How much profit can be earned from the system by an organization?
- Would it be cost-effective to develop the system or it is worthwhile to remain with the current system?

As the development tools are available free of cost, there isn't any burden of buying them. Pragati does not have Overlays of the GIS, so it is certainly required .Extra funds are not required to develop the system, so it is economically feasible to the organization.

Implementation feasibility

Under the study of Implementation feasibility following issues are considered:

- Is it possible to install the software within the given environment?
- Will organization, user support for the installation of software?
- Will proposed system cause any harm to the operations of the organization?

Operationally, this system can be installed and it will work according to its functionality.

Chapter 4
SYSTEM DESIGN

- System Design
- Architectural Design
- Flow Charts
- Object Oriented Design
- Structure Oriented Design
- Dynamic Modeling

4.1 SYSTEM DESIGN

During the designing of the system I followed the modular approach .Firstly I had discussed with Senior Scientist the basic needs and usages of the project to be undertaken. Then I took help from Arc View which includes the Geo-Processing Wizard extension which contains Overlays functions like Union, Intersection, Clipping, Dissolve etc. in order to understand it's operation, so that various parameters and conditions to be considered can be known. After that I started the algorithmic design of the system.

4.2 ARCHITECTURAL DESIGN

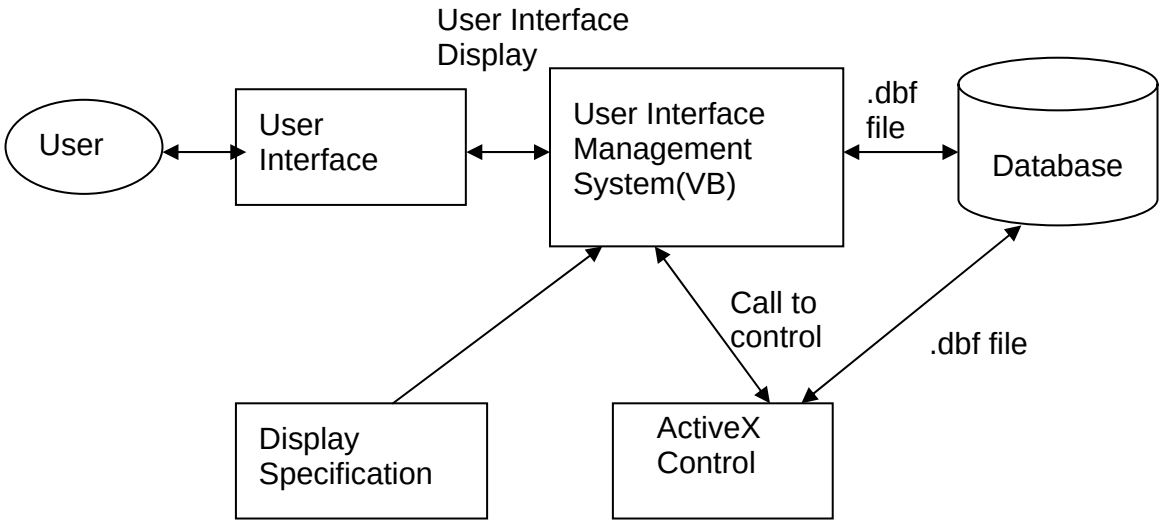


Figure 4.1 Architectural Design of the Proposed System

The figure shows the interaction of user with system. The major components of the model are:-

User:

User interacts with the container (Visual Basic). It does not know the internal operation of the system.

User Interface Management System:

This calls various methods, properties developed in the ActiveX control. It interacts with the ActiveX control.

ActiveX Control:

This does the internal operations, various methods and properties are developed in this control.

Database:

Database stores data necessary for the process.

4.3 FLOW CHARTS

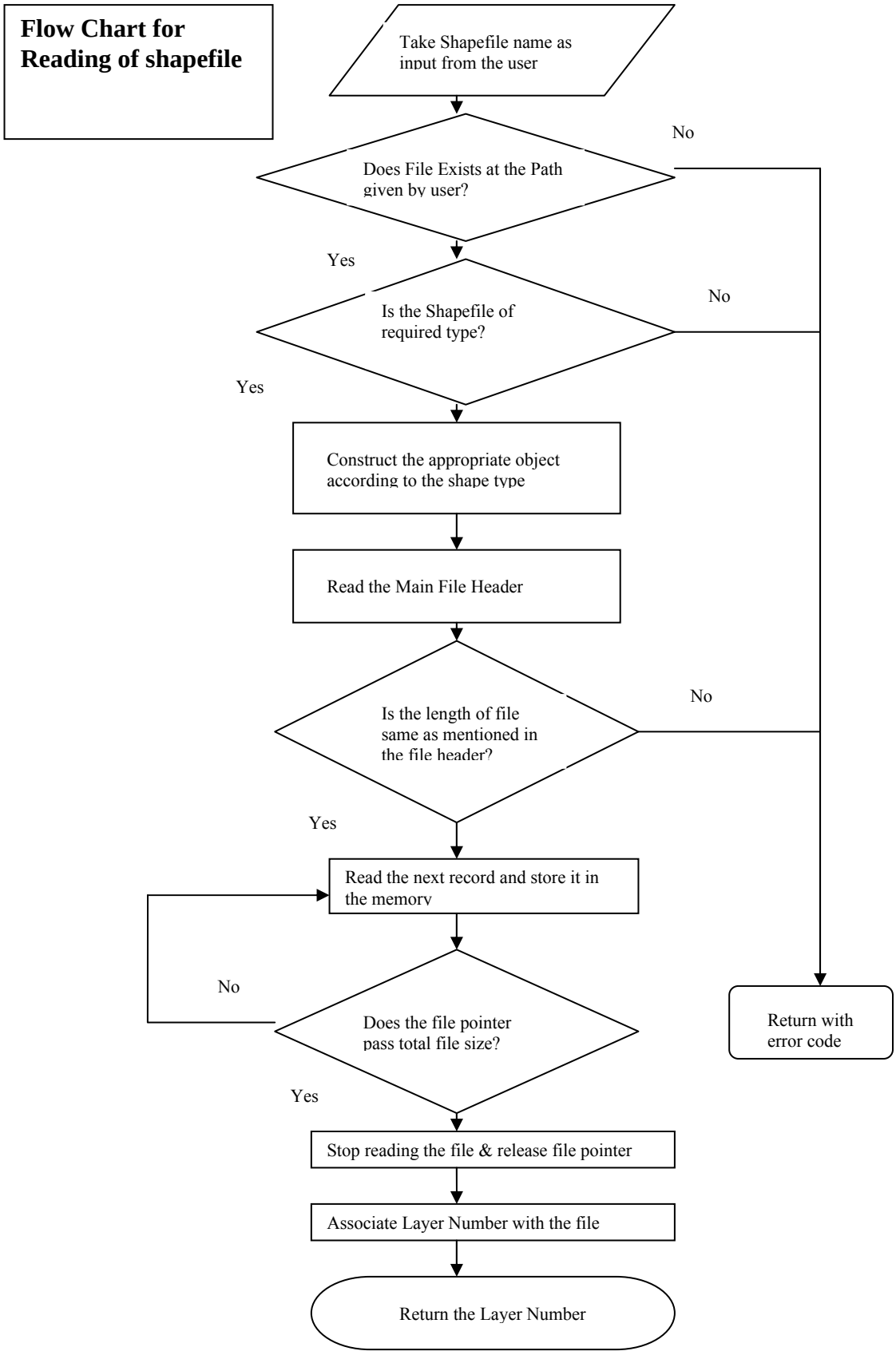


Figure 4.2 Flow Chart for Reading of shapefile

Flow Chart for Display of shapefile

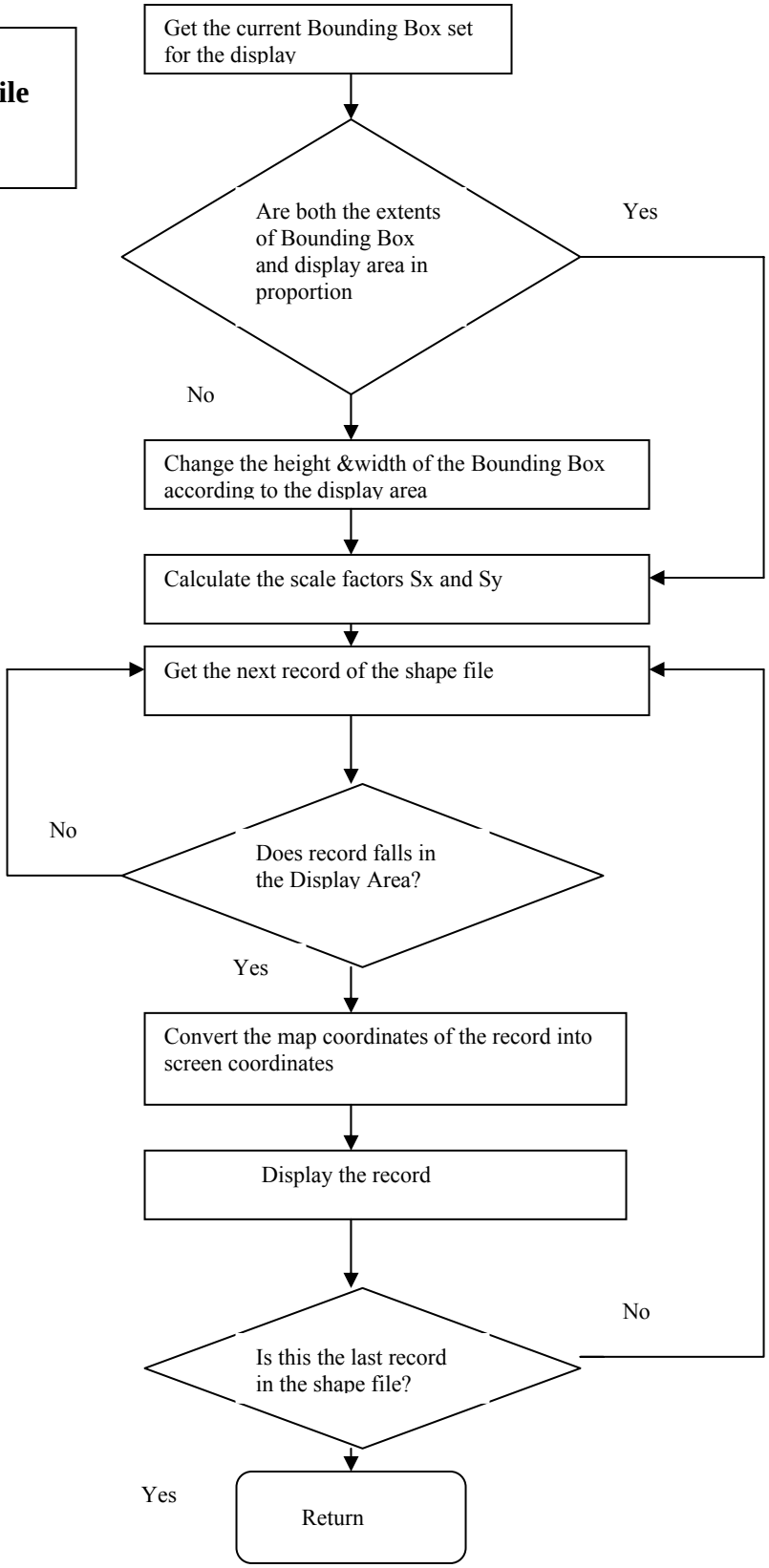
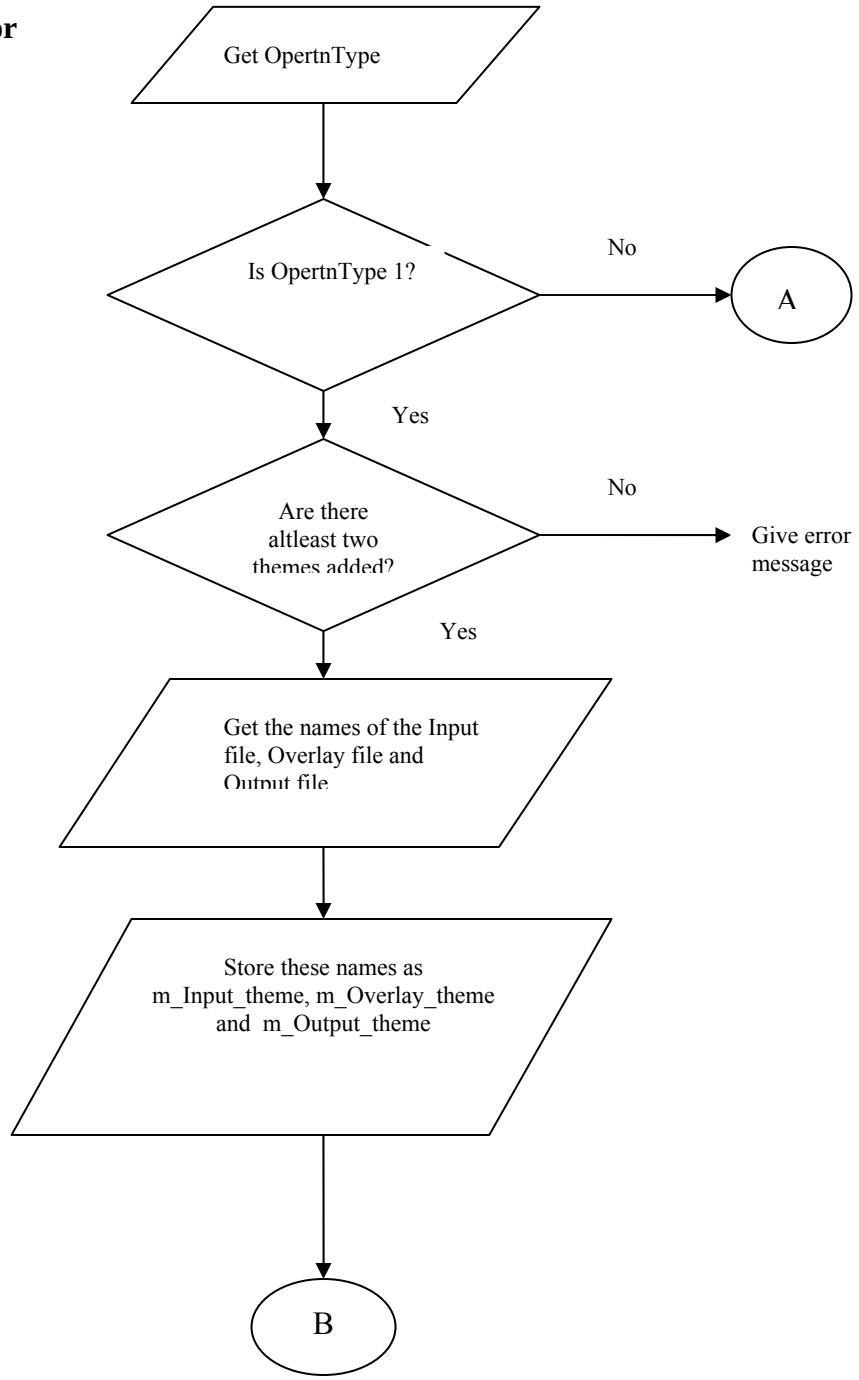
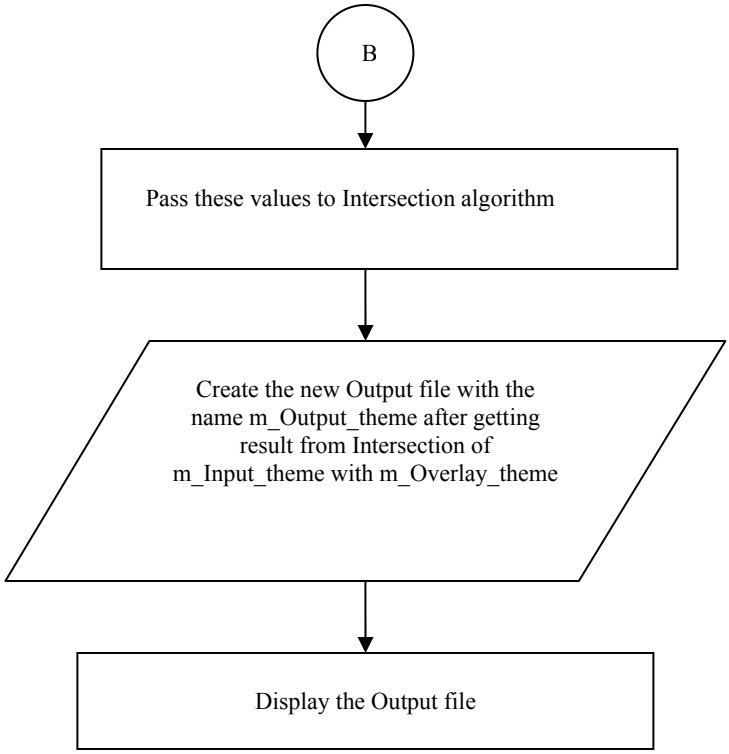
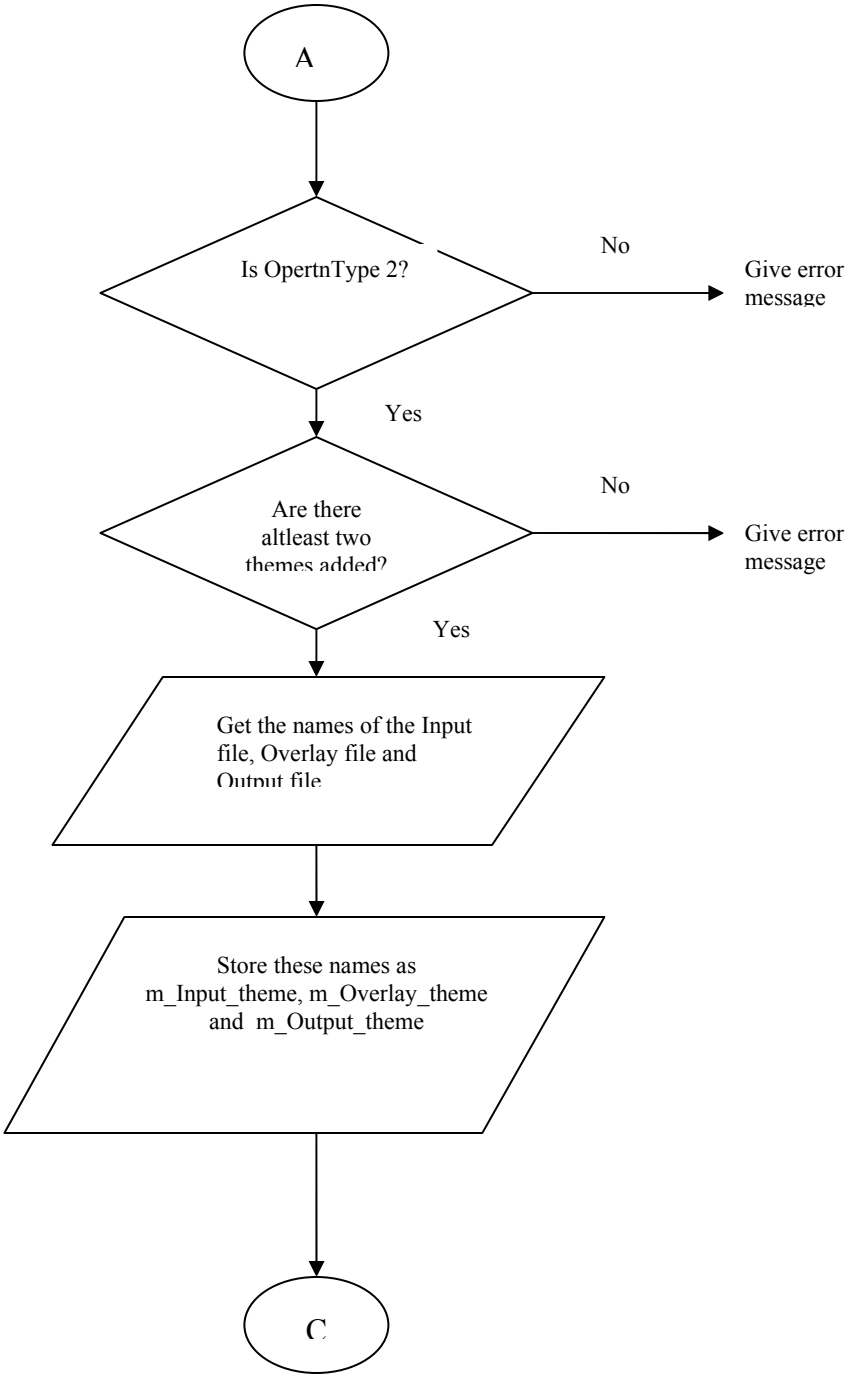


Figure 4.3 Flow Chart for Display of shapefile

Flow chart for finding the Intersection or Union




B
Pass these values to Intersection algorithm
Create the new Output file with the name m_Output_theme after getting result from Intersection of m_Input_theme with m_Overlay_theme
Display the Output file


A
Is OpertnType 2?
No
Give error message
Yes
Are there altleast two themes added?
No
Give error message
Yes
Get the names of the Input file, Overlay file and Output file
Store these names as m_Input_theme, m_Overlay_theme and m_Output_theme
C

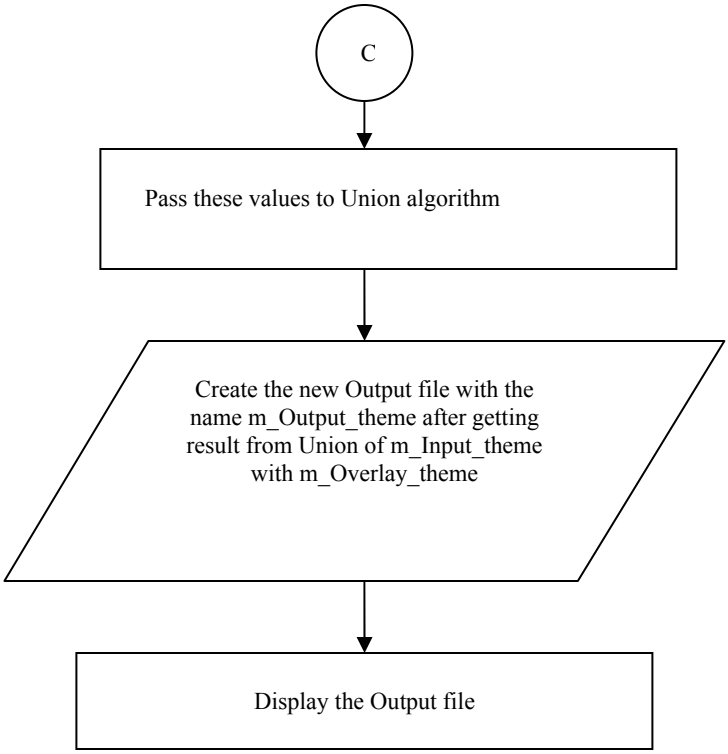


Figure 4.4 Flow chart for finding the Intersection or Union

4.4 OBJECT ORIENTED DESIGN

The object model describes the structure of objects in a system-their identity, their relationship to other objects, their attributes and their operations.

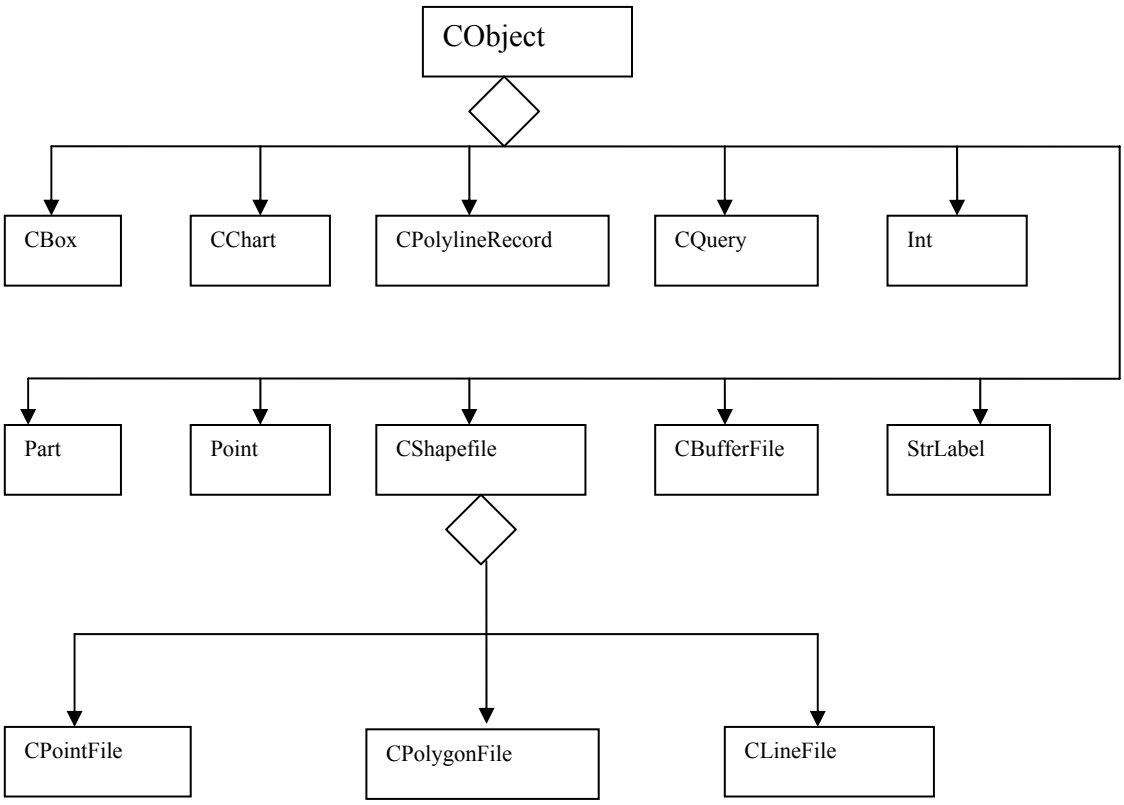


Figure 4.5 Object model for ActiveX Control

4.5 STRUCTURE ORIENTED DESIGN

Structural model shows how a function is realized by a number of other functions it calls. Structural charts are the graphical way to represent decomposition hierarchy.

Structured Charts

Structured Diagram steps

1) Identify system processing transformations
2) Identify input transformation.
3) Identify output transformation.

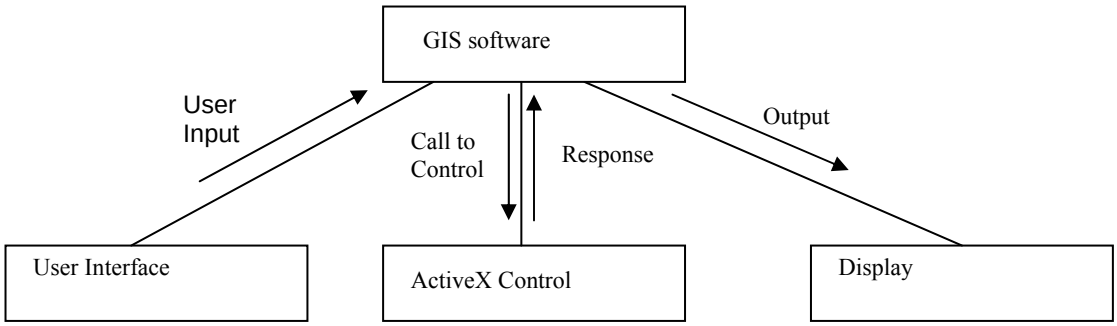


Figure 4.6 Initial structured chart of the system

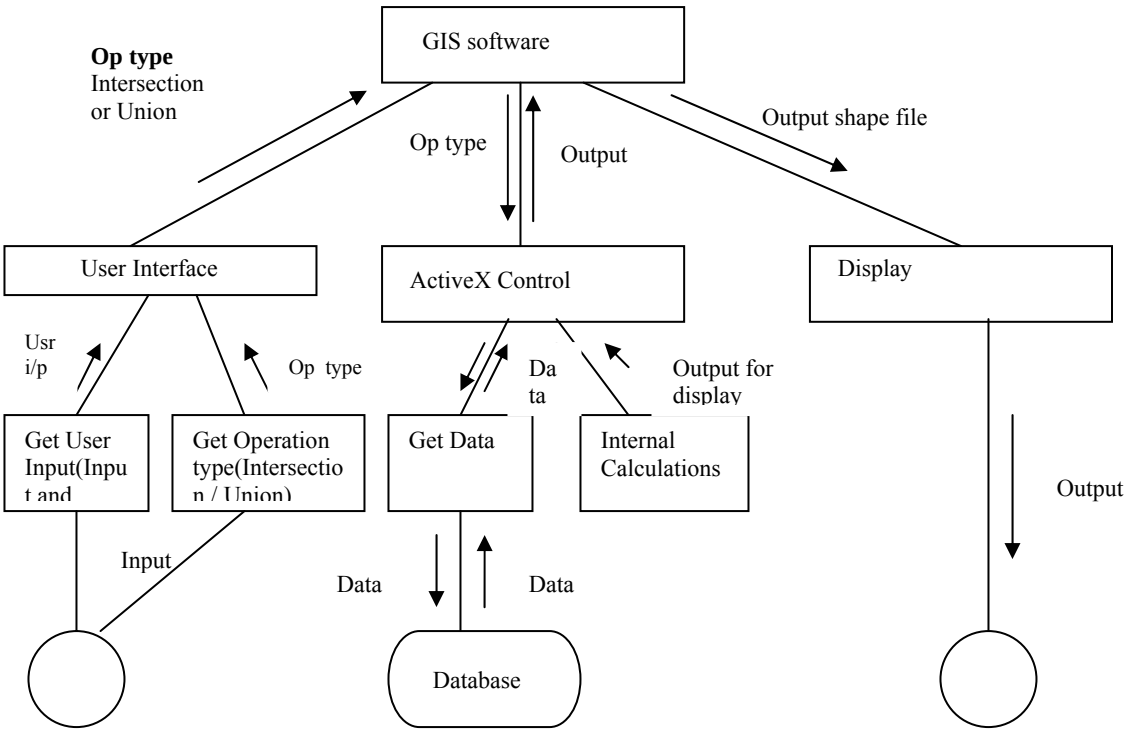


Figure 4.7 Final Structured chart of the system

4.6 DYNAMIC MODELING

State diagrams are used to represent dynamic modeling.

Conventions used in state diagram:

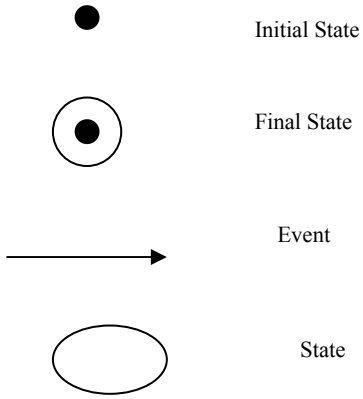
Event:

Event is something that happens at a point of time.

States:

The attribute values and links held by an object are called states.

Conventions used in state diagram are:-



State Diagram for finding Intersection and Union

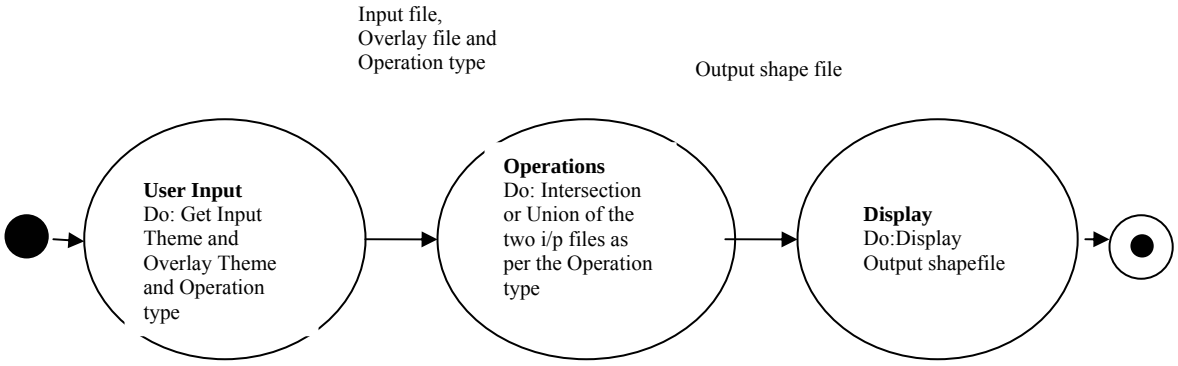


Figure 4.8 State Diagram for finding Intersection and Union

Chapter 5

Procedural Design

- Algorithms
- Data Flow Design

5.1 Algorithms

Why efficient Algorithm?

GIS algorithms for complex processes are often built up from simple ones

- The algorithm illustrates one of the principles of this type of programming, that there are numerous special cases which have to be dealt with.
- In general, the cost of obtaining a solution increases with the problem size
    - If the size of the problem is sufficiently small, even an inefficient algorithm will not cost much to run
    - Consequently, the choice of an algorithm for a small problem is not critical unless the problem has to be solved many times.

As mentioned before shapefile is consisting of hundreds of points, lines and polygons so we have big problem size here, also strategy like point in polygon is applied for both intersection and union repeatedly while doing intersection and union of two themes so we require to have more efficient algorithms over here.

Main work to do

- **Intersection**
  Line-Line intersection
  Line-Polygon intersection
  Polygon-Polygon intersection
- **Union**
  Polygon-Polygon union

For both the Intersection and Union we have to check repeatedly that whether point lies inside or outside of the Polygon so one more thing to do is

- **Point in Polygon strategy**

So I will start with Point in Polygon strategy, than Intersection and at the end I will explain Union.

5.1.1 Determining if a point lies on the interior of a polygon

Inside vs. Outside

One definition of whether a point is inside a region is the Jordan Curve Theorem. Essentially, it says that a point is inside a polygon if, for any ray from this point, there is an odd number of crossings of the ray with the polygon's edges (Figure 1). This definition means that some areas which are enclosed by a polygon are not considered inside. The center pentagonal area inside a star is classified as outside because any ray from this area will always intersect an even number of edges.

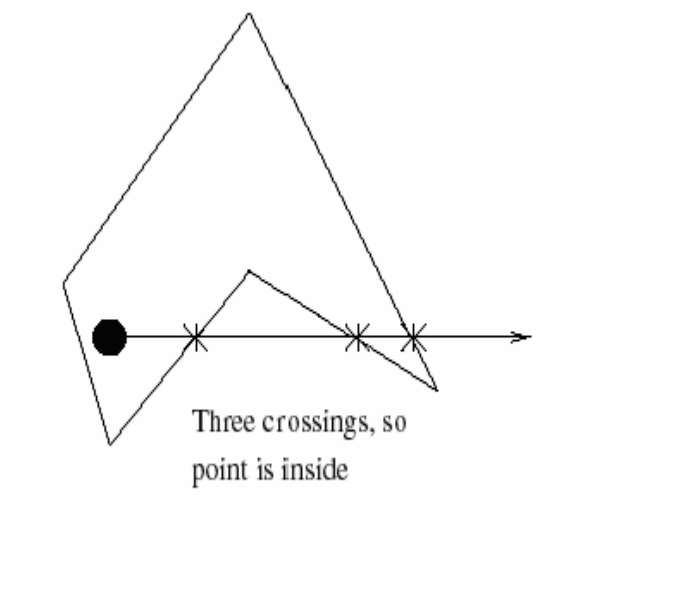


Figure 5.1 Crossings Test

If the entire area enclosed by the polygon is to be considered inside, then the winding number is used for testing. This value is the number of times the polygon goes around the point. For example, a point in the center pentagonal area formed by a star has a winding number of two, since the outline goes around it twice. If the point is outside, the polygon does not wind around it and so the winding number is

zero. Winding numbers also have a sign, which corresponds to the direction the edges wrap around the point.

Complex polygons can be formed using either polygon definition. A complex polygon is one that has separate outlines (which can overlap). For example, the letter "R" can be defined by two polygons, one consisting of the exterior outline, the other the inside hole in the letter. Most point in polygon algorithms can be easily extended to test multiple outline polygons by simply running each polygon outline through separately and keeping track of the total parity.

3D to 2D

In ray tracing and other applications the original polygon is defined in three dimensions. To simplify computation it is worthwhile to project the polygon and test point into two dimensions. One way to do this is to simply ignore one component. The best component to ignore is usually that which, when ignored, gives the largest area polygon.

This is easily done by taking the absolute value of each component of the polygon plane's normal and finding the largest. The corresponding coordinates in the polygon are then ignored. Precomputing some or all of this information once for a polygon uses more memory but increases the speed of the intersection test itself.

While the above method is simple and quick, another projection may be useful when the polygon vertices are not guaranteed to lay on a single plane. While such polygons could be considered ill defined, they do crop up. For example, if a spline surface is tessellated into quadrilaterals instead of triangles, then the quadrilaterals are likely to be ill defined in this way. If a component is simply dropped, cracking can occur between such polygons when they are cast upon different planes.

One solution is to tessellate such polygons into triangles, but this may be impractical for a variety of reasons. Another approach is to cast all polygons tested onto a plane perpendicular to the testing ray's direction. A polygon might have no area on this plane, but the ray would miss this polygon anyway. Casting a polygon

onto an arbitrary plane means having to perform a matrix transformation, but this solution does provide a way around potential cracking problems.

Bounding Areas

Point in Polygon Algorithms benefit from having a bounding box around polygons with many edges. The point is first tested against this box before the full polygon test is performed; if the box is missed, so is the polygon. Most statistics generated in this Gem assume this bounding box test was already passed successfully.

In ray tracing, Worley points out that the polygon's 3D bounding box can be treated like a 2D bounding box by throwing away one coordinate, as done above for polygons. By analysis of the operations involved, it can be shown to be generally more profitable to first intersect the polygon's plane then test whether the point is inside the 2D bounding box rather than first testing the 3D bounding box. Other bounding box variants can be found in Woo.

Crossings Test

One algorithm for checking a point in any polygon is the crossings test. The earliest presentation of this algorithm is Shimrat, though it has a bug in it. A ray is shot from the test point along an axis (+X is commonly used) and the number of crossings is computed (Figure 5.1). Either the Jordan Curve or winding number test can be used to classify the point.

What happens when the test ray intersects one or more vertices of the polygon? This problem can be ignored by considering the test ray to be a half-plane divider, with one of the half-planes including the ray's points. In other words, whenever the ray would intersect a vertex, the vertex is always classified as being infinitesimally above the ray. In this way, no vertices are intersected and the code is both simpler and speedier.

One way to think about this algorithm is to consider the test point to be at the origin and to check the edges against this point. If the Y components of a

polygon edge differ in sign, then the edge can cross the test ray. In this case, if both X components are positive, the edge and ray must intersect and a crossing is recorded. Else, if the X signs differ, then the X intersection of the edge and the ray is computed and if positive a crossing is recorded.

MacMartin pointed out that for polygons with a large number of edges there are generally runs of edges which have Y components with the same sign. For example, a polygon representing Brazil might have a thousand edges, but only a few of these will straddle any given latitude line and there are long runs of contiguous edges on one side of the line.

So a faster strategy is to loop through just the Y components as fast as possible; when they differ then retrieve and check the X components. Compared to the basic crossings test the MacMartin test was up to 1.8 times faster for polygons up to 100 sides, with performance particularly enhanced for polygons with many edges.

Other optimizations can be done for this test. Preprocessing the edge list into separate X and Y component lists, with the first vertex added to the end, makes for particularly tight loops in the testing algorithm. This is not done in the code provided so as to avoid any preprocessing or additional memory costs.

A somewhat faster crossings test is by turning the division for testing the X axis crossing into a tricky multiplication test this part of the test became faster, which had the additional effect of making the test for "both to left or both to right" a bit slower for triangles than simply computing the intersection each time. The main increase is in triangle testing speed, which was about 15% faster; all other polygon complexities were pretty much the same as before.

On machines where division is very expensive (not the case on the HP 9000 series on which I tested) this test should be much faster overall than the old code. Your mileage may (in fact, will) vary, depending on the machine and the test data, but in general I believe this code is both shorter and faster.

Angle Summation Test

The worst algorithm in the world for testing points is the angle summation method. It's simple to describe: sum the signed angles formed at the point by each edge's endpoints. If the sum is near zero, the point is outside; if not, it's inside (Figure 5.2). The winding number can be computed by finding the nearest multiple of 360 degrees.

The problem with this scheme is that it involves a square root, arc-cosine, division, dot and cross product for each edge tested. In timing tests the MacMartin test was up to 63 times faster than the angle summation test for polygons with up to 100 sides. In fairness, the angle algorithm can be sped up in various ways, but it will still always be faster to use any other algorithm in this Gem.

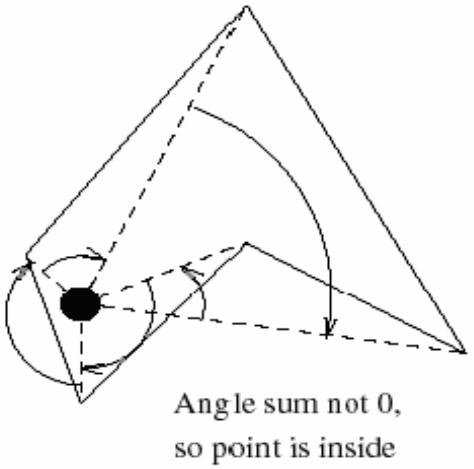


Figure 5.2 Angle Summation Test

Triangle Tests

In Graphics Gems, Didier Badouel presents a method of testing points against convex polygons. The polygon is treated as a fan of triangles emanating from one vertex and the point is tested against each triangle by computing its barycentric coordinates. As Berlin points out, this test can also be used for non-convex polygons by keeping a count of the number of triangles which overlap the point; if odd, the point is inside the polygon (Figure 5.3).

Unlike the convex test, where an intersection means that the test is done, all the triangles must be tested against the point for the non-convex test. Also, for the non-convex test there may be multiple barycentric coordinates for a given point, since triangles can overlap.

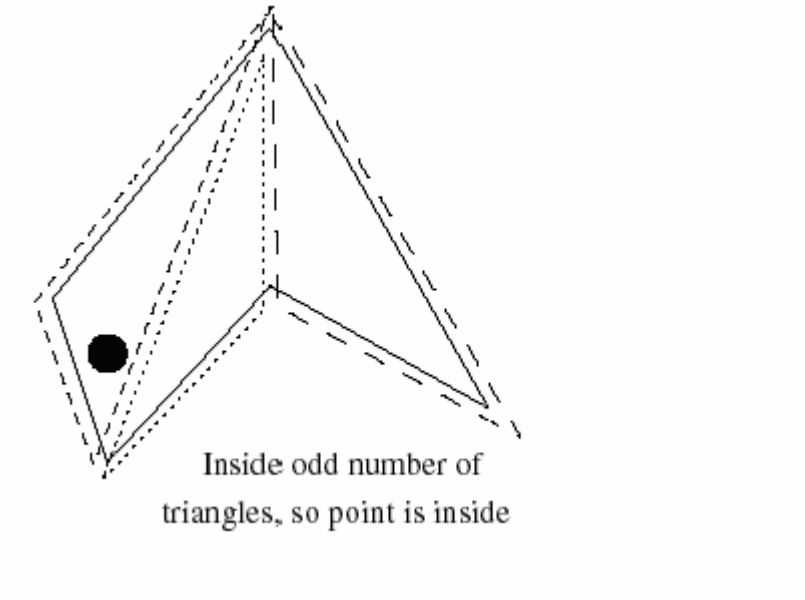


Figure 5.3 Triangle Fan Test

When the number of sides is small, the barycentric test is comparable to the MacMartin test in speed, with the additional bonus of having the barycentric coordinates computed. As the number of sides approached 100, the MacMartin test becomes 3 to 6 times faster than the barycentric method.

A faster triangle fan tester proposed by Green is to store a set of half-plane equations for each triangle and test each in turn. If the point is outside any of the three edges, it is outside the triangle. The half-plane test is an old idea, but storing the half-planes instead of deriving them on the fly from the vertices gives this scheme its speed at the cost of some additional storage space.

For triangles this scheme is the fastest of all of the algorithms discussed so far, being almost twice as fast as the MacMartin crossings test. It is also very simple to code and so lends itself to assembly language translation.

Both the half-plane and barycentric triangle testers can be sped up further by sorting the order of the edge tests. Worley and Haines note that the half-plane triangle test is more efficient if the longer edges are tested first. Larger edges tend to cut off more exterior area of the polygon's bounding box, and so can result in earlier exit from testing a given triangle. Sorting in this way makes the test up to 1.6 times faster, rising quickly with the number of edges in the polygon. However, polygons with a large number of edges tend to bog down the sorted edge triangle algorithm, with the MacMartin test being from 1.6 to 2.2 times faster for 100 edge polygons.

For the general test a better ordering for each triangle's edges is to sort by the area of the polygon's bounding box outside the edge, since we are trying to maximize the amount of area discarded by each edge test. This ordering provides up to another 10% savings in testing time. Unfortunately, for the convex test below, this ordering actually loses about 10% for regular polygons due to a subtle quirk. As such, this ordering is not presented in the statistics section or the code.

Convex Polygons

Convex polygons can be intersected faster due to their geometric properties. For example, the crossings test can quit as soon as two Y sign difference edges are found, since this is the maximum that a convex polygon can have. Also, more polygons can be categorized as "convex" for the crossings test by checking only the change in the Y direction (and not X and Y as for the full convexity test).

For example, a block letter "E" has at most two Y intersections for any test point's horizontal line, so it can be treated as convex when using the crossings test. Convexity is sufficient but not necessary for all of the algorithms discussed in this section.

The triangle fan tests can exit as soon as any triangle is found to contain the point. This algorithm can be enhanced by both sorting the edges of each triangle by length and also sorting the testing order of triangles by their areas.

Larger triangles are more likely to enclose a point and so end testing earlier. Using both of these sorting strategies makes convex testing 1.2 times faster for squares and 2.5 times faster for regular 100 sided polygons.

Another strategy is to test the point against each exterior edge in turn. If the point is outside any edge, then the point must be outside the entire convex polygon. This algorithm uses less additional storage than the triangle fan and is very simple to code.

The order of edges tested affects the speed of the algorithm; testing edges which cut off the most area of the bounding box earliest on is the best ordering. Finding this optimal ordering is non-trivial, but doing the edges in order is often the worst strategy, since each neighboring edge usually cuts off little more area than the previous. Randomizing the order of the edges makes this algorithm up to 10% faster overall for regular polygons. However, even then the triangle fan algorithm with sorting is up to 1.35 times faster for 100 edge regular polygons.

The exterior edge strategy looks for an early exit due to the point being outside the polygon, while the triangle fan convex test looks for one due to the point being inside. For example, for 100 edge polygons if all points tested are inside the polygon the triangle fan is 1.7 times faster; if all are outside the exterior test is more than 11 times faster (but only 3 times faster if the edges are not randomized). So when the polygon/bounding box area is low the exterior edge strategy might be best.

A method with O(log n) performance is discussed by Preparata and Shamos. The polygon is preprocessed by adding a central point to it and is then divided into wedges. The angles from an anchor edge to each wedge's edges are computed and saved, along with half-plane equations for each wedge's polygon edge. When a

point is tested, the angle from the anchor edge is computed and a binary search is used to determine the wedge it is in, then the corresponding polygon edge is tested against it (Figure 5.4). This algorithm is slower for polygons with few edges because the startup cost is high, but the binary search makes for a much faster test when the number of edges is high.

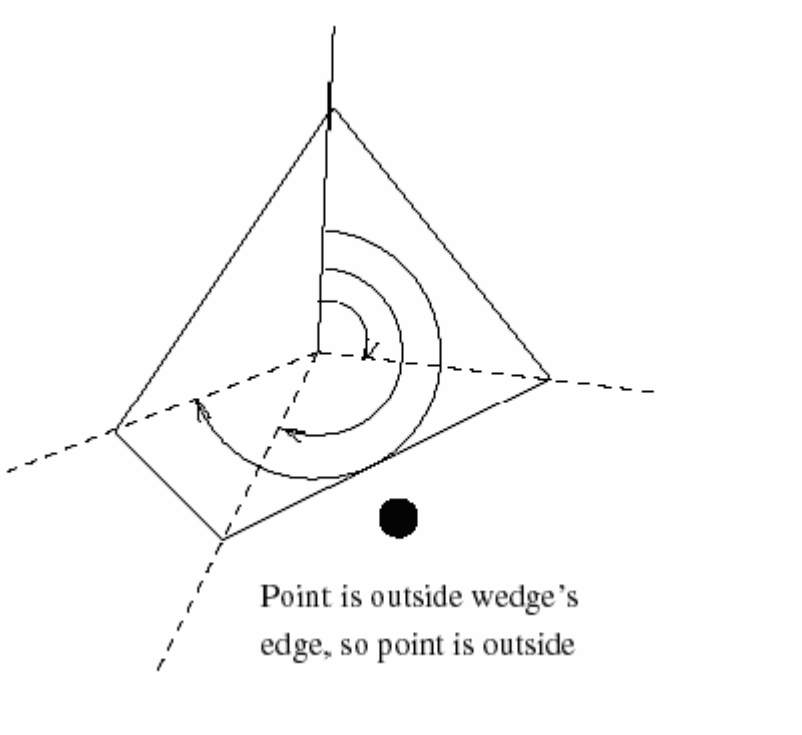


Figure 5.4 Convex Inclusion Test

Edge Problems

A problem that is sometimes important is determining if a point is exactly on an edge of a polygon. Being "exactly" on an edge entails computing error bounds and other issues in numerical analysis. While the "on" condition is of interest in CAD and elsewhere, for most computer graphics related operations this type of classification is unnecessary.

Problems occur in triangle fan algorithms when the code assumes that a point that lies on a triangle edge is inside that triangle. Points on the edges between test triangles will be classified as being inside two triangles, and so will be

classified as being outside the polygon. This problem does not happen with the convex test. However, another problem is common to this family of algorithms. If a point is on the edge between two polygons, it will be classified as being inside both.

The code presented for these algorithms does not fully address either of these problems. In reality, a random point tested against a polygon using either algorithm has an infinitesimal chance of landing exactly on any edge. For rendering purposes this problem can be ignored, with the result being one mis-shaded pixel once in a great while.

The crossings test does not have these problems when the 2D polygons are in the same plane. By the nature of the test, all points are consistently categorized as being to one side of any given edge or vertex. This means that when a point is somewhere inside a mesh of polygons the point will always be in only one polygon. Points exactly on the unshared edge of a polygon will be classified as arbitrarily inside or outside the polygon by this method, however. Again, this problem is rarely encountered in rendering and so can usually be ignored.

Faster Tests

The algorithms presented so far for the general problem have been O(n); the order of the problem is related to the number of edges. Preparata and Shamos present a fascinating array of solutions which are theoretically faster. However, these algorithms have various limitations and tend to bog down when actually coded due to expensive operations.

In this section are some practical methods inspired by their presentation that perform well under ordinary circumstances. They are still O(n) for pathological cases, but for reasonable polygon data they are quite efficient. These methods use much more memory and have higher initialization times before testing begins, but are much faster per tested point.

Bins Method

One method to speed up testing is to classify the edges by their Y components and then test only those edges with a chance of intersecting the test point's X+ test ray. The bounding box surrounding the polygon is split into a number of horizontal bins and the parts of the edges in a bin are kept in a list, sorted by the minimum X component.

The maximum X of the edge is also saved, along with a flag noting whether the edge fully crosses the bin's Y bounds. In addition, the minimum and maximum X components of all the edges in each bin are recorded.

When a point is to be classified, the proper bin is retrieved and if the point is outside the X bounds, it must be outside the polygon. Else, the list is traversed and the edges tested against the point. Essentially, a modified crossings test is done, with additional speed coming from the sorted order and from the storage of the "fully crosses" condition (Figure 5.5).

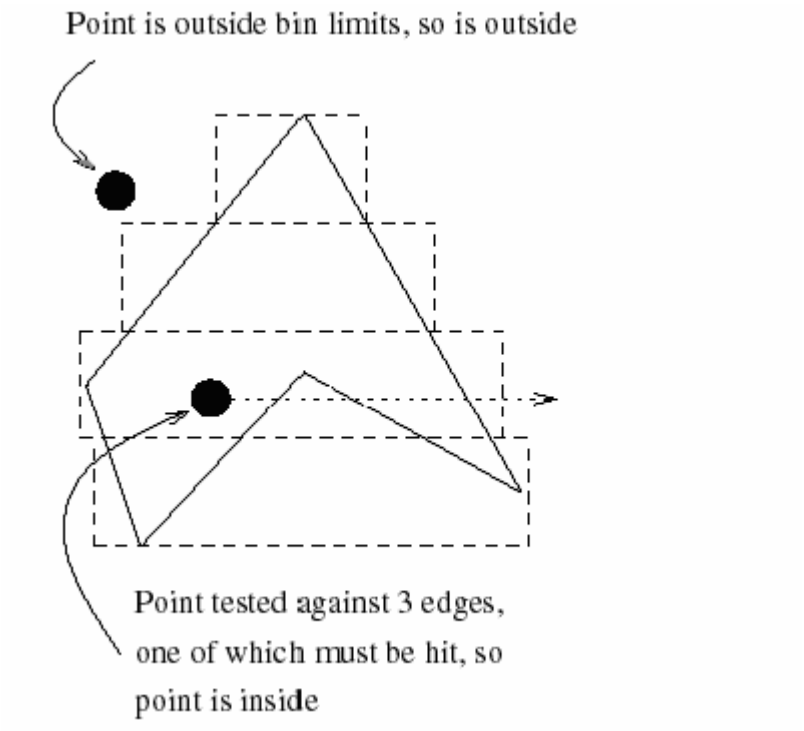


Figure 5.5 Bin Test

Grid Method

An even faster, and more memory intensive, method of testing for points inside a polygon is using lookup grids. The idea is to impose a grid inside the bounding box containing the polygon. Each grid cell is categorized as being fully inside, fully outside, or indeterminate. The indeterminate cells also have a list of edges which overlap the cell, and also one corner (or more) is determined to be inside or outside using a traditional test. It is quick to determine the state of these corners by dealing with those on each latitude line by flipping the state of each corner to the right of the edge's crossing point.

To test a point against this structure is extremely quick in most cases. For a reasonable polygon many of the cells are either inside or outside, so testing consists of a simple look-up. If the cell contains edges, then a line segment is formed from the test point to the cell corner and is tested against all edges in the list. Since the state of the corner is known, the state of the test point can be found from the number of intersections (Figure 5.6).

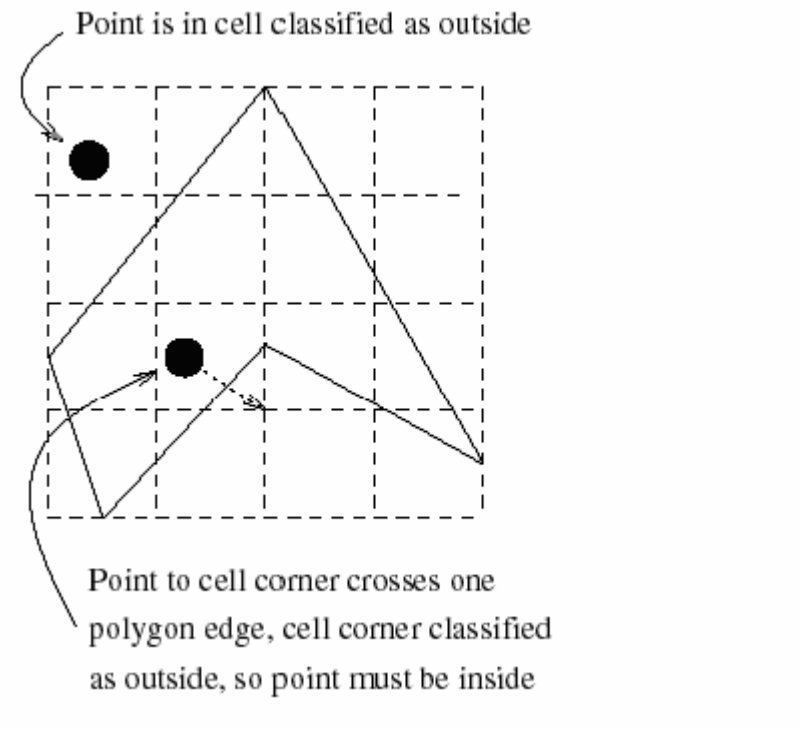


Figure 5.6 Grid Cell Test

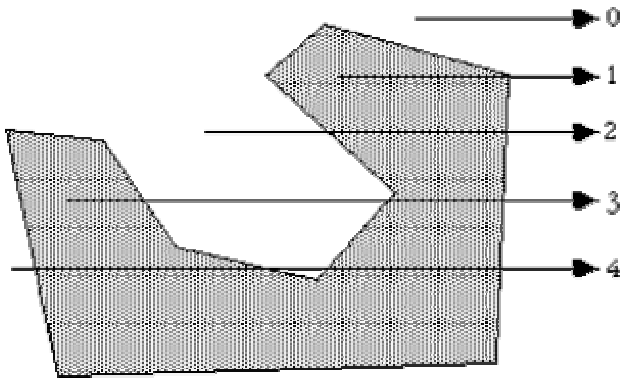
Care must be taken when a polygon edge exactly (or even nearly exactly) crosses a grid corner, as this corner is then unclassifiable. Rather than coping with the topological and numerical problems involved, one simple solution is to just start generating the grid from scratch again, giving slightly different dimensions to the bounding box. When testing the line segment against the edges in a list, exact intersections of an edge endpoint must be counted only once.

One additional speed up is possible. Each grid cell has four sides. If no edges cross a side, then that side will be fully inside or outside the polygon. A perfectly horizontal or vertical test line segment can then be generated and the faster crossings test can be used against the edges in the cell. The only test case where this made a significant difference was with a 20x20 grid imposed on a 1000 edge polygon, where the grid cell side test was 1.3 times faster.

Solution first (2D)

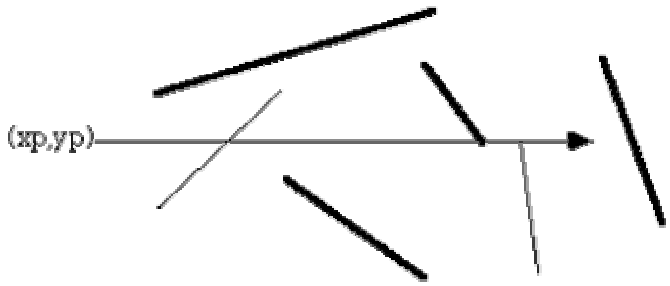
The following is a simple solution to the problem often encountered in computer graphics, determining whether or not a point (x,y) lies inside or outside a 2D polygonally bounded plane. This is necessary for example in applications such as polygon filling on raster devices. hatching in drafting software, and determining the intersection of multiple polygons.

Consider a polygon made up of N vertices (xi,yi) where i ranges from 0 to N-1. The last vertex (xN,yN) is assumed to be the same as the first vertex (x0,y0), that is, the polygon is closed. To determine the status of a point (xp,yp) consider a horizontal ray emanating from (xp,yp) and to the right. If the number of times this ray intersects the line segments making up the polygon is even then the point is outside the polygon. Whereas if the number of intersections is odd then the point (xp,yp) lies inside the polygon. The following shows the ray for some sample points and should make the technique clear.



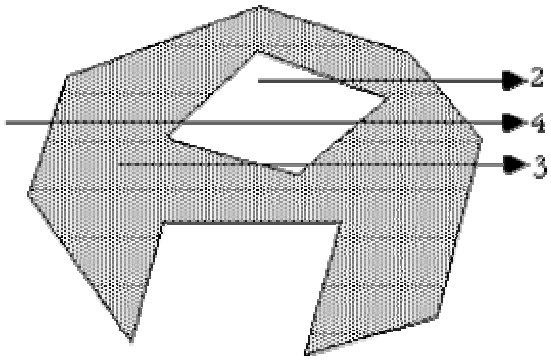
For the purposes of this discussion 0 will be considered even, the test for even or odd will be based on modulus 2, that is, if the number of intersections modulus 2 is 0 then the number is even, if it is 1 then it is odd.

The only trick is what happens in the special cases when an edge or vertex of the polygon lies on the ray from (xp,yp). The possible situations are illustrated below.



The thick lines above are not considered as valid intersections, the thin lines do count as intersections. Ignoring the case of an edge lying along the ray or an edge ending on the ray ensures that the endpoints are only counted once.

Note that this algorithm also works for polygons with holes as illustrated below



The following C function returns INSIDE or OUTSIDE indicating the status of a point P with respect to a polygon with N points.

```
#define MIN(x,y) (x < y ? x : y)
#define MAX(x,y) (x > y ? x : y)
#define INSIDE 0
#define OUTSIDE 1

typedef struct {
   double x,y;
} Point;

int InsidePolygon(Point *polygon,int N,Point p)
{
  int counter = 0;
  int i;
  double xinters;
  Point p1,p2;

  p1 = polygon[0];
  for (i=1;i<=N;i++) {
    p2 = polygon[i % N];
    if (p.y > MIN(p1.y,p2.y)) {
      if (p.y <= MAX(p1.y,p2.y)) {
        if (p.x <= MAX(p1.x,p2.x)) {
```

```
            if (p1.y != p2.y) {
              xinters = (p.y-p1.y)*(p2.x-p1.x)/(p2.y-p1.y)+p1.x;
              if (p1.x == p2.x || p.x <= xinters)
                counter++;
            }
           }
         }
        }
        p1 = p2;
      }

      if (counter % 2 == 0)
        return(OUTSIDE);
      else
        return(INSIDE);
}
```

Solution second (2D)

Another solution forwarded by Philippe Reverdy is to compute the sum of the angles made between the test point and each pair of points making up the polygon. If this sum is 2pi then the point is an interior point, if 0 then the point is an exterior point. This also works for polygons with holes given the polygon is defined with a path made up of coincident edges into and out of the hole as is common practice in many CAD packages.

The inside/outside test might then be defined in C as

```
typedef struct {
  int h,v;
} Point;

int InsidePolygon(Point *polygon,int n,Point p)
```

```
{
  int i;
  double angle=0;
  Point p1,p2;

  for (i=0;i<n;i++) {
    p1.h = polygon[i].h - p.h;
    p1.v = polygon[i].v - p.v;
    p2.h = polygon[(i+1)%n].h - p.h;
    p2.v = polygon[(i+1)%n].v - p.v;
    angle += Angle2D(p1.h,p1.v,p2.h,p2.v);
  }

  if (ABS(angle) < PI)
    return(FALSE);
  else
    return(TRUE);
}

/*
  Return the angle between two vectors on a plane
  The angle is from vector 1 to vector 2, positive anticlockwise
  The result is between -pi -> pi
*/
double Angle2D(double x1, double y1, double x2, double y2)
{
  double dtheta,theta1,theta2;

  theta1 = atan2(y1,x1);
  theta2 = atan2(y2,x2);
  dtheta = theta2 - theta1;
  while (dtheta > PI)
```

```
    dtheta -= TWOPI;
  while (dtheta < -PI)
    dtheta += TWOPI;

  return(dtheta);
}
```

Solution third (2D)

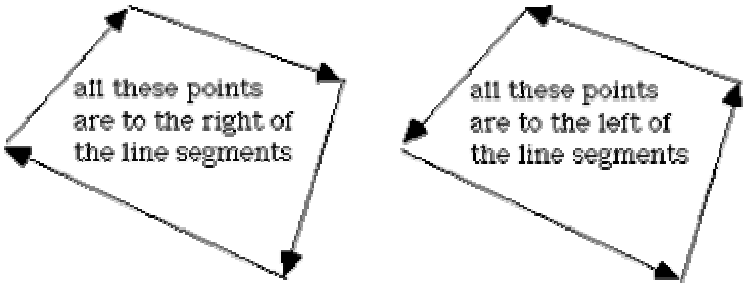
There are other solutions to this problem for polygons with special attributes. If the polygon is convex then one can consider the polygon as a "path" from the first vertex. A point is on the interior of this polygons if it is always on the same side of all the line segments making up the path.

Given a line segment between P0 (x0,y0) and P1 (x1,y1), another point P (x,y) has the following relationship to the line segment.

Compute

(y - y0) (x1 - x0) - (x - x0) (y1 - y0)

if it is less than 0 then P is to the right of the line segment, if greater than 0 it is to the left, if equal to 0 then it lies on the line segment.



points here are on the left of some line segments and to the right of others

Timings are in microseconds per test, and appear to be within roughly +- 10% accuracy. However, the best way to get true timings is to run the code on the

target machine; the code provided on disk has a test program which can be used to generate timings under various test conditions.

The performance of all the algorithms is practically linear; as such, the ratios of times for the 1000 edge polygons are representative of performance for polygons with a large number of edges.

Table 5.1 General Algorithms, Random Polygons

	Number of edges per polygon				
	3	**4**	**10**	**100**	**1000**
MacMartin	2.9	3.2	5.9	50.6	485
Crossings	3.1	3.4	6.8	60.0	624
Triangle Fan+edge sort	1.1	1.8	6.5	77.6	787
Triangle Fan	1.2	2.1	7.3	85.4	865
Barycentric	2.1	3.8	13.8	160.7	1665
Angle Summation	56.2	70.4	153.6	1403.8	14693
Grid (100x100)	1.5	1.5	1.6	2.1	9.8
Grid (20x20)	1.7	1.7	1.9	5.7	42.2
Bins (100)	1.8	1.9	2.7	15.1	117
Bins (20)	2.1	2.2	3.7	26.3	278

Table 5.2 General Algorithms, Regular Polygons

	Number of edges per polygon				
	3	**4**	**10**	**100**	**1000**
MacMartin	2.7	2.8	4.0	23.7	225
Crossings	2.8	3.1	5.3	42.3	444
Triangle Fan+edge sort	1.3	1.9	5.2	53.1	546
Triangle Fan	1.3	2.2	7.5	86.7	894
Barycentric	2.1	3.9	13.0	143.5	1482
Angle Summation	52.9	68.1	158.8	1489.3	15762
Grid (100x100)	1.5	1.5	1.5	1.5	1.5
Grid (20x20)	1.6	1.6	1.6	1.7	2.5
Bins (100)	2.1	2.2	2.6	4.6	3.8
Bins (20)	2.4	2.5	3.4	9.3	55.0

Conclusion of Point in Polygon strategies

So from the results we can say that Mac Martin test is most efficient strategy and so it should be used for the point in polygon strategy.

5.1.2 Line-Line Intersection

SIMPLEST CASE

Procedure

- Find the equations of the two lines, and solve them simultaneously for the intersection
    - the equation of a line is:

      y = a + bx where b is the slope

- Given two points on the line at (x1,y1) and (x2,y2), the slope b can be determined by the expression:

  b = (y1 - y2) / (x1 - x2)

    - the value of a can then be found by solving the equation using either point

General form

- In general, the two lines

  y = a1 + b1x and y = a2 + b2x intersect at:

  xi = - (a1 - a2) / (b1 - b2) yi = a1 + b1xi

- An intersection point at xi lies between x1 and x2, i.e. on line 1, if:

  (x1 - xi) (xi - x2) >= 0

- Similarly the point lies on line 2 with endpoints (u1,v1) and (u2,v2) if:

  (u1 - xi) (xi - u2) >= 0

SPECIAL CASES

- Unfortunately this algorithm will get into trouble in certain special cases:

Vertical lines

- If line is vertical, then it will cause an error because of an attempt to divide by zero, as numerical processors cannot deal with infinity

Solution

- To deal with these special cases we must make the algorithm a little more complex:
- These special cases occur in many simple geometrical algorithms.
  - they can be avoided to some extent by using different approaches

COMPLEX LINES

- Consider two complex lines of n1 and n2 straight line segments respectively:
  - these can be processed for all intersections by looping the simple algorithm, testing every segment in one line against every segment in the other
  - the amount of work to be done is proportional to the product (n1 x n2)

5.1.3 Line-Polygon Intersection

Line-Polygon intersection operations should comprise the following cases

- totally plotted
- partially plotted
- not plotted at all

Even though neither of two vertices is within the window, certain part of the line segment may be still within.

There are many different techniques for clipping lines in 2D
The fundamentals are

(1) Line equations and
(2) Intersection computation

Could apply point test to all points on line
– Too much work
–Need a simple test involving line's endpoint coordinates

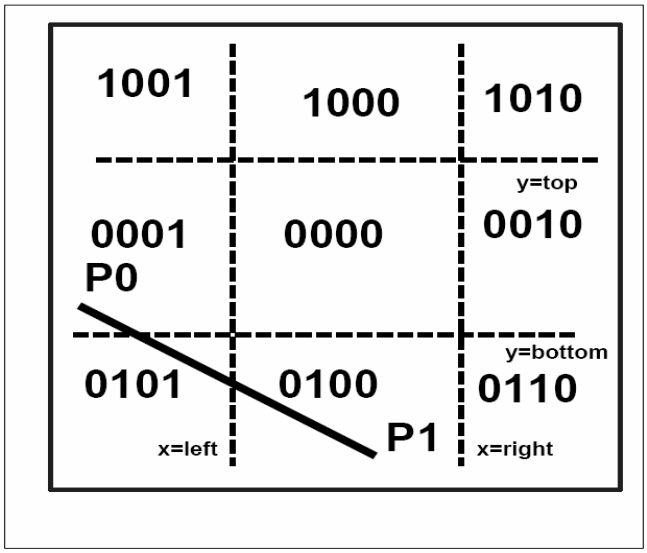
Next, we will discuss Cohen-Sutherland algorithm

5.1.3.1 Cohen-Sutherland Algorithm

It is not the most efficient algorithm
It is one the most commonly used
The key technique is 4-bit code:

TBRL where
T is set (to 1) if y > top
B is set (to 1) if y < bottom
R is set (to 1) if x > right
L is set (to 1) if x < left

Window Regions

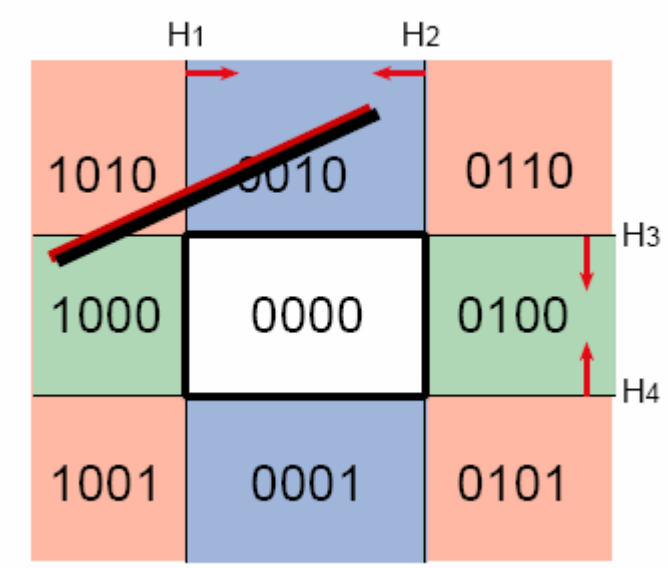


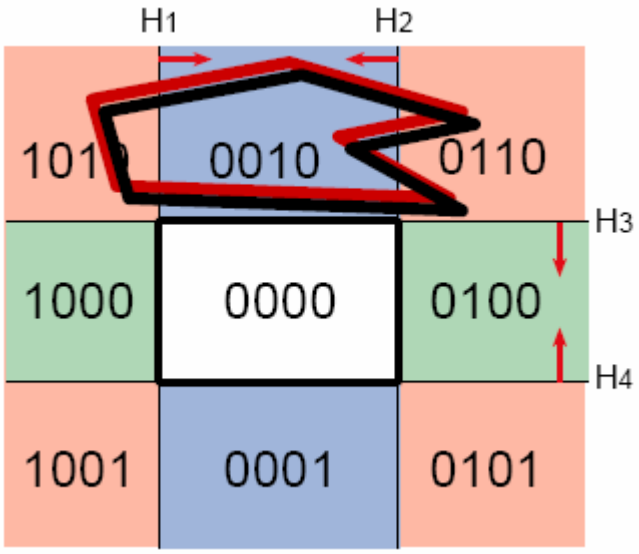
Outcode of p : 1000

Outcode of q : 0010

Outcode of [pq] : 0000

Not rejected





Outcode of p : 1010

Outcode of q : 1010

Outcode of r : 0110

Outcode of s : 0010

Outcode of t : 0110

Outcode of u : 0010

Outcode : 0010

Rejected

Algorithm

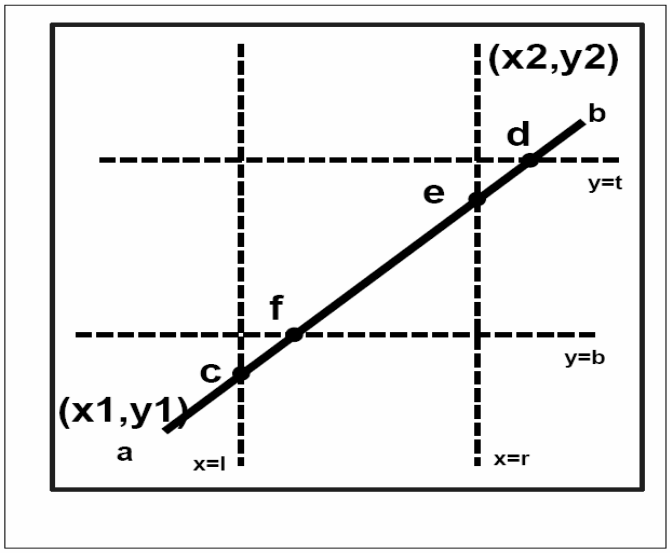
Assume two endpoints are p0 and p1

If code(p0) OR code(p1) is 0000,the line can be trivially accepted, the line is drawn

If code(p0) AND code(p1) is NOT 0000,the line can be trivially rejected, the line is not drawn at all

Otherwise, compute the intersection points of the line segment and window boundary lines (make sure to check all the boundary lines)

Line Intersection



Intersection Computation

Line equation

$y = y1 + m(x - x1)$,where $m = (y2 - y1)/(x2 - x1)$

Line intersection with the left vertical boundary

Line intersection with the left vertical boundary $x = l$

Assume the intersection is c

$x = l$

$y = y1 + m(l - x1)$

Line ab is clipped w.r.t. $x = l$, now it becomes cb

Line intersection with the top boundary $y = t$

Assume the intersection is d

y = t

x = 1/m(t-y1)+x1

Line cb is clipped w.r.t. y = t, line cb becomes cd

Line intersection with the right boundary x = r

Assume the intersection is e

x = r

y = y1 +m(r - x1)

Line cd is clipped w.r.t. x = r, line cd becomes ce

Line intersection with the bottom boundary y = b

Assume the intersection is f

y = b

x = 1/m(b-y1)+x1

Line ce is clipped w.r.t. y = b, line ce becomes fe

So, the entire process is

ab - cb - cd - ce –fe

Pseudo code

C-S-Clip(P0 = (x0,y0) , P1 = (x1,y1) , xmin , xmax , ymiz , ymax)

C0 ← Code(P0) ; C1 ← Code(P1);

if ((C0 and C1) ≠ 0) then
return;
if ((C0 or C1) == 0) then
draw(P0, P1);
else if (OutsideWindow(P0)) then

begin

Edge (Window boundary of leftmost non-zero bit of C0;

P2 ← line(P0,P1) ∩ Edge;

C-S-Clip(P1 , P2 , xmin , xmax , ymin , ymax);

End

Else

begin

Edge (Window boundary of leftmost non-zero bit of C1;

P2 ← line(P0,P1) ∩ Edge;

C-S-Clip(P0, P2 , xmin , xmax , ymin , ymax);

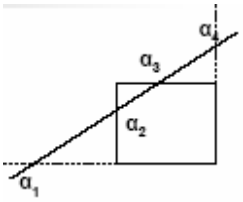
end

Liang-Barsky Clipping

$1 > \alpha 4 > \alpha 3 > \alpha 2 > \alpha 1 > 0$

Line meets l before t

Line between α2 and α3 is inside the window

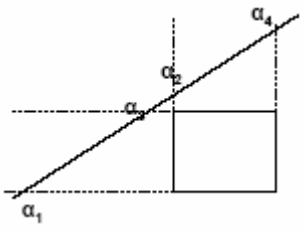


$1 > \alpha 4 > \alpha 2 > \alpha 3 > \alpha 1 > 0$

Line meets t before l

Whole line is outside

Reject completely



Efficiency Improvements

Compute intersections one by one

-May need less than four intersections to reject

Compare without floating point division

-If (α2 < α3) then (ymax-y1)(x2-x1) <(xmin-x1)(y2-y1)

Line-Segment Algorithm Assessment

- Cohen-Sutherland
  - Works well if many lines can be rejected early
  - Recursive structure (multiple subdiv) a drawback
- Liang-Barsky
  - Avoids recursive calls (multiple subdiv)
  - Many cases to consider (tedious, but not expensive)
  - Used more often in practice

5.1.3.2 Presented Line-Polygon Intersection Algorithm

In real problem polygons are not that much perfect rectangular type they can be regular or convex type so proposed algorithm is as described below :

Compare line bounding box with the polygon bounding box

If they intersect than

Further compare bounding box of line segment with that of polygon

If they intersect than

Take two end points of the line segment p1(x1,y1) and p2(x2,y2)

Check whether they are outside or inside the polygon using point in polygon strategy as described before

If both points are within the polygon line segment is totally inside the polygon so save both end points

If both points are outside the polygon than compute the intersection of line segment with polygon sides

If two intersections found save both intersection points

If none intersection found than line segment is outside the polygon so reject it

If p1 is outside and p2 is inside the polygon

If line is outside to inside save the intersection point and p2

If line is inside to outside save the intersection point

5.1.4 Polygon-Polygon Intersection

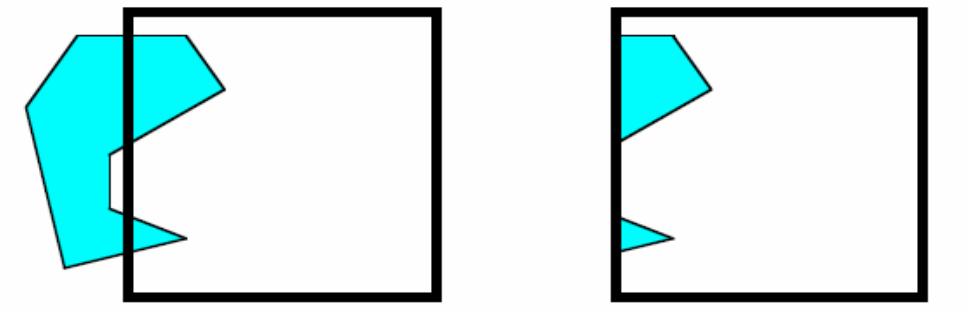
There are mainly two intersection algorithms

- Weiler algorithm
- Vatti algorithm

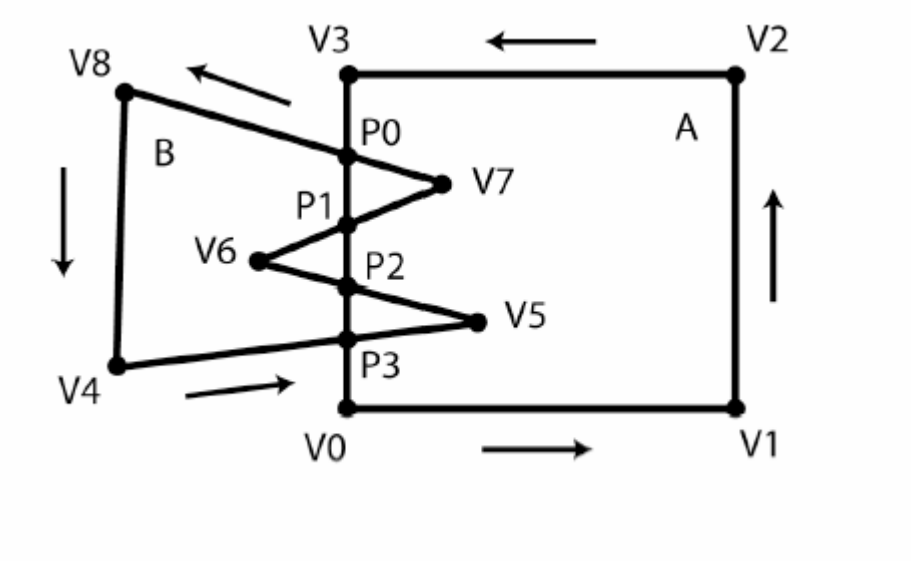
So I have explained here weiler algorithm in depth and described why the algorithm presented over here is more beneficial than these two algorithms.

Weiler-Atherton Algorithm

- General clipping algorithm for concave polygons with holes
- Produces multiple polygons (with holes)
- Make linked list data structure
- Traverse to make new polygon(s)



• Given polygons A and B as linked list of vertices (counter-clockwise order)
• Find all edge intersections & place in list
• Insert as "intersection" nodes
• Nodes point to A & B
• Determine in/out
status of vertices



Intersection

• Start at intersection point
– If connected to an "inside" vertex, go there
– Else step to an intersection point
– If neither, stop
• Traverse linked list
• At each intersection point switch to other polygon and remove intersection point from list
• Do until return to starting intersection point
• If intersection list not empty, pick another one
• All visited vertices and nodes define and'ed polygon

5.1.4.1 Why this presented Polygon-Polygon Intersection algorithm?

Weiler-Atherton Algorithm

The Weiler-Atherton algorithm is capable of clipping a concave polygon with interior holes to the boundaries of another concave polygon, also with interior holes.

But it is based on a graph representation of the polygons and was rather complicated.

In order to generate the resultant polygon, a complete traversal of a tree structure was necessary.

Here in present system to be developed we dont need to do intersection of polygons with the holes and also tree structure and graph makes things more complex and less efficient so we used method which uses only array data structure and simple algorithm rather than complex.

Vatti algorithm

Vatti proposed a new algorithm. His method was able to perform some other boolean operations on the two input polygons and offered support to an eventual further filling process.

However, the data structure they use was more complex that in the algorithm presented. Moreover, the intersection between the cross intersecting edges of a polygon has also to be computed. This leads to poorer results in comparison with the present algorithm.

So in this case also proposed algorithm is more beneficial because of its simple datastructure and algorithm.

5.1.4.2 Presented Polygon-Polygon Intersection Algorithm

Subproblem:

- **Input** : polygon (vertex list) of InputPolygon and OverlayPolygon
- **Output** : new (clipped/intersected) polygon (vertex list)

To get the intersection of two polygons we need to clip the InputPolygon using the boundary of the OverlayPolygon

To clip vertex list (polygon) against other polygon:

- Test first vertex. Output if inside, otherwise skip.
- Then loop through list, testing transitions

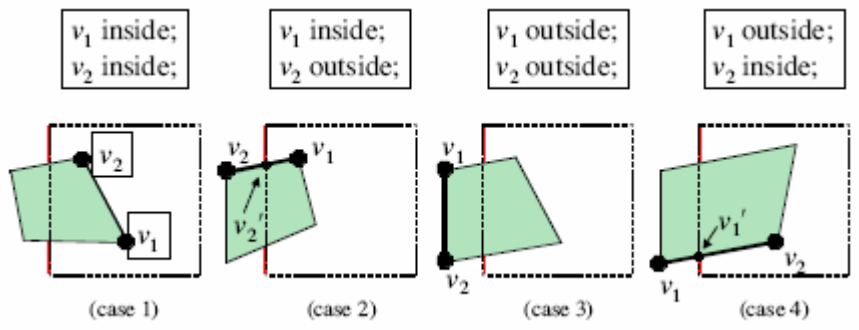
• In-to-in : output vertex
• In-to-out : output intersection
• Out-to-in : output intersection and vertex
• Out-to-out : no output

- Will output clipped polygon as vertex list

Can form a pipeline

Process vertex list concurrently

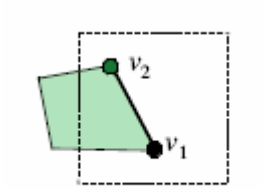
• For each clipper, there are 4 possible situations for the vertices at the endpoints of any polygon edge:



All of the clippers follow the same procedure:

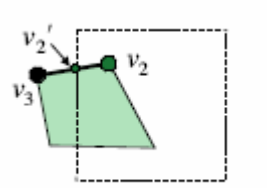
- In case 1: the line segment is not clipped; the second vertex v2 is forwarded to the next clipper

- In case 2: the intersection point, v2', of the line segment with the clipping edge is determined and forwarded to the next clipper
- In case 3: the line segment is entirely outside the clip region; no vertices are forwarded to the next clipper
- In case 4: the intersection point, v1', of the line segment with the clipping edge is determined and forwarded to the next clipper, along with the vertex v2



Edge 1:

- for all clippers, the vertices satisfy case 1;
- nothing is clipped;
- vertex v2'is output



Edge 2:

- for the bottom, right, and top edge clippers, the vertices satisfy case 1;
- for the clipper corresponding to the left edge, the vertices satisfy case 2;
- intersection point v2'□is output

5.1.5 Polygon-Polygon Union

5.1.5.1 Difference between Union and Intersection

- Union is very similar to Intersection, but in Union the parts of the polygons that do not intersect have zero for the non-defined attributes.
- Intersection only gives the part of the input feature which is within the overlay feature whereas in union individual parts and common parts are included in the output file.
- So as we know algorithms related to the union is based on the intersection algorithms for the vectors.
- So there are mainly two intersection algorithms based on which we can develop union algorithm

  - Weiler algorithm
  - Vatti algorithm

5.1.5.2 Why this presented Polygon-Polygon Union algorithm?

Clipping an arbitrary polygon against an arbitrary polygon is a basic routine in computer graphics. In rendering complex images, it may be applied thousands of times. The efficiency of these routines is therefore extremely important.

So far though, only two other algorithms have been proposed, Weiler designed for the first time an algorithms which was able to clip arbitrary polygons. His approach was based on a graph representation of the polygons and was rather complicated.

In order to generate the result polygon, a complete traversal of a tree structure was necessary. Vatti proposed a new algorithm. His method was able to perform some other Boolean operations on the two input polygons and offered support to an eventual further filling process.

However, the data structure they use was more complex that in the algorithm presented. Moreover, the intersection between the cross intersecting edges of a polygon has also to be computed. This leads to poorer results in comparison with the present algorithm.

Obviously if we want to do union of two polygons we need to modify intersection algorithm to do it. Also Weiler-Atherton algorithm is more complex and uses tree structure and graphs so I have modified Weiler-Atherton algorithm to do the union of the two polygons and used simple data structure than graphs and tree.

5.1.5.3 Data structures

The algorithm computes the unioned polygon in three phases. In the first phase it determines (and marks) the intersection points. The points are then inserted in both lists, in their proper place by ordering them using the alpha values.

If no intersection points are found in this phase we know that either the subject polygon lies entirely inside the clip polygon or vice versa, or that both polygons are disjoint. By performing a containment test for any of the vertexes we are able to determine which case we have. Then we either return the inner polygon or nothing at all.

In order to efficiently implement this, the algorithm uses two linked lists to represent the polygons. One list is for the OverlayPolygon and one for the InputPolygon (SubjectPolygon). In each list, a node represents a distinct vertex. Obviously, the lists will contain the x and y coordinates of the vertexes as well as the necessary index value to the previous and next node in the list. In addition to these however, the algorithm needs some more information:

```
int x, y;
int next;
int prev;
int intersect;          /* 1 if an intersection point, 0 otherwise */
```

```
int entry;          /* 1 if an entry point, 0 otherwise */
int visited;        /* 1 if the node has been visited, 0 otherwise */
float alpha;        /* intersection point placemet */
```

Intersect is a boolean value that is set to true if the current node is an intersection point and to false otherwise. Similarly, entry is flag that records whether the intersecting point is an entry or an exit point to the other polygon's interior.

The visited flag is used to mark the nodes already inserted in the result (it is important to notice here that every intersection point belongs to the resulting polygon (since by definition it belongs to both interiors).

The alpha value is a floating point number from 0 to 1 that indicates the position of an intersection point reported to two consecutive non-intersection vertexes from the initial list of vertexes.

5.1.5.4 Presented Polygon-Polygon Union Algorithm

Phase one

The first phase is basically an iteration through the both lists (the complexity is at least m*n, where m and n are the number of vertexes for the two polygons). The intersection points between all the edges are determined.

Intersection can be found by using the following equations:

Intersecting Two Edges

- Edge 0: (P0, P1)
- Edge 2: (P2, P3)
- E0 = P0 + t0 * (P1-P0) D0 ≡□ (P1-P0)
- E2 = P2 + t2 * (P3-P2) D2 ≡ □(P3-P2)
- P0 + t0 *D0= P2 + t2 * D2
- x0 + dx0 * t0= x2 +dx2 * t2
- y0 +dy0 * t0 = y2 +dy2 * t2

Solve for t's

- t0 = ((x0 – x2) * dy2+ (y2 - y0) * dx2) / (dy0 * dx2- dx0 * dy2)
- t2 = ((x2 - x0) * dy0+ (y0 - y2) * dx0) / (dy2 * dx0- dx2 * dy0)
- Edges intersect if $0 \leq t0, t2 \leq 1$
- Edges are parallel if denominator = 0

An important thing to remark is that here, unlike in the Vatti's algorithm, only the intersection points between edges from different polygons are considered.

Once a new intersection point is determined, it is inserted in both the two lists. While computing an intersection point, the algorithm determines also its alpha value.

The alpha value shows basically the placement of the intersection point between the two polygon vertexes. If the length of the new formed segment is a and the length of the edge it belongs to is b, then the alpha value is defined as a/b (the value is obviously between 0 and 1). Alpha is used when inserting the new intersection points. They basically define an ordering relation on the number of intersection points between the two vertexes of the same edge.

Phase two

Phase two is a simple process of marking the intersection points as entries or exits. Practically, the intersection points are labeled as entry/exit alternatively when traversing the polygon lists, since we cannot have two entry/exit points in a row.

The process starts by first determining the inside/outside value for one vertex of each of the polygons. If, for example, it is outside the other polygon, then the next intersection point that come across will be marked as entry. What it is interesting here is that, by changing the order in which one starts marking the nodes, other boolean operations on the set of polygons (specifically union and difference) can also be performed.

Consequently, the only problem in this phase is finding a "containment test" that, given a point and a polygon, determines whether the point lies inside that polygon or not.

But I have already found a good method to test whether the point is within the interior of the polygon or not so it is not a problem anymore here.

Phase three

The goal of phase three is to generate the unioned polygon. The generation process is a back and forth navigation through the two lists, guided by the entry/exit values.

Generation of a single polygon starts by traversing through the InputPolygon in counter clock wise direction and adding each point of that list until first Intersection point comes.

If the point is an entry then switch the list and go forward in the new list till the next intersection point is reached (otherwise go backward). From this exit point switch the list. Further go forward or backward depending on whether the "neighbor" vertex is an entry or an exit.

This process iterates until the starting point is reached, when we complete the generation of the polygon. In case that there still are unvisited exit nodes (every node is marked as visited or not) then the result is composed of multiple rings and a new iteration begins.

The execution of the algorithm for a small example is illustrated below:

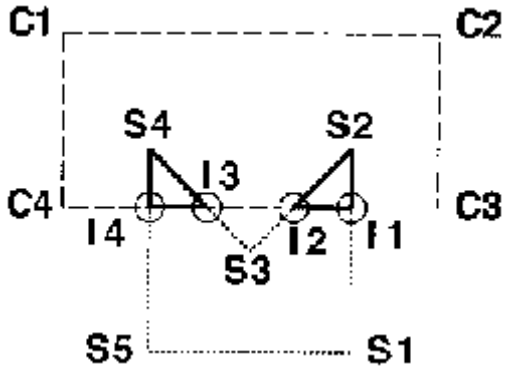


Figure 5.7 Small example

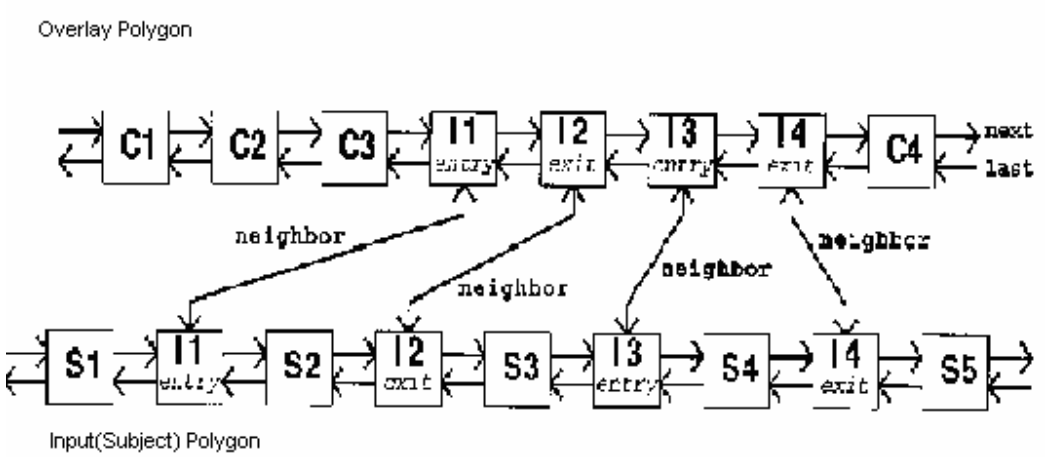


Figure 5.8 Two lists of Input and Overlay Polygon

5.1.5.5 Algorithm for the union with example

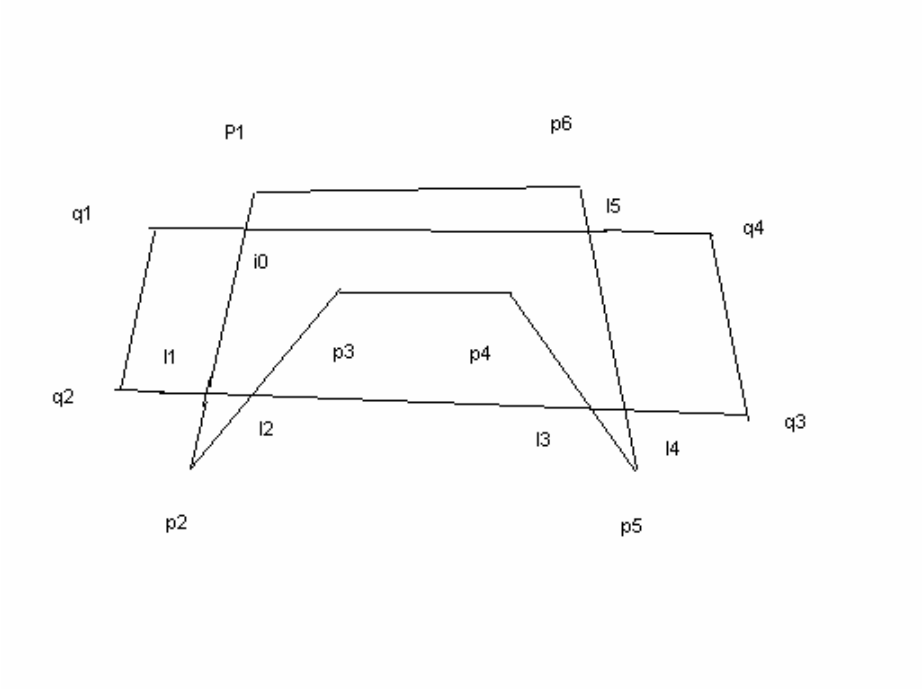


Figure 5.9 InputPolygon P and OverlayPolygon Q

1) We calculate intersections of sides and put them in two lists :

For polygon P -> P1 ; I0 ; I1 ; P2 ; I2 ; P3 ; P4 ; I3 ; P5 ; I4 ; I5 ; P6

For polygon Q -> Q1 ; Q2 ; I1 ; I2 ; I3 ; I4 ; Q3 ; Q4 ; I5 ; I0

2) We have also list of exiting points for P : I5 ; I3 ; I1

3) We begin with first point of polygon P , P1 and walk along the polygon in counter clockwise direction untill we get the intersection point means

We take next point on list for P until we get intersection point I0

As I0 is an entering point we change the list and walk along the polygon Q ,so we get the next point Q1,than Q2 and than I1

Here I1 is an exiting point so we go to list for P and take next point P2 than we take I2 but it is an entering point so we further switch to Q's list

So now next point is I3 which is exiting point so we go to list for P, so P4 is a next point and than I4 but it is entering for P so we go to list for Q and get the next point Q3,than Q4 and than I5 which is an exiting point so further we switch to P's list and get the points P6 and P1 so we got all the points to generate new resultant polygon which is a union of the two polygon P and Q.

4) We delete exiting points from exiting list

So output Polygon will be as given below

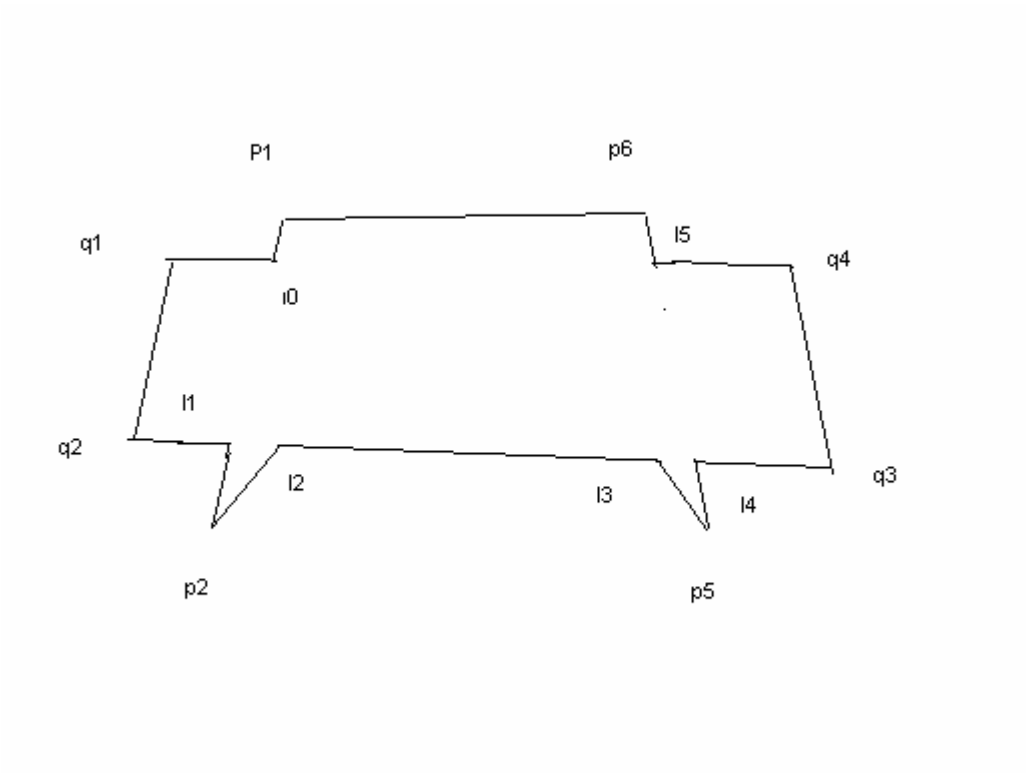


Figure 5.10 Unioned OutputPolygon

We can describe this algorithm in short as follows:

- Find a vertex of A outside of B
- Traverse linked list
- At each intersection point switch to other polygon
- Do until return to starting vertex
- All visited vertices and nodes define union'ed polygon

Special Case

If polygons don't intersect

- Union
  - If one inside the other, return polygon that surrounds the other
  - Else, return both polygons

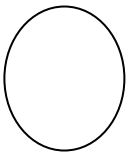
5.2 DATA FLOW DESIGN

Data flow design is concerned with designing a sequence of functional transformations that convert system inputs into the required outputs. The design is represented as data flow diagrams. These diagrams illustrate how data flows through a system and how output is derived from the input through a sequence of functional transformations.

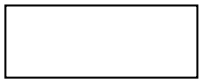
Data-flow diagrams are a useful and intuitive way of describing a system.They are normally understandable without special training, especially if control information is excluded. They show end to end processing.

Conventions used in drawing the data flow Diagrams are:-

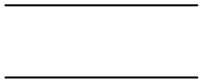
Data Flow



Processing
Function



Start or Stop point



Database

5.2.1 CONTEXT LEVEL DIAGRAM

Context level diagram shows the main task of the process.

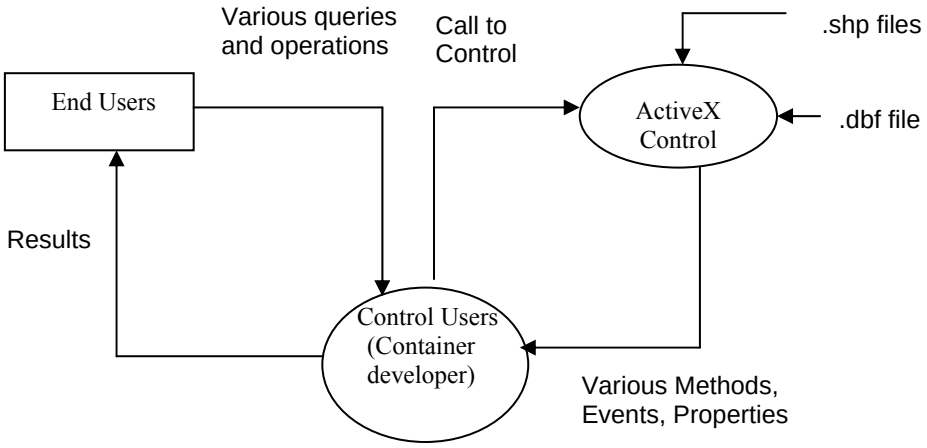


Figure 5.11 Context level diagram

End User: The user of the system.

ActiveX Control: This is developed in microsoft Visual C++ for internal operations and is not application dependent. This can be used for any type of software.

Container (User Interface): This is developed in Microsoft visual Basic and is visible to user. This is user friendly and easily understandable to both technical and non technical users.

5.2.2 First Level Data Flow Diagram

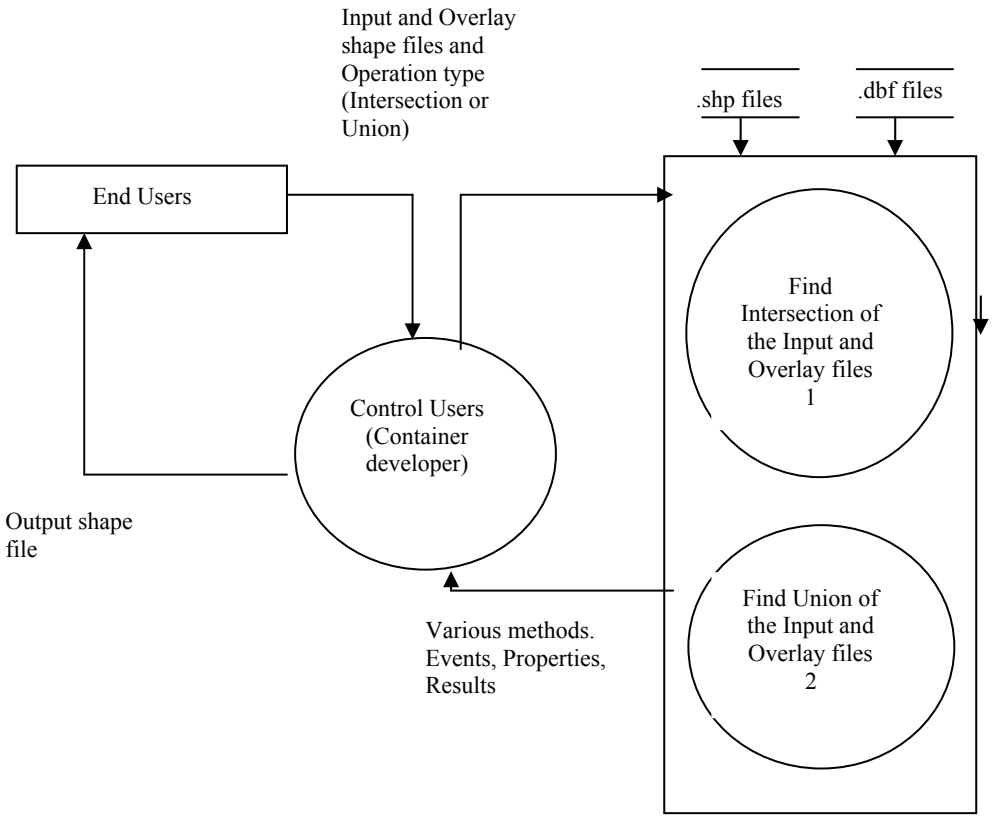


Figure 5.12 First level Data Flow Diagram

5.2.3 Second Level Data Flow Diagrams

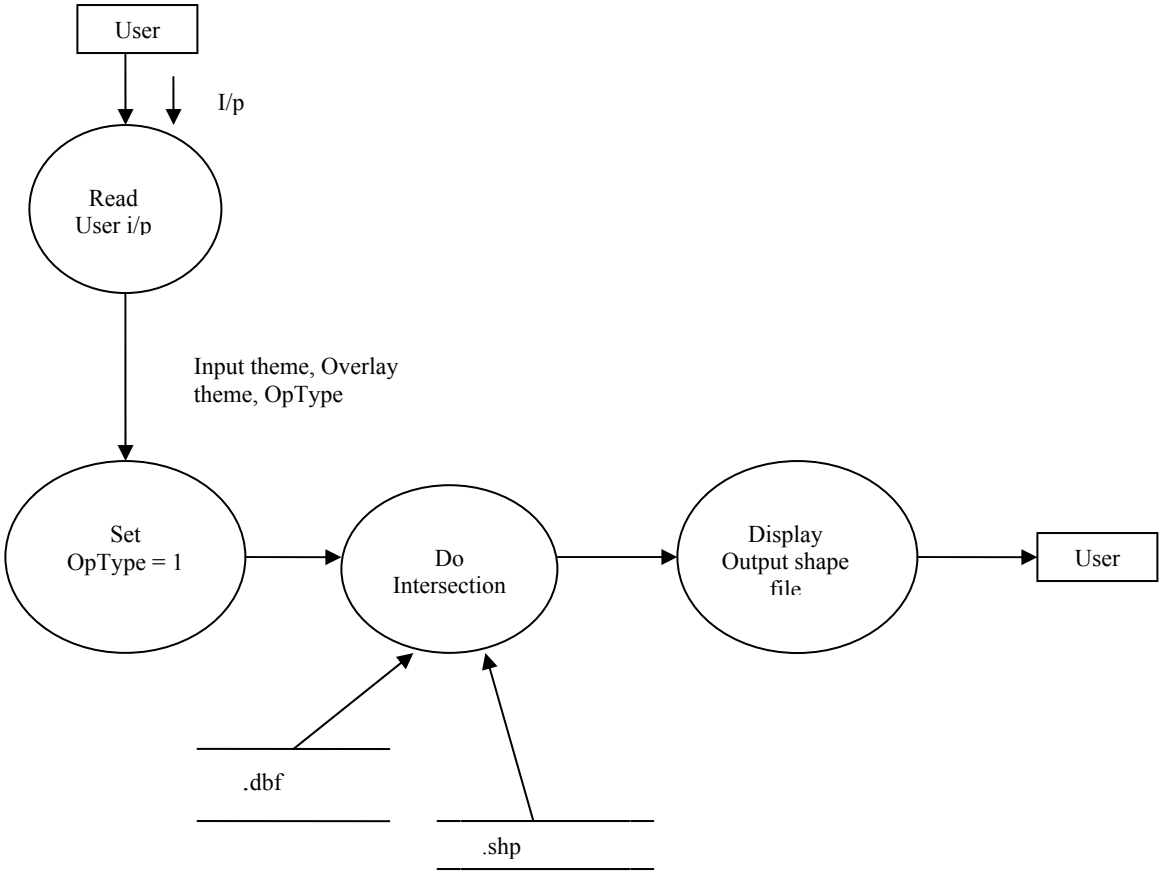


Figure 5.13 Second Level Data Flow Diagram for Intersection

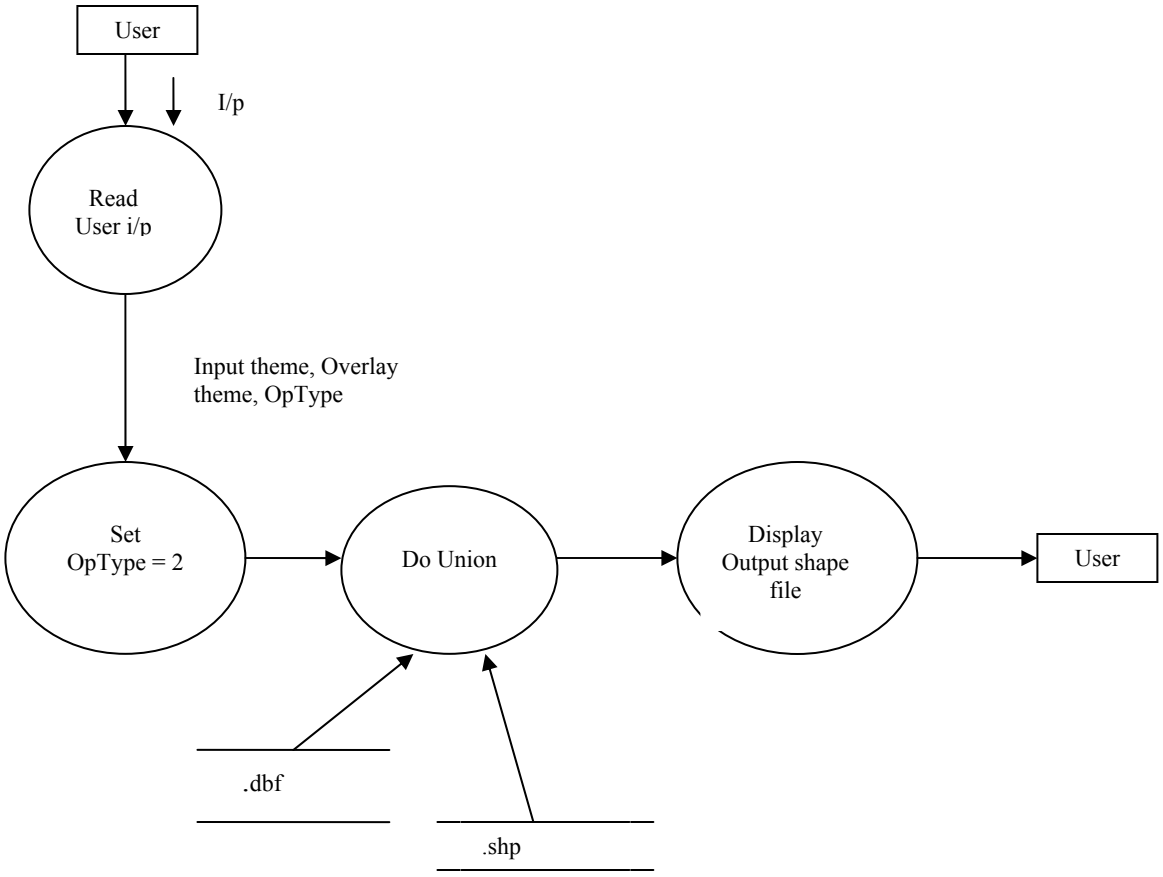


Figure 5.14 Second Level Data Flow Diagram for Union

Chapter 6

FUNCTIONAL SPECIFICATION

- Input Specifications and Input Snapshots
- Output Specifications and Output Snapshots
- Function and Performance

6.1 Input Specifications and Input Snapshots

As the Intersection and Union both requires two files as input one is input file and other is overlaying file so user first need to add them as a themes by browsing the files.

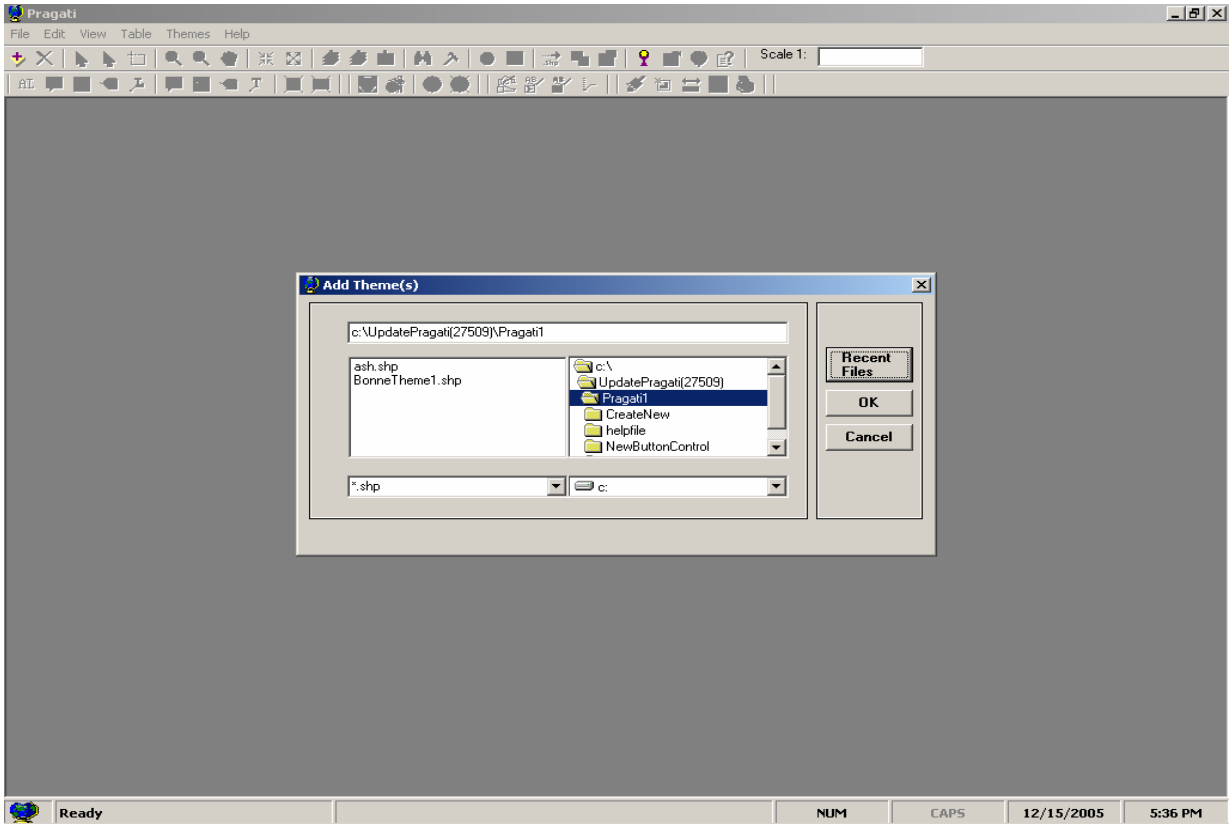


Figure 6.1 Add theme window

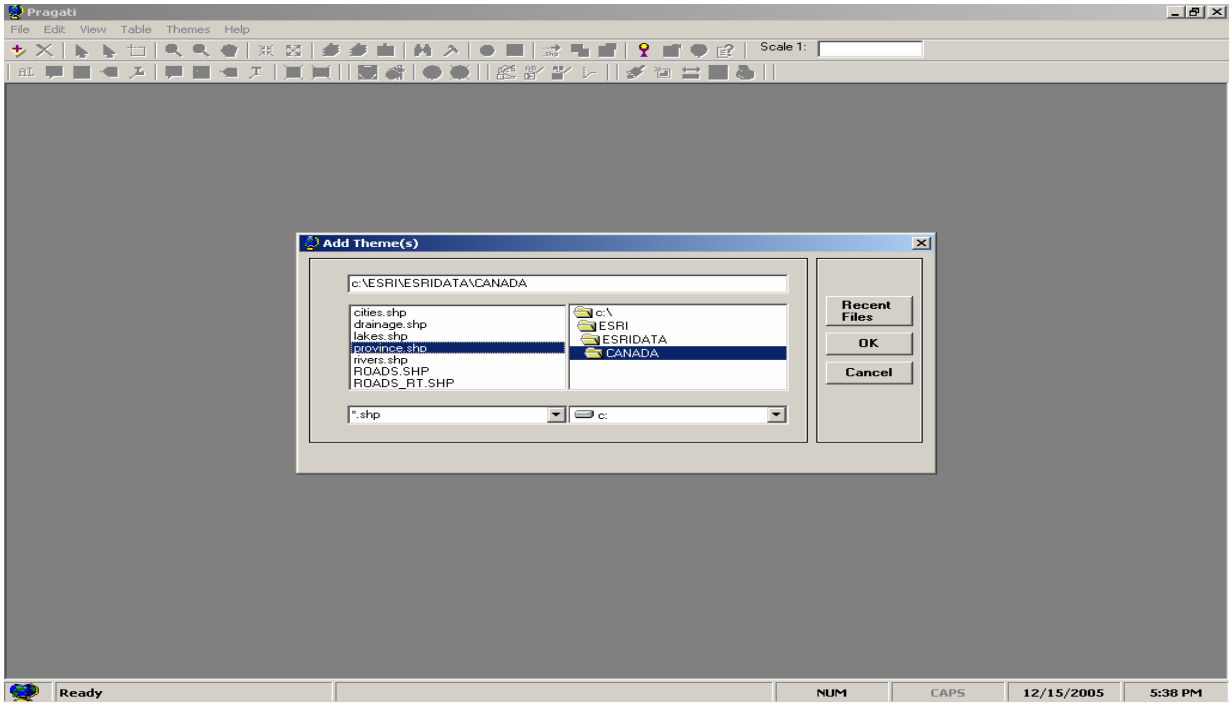


Figure 6.2 Adding of file province.shp

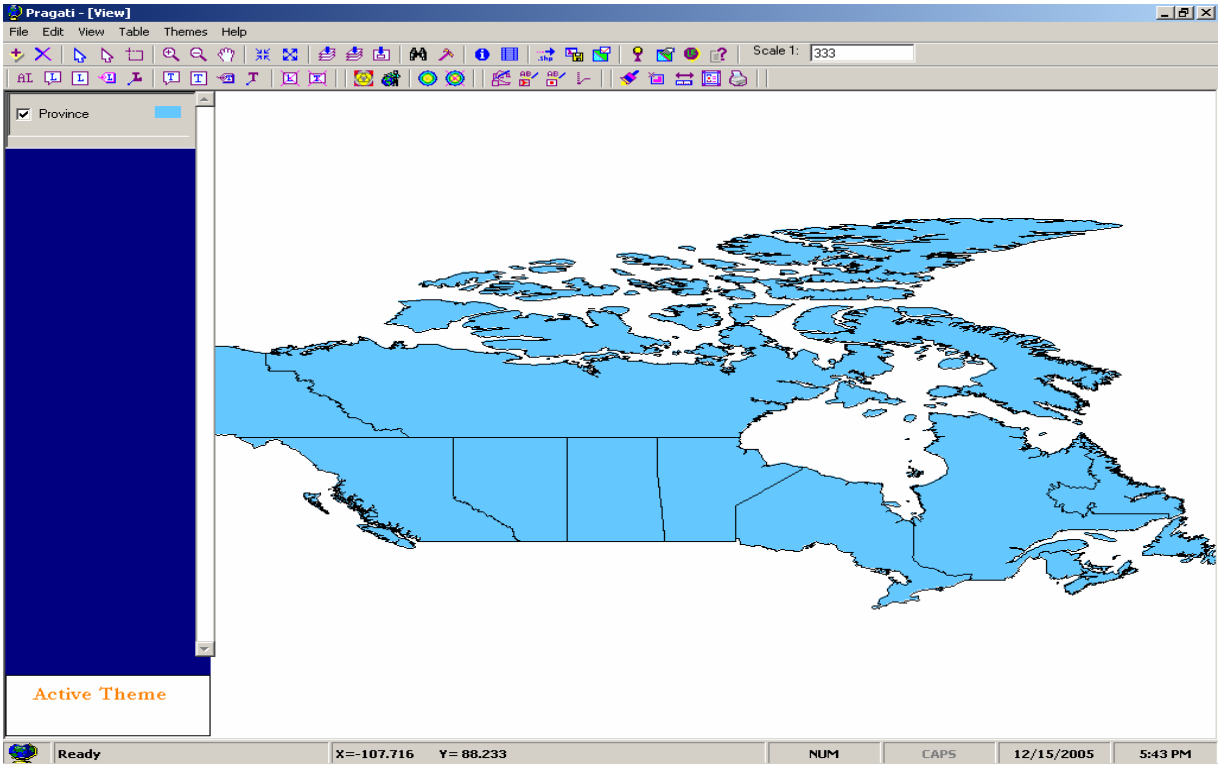


Figure 6.3 Display of file Province.shp

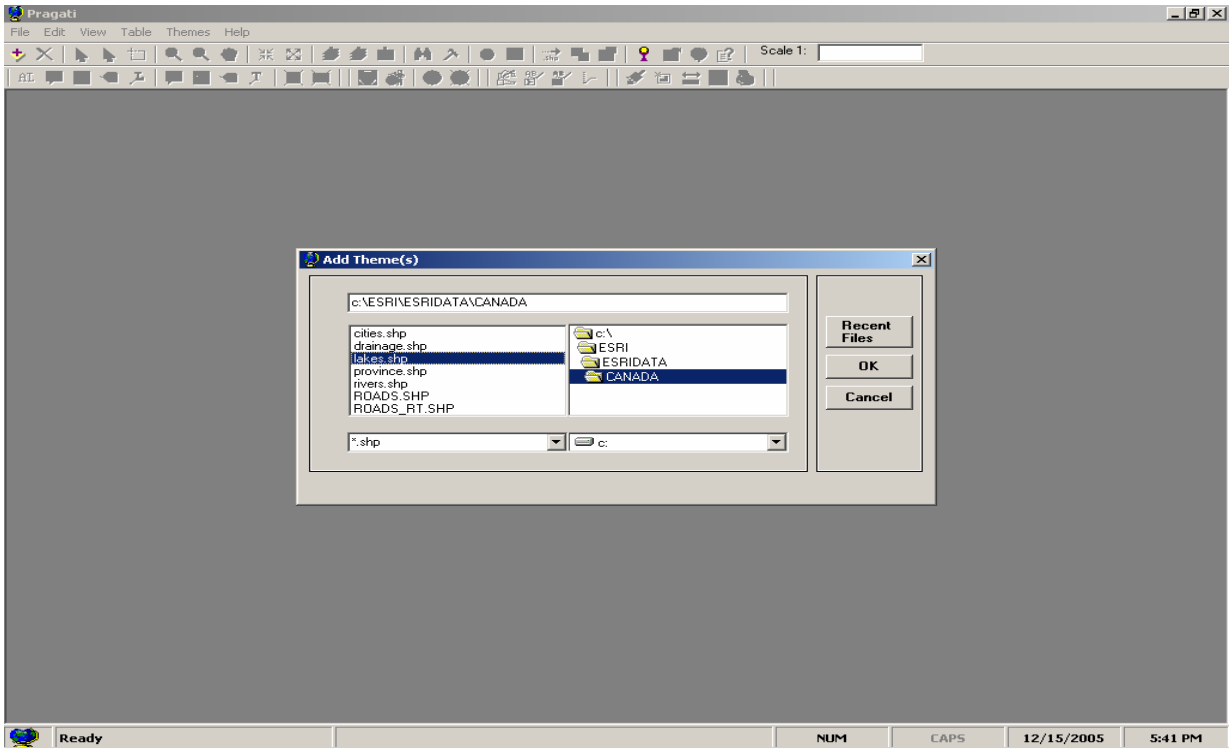


Figure 6.4 Adding of file lakes.shp

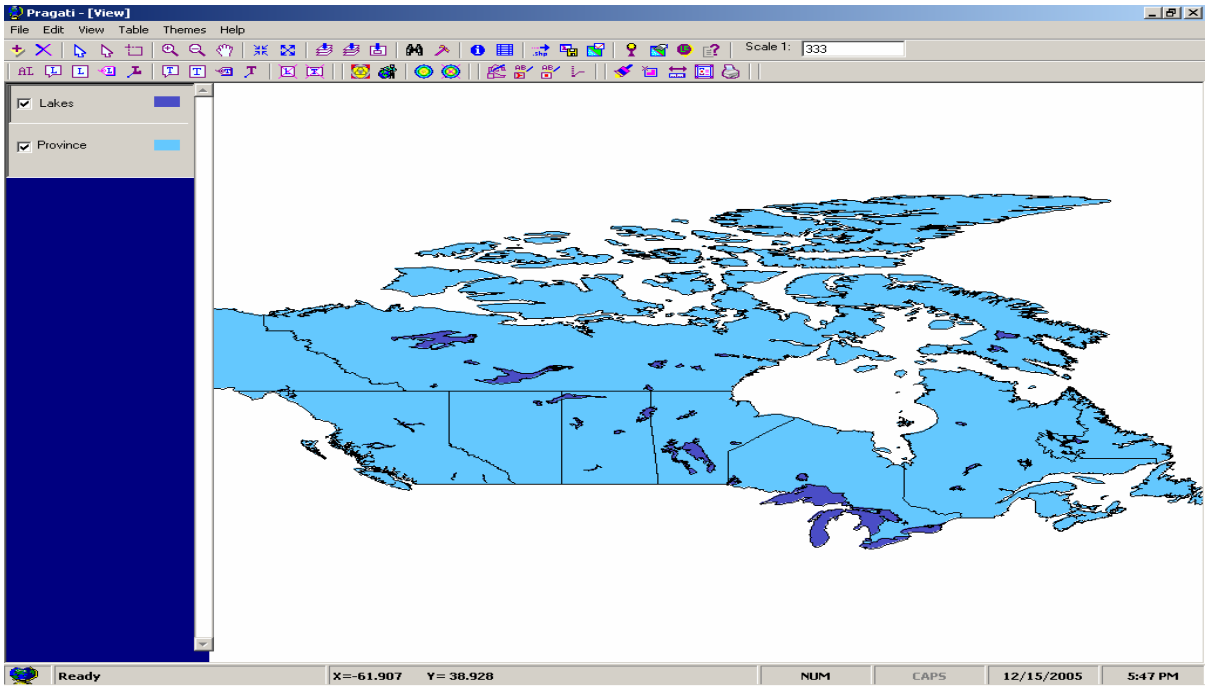


Figure 6.5 Display of the both files lakes.shp and province.shp

After that user can select "GeoProcessing Wizard" for selecting one of the two options for Intersection and Union from it.

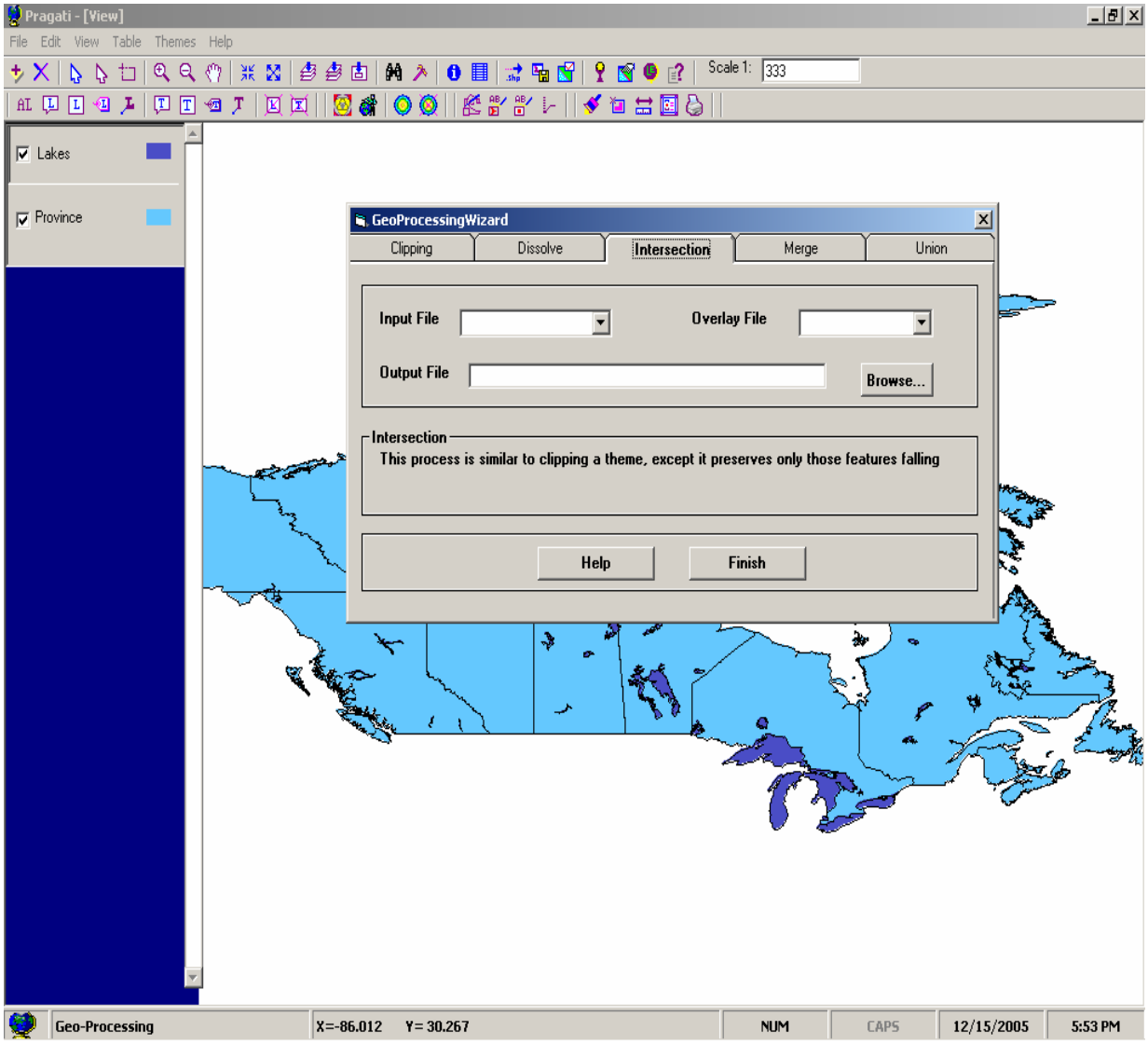


Figure 6.6 Geoprocessing Wizard with Intersection option

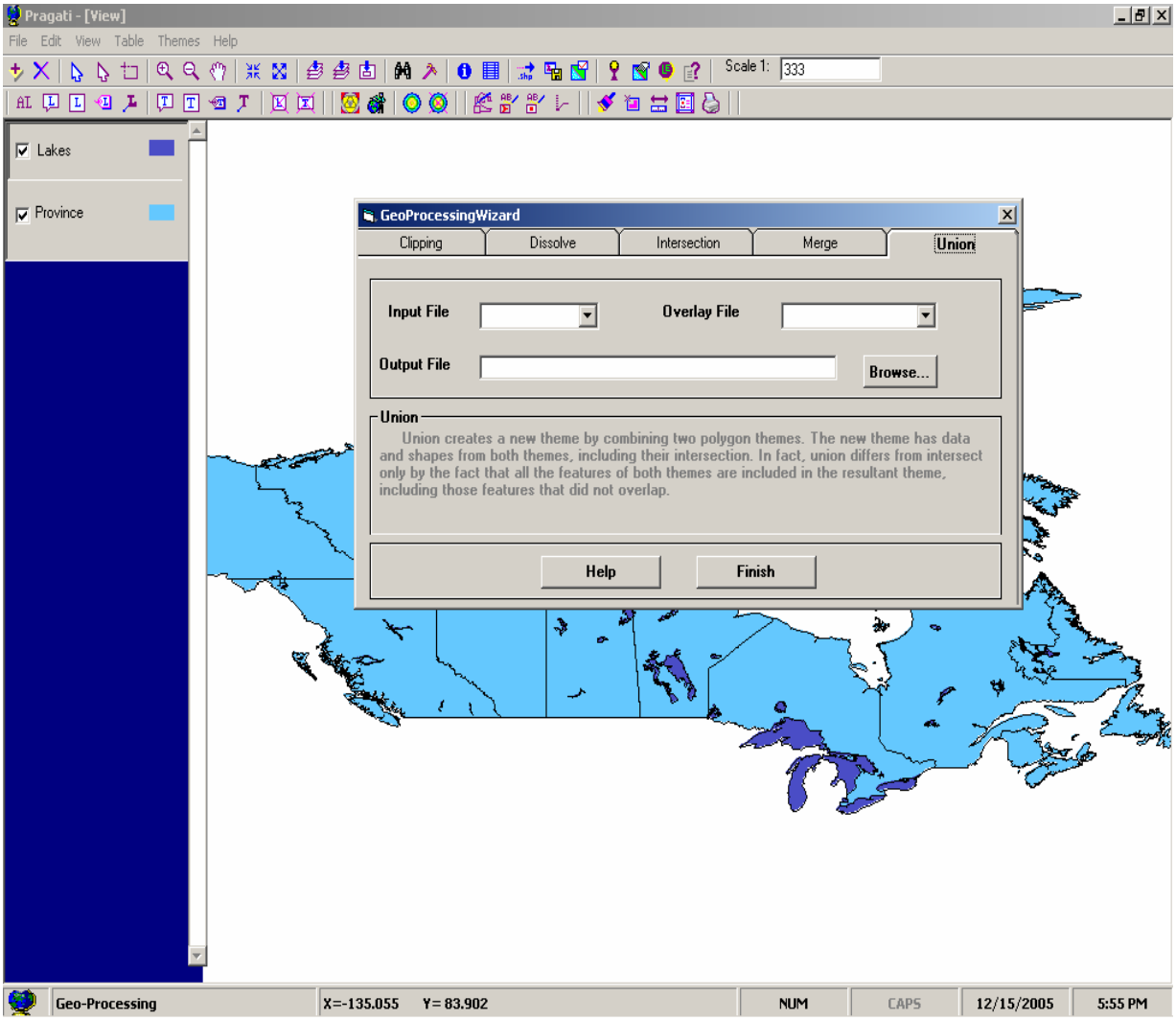


Figure 6.7 Geoprocessing Wizard with Union option

Here user requires to input two added files and result will be stored in third file which is Outputfile.

According to the values of these parameters means files, user can get the desired output either for Intersection or Union.

6.2 Output Specifications and Output Snapshots

As we are using VB to develop User Interface, the user is provided with an excellent Graphical User Interface (GUI) and project is very user friendly. So output or result will be displayed as an image to the user and it is stored in the shape file format. This graphical display of the output is strored in one of the three feature types point, line, and polygon in file with the format .shp.

As it has attributes related to each feature point, line, polygon stored in the database file, user can have the required attribute values of the resultant map in the database file stored with the .dbf format.

6.2.1 Line-Polygon Intersection Snapshots

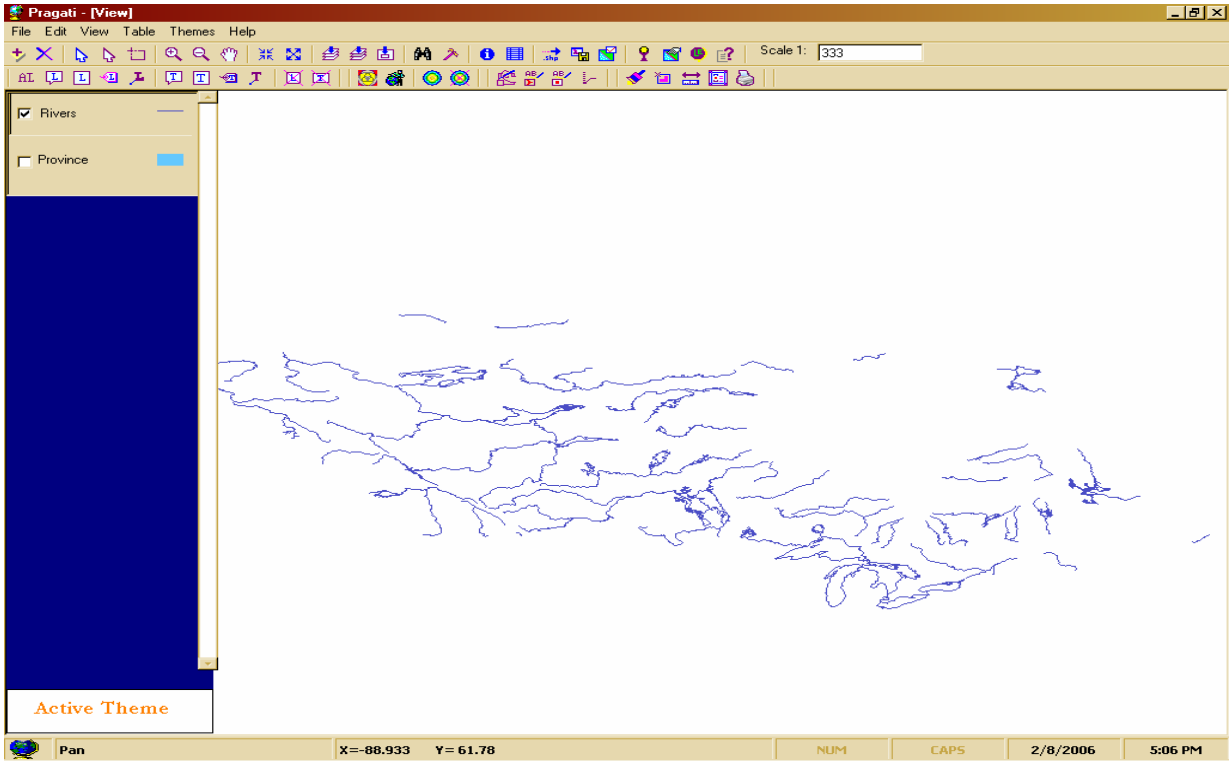


Figure 6.8 River map – line feature – input file

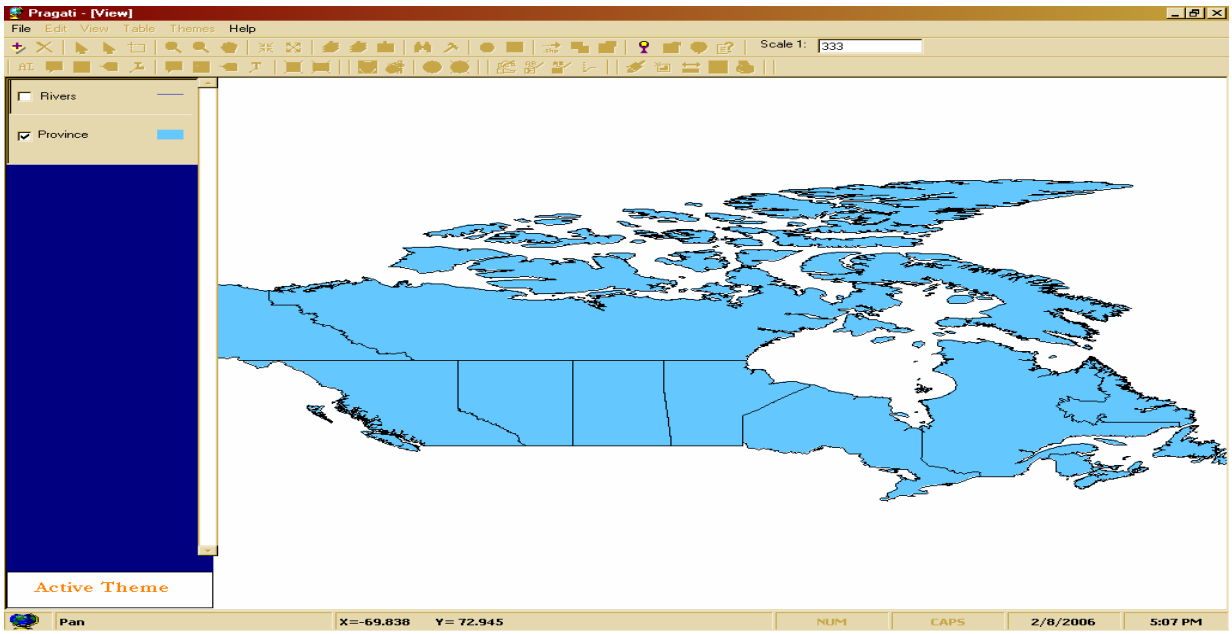


Figure 6.9 Province map – Polygon feature – Overlay file

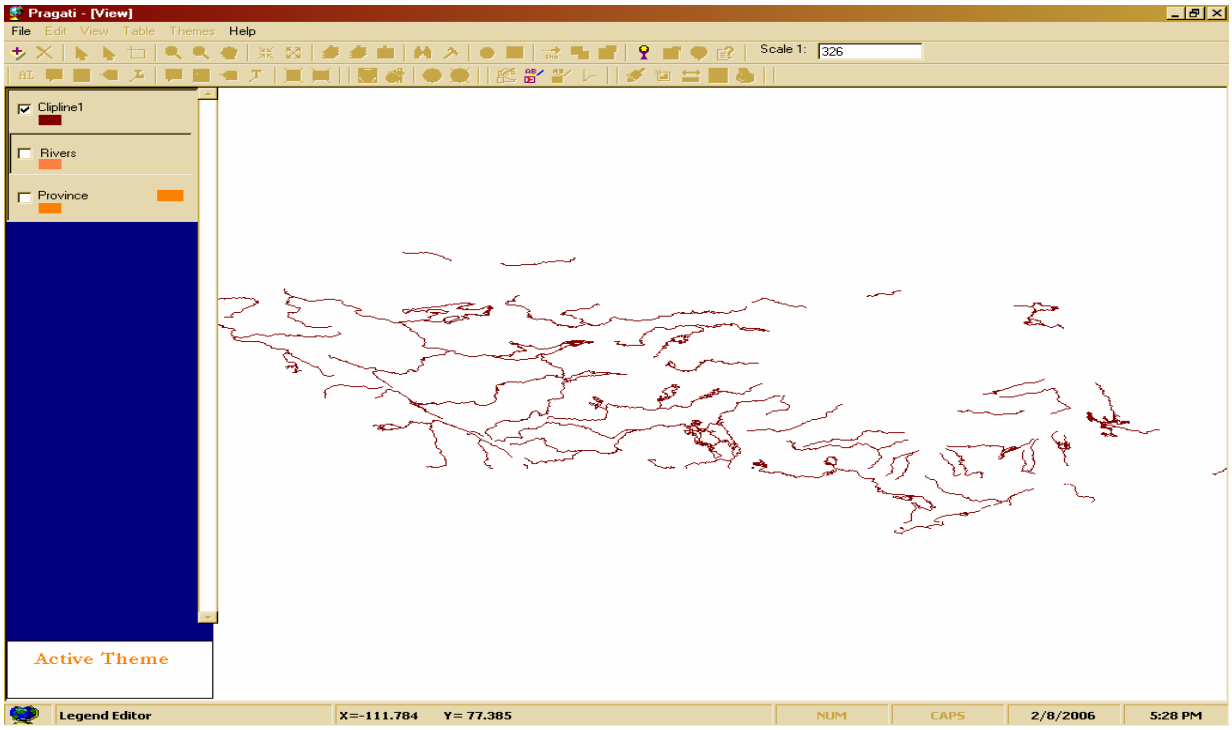


Figure 6.10 Intersected river map – line feature – output file

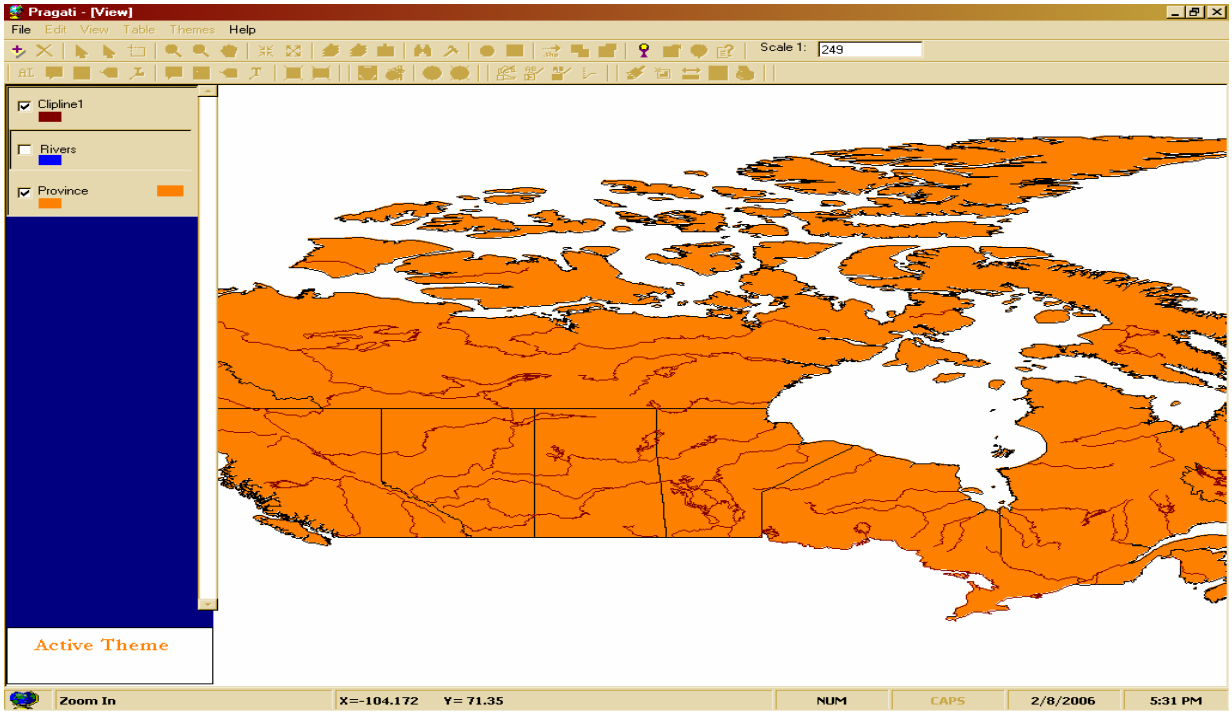


Figure 6.11 Province and intersected file together

6.2.2 Polygon-polygon Intersection Snapshots

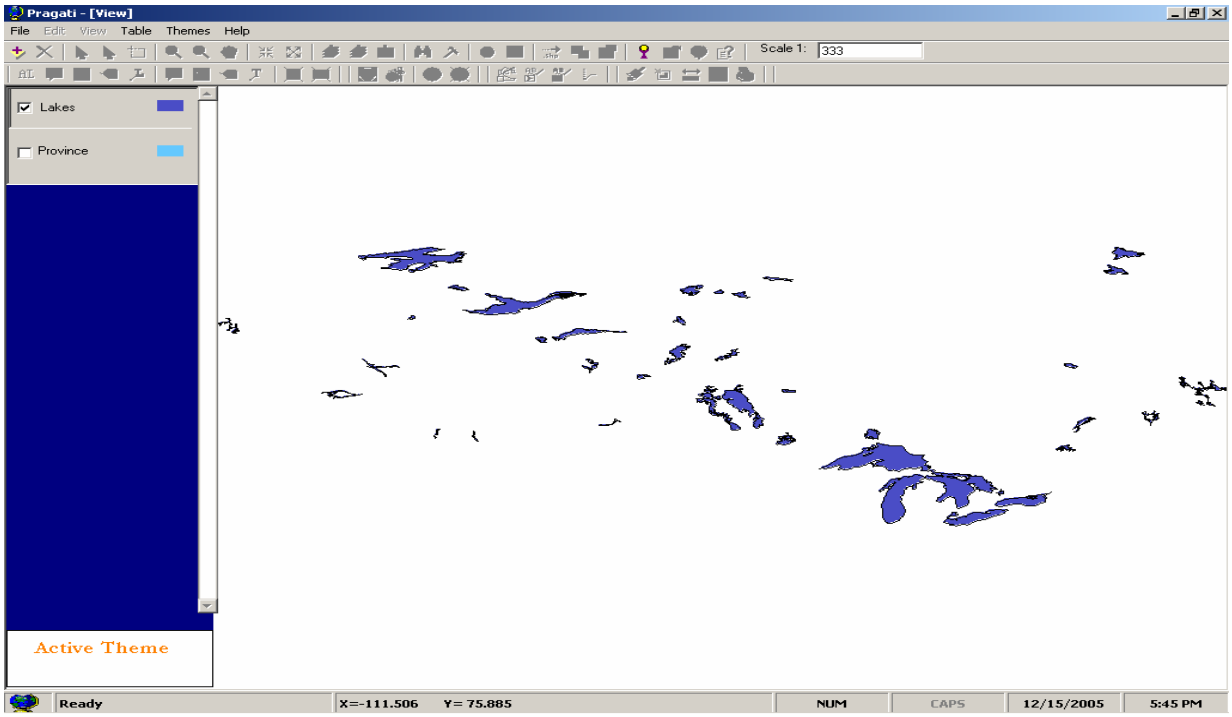


Figure 6.12 Lakes file – Polygon feature - Input file

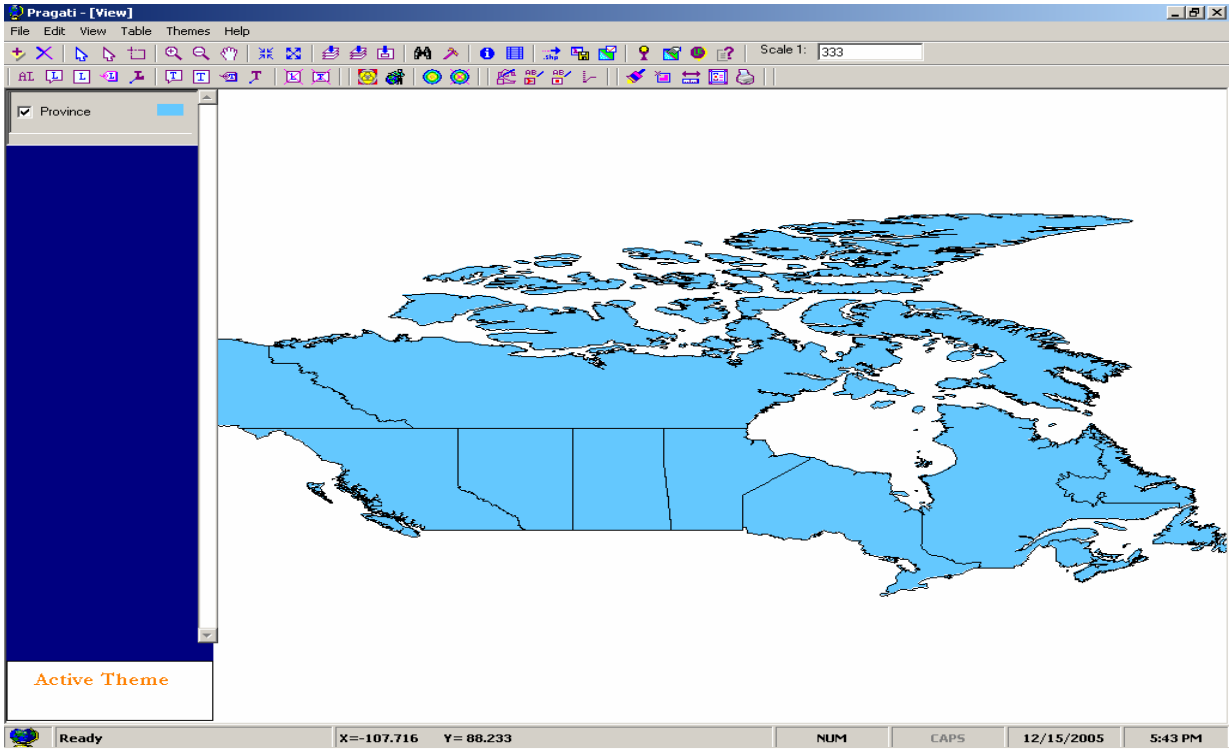


Figure 6.13 Province file – Polygon feature – Overlay file

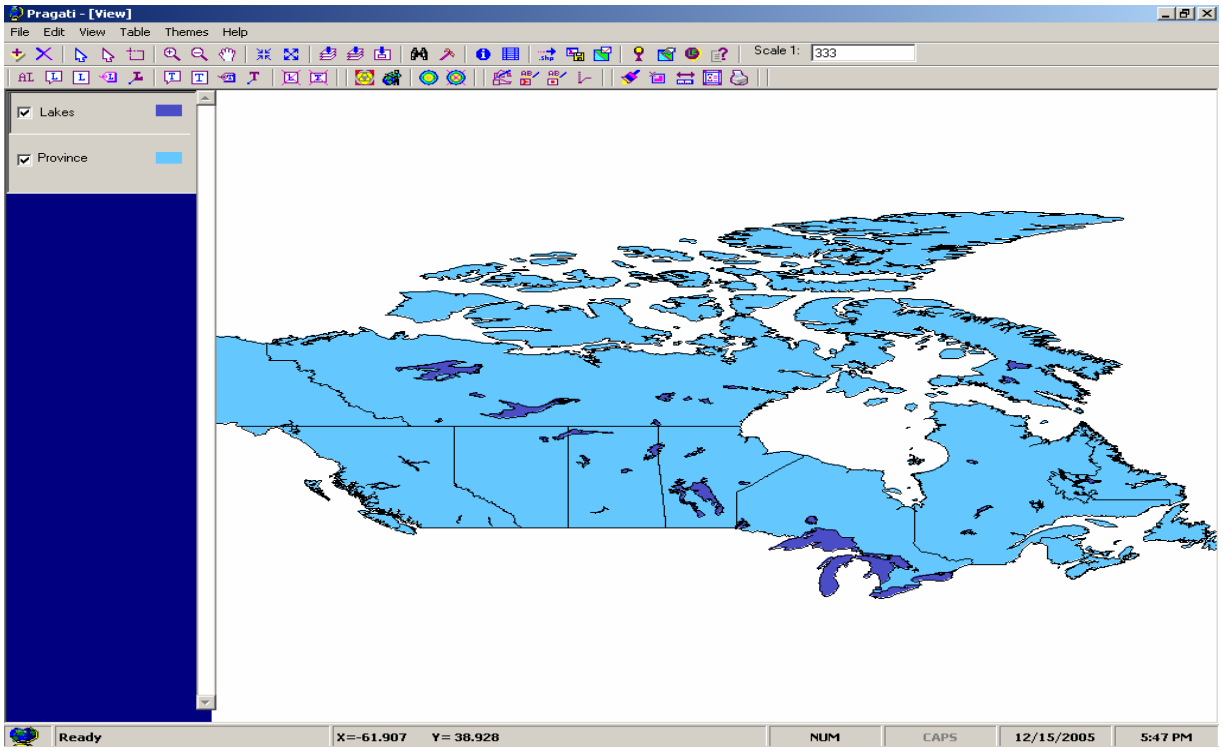


Figure 6.14 Lakes and Province file together

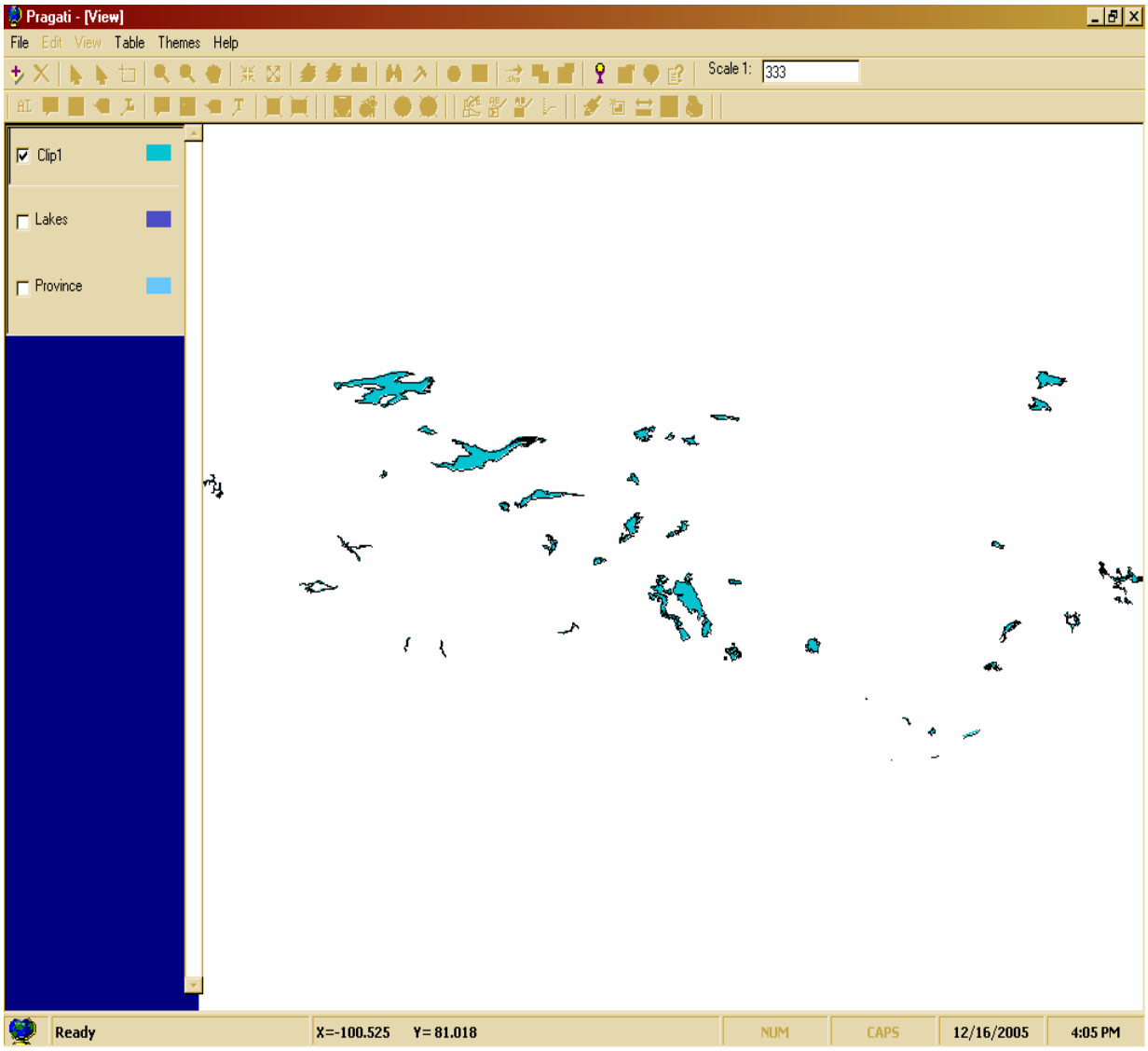


Figure 6.15 Intersected file – Polygon feature – Output file

6.2.3 Polygon – Polygon Union Snapshots

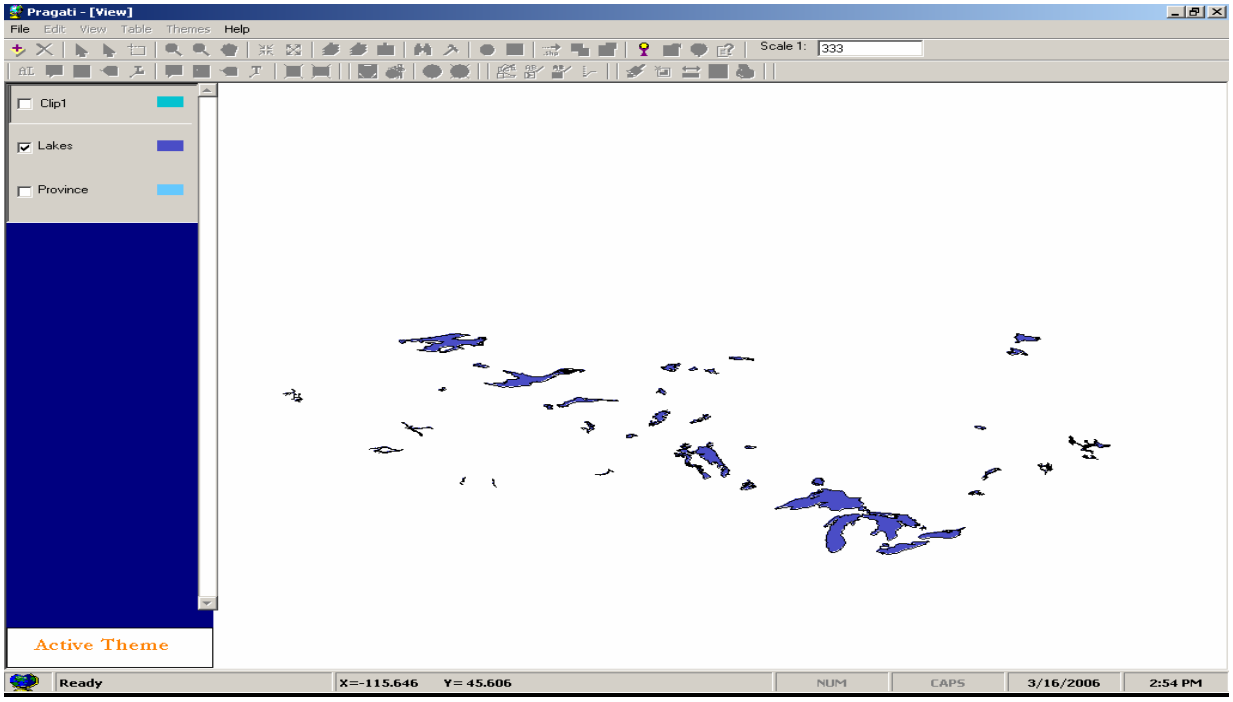


Figure 6.16 Lakes file – Polygon feature – Inputfile

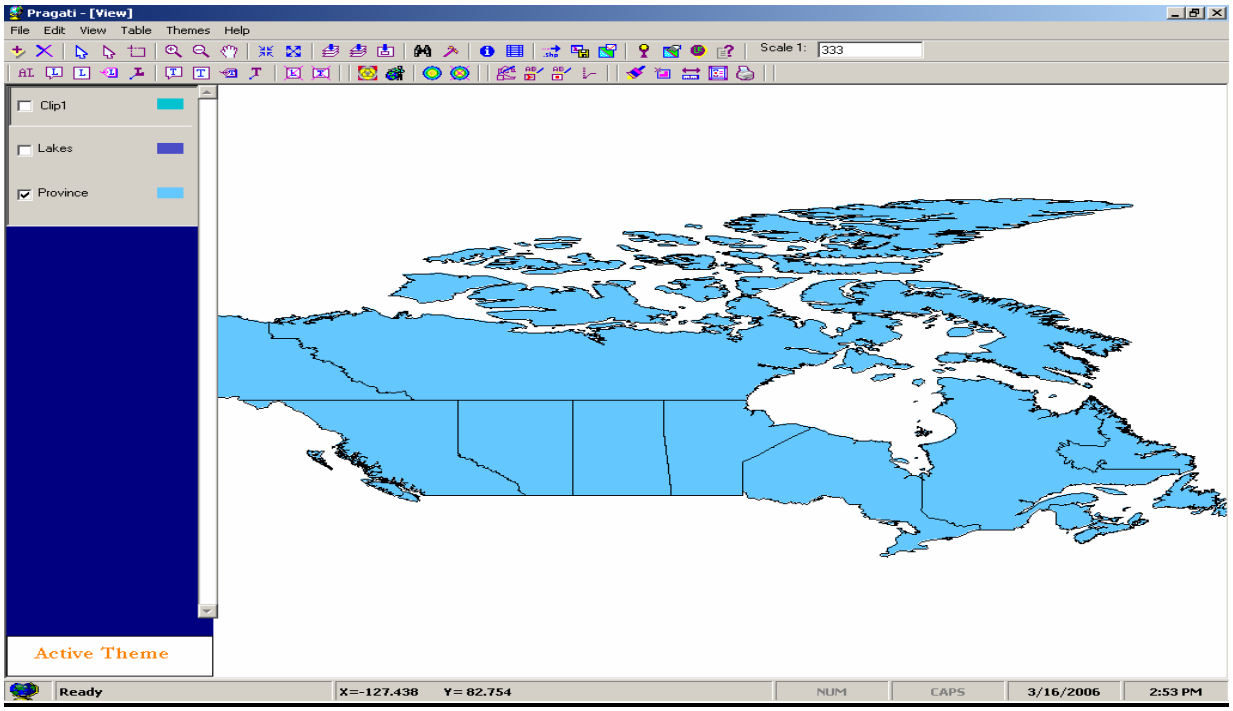


Figure 6.17 Province file – Polygon feature - Overlayfile

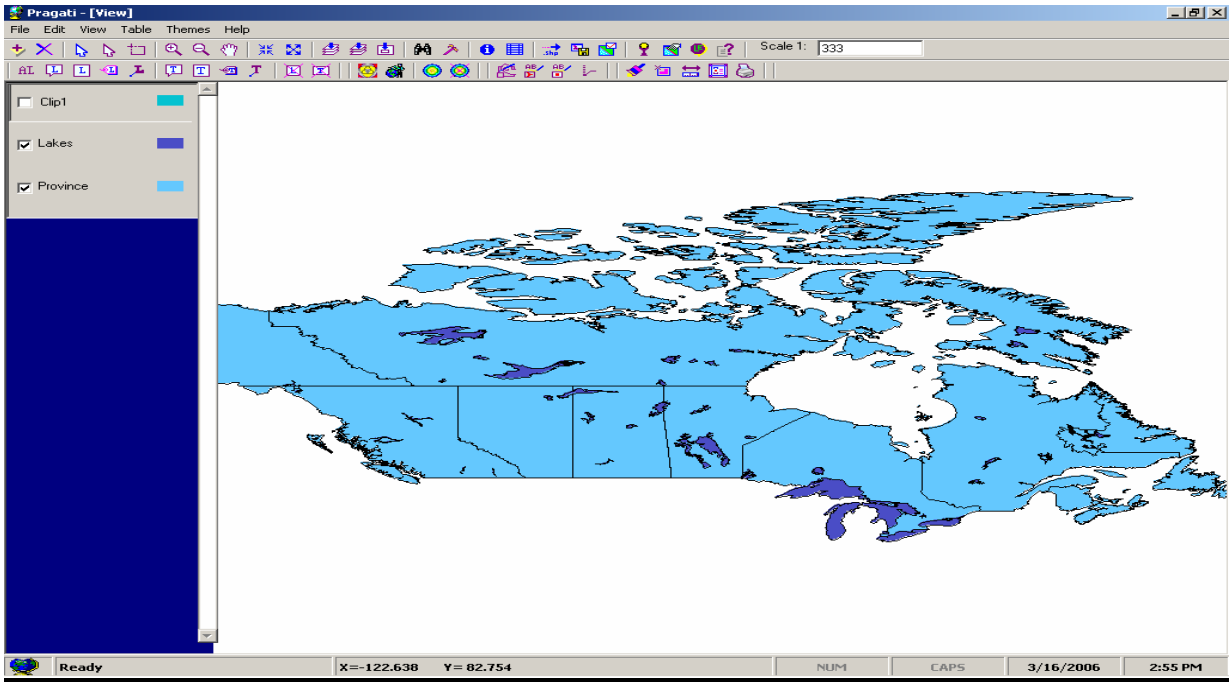


Figure 6.18 Both Province and Lakes files together

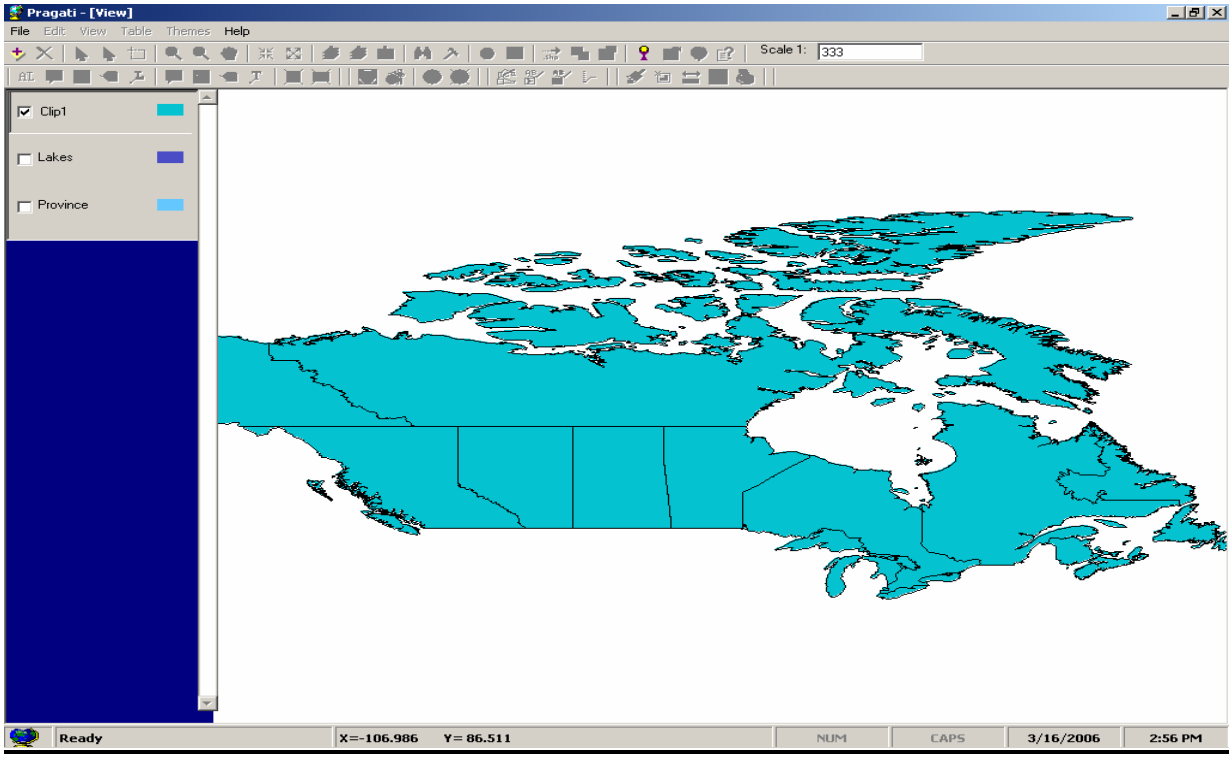


Figure 6.19 Unioned file – Polygon feature - Outputfile

6.3 Function and Performance

In order to allow user to input the parameters he must be provided with good GUI and then the values of this parameters must be stored in variables. Then these values must be passing to the GGIS control which contains the coding of the whole thing. So GGIS has the implementation of the algorithm in Visual C++. And GUI is added in Pragati using Visual Basic.

Chapter 7

IMPLEMENTATION

- Member Variables
- Member Functions
- Languages and Tools
- Platform Dependencies
- Constraints
- Implementation Challenge

7.1 MEMBER VARIABLES

m_shapeType - specifies what type of feature(shape) file is having

m_numPoints - Total number of points in the polyLine

m_numParts - Total number of parts, mainly line is consist of the line segments so total number of segments in the line and total number of rings in case of the polygon

m_points - Pointer object to the point feature ,to access the points of the line

m_parts - Integer pointer pointing to an array having starting index of the each part stored in it

m_box - Bounding box of the line or polygon,double array having four values stored in it

xmin - minimum x co-ordinate value
ymin - minimum y co-ordinate value
xmax - maximum x co-ordinate value
ymax - maximum y co-ordinate value

m_Overlay_theme --- Shape file which is used to Intersect the other file

m_Input_theme --- Shape file which is used as input file to be clipped, intersected or unioned with the overlay file

m_Output_theme --- Shape file which is used to store the output got by doing intersection or union of the two shape files namely input file and overlay file

interX[] --- Stores the x co-ordinates of the intersection points

interY[] --- Stores the y co-ordinates of the intersection points

interCount --- Stores the total number of intersections of the line segment consist of ending points p1 and p2 with overlaypolygon

totalPolylines --- Total number of polylines shape file is having

totalInputPolygons --- Total number of polygons in the InputFile

totalPolygons --- Total number of polygons in the OverlayFile

totalParts --- Total number of parts(segments) polyLine is having or total number of parts(rings) polygon is having

firstIndex --- index of the first point of part

lastIndex --- index of the last point of part

7.2 MEMBER FUNCTIONS

void CGGISCtrl::GeoProcessingInterLine(LPCTSTR m _Overlay_theme, LPCTSTR m_Input_theme, LPCTSTR m_Output_theme)

This function is used to do the Intersection of the input file(m_Input_theme) having the feature polyline with the overlay file having the feature polygon (m_Overlay_theme)and stores the output in the output file(m_Output_theme)

void CGGISCtrl::GeoProcessingInterPolygon(LPCTSTR m_Overlay_theme, LPCTSTR m_Input_theme, LPCTSTR m_Output_theme)

This function is used to do the Intersection of the input file(m_Input_theme) having the feature polygon with the overlay file having the feature polygon (m_Overlay_theme)and stores the output in the output file(m_Output_theme)

void CGGISCtrl::GeoProcessingUnion(LPCTSTR m_Overlay_theme, LPCTSTR m_Input_theme, LPCTSTR m_Output_theme)

This function is used to do the Union of the input file(m_Input_theme) having the feature polygon with the overlay file having the feature

polygon(m_Overlay_theme)and stores the output in the output file(m_Output_theme)

void CShapePolygon::Intersect(CShapePoint p1, CShapePoint p2, double interX[], double interY[],int & interCount)

This function is called with the two points p1 and p2 and it gets the intersections of the line segment consist of these points as end points with the OverlayPolygon
So mainly intersection strategies are performed over here

BOOL CShapePolygon::InOut(double x1,double y1)

Returns true if the point with (x1,y1) co-ordinates is within the polygon
So basically it performs "point in polygon" strategies

CShapePoint::CShapePoint(int shape, double x, double y)

Constructor to initialize a point with (x,y) co-ordinates

double CShapePoint::GetX()

Returns x co-ordinate of the point

double CShapePoint::GetY()

Returns y co-ordinate of the point

void CShapePoint::SetXY(double x, double y)

Sets co-ordinates of the point as (x,y)

CShapePolyLine::CShapePolyLine(int shape, int nparts, int npoints, CShapePoint *points, int *parts, double *box)

Constructor to initialize polyLine values when we want to create a new polyLine mainly when we want to create output polyLine

double CShapePolyLine::GetBoundPoint(int i)

This function returns one of the bounding points

returns xmin value if i = 0
returns ymin value if i = 1
returns xmax value if i = 2
returns ymax value if i = 3

int CShapePolyLine::GetTotalParts()

This function returns total number of parts(segments) polyLine is having

int CShapePolyLine::GetTotalPoints()

This function returns total number of points of the polyLine

int CShapePolyLine::GetPartFirstIndex(int l)

This function returns index of the first point of part(segment) l

int CShapePolyLine::GetPartLastIndex(int l)

This function returns index of the last point of part(segment) l

CShapePoint CShapePolyLine::GetPoint(int i)

This function returns point having index i

BOOL CShapePolygon::CompareBox(double x1, double y1, double x2, double y2)

This function returns true if bounding box of the InputPolyLine or InputPolygon is intersecting with the bounding box of OverlayPolygon

BOOL CShapePolygon::ComparePartBox(double xmin, double ymin, double xmax, double ymax, double xminP, double yminP, double xmaxP, double ymaxP)

Compares that bounding box of the polygon is intersecting with the bounding box of OverlayPolygon or not

double CShapePolygon::GetBoundPoint(int i)

Returns the ith point of the bounding box

CShapePoint CShapePolygon::GetPoint(int i)

Returns a point with index i

int CShapePolygon::GetPartFirstIndex(int l)

Returns part's first index

int CShapePolygon::GetTotalParts()

Returns total number of parts(rings)

int CShapePolygon::GetTotalPoints()

Returns total number of Points in polygon

int CShapePolygon::GetPartLastIndex(int l)

Returns part's last index

BOOL CShapePolygon::CompareBox1(double x1, double y1, double x2, double y2)

Returns true if the bounding box of the OverlayPolygon is same as InputPolygon with the bounding box having bounds x1,y1,x2,y2

int CGGISCtrl::GetTotalInputPoly()

Returns total number of polygons of the InputFile

int CGGISCtrl::GetTotalPoly()

Returns total number of polygons of the OverlayFile

7.3 Languages and tools

To implement, test and validate the efficiency of the algorithms concerning the Intersection and Union of the maps we need to develop language skill in

-Visual Basic

-Visual Studio

We need to implement our research work using two different languages because we need to do two things

1) GUI is to be done in **Visual Basic** which has to be integrated in Pragati S/W
2) Coding is to be done in **Visual C++** which has to be integrated in GGIS control

So we use "Microsoft Visual Studio" for the implementation of research work in Visual Basic and Visual C++.

We found wide range of facilities provided by VC++ that would be of great help while we progressed further with the work.

Visual C++ 6.0 includes the comprehensive Microsoft foundation classes, which simplify and speed development of window applications. It includes sophisticated resource editor to design the complex dialog boxes, menu, toolbars, images, and many other elements of modern window application. There is an excellent integrated development environment called developer studio that present graphical views of your application's structure as you develop it.

Totally integrated debugging tool lets you examine in minute detail every aspect of your program as it runs.

7.4 Platform dependencies

Current Platform

I am implementing my work using the Windows 2000 Professional.

Windows 2000 Professional is suitable operating system for geographic system development and also for the use of the software related to the geographic system. Also it gives more update facilities as Microsoft's new Web-based resource site, automates driver and system file updates, and provides up-to-date technical support. Windows 2000 can review device drivers and system software on your computer, compare those findings with a master database on the Web, and then recommend and install updates specific to your computer. You can also revert to a previous device driver or system file using the uninstall option.

Dependency

As we use Microsoft Visual Basic and Microsoft VC++ ,our developed system is dependent on the "Windows" platform.

7.5 Constraints

- Memory size
- Processor speed
- Maps are stored as shape file
- Dependency on the Windows platform

Shape file is a constraint as we can only use the information of the map which is stored as shape file to get the information from it and store the processed map in only shape file format.

So we can not use maps stored as image in other format like jpg , gif etc.

7.6 Implementation Challenge

I am developing algorithms for the overlays related to the map which mainly covers Intersection and Union. Here map is stored as shape file. Shape file can be one of three types

- Point
- Line
- Polygon

Point file is used to show the particular offices, centers, nodes etc within defined area.

Line is used to show rode network, water canals etc within defined area.

Polygon is used to show areas like forest area, polutated area etc.

So here comparison is required between the segments of one shape file with other shape file to know the common part and uncommon part for the intersection and union.

So we need to check for the line-line and polygon-polygon intersection for the overlays. Shape file for the big map is consisting of hundreds or more lines and polygons.

So for the small map comparison for the intersection and union required is much less but for the big map this turns in more memory utilization to store the local variables and data .Also more processor utilization is there because of more computation is needed in the comparison.

- Efficient Memory utilization

- Efficient Processor utilization

So my role is to develop efficient algorithms for supporting overlays as it requires efficient memory utilization to control memory leakages and memory overflow and to speedup the processing

Also we need to learn and understand the shape file structure to access the information from it to use it to make proper algorithm using its information and also to store the information in shape format to implement proposed algorithm.

Chapter 8

MAINTANANCE

- Corrective maintenance
- Adaptive maintenance
- Perfective maintenance
- Preventive maintenance

Once you develop a project a duty is not completed. Sometimes the application is unstable. It still works, but every time a change is attempted, unexpected and serious side effects occur. Yet the application must evolve.

The maintenance of the existing software can account for over 60 percent of all effort expended by a development organization, and the percentage continues to rise as more software is produced.

Much of the software we depend on today is on average 10 to 15 years old. Even when these programs were created using the best design and coding techniques known at the time, they were created when program size and storage space were principle concerns. They were then migrated to new platforms, adjusted for changes in machine and operating system technology and enhanced to meet new user needs all without enough regard to overall architecture.

The result is the poorly designed structures, poor coding, poor logic and poor documentation of the software systems we are now called on to keep running…

The ubiquitous nature of change underlies all software work. Change is inevitable when computer-based systems are built; therefore, we must develop mechanism for evaluating, controlling and making modifications.

"Software Maintenance" is, of course, far more than "fixing mistakes". We may define maintenance by describing four activities that are undertaken after a program after a program is released for use. We have defined four different activities:

8.1 Corrective maintenance

Even with the best quality assurance activities, it is likely that the customer will uncover defects in the software. Corrective maintenance changes the software to correct defects.

8.2 Adaptive maintenance

Over time, the original environment (e.g., CPU, operating system, business rules, external product characteristics) for which the software was developed is likely to change. Adaptive maintenance results in modification to the software to accommodate to its external environment.

8.3 Perfective maintenance

As software is used, the customer/user will recognize additional functions that will provide benefit. Perfective maintenance extends the software beyond its functional requirements.

8.4 Preventive maintenance

Computer software deteriorates due to change, and because of this, preventive maintenance must be conducted to enable the software to serve the needs of its end users. In essence, preventive maintenance makes changes to computer programs so that they can be more easily corrected, adapted, and enhanced.

Only about 20percent of all maintenance work is spent "fixing maintenance". The remaining 80 percent is spent adapting existing systems to changes in their external environment, making enhancements requested by users, and reengineering an application for future use.

CHAPTER 9
CONCLUSION AND SCOPE OF FUTURE WORK

- Conclusion
- Limitations
- Scope of Future Work

9.1 Conclusion

As we know that GIS systems mainly concerning to vector systems deal with hundreds of points, lines and polygons , also it requires repetitive comparison of its elements and also repetitive calling of same methods like point in polygon strategy is called repetitively in both Intersection and Union Procedures.

So it uses more processing power and efficiency decreases in terms of time and processing power also efficiency decreases because of the complex algorithms and datastructure.

So algorithms presented over here are simple and datastructure is also simple as we use array for the list. Also it decreases number of comparisons between the vector elements by using efficient heuristics and strategies. So this algorithms do optimal use of the processing power and increases efficiency in terms of the time.

9.2 Limitations

Platform Dependency : As application is developed using Visual Basic and Visual C++ it is Dependent on windows platform

Fix file format : Applicable to only images having shape file format

9.3 Scope of Future work

Same thing can be developed using platform independent language like Java so it can be applicable to any platform and we can remove the platform dependency constraints of the current platform.

Strategies explained here and used in this system uses list of x, y co-ordinates and edges. But it is done on the fly in this system so we can improve the efficiency by preprocessing this list before doing actual function. It also cost prior

processing power to do the preprocessing and additional complexity but we can do it and compare the net efficiency between current system and new system.

REFERENCE

[1] A. E. Middleditch,T. W. Stacey,S. B. Tor , "Intersection algorithms for lines and circles",ACM Press New York, NY, USA,1988

[2] George Nagy,Sharad Wagle , "Geographic Data Processing" ,ACM Press New York, NY, USA ,1979

[3] D. P. Dobkin,D. Silver, "Recipes for geometry and numerical analysis - Part I: an empirical study" ,ACM Press New York, NY, USA ,1988

[4] Henry G. Baker ,"Corrigenda: intersection algorithms for lines and circles",ACM Press New York, NY, USA,1994

[5] Practical Visual C++ 6 by Jone Bates and Tim Tompkins

[6] ARC/INFO user Manual by ESRI

[7] Paul A. Longley, Michael F. Goodchild, David J. Maguire, Geographic Information Systems and Science – Second Edition

[8] Desktop Application VC++ published by Microsoft Press

[9] www.gis.com/whatisgis/index.html

[10]www.esri.com/software/arcgis/arcview/index.html

[11]http://www.esri.com/software/arcgis/about/desktop.html