INTERACTIVE ACCESS TO GIS DATA

By

SOHAIL AMJAD

Master of Science

University of Karachi

Karachi, Pakistan

1992

Submitted to the faculty of the
Graduate College of the
Oklahoma State University
in partial fulfillment of
the requirements for
the Degree of
MASTER OF SCIENCE
July 1996

INTERACTIVE ACCESS TO GIS DATA

Thesis Approved:

[illegible]

Thesis Adviser

Blayne E. Mayfield

J Chandler

Thomas C. Collins

Dean of the Graduate College

OKLAHOMA STATE UNIVERSITY

ACKNOWLEDGMENTS

I wish to express my appreciation and gratitude to my advisor Dr. Mitchell L. Neilsen for accepting to be my major advisor, his advice, and his intelligent guidance, for the completion of my thesis work. His perseverance, and hard work inspired me to venture into the advanced aspects of this work. I would like to express my sincere thanks to Dr. J.P Chandler for the guidance and help he has given me during the entire period of my graduate studies. I also wish to thank Dr. Blayne E. Mayfield for serving on my graduate committee.

Additionally, I want to thank Dr. David Waits, my supervisor in the Department of Geography, Oklahoma State University, for his support and for employing me as a Graduate Research Assistant.

My respectful thanks goes to my parents Mr Altaf Hussain and Mrs. Altaf Hussain. Last but not the least, I would like to express my sincere gratitude to all other member of my family for the love, continued support and encouragement, without which this endeavor would not have been successful.

TABLE OF CONTENTS

OKLAHOMA STATE UNIVERSITY

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

In the early days of stand-alone, non-networked computing, users executed programs only on local machines. With the growth of networks and distributed computing, the client/server paradigm emerged. In this paradigm a client requests services from a remote server. After some processing, the server replies and typically data get displayed back to the client. The advantage of this model is that users with low cost computers can share access to high cost services, and organizations can optimize company-wide distribution of the services to minimize cost and maximize efficiency. A number of protocols and programming interfaces, such as HyperText Transport Protocols (HTTP) [1], OSF/DCE, have been developed within the client/server paradigm.

However, this paradigm does not automatically scale up very well in the new world of the Internet and laptop computing. This new world is characterized by :

- *Wide Area Interconnected Networks.* People using the Internet are connecting from their sites (possibly their homes) to servers located anywhere in the world, crossing several types of physical networking (satellite, optical fiber, wireless, etc.) in one connection, each with a very different type of bandwidth and latency

OKLAHOMA STATE UNIVERSITY

properties. The overall performance and behavior are not that of local network or leased lines.

- *Nomadic Computing*: People travel with their laptops and turn them on and off, which implies network connection are geographically moved and turned on and off as well.
- *Platform Independence*: Services offered to users where the service provider has no control of the user's platform unlike an enterprise context, where desktops and servers follow a corporate equipment plan.

Although the typical connection-oriented client/server model may still be appropriate in this new world, for some applications it is not, for example, applications where a client wants to have the server continue computations even though the client has been turned off, and get the results when it is booted again, or applications where the network bandwidth or latency make the application impossible to run with satisfactory response time; e.g. applications which require the visualization of complex data. An image or symbolic diagram is computed from data and displayed to the scientist. As a scientist runs experiments that modify the data, the image is modified.

As long as scientists visualize static displays, the bandwidth and latency of the network are minor but not overwhelming issues. When it comes to dynamic displays, it no longer makes sense to compute images on the server side and send every image back to the client. The network becomes a bottleneck.

OKLAHOMA STATE UNIVERSITY

A solution to those problem is that of mobile code. In this model, code resides on a computer is shipped over the network to execute onto a remote place. Mobile code can be used to run remote computation while the original computer is down, or it can be used to overcome the latency and bandwidth issues by first shipping code and data over the network and then executing that code locally, without requiring any further network traffic.

Mobile code inherently requires open systems; mobile code applications shipped over the network must execute on all the platforms available. It also raises issues of portability , security, interpretability and salability, as well as new challenges in distributed computing.

In 1995, Sun Microsystems introduced the Java[1] technology and the associated HotJava Web browsers. Java has the potential to significantly enhance the capability of the World Wide Web (WWW) and more generally mobile code technology.

Java's contribution is to augment the present WWW capabilities to transmit static information with a new capabilities to transmit programs in a secure and portable fashion Using Java, smart data, new protocols, animation, distributed programs, multimedia publishing, scientific visualization, process control, networks games, network management tools can be disseminated via the Internet.

1.1 Thesis

Geographic Resources Analysis Support System (GRASS) software was made available via WWW using Common Gateway Interface (CGI) and Java language. This facilitated access to geographical data sets and allowed simple analyses to be performed without

[1] Java and HotJava are the registered trademarks of Sun Microsystem, Inc.

actually downloading data or software in an interactive, and intuitive way. A platform-independent, display-only map creation interface provided a good browsing facility for potential data consumers. The data conversion capabilities of GIS were used to demonstrate the presentation of spatial data using image maps allowed users to reach particular data sets quickly and efficiently. This interface demonstrated the capacity to view manipulate, and distribute geographic data via WWW in an efficient, organized, interactive, and user-friendly manner. This Interface is a major advancement in information sharing for GIS data.

This work also demonstrate the potential of the Internet for overcoming spatial data access problems and for facilitating the use of GIS by large numbers of diverse users of Internet. The Internet is an international communication infrastructure comprised of thousands of regional networks scattered throughout the globe [5]. This world-wide connectivity includes more than 13,500 foreign networks and over 20 million users in over 50 countries. Presently, more than 15,500 billion bytes of information are transferred per month across these networks.

The objective of this work is to transfer a simple task from the server side to client side in order to improve performance, and the integration of heterogeneous, distributed GIS methods and data. This is achieved using the Java language after a convenient way to transfer Java executable programs called applets to the client side. These applets improved performance with respect to user interaction and network bandwidth latency

We developed a prototype to access current, ubiquitous, intuitive, and interactive GIS data. The main features of the prototype are user-controllable image overlaying, client side

OKLAHOMA STATE UNIVERSITY

computing of Universal Transverse Mecator (UTM), and threaded loading of resources to minimize stagnant time while applet set up.

GRASS was chosen for this project because it has an open file format [10] and it is a public domain package. GRASS was developed using federal money and distributed without charge. GRASS includes a graphics production system with powerful map production capabilities. This package is written in C and is UNIX [2] oriented. Like the other GIS packages, however, GRASS requires its users to have significant knowledge of not only the GIS, but also of computers, including the UNIX system as well as the input/output devices, the database being analyzed, the areas under analysis, and the requirements of the analysis [18]. The ability to produce thematic maps is only a small part of GRASS's powerful functionality, but users still must go through a learning curve to accomplish this job. It is impractical to ask a casual user of GRASS to take a great deal of time to go through the entire learning process for occasional use.

1.2 Organization

The thesis is divided into the following chapters

- Chapter 2: A discussion on previous work related to GIS and WWW is presented.
- Chapter 3: Literature overview is presented in this chapter.
- Chapter 4. A detail discussion of the work and implementation details are given.

[2] UNIX is the registered trademark of UNIX System Laboratories, Inc.

OKLAHOMA STATE UNIVERSITY

- Chapter 5: Performance metrics used to evaluate the performance of GRASS-applet and the result are explained.
- Chapter 6: A summary of the thesis and suggestions for future work are presented.

OKLAHOMA STATE UNIVERSITY

CHAPTER 2

RELATED WORK

In this section, we place our work in the context of GIS and World Wide Web and accessing database system from GRASS-applet. This list of approaches, concepts, and systems discussed does not claim to encompass all relevant work in the area. We restrict the discussion to those that we think to bear the most direct relationships to our problem and solution.

2.1 GIS and WWW

In a related project [11] ArcView is used at the client side to display maps. This method also has the advantage of avoiding the processing of data at the server side, but additional software is necessary for every client. Our approach aims at a system which is accessible with a standard WWW browser. This enables the system to provide data and methods to everyone and avoid a lot of cost for large institutions using the data.

Some other scientists are working with GIS and WWW; most of them are situated in the United States. Often information about the systems are only available on the web itself. Good starting points for search are [28], [26], [3] and [19]. Some examples of the capabilities of the other system are located at [19], [12], [14] and [17] These

OKLAHOMA STATE UNIVERSITY

system offers different subsets of the functionality required in Environmental Information System (EIS). Our approach includes additional features such as combination of maps (overlaying images), client side computing of Universal Transverse Mecator (UTM) coordinates and user-friendly interface as well as to transfer simple tasks from the server side to the client in order to provide the functionality of Geographic Information System using off-the-self WWW browsers. No additional software tools at the client side are required. Furthermore, GRASS-applet improves the performance with respect to user interaction. It can be used to overcome some of the current limitation of the World Wide Web.

OKLAHOMA STATE UNIVERSITY

CHAPTER 3

LITERATURE OVERVIEW

First, a brief overview of GIS integrated GIS data organization, browsing, analysis, and transfer system is given. Secondly, an overview of Object-Oriented methodology and brief introduction to Java Programming Language is presented. Following this overview is a brief tour of important WWW protocols and conventions.

3.1 Geographic Information System

In the strictest sense, GIS is a computer system capable of assembling, storing, manipulating, and displaying geographically referenced information; i.e. data identified according to their location [19]. Practitioners also regard the total GIS as including operating personnel and the data that go into the system

GIS data are usually stored in one of two representations: raster or vector. The former uses a cell-based approach for data handling and performing analyses. Raster data may be viewed as a matrix of geographically-referenced cells, each containing an attribute value. The latter representation handles the data in terms of vectors (such as points, arc, and polygons). Both approaches have their advantages and disadvantages Selection of an appropriate data representation or format depend upon problem at hand.

OKLAHOMA STATE UNIVERSITY

Geographic information system technology can be used for scientific investigation, resource management, and development planning. For example, a GIS might allow emergency planners to easily calculate emergency response times in the event of a natural disaster, or a GIS might be used to find wetlands that need protection from pollution.
An active GIS market has resulted in lower costs and continual improvements in the hardware and software components of GIS. These developments will result in a much wider application of the technology throughout government, business and industry. Through a function known as visualization, a GIS can be used to produce images; not just maps, but drawing, animation, and other cartographic products. These images allow researchers to view their subjects in ways that have never been seen before. The images often are equally helpful in conveying the technical concepts of GIS study subjects to non-scientists.

3.2 GRASS: A Powerful GIS Package

GRASS is a powerful, general purpose software package with a great flexibility. The basic requirements to use the software are relatively high. Users must have knowledge of the Geographical Information System as well as computer including input/output devices, and operating systems.

GRASS was designed and developed by researchers at the U.S Army Construction Engineering Research Laboratory. It is written in C and is UNIX oriented [18]

OKLAHOMA STATE UNIVERSITY

GRASS was first released in August, 1985 and version 4.2 was released in August, 1995. In version 4.0, each command can be used either interactively or through the command-line interface. The command-line interface provides programmers with a great working flexibility. GRASS is distributed with source code. Portability is the first priority in GRASS design. It comes before a friendly user interface and execution speed.

GRASS has many capabilities, including the handling of different representations of data. These include

- *Raster Data*: raster (or grid cell type) data can be used for analyzing, overlaying and modeling aerials features such as soil types or forested areas.
- *Vector Data*: vector data can be used to represent linear features such as roads, streams or areas edged and can be combined with raster data for display purpose or for analysis.
- *Point Data*: point data can be used to represent landmarks or the location of significant sites.
- *Imagery Data*: the ability to display, Geo-refrence, compare and classify satellite and aerial photographic imagery.
- *Mapdev*: the ability to input map data and print hard copies on various printers, including the ability to create output to a pen plotter that runs or emulates HPGL.
- *DBMS*: the ability to link to a Database Management System for help in managing data.

OKLAHOMA STATE UNIVERSITY

With the current release of GRASS, the development staff also distribute a Spearfish, South Dakota data set. The Spearfish data set covers two topographics 1:24,000 quads in western South Dakota. The names of quads are Spearfish and Deadwood North, SD. The area covered by data set is in the vicinity of Spearfish, SD and includes a majority of the Black Hills National Forest. We use this data set as a test data for the GRASS-applet.

3.3 Object-Oriented Programming

An object-oriented programming (OOP) is a new step of the programming concept. It takes recent developments in programming language to their next logical step to increase clarity, modularity, and programming efficiency [27]. An object oriented language is a language which centers its programming paradigm on data, rather than the traditional procedural approach of non-object-oriented languages. Data and operations on that data are logically grouped Operations on these data are called methods. The resulting package, comprising the data plus the methods specific to data, is called object. Objects are organized into classes according to the information they represents and the methods they share to operate on that data. The data that an object holds are often called the instance variables of the objects.

Object-oriented languages usually provide mechanisms to support four key characteristics: encapsulation, inheritance, polymorphism, and dynamism.

OKLAHOMA STATE UNIVERSITY

3.3.1 Encapsulation

The notion of encapsulation refers to the ability to hide the particular data structures and methods used in the implementation of particular class [6]. It helps the programmer organized the data and structure of a program in a more modular manner by effectively viewing implementation as a solved problem, thus facilitating incremental development and software re-use. Encapsulation is also used to solve a name-space problem so that instance variables and methods of different classes can use the same name.

3.3.2 Polymorphism

Polymorphism is an object's ability to select the correct internal method based on the type of data received in a message [6]. A print object may receive a message containing an integer, a real number, or an ASCII string. The print object may take an appropriate action depending on the incoming message without knowing the contents of the message.

3.3.3 Inheritance

Inheritance is the ability to create classes that will automatically model themselves on other classes. It is a language mechanism for deriving a new class of objects from an existing class [6]. When a class B has been defines so as to model itself on class A, class B is a child class of class A. It inherits data and method from class A. Class B need only contain the actual code and data for new or changed methods. Normally a class can have zero or more parent calls or child class.

OKLAHOMA STATE UNIVERSITY

3.3.4 Dynamism

Dynamism encompasses a wide range of capabilities, both at compile and runtime, to make the language and system more flexible and powerful for both the user and developer. Dynamic typing means the ability to determine at runtime the type of an object and its operations. In other words, an object's type can remain ambiguous at compile-time but will be determined at runtime. Dynamic loading and binding make it possible for an object's definition and instantiation to be deferred until when it is needed at runtime, thereby allowing for the addition of object components dynamically as a program runs.

3.4 Java Programming Language

In the context of distributed programming languages for the World Wide Web, Sun Microsystems has recently introduced the Java programming language environment. Java was originally designed by Sun to facilitate the development of embedded system software [24], but has initially been positioned as a language for web programming because of its ability to simplify the development of flexible, portable, distributed applications with a high-level graphical user interface [25]. Java is derived from C and C++, but the language has been restricted to eliminate many of the most costly common programming errors. Java shows promise of greatly improving developer productivity in the great domain.

Java originated as part of a research project to develop advanced software for a wide variety of networked devices and embedded systems[23]. The research project initially chose to use C++ for the development. But subsequently the developers encountered so many difficulties with C++ that they decided it would be best to design an entirely new

OKLAHOMA STATE UNIVERSITY

language environment. Java offers a number of important improvements over developing software in currently popular languages such as C and C++.

Java borrows the familiar syntax of C and C++. Like C++, Java is object-oriented, but it is much simpler than C++. Java's designers intentionally discarded redundant language features that were present primarily to support backward compatibility with legacy code. An additional benefit of its simplicity is the small size of its run-time system. Sun reports that the basic interpreter is about 40KB, and that basic libraries and thread support add approximately 175KB[25].

The development cycle is much faster because Java supports both interpreted and just-in-time compiled implementations. During development and rapid prototyping, developers save time by using the interpreter.

Application software is more portable because the Java environment carefully specifies a machine independent intermediate bytecode representation which can be transferred between heterogeneous network nodes and interpreted or compiled to native code on demand by the local Java run-time environment (see Fig 3.1).

Application software is more robust because Java's run-time environment provides automatic garbage collection. The Java language has been designed to eliminate the possibility of dangling pointers and memory leaks.

Applications are adaptable to changing environments because code modules can be downloaded dynamically from other network nodes without necessitating a system restart. Security is enforced by built-in protection against viruses and other tampering. This protection is implemented by simple theorem provers that analyze downloaded bytecode

OKLAHOMA STATE UNIVERSITY

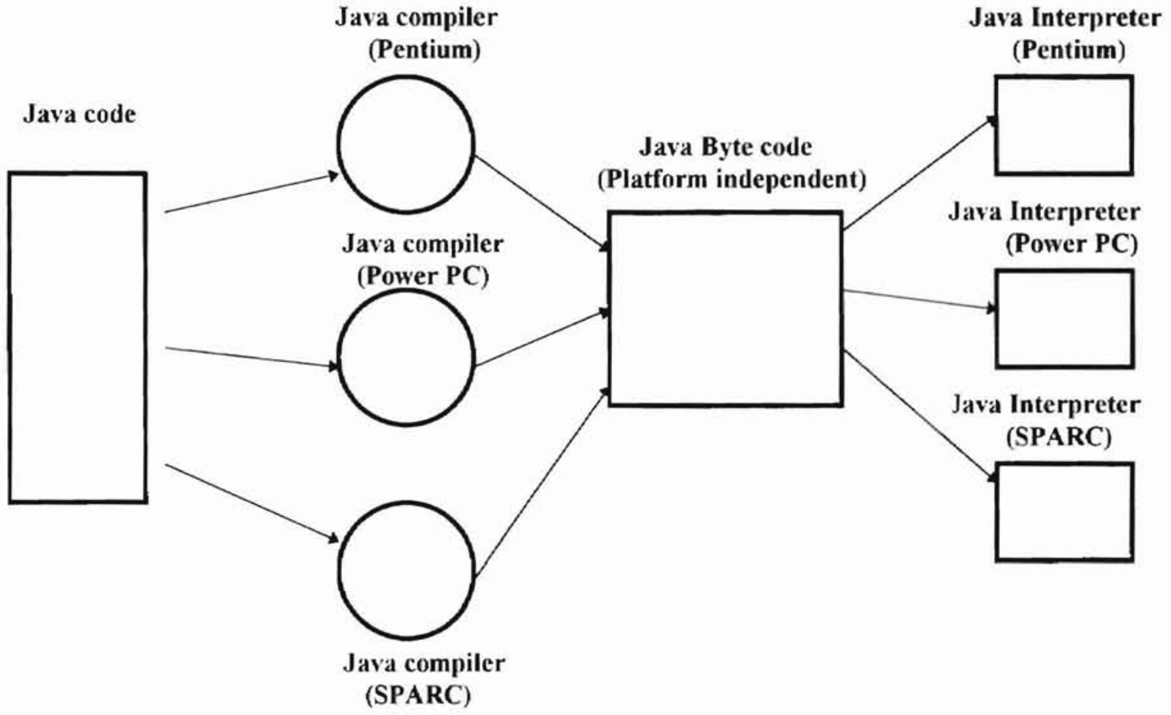


Fig 3.1 Representation of Java Platform Independent Bytecode

OKLAHOMA STATE UNIVERSITY

before attempting to execute them.

High performance is achieved by incorporating support for just-in-time translation of portable bytecode to the native machine language of the local host. According to Sun, performance of translated code is roughly equivalent to the speed of current C and C++ programs.

3.5 WWW Conventions and Protocols

The World Wide Web (also called WWW, W3, or Web) is a Wide-area hypermedia information retrieval system providing access to a myriad of documents and data on the Internet. WWW is also a body of software and a set of Protocols and Conventions to provide easy and consistent access to information on the Web.

With the development of the WWW, opportunities arose to organize various resources found on the Internet in an efficient and user-friendly manner. One major type of resource is GIS data.

Following is a brief overview of important Protocols and Conventions of WWW. These includes the Hypertext Markup Language, the Hypertext Transfer Protocols, and Common Gateway Interface.

3.5.1 Hypertext/Hypermedia

The operation of the Web relies mainly on hypertext and hypermedia as means for interacting with the users. Hypertext is basically the same as regular text but it "points" to other documents in the case of hypertext on the Web, these other documents are on the

OKLAHOMA STATE UNIVERSITY

Internet. Hypermedia is hypertext with a slight difference. Hypermedia documents have hyperlinks not only to text but other multimedia forms, such as images, sound files, video files, etc. Hypermedia can be viewed as a combination of hypertext and multimedia. Hypertext Mark Language (HTML) is standard markup language for creating and recognizing Web documents[2, 4]. As a markup language, HTML allows the user to control the information presented in a number of ways (e.g., fonts and colors). HTML documents are typically 7-bit ASCII files with formatting codes that contain information about document structure and hyperlinks. WWW uses Uniform Resource Locators (URLs) to represent hypermedia links and links to network services(hyperlinks) within HTML documents [2].

3.6.2 HyperText Transfer Protocol (HTTP)

Web software is designed around a distributed client-server architecture. A Web browser is client software that can send requests for documents to any Web server. A Web server is a program that, upon a request for a document, processes the client's request and sends back the document or an appropriate message (e.g., an error message). The processing of a request is done by the server and presentation of data is left to the client. The language that the Web client-server interface uses is the Hypertext Transfer Protocol (HTTP)[1]. It utilizes the Transmission Control Protocol/Internet Protocol (TCP/IP) for communication between Internet hosts [13].This processing of client-server interaction involves the following process.

OKLAHOMA STATE UNIVERSITY

- Client makes TCP/IP connection to the server using the URL address (domain name and port; the default port is 80 for HTTPD server)
- Server accepts the connection
- Client sends a request for the URL using HTTP
- Server processes the request and sends back the requested document as a byte stream.
- Server closes the connection and client terminates the TCP/IP connection

All Web clients and servers must be able to speak HTTP in order to send receive hypermedia documents.

There are many Web servers, including the CERN server, NCSA server, the PERL server, and other. These servers, like FTP deamon [21], are programs that respond to an incoming connection and provide a service to the client. Hypertext Transfer Protocol Daemon (HTTPD) is a public domain Web server developed by the NCSA [17] written in C for UNIX platform. HTTPD also records the date and time of requests along with the IP number of the client, which is useful for keeping track of traffic.

3.7.3 Common Gateway Interface

The Common Gateway Interface (CGI) [15] is an interface under a Web server, such as HTTPD, for running external programs or gateways. CGI facilitates the handling of the information requests and can act as a gateway for returning the appropriate document or creating a document *on-the-fly*. With a CGI, a Web server can provide information which

OKLAHOMA STATE UNIVERSITY

is not in a form readable by the client (e.g., GRASS binary raster files) and can act as a gateway between the server and the client for interaction.

Gateway programs or scripts are server-side executable programs that are run (upon request from a client) to server information. These gateways are initiated when the client requests the URL corresponding to the gateway. Since these scripts are executed on the server, gateway programs are independent of the client's operating environment. Gateways interact with the client and server using the HTTP. Gateways conforming to the HTTP specifications can be developed in any programming language, such as C, FORTRAN, Pascal, PERL, Bourne Shell, C Shell, etc.

Information requested from the server to CGI script is handles using command line arguments as well as environment variables. The environmental variables used in this study were:

- REQUEST_METHOD: The method with which the request was made. Request method "POST" is used
- QUERY_STRING: The information that follows the "?" in the URL when the gateway script is referenced. This QUERY_STRING should be decoded in the gateway script.
- CONTENT_TYPE: This defines the type of data attached with the request to the server
- CONTENT_LENGTH: The length of the content attached to the URL which is required to decode the CONTENT of the request from the client.

OKLAHOMA STATE UNIVERSITY

CHAPTER 4

IMPLEMENTATION

The following sections describe the environment in which the GRASS-applet was developed and its implementation in details. The principal objectives in designing this project are to reduce work load on the server and increase interactive performance with respect to user interaction. The model presented here is designed with these motives in mind.

4.1 Environment

The GRASS-applet is implemented on a Sun[3] SPARCstation running Solaris 2.4. This operating system along with OpenWindows Graphical User Interface (GUI), productivity tools, and DeskSet make up the complete Sun UNIX environment. GRASS-applet is coded in Java network programming language, and uses Advance Window Toolkit (AWT) function for GUI.

Being implemented in Java, GRASS-applet can be executed as an applet within HTML pages, while being browsed by Java-enabled browsers like Netscape 2.0. GRASS applet conforms to Beta Application Programmer Interface (API) and can also be executed standalone by AppletViewer provided with Java Development Kit (JDK).

[3] Sun and Solaris 2.x are the registered trademarks of Sun Microsystem, Inc.

OKLAHOMA STATE UNIVERSITY

We have developed a set of classes (a class is roughly a unit of software capable of performing a set of operation on a certain kinds of data) and CGI scripts. These classes perform interactive operations and provide GUI interface while CGI scripts runs GRASS on the server in order to generate requested maps. Following are descriptions of these classes, GUI, and CGI scripts used in this project.

4.2 Classes Description

GrassApplet Class

GrassApplet is a subclass of Java Applet and serve as a root class. It creates and lay out the interface, initiates help and other classes instances, and dispatches repaint requests to them and handles user input. User input comes from pressing on control buttons or clicking the mouse anywhere in the applet's panel. Button presses typically display sub-panel giving the user a chance to select required data. GrassApplet class creates a thread so that it can continuously update (repaint) its image canvas without taking over the process in which it is running. This class uses Runnable interface to provide its threaded behavior.

ImageMap Class

This class is used for Imagehyperlink and Information Extraction. This class retrieves additional information from the server that specifies pre-defined area in the image and display information as a pointer passes over them, and hyperlink to additional resource

OKLAHOMA STATE UNIVERSITY

when these areas are clicked. For example, in an image map of Oklahoma State when user moves the pointer the name of county is also shown on the status line of the browser, suppose that client select the Texas County. Clicking on that area leads user to Texas County page and runs GRASS-applet there. Fig 4.1 shows the sessions running ImageMap class. This image map was created using the *mapedit*
software.

ImageCanvas Class

This class implements ImageObserver which provide an asynchronous update interface for receiving notifications about image information as the image is constructed. The class is used to composite several images into a single image. When a user should turn off or on the components, the layer of composite containing it would simply be removed or replaced. Fig 4.3 shows that user has selected three different image simultaneously.

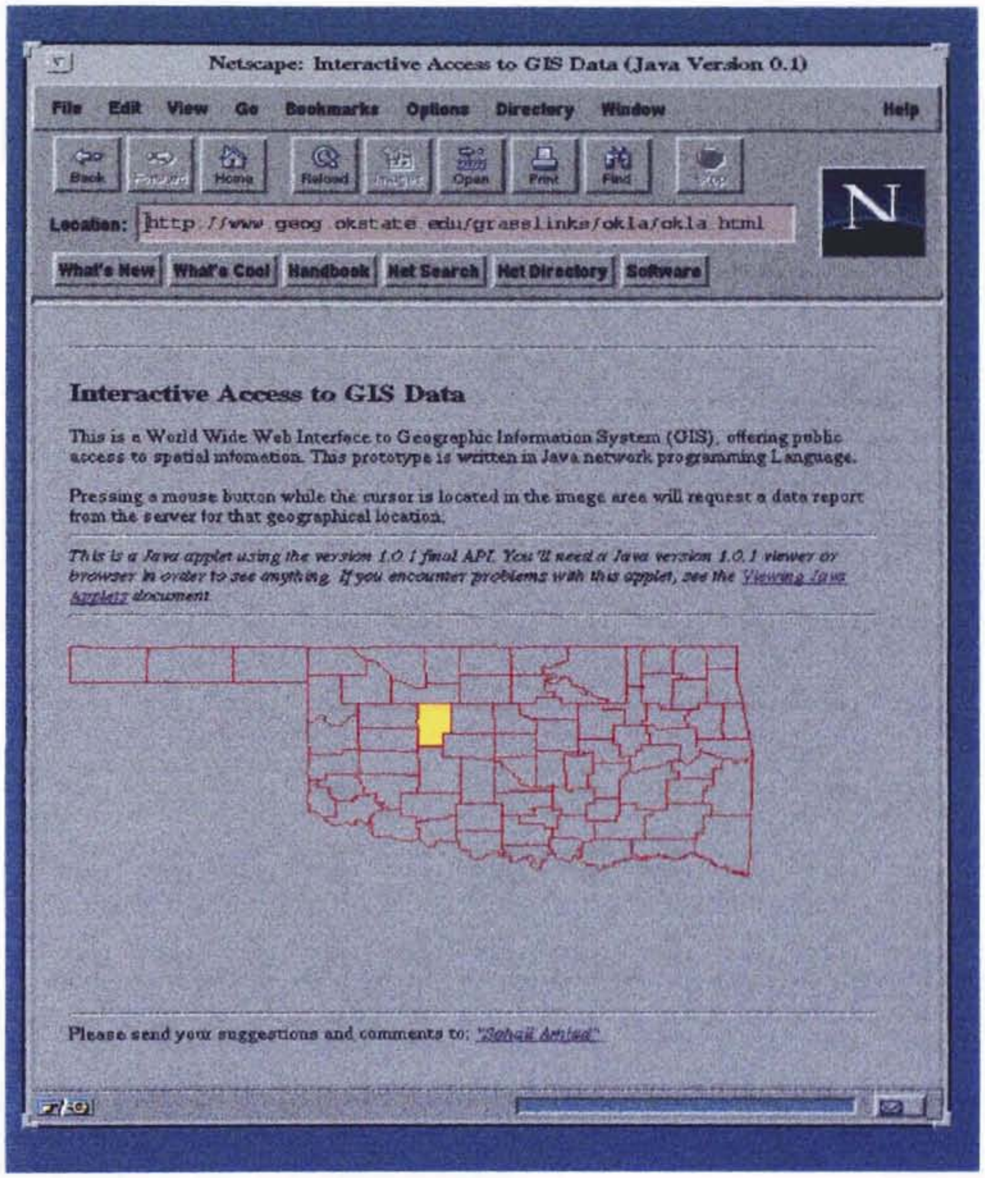
County Class

This class inherits properties from parent Frame class provided in java.lang.awt package. The user can selecte a county from given lists of counties in the Oklahoma State. Clicking on the "Query Server" causes the document of the selected county be loaded on the client browser. Where user can choose data and maps available in that particular county

Raster Class

This class creates a frame through which user can select available raster, vector or sites

OKLAHOMA STATE UNIVERSITY


Netscape: Interactive Access to GIS Data (Java Version 0.1)
File Edit View Go Bookmarks Options Directory Window Help
Back Forward Home Reload Images Open Print Find Stop
Location: http://www.geog.okstate.edu/grasslinks/okla/okla.html
What's New What's Cool Handbook Net Search Net Directory Software

Interactive Access to GIS Data

This is a World Wide Web Interface to Geographic Information System (GIS), offering public access to spatial information. This prototype is written in Java network programming Language.

Pressing a mouse button while the cursor is located in the image area will request a data report from the server for that geographical location.

This is a Java applet using the version 1.0.1 final API. You'll need a Java version 1.0.1 viewer or browser in order to see anything. If you encounter problems with this applet, see the Viewing Java Applets document.

Please send your suggestions and comments to: "Sohail Ahmad"

Fig 4.1 Session Running Client Side Image Map Program.

OKLAHOMA STATE UNIVERSITY

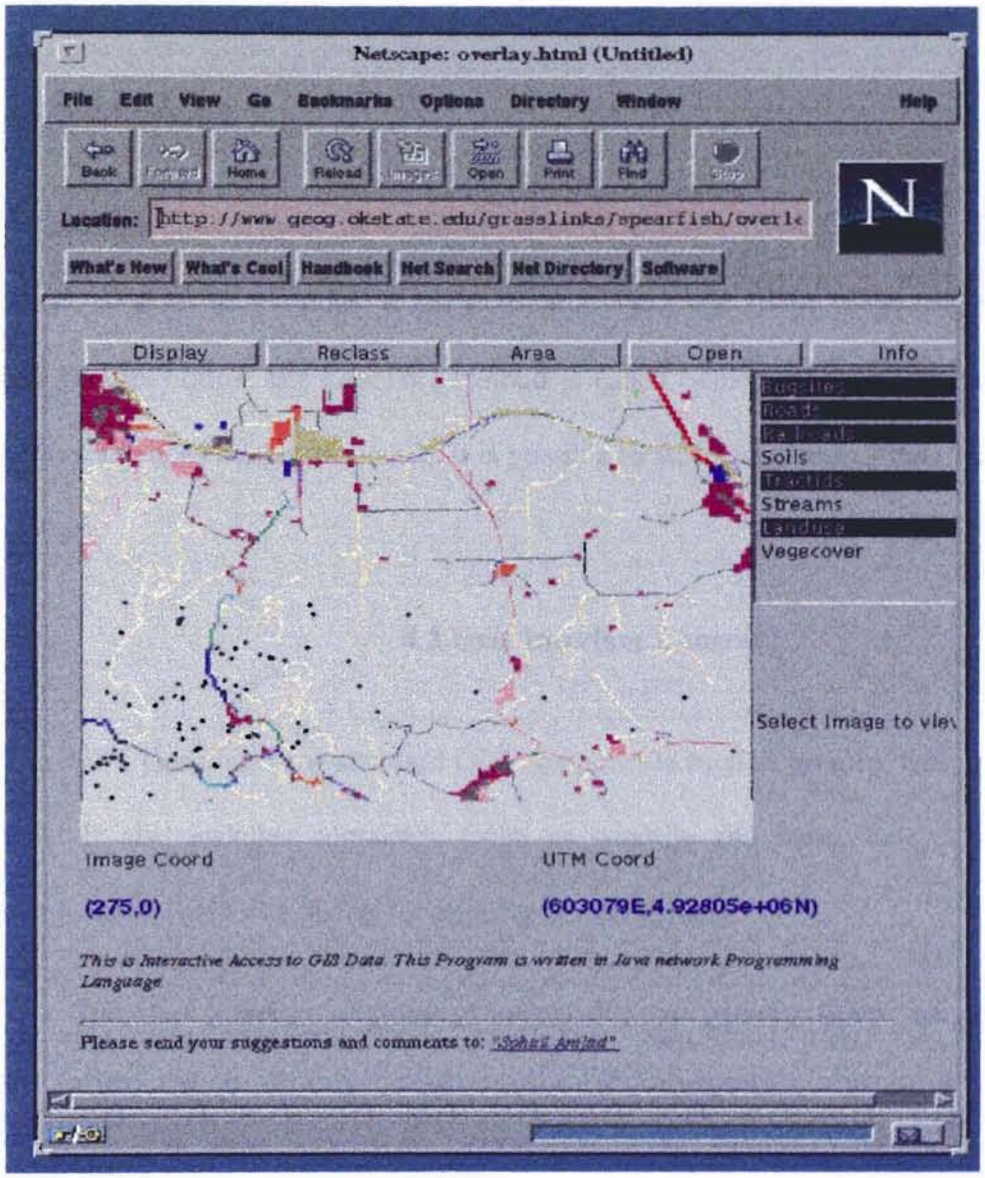

Netscape: overlay.html (Untitled)
File Edit View Go Bookmarks Options Directory Window Help
Back Forward Home Reload Images Open Print Find Stop
Location: http://www.geog.okstate.edu/grasslinks/spearfish/overl
What's New What's Cool Handbook Net Search Net Directory Software
Display Reclass Area Open Info
Bugsites
Roads
Railroads
Soils
Tractids
Streams
Landuse
Vegecover
Select Image to vie

Image Coord

(275,0)

UTM Coord

(603079E,4.92805e+06N)

This is Interactive Access to GIS Data. This Program is written in Java network Programming Language

Please send your suggestions and comments to: "Sohail Amjad"

Fig 4.2 Snapshot Showing UTM Coordinates and Overlaying of Different Themes.

maps to be displayed on the client machine. The method PostData() of this class assemble selected items into a data block exactly the same as POST method does to send data to the http server.

BookMark Class

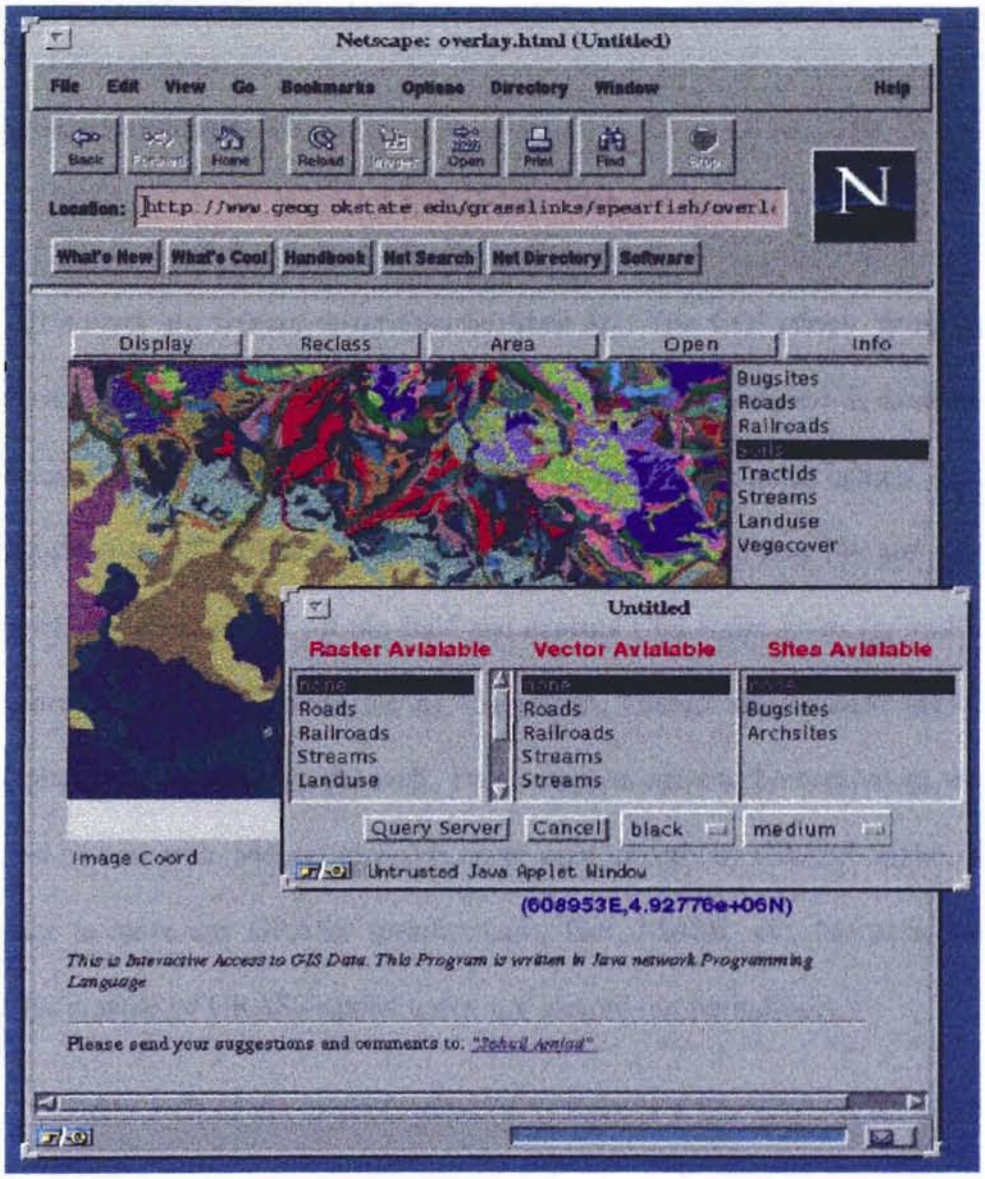
BookMark class creates the Uniform Resource Locator (URL) objects of the documents where counties data resides. It has two parts: name and URL. When user selects county and presses button, the LinkTo() method is called, which tells the browser to load the URL referenced by that Boomark. It is possible to store each county data on different http server.

4.2 User Interface Control

The AWT provides many standard GUI components such as buttons, lists, menus and text area. It also includes containers (such as window and menu bars) and higher-level components (such as a dialog for opening or saving files).

The User Interface is provided mainly through separate Frame window. All action are carried out by clicking buttons on the GRASS-applet a separate window pops up whenever one these buttons are presses for further action to be taken. For example clicking on "Display" button pops up new window on the client machine (see Fig 4.2). The user has given the option to select one raster layer, multiple vector layers and one site layer. The selected data layers along with the location name are posted when user click on query server button which in turn runs CGI scripts on the server. Since we cannot submit form data in the applet a separate function is written which mimic what a WWW browser

OKLAHOMA STATE UNIVERSITY


Netscape: overlay.html (Untitled)
File Edit View Go Bookmarks Options Directory Window Help
Back Forward Home Reload Images Open Print Find Stop
Location: http://www.geog.okstate.edu/grasslinks/spearfish/overl
What's New What's Cool Handbook Net Search Net Directory Software
Display Reclass Area Open Info
Bugsites
Roads
Railroads
Soils
Tractids
Streams
Landuse
Vegecover
Untitled
Raster Avialable
Vector Avialable
Sites Avialable
none
Roads
Railroads
Streams
Landuse
none
Roads
Railroads
Streams
Streams
none
Bugsites
Archsites
Query Server
Cancel
black
medium
Image Coord
Untrusted Java Applet Window

(608953E,4.92776e+06N)

This is Interactive Access to GIS Data. This Program is written in Java network Programming Language

Please send your suggestions and comments to: "Sohail Amjad"

Fig 4.3 Snapshot Showing Dialog Box for Selecting Raster, Vector or Site Layers.

OKLAHOMA STATE UNIVERSITY

does to send data using POST method. The technique is straightforward, open a socket to the server and send data with header of the form:

```
POST cgi-script HTTP/1.0
Content-type:
Content-length:
```

4.3 CGI-Interface to GRASS

CGI scripts are written using Bourne Shell [3]. The CGI script "grass.cgi" is invoked when GRASS-applet send data to the server. Program *postquery* is used to encodes data receives from the client and to executes scripts to generate images on-the-fly i.e., if requested map's gif file already exits then the applet load that file and add it to the list panel, otherwise it fork off the child and execute CGI script *grass.cgi* with the user name determine by the http.conf file on the server The six maps called grass0....grass6 are implemented, all own by the web. These mapsets need to be created so that they can be used by user web. Mapset grass0 is never used directly by GRASS-applet, but serves as a place to store any GRASS specific maps, files, MASK, etc. Mapsets grass (0-5) have maps created by GRASS-applet users, are cleared out periodically

Within each of the mapsets grass (0-5) there is a file "UNLOCK" which is moved to "LOCK" to prevent concurrent use of mapsets. This file can an empty file created by *touch.*

When the grass.cgi starts GRASS it checks in the home directory of the appropriate user name (e.g., web) for the .grassrc (1-5) file. The home directory variable for GRASS-applet is set in grass.cmd file. All .grassrc (1-5) have LOCATION, MAPSET, GISBASE,

OKLAHOMA STATE UNIVERSITY

LOCATION_NAME, GISDBASE, PAINTER, and MAPLP variable set, corresponding to that mapsets (1-5).

The selected data layers and corresponding map compositions are sent to another file rast2gif which process arguments and develop a script file that can be redirected into *p.map.new*, the map making program of GRASS. By using NetPBM (Extended Portable Bitmap Toolkit), the map file is converted into (Portable Pix Map) PPM raw format. After converting into PPM raw format using a data conversion software, *ppmtogif*, the PPM file is converted to transparent (Graphical Interchange Format) GIF file. The applet finally add the file name into list panel. Selecting an item in the list panel displays the area overlayed on the base image on the image canvas.

OKLAHOMA STATE UNIVERSITY

CHAPTER 5

PERFORMANCE

The metrics used for the performance analysis includes:

- The utilization of CPU on the client-server machine.
- The variation of CPU utilization on the Server with increasing number of clients.

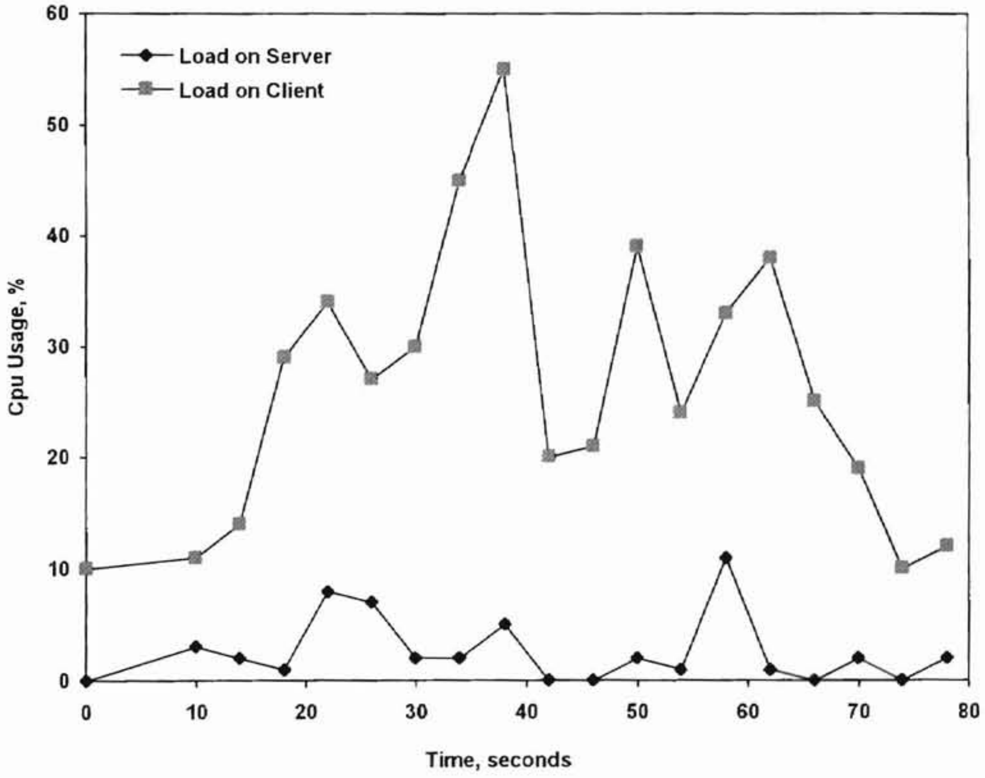


Figure 5.1 Comparison of CPU Usage on Client-Server Machines

OKLAHOMA STATE UNIVERSITY

CPU usage was measured by using Performance Meter (perfmeter), tool for monitoring various system parameter provided by SunOS Solaris 2.4.

Fig 5.1 shows the comparison of CPU utilization on the client/server machines. From the graph it is evident that the program developed uses CPU cycles on the client machine rather than on server's. After distributing codes on the client machine. The processing involved in calculating UTM coordinates, overlaying of maps and other user interaction part is done on the local machine in real time.

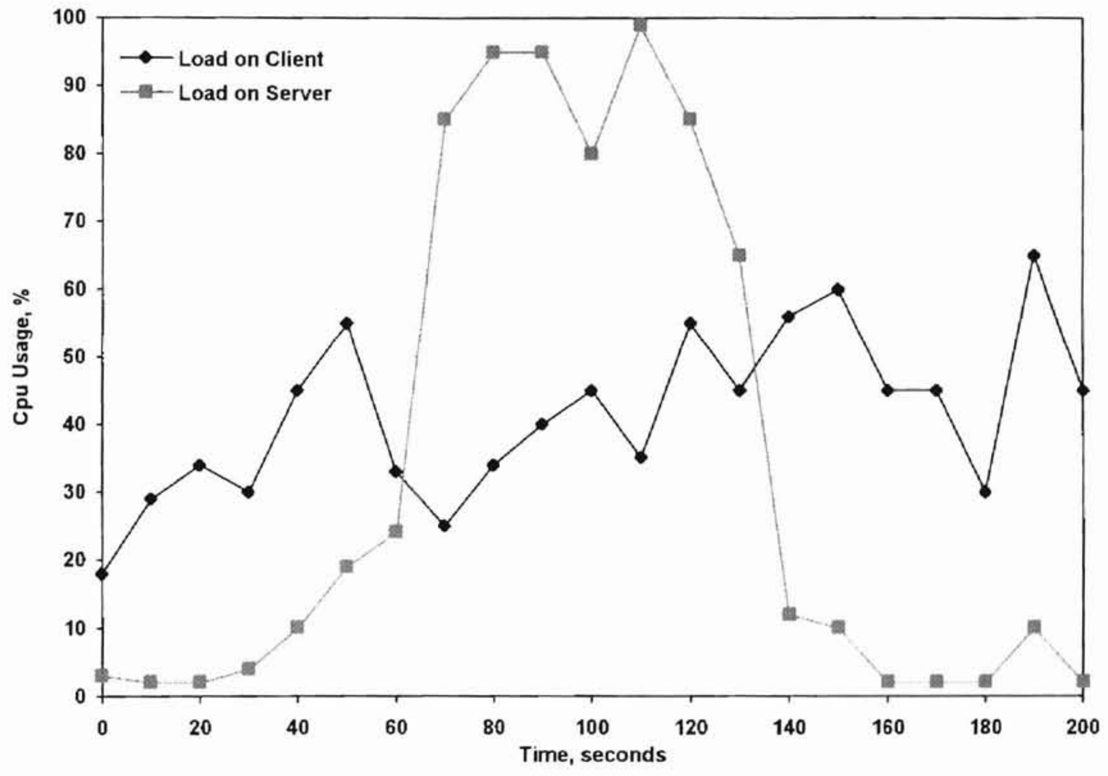


Fig 5.2 CPU Load Characteristics on Client-Server Machine

If the requested map already available, the server provides the requested map on-the-fly to the client, i.e., without generating a new map. otherwise it has to create new map

OKLAHOMA STATE UNIVERSITY

this is shown in Fig 5.2. The CPU usage on the server increase only if the requested map does not exist. This is due to running CGI scripts and generating map which requires resources on the server side. Approximately 60 seconds are required to create new map.

Fig 5.3 shows the CPU usage on the server with increasing number of client. The program was tested to run from the different machine simultaneously. The speed of machines used were different. Machine *neptune* was the fastest machine used in this analysis, therefore it has lower CPU usage as compared to other client machines.

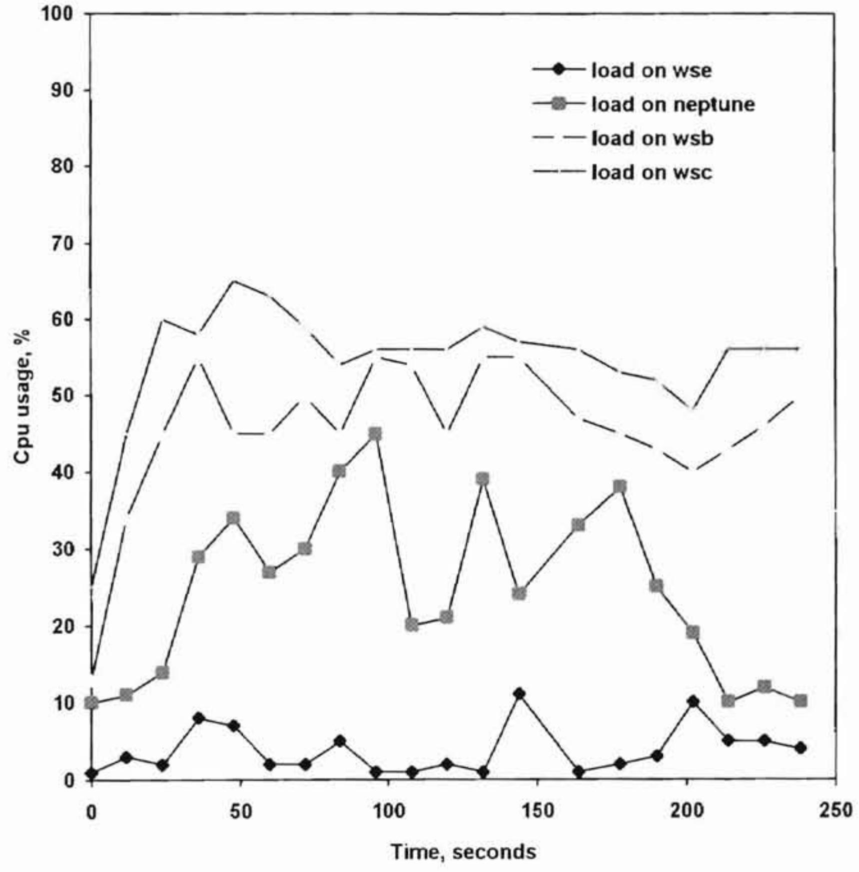


Figure 5.3 Variation of Load on Server with Increasing Number of Clients

OKLAHOMA STATE UNIVERSITY

In this study CPU usage on the server is not significantly affected by the increasing number of clients simultaneously. This feature help in reducing load on the server as number of clients increases and optimize load balancing of client-server resources to provide better performance of distributed programs.

OKLAHOMA STATE UNIVERSITY

CHAPTER 6

CONCLUSION

5.1 Summary

GRASS-applet interactive graphical interface to the GRASS GIS was made available via WWW using CGI and Java network programming language. The work presented here has convinced us that Web-based applets are an ideal solution for presenting our GIS data to the community.

The distributed computing client/server paradigm is showing limitation, in particular in light of the extraordinary growth of the Internet. We have seen from the previous chapter the client/server model can be advantageously extended with mobile code technology. Mobile code makes it possible to reduce the network traffic, to optimize load balancing of the client-server resources, and to provide better performance.

Mobile code also offers a new possibility for software distribution at large, if the technology runs across multiple platforms. Thus, mobile code technology is becoming of increasing importance

Undoubtedly, mobile code technology has additional requirements that make it hardly possible to run with performance of comparable to native code. But, of course, native code is not mobile.

OKLAHOMA STATE UNIVERSITY

The Java mobile code technology as proposed by Sun Microsystems is a good candidate for becoming the mobile code standard in the open systems industry. It offers a simple but powerful programming language. Java programs should be easy to write as programs in other programming languages, but probably easier to maintain and exhibits better defect density. Because of the dynamic loading and security features, Java offers a reasonable base to deploy mobile code over the Internet.

5.2 Future Work

Future work may evaluate additional GIS-related application via WWW. A similar interface can be developed using other GIS software, such as ARC/INFO, by taking advantage of mobile code technology to move processing from the server side to the client side in order to use improve performance, and the integration of heterogeneous, distributed GIS methods and data.

OKLAHOMA STATE UNIVERSITY

BIBLIOGRAPHY

1. Berners-Lee, T , 1994a. HTTP: A protocols for networked information, Internet Draft. Internet Engineering Task Force.

2. Berners-Lee, T. 1994b. Uniform Resource Locator, a syntax for the expression of access information of objects on the network, Internet Draft. Internet Engineering Task Force. W3

3. Behrens, C., Charles, C.S., Geosight Project. Technical report, 1995.

4. Berners-Lee, T. and Connolly, D.W. 1995. Hypertext markup language - 2.0, Internet Draft. Internet Engineering Task Force. W3 Consortium and MIT Laboratory for Computer Science, 545 Technology Square Cambridge, Massachusetts

5. Comer, D.E. 1995. The Internet Book, Prentice Hall, Englewood Cliffs, N.J

6. Duncan, R., 1991. Power Programming: A look at Difference Between C and C++. PC Magazine (July): 444.

7. ERIN. Environmental Research Institute (ERI) Australia. Technical report, 1995

8. GRASS 4.1 User's Menu, Engineers Construction Engineering Research Laboratory (Unpub.), July, 1991.

9. GRASS 4.1 Programmer's Menu. Engineers Construction Engineering Research Laboratory (Unpub.) August, 1992.

10 Gardels, K. 1993. What is open GIS?, GRASS CLIPINGS: The Journal of Open Geographic Information Systems 7(1):40.

11. Henning, R. Mayer-Foll, M. Muller, E. Schmid, H.Spandl. Projekt GLOBUS-Konzeption und prototypische Realisierung einer aktiven Auskunftskompponente fur globale Umwelt-Sachdaten im Umwelt informations system Baden-Wurttemberg, Phase II-1995.

12. Huse, S. Grasslinks (A system based on the PD GIS GRASS). Technical report, 1995.

13. Hunt, C., 1992. TCP/IP Network Administration, A Nutshell Handbook, May 1994 edn, O'Reilly & Associates, Inc. Sebastopol, Calif.

OKLAHOMA STATE UNIVERSITY

14. Illinois State Museum. Faunmap (base on ARC/INFO data). Technical report, 1995.

15. McCool, R., 1995. The common gateway interface, Software available from National Center for Supercomputing Applications at University of Illinois in Urbana-Champaign.

16. Netscape Communication Corporation 1995. Welcome to Netscape, Software available from Netscape Communication Corporation, 501 E. Middlefield Rd., Mountain View, California.

17. NCSA 1995a. NCSA htttp 1.4, Software available from the National Center of Supercomputing Application at the University of Illinois in Urbana-Champaign.

18. NSCA 1995b NCSA mosaic, Software available from the National Center of Supercomputing Application at the University of Illinois in Urbana-Champaign.

19. OGIS,Open GIS Consortium. Technical report, 1995.

20. PBMPLUS, pbmplus manuals and sources. Technical report, 1995.

21. Postel, J. and Reynold, J. 1985. File Transfer Protocols (FTP), Internet RFC-959, Internet Engineering Task Force.

22. Papas, Chris H. and William, H. Murray. 1990. Turbo C++ Professional Handbook. New York: McGraw-Hill, Inc.

23. Ritchey, T., Java 1995, Indianapolis, Indiana: New Riders Publishing. pp. 365.

24. Sun Microsystems Inc., The Java Language Environment: A White Paper, 1995, Sun Microsystems Inc.: Mountain View, CA.

25. Sun Microsystems Inc., The Java Language Overview. 1995, Sun Microsystems, Inc. Mountain View, CA

26. Thoen, B.,WEB-GIS. Technical report, Oct. 1995.

27. Tello, Ernes R. 1991. Object-oriented Programming for Windows. New York: John Willey & Sons, Inc

28. U.S Census Bureau, Tiger Mapping Service TMS.. Technical report, 1995.

OKLAHOMA STATE UNIVERSITY

APPENDICES

OKLAHOMA STATE UNIVERSITY

APPENDIX A

GRASS COMMAND USED IN

THIS PROTOTYPE

OKLAHOMA STATE UNIVERSITY

GRASS COMMANDS USED IN THIS PROTOTYPE

For reference, all the GRASS commands used in the map making script are listed in this appendix. The source for the description of each command is from the GRASS Reference Manual.

g.region	-	Program to manage the boundary.
g.gisenv	-	Outputs the user's current GRASS variable setting.
mon.release	-	Releases the graphics monitor.
p.map	-	Hardcopy color map output utility.
exit	-	Exits the user from the current GRASS.
p.map.new	-	Produces color maps for output on a color hardcopy. Output can include raster map, any number of vector overlays, site data, text labels, and other map elements.
r.what.rast	-	Query the catagory contents of multiple cells in one or more of multiple raster layers.
r.stats	-	Generates area statistics for raster map layers.

OKLAHOMA STATE UNIVERSITY

APPENDIX B

PROGRAM LISTING

```
/*   This is Grass applet main program. It create instance of other classes and  import
java packages used in this program. It extends one of java class Applet found in
java.applet and implment it as a Runnable i.e., Applet start in a different thread to improve
performance of the program.
*/

import      java.awt.*;
import      java.awt.image.*;
import      java.lang.*;
import      java.util.*;
import      java.applet.*;
import      java.net.*;
import      java.io.*;

/*      This class does all the initialization, read files from the server and add
*     component ( canvas, panel, buttons etc) to itself.
*/
public class imageoverlay extends java.applet.Applet implements Runnable {

        int       i1, i2, i3, i4, tempcbint, i;
        String    labelparam, imageparam, tempstring;
        Image     im,

        Graphics offscreen;              //this is used in the double -buffering lines.
        imageoverlaycanvas   icanvas;
        Vector               labels, images, cbvec, utm;
        GridBagLayout        gb;
        GridBagConstraints   gbc;
        Checkbox             tempcb;
        Label                myname, lon, lat,
Thread                  mainthread;
boolean                 tempcbval;
int[]                   listitems;
AboutBox                aboutbox;
List                    itemlist;
County                  county;
String                  line;
DataInputStream         fis, fis1, utmptr;
int                     n, s, e, w;
int                     rows, cols;
URL                     inputfile, inputfile1, utmfile;
InputStream             conn1 = null;
Raster                  raster;
String       url = "http://www.geog.okstate.edu/grasslinks/spearfish/img.txt",
String       url1 = "http://www.geog.okstate.edu/grasslinks/spearfish/labels.txt";
String       url2 = "http://www.geog.okstate.edu/grasslinks/spearfish/utm.dat";

public void    init() {
        try {
                im = createImage((this.size()).width, (this.size()).height);
                //a pplet size is width * height
                offscreen = im.getGraphics();
        }
        catch(Exception e) {
                offscreen = null;
        }
                    icanvas = new imageoverlaycanvas(this);
                    labelparam = getParameter("labels");
                    imageparam = getParameter("images");
                    labels = new Vector(1, 1);
```

```
          images = new Vector(1, 1);
          utm = new Vector(1, 1);
try {

          inputfile = new URL(url);
}
catch (MalformedURLException e) {
          System.out.println("Bad Url : " + inputfile);
}
try {
     InputStream    conn = inputfile.openStream();
     fis = new DataInputStream(new BufferedInputStream(conn));
     while ((line = fis.readLine()) != null) {
          System.out.println(line);
          images.addElement(line);
          System.out.println("Image Size = " + images.size());
     }
}
catch(IOException e) {
          System.out.println("IO Error : " + e.getMessage());
}
try {
     inputfile1 = new URL(url1);
}
catch(MalformedURLException e) {
               System.out.println("Bad Url : " + inputfile1);
}
try {
     InputStream    conn1 = inputfile1.openStream();
     fis1 = new DataInputStream(new BufferedInputStream(conn1)),
        while ((line = fis1.readLine()) != null) {
                System.out.println(line);
                labels.addElement(line);
                System.out.println("Image Size = " + labels.size());
        }
}
catch (IOException e) {
        System.out.println("IO Error : " + e.getMessage());
}
try {
        utmfile = new URL(url2);
}
catch(MalformedURLException e) {
        System.out.println("Bad Url : " + utmfile);
}
try {
        InputStream     utmcon = utmfile.openStream();
        utmptr = new DataInputStream(new BufferedInputStream
                                                        (utmcon));
        while ((line = utmptr.readLine()) != null) {
                System.out.println(line);
                utm.addElement(line);
                System.out.println("Image Size = " + utm.size());
        }
}
catch (IOException e) {
        System.out.println("IO Error : " + e.getMessage());
}
n = Integer.valueOf((String) utm.elementAt(0)).intValue();
s = Integer.valueOf((String) utm.elementAt(1)).intValue();
```

```
e = Integer.valueOf((String) utm.elementAt(2)).intValue();
w = Integer.valueOf((String) utm.elementAt(3)).intValue();
rows = Integer.valueOf((String) utm.elementAt(4)).intValue();
cols = Integer.valueOf((String) utm.elementAt(5)).intValue();
System.out.println(" North : " + n);
System.out.println(" South : " + s);
System.out.println(" East : " + e);
System.out.println(" West : " + w);
System.out.println(" Rows : " + rows);
System.out.println(" Cols : " + cols);
Panel         bottomPanel = new Panel();
Panel         centerPanel = new Panel();
setLayout(new BorderLayout());
bottomPanel.setLayout(new GridLayout(2, 2, 5, 5)),
bottomPanel.add(new Label("Image Coord", Label.LEFT)),
bottomPanel.add(new Label("UTM Coord", Label.LEFT)),
lat = new Label("0", Label.LEFT),
lat.setForeground(Color.blue);
lat.setFont(new Font("Helvetica", Font.BOLD, 14));
bottomPanel.add(lat);
lon = new Label("0", Label.LEFT);
lon.setForeground(Color.blue);
lon.setFont(new Font("Helvetica", Font.BOLD, 14));
bottomPanel.add(lon);
add("South", bottomPanel);
itemlist = new List(3, true);
Panel         p = new Panel();
p.setLayout(new GridLayout(2, 0));
p.add(itemlist);
Label         myname = new Label("Select Image to view"),
        p.add(myname);
        add("East", p);
        add("Center", icanvas);         /* add image canvas to applet */
        Panel       buttons = new Panel();    /* create panel and add button */
        buttons.setLayout(new GridLayout(0, 5));
        buttons.add(new Button("Display"));
        buttons.add(new Button("Reclass"));
        buttons.add(new Button("Area"));
        buttons.add(new Button("Open"));
        buttons.add(new Button("Info"));
        buttons.show();
        add("North", buttons);
        aboutbox = new AboutBox(this);
        add(aboutbox);
        aboutbox.reshape(80, 80, 200, 150);

        for (i1 = 0; i1 < labels.size(); i1++) {
                itemlist.addItem((String) (labels.elementAt(i1)));
        }
        icanvas.setimages(images),
}
/* create thread to load images and watach for user interaction
 */
public  void    start() {
            if (mainthread == null) {
                mainthread = new Thread(this),
                mainthread.start();
            }
}
```

```
public void    update(Graphics g) {
        paint(g);
}

public void    update(int x, int y) {

        lat.setText("(" + x + "," + y + ")");
}
/ * calculate UTM coordinates from the coordinate of image where
    the mouse pointer is currently pointing
 */
public void    utmupdate(int x, int y) {
        float         east, north;

        east = (float) (e - w) * (x - 1) / cols + w;
        north = (float) n - ((n - s) * (y - 1) / rows);
        lon.setText("(" + east + "E," + north + "N)");

  }

 public void    realpaint(Graphics g) {

  }
  /* run thread forever */
  public  void    run() {

         while (true) {
            try {
                        mainthread.sleep(15),  /* take a nap */
                 }
```

```
                catch(Exception e) {
                }
                mainthread.yield();
                for (i4 = 0; i4 < labels.size(); i4++) {
                        tempcbval = ((Checkbox) (cbvec.elementAt(i4))).getState();
                        if (tempcbval == true) {
                                tempcbint = 1;
                        }

                        else {
                                tempcbint = 0;
                        }
                        icanvas.setimagestate(i4, tempcbint),

                }
                /* set image current state in the canvas */
                listitems = itemlist.getSelectedIndexes();
                for (i4 = 0; i4 < labels.size(); i4++) {
                        icanvas.setimagestate(i4, 0);
                }
                for (i4 = 0; i4 < listitems.length; i4++) {
                        icanvas.setimagestate(listitems[i4], 1);
                }
                 /* repaint canvas area all the time */
                icanvas.repaint();
        }
}
/* stop thread if user leave */
public void   stop() {
        if (mainthread != null) {
                mainthread.stop();
                mainthread = null;
        }
}
public void     paint(Graphics g) {

        if (offscreen != null) {
                /* more double  -buffering  */
                     realpaint(offscreen);
                    /* the REAL paint method  */
                     g.drawImage(im, 0, 0, this);
        } else {
                realpaint(g);
                //the REAL paint method
        }
}
/*  handle all the user action clicks. This can be done by overiding
    handle evenet method.
 */
public boolean  handleEvent(Event evt) {
        switch (evt.id) {
                case Event.ACTION_EVENT:
                     if (evt.target instanceof Button) {
                            String   label = ((Button) (evt.target)).getLabel();
                            if (label.equals("Info")) {
                                 aboutbox.show();
                             }
                             else if (label.equals("Open")) {
                                 county = new County(this);
                                 county.pack();
```

```
                                        county.show();
                                    }
                                    else if (label.equals("Display")) {
                                        raster = new Raster(this);
                                        add(raster);
                                        raster.pack();
                                        raster.show();
                                    }
                                }
                                break;
                    default:
                            return super.handleEvent(evt);
                }
                return true;
        }


}
```

```
/*   This class use to create image canvas. It is implement as ImageObserver. simply
wait  until image is constructed
*/
 class  imageoverlaycanvas extends java.awt.Canvas implements ImageObserver {


        Graphics      offs;
        Image         warpedimage, im;
        Graphics      actual;
        Dimension     thisd = new Dimension(200, 200);
        Color         col = new Color(220, 220, 220);
        //back color
```

```
    int             i1, i2, i3, i4;
    Vector          ivec = new Vector(1, 1);
    URL             codeb;
    Applet          parentapp;
    imageoverlay    myapplet;
/* constructer for imageoverlay canvas
 */
    public imageoverlaycanvas ( Applet  parent )  {

        this.parentapp = parent;
        this.myapplet = (imageoverlay) parent;
        try {
            thisd = parentapp.size();
            thisd.width = thisd.width - 150;
            this.resize(thisd);
            im = createImage((this.size()).width, (this.size()).height);
            //applet size is used for teh float buffered image size.
                offs = im.getGraphics();
               //gets the graphics object representation of Image im.
        }
        catch (Exception e) {
                                            //if there 's a problem...
             offs = null,                   //make offscreen null

        }

            codeb = parentapp.getCodeBase(),
        }
        /* paint canvas */
        public void    paint ( Graphics g ) {
```

```
        //this is straightforward, hopefully...
        if ((im == null) || (offs == null)) {
                im = createImage((this.size()).width, (this.size()).height);
                //applet size is used for teh float buffered image size.
                        offs = im.getGraphics();
        }
        if (offs != null) {
                //if Graphics offscreen is not undefined...
                realpaint(offs);
                //paint on it(image of graphics object stored in Image im,
                                                        previously defined)
                g.drawImage(im, 0, 0, this);
                //draw the image of Graphics offscreen
        }
        else {
                realpaint(g);
                //execute the true paint method
        }
}
/* update call paint method */
public void    update(Graphics g) {
        paint(g);
}
/* set images into vector give the abstraction of link list. */

public void   setimages ( Vector  stringvec )  {
        //just a little OOP stuff...
        i1 = 0,
        for (i1 = 0; i1 < stringvec.size(); i1++) {
                try {
                        ivec.addElement(parentapp.getImage(codeb,
                        (String) (stringvec.elementAt(i1))));
                    }
                    catch(Exception e) {
                    }
                    ivec.addElement(new Integer(0));
            }
    }
    /* set image state whether to display or not */
    public void    setimagestate(int image, int state) {
            ivec.setElementAt(new Integer(state), (image * 2) + 1);
    }

    public void   realpaint ( Graphics  g )  {
            g.setColor(col);
            g.fillRect(0, 0, this.size().width, this.size().height);
            for ( i1 = 0; i1 < (int) (ivec.size() / 2); i1++) {
                 if (((Integer) (ivec.elementAt((i1 * 2) + 1))).intValue() ==
                                                                          1) {
                      g.drawImage((Image) (ivec.elementAt(i1 * 2)), 0, 0,
                                                                          null);
                 }
            }
    }
    /* handle event to calculate coordinate as user moves the
       pointer on the image.
     */
    public synchronized  boolean  handleEvent ( Event  evt )  {

            switch (evt.id) {
                    case Event.MOUSE_MOVE:
                        myapplet.update(evt.x, evt.y);
                        myapplet.utmupdate(evt.x, evt.y);
                        return true;
                     default:
                      /* all other event handle by super class */
                        return super.handleEvent(evt);
                 }
        }
 }
/* class to contruct about the box dialog showing name of author
   and help if user click on help button
*/
class  AboutBox extends Panel {

        public   AboutBox ( Panel  parent ) {
                setBackground(new Color(150, 150, 150));

                Label l  =  new Label("GIS Interface", Label.CENTER);
                 l.setForeground(Color.red);
                 l.setFont(new Font("Helvetica", Font.BOLD, 14));
                 add("North", l);
                 l = new Label("Developed By Sohail Amjad", Label.CENTER);
                 l.setForeground(Color.blue);
                 l.setFont(new Font("Helvetica", Font.ITALIC, 10));
                 add("Center", l);
                //add to panel

                 Panel p = new Panel();
                 //p.setLayout(new BorderLayout()),
                p.add(new Button("Ok"));
                p.add(new Button("Help"));
                add("South", p);
                hide();
        }
        public void    paint ( Graphics  g )  {

              Rectangle   bounds  =  bounds();
                   g.setColor(getBackground());
                   g.draw3DRect(0, 0, bounds.width - 1, bounds.height - 1,
                                                                          true);
        }

        public Insets   insets() {
              return new Insets(5, 5, 5, 5);
        }
        /* simply hide dialogbox */
        public boolean  action ( Event  evt, Object  obj ) {
              hide();
              return true;
        }

  }          //end class aboutbox

/ * create a frame in order to choose counties from the list
   of  Oklahoma counties */
 class   County   extends   Frame {

        List          l = new List(5, false);
        String        link;
```

```
Bookmark   blist[] = new Bookmark[3];
Applet        parentapp;

public  County ( Applet parent ) {
        parentapp = parent;
        //super(parent);
        setBackground(new Color(192, 192, 192));
        //setLayout(new GridLayout(1, 1));
        Label      label = new Label("Select County", Label.CENTER);
        label.setForeground(Color.red);
        label.setFont(new Font("Helvetica", Font.BOLD, 14));
        add("North", label);

        l.addItem("Alfalfa");
        l.addItem("Blaine");
        l.addItem("Cimarron");
        l.addItem("Canadian");
        l.addItem("Delware");
        l.addItem("Ellis"),
        l.addItem("Garfield");
        l.addItem("Grant");
        l.addItem("Hughes");
        l.addItem("Johnston");
        l.addItem("Lerflore");
        l.addItem("Latimer");
        l.addItem("Muskogee");
        l.addItem("Oklahoma");
        l.addItem("Wasington");
        add("Center", l);
        Panel          p = new Panel();
```

```
            p.add(new Button("Query Server"));
            p.add(new Button("Cancel"));
            add("South", p);
             hide();
    }

    public void     init() {

    blist[0] = new Bookmark("Alfalfa",
            "http://www.geog.okstate.edu/grass/html/Alfalfa.html");
    blist[1] = new Bookmark("Blaine",
            "http://www.geog.okstate.edu/grass/html/Blaine.html");
    blist[2] = new Bookmark("Cimarron",
            "http://www.geog.okstate.edu/grass/html/Cimmarron.html");
    blist[3] = new Bookmark("Canadian",
                    "http://www.geog.okstate.edu/grass/html/Canadian.html");
    blist[4] = new Bookmark("Delware",
                    "http://www.geog.okstate.edu/grass/html/Delware.html");
    blist[5] = new Bookmark("Ellis",
            "http://www.geog.okstate.edu/grass/html/Ellis.html");
    blist[6] = new Bookmark("Garfield",
                    "http://www.geog.okstate.edu/grass/html/Garfield.html");
    blist[7] = new Bookmark("Grant",
                    "http://www.geog.okstate.edu/grass/html/Grant.html");
    blist[8] = new Bookmark("Hughes",
                    "http://www.geog.okstate.edu/grass/html/Hughes.html");
    blist[9] = new Bookmark("Johnston",
                    "http://www.geog.okstate.edu/grass/html/Johnston.html");
    blist[10] = new Bookmark("Lerflore",
                    "http://www.geog.okstate.edu/grass/html/Lerflore.html");
```

```
blist[11] = new Bookmark("Latimer",
            "http://www.geog.okstate.edu/grass/html/Latimer.html");
blist[12] = new Bookmark("Muskogee",
            "http://www.geog.okstate.edu/grass/html/Muskogee.html");
blist[13] = new Bookmark("Oklahoma",
            "http://www.geog.okstate.edu/grass/html/Oklahoma.html");
blist[14] = new Bookmark("Wasington",
            "http://www.geog.okstate.edu/grass/html/Wasington.html");


}


public boolean  handleEvent(Event evt) {

      switch (evt.id) {
         case Event.ACTION_EVENT:
            if ( evt.target instanceof Button ) {
               String  label = ((Button) (evt.target)).getLabel(),
               if ( label.equals("Cancel") )
                     hide();
               /* link to the page where county data exits */
               else if ( label.equals("Query Server")) {
                     link = l.getSelectedItem();
                     link += ".html";
                      System.out.println(link);
                      LinkTo(link);

               }
            }
            break;
         default:
            return super.handleEvent(evt);
```

```
                }
                        return true;
        }

        /* link to selected counties page */
        public void     LinkTo(String name) {

                URL  theURL = null;
                 init();
                for ( int i = 0; i < blist.length; I++ ) {
                        if (name.equals(blist[i].name)) {
                                theURL = blist[i].url;
                        }
                 }
                  /* get the applet context and show document */
                 if ( theURL != null )
                        System.out.println("now loading: " + theURL),
                        parentapp.getAppletContext().showDocument(theURL);
        }

}

/* conturcter for creating  URL object. */
 class Bookmark {

        String          name;
        URL             url;

         Bookmark(String name, String theURL) {
                this.name = name;
                try {
                        this.url = new URL ( theURL );
                }
                catch(MalformedURLException e) {
                        System.out.println("Bad URL:" + theURL);
                }
        }
}
/*    create frame and add all available raster, vector and sites maps available
   to the counties
 */
class Raster extends Frame {

     List              l = new List(5, false);
     Applet            parentapp;
     private String    item;
     private String    raster = "ANALYSIS=displayer1&RASTER=";
     private String    vector = "VECTOR=";
     private String    vcolor = "VCOLOR=";
     private String     sites = "SITES=";
     private String     scolor = "SCOLOR=black&",
     private String     region = "REGION=12 County Bay and Delta Area&";
     private String            gif_size = "GIF_SIZE=";
     private final String     script = "/cgi-bin/myquery";
     private final String     ctype = "application/x-www-form-urlencoded";
     private String  sdata =
"VECTOR=none&VCOLOR=black&SITES=none&SCOLOR=black&REGION=
     12 County Bay and Delta Area&GIF_SIZE=medium";
     private String    rdata = "";
     private String    home,
private int        port;
Socket             sock;
OutputStream    outp;
InputStream       inp;
DataOutputStream    dataout;
DataInputStream       datain;
List  l1  =  new  List (5, false);
List       l2 = new List(5, false);
List       l3 = new List(5, false);
Choice   c, size;
String    color;
/* constructer for frame Raster */
public   Raster ( Applet parent ) {
        parentapp = parent;
        setBackground(new Color(192, 192, 192));
        setLayout(new GridLayout(1, 1));
        Label  label = new Label("Raster Avialable", Label.CENTER);
        label.setForeground(Color.red);
        label.setFont(new Font("Helvetica", Font.BOLD, 14));
        add(label);
         //adding raster files
         Panel   bottomPanel =  new  Panel();
         Panel   centerPanel   =  new  Panel();
         setLayout(new BorderLayout());
         bottomPanel.add(new Button("Query Server"));
         bottomPanel.add(new Button("Cancel"));
         c = new Choice();
         c.addItem("black");
         c.addItem("white");
         c.addItem("red");
```

```
 c.addItem("orange");
 c.addItem("blue");
 c.addItem("indigo");
 c.addItem("brown");
 c.addItem("yellow");
 c.addItem("green");
 bottomPanel.add(c);
 size = new Choice();
 size.addItem("maximum");
 size.addItem("medium");
 size.addItem("minimum");
 size.select("medium");
 bottomPanel.add(size);
 add("South", bottomPanel);
 centerPanel.setLayout(new GridLayout(1, 3));
  /* add available raster maps */
Panel       p1 = new Panel();
p1.setLayout(new BorderLayout());
label = new Label("Raster Avialable", Label.CENTER);
label.setForeground(Color.red);
label.setFont(new Font("Helvetica", Font.BOLD, 14));
p1.add("North", label);
l1 = new List(5, false);
l1.addItem("none");
l1.addItem("Roads");
l1.addItem("Railroads");
l1.addItem("Streams");
l1.addItem("Landuse");
l1.addItem("Tractids");
l1.addItem("Vegecover");
```

```
l1.addItem("Soils");
l1.addItem("Geology");
l1.select(0);
p1.add("Center", l1);
centerPanel.add(p1);    /* add this to center of panel */
Panel         p2 = new Panel();
p2.setLayout(new BorderLayout()),
/* add all availlable vector to the frame lists */
 label = new Label("Vector Avialable", Label.CENTER);
label.setForeground(Color.red);
label.setFont(new Font("Helvetica", Font.BOLD, 14));
p2.add("North", label);
l2 = new List(5, false);
l2.addItem("none");
l2.addItem("Roads");
l2.addItem("Railroads"),
l2.addItem("Streams"),
l2.addItem("Streams");
l2.select(0);
p2.add("Center", l2);
centerPanel.add(p2);
Panel         p3 = new Panel();
p3.setLayout(new BorderLayout()),
label = new Label("Sites Avialable", Label.CENTER);
label.setForeground(Color.red),
label.setFont(new Font("Helvetica", Font.BOLD, 14)),
p3.add("North", label);
l3  =  new  List ( 5, false );
l3.addItem("none"),
l3.addItem("Bugsites");
```

```
        l3.addItem("Archsites");
        l3.select(0);

        p3.add("Center", l3);
        centerPanel.add(p3);
        add("Center", centerPanel);
        hide();
}
/*    create data block as other browser use to send
     data to the http servers. using method POST.
  */
public boolean  handleEvent ( Event  evt )  {
        switch (evt.id) {
           case Event.ACTION_EVENT:
                if ( evt.target instanceof Button) {
                     String   label = ((Button) (evt.target)).getLabel();
                      if ( label.equals("Cancel"))
                            hide();
                       else if ( label.equals("Query Server")) {
                            item = l1.getSelectedItem(),
                            raster += item,
                            raster += "&";
                            item = l2.getSelectedItem();
                            vector += item;
                            vector += "&";
                            raster += vector;
                            item = c.getSelectedItem();
                            vcolor += item;
                            vcolor += "&";
                            raster += vcolor;
```

```
                                        item = l3.getSelectedItem();
                                        sites += item;
                                        sites += "&";
                                        sites += scolor;
                                        raster += sites;
                                        raster += region;
                                        item = size.getSelectedItem();
                                        gif_size += item;
                                        raster += gif_size;
                                        System.out.println(raster);
                                        hide();
                                        PostData(raster);   /* send data to server */

        /*  server updates files, therefore load document again */
        parentapp.getAppletContext().showDocument(parentapp.getDocumentBase());
                        }
                    }
                    break;
            default:
                            return super.handleEvent(evt);
        }
                                return true;
  }

/*      This function send data to client similar to what other browser do to
    send data to server.
  public void     PostData(String data) {
                /* get the port where http is ruuning on */
                home = parentapp.getDocumentBase().getHost();
                port = parentapp.getDocumentBase().getPort();
```

```
if (  port == -1 )
    port = 80;


 /* open up a scoket */
   try {
         sock = new Socket(home, port);
  }
  catch ( Exception  e ) {
         rdata = e + " (socket: Host: " + home + "\tPort: " + port + ")";
                 return;
   }
   try  {
         /* get the output stream to put data on */
          outp = sock.getOutputStream();
         inp = sock.getInputStream();    /* get input stream to read data */
    }
   catch ( Exception  e ) {
            rdata = e + " (getstream)";
    }
        /* if error occur close socket */
   try  {
          sock.close();
          catch(IOException ee);
          return;
    }
        /* create objects for Input/Output Data stream */
   try {
         dataout = new DataOutputStream(outp);
         datain = new DataInputStream(inp);
     }
```

```
catch ( Exception  e ) {
        rdata = e + " (Dstream)";
   }
   /* close scoket if error occur in opening input/output stream */
    try {
        sock.close();
        catch(IOException ee),
        return;
    }

    //send http stuff
    try
    {
          dataout.writeBytes("POST " + script + " HTTP/1.0\r\n");
          dataout.writeBytes("Content-type: " + ctype + "\r\n");
          dataout.writeBytes("Content-length: " + data.length() + "\r\n");
          dataout.writeBytes("\r\n");
          System.out.println(data);
          dataout.writeBytes(data);
          dataout.writeBytes("\r\n");
          boolean        body = false;
          String         line;
          while (( line = datain.readLine()) != null ) {
                  if ( body )
                      rdata += "\n" + line;
                      else if (line.equals(""))
                              body = true;
          }
    }
    catch(Exception e) {
```

```
                rdata = e + " (write)";
                 try
                 sock.close();
                 catch(Exception ee);
                 return;
        }

        //close socket
        try
        {
              dataout.close();
              datain.close();
        }
        catch(IOException e);
        try
           sock.close(),
        catch(IOException e),
}


public void   paint ( Graphics g ) {

        StringTokenizer st = new StringTokenizer(rdata, "\n");
        int     line = 1,
        int     line_sp = getFontMetrics(getFont()).getHeight() + 1;

        while ( st.hasMoreTokens())  {

                g.drawString(st.nextToken(), 5, line * line_sp),
                        }
                    }




              }
```

APPENDIX C

CGI SCRIPT

```
#!/bin/sh
##########################################################################
#                                                                        #
#                                                                        #
#              File Name      .   grass.cmd                              #
#                                                                        #
#                                                                        #
#   This scripts setup path for home directory of http user and setup environment   #
#       variable   for the GRASS.                                        #
#                                                                        #
#                                                                        #
##########################################################################

#: ${GISBASE?}
trap '/bin/mv -f $LOCATION/LOCK $LOCATION/UNLOCK' 0 1 2 3 5 9 15

GISBASE=/dsk3/grass
export GISBASE
home=/dsk3/home/http
GISRC=$home/.grassrc
export GISRC
GISDBASE=/dsk3/data
export GISDBASE
LOCATION_NAME=spearfish
export LOCATION_NAME
GIS_LOCK=$$
export GIS_LOCK
ETC=$GISBASE/etc
PATH=$GISBASE/bin:$GISBASE/scripts:$GISBASE/garden/bin:$PATH
export PATH
```

```
#eval `SHELL=/bin/sh tset -s -Q`
#stty -tabs

#g.gisenv MONITOR=
#eval `g.gisenv`

############################################
# step through available grass mapsets.
locnumber=1
location=$GISDBASE/$LOCATION_NAME/grass
while test $locnumber -lt 10
do
            if mv -f $location$locnumber/UNLOCK $location$locnumber/LOCK
    then
                    touch $location$locnumber/LOCK
                    mapset=grass$locnumber
      location=$GISDBASE/$LOCATION_NAME/$mapset
                    gisrc=$home/.grassrc$locnumber
                    break
    else
      locnumber=`expr $locnumber + 1`

      if [ $locnumber = 6 ]
      then
                            sleep 5
              locnumber=1
      fi
    fi
done
```

```
############################################
# reset GRASS environment variables

location=$GISDBASE/$LOCATION_NAME/$mapset

GISRC=$gisrc
export GISRC
LOCATION=$location
export LOCATION
MAPSET=$mapset
export MAPSET
g.gisenv GISDBASE=/dsk3/data
g.gisenv LOCATION_NAME=spearfish
g.gisenv LOCATION=$LOCATION
g.gisenv MAPSET=$MAPSET
eval `g.gisenv`


############################################
# Run the GRASS command
#sh="`basename ${SHELL=/bin/sh}`"

        $*



############################################
# Exit GRASS

monitor=`g.gisenv MONITOR`
```

```
if [ "$monitor" ]
then
        $ETC/mon.release -v $monitor
fi
g.gisenv MONITOR=

eval `g.gisenv`
LOCATION=${GISDBASE?}/${LOCATION_NAME?}/${MAPSET?}
```

```
#!/bin/sh
#           File Name  :   grass.cgi
#
#       This cgi script use to set up some path and runs the appropriate scripts.
#
trap "rm -f /export/wohler/tmp_grass/posted.$$; exit 1"  0 1 2 3 5 9 15

PROG_PATH=/dsk2/httpd/htdocs/grass/spearfish/scripts
TMP_PATH=/tmp/tmp_grass
GIF_PATH=/dsk2/httpd/htdocs/grass/spearfish/scripts
GISBASE=/dsk3/grass
PBM_PATH=/dsk2/httpd/htdocs/grass/spearfish/scripts
export PROG_PATH
export TMP_PATH
export GISBASE
export PBM_PATH
export GIF_PATH


ANALYSIS=`grep "ANALYSIS =" $TMP_PATH/posted.dat | sed 's/ANALYSIS = //'`
echo $ANALYSIS
$PROG_PATH/timer 300 $PROG_PATH/grass.cmd $PROG_PATH/$ANALYSIS
$TMP_PATH/posted.dat

if  [ $? -eq 137 ]
then
        echo "The process you requested has timed out.<p>
        This may have been because our computer is overloaded,
        but it could also be because your request was incomplete
        or specified a non-terminating query.<p>
        Please check that all the entries in the query form are set
        correctly before resubmitting.<p>"
fi
```

```
#!/bin/sh
##########################################################################
#                                                                        #
#                                                                        #
#                  File Name :          display.cmd                      #
#                                                                        #
# Starts a GRASS  and run rast2gif script for the  selected map.Also calculate the   #
#  UTM coordinates of the selected map.                                  #
#                                                                        #
##########################################################################


RASTER=`grep "RASTER =" $1 | sed 's/RASTER = //'`
   GRAST1=`echo $RASTER | sed 's/ /+/g'`
        GRASTER=`$PROG_PATH/parser_rast $GRAST1`
REGION=`grep "REGION =" $1 | sed 's/REGION = //'`
        GREG1=`echo $REGION | sed 's/ /+/g'`
        GREGION=`$PROG_PATH/parser_reg $GREG1`
GIF_SIZE=`grep "GIF_SIZE =" $1 | sed 's/GIF_SIZE = //' | awk '{print $1}'`
VECTOR=`grep "VECTOR =" $1 | sed 's/VECTOR = //'`
        GVECT1=`echo $VECTOR | sed 's/ /+/g'`
        GVECTOR=`$PROG_PATH/parser_vect $GVECT1`
SITES=`grep "SITES =" $1 | sed 's/SITES = //'`
        GSITE1=`echo $SITES | sed 's/ /+/g'`
        GSITES=`$PROG_PATH/parser_sites $GSITE1`
VCOLOR=`grep "VCOLOR =" $1 | sed 's/VCOLOR = //'`
SCOLOR=`grep "SCOLOR =" $1 | sed 's/SCOLOR = //'`


if [ $RASTER != "none" ]
```

```
then
  GIF=$RASTER.gif
fi

if [ $VECTOR != "none" ]
then
  GIF=$VECTOR.gif
fi

if [ $SITES != "none" ]
then
  GIF=$SITES.gif
fi


case $GREGION in
        raster)
                g.region rast=$GRASTER
                ;;
        *)
                g.region region=$GREGION
                ;;
esac
g.remove rast=MASK > /dev/null 2>&1

COORDS=`g.region -p`



ZONE=`echo COORDS   | awk '{print $5}'`
```

```
NORTH=`echo $COORDS | awk '{print $7}'`
SOUTH=`echo $COORDS | awk '{print $9}'`
EAST=`echo $COORDS  | awk '{print $11}'`
WEST=`echo $COORDS  | awk '{print $13}'`
GRES=`echo $COORDS  | awk '{print $15}'`
COLS=`echo $COORDS  | awk '{print $19}'`
ROWS=`echo $COORDS  | awk '{print $21}'`


echo "$NORTH" >> $TMP_PATH/utm.$$
echo "$SOUTH" >> $TMP_PATH/utm.$$
echo "$EAST" >> $TMP_PATH/utm.$$
echo "$WEST" >> $TMP_PATH/utm.$$
echo "$COLS" >> $TMP_PATH/utm.$$
echo "$ROWS" >> $TMP_PATH/utm.$$


$PROG_PATH/rast2gif $GIF $GRASTER $GVECTOR $GSITES $VCOLOR
$SCOLOR $GIF_SIZE
```

```
#!/bin/sh
########################################################################
#                                                                      #
#               File Name :      rast2gif                              #
#                                                                      #
# This file use to create transparent gif file. It store varaible in script file and runs    #
# GRASS utility p.map.new to generate map in raw format. Finally it ppmtogif        #
#  used to convert ppm file to gif file.                               #
#                                                                      #
########################################################################

trap "/bin/rm -f $TMP_PATH/grass.$$.ppm"  0 1 2 3 5 9 15

GIF=$1
GRASTER=$2
GVECTOR=$3
GSITES=$4
VCOLOR=$5
SCOLOR=$6
GIF_SIZE=$7
SCRIPT=$TMP_PATH/pmap_script.$$
GIF_PATH=/dsk2/httpd/htdocs/grass/spearfish


touch $SCRIPT
p.select ppm > /dev/null 2>&1

if [ $GRASTER != "none" ]
then
        echo "raster $GRASTER" >> $SCRIPT
```

```
fi

if [ $GVECTOR != "none" ]
then
        echo "vector $GVECTOR" >> $SCRIPT
        echo "    color $VCOLOR" >> $SCRIPT
        echo "    end" >> $SCRIPT
fi

if [ $GSITES != "none" ]
then
        echo "sites $GSITES" >> $SCRIPT
        echo "    color $SCOLOR" >> $SCRIPT
        echo "    end" >> $SCRIPT
fi

echo "end" >> $SCRIPT

p.map.new input=$SCRIPT > /dev/null 2>&1
case $GIF_SIZE in


( $PBM_PATH/pnmscale -xysize $MAX_RES $MAX_RES < $MAPLP |
$PBM_PATH/ppmquant 256 | $PBM_PATH/ppmtogif -transparent white >
$GIF_PATH/$GIF ) 2> /dev/null
```

```
#! /bin/sh
#               File  Name :      parser_ras
#
# convert requested raster map selected in html to full map names

case $1 in

none) MAP="none" ;;

Roads) MAP="roads@PERMANENT" ;,

Railroads) MAP="railroads@PERMANENT" ,;

Streams) MAP="streams@PERMANENT" ;;

Landuse) MAP="landuse@PERMANENT" ;,

Tractids) MAP="tractids@PERMANENT" ;;

Vegetation Cover) MAP="vegcover@PERMANENT" ;;

Soils) MAP="soils@PERMANENT" ;;

Geology)  MAP="geology@PERMENENT" ;;

Restricted Areas) MAP="rstrct@PERMANENT" ;;

Soil  PH  Values) MAP="soils.ph@PERMANENT" ,;

Erosions) MAP="erode@PERMANENT" ;;

Fields in Spearfish) MAP="fields@PERMANENT" ;,

Soils K factor) MAP="soils.Kfactor@PERMANENT" ;,

Soils T factor) MAP="soils.Tfactor@PERMANENT" ,;

Range of Soil) MAP="soils.rang@PERMANENT" ;;

Transportation Misclleneous) MAP="transport.misc@PERMANENT" ;;

USGS Quads) MAP="quads@PERMANENT" ;;

Geology)  MAP="geology@basis" ,;
```

```
Elevation Dimension) MAP="elevation.dted@PERMANENT" ;;

Elevation) MAP="elevation.dem@PERMANENT" ;;

Slope) MAP="slope@PERMANENT" ;;

Elevation) MAP="elevation.dem@PERMANENT" ;;

Texture) MAP="texture@PERMANENT" ;;

Rushmore) MAP=rushmore@PERMANENT" ;;

*) MAP=$1 ;;

esac

echo $MAP
```

```
#! /bin/sh
#          File Name  :  parser_vect
#
#  convert requested map selected in html to full map names


case $1 in

none) MAP="none" ;;

Roads) MAP="t.roads@PERMANENT" ,;

Railroads) MAP="t.railroads@PERMANENT" ;;

Streams) MAP="streams@PERMANENT" ;;

Fields) MAP="fields@PERMANENT" ;;

USGS Quads) MAP="quads@PERMANENT" ;;

Restriction Areas) MAP="rstrct@PERMANENT" ;;

Sections) MAP="sections@PERMANENT" ;;

Soils) MAP="soils@PERMANENT" ;;

Tracts) MAP="t.tracts@PERMANENT" ;;

Roads Information) MAP="t.roads.inf@PERMANENT" ;;

Tractids)  MAP="t.tracts@PERMANENT" ;;

Transport Misclleneous)="transport.misc" ;;

Power Lines) MAP="t.powerlines@PERMANENT" ;;

Hydro) MAP="t.hydros@PERMANENT" ;;

Primary Roads) MAP="t.roads.prime@PERMANENT" ;;

Secondary Roads) MAP="t.roads.second@PERMANENT" ;,

Twp Range) MAP="twp.range@PERMANENT" ;,
esac
```

```
echo $MAP
```

```
#! /bin/sh
#                   File Name  :  parser_sites
#
# This script convert requested map selected in html to full map names

case $1 in

none) MAP="none" ;;

Bugsites)  MAP="bugsites@PERMANENT" ;;
Archsites)  MAP="bugsites@PERMANENT" ;;

esac
echo $MAP
```

```
#!/bin/sh
########################################
#
# Checks for files older than one hour
# and deletes them.
#
########################################

LOCATION_NAME=/dsk3/data/spearfish
PROG_PATH=/dsk2/httpd/htdocs/grass/spearfish/scripts

# clean out GRASS tmp directory
cd /tmp/tmp_grass
/bin/ls |
$PROG_PATH/checktimes |
/bin/xargs /bin/rm -f

# clean out GRASS tmp directories
for i in 1 2 3 4 5
do
        DIR=/dsk3/data/spearfish/grass$i
        cd $DIR/tmp
        /bin/ls |
        $PROG_PATH/checktimes |
        /bin/xargs /bin/rm -rf
done

# clean out grass mapsets by
# deleting all files in the mapset subdirectories, but none
# of the top level files or directories.
```

```
for i in grass1 grass2 grass3 grass4 grass5
do
        (
        cd $LOCATION_NAME/$i
        for i in *
        do
                (
                        if [ -d $i ]
                        then
                                cd $i
                                if [ ! -z "`/bin/ls`" ]
                                then
                                        find `/bin/ls` -print |
                                        $PROG_PATH/checktimes | /bin/xargs /bin/rm -rf
                                fi
                        fi
                )
        done
        )
done
```

```
/*      File  Name :  postquery.c
      This program decodes data submitted by url

*/


#include <stdio.h>
#include <unistd.h>
#include <string.h>
#include <stdlib.h>

#define MAX_ENTRIES 10000
#define PATH2 "/dsk2/httpd/cgi-bin/grass.cgi"
#define PATH3 "/dsk2/httpd/htdocs/grass/spearfish/"

typedef struct {
   char *name;
   char *val;
} entry;

char *makeword(char *line, char stop);
char *fmakeword(FILE *f, char stop, int *len),
char x2c(char *what);
void unescape_url(char *url);
void plustospace(char *str);
FILE *fp, *fptr1, *fptr2;


main(int argc, char *argv[]) {

      entry entries[MAX_ENTRIES];
```

```
  register int x,m=0;
  int cl;
  char filename[100];
  char answer[100];
   int ret_val, status;

if( strcmp(getenv("REQUEST_METHOD"),"POST")) {
      printf("This script should be referenced with a METHOD of POST.\n");
      printf("If you don't understand this, see this ");
      printf("<A HREF=\"http://www.ncsa.uiuc.edu/SDG/Software/Mosaic/Docs/fill-
                              out-forms/overview.html\">forms overview</A>.%c",10);
      exit(1);
  }
  if( strcmp(getenv("CONTENT_TYPE"),"application/x-www-form-urlencoded")) {
      printf("This script can only be used to decode form results. \n");
      exit(1);
   }
   cl = atoi(getenv("CONTENT_LENGTH"));

   for( x=0;cl && (!feof(stdin));x++) {
       m=x;
       entries[x].val = fmakeword(stdin,'&',&cl);
       plustospace(entries[x].val);
       unescape_url(entries[x].val);
        entries[x].name = makeword(entries[x].val,'=');
   }

   fp = fopen("/tmp/tmp_grass/posted.dat","w+");
   for( x=0; x <= m; x++)
        fprintf(fp,"%s = %s%c",entries[x].name,  entries[x].val,10);
```

```
fclose(fp);

fptr1 = fopen("/dsk2/httpd/htdocs/grass/spearfish/img.txt","a");
fptr2 = fopen("/dsk2/httpd/htdocs/grass/spearfish/labels.txt","a");

if( strcmp("none",entries[1].val) ){
    strcpy(answer,entries[1].val);
    sprintf(filename,"%s%s.gif",PATH3,entries[1].val);
}
else
if( strcmp("none",entries[2].val)) {
    strcpy(answer,entries[2].val);
    sprintf(filename,"%s%s.gif",PATH3,entries[2].val);
}
else
if( strcmp("none", entries[4].val) ) {
    strcpy(answer,entries[4].val);
    sprintf(filename,"%s%s.gif",PATH3,entries[4].val);
}

 /*  check if the requested map already exits  */

if( access(filename,F_OK)) {

      fprintf(fptr1,"%s.gif",answer);        /* update files */
      fprintf(fptr2,"%s",answer),
      fprintf(fptr1,"\n");
      fprintf(fptr2,"\n");
      fclose(fptr1);
      fclose(fptr2);
```

```
/* *************************************************************** *
*         File Name :    checktime.c                                *
*                                                                   *
*     This program check the file older than one hour and deletes them from the   *
*          directories where tmp maps are created.                  *
*                                                                   *
*                                                                   *
*****************************************************************/

#define POSIX_4D9
#include <sys/types.h>
#include <sys/stat.h>
#include <sys/timers.h>
#include <stdio.h>
struct stat files_info;
struct timespec current_time;
char buffer[2000];
extern int errno,

main (int argc, char **argv)
{
if (getclock( TIMEOFDAY , &current_time)) {
        fprintf(stderr, "could not get time of day\n"),
        exit (1);
        }
while (gets (buffer) ) {
        if (stat ( buffer, &files_info )) {
                fprintf(stderr, "could not stat %s\n", buffer);
                fprintf(stderr, "error number = %d\n", errno);
                continue;
                }
        /*printf ( "%d\n", current_time.tv_sec - files_info.st_atime); */
        if (current_time.tv_sec - files_info.st_atime > 3600 ) {
                printf ("%s\n", buffer);
                }
        }
}
```

```
/*****************************************************************
*        File Name  :   time.c                                   *
*                                                                *
*         This file check timer approximately 30 seconds. if the map can not be     *
*              made within 30 seconds the process got kill on the server           *
*                                                                *
*****************************************************************/


#include <unistd.h>
#include <signal.h>
#include <stdio.h>

die()
{
        /* killing process 0 really means to kill all process in my */
        /* process group -- see man 2 kill */
        kill(0, SIGKILL);
}

main(argc, argv, envp)  {
        int         argc;
        char        **argv;
        char        **envp;
        int         status;

        /* make my a new process group */
        /* so that only this process and its subprocesses get killed */
        setsid();
        /* set alarm specified in grass.cgi file */

        signal(SIGALRM, die);
        alarm(atoi(argv[1]));

        /*  fork off child to do work */

        if ( fork() ) {
                wait(&status);
                exit(status);
        } else {
                execve(argv[2], argv + 2, envp);
                fprintf(stderr, "path= %s\n", argv[2]);
                fprintf(stderr, "argv= %s\n", *(argv+2));
                fprintf(stderr, "argv+1= %s\n", *(argv+3));
                fprintf(stderr, "envp= %s\n", envp);
                fprintf(stderr, "execve failed\n");
        }
}
```

VITA

Sohail Amjad

Candidate for the Degree of

Master of Science

Thesis: INTERACTIVE ACCESS TO GIS DATA

Major Field: Computer Science

Biographical Data:

Personal Data: Born in Karachi, Pakistan, on April 04, 1969, the son of H. Altaf Hussain and Fahmida Begum.

Education Graduate from Karachi National High School, Karachi, Pakistan;recieved Bachelor of Science in Engineering and Master of Science in Applied Physics from University of Karachi, Karachi, Pakistan in December 89 and May 1992, respectively Completed the requirments for the Master of Science degree with a major in Computer Science at Oklahoma State University in July 1996.

Experience: Research Assistant, Department of Geography, Oklahoma State University, August, 1995 to present Teaching Assistant, Department of Applied Physics, University of Karachi, Karachi, Pakistan, January, 1993 to June 1993.