Internet GIS and UMN Mapserver

Maria Antonia Brovelli
First Internet GISs

The Xerox PARC Map Viewer was created in June 1993 by Steve Putz at Xerox Corporation's Palo Alto Research Center, as an experiment in providing interactive information retrieval via the World Wide Web. (http://www2.parc.com/istl/projects/mapdocs/) At the moment has been deactivated.
Map Viewer: world 46.92N 8.34E (64.1X)

Select a point on the map to zoom in (by 2), or select an option below. Please read About the Map Viewer, FAQ and Details.

Options:

- Zoom In: [2], [5], [10], [25]; Zoom Out: [1/2], [1/5], [1/10], [1/25]
- Features: Default, All, -borders, +rivers
- Display: color; Projection: elliptical, rectangular, sinusoidal; Narrow, Square
- Change Database to USA only (more detail)
- Hide Map Image, No Zoom on Select, Reset All Options

Options can also be typed in as search keywords (e.g. "lon=100", see details). Current region is 5.62 deg. wide by 2.81 deg. (193.89 miles) high.

http://www2.parc.com/istl/projects/www94/mapviewer-example2.html
First Internet GISs

1997

- The University of Minnesota (UMN) released MapServer 1.0, an open source development environment for building spatially-enabled Internet applications (http://mapserver.gis.umn.edu/).

- ESRI released ArcView Internet Map Server (IMS), a proprietary tool for publishing GIS data over the Internet. (http://www.esri.com/software/arcgis/arcims/index.html)
Internet

- ARPANET (70s) – developed by U.S. DoD (Department of Defense): a network that could survive a nuclear war
- 1983: Arpanet adopts TCP/IP (Transmission Control Protocol/Internet Protocol) as communication protocol

Computer Networks

- Two or more computers and other devices (printers, plotters, …) connected by any means (e.g. dedicated cable, telephone line, optical fibers, via satellite).
- The sharing of resources among connected computers is known as “Networking”
TCP/IP

- It’s a combination of two protocols:
  - TCP: rules to exchange information packets over the network
  - IP: rules to define data format and exchange order (“mailing” address system for the machines on a network)
- Its main feature is the support of different network typologies and environments (interoperability). The communication depends on the protocol and not on the computer type or brand, operating system or used software.
Since IP addresses are difficult to be remembered, in 1984 Paul Mackapetris conceived the DNS (Domain Name System), that is a distributed database used by TCP/IP applications to associate symbolic names, also known as domain names, to IP addresses.

A domain name is formed by three or more parts separated by points. An example:

www.polimi.it

- 2nd level domain: agency or company the computer belongs to
- 1st level domain: related to the originating country or, for the U.S.A., the kind of organization
WWW: World Wide Web

- It is a
  - multimedia system (data, sounds, images, videos, …)
  - hypertextual system (documents are organized in mutually linked pages)

for the publication of documents and information on the Internet

- It is constituted by computers that publish pages, which contain documents. These are the web servers (computers executing appropriate software). In order to be able to access such pages it is enough to use a computer connected to the Internet (client), geared with a peculiar software, known as Browser (e.g. Firefox, Internet Explorer, …).
Web Software

Web client software (browser) allows to make requests and read answers

Web server software allows to read requests, to process them and send answers

- IIS
- Sun Java System Web Server
- AOLserver
- Netscape
- Mozilla Firefox

The Apache Software Foundation
http://www.apache.org/
HTTP (HyperText Transfer Protocol): it is a set of rules to exchange resources (documents) between a web server and a web browser. It transports the browser requests to the servers and the requested information from the servers to the browsers.

A HTTP server is a computer with a daemon installed (HTTP Daemon), which waits for requests from clients and answers by sending hypertextual pre-composed documents or other kinds of documents. For each call from a client, HTTP establishes the client-server connection; the server replies the client request and sends the answer. At this point the connection is interrupted and the server doesn’t recall what it sent (“stateless Remote Procedure Call – RPC”).
URI

- The client accesses the server resources by using a URI.
- URI (Uniform Resource Identifier) provides a schema to identify names and locations of web resources; it consists of 4 parts:
  - applicative protocol to be used to access the resources (e.g. http://)
  - domain name: internet address of the computer that contains the resource to be accessed
  - number which points to the web server port needed by the used application to send and receive data. If the web server uses the standard port assigned to the used application, this number can be omitted (e.g. 80 is the standard for HTTP, then if the application uses HTTP as protocol, 80 can be omitted)
  - resource localization: specify the path on the web server file system (e.g. folder/subfolder/…) and the resource name.

http://www.polimi.it:80/corsi/orari.html

HTTP limits

- The web resource the URI points to can be a HTML (HyperText Markup Language) document, an image, a program (e.g. CGI – Common Gateway Interface), a java applet, ...

- By using HTTP alone it is not possible to execute “advanced operations” such as for example selecting a rectangular area on a map, because HTTP does not allow the server to remember the first mouse click (client) once the mouse moves to the second point.
HTML

- HTML (HyperText Markup Language): formatting language which defines the appearance of the document. It contains the text to be published and MARKERS (or TAGS) that define how each part of the document will be visualized.

- Therefore a HTML document is a structured document with different components or “elements” marked with standard tags.

- As each computer receives the marked file, this is displayed in the most proper way according to its characteristics (wrong visualization of the resources is avoided).
TAG and element content

- Each HTML tag starts with a couple of brackets (<>) and ends with a couple of brackets containing a slash.
  - example:
    - <html> opening tag beginning of html document
    - ...
    - </html> ending tag end of html document

- The text included between initial and final tags describes the CONTENT of the element.
  - example:
    - <h1> Internet GIS course </h1>
The HTML element

\[<h1> text </h1>\]

causes the \textit{text} to be displayed as a level 1 heading. This is the boldest, largest heading form available. The \textit{text} may be centred with one or two lines of clear space after the text:

This is \textit{<h1> style}
HTML limits

- HTML has a fixed set of tags and each of them has a definite meaning. Web developers can’t develop custom tags. Therefore a tag `<h1>` always identifies a 1st level title. We can’t develop a tag `<region>` to identify Lombardy as a region, and GIS users can’t create a tag `<line>` or `<polygon>` to represent linear or surface features. This lack of creativity and extensibility makes it difficult to manage and publish spatial data by means of HTML.

- Although standard HTML tags are approved and maintained by the w3c (World Wide Web Consortium), not all of the used tags are standardized. Some of the tags are created by companies that produce browsers and these tags can be interpreted only by their specific browser.

- Tags are used only to describe the structure and visualization of a resource, they do not give any information about its content. In order to transfer geographic data we should therefore use a different language.
Form

- HTML pages which allow the user to send a model (FORM) with data to the web server, in order to be processed by a server-side application, which generates and sends a page in response.

- example:
User-browser client-web server interaction:

The user selects the URI of the page to be displayed.

The client sends to the server a HTTP request, in order to access the searched page.

The server reads the page from its own disk and sends it to the browser through HTTP.

Or it queries a database, receives the data, generates the page and sends it to the browser.

It receives the HTML page, that is interpreted and displayed.

USER

CLIENT

SERVER
Static web cartography uses:

- **FORM** (client-side): for user inserted data
- **CGI** (Common Gateway Interface / server-side): processes requests from the user
- Practically CGI is a protocol that manages messages, an interpreter that receives input from the user and reorganizes it in variables parameters to be used in mapservers and other GIS applications.
The web server receives the HTTP message but it cannot answer, since it can understand only HTML documents. It then passes the request via the CGI to applications able to fulfill it.

The role of a map server is to process the user requests and to send back the results to the web server via CGI protocol.

All the user requests are processed by server-side applications.
Shortcomings

- Low performances: a new process must be created for each request, occupying RAM on the server.
- Statelessness: each request, for example a zoom in or zoom out, needs to go through the whole process. The stateless nature limits the user interactivity at the web client. In addition, the output is still a map image: users cannot directly interact with it.
- It is platform dependent.
- It could pose security risks.
Interactive web mapping - client

- A simple HTML viewer with forms is very limited in terms of user interactivity, especially when dealing with maps and spatial objects.
- One major characteristic of interactive web mapping is that it offers more interactions between the user and the client interface and more client-side processing and functionalities than static web mapping.
- Interactive viewers have been developed such as dynamic HTML and client-side applications such as plug-ins, ActiveX controls and Java applets.
Interactive web mapping - server

- Several extensions are used to overcome the previously discussed shortcomings of the CGI. These extensions run server-side scripts in the same address as the web server.

- The majority of interactive WebGIS applications are based on this model: a dynamic viewer and CGI or CGI extensions.

- WebGIS applications of this kind:
  - MapServer
  - ArcIMS
  - GeoMedia WebMap
  - MapExtreme
  - MapGuide
Server-side extensions

- Used to overcome the previously discussed shortcomings of the CGI. Rather than create a separate process and close it for each request received, as the CGI does, the server-side CGI extensions stay in the memory and are always ready to service other requests. Furthermore, unlike CGI programs that run in different processes, the CGI extensions run server-side scripts in the same address as the web server. Therefore, all the resources that are made available by the HTTP server process are also available to CGI extensions.

- Among the extensions we can have PERL (Practical Extraction and Report Language) scripts, PHP (Personal Home Page), Java servlets, ASP (Active Server Pages) or JSP (Java Server Pages). The power of these languages lies in the fact that they can generate, in the same page, different layouts based on the result of functions written by the programmer.
Implementation strategies in a WebGIS

There are three possible strategies to create a WebGIS, in function of how the operations between client and server are partitioned:

- **server-side** strategy,
- **client-side** strategy,
- **hybrid** strategy.
Server-side strategy

Advantages:

- The user can access complex and “heavy” data, hardly transferable via the Internet and/or locally processable.
- The user can execute geographic analyses (also complex ones) in a short time by using low-performance computers.
- A more efficient control on data use can be applied, both to manage permissions and to ensure proper use of the data.

Disadvantages:

- Each operation (even the smallest) must pass through the Internet with consequent loss of time.
- Efficiency depends on the connection speed (mainly when the transfer of big amount of data is required) and on the server performances.
- Applications do not make use of the operational power of the client machine, they load every request on the server (risk of overload).
Client-side strategy

Advantages:

- Applications make use of the operational power of the users’ computers, thus reducing the workload on the server.

- The user has full control of the data analysis process.

- Once the data and the requested applications have been received from the server, the user can work without interacting with the server.

Disadvantages:

- The transfer of large data files can cause delays.

- The processing of “heavy” files or complex functions can burden low-performance computers.

- The use of the sent data made by the user cannot be checked.
Hybrid strategy

- Uniquely server-side or client-side strategies have, as previously described, some disadvantages that can be overcome by using a mixed strategy.

- Duties are then partitioned:
  - to the server those operations that require a fast computer (e.g. frequent access to databases or complex analyses),
  - to the client the simple operations of analysis or those which require much intervention from the user.

- In this case the client and the server are in continuous interaction.
Mapserver

- Development Environment for the implementation of WebGIS

- Realized by (ForNet project):
  - University of Minnesota (UMN),
  - NASA (National American Space Agency),
  - Minnesota Department of Natural Resources (MNDNR)

- Developed by:
  - MNDNR,
  - Minnesota Land Manager Information Center (LMIC).

- Research and development currently within the project “TerraSIP”:
  - sponsored by: NASA
  - Managed by: UMN and a consortium of agencies dealing with land planning.

- Member of the OpenGIS Consortium (OGC)
Official site: http://mapserver.org/index.html

- Sections:
  - Description (Home page)
  - Documentation
  - Issue tracker to keep track of problems encountered with MapServer ("bugs")
  - FAQ
  - Download

- Mailing List:
  http://www.mapserver.org/community/lists.html
Interaction functions

- Interactions
  - navigation (pan, zoom in, zoom out, predefined views)
  - query (single or multiple queries, based on search criteria or on a feature).

- A high number of parameters allows developers to program the possible interactions with the map depending on the specific needs that have to be met. In fact, besides the substantive variety of values available for the MODE parameter, which regulates the interactions, MapServer provides several other parameters able to supply results useful for the refinement of the interaction functions and for the graphic render of the web interface.

- MapServer in its CGI version is quite limited. In order to have advanced functionalities such as complex queries, buffer, zoom, rectangular selections or routing functions it’s necessary to switch to PHP or Java architectures or to search for Javascript solutions to support HTML.
Technologies
(that you will see in the course)

● creation of dynamic pages by calling the mapserv program (/cgi-bin directory)
  ● MapServer CGI

● use of scripting languages to create dynamic HTML pages and access to MapServer functions via API calls (Application Programming Interface)
  ● Example: PHP/MapScript (p.mapper based PHP/MapScript)
MapServer: basic architecture
MapServer - CGI: System architecture

- Operating System: Windows or UNIX/Linux
- Components:
  - HTTP server (ex: Apache/IIS)
  - CGI: mapserver program (mapserv)
  - Map file (“configuration file”)
  - Initialization File
  - Template file (“user-WebGIS interface”)
  - Browser
1) The user’s browser displays a template file (HTML file)

2) The user sends a request to the CGI based on the template file

3) The CGI processes the request using the parameters passed via the template file and the map file configurations

4) It loads the geographic data creating a map

5) It sends back the map as an answer to the template file, thus to the browser
MapServer - CGI:

-data types

- Vector and database:
  - ESRI Shapefiles (SHP), PostGIS/PostgreSQL, OGR, MapInfo, WFS, GML, Virtual Spatial Data, ArcInfo, ArcSDE, DGN, S57, ESRI Personal Geodatabase (MDB), Inline, KML - Keyhole Markup Language, Oracle Spatial, MySQL, NTF, SDTS, USGS TIGER, GPS Exchange Format (GPX)

- Raster:
  - TIFF/GeoTIFF, GIF, PNG, JPEG, Erdas .LAN/.GIS
MapServer – CGI: folder structure

- The Apache HTTP server configuration file is contained into the httpd.conf file under the conf folder. For example if use the MS4W (http://www.maptools.org/ms4w/) installation under the disk D we have it into: D:\ms4w\Apache\conf.

  In case of our exercise, since the server is under the linux OS, this path is /etc/httpd/httpd.conf

- The Mapserver program (mapserv.exe) is under Apache folder, in cgi-bin (e.g.: D:\ms4w\Apache\cgi-bin). In our linux server case it is in /var/www/cgi-bin/
MapServer – CGI: folder structure

- htdocs is the Apache folder where all the applications of the web server are located. In our case we have a common folder containing the webGIS for all students (utentexx).

  - if http://webgis.como.polimi.it/GIScourse10 is the name of the computer where Apache is located, by typing in the browser http://webgis.como.polimi.it/GIScourse10/utentexx/ the content of the webgis folder is displayed.

  - With the link: http://webgis.como.polimi.it/cgi-bin/mapserv10?map=/u/www_webgis/GIScourse10/completo/ethex.map&zoomsize=2&program=/cgi-bin/mapserv10&web_imagepath=/usr/local/mapserver_tmp/&web_imageurl=/tmp/&utente=utentexx, the HTML home page of the WebGIS mapserver-cgi application is displayed.

- Therefore for implementing the mapserver-cgi webgis, a student X must work into:
  http://webgis.como.polimi.it/GIScourse10/upload.php
  with User: utenteXX
MapServer - CGI: CGI-mapfile-template file communication

- The communication between the CGI, the map file and the template file is accomplished by using:
  - the map file code (mapfile reference)
  - the CGI parameters, that are sent to the CGI via the template file, as attribute-value pairs; the transfer is done by means of FORM or URL
  - the template references, that are variables inserted in the HTML code of the template file; the CGI substitutes these variables with an image or the result of an elaboration
Elements created by MapServer

- In each work session MapServer
  - creates a session identification (<id>)
  - on demand, it creates a map named: <mfname><id>.gif
  - on demand, it creates a legend named: <mfname>leg<id>.gif
  - on demand, it creates a scale bar named: <mfname>sb<id>.gif
  - on demand, it creates a general reference map named: <mfname>ref<id>.gif

  with <mfname> = name of the map file

- The images are saved in a temporary folder. This generated temporary files folder in our server is [http://webgis.como.polimi.it/tmp/](http://webgis.como.polimi.it/tmp/) (in case of MS4W: ms4w D:\ms4w\tmp\ms_tmp)
The web server generates the new template file by inserting the new images.
The two main elements to be built up: Mapfile and Template file

- **Mapfile**: definition of the map visualization properties (definition of layers, colors, symbols, visualization scale, attributes to be queried,...). It is defined by the user as a text file.

- **Template file**: it’s a HTML page (eventually with javascript) that represents the interface between Mapserver and the user.