Incorporating spatial indexing and query into WAIS and Z39.50

Doug Nebert
April 5, 1993

Background

Existing Wide Area Information Server (WAIS) software was designed to support unstructured ad-hoc query into text files by users unfamiliar with specific data holdings, structures, or formats using a single user interface. Search and retrieval of documents is achieved by indexing all words in target text files. WAIS software is ideal for rapidly prototyping the indexing and dissemination of digital text files and related files including images, PostScript documents, and binary data files.

As interest in the WAIS product has grown, more functionality has been incorporated into the indexer, server, and client. Numerous specialized interfaces have been developed that go beyond the unstructured text query and allow searching of ‘fields’ of information for specific terms. The intent has been to supplement the simple query model with more robust tools that can be optionally used by the advanced user -- but to not preclude the support of simple text query by less-advanced users.

The existing WAIS software has used a form of the 1988 version of the National Information Standards Organization Z39.50 standard, known as the Information Retrieval Service Definition and Protocol Specification. This specifies the handling of dialog between the client and server in serving information. A more recent version of this standard (ANSI/NISO, 1992) has been published which allows access to more sophisticated organization of information, defined through official sets of attributes encoded using Abstract Syntax Notation (ASN.1) for application portability. Recent attribute sets have focused on common information retrieval needs defined by the library community (info-1 and bib-1).

In July, 1992, the U.S. Geological Survey and the Clearinghouse for Network Information Discovery and Retrieval (CNIDR) began the development and testing of spatial extensions to WAIS to support the dissemination of earth science information -- most of which is spatially referenced. This paper describes the steps taken and assumptions made in the design of support for spatially-referenced documents in the public-domain version of WAIS.

Current configuration of Spatial WAIS

WAIS employs a client-server model of information retrieval where information is served on multi-user computers (under UNIX or VMS) but can be queried from a variety of client platforms including DOS Windows, Macintosh, XWindows, and simple terminal emulation. At present, the spatial information is extracted from a GIS or entered by the user into a text file whose contents are spatially indexed. Likewise, metadata are entered by a user into a text file or are extracted from an on-line database into a text file for word indexing. Additional documents, such as related graphics or images, are prepared by the user. Although this configuration, illustrated in Figure 1, causes information to be duplicated, it allows prototype systems to be implemented rapidly, drawing on report-making tools of GIS and other software to prepare indexable documents, while keeping the formats and server in the public domain.

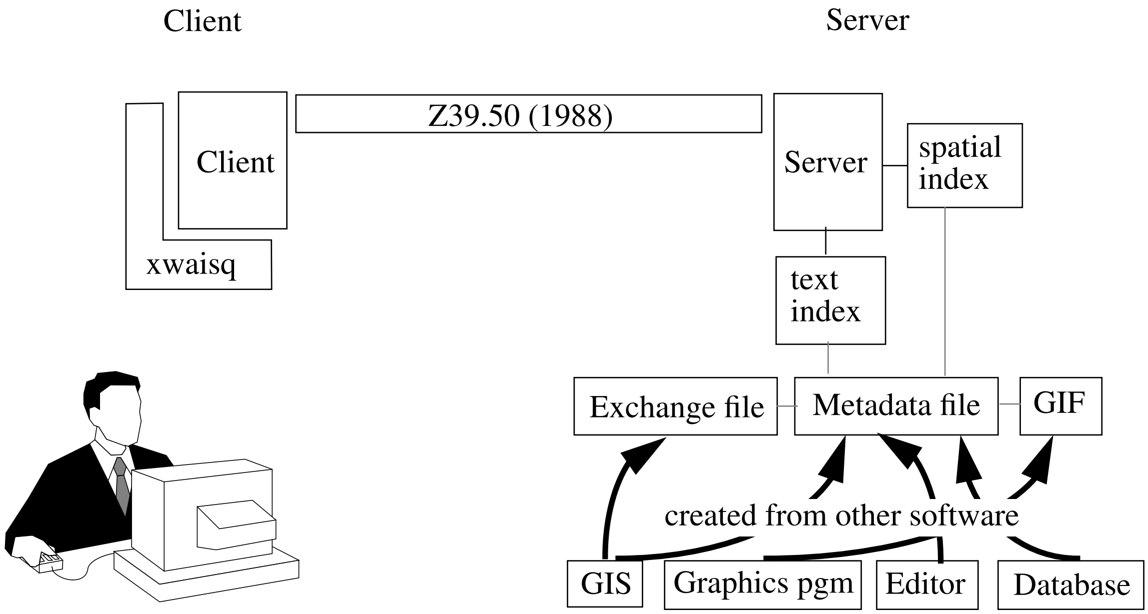


Figure 1. Existing client-server configuration of Spatial WAIS.

Spatial nature of documents

Documents that are managed by WAIS should be thought of as being potentially spatial -- an extra characteristic that could be managed for discovery of a data set. Most environmental data are spatial; all map data are spatial; therefore a large amount of information can be linked to the earth.

To accommodate this feature, all document types should be allowed to be thought of as spatial, in addition to their explicit formatting. In other words, there should not be a type 'spatial' -- rather it is an extra characteristic of any type of document. This will be accomplished two ways: 1) by establishing specifically-formatted geographic references to be parsed from target text files and 2) to enable the support of a companion text file to describe where the spatial keys are stored and how they should be interpreted.

The description of spatial coordinates can be done in a variety of ways. Two constructs were selected in the prototype implementation to provide two different types of locational information. The first construct is that of a bounding rectangle, described in latitude and longitude. This is expressed, using the NASA DIF format using four coordinates in the following format:

Minimum_latitude
Minimum_longitude
Maximum_latitude
Maximum_longitude

All spatial data sets managed by a geographic information system (GIS) whether vector or raster can easily provide the coordinates of a bounding rectangle for searching.

The second construct is that of a detailed, polygonal outline, or footprint of the spatial data on the ground. Whereas the bounding rectangle is easy to produce and process, it does not always portray a reliable region of data coverage. A bounding rectangle defined around a data set for the state of California would imply -- erroneously -- full coverage of the state of Nevada. For this purpose, support is provided for the management of a detailed bounding polygon. The format of this information is expressed as the keyword "inside" and a chain of coordinates stored within parentheses:

inside(45.204,-120.00 46.321,-121.366 45.171,-123.204)

All spatially referenced documents must therefore be described by a minimum of a bounding rectangle, and optionally, by a bounding polygon. If a bounding polygon is supplied, a rectangle can be calculated.

Given that existing respositories of data, or for that matter data in native GIS format, do not contain either the 'inside' phrasing or the declared minimum and maximum latitude and longitude elements, a format declaration can be made using a format file, or a script that describes where in the incoming document the spatial coordinates exist. This would allow information to be represented in its native form, and have the meaning of the coordinates extracted and indexed for query. This functionality has not yet been implemented in the spatial WAIS code, but can be accomplished by writing collection-specific parsing programs to prepare the spatial index file.

Format for spatial coordinates

In order to make the entry and identification (not to mention debugging) of spatial coordinates more intuitive, coordinates are expressed in the query and indexer as positive or negative decimal degrees. Coordinate pairs are expressed in latitude-longitude order for consistency in geographic references, rather than an 'x' 'y' order.

In the future, the client process should allow the selection of coordinate format and presentation during the query and presentation. Regardless of the presentation format, the format of the coordinates expressed internally between the client and server and processed by the indexer and server should be consistent. Whereas precision can be lost in translation it is suggested that coordinates be passed and managed as character strings representing integer decimal seconds.

Indexing spatial data

A spatially-referenced data set is a collection of related information which may be located in a single file or in multiple text files, or possibly even in a data base. To provide access to these spatial attributes through more than a random text query one must support the management and indexing of the spatial characteristics for an indexed retrieval. To accomplish this, a separate file is created during the waisindex process that includes one line per indexed document with the following elements:

begin_byte (int)
end_byte (int)
doc_name (char)
(number of latitude/longitude pairs in decimal degrees expressing bounding area)
doc_title (char)

This would be more efficiently expressed with the following syntax to provide for identification of potentially overlapping targets which could then be visited with a polygon overlay routine:

begin_byte (long)
end_byte (long)
doc_id (long)   -- (links to file described below)
minimum_latitude (float)
minimum_longitude (float)
maximum_latitude (float)
maximum_longitude (float)
number of latitude/longitude pairs to define precise footprint (optional)

Related file:

doc_id (long)
doc_filename (char)
doc_title (char)

At present, the target source of information is searched for geographic keys and the index file is built to identify documents that overlap the search region. Documents that do fall in the search region are then read and the percentage of polygon overlap is calculated and stored for each. All documents that have overlap are tagged. If text is also being searched, those documents that meet the text search criteria are identified. If both text and spatial query are posed, the two sets are compared and only those documents that both fall in the search region and meet the text search criteria are tagged and presented to the client.

The current process uses a serial/sequential search to identify all documents that fall in the search region by applying the polygon overlay technique to all entries. An improvement can be made by modifying the index structure to calculate or load (if trusted) bounding coordinates. These minimum and maximum values can quickly be compared with the search region bounding rectangle and candidates identified. then only the candidates would be processed for polygon overlap where such extended coordinates are provided.

A serial validation of range overlap can be made with the following equation:

$xmin_s$ le $xmax_t$ and $ymin_s$ le $ymax_t$ and $xmin_s$ ge $xmin_t$ and $ymin_s$ ge $ymin_t$

where xmin = minimum longitude
ymin = minimum latitude
xmax = maximum longitude
ymax = maximum latitude
le = less than or equal to
ge = greater than or equal to
subscript s = search bounding rectangle
subscript t = target bounding rectangle

Additional speed could be gained by indexing some or all of the bounding coordinates and employing a binary search on each sorted list using a b-tree, quadtree, or r-tree structure to flag potential targets.

This is where the point-in-polygon and polygon overlap routines would need to be called. A specific vocabulary needs to be defined within the Z39.50 protocol to describe these regions and polygons. For the purposes of query the user may type in the phrase:

inside(lat,long lat,long... lat,long)

or can have this information extracted from the spatial locator window as part of the query. At present this gets passed directly to the server where it is interpreted not as text but as spatial information and is processed against the index file.

An additional construct may be used for query at a point -- which is far faster than polygon overlap. The suggested construct here is the phrase:

point(lat,long)

Processing steps:

1. The server process identifies that the phrase constructs listed above exist in the incoming query. If they do then branch into one of the two following directions:

a. The string includes "point("and includes two coordinates (real or integer) separated by a comma and/or space and terminated with a ")". Check coordinates against a theoretical limit of latitude and longitude (-90 < lat < 90) and (-180 < long < 180) assuming latitude to be the first token and longitude to be the second. If the number exceed values, reject search and return control to text search program.

Open index file and compare point values of lat and long against the ranges in each line, flagging all those document identities that fall within the x range and y range. If a index file exists then use the marked target documents as a list to run the point in polygon against. Flag verified targets and return to text searching.

b. The string contains “inside (” and includes n number of coordinate pairs within braces, each element and each pair separated by commas and/or spaces and closed by a “)”. Check each pair for valid ranges and proceed. If any coordinate falls outside the region, reject search and return control to text search program.

Open index file and identify entries which overlap each of the ranges. If a spatial index file exists then use the marked target documents as a list to run the polygon overlap against. Flag verified targets and return to text searching.

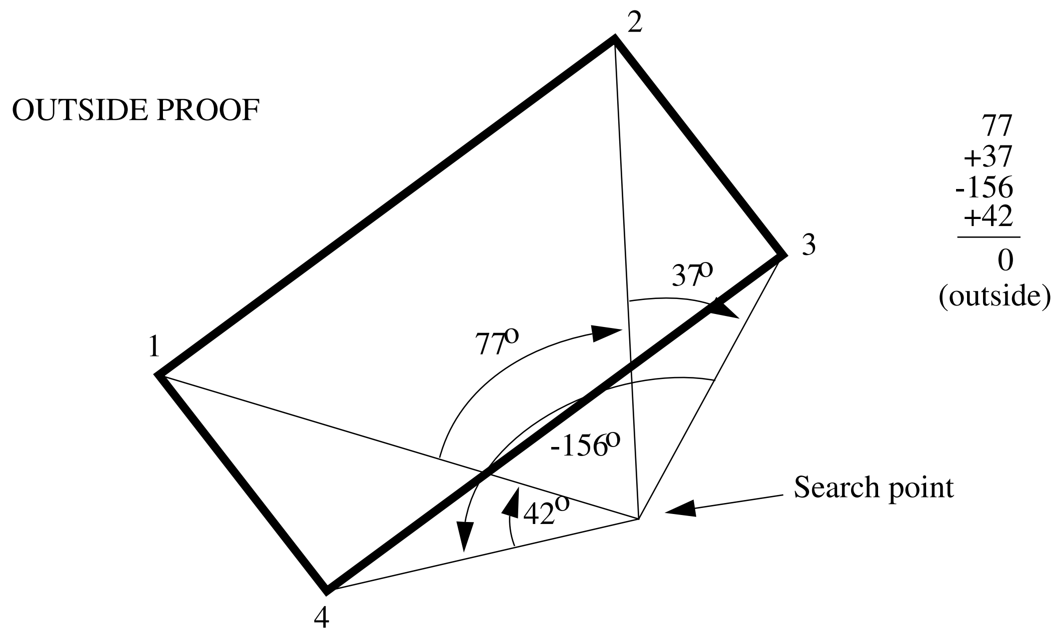
2. Point-in-polygon processing:

a. Open spatial index file
b. Move to first/next entry that has an extent overlap with search point
c. Read in the number of coordinate pairs to work through
d. For each “chord” or point-to-point pair that defines the perimeter, calculate the angle between the first point, the search point (vertex or angle point), and the second point. Place this angle value (positive or negative) into a summed variable and continue.
e. The sum of the angles measured from each point in sequence as positive (clockwise) or negative (counter-clockwise) is equal to 360 (or -360) if the point is on the boundary or inside the boundary, or zero if it is outside, as illustrated in Figure 2.

3. Polygon overlap processing:

a. Open spatial index file
b. Load the query bounding coordinates into a 2-D array
c. Identify a bounding box that contains the query bounding coordinates as they are loaded in array
d. Move to first/next entry that has an extent overlap with search polygon
e. Determine if any line segment from the query coordinates intersects any line segment of the bounding region of each indexed data set:

Determine the equation of each pair of line segments, one from target, one from query
Increment through all target segments, as shown in Figure 3.



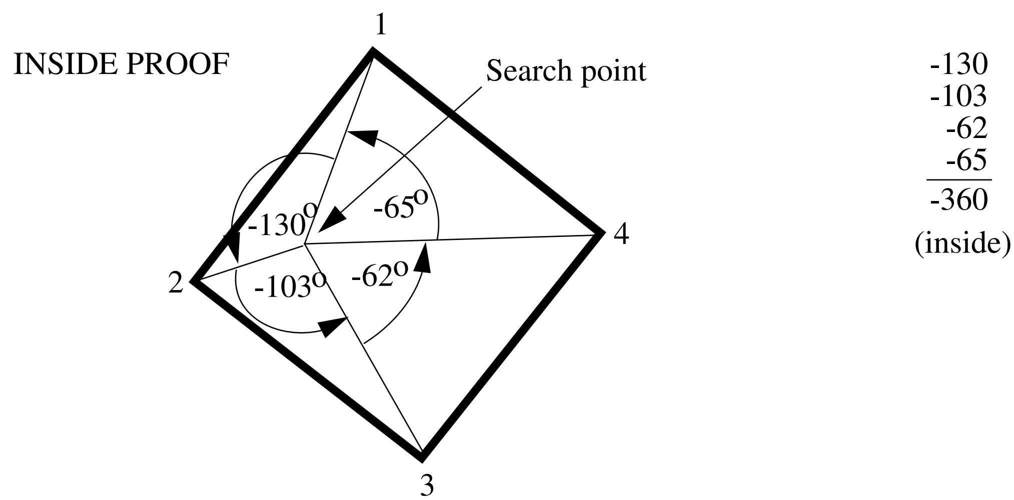
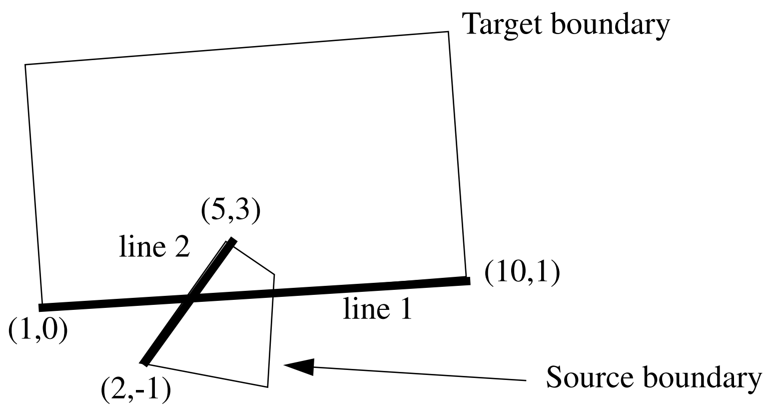


Figure 2. Point-in-polygon seach algorithm proof

Note: bounding polygon does not need to be clockwise or counter-clockwise. By taking the absolute value of the sum the result is the same.



Equation for a line segment

$$y = mx + b$$

$$x = \frac{y - b}{m}$$

where $m = \dfrac{\text{delta y}}{\text{delta x}}$

$$b = y_1 - \frac{\text{delta y}}{\text{delta x}} x_1$$

$$m_1x + b_1 = m_2x + b_2$$

$$\frac{y - b1}{m1} = \frac{y - b2}{m2}$$

Line 1

$$y = \frac{(1-0)}{10-1}x + b$$

$$y = 1/9x + b$$

$$b = 0 - 1/9(1)$$

$$b = -1/9$$

$$y = 1/9x - 1/9$$

Solve for x value:

$$1/9x - 1/9 = 4/3x - 3.6667$$

$$x = 2.909$$

Line 2

$$y = \frac{(3+1)}{5-2}x + b$$

$$y = 4/3x + b$$

$$b = 3 - 4/3(5)$$

$$b = 3 - 6.66667$$

$$b = -3.66667$$

$$y = 4/3x - 3.66667$$

Solve for y value:

$$\frac{y + 1/9}{1/9} \qquad \frac{y + 3.6667}{4/3}$$

$$y = 0.43$$

Figure 3. Polygon intersection proof

Serving multiple formats of data

In many cases, there will be several related information objects (documents) that could be shared as the result of a query. For example, one may wish to post both a text format file (ASCII) and a PostScript file of the same report. In the spatial realm, it is envisioned that the metadata collection is one identifiable store of information -- almost in the form of a report -- which can be augmented by a graphic portrayal of the data set contents. Where the data set is also on-line, the digital spatial data set in some exchange format would be a third object that should be available for retrieval via WAIS. For the prototype we have identified these three data types to be linked for each digital spatial data product posted in WAIS.

To create a multiple-format server, the filenames must begin with the same word prefix, so that the suffixes can be used to interpret the different file format types. These file format types are interpreted by the client and are processed using filter programs listed in the local Xwais parameters file. For the prototype, the code was modified to link files of type .gif, type .doc and type .e00.Z as kindred documents. This was accomplished in the indexer code by starting the name of the index file set with the phrase ‘sp-multi-’. The syntax for the command would be:

```
waisindex -d /usr/opt/wais_data/fullton/sp-multi-reston /usr/opt/wais_data/reston/*
```

The -d option identifies the rootname and path of the index files to be created. The pathname at the end specifies the pathname to the objects to be indexed. This directory contains the linked files with the .gif, .doc, and .e00.Z suffixes.

WAIS client extensions to assist in spatial query

In response to requirements for local (off-network) access to documents in the PC environment, the USGS Information Services Division (ISD) programmed a Windows 3.0 client to run against local or network WAIS data. Included in this release was a spatial query engine driven by a reference map of the globe in orthographic projection on which one can trace areas of interest for query. This functionality is matched by a prototype WAIS document server that is capable of indexing and serving map documents (a spatial ’footprint’ and associated documentary text) based on this combined spatial and text query. The ability to graphically pick a search area exists only on the DOS client, and although one can type the coordinates of a bounding area into the dialog area, the functionality in the X version is limited.

The Windows client (WINWAIS) was written in QuickBASIC on a 486 PC. The graphics and map orientation were taken from prototype software from the EROS Data Center of the USGS. This portion was written in C with a MOTIF user interface. The main WAIS interface was styled after the XWAIS and XWAISQ clients but code was written from scratch.

A port of the WINWAIS product, with specific modifications, was required into an X-Windows compatible product that will run on the DG Aviion workstation, using X11R4 (minimum). It is desired that this be a generic port to allow for use on other platforms and under other windowing systems (OpenLook) rather than be a specific MOTIF port.

Client design pproach

Existing WAIS b5 code that executes properly on the DG shall be used in the port. This shall be the basis for the enhancements. The b5 code should include the field definition (added by Tim Gauslin), and stemming and multi-format retrievals added by Jim Fullton. Starting from this point the xwaisq client needs only minor modifications but will need an entire new child window to display and collect the geographic information, as is done in WINWAIS. A mocked-up xwaisq client screen (using MOTIF widgets) is shown in figure 4 to illustrate the user interface.

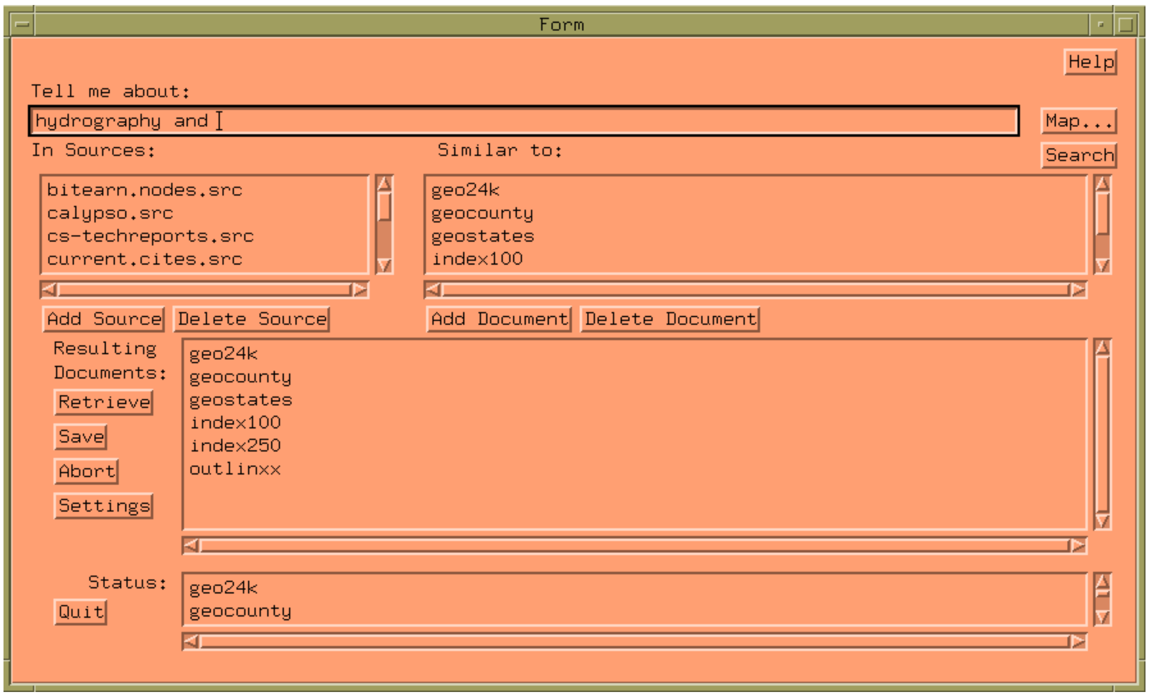


Figure 4. Motif mock-up of spatial WAIS query window.

This first screen approximates the position of features in the b5 xwaisq client release. The only changes required are the addition of a 'Map...' button near the Search button and the changing of the title used on the top button near 'Resulting Documents'. The Map button was placed there because spatial constraints should be placed on a query just prior to executing the search. The Help button was placed in the upper right to conform more to MOTIF, but this can be left in its present location.

The Tell Me About widget must be able to accommodate multiple lines of text as the geographic coordinate query may require several lines. This should display two lines at a time with a scroll handle. The order of execution is 1) select one or more sources, 2) place additional references in Similar To, 3) provide text criteria for search in the Tell Me About field, 4) press Map... to impose spatial constraints on the search, and 5) issue a search against the sources.

By pressing the Map... button a new screen assumes control from the parent screen. Input must be directed through the Map screen. A portion of the map control screen is shown in figure 5. These controls allow one to center the map and zoom in or out to identify search regions.

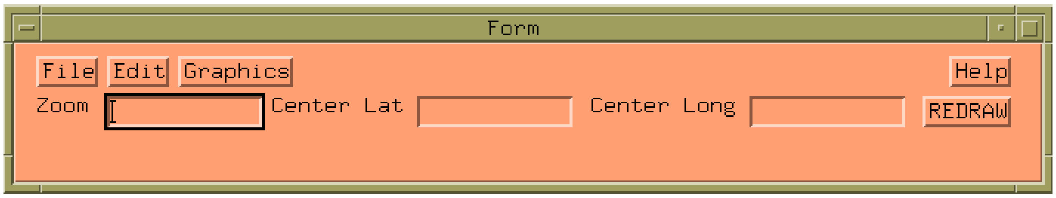


Figure 5. Mock-up of controls for map query window.

This should be a multi-paned screen with a pull-down bar across the top with the File, Edit, Graphics, and Help options available as pulldown or cascade-style menu options. The Zoom, Center Lat, and Center Long should also include toggles to permit the incrementing of the numbers up or down -or- the entry of the number into the field.

The contents of the File, Edit, Graphics, and Help submenus are desceribed below with the resulting actions or events, described later in this document.

File	Edit	Graphics	Help
Define search region	Edit search region	World Map...(on/off)	
Save search region	Merge search regions	Graticule...(on/off)	
Load search region	Rename search region	Other Map...(path search and set)	
Add to query		Set Color... Map Color	
Exit		Search Color	

This screen should have the pulldowns at the top, additional buttons beneath, and a square drawing region for the map appended within the same frame below. The general orientation of the WINWAIS is a good example.

The File options are associated with the saving and loading of stored search regions or ’footprints’ and also the standard way to exit. The Edit submenu allows one to edit search regions. The Graphics submenu allows one to toggle on/off the world map and the graticule. This is actually a cascading menu where when one depresses World Map a sidebar menu with the options ’Draw Map’ and ’Hide Map’ appear. Same condition exists for the lines of latitude and longitude (graticule). Users have expressed an interest in drawing their own maps into the background for reference. This option merely redirects the input of the map graphics to a new source file in the same format as the WDBII data set. For ease of use in a variety of GIS environments, the world map is drawn directly from a Digital Line Graph (DLG-3 Optional) format file, units in decimal seconds. Any other backdrop maps may be supplied by the users in this public-domain DLG format. An option to change color for the current drawing session is provided. Help should be available for every menu and submenu topic. WAIS is not so easy for many users despite how obvious it seems to us...

Steps to programming the spatial client events

1A. Add button to xwaisq screen. Button label is "Map..." When depressed this causes a new window to be opened and all control is passed to this child window when the pointer is within the window.

1B. Modify "View" event in xwaisq such that if a document is returned and has a spatial footprint, the footprint information (in inside() syntax) is displayed in the MapQuery window, if open, using the correct search_color, defined below.

2. Create new window for map query functions, modeled after the WINWAIS. This window should be titled "Map Query" and would have a pull-down menu with three options on the left, titled "File" "Edit" and "Graphics" with a Help button on the far right as is standard in MOTIF. This could be implemented as buttons using Athena widgets then migrated to MOTIF as necessary. Under the pulldown is a panel of "Zoom", "Center Lat" and "Center Long" input fields that may also be set by valuators next to them (see WINWAIS). A re-draw button is show on the right hand side to replay the latest settings. A graphics canvas (part of the same window) shall cover the lower 80% of the window and will be where the reference map graphics are drawn (from ref_aid source). Window should also allow for messages to user.

3. File menu options.

Add X resource called: Xwais*commonLocationDirectory
to be where location definition files are stored for common use

Initialize_map event: On beginning a graphics session, initialize coordinate buffer
Set Globe: true, Grid: true, Back_color: white, Other_Map: null, Map_color: black,
Search_color: red

Define_location event: Notify user to begin defining a selection region using 1 button,
2 button to un-do last coordinate, 3-button to terminate or close the selection region
Pick key,x,y coordinates from screen, draw connector lines in Search_color
Translate coordinates into Decimal Degree values from orthographic projection
Store in buffer as chain, or if only one point is entered, as a single point.

Save_location event:
If ~userid/wais-locations doesn't exist, then create it
If Coord_buffer is empty then Do
Inform_user: No coordinates have been defined to save...
Doend

Else Do
Open_window to allow user to define name of text file to save location
Default directory is ~userid/wais-locations
Save buffer to textfile:
If npnts = 1, then write “point(xcoord,ycoord)
If npnts = 2, then tell user “Must have at least three points to define region” No save
If npnts ge 3, then write “inside(xcoord,ycoord xcoord,ycoord ...)” to file
Doend

Load_location event:
Open_window to display alphabetical file list of Xwais*commonLocationDirectory and of ~userid/wais-locations contents (merged together) (See add source window generation under xwaisq for examples). Window is scrolling list.
Single click on entry in scrolling list loads location variable
Open and read location file into buffer
Translate lat-long coordinates back to orthographic coordinates and display in Search_color

Add_to_query event:
If Coord_buffer is not empty then do
Append text of the location (buffer) into the Query window in xwaisq
Clear buffer
Doend

Exit event:
If Coord_buffer is not empty then do
Query_user: Locations referenced on map not saved. Do you want to save?
If query is true then save_location
Query_user: Do you want to append this search area to text query?
If query is true then add_to_query
Clear buffer
Doend
Close_Window
Return control to xwaisq window

4. Edit Menu Options

Edit_location event:
If Coord_buffer is empty then do
Inform_user: Select a location to edit...
load_location
Doend
Inform_user: Add point with 1 key, delete point with 2 key, 3 to finish
Get_edits loop:
If key = 1 then do
Search for nearest two points

Break chain
Insert new coordinate into buffer
Get_edits
Doend
If key = 2 then do
Search for nearest point
remove point from chain in buffer
Get_edits
Doend
If key = 3 continue

Merge_location event:
If Coord_buffer is empty then do
Inform_user: Select a location to edit...
load_location
Doend
Query_user: What location do you want to add to:
load_location
save_location

Rename_location event:
load_location
Query_user: New file name?
rename location file in ~userid/wais-locations

5. Graphics Menu Options

Set_Globe event:
Cascade to toggle World Map on or off
If Globe is true then Inform_user: World Map is on
If Globe is false then Inform_user: World Map is off

Set_grid event:
Cascade to toggle lat-long grid on or off
If Grid is true then Inform_user: Lat-Long grid is on
If Grid is false then Inform_user: Lat-long grid is off

Set_other_map event:
Open_window to browse directories to find reference data sets
Set Other_map to full path of selected data set

Set_map_color event:
Popup menu of named X11 colors that they can draw
Set Map_color to selected color

Set_search_color event:

Popup menu of named X11 colors that they can draw
Set Search_color to selected color

Clear_graphics event:
Initialize graphics portion of window

Redraw event:
Fill background using back_color
If Grid is true then draw_grid using map_color
If Globe is true then draw_globe using map_color
If Other_map is not null then draw_other using map_color
If coord_buffer is not empty then draw coord chain using search_color

Future programming tasks

The prototype spatial WAIS software allows one to index multiple related documents and extract spatial location to provide relevance in searching. A visual tool is provided to allow users to define a search region of interest and have the coordinates passed with the text query. The system works well with small collections of information (hundreds of entries per server) but is slow for larger searches due to its sequential search technique.

A binary search technique is recommended for the public-domain version of WAIS that allows for the more rapid indexing and discovery of a collection of spatial data sets. This may occur in a database such as Postgres, where spatial structures and operators exist, or as a series of sorted files outside of a database. Based on the range of potential users there should be a variety of sort/searching solutions available, depending on database size and level of expertise in maintaining a transactional database.

WAIS currently creates multiple-format or spatial sources when the root filename is altered to include the 'sp' or 'multi' phrase in it when indexing.

WAIS should support the discovery and indexing of spatial characteristics in all supported data types that are text oriented looking for standard constructs first or enabling parsing of spatial attributes via a companion format file. Similarly, WAIS should support multiple-format service among all supported data types by declaration of parent and child documents in a companion format file. the parent document would be the one to be searched for text relevance (like the documentation object), whereas the GIF graphic files and exchange files are listed as part of the multi-format retrieval if the parent is selected.

WAIS currently accesses user-defined files -- typically text files -- to search for text relevance. This works well where the documents are the true objects being accessed, but presents some problems in efficiency and currency where the information is actually stored in a database or other foreign data structure and must be dumped to a textfile for indexing. In the GIS environment, this includes both the metadata elements and the spatial coordinates.

A Z39.50 server and client toolkit that complies with the 1992 version of the protocol should be made available to software vendors to build specialized interfaces between queries submitted via Z39.50 and custom data stores.

Spatial and descriptive information should not need to be duplicated for serving via WAIS.

Vendors should build interpreters that can translate query about specific objects received via Z39.50 into a query against the native GIS data structures, thereby providing direct access to the spatial and attribute data bases. To accomplish this, a formal attribute set must be described and formalized working with the spatial data user community and the Z39.50 implementors. Once established, the specific terms describing coordinates metadata would be more easily interpreted and made available for response. It is suggested that this attribute set include terms described in the Spatial Data Transfer Standard (SDTS) and the Spatial Metadata Content Guideline promoted by the Federal Geographic Data Committee and ASTM.

<u>Future configuration</u>

Figure 6 illustrates the potential configuration of spatial WAIS in the future as the 1992 version of the protocol is established within the WAIS community. A user will be able to connect to one or more servers holding spatial information and through what is called the 'Explain' request, get information on the format and content of the target information collection. This will allow the user to compose detailed queries against specific 'fields' of information or default to random-text

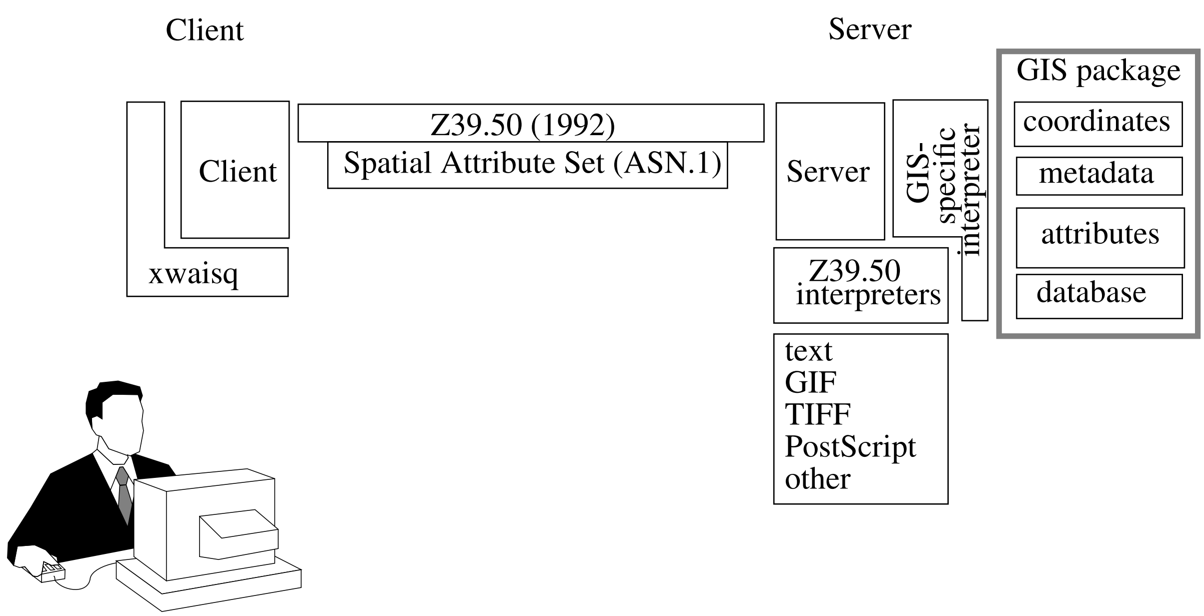


Figure 6. Potential client-server configuration of Spatial WAIS.

query. In order to have a stable vocabulary of spatial metdata an attribute set should be defined using the Abstract Syntax Notation (ASN.1) and registered with the Z39.50 standards community. The content of these terms should be derived from the SDTS and metadata guidelines for consistency. This will allow users with different user interfaces, perhaps operating in different GIS or WAIS implementations to compose queries that can be understood on a remote, Z39.50 compliant system. Such an implementation will take commitment by the information provider, software vendor, and user communities but would result in wide information access and would require a minimum of translators between software systems to provide such interoperability.

References

ANSI/NISO Z39.50-1992, 1992, American National Standard Information Retrieval Application Service Definition and Protocol Specification for Open Systems Interconnection. National Information Standards Organization, Bethesda, MD, 37 p.