

GRASS 4.0

*Introduction to Geographic Information
Systems Using the Geographical Resources
Analysis Support System (GRASS).*

*John E. Parks
Center for Advanced Spatial Technologies
National Center for Resource Innovations, SW
University of Arkansas
Fayetteville, AR
November, 1991*

Draft Document - Do not cite without permission of author.

Chapter 1 *History & Origins of GIS*

Objectives:

This chapter is meant to introduce you to the background of Geographic Information Systems. It was written by Jim Carrington of the Soil Conservation Service. It states the major points of the development of Geographic Information Systems in the U.S. and elsewhere.

After reading this chapter you should be able to:

1. Describe the basic functions of a GIS.
2. Give a date to the introduction of both manual and automated GIS's.
3. Name several of the GIS's developed by the U.S. government as well as Canada.
4. Name a few of the government agencies which use Geographic Information Systems.

DRAFT

History and Background of GIS:

GIS's are known today as systems that can collect, manage, manipulate, analyze and display spatial and attribute data. These systems are known today for the ability to take and work with many types of data, and draw conclusions through the analysis of many data layers. For example: show me the occurrences of highly erodible soils, less than fifty meters from perennial streams, with a 10% slope, that fall in Tarrant County, and on land that is publicly owned.

Some have estimated there are some 83,000 GIS's found throughout the country, in towns, cities, counties, states and federal agencies as well as private industry. There are at least 50 federal agencies using GIS technology today. A recent survey found at least 62 different GIS packages in existence. There are plenty of "home spun" college, governmental and private industry GIS packages as well.

The first usages of GIS's may have been with the Public Land Survey of 1785 (with a referenced geographic land system) and possibly with the beginnings of thematic cartography in the 1800's. In addition, the development of the computer punch card by Hollerith in 1890 fostered the growth of automated geo-processing.

The modern era of geographic information systems is attributed to the refinements of cartographic technique - like stable map base materials and color separation, as well as the growth of spatial analysis - through such work as Ian McHarg's "Design with Nature" and lastly through the rapid development of the computer - from main computing to powerful network computer workstations.

Computer cartography has allowed for:

- a. maps being produced more quickly
- b. maps being produced inexpensively
- c. map production which was not possible before
- d. maps designed for specific user needs
- e. cartographic experimentation
- f. facilitates analysis of spatial and attribute data
- g. minimizes map as a data store
- h. maps created that are impossible to produce manually
- i. maps produced where the parameters can be defined
- j. automation throughout the map making process
- k. maps produced from digital data

Through the 60's and into the 70's much development and

D
R
A
F
T

application of automated cartography was undertaken - most notable in the digitizing and plotting of data. As computers got cheaper and more efficient software was designed; two directions were established. The first being the continual automation of cartographic tasks with emphasis in cartographic accuracy and visual quality. And secondly, the beer methods for the analysis of spatial data were developed. The latter being such systems as STORET by the Public Health Service and the former being systems such as MIADS by the Forest Service.

The first GIS in modern time is generally acknowledged to be the Canada Geographic Information Systems (CGIS) in 1964. (See Appendix A for more information) Many other GIS's were developed later on, particularly by the federal government like MOSS by the Forest Service, ELAS by NASA and DIME by the Census Bureau. After public domain GIS's came about, commercial vendors took up interest in GIS. Among these were ESRI^{®1} with ARC/INFO^{®2} and Intergraph Corporation^{®3} with MGE^{®4}. These and others have been concerned with the positioning of data with respect to a known coordinate system, attribute information on the data and the data's spatial or topological relationship to other data - but do so in their own unique fashion.

GIS differ from computer graphics because graphics are concerned with what appears on the screen and manipulates that image. GIS uses the display as soft copy, in the manner of a computer print-out or plot. The graphic display serves as s by product of the GIS software.

Some of the professional societies that are involved with GIS include: AAG - Association of American Geographers, URISA - Urban & Regional Information Systems Association, ACSM - American Congress of Surveying and Mapping, ASPRS - American Society of Photogrammetry and Remote Sensing, and NCGA - National Computer Graphics Association.

Some of the federal agencies using GIS include: Census, USGS, Forest Service, BLM, Corps of Engineers, National Park Service, EPA, and DOD to name a few. Magazines are published by the above mentioned professional societies, and serve as very good

1. ESRI[®] is the company name and registered trademark of Environmental Systems Research Institute, Inc., Redlands, CA. USA

2. ARC/INFO[®] and PC ARC/INFO[®] are registered trademarks of Environmental Systems Research Institute, Inc., Redlands, CA. USA

3. Intergraph is a registered trademark of Intergraph Corporation, Huntsville, AL. USA

4. MGE is a product of Intergraph Corporation, Huntsville, AL. USA

sources of information. In addition, “GIS World” magazine and GIS college textbooks are extremely useful.

DRAFT

Chapter 2 *Map Projections*

Objectives:

This chapter will explain some cartographic principles as well as define the meaning of several map projections. The information provided is useful to either manual or computer aided mapping. It should also help you understand a few of the map projections used by the U.S.G.S. (United States Geological Survey).

At the end of the chapter you should be able to:

1. Define the following terms:
 - geocentric
 - geoid
 - graticule
 - grid
 - projection
 - spheroid/ellipsoid
 - UTM
2. Explain the difference between a grid and a graticule.
3. Locate the projection, grid/graticule and datum information on a map.
4. Name the most common map projections used by cartographers as well as name the projection used by GRASS.

The text in this chapter is extracted primarily from a training manual used by the Soil Conservation Service for teaching GRASS. Several changes have been made to the text for inclusion in this manual.

DRAFT

Development of Coordinate Systems:

Several methods for representing the spherical earth on a flat surface have been developed from the time of the Greek philosophers. Mathematical representations were developed through the efforts of Mercator and others in the 1600's. Since then refinements to various projections have occurred in attempt to better represent the true surface form of the earth. Several such projections are represented in the reading which follows.

Before we can understand coordinate systems on maps, we must take a look at a globe, a scaled representation of the earth, and note its characteristics. Essentially, the earth is a sphere which has two points that will help us to develop a reference system for points we wish to locate on its surface. The earth rotates on its axis and the two ends of this axis form our north and south reference points, the poles. Between these poles we can determine an *equator*, a line equidistant from the poles and forming a *great circle*. This great circle divides the sphere into two equal hemispheres as it passes through the center of the earth.

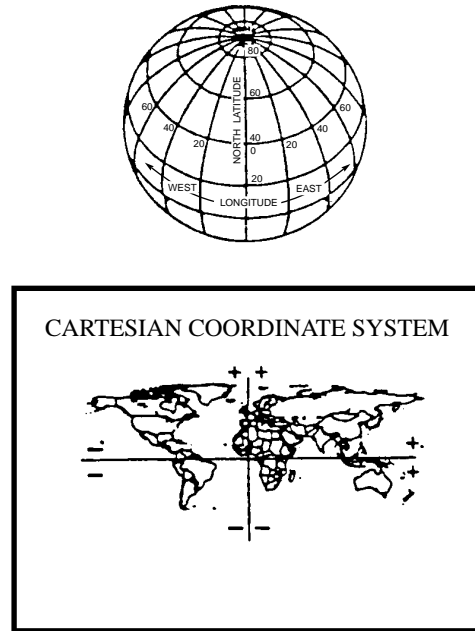
With the polar points and the equator, we now have a method of labelling locations north and south of the equator, called *latitude*. By convention, we use a *sexagesimal* system of assigning coordinate values; we call the equator zero and count northward or southward to 90 degrees, which is the value given the poles. Each of these degrees may then be broken into 60 minutes, each minute into 60 seconds, and each second into decimal tenths, hundredths, etc. to attain any desired accuracy. This is not the only way we could count, but it is a time honored convention.

A latitude, however, is only half a coordinate. At 40 degrees north latitude, we could be anywhere around the globe, in the U.S., Spain, Italy, Greece, Turkey, the Soviet Union, China, Korea or Japan. An east-west reference is also necessary to define a point. Unfortunately, there is no point in the east-west direction which easily defines the starting point for *longitude*. For years, countries set their own starting point, their *prime meridian*, the French in Paris, the Americans in Washington, etc. It was not until the late 1800's that the major mapping nations arbitrarily chose a longitude line passing through England's Greenwich Observatory to use as the international prime meridian, zero degrees longitude. Sexagesimal values were then assigned from zero degrees to 180 degrees both east and west, meeting at a line in mid-Pacific Ocean. Again, this is a conventional numbering choice.

D
R
A
F
T

The latitude and longitude lines, then, apply a Cartesian coordinate system to the globe.

Figure 1. The Cartesian Coordinate System



This is important when working with geographic coordinates (latitude and longitude) in GIS. The United States, for example, falls in the quadrant where latitude is always north of the equator (+) and longitude is always west of the prime meridian (-). A computer, given north longitude values in the United States would be able to “flip” the product - a map of the contiguous U.S. could show California on the east coast, Florida on the west. Maine and Texas would remain in the north and south, respectively, even though they would be plotted in reverse orientation laterally.

While you are looking at the globe, notice also that the latitude lines form *parallels* and that the equator is the only latitudinal great circle, dividing the globe equally into two half-spheres or hemispheres. The remaining parallels are *small circles*, decreasing in size northward or southward to the poles which are a point rather than a small circle. All longitudes included with their counterparts on the other side of the globe, form great circles and each longitude line, or *meridian*, converges with the others at the poles. As we shall see later, these fundamental relationships may change as we construct flat representations (projections) from the spherical earth.

This, then, is the most basic of earth’s reference systems, lati-

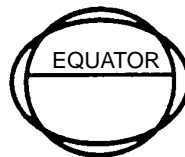
tude and longitude. The units are usually sexagesimal (based on units of 60) but degrees can be quoted in decimals, i.e., 40.50, 40 degrees and five tenths of a degree, or 40 degrees and 30 minutes.

Definitions of Basic Terms:

The problem presented in this portion of the course is to transfer the referencing system on the spherical earth to a flat plane, the computer screen or a map, with an acceptable degree of accuracy. Once the general principles of this process are understood, we can discuss some alternate referencing systems, the grids, which are related both in theory and positioning to the geographic coordinate system.

To know our position on the earth with any reasonable degree of precision, we must know the size and shape of the earth to which the coordinate systems has been applied. As the earth rotates around its axis, it forms an *ellipsoid* due to centrifugal force, bulging at the equator and flattening at the poles. This polar flattening is approximately one part in 300.

Figure 2. An approximation of the true figure of the Earth as an oblate Spheroid. The flattening at the poles and bulging at the Equator results from the forces of centrifugal rotation.



The earth in this rotational form can also be referred to as spheroidal. (*spheroid* and ellipsoid are often used interchangeably.)

There have been many estimates of the size of the earth and the polar flattening, but the estimate which best fits the globe in the North American area, until recently, is called the Clarke 1866 Spheroid or Clarke 66 for short. This spheroid is the basis for much U.S. mapping and is the default choice on many ellipsoid menus in GIS. The Hawaiian Islands, falling as they do in mid-Pacific, use the International Spheroid and foreign areas have been placed on still other ellipsoids, depending on which estimate of size and shape best fits their particular geographic location.

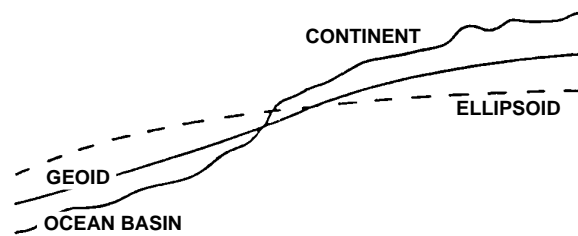
In the last two decades, using highly advanced earth measurement techniques, a new “best fit” estimate for the entire world has

D
R
A
F
T

been derived. It is called the Geodetic Reference System, 1980 or GRS 80. It is also a choice on ellipsoid menus, but should be used only with newer U.S. materials which state that they are on North American Datum 1983.

Another basic word describing the shape of the earth is *geoid*. The geoid is an equipotential surface of the earth's gravity field; it can be envisioned as a continuous sea level surface extending below the continents. The geoid undulates in response to distribution of the earth's mass and is, therefore, not useful for horizontal surveying and mapping measurements although it is used in levelling (height determination). For horizontal measurement, the ellipsoid is used.

Figure 3. Relationship (exaggerated between an ellipsoid, such as Clarke 1866 and the geoid. Deviations are due to variations in the gravitational field caused, in part, by unequal mass distribution of crustal materials.



Once we know the size and shape of the ellipsoidal earth and can measure thereon, we are faced with the problem of flattening the curved surface onto a plane such as a map. This flattening process creates a projection, the basis framework of reference lines for a map.

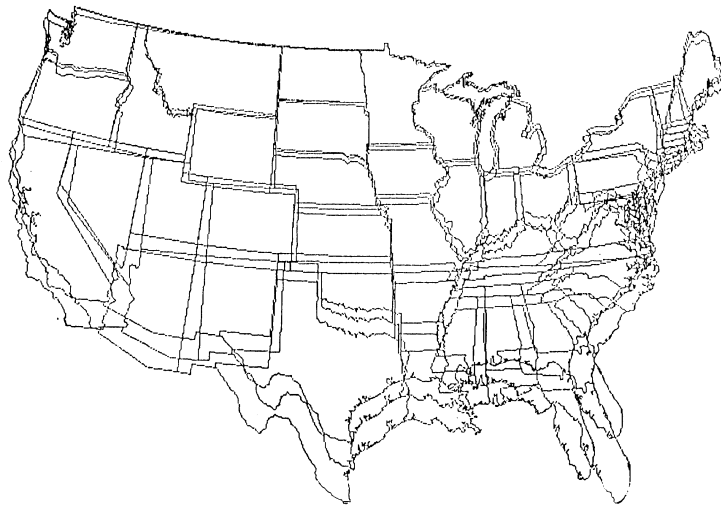
A *projection* (with longitude and latitude lines is called a *graticule*) may have some measurement disadvantages caused by curved parallels and meridians and the sexagesimal numbering system. To alleviate some of these problems, another type of referencing system, a *grid*, has been developed. A grid consists of two sets of straight parallel lines, equally spaced and perpendicular to each other. This system may be overlaid upon, and mathematically related to the projection. It offers the advantages of straight lines for measurements and decimal numbering and, since it is designed to use only the upper right hand quadrant of the Cartesian coordinate system, all values may be read in a plus direction. Values should always be read “right and up.”

One last term should be defined before we look at specific projections. This is the word *datum*, a quantity or set of quantities which serve as a referent or basis for calculation of further quantities. In the cartographic sense of the word, we speak of a datum as the reference point or plane for the horizontal and vertical surveys which establish the position of features on the earth.

The Importance of Reference Systems in GIS:

The importance of using or converting to consistent referencing systems in a GIS is illustrated in Figure 4. This plot of the United States indicates clearly the differences which arise in plotting the same area at the same scale but on different projections. Should you try to hold a computer plot against a piece of source material, for example, or to match sheets, you may encounter serious mismatches unless the referencing systems are the same.

Figure 4. This diagram shows the variations in the Albers Equal Area projection (tip of Florida farthest north), the Transverse Mercator projection (tip of Florida farthest west) and the Lambert Conformal Conic projection (tip of Florida farthest south). All three maps are plotted at the same scale.



DRAFT

Projections:

As we have said, there is no way to perfectly transform the spheroidal shape of the earth into the flat plane needed for a map. You can illustrate this to yourself by attempting to flatten a half, a quarter or an octant of a rubber ball with your hand...there will always be a bulge somewhere. However, as the pieces of the ball get smaller, the bulging effect is less and less apparent and the same is true of maps as they become larger and larger scale, covering smaller and smaller areas of the earth. At scales of 1:24000 or larger, the difference between two projections over the same area may be less than the amount of paper shrinkage the map can suffer (which may approach an eighth of an inch). Projection mismatches between sheets, however, show a more pronounced problem as you attempt to join one map to another at the *neatlines*. We shall see examples of this in the following projection diagrams.

There are four major elements which may become distorted in changing from a spheroid to a plane. These are shape, area, distance, and direction. When a map preserves spheroidal shapes in an acceptable fashion, it is labelled *conformal* or *orthomorphic* and all directions around each point are represented correctly. It does not mean, however, that this correctness applies from point to point. All parallels and meridians meet at right angles on the flat map. Local scale is constant but is not so over large areas.

Maps which preserve area relationships well are called *equal-area* or *equivalent* or *homolographic*. This means that the areas are represented in correct relative proportions but spheroidal angles are distorted. Unfortunately, conformity and equivalency are two of the most sought after elements in the choice of projections and a map cannot maintain both at once.

The third element is accurate distance measurement, which derives from constancy of scale; i.e., the scale must remain uniform on a line connecting any two points. This is not perfectly attainable, but scale variations can be quantifiable. Scale may be maintained in one direction, for example north-south or east-west, or it may be maintained in all directions from one or two points only (equidistance). Overall, distance can only be correctly measured if you are aware of the inherent scale variations at any given point.

Lastly, we cannot represent all earth directions with straight lines on a flat map. Plotting true directions on a globe reveals that all great circles are straight lines which cross successive graticule lines at a constant angle. Some maps approach this by showing lines of constant bearing as straight lines, but these *rhumb lines* are not always great circles.

Bearing these limitations in mind, we should now look at various projections in common use. These projections fall into groups, those made on cones or cylinders or planes. These geometric forms are called “developable surfaces.”

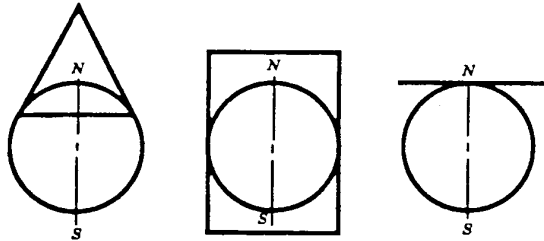
There are over 250 known approaches to the problem of flattening the spheroid; in this part of the course, we will cover only the common ones we use every day. Each of these projections treats spheroidal coordinate characteristics such as parallel latitude lines or convergent meridians in a different fashion and none accomplish the perfect transformation.

The conic projections are formed by placing an imaginary cone

DRAFT

over the earth so that it touches the spheroid at one circle around its circumference. Spheroidal data is projected onto the cone, which is then cut and laid flat. The cone can be placed over the earth in any aspect, but its axis usually coincides with the earth's polar axis. Azimuthal or plane projections involve a plane touching the earth at only one point. Cylindrical projections involve cylinders slipped over the globe and touching at one great circle.

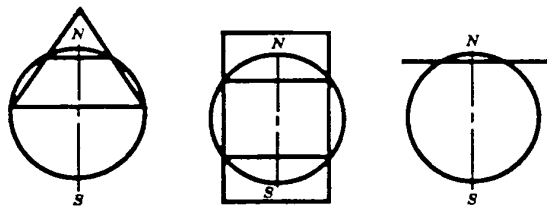
Figure 5.



When these developable surfaces are used, the map is truly accurate only at the line or point where the cone, cylinder or plane touches the spheroid. This line or point is said to be *tangent* and the *standard line* or point.

To improve the accuracy of maps made from these geometric forms, the points of tangency can be doubled:

Figure 6.



This procedure offers two lines on the projection which are accurate, the lines at which the surface intersects the spheroid. This intersecting condition is called *secancy* and the accurate lines are called standard lines or, if the projection is oriented north and south, *standard parallels*. Secancy provides a predictable error factor for the areas between the intersections and for some distance outside of their boundaries.

The following pages acquaint you with the general characteristics of four different projections, the Lambert Conformal Conic, the Albers Equal Area Conic, the Polyconic, and variations on the Mercator, all of which USGS and SCS use extensively.

Figure 7. The **Lambert Conformal Conic Projection** as it transects the earth. Two points of tangency are used to improve the accuracy of the planimetric map. This projection results in a map with parallels represented by unevenly spaced, concentric circles more closely spaced in the center of the map. It is used most often for countries with predominant east-west expanse. Lambert presented it in 1772.

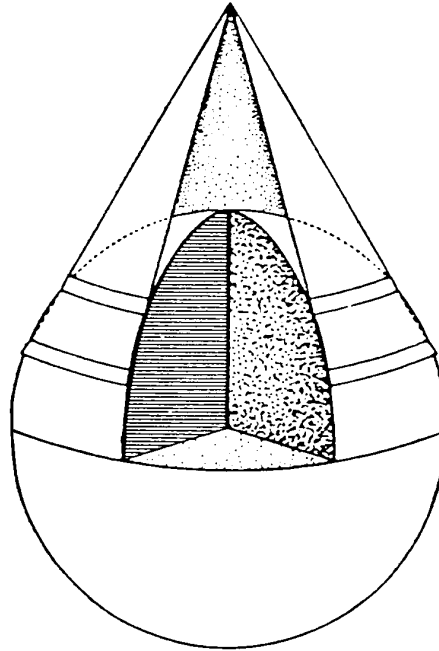


Figure 8. Lambert Conformal Conic Projection, with standard parallels 20° and 60° N. North America is illustrated here to show the change in spacing of the parallels. When used for maps of the conterminous United States or individual states, standard parallels are 33° and 45° N.



DRAFT

Figure 9.

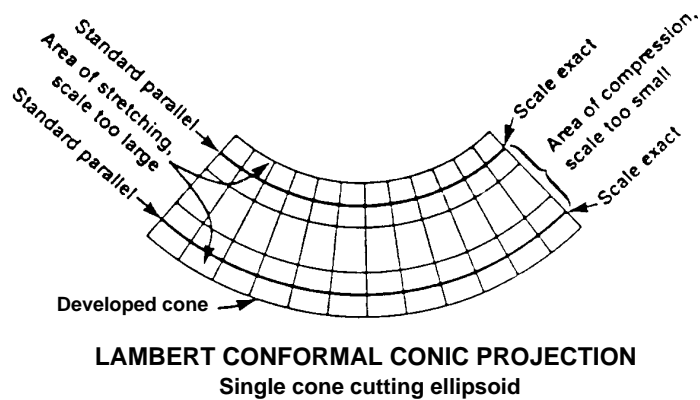
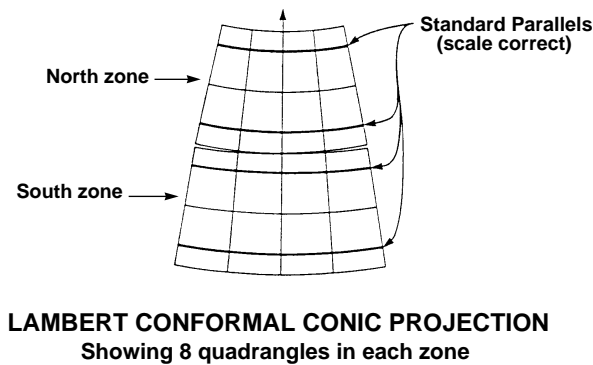


Figure 10.



A note should be made here that it is dangerous to try to identify a projection by visually scanning its characteristics and without deeper analysis, especially at larger scales. You should verify the projections of map sources by inspecting marginal data. On USGS 1:24000 scale maps, the projection (and grid) note should be found with the source data in the lower left margin of the sheet. On the 1:25000 scale maps produced by USGS/DMA, the projection note will be in the lower left legend, as on the 1:24000 scale sheets, or in the lower center, below the bar scale. On county highway maps, all done by various agencies and companies, placement of the projection note is unpredictable, although usually in the lower margin. Sometimes, it is not stated. For GIS products such as GRASS, the coordinate system used in digitization should be stated in the header.

Grids:

We have already defined grids as arrays of lines forming rectangles which simplify map measurement and direction determination. There are a variety of grids in use in the world today, ranging from the simple ABC...123 versions found on state highway maps and

DRAFT

applicable only to that map to the more complicated military versions, which can be applied to large areas on many maps or world wide. We will describe only two here, the commonly used Universal Transverse Mercator(UTM) grid and the State Plane Coordinate System(SPCS).

USGS 1:24000 scale maps carry both grids, as do the 1:25000 sheets. Highway maps may carry either. On the federally produced maps, grid identification is usually in the lower left marginalia.

The following three pages are excerpts from the Department of the Army Technical Bulletin TM 5-241-1, Grids and Grid References, June 1967 (now out of print). They outline the basic data for the UTM grid. As you can see, the employment of the principle of secancy provides two lines of unity scale factor (correct scale). Between these lines, scale decreases, beyond them, it increases rapidly. The six degree zones may be extended only 25 miles beyond their longitudinal limits before error becomes unacceptable.

The second grid we use frequently is the State Plane Coordinate System, designed for surveyors and engineers. As its name implies, each of the United States has its own SPCS and a grid change must occur along state lines. In addition, to control permissible error, 39 states (excluding Connecticut, Delaware, Maryland, Montana, Nebraska, New Hampshire, New Jersey, North Carolina, Rhode Island, South Carolina, and Vermont) have been subdivided into multiple north-south or east-west zones (see Figure 2).

Figure 11. State Plane Coordinate System, Follows NAD-27. This system uses several projections varying from state to state. It uses the Lambert Conformal Conic for most of the contiguous states, the Transverse Mercator (Conformal Cylindrical) for Wyoming and the Hotine Oblique Mercator for parts of Alaska of which all are based on the Clarke 1866 ellipsoid.

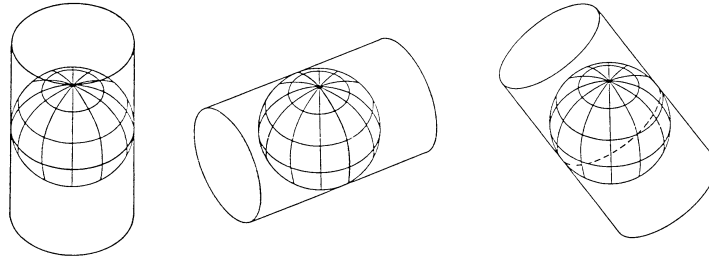


These grids are based on one of three projections, depending on the geographical orientation of the state. The Lambert Conformal

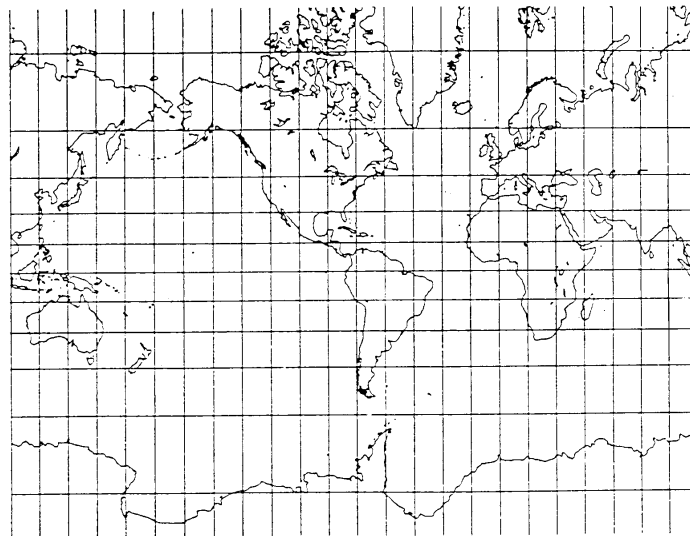
DRAFT

Conic (figure 6) is used in states that are long in the east-west direction or for states which prefer to be divided into several east-west zones. The Transverse Mercator projection is used in states (or zones within states) that are long in the north-south direction. Oblique Mercator is used in one zone of Alaska, the Panhandle, where neither of the above directional conditions apply. The previous figure shows states where each projection applies.

Figure 12. A **cylindrical projection** has both parallel lines of latitude and parallel lines of longitude. The meridians are equally spaced while parallels are unevenly spaced, closest near the center of the map. The scale is true at the tangency only (at both tangencies if the cylinder transects the globe).



Rhumb lines (loxodromes) are straight lines and the poles are at infinity, causing great amounts of areal distortion in the polar regions. This type of map is and has been used for navigation as courses can be plotted as straight lines. Mercator presented this projection in 1569.



When dealing with either SPCS or UTM grid coordinates, remember that the coordinate values will change if the grid is converted from North American Datum 1927 to North American Datum 1983. This is because plane coordinates are derived from geodetic coordinates, the defining constants of several SPCS zones have changed, and a unit of measure change from feet to meters will be made in the SPCS system.

When using source maps for GIS or for any other purpose, be careful to read the sets of coordinate values along the neatline very carefully. It is easy to confuse ticks when two or more systems are

DRAFT

shown. Read the legend data in the margin and remember that the NAD 27 SPCS ticks will have values labelled in consistently sized numerals and will be labelled “feet.” UTM grid values are in dual sized numerals followed by a lower case “m” for meters and a direction. Graticule ticks are labelled with degree and/or minute and second marks. All systems are differentiated by tick placement on the graticule or by weight of line.

Datums:

The last topic we need to consider is the subject of datums. As we have previously defined, a datum is the basis for calculation of further quantities. In the U.S. mapping, cartographers are concerned with two kinds of datums, one for the horizontal surveys of the land and one for the vertical determination of heights above sea level. We shall consider only the first here.

For over 60 years, the United States, Canada and Mexico have based surveys and maps on North American Datum 1927 on the Clarke 1866 ellipsoid. NAD 27 has an initial point at Meades Ranch, Kansas, in the area where major transcontinental survey chains (east-west and north-south) cross. These chains and other surveys were mathematically adjusted to this origin. Other datums are used in foreign areas, such as Indian Datum on the Everest Spheroid for India and European Datum on the International Ellipsoid.

In 1980, the U.S. adopted a new estimate of the size and shape of the earth called the Geodetic Reference System (GRS) 1980. Using this new ellipsoid, NAD 27 has been adjusted to become NAD 83(adopted in 1986), the preferred current datum. This change altered the coordinate values of all points which had been based on NAD 27, affecting graticules and, as we have mentioned, both of our commonly used grids. These differences arise because:

- Surveys were adjusted to NAD 27 on a piecemeal basis; the adjustment to NAD 83 was simultaneous.
- New techniques such as electronic distance measurement and very long baseline interferometry have improved the 1983 network of surveys.
- The GRS 80 is a better approximation of the size and shape of the earth than the Clarke 1866 spheroid.

Chapter 3 *Display*

Objectives:

Most of the map preparation within a GIS is done on a graphics screen. This allows for cartographic and analytic experimentation. It also is much cheaper than printing various paper maps and more effective than some textual representations of data. This chapter will introduce you to the display commands within GRASS.

At the end of this chapter you should be able to:

1. Start and select a graphics monitor in GRASS for your machine.
2. Select, unlock and force the release of a graphics monitor.
3. Use the following GRASS commands:
 - d.mon [start=, select=, unlock=, stop=]
 - d.frame [-e], d.erase [color]
 - d.rast [-o]
 - d.what.rast
 - d.what.vect
 - d.where
4. Start and stop GRASS.

DRAFT

Starting GRASS:

Normally, you would probably start GRASS from the graphics terminal so as to initiate the graphics driver. We will use this method to start GRASS, however, there are several ways to start GRASS from other than the graphics terminal. The steps used to start GRASS on the graphics terminal are as follows.

1. Login to the graphics terminal.
2. Start X-windows (OpenWindows or Motif window managers).
3. Start GRASS by typing *grass4.0*.
4. Choose a Location and Mapset.
5. Run the *d.mon* program in GRASS.
 - b. Start a graphics monitor (1).
 - i. list the available monitors.
 - ii. start one of the X-windows drivers (i.e. *x0*)
 - b. Select a graphics monitor (3)
 - c. Quit *d.mon* (*[return]*).

GRASS d.mon program menu.

```

MONITOR MENU

Making sure that the graphics monitor is running

  1 - start a graphics monitor
  2 - stop a graphics monitor

Choosing a graphics monitor for your graphics

  3 - select a graphics device for output
      (currently selected monitor: x0)

  4 - relinquish control of the graphics device
      (let someone else use it)

RETURN - quit

>

```

You are now be ready to display to the X-graphics window.

Note: *A unix shell script can be written to start the graphics window without having GRASS running. A description of how to do this wil be included in a future Appendix. Also, a common way to use GRASS is to devote the graphics terminal solely to graphics display and to use a “dummy” terminal to enter commands. If you wish to use this method, 1) exit GRASS, 2) log on to the “dummy” terminal, 3) start GRASS on the “dummy” terminal and, 4) use **d.mon** to reselect the monitor.*

For starting GRASS on your current system, use the following

DRAFT

steps:

<pre> Console (Graphics Monitor) mach login: class Password: learngrass Logs you onto the system. class on mach_name: openwin Open Windows starts and windows appear... move mouse to cmdtool window, click left and then type: mach_name d.start.x0 Note: This is a special script which may not exist on your system. Consult the GRASS 4.0 manual for information on developing such a script. The X-windows graphics driver starts. Choose full-size from the window header menu to enlarge the window. Next, move to the "dummy" terminal to start GRASS. </pre>	<pre> Dummy Terminal mach login: class Password: learngrass grass on mach_name: grass4.0 Select your location and mapset then hit [Esc] to continue. Mapset <name> in Location <name>. GRASS 4.0 > d.mon select=x0 Mapset <name> in Location <name>. GRASS-GRID> You may now enter any valid GRASS command. </pre>
---	---

DRAFT

You should now have GRASS started and ready to use.

Displaying Data Layers:

***d.rast* and *d.vect*:**

GRASS has several tools for displaying data layers (raster and vector data). The two most commonly used are *d.rast* (display raster map) and *d.vect* (display vector map). *d.rast* will display grid cell maps in the current graphics region and *d.vect* will display vector (*v.digit*) maps in the current graphics region. First try using the *d.rast* command:

```
d.rast elevation.dem
```

The elevation data layer should appear on the screen. Now see what happens when you *d.rast* another map. *d.rast geology*:

```
d.rast geology
```

The second map overwrites the first. *d.rast* will always overwrite what is currently on the graphics screens if no options (flags) are specified. Next, clear the screen using *d.erase*.

```
d.erase
```

Now look at a vector map. Use *d.vect* to display the roads data layer.

```
d.vect roads
```

d.vect has 9 options for color. Try the same command again but this time attach *color=red* to the end.

```
d.vect roads color=red
```

The roads appear in red. Use *g.list vect* and *g.list rast* to see what vector and raster files are available to you in your current location.

```
g.list vect  
g.list rast
```

Now experiment with the commands that you just learned, *g.list*, *d.rast*, *d.vect* and *d.erase* to view several of the data layers.

***d.what.rast*, *d.what.vect* and *d.where*:**

GRASS allows you to query raster and vector maps on the screen for attribute information and also for location. It does this

with three commands: *d.what.rast* , *d.what.vect* and *d.where*. Execute the following commands to use *d.what.rast*.

```
d.erase
d.rast elevation.dem
d.what.rast
```

Now move you mouse around the screen and click on a area. GRASS reports to you the category value at that location. *d.what.rast*, when executed with no parameters (flags), reads from the data layer which is currently displayed. However, you may specify up to 14 layers for *d.what.rast* to query at once. To try this execute the following:

```
d.what.rast map=elevation.dem,geology,soils,landuse,owner
```

GRASS reports to you the category values for each of the 5 layers. Next, use *d.where* to read only locations from the screen.

```
d.where
```

Move to any location on the screen and click. GRASS reports to you the coordinates that you have pointed to.

***d.rast -o* (Raster Overlay):**

Some maps are not complete coverages, but rather resemble patchwork quilts of data and no-data areas. First *d.rast* the landuse data layer and then use *d.what.rast* to spot check several of the areas.

```
d.erase
d.rast landuse
d.what.rast
```

It is often valuable to view two maps together, one a patchwork and the other a complete coverage. GRASS does this with the *-o* flag and the *d.rast* command. First, display the complete coverage using *d.rast*. Next, overlay the patchwork layer on (top of) the complete coverage. Think of the no-data areas as holes, GRASS allows the complete coverage to show through the holes.

```
d.rast elevation.dem
d.rast -o landuse
```

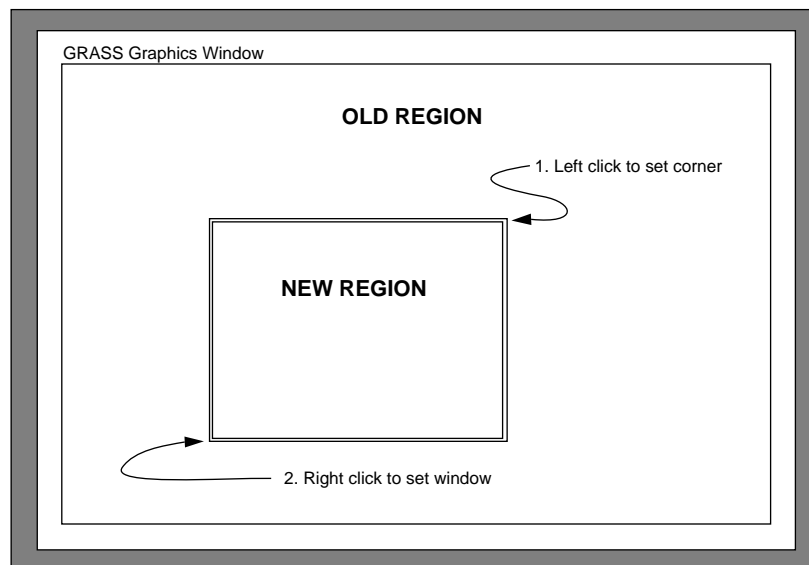
Experiment with several layers using the *d.rast -o* command.

DRAFT

Changing Regions:

So far we have looked at various layers from one region. GRASS has the ability to zoom in and out on data by means of a definable rectangular region, a “window on the world”. To set a new region use the *d.zoom* command. See the diagram below for setting a region.

```
d.rast elevation
d.zoom
```



Before displaying another map, use *d.frame -e* to clear the screen and set a new frame. This not only gives you a clear screen to work with, but also resets the working region. If you did not clear the screen after setting the region, your working region would remain the same.

```
d.frame -e
d.rast elevation.dem
```

Multiple Windows on a Screen:

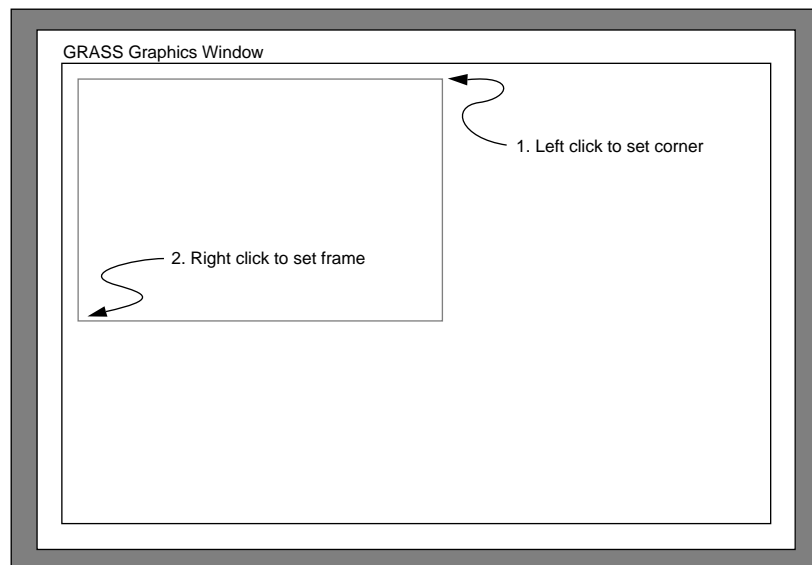
Another useful feature is the ability to create more than one frame on the graphics screen to display graphics to. You use the *d.frame -c* (create) and *d.frame -s* (select) commands to do this. First, reset the window to its default size with the *g.region* com-

DRAFT

mand.

```
g.region default
d.frame -e
d.rast elevation.dem
```

Now make a new frame using *d.frame -c*



Set the new frame. At this point you have created a new frame on the screen but don't have control of it. If you were to *d.rast* a map, the full screen would be used, in fact there is a frame called *full_screen* which covers the entire screen. This is the frame which is used for most graphics. It is created (and selected) by *d.frame -e*. To gain control of your new frame use *d.frame -s* (select).

```
d.frame -s
```

Point to the frame, left click to select it, and then right click to accept it if you have highlighted the one you want. You now have control of the new frame. Erase the contents of the frame using *d.erase*. Using *d.frame -e* now would erase your entire screen and reset the *full_screen* frame.

```
d.erase
```

d.rast a map.

```
d.rast elevation.dem
```

The map appears within the frame. Notice that the region

remains the same from frame to frame, only the size at which the map is displayed is affected.

Try several new frames and display various maps within them. Be careful not to get two frames exactly on top of one another as GRASS could get confused and cause you difficulties.

Summary:

The tools demonstrated in this chapter are some of the most common and useful display commands within GRASS. Learn how these work and get comfortable using them as they are key to almost all operations in GRASS. Other display commands which might be of interest are: *d.3d*, *d.sites* and *d.label*. Use the *g.help* and *g.manual* commands in GRASS to get information on how to execute these commands.

D
R
A
F
T

Chapter 4 Digitizing

Objectives:

In this portion of the course you will become acquainted with the several methods of data input for developing a GIS database. Although various percentages can be found, approximately 75% to 95% of the time spent using a GIS is devoted to data development. Since development of the database involves so much of the actual time spent using GIS software, it seems appropriate to start the first hands on exercises with digitizing.

Digitizing, whether manual or automated, is a crucial component of a GIS. Since your data can only be as accurate as your digitizing, several techniques should be employed to assure the highest possible quality of your data. These techniques will be discussed along with examples for using *v.digit*, the GRASS program used to develop data layers manually.

At the end of this chapter you should be able to:

1. Define the following terms:
 - registration
 - digitizing threshold
 - control points
 - residual
 - topology

2. Use the following GRASS commands:
 - *v.digit*
 - *v.import*
 - *v.support*
 - *v.out.ascii*
 - *v.in.ascii*
 - *v.to.rast*

DRAFT

Digitizing Overview:

The following pages of this chapter are drawn from two GRASS tutorials in the GRASS 3.1 manual. Editing has been made to accommodate the GRASS 4.0 commands and changes.

Definition:

The specific purpose of digitizing in GIS becomes clear when the general definition, process and purpose of digitizing are considered in relation to map information.

Two definitions for the word digitize are in common use. The first states simply that to digitize is to “convert [analog] data into digital form.” A more detailed second definition says that digitizing is “translating analog measurements [of data] into numerical descriptions, based in digits, in a scale of notation.” A scale of notation merely determines the electronic ratio of the assigned value to the real world value.

Digitizing Process:

There are manual and automated processes of digitizing. The term scanning is applied to automated methods. Both techniques have three basic steps.

1. Place the map or document to be digitized on the digitizing surface. This is commonly a table or tablet having a grid of wires imbedded near the surface to which voltages are applied, in the manual case. For scanning, an automatic roll feed is likely to be employed to move the document over a more limited surface area that does not require a wire grid.
2. Registration or Calibration of the document to the digitizing surface is required next. In manual digitizing, this amounts to recording grid voltages corresponding to the positions of control points located on the data source to be digitized. Control point locations are recorded automatically along with other data during scanning. These readings are used to provide the scale of notation for digital quantities translated from voltages. A default scale such as table inches is available in either method. Scanning relies on this type of scale.

D
R
A
F
T

When units other than the default are specified by an operator, a coordinate transformation algorithm is required to convert table units to those desired. Scanned data is transformed after data is digitized with the default scale. This could be done in manual techniques as well.

It is important to note that if the registered positions of control points do not agree with those calculated for the surface by inverse transformation algorithms, the recorded data is usually scaled in one or both dimensions of the digitizing plane. Quite simply, if a distance of 14 cm is recorded between two horizontal control points and the distance should be 14.5 cm for an accurate source, the transformed coordinates along the line connecting the two points are scaled to fit the accurate measure.

3. Digitize features of interest on the document by moving a recording instrument over them and recording their locations on the digitizing surface. Manual operators will likely use a device such as a puck which transfers voltages indirectly through magnetic fields. A scanning head which measures reflected light in terms of voltages is automatically passed over data on a scanning surface. Digitized information is stored in various computer files.

Purpose of Digitizing:

Different kinds of data are digitized for many uses but the primary purpose of digitizing is data automation. An automated system is one "in which many or all of the processes are automatically controlled as by electronic devices." Computers rely upon automatic processes to display and manipulate data stored in files electronically.

Since we are in the business of GIS, we are concerned with constructing an automated digital cartographic database for reliable and efficient display or analysis of map information. Digitizing is a method by which we accomplish this purpose. We use digitizing to automate maps.

The process of digitizing can be costly. Fortunately, it is not the only way that digital data can be acquired. It is always a good idea to search for available data that has already been digitized and is suitable for the purpose at hand. TIGER data from the U.S. Census Bureau or GEODATA from USGS, containing roads, hydrology, and other features, may be purchased for a fraction of the cost of digitiz-

DRAFT

ing it.

Maps for Automation:

In order to accomplish the purpose of digitizing in GIS, maps must first be obtained. Since digitizing is only a means of translating map information, the maps themselves must satisfy all cartographic requirements of the database. Maps which are properly prepared for digitizing with consideration given to database requirements and digitizing technique will facilitate the digitizing process.

A new map composed of a variety of existing maps or information is called a compilation or manuscript in cartography. Those with a theme are called thematic maps. Compilations for digitizing may be simple or complex depending upon the number and types of features present. A simple manuscript may contain only roads for a county, whereas a complex one may depict highly erodible land within 500' of perennial streams.

Once digitized, compilations are represented in the database as a set of geographically registered or referenced layers called coverages, tiles, or modules. The modules may be a direct representation of all data on a source manuscript or they may be components of one or many manuscripts, given the manipulative capability of GIS systems.

Topology:

Map features on compilations are represented in the database by digitally encoded topology. Topology is the form or data structure given to map entities. It may be a series of file entries or files. Topology allows features to be distinguished from or related to others in the data set during coverage manipulation.

Topology is composed of geographical as well as characterizing information. The geographical component of topology relates map features to real world coordinates. It provides the basis for geometrical analysis such as calculation of distances and areas. The characterizing component provides other information about map features. This data is called attributes or categories. Soil name or texture would be examples of this component.

There are three basic topologic entities common to vector and raster types of GIS data. They are the point, line and area. Complex

D
R
A
F
T

topology will vary between types as well as between GIS packages.

Points are represented simply by a coordinate pair and perhaps an identifying number. Vector points and raster cells are the most basic elements of topology. Vector coordinates are in x, y format. Cell coordinates are in row, column format representing grid-cell position. Geographic control points called tics and labels for sites or areas are represented by this element. Points are the building blocks for more complicated map features such as lines and areas.

Lines are a collection of ordered points or cells. They are crucial in constructing vector polygon topology. Linear features such as roads and boundaries or neatlines are shown with lines. A neatline is a line surrounding or limiting the image area of a map. Vertices are points in vector lines. Nodes are endpoints of vector lines.

Polygons are used to represent areas. They may be a collection of contiguous cells or a closed plane figure bounded by vector lines. Lines used to form areas must connect precisely at nodes, otherwise areas and attributes will “leak” into adjoining polygons. A neatline must exist in order to form vector polygon topology at map edges. Topology may affect module sizes if it is dense or complex, due to system limitations on available memory, especially when layers are being manipulated. Compilations may need to be reduced in geographical extent or simplified to compensate for the effects of topology.

DRAFT

Cartographic Requirements:

In GIS, maps must meet certain cartographic requirements. They are required to have a projection, control points, a datum, map scale and must meet standards of accuracy. If compilations do not fulfill these requirements, then neither will the resulting coverages. That old saying “GARBAGE IN.....” applies here as well as anywhere else.

Planar representations of the curved surface of the Earth have distortions in scale and direction that increase with geographical extent. These distortions are compensated for by the use of map projections which represent meridians and parallels as curved lines. A map projection provides the mathematical and graphical transformation required between coordinates on the ellipsoid of the earth and the map plane.

Projections are designed for use over specific geographical

extents. See chapter 2 regarding map projections. An Albers projection is suitable for the coterminous United States. Smaller areas may be placed in UTM, Lambert or State Plane projections. Compilations that extend across projections, UTM zones for example, will develop errors in areas and distances. The map projection for a compilation must be adequate to cover its' geographical extent.

Control points are necessary to geo-reference compilations. They are used during map registration in digitizing to provide the basis for coordinate transformations. Map features should not extend beyond the limits of control to avoid transformation error. Four or more control points are recommended for a rectangular area. Fewer than this number will reduce the ability of transformations to compensate for inaccuracies of the base scale in two dimensions.

Control point locations on compilations to be digitized are affected by the datum used. NAD 27 is based on the Clarke 1866 ellipsoid originating from a point at Meades Ranch, KS. NAD 83 is Earth centered, based on the GRS 80 ellipsoid which is a best fit to the geoid. Control points from NAD 27 are still the most common in use, due to the fact that USGS has not re-drafted quadrangle maps to fit NAD 83 control.

All maps will have a map scale or representative fraction (RF) to indicate the relationship between map distances and distances on Earth. A mapscale of 1/24,000 is a larger mapscale than 1/2,000,000 but one map unit at a mapscale of 1/2,000,000 is a larger distance than one map unit on a 1/24,000 scale map. Units are unimportant in fractional scales because they always drop out. One inch to 24,000 inches is the same RF as 1 centimeter to 24,000 centimeters. The RF of a compilation is important because it determines what the accuracy of a map should be. RF can also be written like 1:24,000.

National map accuracy Standards for the U.S. states that a certain number of test points must be accurate within 0.03" for maps with scales larger than 1:20,000 and within 0.02" for maps with scales less than or equal to that scale. Soil or forrest boundaries are not considered test points. SCS has specified that soils data will be accurate within 0.01" at whatever mapscale is being used. This would correspond to 20' at a RF of 1:24,000.

Compilations containing data from maps with differing scales should only be considered as accurate as the least accurate scale used. Error migration will occur when database modules are plotted at scales other than those of the source map. Errors will increase or decrease with mapscale. Points located accurately within 0.01" on a

D
R
A
F
T

map with a RF of 1/100,000 will be within 0.02" on a map plotted at 1/50,000 and within 0.005" when plotted at 1/200,000.

Map coordinates may be specified in a variety of units. The type of map units determines the size or range of coordinate values stored for map features. If the numeric precision of the GIS package being used is inadequate for the task, map accuracy will be in error. Double precision numbers or data shifts may be required in order to capture data within required accuracy.

Map resolution is the minimum distance allowed between map features or the minimum size of a feature that may be represented. When given in ground units it is called ground resolution. The resolution for map coverages in the database will depend upon the resolution of the digitizing equipment or parameters set by the digitizer operator.

High resolutions are represented by smaller numbers than low resolutions. A digitizing surface must have the resolution necessary to capture data within standards of accuracy. Map resolutions set during raster digitizing must not go lower than the accuracy required and cannot exceed that of digitizing equipment. Resolutions which are higher than necessary may result in increased data storage requirements if the quantity of data collected cannot be generalized.

Compilation modules must be edgematched physically to all adjoining modules before digitizing. This is done to insure the continuity of features or attributes extending across map boundaries. Lines for map features terminated by the neatline, must intersect it and be within 0.005" (centerline to centerline) of matching elements on adjacent modules. Edgematching is usually completed by computer, since drafting or digitizing within this tolerance requires special equipment or abilities.

Compilation Materials:

Materials used in map compilations should be accurate and stable. Paper can shrink or swell on the order of 1% with temperature or humidity. Vellum is more stable but film or mylar is recommended for use. Paper can provide satisfactory results when used in a controlled environment over short durations. USGS topographic quadrangles or orthophotoquads and maps should be used to obtain accuracy. Only orthophoto imagery is recommended for use because mosaics and aerial photographs are not corrected for relief. Never use photocopies.

DRAFT

Only fine line drafting equipment should be used. Choose colors that reduce confusion and eyestrain or aid scanning. The physical size of compilations must not exceed the usable portion of digitizing surfaces.

Use as few overlays as possible since they tend to bubble or slide. They also can be confusing, difficult to see through and cause eye fatigue. Overlays must contain precise control in order to match base maps and be registered properly. Use pinbars to keep bases and overlays together during compilation if possible. If the geographical extent or types of map features are limited, overlays can be eliminated by compiling maps that contain several types of topology. This technique is called integrated terrain unit mapping. Data types may be digitized separately or together later to form layers in the database. The most accurate feature common to several layers should be used as a template for all layers so that it will always be the same in the database.

D
R
A
F
T**Equipment:**

Digitizing equipment must be functioning properly and be capable of capturing data with acceptable precision and resolution. Most equipment has diagnostic tests which can be run. In addition, operators need to possess the skills required to operate the equipment effectively. The best compilations for digitizing can not compensate for inadequate equipment or poor technique.

All of the equipment shown for a typical GIS in the course module for the History and Overview of GIS can be considered digitizing equipment, since it is or may be used somewhere in the digitizing process. The computer is required to run the digitizing software and manipulate digital data. The graphics display and keyboard provide a user interface with the equipment and database. Printers and plotters are used to obtain hard copy for data verification. Only digitizing surfaces and recording instruments require further mention at this point.

Digitizing tablets designed for menu operation generally do not have the resolution required for digitizing in GIS. High resolution tables or tablets with an accuracy of 0.005" or less or scanners capable of 200 Dots Per Inch (DPI) or more should be used.

The most commonly used instrument in manual digitizing is the puck. It is much easier to position than a stylus over map features

since it rests on the table and has crosshairs. Pucks are usually more versatile because they contain more function keys. Scanners record data automatically with a scanning head. Since points can be entered on digital maps using a cursor, it qualifies as a recording instrument. A keyboard or mouse is used to position the cursor. A keyboard is more accurate than a mouse but is more difficult to operate.

Digitizing Technique:

Digitizing technique involves the proper use of equipment and procedures. It must be developed. Some methods are quicker than others but may be more difficult to accomplish. The best technique is the one which provides an accurate dataset for the least cost. This requires that editing be part of digitizing technique. Scanning is automated, fast and accurate. Scanned data may require more editing but the time saved collecting the data is well worth it. The following paragraphs pertain to manual methods.

- The manuscripts to be digitized should be secured to the table. Overlays must be positioned accurately and secured as well. Pinbars for punched bases and overlays will help keep overlays in the proper position.
- Careful registration must be used in order to obtain the lowest possible transformation error. RMS (Root Mean Square) errors less than or equal to 0.007" have yielded good results. The more control used, the more chance there is for operator error or inclusion of bad points. This can result in a higher RMS or residual error. Acceptable Cell Residual errors for some ideal maps are listed in the table below by scale.

Acceptable cell residual errors for some ideal maps.

MAPSCALE (RF)	RESIDUAL (meters)
1:100,000	20-30
1:50,000	5-10
1:24,000	1.2-2.4
1:4,800	0.5-1.0

- The proper digitizing environment must be set during data capture. Snaps or traps which cause entities to be snapped together if they are within a specified distance, will allow less than super-human operators to connect nodes. Weeds or thresholds can be set to generalize data as it is being collected and reduce data storage requirements. Additionally, the graphics display can be

D
R
A
F
T

set to show potential problems as data is being collected.

- Data capture must be accomplished using edit reducing techniques. Data is generally recorded point by point. For vectors this is called line segment digitizing. Grid-cell is the term applied to raster data. Points should be collected about every 1” and at inflection points. Keep the number of points collected under 500 for vector lines to obtain frequent rest, avoid loss of work in the event of a mistake and allow usage of the data on systems having limitations.
- Two point lines greater than 7.5’ or 0.125 decimal degrees should not be digitized unless they contain interior points. They are not transformed with the proper curvature.
- Use system generated neatlines. Those which are digitized do not edgemark as well.
- Avoid digitizing the same entity twice to avoid extras and slivers.
- Digitize only one type of topology at a time to save time lost switching input modes.
- Edgemark while digitizing by snapping to data from adjoining maps which have been copied into the coverage or are in background. A checkerboard scheme for modules will enhance this method. In lieu of this, for small uncomplicated maps, composite the data from all into a single larger module as it is collected. Clip out maps later.
- Take advantage of automated processes if possible. Stream mode digitizing may save some time provided the proper parameters are set (distance and or time) and the operator has sufficient skill. Spaghetti digitizing can be used with point to point or stream digitizing to avoid the need for node or cell snapping altogether. This type of data is cleaned up later with automated editing to find intersections and fix overshoots or undershoots.
- Leaning on the digitizing surface can create problems. Compilations might get shifted or become unattached, losing calibration. It may also be possible to warp the surface, changing the distance between the grid and recording instrument and thus readings in that area.

- Finally, the data is not digitized until it has been verified and archived. Rely on automated editing as much as possible. Perform screen edits before preparing hardcopy proof plots. Plots require money for time, paper, and ink. Check for missing, mislabeled and extra features or attributes. Edgematching and accuracy must also be validated.

DRAFT

Tutorial *v.digit: Its Use and Features*

Introduction:

GRASS map development entails the production of raster, vector, and support files that can be used within GRASS to represent map features. The GRASS programs that are used in map development are: *v.digit*, *v.import*, *v.support*, *v.in.ascii*, *v.out.ascii*, and *v.to-rast*.

The *v.digit* program is a highly interactive and menu-driven digitizing, labeling, and editing package. Most aspects of digitizing can be done entirely from within *v.digit*. This tutorial discusses all *v.digit* options and walks the user through a sample digitizing session.

Contents:

1. Preparing a Paper Map Suitable for Digitizing
2. Entering the *v.digit* Program
3. Moving Through the *v.digit* Menus
4. Using the Mouse
5. Digitizing
6. Editing
7. Labeling
8. Windowing
9. Using the Customize Menu
10. Using the Toolbox Menu
11. Diagnosing Problems

1. Preparing a Paper Map Suitable for Digitizing:

Although a great amount of information is stored on paper maps, this information can be much more easily manipulated in a computer database. To get map data into a GRASS database, map data (analog data) is converted into digital data by tracing relevant map features using a digitizer. The digitizer and *v.digit* convert map data into a GRASS vector format.

Frequently, map data cannot be digitized directly from original paper source maps. Paper maps will generally contain many types of

D
R
A
F
T

information. The person that is digitizing must know exactly what information is to be digitized. It is therefore generally necessary for the user to prepare a map(s) containing only the information that is to be digitized before beginning to digitize.

A detailed GRASS tutorial entitled Map Preparation explains how maps should be prepared maps for digitizing and should be read prior to digitizing.

Have the following items before beginning to digitize.

- a stable medium, i.e., mylar, plastic, etc. on which the features to be digitized appear clearly and accurately,
- at least 10 marked points on the medium corresponding to UTM, state plane, x,y, or lat/lon coordinates,
- a window in UTM, x,y, or lat/lon coordinates defining the area being digitized,
- a window in UTM, x,y, or lat/lon coordinates defining the area of the resultant cell file,
- masking or drafting tape to secure the medium, and
- a planned final window and resolution for the cell file.

2. Entering the *v.digit* Program:

To walk through the examples in this tutorial you will need a digitizing workstation consisting of a graphics monitor, a pointing device (mouse), a digitizer, and optionally an ascii 'dumb' terminal with keyboard. Currently, *v.digit* runs on GeoGraphics, Altek, Calcomp, Hitachi, Numonics, and Kurta digitizers. If you are not already in GRASS, enter GRASS by typing the command:

```
grass4.0
```

See Chapter 3 for instructions on starting GRASS. You will then be asked to enter a location, mapset, and database with which to work. Select *spearfish* as the location and *PERMANENT* as the Mapset. The Database should be set to the directory in which your GRASS data is stored (i.e. /home/grass4/data).

```
spearfish
PERMANENT
```

Follow the instructions in Chapter 4, Display for starting GRASS. Start the graphics terminal following the startup instructions and then run the GRASS program *d.mon select=x0* to link the

graphics monitor to your present session.

When you are back at the GRASS prompt you can then start the digit program by entering the command *v.digit*.

v.digit

The first message to appear one inside digit is a listing of the available digitizers on your system (i.e., those that were installed by your system manager). Note that you have the option of choosing to use no digitizer by selecting option *[none]*. The digit program may be run both with and without a digitizer. Many of its options are still available if no digitizer is chosen. For the purposes of this tutorial, chose the digitizer *[kurta]* here, by typing the number associated with it at the prompt.

Next you will be prompted for the name of the vector file to be used during the digitizing session. Both existing vector files and new vector files can be chosen. Entering the command *list* will show all the existing files under dig directory in your location.

list

Type a new vector file name and hit *<return>*.

myfile

You will now be asked if the vector file on which you have chosen to work is referenced in that LOCATION's coordinate system. The coordinate systems used in GRASS are the Universal Transverse Mercater (UTM), State Plane Systems, x,y, and lat/lon. Mixing of the two coordinate systems in the same LOCATION is not allowed in GRASS. If you choose a coordinate system other than that LOCATION's coordinate system digit will automatically exit and return you to the GRASS prompt.

The next screen (resembling that shown below) asks the user to enter information about the file to be worked on. This information is stored along with the user's mapset for use in future GRASS sessions. If this is a new file, this screen will contain blank lines. It is likely that the user will later need this information both inside and outside of *v.digit*. In addition, the user must fill most of the lines on this screen to continue with digit.

To move around this screen, use *<return>* or "*ctrl j*" to descend a line, "*ctrl k*" to ascend a line, and *<return>* or "*ctrl j*" to reach the top line from the bottom line. When happy with your responses,

press the <ESC> key to continue to the next screen.

v.digit information screen.

Provide the following information:	
Your organization	Arkansas Archeological Survey
Today's date (mon, yr)	8/90_____
Your name	grass_____
Map's name	Arkansas Base Map_____
Map's date	1967_____
Map's scale	1:500000____
Other info	
Zone	15_____
West edge of area	350000_____
South edge of area	3645000_____
East edge of area	810000_____
North edge of area	4045000_____
AFTER COMPLETING ALL ANSWERS, HIT <ESC> TO CONTINUE	

The coordinates on this page define a “digitizing” region. Once this region is established, only data that falls within this region will be recognized by *v.digit*. Therefore, the user should be careful to make this region large enough to completely cover the area of interest. A good practice is to make this region the same size as the default region for this location. This default region is located in the PERMANENT directory of your present location. You can find out what the default coordinates for your current location are by running the GRASS *g.region* command while you are outside of the *v.digit* program.

[Esc]

After the user hits the [Esc] key, the following message will appear:

v.digit - continue screen.

Shall we continue? [y]

[return]

Here simply enter *y* to continue, or *n* to exit the digit program. If you exit digit at this point, your title page will not be saved.

Note: If the digitizer you chose was [none], you will not see the following messages. Instead, you will continue on to *Moving Through the v.digit Menus*, section 3 of this tutorial.

Tape the map securely to the digitizing surface, making sure that all areas to be digitized fit within the reach of the digitizing arm or pointer.

If this file has been worked on before, it may contain registration points that were used in the last digitizing session. You may use these same registration points, add additional ones, or add completely new points in this session. If this is a new file, you will not see the following message:

v.digit - last session registration points screen.

Use set of registered points from last session (y/n)?

Enter *y* to use the same registration points you used in the previous session, or enter *n* if you wish to register your map using new points.

Now input the coordinates of the registration points appearing on your map overlay. While *v.digit* requires that only four points be registered, it is a good practice to enter at least eight registration points, since you will probably have to eliminate some points later for various reasons. These points should be distributed throughout your map. Also, it is convenient to enter the coordinates in some order. For example, you might begin by entering the coordinates of the registration point nearest the lower-left corner of your map, and continue entering points in a clockwise direction until all of the coordinates are entered.

D
R
A
F
T

v.digit - screen for entering registration points.

```

MAP REGISTRATION POINTS
Enter 4 - 10 points:  points registered 0

Point #      X coord      Y coord
1      590000.00__  4915000.00__
2      590000.00__  4927000.00__
3      599000.00__  4927000.00__
4      600000.00__  4927000.00__
5      609000.00__  4927000.00__
6      609000.00__  4915000.00__
7      600000.00__  4915000.00__
8      599000.00__  4915000.00__
9      0.00__      0.00__
10     0.00__      0.00__

Enter 0.0 to delete a coordinate pair.
Those marked by '*' are registered.

AFTER COMPLETING ALL ANSWERS, HIT <ESC> TO CONTINUE
(OR <Ctrl-C> TO CANCEL)

```

Here, enter the coordinates of the registration points. Move down through the menu by entering *[return]*. If you make errors while entering points, use “*ctrl k*” to move back one entry or “*ctrl h*” to move the cursor left. re-enter any mistyped points. You may, at most, enter ten points. Once you finish entering the coordinates of your registration points, press the *[Esc]* key to continue.

[Esc]

The following screen will then appear:

DRAFT

v digit - point registration screen.

```

-----
                POINTS TO REGISTER
POINT          EASTING (X)      NORTHING (Y)
* 1.           590000.00        4915000.00
* 2.           590000.00        4927000.00
* 3.           599000.00        4927000.00
-> 4.           600000.00        4927000.00
5.           609000.00        4927000.00
6.           609000.00        4915000.00
7.           600000.00        4915000.00
8.           599000.00        4915000.00

Number of points: 8,   Points registered: 0

-----
                USING DIGITIZER CURSOR FOR INPUT
Key <1> - register point,      Key<4> - add more points
Key <2> - skip point,         Key<5> - accept residuals
Key <3> - unregister point
-----

```

This screen provides you with many ways of dealing with your registration points. To register a point, move the pointer of the digitizer onto the registration point and hit key [1]. As you register points, an asterisk will appear beside each point registered. Once a point is registered, *v digit* will automatically take you to the next point. If you don't want to include a particular point in the registration, enter key [2] to skip to that point followed by key [3] to unregister it. You may for whatever reason wish to enter the coordinates of additional points that do not appear on this screen. Choosing the key [4] option will bring back the previous screen, enabling you to add more points.

Once you register at least four points, a fourth column will appear listing the residual error associated with their placements. Each residual represents the accuracy of this point in relation to each of the other registration points. Try to make the overall residual less than 10, and try to make each individual residual fall within 2 integers of the other residuals listed. Residuals can sometimes be lowered by re-entering points, or by eliminating specific points using the option key [3]. Once you are happy with the residuals, choose key [5] to continue. If you return to this file in a different session, register your map using the same registration points if possible.

The next feature allows you to list the coordinates of any point on your overlay, to momentarily suspend display of the coordinates, and check the operation of the foot switch (if one is included with your digitizer). This feature is used to test the positions of the regis-

tration points and to collect the coordinates of points to be used in GRASS site files and in geo-referencing.

v.digit - accuracy check screen.

```
CHECK MAP:  Key '1' to preserve point; Any other Key to continue

Coordinates:          X - Current - Y          X - Saved - Y
0                    267300.95  52141.00      0.00      0.00
```

If you are not happy with what you have done so far, and wish to re-register your overlay, answer *no* (key [2]) to the following prompt. Otherwise answer *yes* (key [1]).

v.digit - accept registration screen.

```
USING DIGITIZER CURSOR FOR INPUT
Key '1' for YES   :   Key '2' for NO
|
If satisfied with the registration, enter 'y', else 'n'
```

Answering *no* will return you to the registration part of digit.

At this point you have entered *v.digit* and registered your map layer; you are now ready to begin work. The “digitizing region” that you created (defined by the coordinates shown on the title page) will almost always be too large a window in which to work comfortably. You will therefore define a “working region” for *v.digit* using the digitizing cross-hairs (see screen below), describing any subset of your “digitizing region” in which you wish to work throughout your *v.digit* session. It will not affect any other GRASS program, nor will it be saved after you exit *v.digit*. It is only used to conveniently zoom in on and display small areas in which you are digitizing. Understanding the differences between the “digitizing,” “working,” and “current” regions is very important.

v.digit - working region definition screen.

```
Identify corners of graphics window.
Locate digitizer cursor on one corner of desired window.
Then hit any digitizer <Key>
```

Use the digitizer cursor to identify the two diagonal corners which will define your working windows.

DRAFT

3. Moving Through the *v.digit* Menus:

All of the options are presented in the Main, Digitize, Edit, Label, Customize, Toolbox, and Window menus.

Only one menu appears on the terminal screen at any given time. Options may be chosen only when they appear on the current menu page. Each menu contains at least two fields: a title field (top field), and a global field (bottom field). The top field displays the menu name, while the bottom field displays global options. Movement between menus is done by choosing a global option found in the global field. For example, the menu now appearing on the terminal screen is the Main Menu. Choosing “D” (for Digitize) from this menu will cause the Main Menu to be replaced by the Digitizing Menu. Each menu has its own set of global options - where each set is a subset of the global options found at the bottom of the Main Menu. To choose a global option, type in the first letter of that option (using upper case letters only).

Take some time to look at the layout of this Main Menu screen and the information it contains. Note that most of the information on it was taken from the title page information entered by you earlier.

v.digit - main menu screen.

```

-----
GRASS-DIGIT Version  4.00                               Main menu
-----
MAP INFORMATION                                         AMOUNT DIGITIZED
Name:      Arkansas Base Map                          # Lines:      0
Scale:     500000                                       # Area edges: 280
Person:    grass                                       # Sites:      0
Dig. Thresh.: 0.0300 in.
Map Thresh.: 381.00 meters
-----
OPTIONS:
Digitizer:  Disabled
-----
Digitize Edit Label Customize Toolbox Window Help Zoom Quit * ! ^
GLOBAL MENU: Press first letter of desired command. [Upper Case Only]
-----

```

The first technique to master before actually using *v.digit* is moving through the menus. The first six global options in the Main Menu are themselves menu names. Typing the first letter of any of These six global options will replace the present menu with the menu just chose.

There are three Main Menu global options that are not menu names. The global option “*Quit*” will exit you from digit. saving all your work; the “*Quit*” option is only available from the Main Menu. The “*Help*” option displays on-line help to the user. The option “*!*” replots the terminal screen text without altering any work there ‘to erase garbage appearing on the screen and the “***” option redraws the graphics screen. The “*Help*”, “***” and the “*!*” options can be chosen from any menu.

Enter the letter *D* to move to the Digitize Menu, shown below:

v.digit - digitize menu screen.

```

-----
GRASS-DIGIT Version 4.00                               Digitize menu
-----
Mouse Digitizer                                         AMOUNT DIGITIZED
# Lines: 0
# Area edges: 280
# Sites: 0
-----
Total points: 4889
-----
Digitize options:                                       CURRENT DIGITIZER PARAMS.
<space> Digitize                                       MODE      TYPE
- Toggle MODE                                       >POINT<   line
t - Toggle TYPE                                       stream    >AREA EDGE<
l - Auto Label                                       site
q - Quit to main menu
AutoLabel: DISABLED
-----
Edit Label Customize Toolbox Window Help Zoom * ! ^
-----
GLOBAL MENU: Press first letter of desired command. [Upper Case Only]
-----

```

Notice the menu name in the field at the top of the screen. Ignore for the moment the center two fields and look at the bottom field. Notice the global options. These are slightly different than the ones available in the Main Menu. Typing the first letter of any of the available global options presented here will cause the Digitize Menu to be replaced with the menu option chosen.

Enter *E* to bring to the screen the Edit Menu:.

v.digit - edit menu screen.

```

-----
| GRASS-DIGIT Version 4.00                                     Edit Menu |
-----
| Edit options:                                              |
| r - Remove line                                           |
| i - Remove site                                           |
| s - Snap line to node                                     |
| b - Break a line                                          |
| m - Move a point                                          |
| M - Move a line or site                                  |
| t - Re-type a line (AREA/LINE)                           |
| d - Display nodes in threshold                           |
| R - Remove BLOCK of lines                                |
|                                                           |
| q - Quit to main menu                                    |
|                                                           |
-----
| Digitize Label Customize Toolbox Window Help Zoom * ! ^  |
-----
| GLOBAL MENU: Press first letter of desired command. [Upper Case Only] |
-----

```

Once again, notice the global options at the bottom of the screen.

Now let's try returning to the Main Menu by typing M. Notice that nothing happens, because the global option "Main Menu" does not exist above. To return to the Main menu, simply type *q*.

Practice moving through the various menus. When you are confident with the menu structure, continue with the next section.

4. Using the Mouse:

Many *v.digit* options make use of a pointing device (mouse). A mouse usually has three buttons: a left, center, and right. (Some mouses have only two buttons; in this case, use of the middle button is simulated by pressing the left then right mouse buttons in quick succession.) Once the mouse is activated, movement of the mouse corresponds with movement of a pointer on the monitor screen. (Although this pointer may appear as the edge of a movable box, two perpendicular lines, an arrow, or some other type of pointer, we will here refer to this simply as the pointer.) The mouse is used to identify or define the extent of something shown on the monitor.

1. When in *v.digit* you can identify an object using the mouse by moving the pointer onto the object to be identified, then pressing the left mouse button. The object identified will be high-

lighted on the monitor in yellow. (This highlight color may be changed with an option in the *v.digit - customize menu*). If the object identified is not the desired one, move the pointer to the feature desired and choose it by pressing the left button. You can identify an unlimited number of objects. The last one chosen will be the only object identified.

2. Once you have identified the object you want, accept your choice by pressing the right mouse button. To end or abort use of the mouse at any time, press the middle button.

5. Digitizing:

Digitizing is the process of converting analog data into digital data by tracing the desired objects with the sight of a digitizer. (Here, sight is used to refer to that part of the digitizer by which the user traces features (sometimes referred to elsewhere as pointers, cursors, arrows, cross-hairs, etc.) In general, it consists of two perpendicular lines, several circles, or a combination of other objects, imposed on a transparent surface.) The digitizer collects points that are traced by the user through the sight in one of two modes: “point” mode, or “stream” mode. In point mode the digitizer will only collect points when the user tells it to, no matter where the digitizer’s sight is moved. In stream mode, the digitizer continually collects points as its sight is moved. Point mode is useful for digitizing straight lines and features requiring a small number of defined points (i.e., straight roads, parts of boundaries, transect lines, etc.), where the user does not wish to pick up extraneous points. Stream mode works well when digitizing curves, complex lines, soil polygons, streams and features where frequent collection of points is needed.

All features to be digitized are grouped into three types: points, lines and area edges. Area edges contain something within them. For example, the map boundaries circumscribing a roads map would be digitized as area edges (because they contain the roads within them), although the roads themselves would generally be digitized as lines. Such features as soils polygons and area geology would be digitized as area edges, while such features as roads and railroads would generally be digitized as lines. Sites indicate point locations such as water wells.

The Digitizing Menu is shown below:.

v.digit - digitize menu screen.

```

-----
| GRASS-DIGIT Version 4.00                                     Digitize menu |
-----
| Mouse Digitizer                                           | AMOUNT DIGITIZED |
|                                                           | # Lines:         0 |
|                                                           | # Area edges:    280 |
|                                                           | # Sites:         0 |
|                                                           | -----         |
|                                                           | Total points:    4889 |
|                                                           | -----         |
| Digitize options:                                         | CURRENT DIGITIZER PARAMS. |
|                                                           | MODE      TYPE |
| <space> Digitize                                         | >POINT<    line |
|   - Toggle MODE                                         | stream    >AREA EDGE< |
| t - Toggle TYPE                                         |             site |
| l - Auto Label                                          | AutoLabel: DISABLED |
| q - Quit to main menu                                   | -----         |
| Edit Label Customize Toolbox Window Help Zoom * ! ^ |
|-----
| GLOBAL MENU: Press first letter of desired command. [Upper Case Only]
|-----

```

At this point you should be familiar with the title (top) field and the global (bottom) field (refer to section 3 of this tutorial). Now look at the two center fields.

The first field is entitled “*KURTA digitizer*”. This field will differ depending on the type of digitizer in use.

The menu’s right side has a field entitled “*AMOUNT DIGITIZED*”. The numbers here reflect the number of lines and areas digitized, and the amount of points digit required to represent these objects. These numbers continually change to reflect changes in what has been digitized.

The field titled “*CURRENT DIGITIZER PARAMS*” displays the status of the digitizing parameters. Digitizing can occur either in point or stream mode. Also, objects may be lines (i.e., roads), sites (i.e. water wells), or area edges (i.e., soil polygons). This field lists which of these parameters are currently set.

The field titled “*DIGITIZE OPTIONS*” contains several options.

[space] Activates the digitizer, and begins reading and recording points.

m Toggles between the two modes of picking points: point/STREAM.

- t* Toggles between the two types of objects capable of being digitized: *Line/AREA EDGE*. (Active settings are shown in *UPPERCASE* letters.)
- q* Returns the user to the main menu, saving any work done in this menu.

6. Editing:

This is the Edit menu:

v.digit - edit menu screen.

```

-----
GRASS-DIGIT Version 4.00                               Edit Menu
-----
Edit options:
r - Remove line
i - Remove site
s - Snap line to node
b - Break a line
m - Move a point
M - Move a line or site
t - Re-type a line (AREA/LINE)
d - Display nodes in threshold
R - Remove BLOCK of lines

q - Quit to main menu

-----
Digitize Label Customize Toolbox Window Help Zoom * ! ^
-----
GLOBAL MENU: Press first letter of desired command. [Upper Case Only]
-----

```

DRAFT

Edit Options:

- r* Removes a previously drawn line. Position the pointer on the line to be removed. Choose a line by pressing the left mouse button. If the line highlighted is the desired line, confirm it by pressing the right mouse button. The line will then be removed.
- s* A node that you wished to be connected (i.e., snapped) to another node may not be. you may manually connect one node to another using this option. Move the pointer to the line you want snapped and choose it by pressing the left mouse button. if the node chosen is the correct node, accept your choice by pressing the right button. Now, choose and accept the node you want the line snapped to. The line will be attached to this selected node.

- b* Breaks a line inserting a node. A line is defined by its two endpoints. Choosing this option breaks a line into two separate lines by inserting a node at the break point. Use the selected pointer to choose a place on a line where the break is to occur. (Select pointer in customize menu.)
- m* Moves a point. Choose a line and the point on it that you want moved, and then choose the position where you want the point moved.
- p* Create a temporary node. The node can be used as a location reference or a point to snap another line to. Use either the mouse or digitizer, as selected from within the Customize menu. This node will exist until the end of the session or until you choose the write out session function.
- d* Displays nodes within a given threshold level of distance. When two nodes are within (closer together than) this defined snapping threshold, digit will snap the two nodes together and thereafter consider them to be the same node. this threshold distance is adjustable by the user. The *d* option will display the nodes that fall within a user-specified threshold.
- t* Changes the type of the line. If a segment was digitized as a line, but was meant to be digitized as an area-edge (or vice versa), choosing this option will toggle its type.
- R* Removes a block of objects. Sometimes it is desirable to remove all objects that lie within a defined area (e.g., to make a boundary map by removing all lines within the boundaries from a copy of another existing map). Using the mouse pointer the user outlines the area from which lines are to be removed.
- q* Returns the user to the Main Menu.

7. Labeling:

Labeling is the process of assigning category values to digitized objects. The user chooses a category value, moves the pointer onto the object to be labeled, chooses the object to be labeled, and accepts the chosen object. You may label an object more than once, but only the last label placed will be kept. To abort the labeling process, press the center mouse button.

The labels can only be numerical values. Before beginning to label, identify all the different values to be used. The value “0” is reserved for the category “no data”. You may not label anything with the value 0. Label the objects sequentially beginning with category value 1 and continuing as needed.

Labeling options are presented in the menu below:.

v.digit - label menu screen.

```

-----
GRASS-DIGIT Version  4.00                                     Label Menu
-----
Label options:
a - Label Areas                m - Label Multiple Lines
l - Label Lines                M - Un-Label Multiple Lines
s - Label Sites                c - Label Contours
A - Un-Label Areas            i - Contour interval:   <  5>
L - Un-Label Lines
S - Un-Label Sites

B - Bulk Label Remaining Lines

h - Highlight Lines of category #
d - Display Areas of category #

q - Return to main menu
-----
Digitize Edit Customize Toolbox Window Help Zoom * ! ^
-----
GLOBAL MENU: Press first letter of desired command. [Upper Case Only]
-----

```

Label options:

- a* Activates the mouse for area labeling. Enter a category value at the prompt appearing on the terminal screen. Position the pointer anywhere inside the area you want labeled. Choose and accept the area. Next, position the pointer on any line bounding the area to be labeled. Choose and accept this object.
- l* Activates the mouse for line labeling. Enter a category value at the prompt appearing on the terminal screen. Position the pointer on the line to be labeled. Press the left mouse button to choose the line and the right mouse button to accept and label the chosen line.
- A* Activates the mouse for unlabeling areas. Place the pointer anywhere within the boundary of the area to be removed. Choose this area by pressing the left mouse button, and accept the chosen area by pressing the right mouse button. The area will be removed.

- L* Activates the mouse for unlabeled lines. Position the pointer on the line to be removed. Choose the line by pressing the left mouse button, and accept the chosen line by pressing the right mouse button. The line will be removed.
- B* Bulk label ALL remaining unlabeled lines. You will be prompted for attribute number. Use 0 to abort. note this is only available for labeling lines.
- s* Displays the lines of a given category value. Enter the category value at the terminal prompt. All lines containing the category value will be highlighted.
- d* Displays the areas of a given category value. Enter the category value at the terminal prompt. All areas containing the category value will be highlighted.
- q* Returns the user to the main menu.

DRAFT

8. Windowing:

v.digit - window menu screen.

```

|-----|
| GRASS-DIGIT Version 4.00                               Window Menu |
|-----|
| Window options:                                        |
| a - Show area markers                                W - Define new window |
| A - Show area labels                                  C - Clear window   |
|                                                       |
| i - Show lines                                        c - Display scale  |
| l - Show labeled lines                               w - Where am I    |
| L - Show line labels                                - Display Overlay Map |
|                                                       - Display Backdrop CELL Map |
| s - Show sites                                       |
| S - Show site labels                                |
| n - Show nodes                                       |
|                                                       |
| q - Return from whence we came                       |
|-----|
| Help Zoom * ! ^                                       |
|-----|
| GLOBAL MENU: Press first letter of desired command. [Upper Case Only] |
|-----|

```

When digitizing, you need to be aware of three regions: the current, digitizing, and working regions. Once GRASS is initiated, the user is given a current region. It is within this region that all GRASS commands operate, unless otherwise noted. The current region is

initially set to the location default window, but can be changed by the user numerous times. The current region is remembered from one GRASS session to another. The current GRASS region has little impact on digitizing, until you create a cell file from the digitized map.

The digitizing region is defined in the *v.digit* start-up page. This region defines an area that must be large enough to encompass all of the data to be created within a particular digitizing session. This region is often set to the same dimensions as the default region for a particular location. However, this need not be the case. The setting of the digitizing region has no effect on the current region. The digitizing region only affects the digitizing session.

The working region is also only available inside of *v.digit*. This window has no permanence, i.e., it is not saved from one session to the next. This is the window that is changed to facilitate the zoom option in digit.

Each time a new region is chosen, the monitor screen replots itself. Parameters may be set that control what gets displayed each time the monitor replots. These may be set through the *Display options menu* available as a menu item in the *Customize menu*. The following options allow you to display specific features at any time.

- W* Starts an interactive procedure to define a new window. The options are to zoom in or to zoom out. Use either the mouse or the digitizer sight to define a window.
- a* Displays area markers. These are the locations in which area labels have been placed.
- A* Displays all the area labels.
- i* Displays all the lines.
- l* Displays all the labeled lines.
- L* Displays all the line labels.
- n* Displays all the nodes (line endpoints).
- s* Displays a scale in the upper left corner of the window.
- C* Erases the monitor screen. This erase does not remove any information from the digitized field; it only erases the contents

DRAFT

of the working region displayed.

- q* Returns the user to the location occupied just prior to this one.

9. Using the Customize Menu:

The *Customize menu* contains options with which to customize a digitizing session. Options set within a digitizing session do not carry over into a different session. Default values are reset each time *v.digit* starts up.

v.digit - customize menu screen.

```

-----
| GRASS-DIGIT Version 4.00                                     Window Menu |
-----
| Window options:                                           |
| a - Show area markers                                     W - Define new window |
| A - Show area labels                                     C - Clear window     |
|                                                         |
| i - Show lines                                           c - Display scale     |
| l - Show labeled lines                                    w - Where am I       |
| L - Show line labels                                     - Display Overlay Map |
|                                                         |
| s - Show sites                                           - Display Backdrop CELL Map |
| S - Show site labels                                     |
| n - Show nodes                                           |
|                                                         |
| q - Return from whence we came                           |
|                                                         |
|-----|
| Help Zoom * ! ^                                         |
|-----|
| GLOBAL MENU: Press first letter of desired command. [Upper Case Only] |
|-----|

```

Customize Options:

- d* Sets digitizing threshold.
- s* Sets snapping threshold.
- b* Toggles beeps emitted from the terminal between on off [default on].
- t* Toggles terse mode between on off [default off]. When terse mode is on, unnecessary text does not appear on the terminal screen.
- a* Toggles auto window on off [default off]. When labeling an area often an area being labeled is not completely within the working

- window. When auto window is on, the working window automatically adjusts itself to encompass the area being labeled.
- w* Toggles between mouse/digitizer [default digitizer]. Windowing may be done using either a mouse or a digitizer.
 - p* Toggles between mouse/digitizer [default mouse]. In edit menu, the point marker function and the break line function can be performed with the mouse or digitizer. This option allows you to choose which device to use.
 - O* Overlays a vector file on top of what is being digitized. This option enables the user to overlay an already existing vector file onto the features currently being digitized for visual comparison. References elsewhere affect the way this map is displayed.
 - D* Moves the user into the Display Menu (only accessible through this menu).
 - C* Moves the user into the Color Menu (only accessible through this menu).
 - q* Returns the user to the menu accessed prior to accessing the Customize Menu.

Every time the graphics monitor refreshes itself (displays a new image whether through a change in region or through the beginning of a new *v.digit* session) the monitor only displays what it is told to. For example, it may display lines, area labels, nodes and the points that make up lines, or it may only display area labels. The user can specify which features will appear each time the screen is redrawn. The values shown below are the default settings.

DRAFT

v.digit - customize/display menu screen.

```

-----
| GRASS-DIGIT Version  4.00                               Display Menu |
-----
| Display options:           Current:                     |
| a - Area Labels           OFF                           |
| l - Line Labels           OFF                           |
| s - Site Labels           OFF                           |
| m - Area Markers          ON                            |
| A - Area Border lines     OFF                           |
| L - Labeled lines         ON                            |
| i - Lines                 ON                            |
| S - Sites                 ON                            |
| n - Nodes                 ON                            |
| p - Points in lines       OFF                           |
|
| r - Reset Defaults
|
| q - Return from whence we came
-----
| Help Zoom * ! ^
-----
| GLOBAL MENU: Press first letter of desired command. [Upper Case Only]
-----

```

- a* Displays the category values associated with area. This option should be turned on when labeling areas.
- l* Displays the integer attributes associated with lines. This option should be turned on when labeling lines.
- m* Displays the area markers of a labeled area. When an area is labeled, the location where the area was placed is marked with a point. (This point is only visual and does not affect the contents of the *v.digit* file.) It is often helpful to turn this option “*on*” when not labeling, instead of option “*a*” above. This prevents cluttering of the screen.
- A* Displays, in a unique color, all the line that make up labeled areas.
- L* Display, in a unique color, all the labeled lines.
- i* Displays digitized area lines in blue, and digitized area edges in white.
- n* Displays nodes as dots.
- p* Displays all the points (not just nodes) that describe a line.
- O* Displays a second vector map to aid in referencing the current vector file with it.

DRAFT

- r* Resets default values. This option resets all options to their default values.
- q* Returns the user to the menu accessed prior to accessing the Display Menu and redraws the working window if any parameters have been changed.

Features in *v.digit* are color-coded for ease of identification. Every time *v.digit* is started, colors representing different objects are reset to their default colors.

v.digit - customize/color menu screen.

```

-----
GRASS-DIGIT Version 4.00                                     Color Menu
-----
Color options:                                             Current:
a - Areas                                                grey
l - Lines                                                blue
s - Sites                                                green
A - Labeled areas                                       orange
L - Labeled lines                                       magenta
S - Labeled Sites                                       aqua
1 - Nodes w/ 1 line                                     green
2 - Nodes w/ 2 or more lines                           red
h - Highlight                                           yellow
B - Background                                          black
O - Overlay map                                         white
r - Reset Defaults

q - Return from whence we came

Help Zoom * ! ^

GLOBAL MENU: Press first letter of desired command. [Upper Case Only]
-----

```

DRAFT

The values above are the default settings. The average user should have little need to change these.

- a* Sets area color.
- l* Sets line color.
- s* Sets site color. (At this time, sites are not supported in digit.)
- A* Sets labeled area color.
- L* Sets labeled line color.
- 0* Sets the color of nodes that have no lines attached (see edit option 'p').
- 1* Sets the color of unsnapped nodes (nodes with only 1 line

attached).

2 Sets the color of snapped nodes (nodes with 2 or more lines attached).

h Sets the highlight color. Many options in *digit* make use of highlighting. Any object that is chosen is highlighted. The default highlight color is yellow.

B Sets background color.

O Sets overlay map color.

r Resets all the colors to their default values.

q Returns the user to the Customize Menu.

10. Using the Toolbox Menu:

The Toolbox Menu has additional options that are useful while in *v.digit*. Of special note, it has several commands to help diagnose digitizing problems.

v.digit - toolbox menu screen.

```

-----
| GRASS-DIGIT Version 4.00                                     Toolbox Menu |
-----
| Toolbox options:                                           |
|                                                            |
| w - Write out session                                       |
|   - Register map                                           |
|   - Build Neat line                                         |
|                                                            |
| u - Display Unlabeled Areas                                 |
| o - Display Open area lines                                 |
| d - Display Duplicate lines                                 |
|                                                            |
| n - Display Node lines                                       |
| i - Display Islands                                         |
|                                                            |
| q - return from whence we came                             |
|                                                            |
-----
| Window Help Zoom * ! ^                                     |
-----
| GLOBAL MENU: Press first letter of desired command. [Upper Case Only] |
-----

```

Toolbox Options:

w Writes the present session “to disk” and returns control to

- v.digit*. If your digitizing session lasts for a long period of time, it is good practice to choose this option occasionally. Writing your session to disk involves taking your active session, and writing it to disk.
- R* Allows you to re-register a map to the digitizer surface. This is very useful if, for example, you have more than one quad sheet that will make up a single *v.digit* file.
 - u* Display Unlabeled Areas (see below: Diagnosing Problems).
 - o* Display Open-area Lines (see below: Diagnosing Problems).
 - n* Display each line attached to a node (see below: Diagnosing Problems).
 - q* Returns control to the menu accessed prior to accessing the Toolbox Menu.

11. Diagnosing Problems:

There are a number of tools provided to help find and solve problems that can occur in digitizing. The tools are listed below along with a description of the situations in which they are useful.

- Display Unlabeled Areas [*Toolbox Menu*]. This function is NOT guaranteed to work at all times. It will work best immediately after running the GRASS programs *v.support* or *v.import*. This is because areas are not necessarily known to *v.digit* unless they are labeled or have just been built by *v.support*. This function will attempt to highlight all areas of which it is aware that are currently unlabeled. This will only show those unlabeled areas which are inside of the working window. Not that in a large, cluttered map these areas may be very small and not noticeable. Try using the [*Window Clear*] command before using this function.
- Display Open-area Lines [*Toolbox Menu*]. This function is NOT guaranteed to work at all times OR in all cases. This function can help to locate problem areas. It is recommended that this be used only after labeling all areas OR running the GRASS program *v.support*. If there were any areas built incorrectly or not at all, this function will attempt to locate and highlight the lines of these areas. It works on the assumption that if a line is defined as an area edge, then it should bound an area. All area

DRAFT

edge lines that do not bound any known areas will be highlighted. *v.support* may give warning messages about not being able to match up labels. This function can help find the areas in question in these cases. See the previous entry for more information.

- **Display Node Lines [Toolbox Menu].** Highlights the lines emitting from a chosen node. Select and accept a node and you can then view each line attached to it individually. Press the right mouse button to view lines counter-clockwise around the node, and the left button to view them clockwise. This utility is useful for two common situations. The most common reasons for an area to be built incorrectly or not be built at all are overshoots, and digitizing of the same line twice. Because areas are built in digit by following the angles with which lines go into and come out of nodes, overshoots could cause bad angle information to be stored. This means that users can catch overshoots by noting whether the lines extending from a given node fail to be highlighted in the proper order around the node. Double-digitized lines can be caught by noting whether what was thought to be one line is highlighted twice by *v.digit*. (Note that islands are always highlighted twice; this does not mean that island areas have been digitized twice.)
- There is one more debugging tool that is available in an incomplete and totally unsupported form. It is the Debug Menu, available to the user from any menu by pressing the [-] (minus) key. This menu, designed for debugging *v.digit* during its development, is not part of the supported *v.digit* program and is not guaranteed to be complete. It is useful for finding information on objects by their internal reference numbers. The programs *v.support* and *v.import* may occasionally provide a warning message about finding more than one label for a given line or area. (For this example, we will speak of areas; however, it also applies to line labels.) This can be caused by two events. Either there are in fact duplicate labels in the file for one reason or another, or a label from a nearby area was interpreted to belong to the wrong area. In this case, you would desire to find the problem areas and correct them by hand. These warning messages supply the problem area's internal reference number. To find these areas, you could use the debug menu option "A" to display areas by number. This option will highlight each area individually, and in sequence. To view a specific area, type its number and press [return]. To exit that function, type "q" and press [return].

Chapter 5 GIS Applications

Objectives

In this chapter we will use GRASS to locate imaginary sites which might be used for the location of a landfill in the Spearfish dataset. In doing so we will include data from various layers which will include: roads, streams, geology, slope, soils, and landuse. While the criteria used for the selection of sites in the example are by no means justified, they serve to show how a GIS can be used to suggest answers to similar questions. Real world applications would involve more detailed and defensible criteria.

At the end of this chapter you should be able to:

1. Define the following terms:
 - binary map
2. Use the following commands in GRASS:
 - r.reclass
 - r.report
 - r.mapcalc
 - r.clump
 - d.what.rast
 - r.buffer
3. Explain the difference between a binary data layer and a regular data layer.
4. Structure a problem so that it may be solved as easily and quickly as possible.

DRAFT

Development of Criteria

Before we can begin working exclusively with the computer we must first define the criteria that we will use in finding the imaginary landfill sites. Each data layer will have a different use in the problem so each will have to be defined independently. Here are some arbitrary criteria for each of the layers and a short explanation of why they might be used:

Data Layer	Criteria †	Reasoning
roads	within 500 meters	The landfill must have good access.
streams	beyond 500 meters	To avoid direct runoff into streams.
landuse	away from currently developed areas.	To avoid locating too close to residential areas.
geology	impermeables	Locate on a site which will not be susceptible to penetration by chemicals, etc.
slope	low slopes	To avoid excess erosion near the site.
soils	clays (and loams)	To avoid sandy soils.

† *These criteria are general and should be carefully considered in a real world application.*

By using these criteria we hope to find only a few suitable areas which can be used to locate the site. From this we can further reduce the number of areas by looking at their size and characteristics more closely. In this example we will select sites which are greater than ten acres to further reduce the number of suggested sites.

Before we begin

If you haven't already started GRASS do so now. Refer to Chapter 3 for details. Next, execute the following commands to be sure that you have your window set correctly and the graphics display in fixed mode.

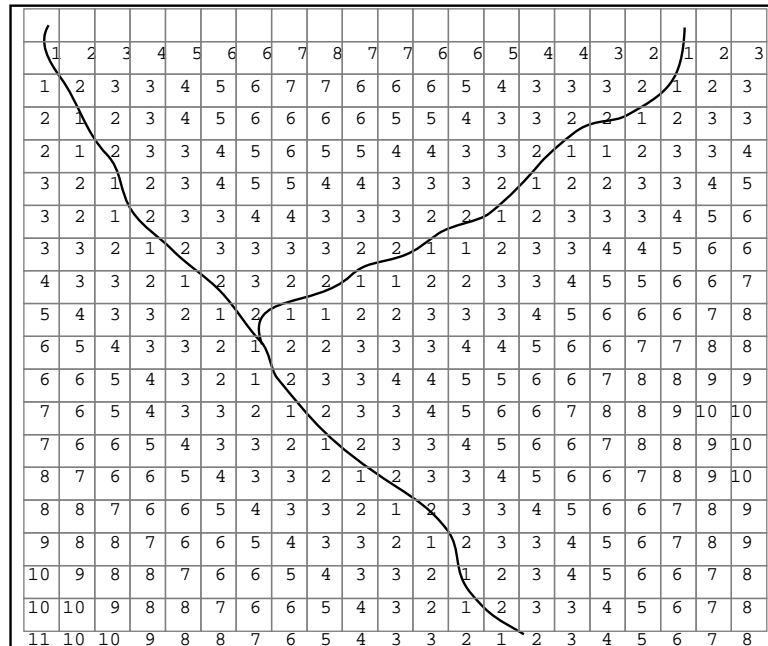
- d.colormode fixed
- g.region -d
- d.frame -e
- g.region -p

Now, write down the values of the rows and columns which will be 140 and 190 respectively from the text terminal. We will use these later.

Distance calculations

To satisfy the first three criteria, within 500 meters of roads, beyond 500 meters of streams, and beyond 500 meters of residential areas, we will have to generate distance data layers. Distance maps are created by representing categories as proximity zones about corresponding categories on an existing data layer. In the following example, distances were calculated in one cell increments from streams. Notice that the streams, the category from which the distances were calculated, are represented as 0 or *no data*. The value of the selected category will always be 0 in a distance map.

Distance map generated from streams. Notice that the value for the streams is 0.

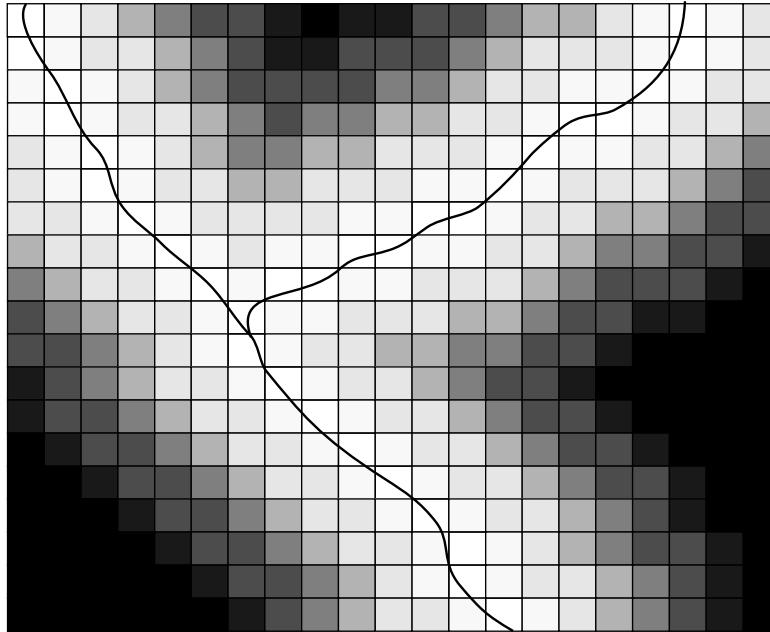


GRASS represents distance maps in shades of gray similar to the map shown below. Categories can be grouped in the distance

DRAFT

function to generalize the map. Below, cells which are 7 to 10 cells from the streams are all represented in black as the seventh category.

Distance map calculated from streams. The areas which are more than seven cells from the streams are grouped into one category.



To generate the maps, first run the *r.buffer* module.

```
r.buffer
```

Specify *roads* as the data layer which contains the categories and *roads.buf* as the new file to contain the distance-from results.

```
roads  
roads.buf
```

The next screen which appears is used to select the distances by which the categories will be defined in the new file. Notice that the distances are measured in meters. You may change this to any of standard unit of measure (i.e. mile, feet, etc).

Since we are interested only in areas which fall within 500 meters of roads, we can group the data into three categories, those areas of no data, those within the 500 meter boundary and those outside the boundary. See the following figure for category values.

ENTER MAXIMUM DISTANCE FOR EACH ZONE DESIRED		
All values in: meters		
ZONE 1:	500	_____
ZONE 2:	33000	_____
ZONE 3:	0	_____
ZONE 4:	0	_____
ZONE 5:	0	_____
ZONE 6:	0	_____
ZONE 7:	0	_____
ZONE 8:	0	_____
ZONE 9:	0	_____
ZONE 10:	0	_____
ZONE 11:	0	_____
ZONE 12:	0	_____
ZONE 13:	0	_____
ZONE 14:	0	_____
ZONE 15:	0	_____
ZONE 16:	0	_____
ZONE 17:	0	_____
ZONE 18:	0	_____
ZONE 19:	0	_____
ZONE 20:	0	_____
ZONE 21:	0	_____
ZONE 22:	0	_____
ZONE 23:	0	_____
ZONE 24:	0	_____
ZONE 25:	0	_____
ZONE 26:	0	_____
ZONE 27:	0	_____
ZONE 28:	0	_____
ZONE 29:	0	_____
ZONE 30:	0	_____
ZONE 31:	0	_____
ZONE 32:	0	_____
ZONE 33:	0	_____
ZONE 34:	0	_____
ZONE 35:	0	_____
ZONE 36:	0	_____
ZONE 37:	0	_____
ZONE 38:	0	_____
ZONE 39:	0	_____
ZONE 40:	0	_____
ZONE 41:	0	_____
ZONE 42:	0	_____
ZONE 43:	0	_____
ZONE 44:	0	_____
ZONE 45:	0	_____
ZONE 46:	0	_____
ZONE 47:	0	_____
ZONE 48:	0	_____
ZONE 49:	0	_____
ZONE 50:	0	_____
ZONE 51:	0	_____
ZONE 52:	0	_____
ZONE 53:	0	_____
ZONE 54:	0	_____
ZONE 55:	0	_____
ZONE 56:	0	_____
ZONE 57:	0	_____
ZONE 58:	0	_____
ZONE 59:	0	_____
ZONE 60:	0	_____

AFTER COMPLETING ALL ANSWERS, HIT <ESC> TO CONTINUE
(OR <Ctrl-C> TO CANCEL)

Notice that the second zone's limit is set very high. Remember those numbers you wrote down? The number used for zone 2 is the result of adding the rows and columns and multiplying by the resolution. It insures that all cells beyond 500 meters will be included in the second zone. A simple way to determine a suitable number to enter for such a value would be to determine a value which is larger than the diagonal distance across the map. In this case add the rows (140) to the columns (190) which equals 330, then multiply 330 by the resolution (100) which is equal to 33,000. Remember, you can

$$(140 + 190) 100 = 33000$$

view the current window settings by typing *g.region -p*. When you have finished with this screen use *Esc* to continue.

[Esc]

We will now have to wait while *r.buffer* calculates the distance map. When it finishes, check the map to see if it is what you expected by using *d.rast roads.buf*.

d.rast roads.buf

Streams:

The next distance map to be calculated should be the distance

from streams. The areas we will be interested in are those beyond 500 meters of the streams. First, let's determine which of the categories in the streams map we will use in calculating distances. Generate a report by typing: *r.report streams*. The report will look similar to the one below.

r.report streams

Report on streams using the r.report command. Categories 1 - 5 are used in the distance calculations.

```

+-----+
| Layer:  [streams] in mapset [PERMANENT] Date: Mon Jun 3 09:55:34 |
| Location: spearfish |
| Title:  Hydrography |
| Mask:   none |
+-----+
| Window: north: 4928000.00 east: 609000.00 |
|         south: 4914000.00 west: 590000.00 |
|         res:   100.00 res:   100.00 |
+-----+
+-----+
| cat#      Name |
+-----+
| 0 | no data |
| 1 | perennial stream |
| 2 | intermittent stream |
| 3 | aqueduct |
| 4 | shoreline |
+-----+

```

Because we will calculate distances from all of the categories (except 0) we will not need to reclass the map before we run *r.buffer*. However, with landuse we will have to first prepare an intermediate map before running the command.

This time to calculate the map, use a less interactive version of *r.buffer*, by specifying input and output maps. Start by typing *r.buffer input=streams output=streams.buf distances=500,33000 units=meters*.

**r.buffer input=streams output=streams.buf
distances=500,33000 units=meters**

GRASS starts the buffer program running. This should only take a few seconds to finish. You might look at the new data layer using *d.rast streams.dist*.

d.rast streams.dist

Landuse:

The next distance map that we will calculate will be the buffer

DRAFT

zone around land already in use. First, generate a report of the data layer so that we can choose which categories to calculate distances from. Use *r.report landuse*. The results should appear as follows.

r.report landuse

```

+-----+
|                                RASTER MAP CATEGORY REPORT                                |
|LOCATION: spearfish                                                    Fri Oct 18 21:29:53 1991|
+-----+
|              north: 4928000    east: 609000                          |
|REGION        south: 4914000    west: 590000                          |
|              res:      100     res:      100                          |
+-----+
|MASK:none                                                            |
+-----+
|MAP: Land Use (landuse in PERMANENT)                                |
+-----+
|                                Category Information                                |
| #|description                                                         |
+-----+
|0|no data                                                             |
|1|residential                                                         |
|2|commercial and services                                           |
|3|industrial                                                         |
|4|other urban                                                         |
|5|reservoirs                                                         |
|6|bare exposed rock                                                 |
|7|quarries, strip mines and gravel pits                             |
|8|transportation and utilities                                       |
+-----+

```

For solving the problem, we will involve 6 of the 8 categories above, excluding bare and exposed rock as well as quarries, strip mines and gravel pits as these seem to be acceptable locations to locate.

This means that we will have to calculate our distances from categories 1 - 5 and 8, giving each a buffer of 500 meters. To do this means that we will have to first reclass the landuse map so that categories 6 and 7 are not included in the buffer calculations. Start with the *r.reclass* command.

r.reclass

DRAFT

```
RECLASS: Program for reassigning category codes

Enter name of data layer to be reclassified
Enter 'list' for a list of existing raster files
Enter 'list -f' for a list with titles
Hit RETURN to cancel request
> landuse
<landuse>

Enter name of NEW RECLASSIFIED map
Enter 'list' for a list of existing raster files
Enter 'list -f' for a list with titles
Hit RETURN to cancel request
> landuse.tmp
```

Specify landuse as the original map layer and landuse.tmp as the new/reclassified map.

landuse
landuse.tmp

```
Please indicate how you would like the reclass table initialized

      0_

0  All values set to zero
1  All values set to the same category number
```

Set all of the categories equal to 0 and continue on to the next screen.

[Esc]

D
R
A
F
T

```

ENTER NEW CATEGORY NUMBERS FOR THESE CATEGORIES

OLD CATEGORY NAME                                OLD   NEW
NUM     NUM

no data . . . . .                                0     0__
residential . . . . .                            1     1__
commercial and services . . . . .                2     1__
industrial . . . . .                             3     1__
other urban . . . . .                           4     1__
reservoirs . . . . .                            5     1__
bare exposed rock . . . . .                     6     0__
quarries, strip mines and gravel pits . . . . . 7     0__
transportation and utilities . . . . .           8     1__

Next category: end__ (of 8)

AFTER COMPLETING ALL ANSWERS, HIT <ESC> TO CONTINUE
(OPTIONAL <Ctrl-C> TO CANCEL)
    
```

Now, place a 1 beside those categories from which we will calculate buffers from. Continue on with the reclass by hitting [Esc]. Since this is only a temporary file, you don't need to bother with the Title and category label page. Simply hit [Esc] again to start the reclass.

[Esc]
[Esc]

We are now ready to generate the buffer map. Use *r.buffer input=landuse output=landuse.dist distances=500,33000 units=meters* to start the process.

```
r.buffer input=landuse.tmp output=landuse.dist
                        distances=500,33000 units=meters
```

GRASS calculates the map and returns a prompt to you. *d.rast* the new data layer for a visual check.

```
Dcell landuse.dist
```

Generation of Binary Data Layers:

The next step in locating the landfill site is to generate binary maps for the three buffer layers that we have just created (roads.buf, streams.buf, landuse.buf) as well as for the geology, soils, and slope data layers. Binary maps are those which are made up of only 1's and

DRAFT

0's. In our case the 1's will be acceptable areas for locating a landfill and 0's will be unacceptable.

To accomplish this we will use a series of reclassifications to group acceptable categories into the category value 1 as well as group unacceptable categories into the category value 0.

Begin with the roads buffer map. Generate a report of the layer using *r.report roads.buf*. You should see the following.

r.report roads.buf

```

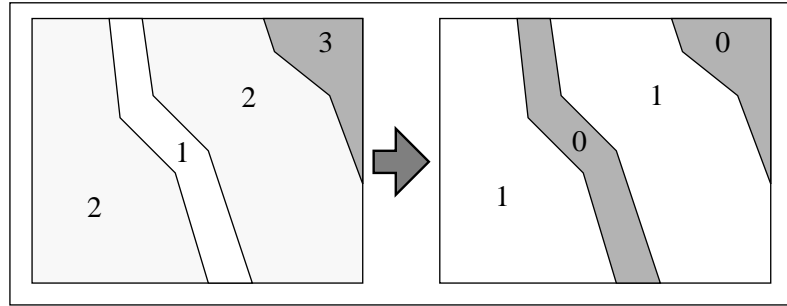
+-----+
|                                     RASTER MAP CATEGORY REPORT                                     |
| LOCATION: spearfish                                     Tue Nov  5 13:59:59 1991 |
+-----+
|           north: 4928000   east: 609000 |
| REGION   south: 4914000   west: 590000 |
|           res:      100   res:      100 |
+-----+
| MASK:none |
+-----+
| MAP: Distance Zones (roads.buf in grassx) |
+-----+
|                                     Category Information                                     |
| #|description |
+-----+
| 1|distances calculated from these locations |
| 2|500 meters |
| 3|33000 meters |
+-----+

```

By looking at the report, we see that there are no areas of no data and that there are three categories found in the map. Returning to our criteria we can develop a reclassification scheme for the data. First, the site really can't be located "on" and existing road but rather adjoining one. There for category 1, "distances calculated from these locations" is an unacceptable category and should be made to be a 0. Secondly, the site must be within 500 meters of a road which is satisfied only by category 2. Category 2 therefore should be reclassified to category 1 or acceptable. Lastly, those areas beyond 500 meters are unacceptable and should be reclassified as category 0.

Consequently if we were to map our pending reclassification for roads.buf it would look like:

DRAFT



Now reclassify the buffer map to a binary data layer by using *r.reclass*.

r.reclass

```
RECLASS: Program for reassigning category codes

Enter name of data layer to be reclassified
Enter 'list' for a list of existing raster files
Enter 'list -f' for a list with titles
Hit RETURN to cancel request
> roads.buf
<roads.buf>

Enter name of NEW RECLASSIFIED map
Enter 'list' for a list of existing raster files
Enter 'list -f' for a list with titles
Hit RETURN to cancel request
> roads.landfill
```

After entering the name of the original file followed by the name of the new map GRASS will ask how you would like the data to appear in the reclass menu.

```
Please indicate how you would like the reclass table initialized

0_

0 All values set to zero
1 All values set to the same category number
```

Select 0 - All values set to zero. GRASS then provides you with a list of categories by which you can select new category numbers to reclass the map. Here we can apply the values that were discussed above. Enter a 1 next to category 2 (500m bufer) and leave the rest of the categories 0.

DRAFT

DRAFT

```

ENTER NEW CATEGORY NUMBERS FOR THESE CATEGORIES

OLD CATEGORY NAME                                OLD   NEW
NUM      NUM

. . . . . 0  0__
distances calculated from these locations . . . . . 1  0__
500 meters . . . . . 2  1__
33000 meters . . . . . 3  0__

Next category: end__ (of 3)

AFTER COMPLETING ALL ANSWERS, HIT <ESC> TO CONTINUE
(OR <Ctrl-C> TO CANCEL)

```

[Esc]

Hit [Esc] to continue to the last page. Here GRASS asks you to enter category labels to the new reclassified map. Enter "Acceptible Roads" for the new title, and since we have two categories, 0-unacceptable and 1-acceptable, we can enter this beside the correct category. See below.

```

[roads.landfill] ENTER NEW CATEGORY NAMES FOR THESE CATEGORIES

TITLE: Acceptible Roads_____
CAT   NEW CATEGORY NAME
NUM
0    unacceptable_____
1    acceptable_____

Next category: end__ (of 1)

AFTER COMPLETING ALL ANSWERS, HIT <ESC> TO CONTINUE
(OR <Ctrl-C> TO CANCEL)

```

GRASS reclasses the map. You can display the new map by using *d.rast roads.landfill*.

```
d.rast roads.landfill
```

Streams

Next, we need to reclassify the streams.buf file to become a binary map layer. This time, however, we will use a faster implementation of the r.reclass command.

The streams reclassification will be much like the roads, only for the streams, the acceptable areas are beyond the 500 meter buffer zone. Therefore, categories 0 through 2 will be set to 0 and category 3 will be set to 1. Use the following command to start the command line version of r.reclass.

```
r.reclass input=streams.buf output=streams.landfill
                                                title="Acceptable Streams"
> 0 thru 2 = 0 unacceptable
> 3 = 1 acceptable
> end
```

GRASS reclasses the map and returns the GRASS> prompt. Check that the map is correct by running the following two commands.

```
d.rast streams.landfill
d.what.rast
```

Check to see if the areas you expected to be acceptable or unacceptable are that.

Landuse

The Landuse map is virtually the same as the streams reclass. Acceptable area will be those beyond 500 meters of a landuse buffer while unacceptable areas are those within the landuse regions and their buffer. Use the following command to perform the reclass.

```
r.reclass input=landuse.tmp output=landuse.landfill
                                                title="Acceptable Landuse"
> 0 thru 2 = 0 unacceptable
> 3 = 1 acceptable
> end
```

Display the map and check its accuracy as you did for streams.

```
d.rast landuse.landfill
```

DRAFT

```
d.what.rast
```

Geology

To prepare the geology map for use in our example, we will reclass the original layer into a binary data layer. We will do this by grouping the hard, less permeable rocks as category one (acceptable) and all other areas as 0 (unacceptable). First, generate a report of the categories in the geology map layer.

```
r.report geology
```

```

+-----+
|                                     RASTER MAP CATEGORY REPORT                                     |
| LOCATION: spearfish                                     Thu Nov 14 16:05:19 1991 |
+-----+
|               north: 4928000   east: 609000           |
| REGION        south: 4914000   west: 590000           |
|               res:           100   res:           100   |
+-----+
| MASK:none                                             |
+-----+
| MAP: Geology (geology in PERMANENT)                   |
+-----+
|                                     Category Information                                     |
| #|description                                         |
+-----+
| 0|no data                                             |
| 1|metamorphic                                         |
| 2|transition                                         |
| 3|igneous                                             |
| 4|sandstone                                          |
| 5|limestone                                          |
| 6|shale                                              |
| 7|sandy shale                                       |
| 8|claysand                                           |
| 9|sand                                               |
+-----+

```

For the reclass, we will use categories 1-3 and 8 as acceptable areas and all others as unacceptable. Run *r.reclass input=geology output=geology.landfill* to start the less-interactive GRASS reclassification program.

```

r.reclass input=geology output=geology.landfill
                                     title="Acceptible Geology"
> 0 4 5 6 7 9 = 0 unacceptable
> 1 2 3 8 = 1 acceptable
> end

```

Display the map and query it to insure its accuracy.

```

d.rast geology.landfill
d.what.rast

```

Slope

We are now ready to do some reclassifications of the slope.7 data layer. A report of the data layer shows the following.

```
r.report slope
```

```

+-----+
|                                RASTER MAP CATEGORY REPORT                                |
|LOCATION: spearfish                                Thu Nov 14 16:20:40 1991|
+-----+
|                                north: 4928000    east: 609000                                |
|REGION  south: 4914000    west: 590000                                |
|                                res:      100     res:      100                                |
+-----+
|MASK:none|
+-----+
|MAP: slope reclassified into groups of percent rise (slope.7 in PERMANENT)|
+-----+
|                                Category Information                                |
| #|description|
+-----+
|0|no data|
|1|0-2%|
|2|3-5%|
|3|6-10%|
|4|11-15%|
|5|16-20%|
|6|21-25%|
|7|>25%|
+-----+

```

For our purposes we will accept slopes which are less than or equal to 10% or categories 0 - 3 and disregard those areas over 10% or 4 through 7. Run `r.reclass input=slope.7 output=slope.landfill title="Acceptable Slopes"` to begin reclassifying the slope map. Execute the following reclass.

```

r.reclass input=slope.7 output=slope.landfill title="Acceptable Slopes"
> 0 thru 3 = 1 acceptable
> 4 thru 7 = 0 unacceptable
> end

```

Soils

The soils will be the final map which will need to be reclassified. Since this is such an extensive reclass, use the highlighted categories from the report below for the reclass.

```
r.report soils
```

DRAFT

DRAFT

RASTER MAP CATEGORY REPORT		
LOCATION: spearfish		Thu Nov 14 16:29:58 1991
REGION	north: 4928000 south: 4914000 res: 100	east: 609000 west: 590000 res: 100
MASK:none		
MAP: Spearfish Soils (soils in PERMANENT)		
Category Information		
#	description	
0	No data	
1	Aab	
2	Ba	
3	Bb	
4	BcB	
5	BCc	
6	BeE	
7	Bhe	
8	Bkd	
9	CBE	
10	CaD	
11	CaE	
12	Cc	
13	GBE	
14	GaD	
15	GcD	
16	GdE	
17	GeD	
18	HBF	
19	Ha	
20	KaB	
21	LaE	
22	MaC	
23	MaD	
24	McD	
25	NaB	
26	NaC	
27	Nac	
28	Nbd	
29	Ncd	
30	Ndb	
31	Ndc	
32	PbE	
33	PcB	
34	PcD	
35	Pe	
36	RBF	
37	RCF	
38	RaE	
39	SaA	
40	SaB	
41	SbB	
42	Sd	
43	ShA	
44	Sk	
45	TaA	
46	TaB	
47	TaC	
48	VEF	
49	VCE	
50	VaA	
51	VaB	
52	VaC	
53	WaA	
54	Wb	

Use the following reclassification rules to perform the reclass.

```
r.reclass input=soils output=soils.landfill title="Acceptible Soils"
> 0 6 7 9 thru 17 28 29 36 thru 38 48 49 53 = 0 unacceptable
> 1 thru 5 8 18 thru 27 30 31 33 thru 35 39 thru 47 50 thru 52 54 = 1
                                     acceptable
> end
```

Check the map as you have done the others for correctness.

```
d.rast soils.landfill
d.what.rast
```

r.mapcalc

We now have six binary files, all of which have a .landfill extension. Use *g.list rast* to list the seven files. They should be:

- roads.landfill
- streams.landfill
- landuse.landfill
- geology.landfill
- slope.landfill
- soils.landfill

In each case, no data areas (0's) are the areas which are unacceptable for our purposes and all other areas (1's) are acceptable. If we were to add all seven maps together at this time, the only areas which would satisfy all of the criteria would be those areas with a value of 6. In other words, in cells where you find a category value of 6 you know that the corresponding cell in all seven binary maps had a value of 1.

To generate this new map we will use the GRASS *r.mapcalc* (map calculator) function. *r.mapcalc* is capable of mathematical, algebraic and boolean operations on map layers. You need only build an expression for it to use. Use *g.manual* to view the *r.mapcalc* command syntax.

```
g.manual
r.mapcalc
```

To add the six maps together enter the following command.

```
r.mapcalc six.landfill = roads.landfill + streams.landfill +
                       geology.landfill + slope.landfill + soils.landfill +
                       landuse.landfill + owner.landfill
```

DRAFT

`r.mapcalc` computes the map and returns the prompt to you. `d.rast` the new map to see how it looks. There should be six categories (colors) in the map. Also, use `d.what.rast` to poke around at the categories to see how many nearly qualify and which areas don't come near to qualifying.

```
d.rast six.landfill
d.what.rast
```

Now, so that we can see which areas are category six, reclass the `six.landfill` map to a binary data layer.

```
r.reclass input=six.landfill output=accept.landfill
> 0 thru 5 = 0
> 6 = 1
> end
```

`d.rast accept.landfill`. Notice that even though there are discrete areas of possible sites, they are all categorized a one group (category 1). To give each of these discrete areas its own category value we will use the `r.clump` command which "clumps" together contiguous areas within a map layer. Execute the following command to clump the areas.

```
r.clump input=accept.landfill final.landfill
```

`d.rast` the new file to view the categories. Also, generate a report for the new file.

```
d.rast final.landfill
r.report map=final.landfill units=a,mi,p
```

If this were the real thing, size considerations would enter in at this point. By examining the report, you could determine how to reclassify the map to meet size requirements. Maps and reports could then be generated and included in a formal report or some similar presentation.

This concludes the GIS Applications Example.

Chapter 6 Watershed Analysis

Objectives:

In this chapter we will duplicate a tutorial which can be found in the GRASS 3.0 manual tutorial section. By using Digital Elevation Models (DEM's) we GRASS can calculate drainage basins for a given outlet point as well as develop a stream network from the model. Such a process is extremely valuable in predictive modelling for things such as availability of water, erosion potential, and similar phenomena which are directly related to the size of a particular watershed.

At the end of this chapter you should be able to:

1. Define the following terms:
 - DEM
 -
2. Use the following GRASS commands:
 - watershed
 - thin
3. Use the SIX steps in the watershed program to develop a watershed.

Before we begin:

Before we start working on the data layers, execute the following commands to be sure that you have your window set correctly (resolution at 30m in this case) and the graphics display in fixed mode.

- Dcolormode fixed
- Gwindow n=4924000 s=4914010 e=600000 w=590220 res=30
- Dscreen

Orientation:

D
R
A
F
T

Land managers are given the job of protecting, maintaining, and developing areas of land. In order to perform this duty, information is often needed which is related to the delineation of a watershed and its stream network. Sometimes information related to stream and divide networks is available within a GIS system. If the channel and divide networks have to be manually interpreted from topographic maps or aerial photographs, with subsequent digitization to enter the information into a GIS system, the work could be quite tedious and time consuming. If this information could instead be derived from digital elevation data which is readily available for many locations, less work would be required, and the results might be more accurate. Accuracy is of course affected by the accuracy of the original elevation data, the resolution of the data used, and the effectiveness of the routines for extracting the information. The series of programs provided within the watershed menu is effective for the delineation and segmenting of channel networks, and for identifying a watershed boundary. Work is continuing to increase the effectiveness of delineating subbasin boundaries as well. The logic for these routines is based primarily on code from Larry Band, Department of Geography, University of Toronto. The original code was changed to include the use of GRASS routines for the accessing and storing of data, and some additions were made to provide additional output.

Getting Started:

The watershed command requires no arguments. The structure is interactive, and primarily menu-driven. Upon entry, the first input required from the user is a project name. A *project* file is created to hold information pertaining to map layer names and input parameter values. This is done in order to complete the process with a minimum of input required at each step, and to maintain a record of parameters used to obtain a particular result.

The user can exit from the program after completing any number of steps. Hence, the option to *Work on an existing project* allows a user to finish any remaining steps, or to redo particular steps, or both.

A new project may be created with all new inputs, or a new project can be created using inputs from a previous project.

The option to remove old files should be used to remove only miscellaneous files created as part of the watershed program. All map layers (cell files) created using this program should be removed

in the usual way using command `Gremove[2G]`. The files which can be removed using option 4 above include those storing pit points and the miscellaneous file storing nodes information created as part of the step to code stream segments. These files should not be removed until the analysis of which they are a part is finished and over. An option is also provided to remove project files. Again, since this is a file which stores information to complete the project and which tells of the input parameters used to create a particular product, this should also not be removed prematurely. See the rest of this document for further explanation of these files and their usage.

Once an old or new project has been chosen, the user will be presented with the following menu and allowed to choose an option:

- | |
|---|
| <ol style="list-style-type: none"> 1. Filtering of elevation data 2. Locating pits 3. Calculating drainage accumulation/
outlining watershed 4. Creating stream network |
|---|

The steps are presented in the order they should be used to complete the total watershed analysis for the purposes of creating all products. While input is being accepted for each individual step, the option is given whether to run the actual program. If the program has been run and completed without error, the word **COMPLETE** will appear on the above menu after the listing of that step.

Example Session:

Following is an example session that can be used on the Spearfish database. The watershed command can only be reached from the quick access version of GRASS. Once you have entered GRASS, simply enter the command `watershed` with no arguments.

Menus as you should see them are presented here. The step which calculates drainage accumulation may take a good deal of time to complete, depending on the size of the current window. That is why we carefully set the current window before starting.

Now enter the GRASS command *watershed* and you should see the following on your screen.

watershed

WELCOME TO THE WATERSHED SOFTWARE SYSTEM

This program is designed to help you follow the correct steps to obtain a GRASS raster file depicting watershed geometry using a raster elevation file (with true elevation values) as a starting point. All steps are recorded under a project name.

Choose desired option:

1. **Create new project**
2. Create new project based on currently existing project
3. Work on an existing project

4. Remove old files

You will need to start by choosing *Create new project*. Then enter a project name. For this example, the project name entered is demo.

PROJECT NAME: watershed

Choose desired option:

1. Filtering of elevation data
2. Locating pits
3. Calculating drainage accumulation/outlining watershed
4. Creating stream network
5. Coding stream segments/finding segment lengths
6. Finding subwatershed basins

7. Quit

You can now begin the series of steps to complete a watershed analysis. the steps are presented in this menu in the order they should be completed, so begin with step 1, *Filtering of elevation data*. Use as input the map layer elevation.

PROJECT NAME: watershed

Step1: Filtering of elevation data

Input:
 Unscaled elevation map layer: **elevation**_____

Output:
 Filtered elevation map layer: **elev.f1**_____

Number of filter iterations: **1**_____

Run filter program? **yes**

Note that at each step you are given the option of running the program. If you enter *no* here, the map layer names and input parameters will be recorded under the project name, but the word *COMPLETE* will not appear on the main menu. If you choose to run the program by entering *yes*, the next time you are returned to the main menu you will see the following provided the routing completed without error.

PROJECT NAME: watershed

Choose desired option:

1. Filtering of elevation data COMPLETE
2. Locating pits
3. Calculating drainage accumulation/outlining watershed
4. Creating stream network
5. Coding stream segments/finding segment lengths
6. Finding subwatershed basins
7. Quit

Now run the next step *Locating pits*.

DRAFT

D
R
A
F
T

PROJECT NAME: watershed

Step2: Locating pits

Input:

Will use filtered elevation data from Step 1.

Output:

Will create pits file under element watershed/pits,
with name of [filtered] elevation data.

Run pits program? **yes**

Before running the next step, *Calculating drainage accumulation/outlining watershed*, you need to know the grid coordinates for the outlet point. For this example, UTM coordinates have been supplied.

Note that the lakes map layer is optional. There is no need for you to save this output in a cell filed unless you are interested in seeing a display of the pit points, their surrounding lake areas, and the redirected route from the pit point to the pouring point for the lake.

This step may take quite some time to complete, depending on the size of the window and the number of pit points found. Processing of the pits in this routine is lengthy. Once the step is completed, you should exit the watershed program and check the results. If the results of this step are not meaningful, the rest of the process cannot be completed. Use `Gdescribe[2G]` to see if the values in the result file are large, or display the resulting map layer to see if it contains a depiction of the desired drainage basin. (This resulting map layer will need to have a color table assigned. This can be done through the use of the `support[1]` command. If the range of category values is large, the map layer may need to be rescaled first using the `rescale[1]` command in order to make a meaningful display on the color monitor screen.) If the results of this step are not satisfactory, attempt to choose a better watershed outlet point before reentering the program.

Once you have evaluated the results of this step, re-enter the watershed command, and this time choose the option to “Work on an

existing project. Again, enter the project name demo, and you should receive the main menu, with the choice of steps, and the word COMPLETE after the first three options. If the results from this routine were not satisfactory, you would rerun the third step. You should have gotten meaningful results with this example, so continue by choosing option 4.

```

PROJECT NAME:  watershed

Step4: Thinning of accumulation file

Input:
  Will use drainage accumulation map layer from Step 3

Output:
  Thinned network map layer:      drain.thin_____
  Number of thinning iterations:  4_____
  Accumulation threshold:         300____

  Run thin program? yes

```

Again, you need to check the results of this step to be sure that the stream network has been thinned to one pixel width. Once this has been accomplished, continue to the next step. If the stream network has not been thinned to one pixel width, an error message will be produced during the next step and it will not complete correctly.

```

PROJECT NAME:  watershed

Step5: Channel segment coding

Input:
  Extended thinned network map layer:  drain.thin_____

Output:
  Coded network map layer (optional):  drain.coded_____
  Nodes file will have extended network name

  Run code program? yes

```

DRAFT

The cell file depicting the stream network with each channel segment “coded” (given a unique category number) is an optional product. You only need to provide a map layer name for this if you wish to save the results for displaying. However, this step must be completed in order to continue to the next and final step in the series. Note that the above menu asks for an extended thinned network map layer. If you are planning to use results from this series for input into a sediment runoff routine (as through the command `grass.arm-sed[1]`), the stream segments may need to be extended to ridge boundaries in order to provide the kind of information needed, such as the lengths of channel segments which completely divide subbasins. The extension can be created in a manner similar to that used for creating incomplete ridge information as explained in Appendix A of this document.

D
R
A
F
T

PROJECT NAME: watershed

Step6: Subwatershed designation

Input:

Will use nodes file from Step 5.

Output:

Subwatershed map layer: **drain.sub**_____

Run subbasin program? **yes**

Once this step is completed, you should now see the main menu with the word COMPLETE after each step (see below).

PROJECT NAME: watershed

Choose desired option:

1. Filtering of elevation data COMPLETE
2. Locating pits COMPLETE
3. Calculating drainage accumulation/outlining watershed COMPLETE
4. Creating stream network COMPLETE
5. Coding stream segments/finding segment lengths COMPLETE
6. Finding subwatershed basins COMPLETE
7. Quit

This concludes the example session.

DRAFT

DRAFT

Chapter 7 Sites Conversion

Objectives:

Often times data which we wish to incorporate into a project is available in digital form but does not come in a format suitable for grass to use. This does not mean that the information cannot be used, but instead that it must be manipulated so that it can be used. In this chapter we will implement various tools and utilities from GRASS, UNIX, and the Visual Editor (VI) to convert digital site information for use as a GRASS sites list. We will also examine the d.sites and Dpoints modules for displaying newly converted sites.

At the end of this chapter you should be able to:

1. Be able to write simple executable scripts which can be used to speed up repeated jobs.
2. Use the following GRASS commands:
 - Mll2u, Mu2ll, Mgc2ll, Mll2gc
 - d.sites, Dpoints
3. Use the following UNIX commands and operators:
 - cut, paste
 - grep
 - more
 - cd, pwd
 - chmod
 - |, <, >, >>
4. Use the following functions in VI.
 - search and replace
 - insert and remove text
 - cursor movement

DRAFT

Before we begin:

Before we start with this project, make sure that your window is set properly and that the monitor is in fixed mode by executing the following commands.

- Gwindow default
- Dcolormode fixed
- Dscreen

Sites conversion:

In the following example, a list of several sites has been provided in a simple ascii text file for use in the Spearfish sample database. However, the file not only contains extra information, such as the words degrees and minutes, but also contains sites which are defined by latitude-longitude. Since GRASS does not use lat/lon coordinates in site files, we will have to convert these locations using the `Mll2u[]` command, then build a site file in the appropriate directory using the converted information.

First, let's look at an actual GRASS sites file so that we can get an understanding of how it is formatted. The file should be in your home directory which you can easily get to by typing `cd` without specifying a path. The sites file is called `grass_sites`. Use the UNIX command `more` to view the file.

```
cd
more grass_sites
```

```
name|archsites
desc|Potential historic and archeological sites

593493|4914730|1 Signature Rock
591950|4923000|2 No Name
589860|4922000|3 Canyon Station
```

Note the order of the coordinates found in the three fields of the file; easting, northing, and label. The easting and northing are all that is required to locate a site. The label is only used to uniquely identify each particular site. Also notice the fields are separated by a “|” (pipe) symbol. This pipe symbol is commonly used as a delimiter in *ascii* databases.

Now, look at the file which contains the new site information by again using the `more` command. The new sites file name is `site.-`

more sitedata

data.

The sitedata file should look like the following:

```
latitude 44 degrees 22 minutes 54 seconds north, longitude 103
degrees 49 minutes 34 seconds west
latitude 44 degrees 27 minutes 23 seconds north, longitude 103
degrees 50 minutes 39 seconds west
latitude 44 degrees 26 minutes 51 seconds north, longitude 103
degrees 52 minutes 14 seconds west
latitude 44 degrees 27 minutes 17 seconds north, longitude 103
degrees 51 minutes 49 seconds west
```

You can see that there are several things that have to be changed to make *site.data* appear similar to *grass_sites*.

- Lat/lon values must be converted to UTM.
- Extra, unnecessary text must be removed.
- Pipe symbols must be added to delimit the data.

Looking ahead then, we can suggest a method for converting the sites to a GRASS sites file. Those steps would be:

1. Edit the file to make it appear as a series of GRASS commands (shell script), all of which send their output to a single, shared file (*sites.share*).
2. Change the file mode of the newly created shell script so that it is executable and run the script.
3. Extract the eastings and northings from the *sites.share* file directing each to their own files; *eastings* and *northings*.
4. Paste the two files, *eastings*, *northings*, as well as *labels*, side by side into one file, *sites.list*.
5. Edit the *sites.list* file to remove extra text and add pipe symbols as delimiters.
6. Copy the file to the appropriate GRASS sites directory so that it may be included in our mapset's sites list.

Generation of Shell Script:

The GRASS conversion program that we will use is `Mll2u` (lat/lon to UTM). `Mll2u` converts geographic coordinates to UTM coordinates. Geographic coordinates are, of course, latitude and longitude. The resulting information yields an easting, a northing, and a UTM zone in which the coordinate lies. The syntax for the command is:

DRAFT

```
Mll2u lat=dd.mm.ss{n|s} lon=dd.mm.ss{e|w} s=spheroid [z=zone]
```

Since the results normally are directed to standard output (the text screen), we can redirect the output to a file using the redirect (>) unix operator or the redirect/append (>>) unix operator to send the information to a particular file. An example might be:

```
Mll2u lat=35.25.31n lon=90.45.16w s=clark66 > myhome
```

Executing this command would create the file *myhome* which would contain an easting, northing, and zone much like the following:

```
e=703866.86  
n=3922323.95  
z=15
```

D
R
A
F
T

What we will do is change the site.data file to resemble this command repeated for the number of sites listed in the file. To make the edits necessary, VI (the Visual Editor) will be used. VI is an ascii text editor which has very powerful commands available within it. However, to an unfamiliar user, VI can be a confusing and unfriendly enemy. Follow the directions below carefully and you will have no trouble using the editor.

VI operates in three modes, text insert/edit mode, command mode and command line mode. To move the cursor use j to go down, k to go up, h to move left and l to move right. The text insert edit mode is used for adding text to a file or for editing text which has just been typed. You cannot delete text created at another time while in text mode. Command mode allows you to perform pre-defined functions on the file like delete characters and lines, insert blank lines and start new text insertions. It also allows you access to the command line mode which is used to read and write files, find text, and replace text among many others. When you are in doubt as to which mode you are in at any given time, hit Esc slowly until you hear a beep from the keyboard. This will always return you to the command mode.

Command line mode can be reached by typing a “:” from the command mode of VI. Upon typing a “:” the cursor will move to the last line of the screen leaving only the colon on that line. When you have reached the command line you may type your VI command and execute it by hitting return.

We will be primarily using the replace text function of the command line mode in this session. It appears rather cryptic at first, but like most functions in VI, it becomes more clear with use.

Open the file *site.data* by executing the command *vi sitedata*.

vi sitedata

The screen appears much like it did when you used the more command, however, you are now in VI. Notice the status line at the bottom of the screen. It tells you the name of the file you are editing as well as the number of lines and characters in the file.

For the purpose of being clear in this document, a • will be used to indicate spaces. Consequently, when using a search and replace command, only type a space where you see a •, otherwise just type all the letters and commands as one long “word.”

Working from left to right we will make each line appear like the M12u command. The first step is to replace *latitude•* with *M12u•lat=*. Go to the command line by typing: . Then follow that with *g/latitude•/s//M12u•lat=/*. This command can be interpreted as follows: Globally (g) search for (/) the text string latitude• and substitute for it (/s//) the text string M12u•lat= replacing the first occurrence on each line (/).

:g/latitude•/s//M12u•lat=/

After you execute this command by hitting return, the text will be replace and the resulting screen should resemble the following figure. If it does not, immediately hit the u key to undo your last command. Try the replace command again.

```
M12u lat=44 degrees 22 minutes 54 seconds north, longitude 103
degrees 49 minutes 34 seconds west
M12u lat=44 degrees 27 minutes 23 seconds north, longitude 103
degrees 50 minutes 39 seconds west
M12u lat=44 degrees 26 minutes 51 seconds north, longitude 103
```

Next, replace the first •degrees• with a . by typing *:g•degrees•/s/\./*. Note the backslash used before the period, it is used to imply a literal meaning to VI. Normally, a “.” means “repeat the last command” to VI, but when preceded with a backslash it gives it a literal meaning.

:g•degrees•/s/\./

Change the •minutes• to a “.”

:g•minutes•/s/\./

Change the •seconds• to a “.”

:g•seconds•/s//

The file should now appear as follows:

```

M112u lat=44.28.53north, longitude 103 degrees 41 minutes 23 seconds
west
M112u lat=44.23.24north, longitude 103 degrees 38 minutes 29 seconds
west
M112u lat=44.28.37north, longitude 103 degrees 43 minutes 30 sec-

```

Next, change the *orth*, *longitude* (leaving the n) to *lon*.

`:g/orth,longitude/s//lon=`

The file adjusts as below:

```

M112u lat=44.22.54n lon=103 degrees 49 minutes 34 seconds west
M112u lat=44.27.23n lon=103 degrees 50 minutes 39 seconds west
M112u lat=44.26.51n lon=103 degrees 52 minutes 14 seconds west
:
:

```

Replace the *degrees* in the second half of each line:

`:g/degrees/s/\./`

Replace the *minutes*:

`:g/minutes/s/\./`

Your file should now look like this:

```

M112u lat=44.22.54n lon=103.49.34 seconds west
M112u lat=44.27.23n lon=103.50.39 seconds west
M112u lat=44.26.51n lon=103.52.14 seconds west
:
:

```

`:g/seconds/s///`

Remove the *seconds*:

`:g/est/s//s=clark66/`

So as to redirect and append the output to a file (*sites.share*) add the necessary redirects to the end of the command:

`:g/$/s//>>sites.share/`

DRAFT

```
Mll2u lat=44.22.54n lon=103.49.34w s=clark66 >> sites.share
Mll2u lat=44.27.23n lon=103.50.39w s=clark66 >> sites.share
Mll2u lat=44.26.51n lon=103.52.14w s=clark66 >> sites.share
:
:
```

The file will not have been created when the first Mll2u command is executed, so we will have to replace the “>>” in the first line with a “>” to create the file. All subsequent Mll2u commands will append their output to *sites.share*.

To do this, move your cursor to the first line by typing *O* followed by *G*. Then type a */* followed by *>>* (and return) to locate your cursor at the first occurrence of a *>>*. Hit *x* to delete one of the *>*'s. Your file should now look like the following:

```
1G
/>>>
x
```

```
Mll2u lat=44.22.54n lon=103.49.34w s=clark66 > sites.share
Mll2u lat=44.27.23n lon=103.50.39w s=clark66 >> sites.share
Mll2u lat=44.26.51n lon=103.52.14w s=clark66 >> sites.share
:
:
```

Save the file by typing *ZZ*. You have now created the shell script.

```
ZZ
```

Changing the File Mode:

Before we can run the shell script as if it were an executable program, we have to change the file mode to add an executable tag to the file. To do this type: ***chmod 700 sitedata***.

```
chmod 700 sitedata
```

The shell script is now executable. To run the script simply type ***site.data*** and ***sites.share*** will be generated.

```
site.data
```

Extraction of Eastings and Northings:

The next step will be to extract the eastings and northings from the newly created `sites.share` file. First look at the file by using the *more* command.

```
more sites.share
```

The file should appear similar to the one below:

```
e=593512.955007
n=4914718.614975
z=13
e=591957.427751
n=4922997.964279
z=13
e=589871.569362
n=4921981.293457
```

To extract the eastings and northings from this file we will use the UNIX commands *cat* and *grep*. First, extract the eastings to a file by typing: *cat sites.share | grep e= > eastings*.

```
cat sites.share | grep e= > eastings
```

Next, extract the northings by typing *cat sites.share | grep n= > northings*.

```
cat sites.share | grep n= > northings
```

Pasting Files:

We now have two files which can be put side by side in one file to resemble a sites list. To save more work, the file *labels* has been provided for you. It is merely a list of site names to accompany the coordinate pairs that we have been working with. Use the command *paste eastings northings labels > sites.list* to put the three files side by side in the new file *sites.list*.

```
paste eastings northings labels > sites.list
```

Paste, by default, delimits files with a tab between the original files. Sometimes these tabs can appear as spaces but don't be fooled, they are really tabs. The next step will be to remove the tabs as well as the extra characters which precede each coordinate.

Editing Pasted File:

Again, we will use VI to edit our file. However, this time we will be attempting to make our file look like a GRASS sites list rather than a series of GRASS or UNIX commands. To begin, type *vi*

vi sites.list

sites.list.

The screen redraws and you see something like the following:

```
e=593512.955007 n=4914718.614975    Signature Rock
e=591957.427751 n=4922997.964279    No Name
e=589871.569362 n=4921981.293457    Canyon Station
:
```

First remove the *e=* from the beginning of each line:

```
:g/e=/s///
```

Then replace the *<tab>n=* with a pipe symbol. The pipe symbol has a specific meaning to VI so it will have to be preceded by a \ to force the literal use of it.

```
:g/<tab>n=/s/^\|
```

Finally, replace the second tab with a pipe symbol:

```
:g/<tab>/s/^\|
```

Your sites.list file should now look like this, a correctly formatted GRASS site file.

```
593512.955007|4914718.614975|Signature Rock
591957.427751|4922997.964279|No Name
589871.569362|4921981.293457|Canyon Station
:
```

Copy the File:

The only thing left to do is copy the file to the appropriate directory for grass to use it. Our location is spearfish and our mapset is PERMANENT which means that all of our data can be found in /home/grass3.data/spearfish/PERMANENT. The site information, then, falls in the /home/grass3.data/spearfish/PERMANENT/site_lists directory. Copy the file to that directory using the command *cp sites.list /home/grass3.data/spearfish/PERMANENT/site_lists*.

```
cp sites.list /home/grass3.data/spearfish/PERMANENT/site_lists
```

Now use *list sites* to make sure that *sites.list* is found in your list of site files.

```
list sites
```

Display of Sites.list:

DRAFT

To view the sites, first put up a background map such as elevation using the *Dcell elevation* command.

Dcell elevation

Now run the GRASS *d.sites* program.

d.sites

Specify *sites.list* as the name of the sites list to be displayed.

sites.list

Choose a symbol type to use (i.e. +).

2

Choose a color; yellow is good.

3

Choose a size for the icon; three is good.

3

GRASS overlays the sites on top of the cell map. At this point you can rerun the *d.sites* program and duplicate the site with another icon or color to make them stand out or display another one of the lists that appears in the mapset.

You could also use the command *Dpoints color=yellow size=3 type=+ </home/grass3.data/spearfish/PERMANENT/site_lists/sites.list*. It is much longer but does the job with no questions asked.

This concludes the sites conversion exercise.

D
R
A
F
T

Chapter 8 *Networking & Hardware*

Objectives:

In this chapter we will discuss aspects of the physical portion, or hardware, of a GIS. As you already know, a GIS is comprised not only of digital data and software to manipulate it, but also the hardware which stores the data and executes the software. CPU's, hard disks, optical storage devices, tape backups, graphic and textual displays, output devices and so on collectively make up the hardware portion of a GIS.

Decisions regarding what types of hardware will be acquired for developing a GIS are crucial to the installation's success. Knowing your hardware requirements (as well as software requirements) can help to develop a system which best serves your needs.

One specific consideration involves data sharing between computers which is accomplished via a network. Networking provides a means of moving data across a room, a campus, or the nation using coaxial cable, phone lines and perhaps even radio frequency transmissions. Understanding the basic operations of a networking system can help you avoid duplication, speed the transferring of files between computers and open another world of electronic communication.

At the end of this chapter you should be able to:

1. Describe the layering concept of a network protocol and the movement of data through the various layers.
2. Understand the various output devices available with in GRASS.
3. Define the following terms:
 - Peripheral
 - LAN, CAN, WAN
 -

Hardware:

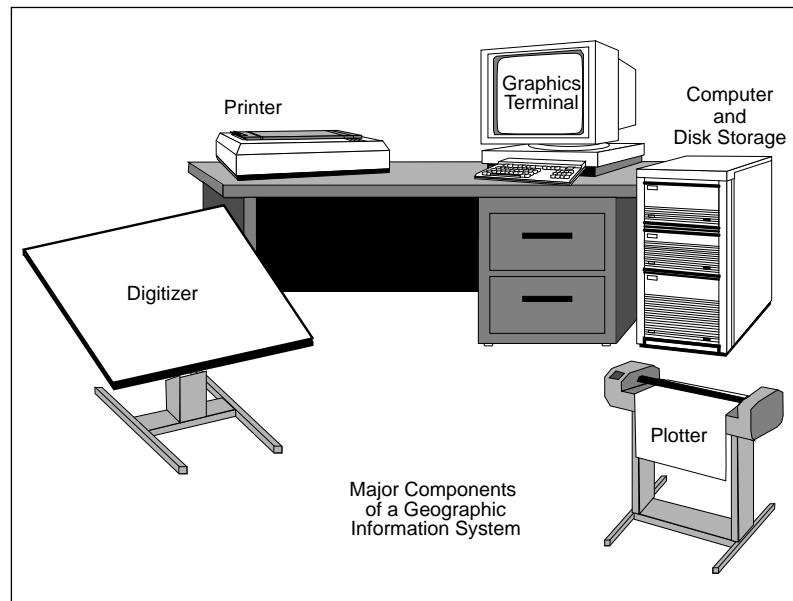
DRAFT

Computers, as you may well know, come in all shapes, sizes and functionalities. They range in size and price from relatively inexpensive notebook sized personal computers to more expensive desktop pc's and workstations and on to mini-mainframes and full-blown mainframe systems possibly costing several million dollars. Each of these types of computers has its advantages.

The hardware of a GIS can be grouped into two general categories, the computer itself and the peripherals attached to or accessible by it. Standard configurations for a computer include a central processing unit (CPU) which usually includes a disk storage device, and a display device which is either a text or graphic display. Peripherals are then digitizers, plotters, and printers.

For a GIS, several components are common. Usually they include:

- Computer / Hard Disk
- Graphics Terminal
- Digitizer / Scanner
- Plotter
- Printer
- Tape Drive (beta, cassette, 9-track)



GIS databases are usually vary large requiring big hard disks. It is not uncommon for a GIS to have 300 to 2000 megabytes of disk storage available. Raster based GIS's are especially demanding of disk space while vector systems are often less demanding. No only

DRAFT

must there be large amounts of disk space, but the CPU must be sufficiently powerful to work with the large amounts of data in a reasonable amount of time. Today computers are getting faster and more powerful so considerations of processing power are not a great issue.

Since one of the stronger functions of a GIS is the softcopy map, a high quality graphics display is needed to show maps before printing. Graphics monitors range in price and quality from under \$1,000 to \$5,000 and more.

Another key ingredient in a GIS is some means for getting analog data into digital format. For this, a digitizer or scanner is used. By tracing map features on a digitizer's surface or by sensing contrast changes with a scanner you can develop digital data layers from a paper map.

Statistical reports are a strong feature of any GIS which require some kind of text output device. For this a line printer may be added to the system.

Map production requires some sort of quality plotter whether it be a large pen plotter or an inkjet printer. There are several plotters on the market today ranging in price from about \$1,500 to over \$250,000.

To use third-party data such as DLG's or DEM's from the USGS some sort of tape device is needed. For example, DLG's and DEM's are provided on 9-track tapes. Tape drives are also used for backing up data on the hard disk.

Together, these components make up a fully functional GIS hardware system. Data and software must be added to complete the GIS.

Networks:

A network is one component which can be added to a GIS to provide the sharing of resources between computers. It is often implemented when more than one GIS is used at a site. By using a network more than one computer can share printers, plotters and other peripherals.

There are several types of networks which can be used at a site involving as little as two computers and as many as hundreds of

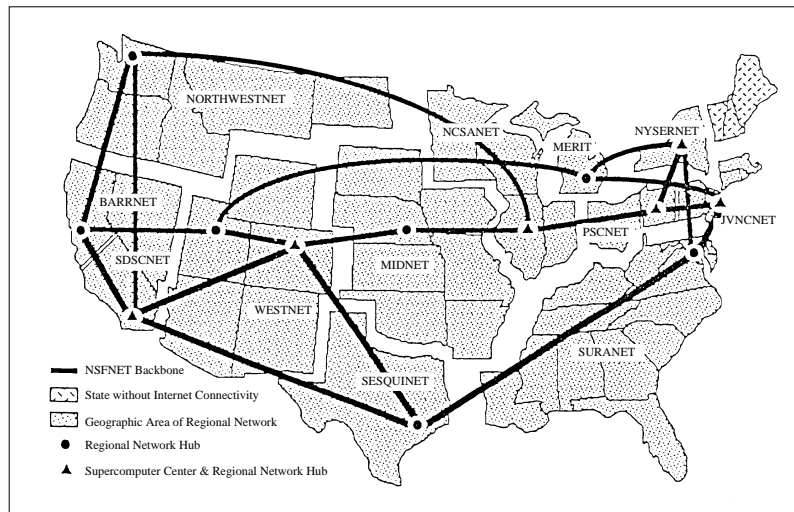
DRAFT

thousands across the nation and the world. A few are listed below.

- LAN - Local Area Network - allows local (in house) computers to exchange resources.
- CAN - Campus Area Network - one network implemented within several buildings - may include more than one LAN.
- Backbone Network - joins LANs, provides high speed inter-connection and joins dissimilar networks.
- WAN - Wide Area Network - allows exchange with other institutions and provides access to other national and international networks.

There are several networks in place across the nation, each with their own protocol. A few are:

- BITNET - Because It's Time Network - almost 500 member institutions of higher ed which share information.
- The Internet - a world-wide system of networks based on the TCP/IP protocol suite originated by the Department of Defense.
- NSFnet - National Science Foundation Network - an international internet consisting of lines running at high speed.
- Midnet - Midwest Network - a regional internet network which connects to NSFnet and to which UARKnet connects.

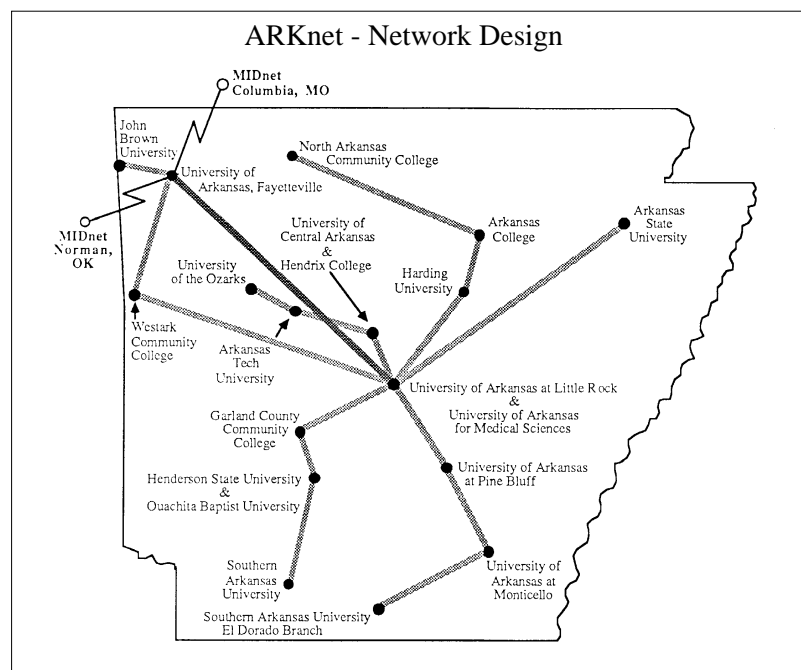


There are also several statewide networks in place or soon to be in place. A few of those are:

- UARKnet - The University of Arkansas Network - a campus area network in the larger scheme of things.
- CERFnet - The California Education and Research Network.

DRAFT

- KRNet - Kansas Regional Network
- MORNET - Missouri Research Network
- MRNet - Minnesota Regional Network
- NYSERNet - New York State Education and Research Network.
- OARnet - Ohio Academic Resource Network
- PREPnet - Pennsylvania Research and Economic Partnership Network.
- THENet - Texas Higher Education Network
- VERnet - Virginia Education and Research Network
- WiscNet - The Wisconsin Network



MIDARK has as a potential extension a GISnet used specifically for the exchange of geographic data.

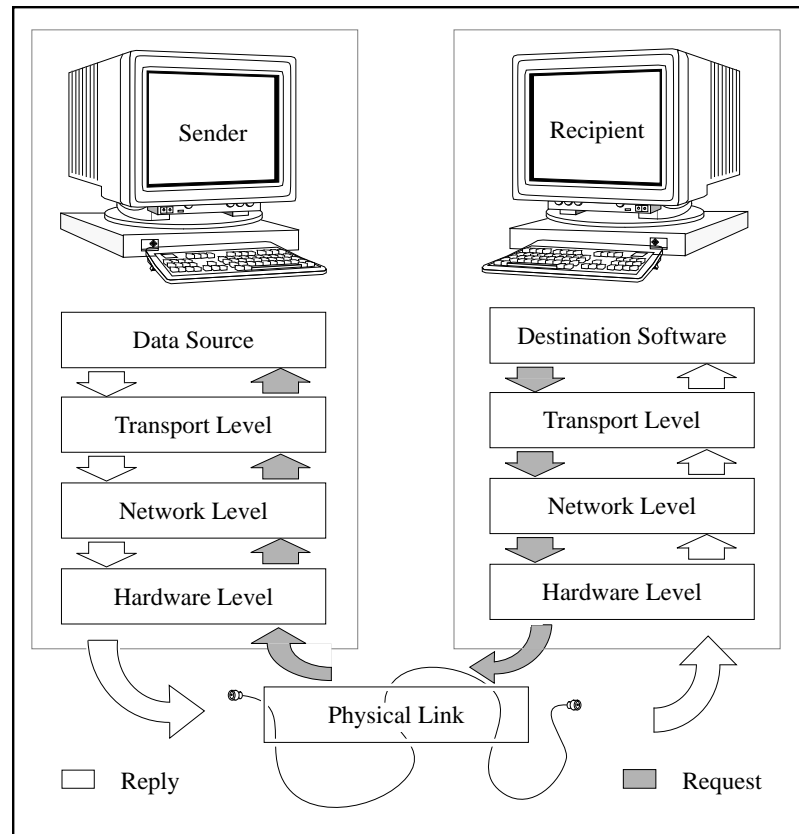
Network Protocol:

Since the cable connecting several computers on a network carries information for all of the computers on it, a means for controlling the flow of information to and from specific computers is necessary. This is accomplished by using a network protocol which formats transmissions that travel over the network.

Usually, data is sent in packets or chunks rather than all at once.

This is for a couple of reasons; 1. since the network is shared, sending large amounts of data in a steady stream would “hog” the network forcing others to wait, and 2. parity checking is used to make sure that data has been correctly received and in the case of a bad transmission it is easier to retransmit a small packet rather than a large one.

Preparing data to travel over a network involves both software and hardware. Four steps (levels) are used on both the sending and receiving computers; a software/data level, a transport level, a network level and finally a hardware level. See the diagram below.



The software/data level interacts with the software and data in use by the user. It implements new software routines which begins the process of preparing data for transmission. Some common software protocols are telnet, ftp (file transfer protocol) and DNS (Domain Name Service).

The transport level serves as a link between processes running on separate machines by making them appear as one to the software/data level.

The network level determines how the data is to be transmitted and where. At this stage destination id's are attached to packets before transmittal. This level also receives incoming by looking at packet id's and "pulling off" those addressed to it. Upon receipt of a packet the network level forwards the packet to the transport level for decoding.

The hardware level is what actually transmits the packets as voltages out onto the network. A common type of hardware level protocol is the Ethernet protocol.

Summary:

Several hardware components make up a GIS. Some of those are CPU and hard disk, graphics terminal, printer, plotter, and digitizer. Additional components include networks which can be used to share data and resources between individual computers making the whole of the network a greater utility than the sum of its parts.

DRAFT

Appendix **A** Readings

The following is a paper by Ian K. Crain which appeared in the Auto-Carto Proceedings for 1989. It does an excellent job of explaining the development of Geographic Information Systems in Canada in the past two decades. Canada has been a pioneer of GIS in North America and offers much in the way of experience to anyone interested in developing a GIS.

FROM LAND INVENTORY TO LAND MANAGEMENT THE EVOLUTION OF AN OPERATIONAL GIS

I.K. Crain
and
C.L. MacDonald
Canada Land Data Systems
Environment Canada, Ottawa, Ontario K1A 0E7

ABSTRACT

Virtually all long-lived information systems, whether spatial or not, proceed (by intent or by unconscious incremental change) through a characteristic evolution from a simple data bank - an inventory tool, to incorporate complex retrieval and statistical processing - an analysis tool, and finally to a system involving modelling and complex decision support capabilities - a management tool. This evolution is evident and still going on in the world's oldest operational geographic information system - the Canada Land Data System (CLDS). The system began with the specific intent of serving the needs of the Canada Land Inventory - to store very extensive and fairly detailed data on the land use potential of Canada. This initial goal having been achieved, increasing demands were placed on the system to explore data and spatial relationships within this huge data bank, and to provide insight for land resource scientists into more localized and specialized concerns - in short, an analysis capability. The CLDS is now entering the third phase of evolution to meet the inevitable demands as a land planning and management tool. The implications of this are mainly: increased user access and interactivity with existing data, forecasting, planning and modelling capabilities, enhanced variety of display outputs, ability to service non-expert users locally and interactively for small-area data sets in a land use planning setting.

DRAFT

INTRODUCTION

Information systems, regardless of their field of application, have certain elements in common - a large central bank of data, facilities for manipulating, retrieving, updating, and reporting of data. The common principles and capabilities have been extensively studied and reported in the literature and will not be re-iterated here. Another commonality amongst information systems, again independent of application - whether it is a personnel system, financial system, management system, is a tendency to progress through a characteristic evolution which can be conveniently divided into three stages. These stages are typical of and perhaps necessary to a successful information system. Systems die, or are murdered, mainly for their inability to evolve in this pattern. In this generalized sense, geographic information systems are no different, and to succeed must be capable of following the same evolutionary pattern.

TABLE 1

P H A S E	SYSTEM CAPABILITIES	PRIMARY ACTIVITY	USER/SUPPLIER RELATIONSHIP
I N V E N T O R Y	-data input -editing -simple retrieval -routine reporting	- data input	-clear division between client and supplier -little interaction
A N A L Y S I S	all above plus: -complex retrievals -ad hoc queries -statistical processing -derived reporting (eg. graphics) -derived data sets	-data retrieval and manipulation	-supplier involved in determining output needs -interactive retrievals and direct data access by user
M A N A G E	all above plus: -modelling, simulation -decision support tools	-data retrieval and manipulation	-user and supplier indistinguishable -fully interactive

This paper describes the principal stages of the evolutionary process and elaborates them in terms of a particular system (or group

DRAFT

of systems) known as the Canada Land Data System (CLDS), which contains the Canada Geographic Information System (CGIS). This system has GIS and has in that period undergone considerable evolution and adaptation, so that it continues to be a highly successful broad-scope GIS.

CHARACTERISTIC STAGES OF INFORMATION SYSTEM EVOLUTION

The stages as outline are essentially arbitrary and dividing lines between them are necessarily fuzzy. It is recognized as well that successful systems probably exist which have not followed this pattern for one reason or another. Such counter-examples should not detract from the general principle described. The three characteristic stages are named (again arbitrarily) by their principal functional purpose as Inventory, Analysis and Management. The main characteristics of the phases are outline in (Table 1).

As the system functionally evolves, the organizational relationship between the user and supplier of system services must change as well. The changes in this environment parallel to some extent the general evolutionary pattern of all EDP organizations as defined by (Nolan 1979).

Phase I: Inventory

In this initial phase of an information system, the reason for its existence or development is to assemble, organize and determine the extent of existence of data on a particular subject. In a personnel system for example, the initial thrust is normally to develop a file of basic common information about each employee. At this phase the system serves to answer data queries of how much?, how many?, where?, with an emphasis on reports which derive directly from the data in the field based on summations, counts and other minor manipulations of the raw data. In short, an inventory function.

The principal activities are data collection, input, and editing - in general those activities which ensure the development of an accurate, high integrity base of primary data.

For a GIS this represents a system whose primary task is the development of a broad integrated collection of mapped or geocoded data which may encompass a broad area, but of very well defined scope, e.g. all the forestry maps for a province, with capabilities to answer direct queries on area by selected criteria, or to redisplay the

DRAFT

data in various ways.

During this phase, the relationship between the user and the supplier of system services tends to be simple and consist of a well defined separation of function - a classic customer/client relationship. System functions are well known and relatively simple, the user "menu" is well defined. The supplier's role is one of ensuring the integrity of the data, ensuring on-going operational continuity of software and hardware and providing standard (albeit complex) reports and outputs routinely or on demand.

Phase II - Analysis

The second phase of the evolution is spawned by the user's desire to make more use of the data than direct tabulations and summaries - the desire to explore data relationships in order to shed light on subject matter problems, to help confirm hypotheses or to provide data for research and modelling. In the personnel system example, the demand might be to examine relationships between occupational level and place of residence, or to search for evidence of relationships between salary and ethnic origin, etc.

The emphasis in this phase is much more on complex retrievals, and queries which may generate additional queries in an unexpected or unstructured way, hence the implications of a need for extensive user interaction. Advanced statistical analysis tools are required and the queries tend to relate much more indirectly to the raw data.

The queries, in fact, become more like those typical of a "scientific" system (Crain 1978) in that they do not relate directly to the record ("tell me all about employee number 1234"), but rather span across records seeking relationships ("tell me the age distribution and correlation with salary of female employees living in the city center").

The principal activity moves from data gathering to data retrieval and manipulation. New data inputs are less likely to be broad and comprehensive, but will be more specialized to focus on particular subject matter concerns.

In the case of a GIS, the primary requirement is for user interaction, usually graphically, with the data - the ability to sort, select, derive new data from old, extract and re-display data on the basis of complex geographic, topological and statistical criteria. New data collections may still represent very high data volumes, but cover

D
R
A
F
T

smaller areas in more detail.

Ap[art from the obvious need for the system supplier to provide interactive services, the supplier of an analysis system must evolve organizationally to become more involved in subject matter and data context issues. In order to develop the necessary system tools to meet user requirements, considerable emphasis must be placed on determining and anticipating user information needs in the short and long term, and an understanding of the existing and evolving data analysis techniques of subject areas. While most data will still be held centrally, there is a growing decentralization of user access and associated hardware.

Phase III - Management

A true management information system must provide the tools to assist directly in the decision making process - not merely an inventory of what exists, not only analyses of data relationships which might hint at problems or solution. The essential new ingredient for a system to evolve to this phase is the addition of forecasting and planning facilities - the ability for the user to ask "What if....". This implies the need for more advanced mathematical forecasting capabilities. To extend the personnel system example, a personnel management system might for instance, be able to model the process of advancement of minority groups, allowing the manager to forecast potential impact of various strategies for improving their advancement opportunity. A land management GIS might provide answers to "what if?" questions related to alternated land use policies or various strategies for optimization of land use under conflicting demand. It is with these modelling and planning tools, superimposed on a broad base of hard raw data that a GIS reaches maturity as a resource management system.

When a system reaches this point in evolution, the distinction between user and supplier virtually disappears. With the exception of some purely technical EDP staff (such as machine operators and software people) all system "users" must be integrated into an overall system management process. Decentralization of data holdings as well as data access will occur and necessary integrated controls must be developed. Knowledge and skill requirements of those involved cannot be divided on the basis of "computer" or "subject matter" and must be divided instead on functional and managerial lines. This change in relationship is by far the most difficult step in the evolution and the greatest barrier to integrating an MIS or GIS truly into the decision process.

DRAFT

D
R
A
F
T

EVOLUTION OF THE CANADA LAND DATA SYSTEM

The assembly of systems now known as the Canada Land Data System has origins that go back more than twenty year to the “ARDA project” of 1962. During that long period, the system has evolved in the characteristic pattern, previously described and I would like to use the CLDS to exemplify the process.

The original purpose for the system (which has had several names, the most well known of which is the Canada Geographic Information System - CGIS) was to store the Canada Land Inventory (CLI). The actual system was delivered in 1968 (now celebrated as the birthdate for CLDS) and full production operation began in 1971. As implied, it commenced life as an inventory system. The initial data collection was of five coverages of land capability for all the arable land of Canada - an enormous area of some 2.7 million Km².

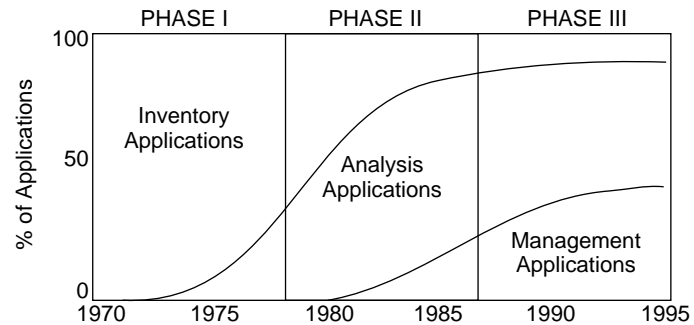
This required the input of approximately 1500 detailed polygon maps, a process which was essentially complete by 1979. These “early days” were concerned by data entry and routine tabular reporting. Raw data tabulations and summary reports were produced on a national and regional basis in standard formats. While considerable topological processing was built into the system - e.g. ability to do multicoverage overlays, reporting and derived graphics facilities were quite limited.

Somewhere in the interval between the commencement of production operations in 1971 and the completion of the Canada Land Inventory in 1979 the system ceased to be solely an inventory system and evolved into an analysis facility. one clear symptom of this was the introduction in 1974 of an interactive graphics subsystem. Users demanded increased ability to relate variables, to select and study particular areas, and to extract derived data for planning, prioritizing and problem analysis.

The subject matter focus changed from a national resource inventory perspective to more specific issues such as arctic land use, coastal zones, national parks, etc. By 1978 more maps of this regional nature were being input than CLI maps. (Figure 1) schematically outlines this evolution. The changes required to bring the system to Phase II include advance statistical program packages, interactive graphics enquiry facilities through a network of terminals

across Canada to provide user access, and improved hard-copy graphics output capability.

Figure 1



Today the system stands fully in Phase II even though data holdings (inventories) still grow at a rate of 1000 maps per year, and is on threshold of Phase III.

Over 350 interactive data bases of complexly overlaid map data are available for user analysis using various systems and subsystems which collectively make up CLDS.

In spite of this, strong demand exists for CLDS to become an integrated land management system and the first steps in this evolutionary phase have begun. Several small applications have used the existing facilities for land planning and management purposes - for camp sit location in National Parks and transportation corridor planning the Beaufort Sea area. Within five years, with suitable system changes, the CLDS will be fully into Phase III.

To accomplish this many new system functions are needed and organizational relationships must change. Land planning models and accompanying facilities to define model parameters from the data bases, and forecast quantitative and qualitative impact must be developed.

At this time, testing is underway of simple linear models which allow for the subjective weighing of land use suitability in order to evaluate the impact of variable land policy options. Data input facilities are being upgraded to reduce the time required to make data available to planners - a necessary step if rapidly changing condi-

DRAFT

tions are to be monitored by resource planners.

The demand is growing for increased local processing capability for local land use management, with continued access to the central regional and national data banks. Preliminary trials are underway to equip mini and micro computers for this purpose.

CHALLENGES TO ACHIEVING PHASE III

The transition to Phase III is not as easy as the movement from inventory to analysis. Many geographic information systems are now facing this transition and face similar challenges. While placed in the specific context of CLDS, these barriers are of general applicability.

- 1) Resource constraints: The resources (both dollars and human) required to enhance and distribute the processing power as needed in Phase III are substantial. In today's cautious economic climate, especially in government, major one time injections of resources for system enhancement are unlikely. Thus the change will have to be made gradually as funds become available. Software development is always very expensive and time consuming, and user demands will always exceed the capacity of software development resources.
- 2) Ensuring continual quality control: Quality control of outputs and maintenance of the integrity of the data bases are increasingly difficult to achieve as the system becomes decentralized. Where the distinction between users and supplier becomes blurred, as it will in Phase III, responsibility for ensuring the correctness of data input, its consistency with global quality and subject matter standards, and its protection from corruption become an administrative issue rather than a technical issue. The degree of collective rather than individual responsibility for this incredibly valuable asset increases greatly and is difficult to achieve.
- 3) Choosing appropriate new technology: This is the constant technical challenge of any evolving system - how to take advantage of the latest technological advance without extreme risk or without jeopardizing future alternatives. It requires constant research and a balance between conservatism - selecting only the proven, and opportunism - selecting the correct new (unproven) machine or software where the risks seem worth it.
- 4) Ensuring continuity of production operation during change: This is both a technical and administrative challenge which is diffi-

D
R
A
F
T

cult to achieve and highly important to on-going success and credibility. It is essential to recognize that a Phase III system is not only a management tool, but will have continued application as an inventory and analysis system as well. The "inventory" of data collected continues to have vital importance and the production processes to maintain it must continue.

SOLUTIONS

The first steps are now underway at CLDS to face the challenges that the transition to Phase III requires. Technical solutions to the need for distributed processing and local data bases are being found through the use of mini computers. The exact configuration is undecided at this time, but the eventual distributed land resource analysis system will likely involve a network of mini computers (or powerful micros) across Canada, linked to the central facilities. As well, it is anticipated that small portable microcomputers will be used in remote locations to analyze small area data sets extracted from the main files. At least one land planning software package is being tested for mini/micro implementation and existing graphics and statistical processing capabilities being upgraded.

The most significant barrier is the organizational/educational one. The disappearance of the distinction between user and supplier can only come through a long process of education and gradual change. The current "user" will need to be greatly more aware of system operations and EDP principles and, moreover, be willing to assume much more responsibility for data control, standards, coordination, quality assurance, etc., than now is the case. The current "suppliers" (EDP professionals) must learn considerably more about the techniques and practices of land use planning, modelling and subject matter data interpretation. It is implied that, in general, knowledge and skill levels of all concerned must be much greater and such multi-qualified people are in extremely short supply. It is expected that this problem will slow the development of a generalized modelling/planning system, although more application specific land management capabilities can be implemented where skill levels are high and appropriate management practices are in place.

As well, the techniques for forecasting land change or quantifying policy impact analysis are not well advanced. Mathematical models which are in use are simple, and clearly preliminary. Much research is required in this field before significant systems advances can be made.

DRAFT

REFERENCES

Crain, I.K. 1978, The Use of Data Base Management Systems in a Scientific Environment: Proceedings Canadian Computer Conference, Canadian Information Processing Society, Montreal, pp. 36-42.

Nolan, Richard L. 1979, Managing the Crises in Data Processing: Harvard Business Review, March-April 1979, pp. 115-126.

D
R
A
F
T

Appendix **B** Glossary

Some of the terms in this glossary have been drawn from the Soil Conservation Service's Fundamentals of Geographic Information Systems training manual. Other terms have been developed from various sources as well as from common knowledge.

- ASCII** - (American Standard Code for Information Interchange) Pronounced "ask-ee." A binary code for data that is used in communications, most minicomputers and all personal computers.
- ASCII file** - A data or text file that contains characters coded in ASCII. Text files, word processing documents, batch files and source language programs are usually ASCII files. Only the first 128 characters (0-127) within the 256 combinations in a byte conform to the ASCII standard. The rest are used differently depending on the computer.
- aspect ratio** - The ratio of width to height of a frame, screen or image. When images are transferred from one system to another, the aspect ratio must be maintained in order to provide an accurate representation of the original.
- attribute** - (1) In relational database management, a field within a record. (2) For printers and display screens, a characteristic that changes a font, for example, from normal to boldface or underlined, or from normal to reverse video.
- benchmark** - A test of performance of a computer or peripheral device. The best benchmark is the actual set of application programs and data files that the organization will use. Running benchmarks on a single user computer is reasonably effective; however, obtaining meaningful results from a benchmark of a multiuser system is a complicated task. Unless the end user environment can be duplicated closely, the benchmark will be of little value. It may be more effective to find a user organization with a similar processing environment and simply monitor the operation. See Linpack, Dhrystones, Whetstones and Khornerstones.
- Bernoulli Box** - A removable disk system from Iomega Corporation that connects to personal computers through a SCSI interface. Introduced in 1983, the original Bernoulli Box was an 8" floppy disk cartridge that held 10MB. Storage was later increased to 20MB. In 1987, the Bernoulli

DRAFT

Box II provided 20MB on a 5 1/4" disk, and in 1989, storage was increased to 44MB per cartridge.

- beta test -** A test of hardware or software that is performed by users under normal operating conditions (prior to formal release).
- binary -** Meaning two; the fundamental principle behind digital computers. All input to the computer is converted into binary numbers made up of the two digits 0 and 1 (bits). For example, when you press the "A" key on your personal computer, the keyboard generates and transmits the number 01000001 to the computer's memory as a series of pulses. The 1 bits are transmitted as high voltage; the 0 bits are transmitted as low voltage. The 1s and 0s are stored as a series of charged and uncharged memory cells in the computer.
- binary file -** A program in machine language form or a file that contains binary numbers. When transmitting files to a remote computer, some protocols handle only ASCII text and cannot be used for binary files.
- dumb terminal -** A display terminal without processing capability. It is entirely dependent on the main computer for processing. Contrast with smart terminal and intelligent terminal.
- environment -** A particular computer's configuration, which sets the standards for the application programs that run in it. It includes the CPU model and system software (operating system, data communications and database systems). It may also include the programming language used. The term often refers only to the operating system; for example, "This program is running in UNIX environment.
- Ethernet -** A local area network developed by Xerox, Digital and Intel that interconnects personal computers via coaxial cable. It uses the CSMA/CD access method and transmits at 10 megabits per second. Ethernet uses a bus topology that can connect up to 1,024 personal computers and workstations within each main branch. Ethernet has evolved into the IEEE 802.3 standard.
- file format -** The specification for the structure of a file. There are hundreds of proprietary formats for database, word processing and graphics files that have become standardized and used for data interchange. See record layout.
- file maintenance -** (1) The periodic updating of master files. For example, adding and deleting employees and customers, making name and address changes and changing the prices in a product file. It does not pertain to an organization's daily transaction processing and batch process-

D
R
A
F
T

ing (order processing, billing, etc.). (2) The periodic reorganization of a computer system's disks. Data that has been continuously updated becomes physically fragmented over the disk space and requires regrouping. An optimizing program is run every cycle (daily, weekly, etc.) that organizes the data contiguously.

- file server -** A high-speed computer in a local area network that stores the programs and data files shared by the users on the network. Also called a network server, it acts like a remote disk drive. If the file server is dedicated to database operations, it is called a database server.
- file sharing protocol -**A communications protocol that provides a structure for file requests (open, read, write, close...) between stations in a network. If file sharing is strictly between workstation and server, it is also called a client/server protocol. It refers to layer 7 of the OSI model.
- file transfer protocol -**A communications protocol that can transmit files without loss of data. It implies that it can handle binary data as well as ASCII data.
- firmware -** A category of memory chips that hold their content without electrical power and include ROM, PROM, EPROM and EEPROM technologies. Firmware becomes "hard software" when holding program code.
- font -** A set of type characters of a particular design and size. In daisy wheel printers, fonts are changed by changing the daisy wheel. In dot matrix printers, the fonts are built in and can be selected by software or a control panel switch. In laser printers, some fonts are built in and others can be plugged in with a cartridge or downloaded from a computer.
- footprint -** The amount of geographic space an object uses. A computer footprint is the amount of desk or floor surface it occupies. A satellite's footprint is the geographic area on earth that is covered by its down-link transmission.
- foreground/background -**The priority given to programs in a multitasking environment. Programs running in the foreground have highest priority, and programs running in the background have lowest priority. For example, online users at terminals are usually given the foreground, and batch processing activities, such as long sorts and updates, are given the background. If the batch processing activities are given the higher priority, terminal response times may slow down considerably. In a personal computer, the foreground program is the one the user is currently working with, and the background program might be a print spooler or a communications program that is ready to receive a mes-

DRAFT

sage from another computer.

fourth-generation language - (4GL) A computer language that is more advanced than traditional high-level programming languages. For example, in dBASE, the command LIST displays all the records in a data file.

In second- and third-generation languages, instructions would have to be written to read each record, test for end of file, place each item of data on screen and go back and repeat the operation until there are no more records to process.

First-generation languages are machine languages; second-generation are machine dependent assembly languages; and third-generation languages are high-level programming languages, such as FORTRAN, COBOL, BASIC, Pascal, and C. Although many languages, such as dBASE, are called fourth-generation languages, they are actually a mix of third and fourth. The dBASE LIST command is a fourth-generation command, but applications programmed in dBASE are third-generation.

Query language and report writers are also fourth-generation languages. Any computer language with English-like commands that doesn't require traditional input-process-output logic falls into this category.

fragmentation - The uneven distribution of data on a disk. As files are updated, they become less contiguous on the disk. When data is added, the operating system stores it in the available free space. As a result, parts of the file wind up in disparate areas of the disk, causing additional arm movement when the file is sequentially read. A disk maintenance, or optimizer, program is used to reorder the files in a contiguous manner.

frame buffer - In computer graphics, a separate memory component that holds a graphic image. Frame buffers may have one plane of memory for each bit in the pixel. For example, if eight bits are used to represent one pixel, there are eight separate memory planes.

full-screen - A programming capability that allows data to be displayed in any row or column on the screen. Contrast with teletype mode.

graphics terminal - (1) An input/output device that is capable of displaying pictures. Images are received via communications or entered with a device, such as a mouse or light pen. The keyboard may have specialized function keys, wheels or dials. Graphics terminals employ raster graphics, vector graphics or combination raster and vector technologies. (2) A terminal or personal computer that displays graphics.

hierarchical - A structure made up of different levels like a company organization

D
R
A
F
T

chart. The higher levels have control or precedence over the lower levels. Hierarchical structures are a one to many relationship; each item having one or more items below it.

In communications, a hierarchical network refers to a single computer that has control over all the nodes connected to it.

- hierarchical file system** - A file organization method that stores data in a top-to-bottom organization structure. All access to the data starts at the top and proceeds throughout the levels of the hierarchy.
- host** - The central computer or controlling computer in a timesharing or distributed processing environment.
- HPGL** - (Hewlett-Packard Graphics Language) A vector graphics file format from HP that was developed as a standard plotter language. Most plotters support the HPGL and DMPL standards.
- instruction set** - The entire group of machine language instructions that a computer can follow. Instruction sets are designed into the CPU and are one of the major components of its architecture. A set may contain from a handful (RISC) to several hundred instructions (CISC). Machine instructions are generally from one to four bytes long.
- internet** - (1) A large network made up of a number of smaller networks. (2) Internet. A national research-oriented network comprised of over a thousand government and academic networks.
- kernel** - The fundamental part of a program, such as an operating system, that resides in memory at all times.
- network** - (1) An arrangement of objects that are interconnected. See local area network and network database. (2) In communications, the transmission channels and supporting hardware and software.
- network administrator** - An individual who is responsible for the operation of a communications network. The network administrator installs applications on the servers, monitors network activity and is generally responsible for its efficient operation.
- open architecture** - A system in which the specifications are made public in order to encourage third-party vendors to develop add-on products for it. For example, much of Apple Computer's early success was due to its open architecture of the Apple II. IBM followed Apple's lead in making the PC open architecture as well.
- Open Look** - An X Window-based graphical user interface (GUI) for the UNIX

DRAFT

operating system that was developed by Sun Microsystems and is defined and distributed by AT&T's UNIX Software Operation (USO). It conforms to POSIX, ANSI C and X/Open's XPG3 standards.

operating system - A master control program that runs the computer and acts as a scheduler and traffic cop. It is the first program loaded (copied) into the computer's memory after the computer is turned on, and the central core, or kernel, of the operating system must reside in memory at all times. The operating system may be developed by the vendor of the hardware it's running in or by an independent software house.

The operating system is an important component of the computer system, because it sets the standards for the application programs that run in it. All programs must be written to "talk to" the operating system. Also called an executive or supervisor, the operating system performs the following functions.

Job Management

In small computers, the operating system responds to commands from the user and loads the requested application program into memory for execution. In large computers, the operating system carries out its job control instructions (JCL), which can describe the mix of programs that must be run for an entire shift.

Task Management

In single tasking computers, the operating system has virtually no task management to do, but in multitasking computers, it is responsible for the concurrent operation of one or more programs (jobs). Advanced operating systems have the ability to prioritize programs so that one job gets done before the other.

In order to provide users at terminals with the fastest response time, batch programs can be put on lowest priority and interactive programs can be given highest priority. Advanced operating systems have more fine-tuning capabilities so that a specific job can be speeded up or slowed down by commands from the computer operator.

Multitasking is accomplished by designing the computer to allow instructions to be executed during the same time data is coming into or going out of the computer. In the seconds it takes one user to type in data, millions of instructions can be executed for dozens, or even hundreds, of other users. In the milliseconds it takes for data to come in from or go out to the disk, thousands of instructions can be performed for some other task.

Data Management

One of the major functions of an operating system is to keep track of

D
R
A
F
T

data on the disk; hence the term DOS, or disk operating system. The application program does not know where the data is actually stored or how to get it. That knowledge is contained in the operating system's access method, or device driver, routines. When a program is ready to accept data, it signals the operating system with a coded message. The operating system finds the data and delivers it to the program. Conversely, when the program is ready to output, the operating system transfers the data from the program onto the available space on disk.

Device Management

In theory, the operating system is supposed to manage all devices, not just disk drives. It is supposed to handle the input and output to the display screen as well as the printer. By keeping the details of the peripheral device within the operating system, a device can be replaced with a newer model, and only the routine in the operating system that deals with that device needs to be replaced.

In the PC world, device management is left up to the vendor of the application, because vendors write their programs to directly access the screen and the printer. They bypass the operating system, because DOS either doesn't support the device or adds too much performance overhead. As a result, software vendors have become responsible for providing drivers (routines) for all the popular display and printer standards, adding an enormous burden to their development efforts.

Security

Multiuser operating systems maintain a list of authorized users and provide password protection to unauthorized users who may try to gain access to the system. Large operating systems also maintain activity logs and accounting of the user's time for billing purposes. They also provide backup and recovery routines to start over again in the event of a system failure.

History

The earliest operating systems were developed in the late 1950s to manage tape and disk storage, but programmers often felt more comfortable writing and using their own I/O routines. In the mid 1960s, operating systems became essential to manage the complexity of timesharing and multitasking. Today, all multi-purpose computers from micro to mainframe use an operating system. Special-purpose devices, such as appliances, games and toys, do not. They usually employ a single program that performs all the required input, output and processing tasks.

In the past, when a vendor introduced a new operating system, users had little understanding of the time and effort required by

DRAFT

computer professionals to convert to it. This is no longer a behind-the-glass- enclosed-datacenter phenomenon, but squarely in the hands of the users. Switching from DOS to OS/2 or UNIX is not a trivial issue. We will no doubt soon find desktop computers running operating systems, such as IBM's VM, which are capable of running multiple operating systems, each controlling their own applications. History has an uncanny way of repeating itself.

Perhaps the Japanese have the right idea with their TRON operating system, which is ultimately intended to be a common interface across all applications from a microwave oven to the largest supercomputer!

Common Operating Systems

PCs use DOS, OS/2, SCO XENIX and AIX. Apple II's use DOS (from Apple) and ProDOS. Macintoshes use the System along with Finder and Multifinder, as well as A/UX. Digital uses VMS and Ultrix. IBM mainframes use MVS, VM or DOS/VSE.

DRAFT

- path -** (1) In communications, the route between any two nodes. Same as line, channel, link or circuit. (2) In database management, the route from one set of data to another, for example, from customers to orders. (3) In programming, a set route taken by the program to process a set of data. (4) In DOS and OS/2, the route to a disk directory. For example, if a file named MYLIFE is located in subdirectory STORIES within directory JOE, the path to retrieve the file is: \JOE\STORIES\MYLIFE.
- RISC -** (Reduced Instruction Set Computer) A computer architecture that performs a limited number of instructions. The concept is that most programs generally use only a few instructions, and if those basic instructions are made to execute faster, performance is increased.
- RISC eliminates a layer of overhead called microcode, which is commonly used to make it easier to add new and complex instructions to a computer. RISC computers have a small number of instructions built into the lowest level circuits that work at maximum speed.
- root directory -** In hierarchical file systems, the starting point in the hierarchy. When the system is first started, the root directory is the current directory. Access to directories in the hierarchy requires naming the directories that are in its path. In DOS and OS/2, going down the hierarchy requires naming the directories in the path from the current directory to the destination directory, but going back up or sideways requires naming the entire path starting from the root directory.
- shell -** An outer layer of a program that provides the user interface, or way

of commanding the computer. Shells are typically add-on programs created for command-driven operating systems, such as UNIX and DOS. The shell provides a menu-driven or graphical icon-oriented interface to the system in order to make it easier to use.

Sun -

(Sun Microsystems, Inc.) A manufacturer of network-based, high-performance workstations founded in 1982. The Sun-3, Sun-4 and Sun386i product lines include stand-alone and networked systems, diskless workstations and file servers. The Sun-4 family is based on the SPARC microprocessor, and the Sun 386i is based on the Intel 386 CPU.

Sun supports an open systems model of computing throughout its product line which allows it to interact in networks of computer systems from other vendors. Its Open Network Computing software is supported by over 100 vendors, including Apple, Digital and HP. Sun's Network File System (NFS) software, which allows data sharing across the network, has become an industry standard.

WYSIWYG -

(What You See Is What You Get) Pronounced "wizzy-wig." Refers to displaying text and graphics on screen the same way it will be printed. Although a 24-point font will show up on screen in a proper ratio to a 10-point font, if the screen and printer fonts are not matched, there will be either slight or obvious differences between the screen characters and the printed characters. In addition, a desktop laser printer, at 300 dots per inch, has a much higher resolution than almost any display screen; therefore, the screen and printout cannot be 100% identical.

X/Open -

A consortium of international computer vendors, founded in 1984, to resolve standards issues. Incorporated in 1987 and headquartered in London, current members include AT&T, Bull, DEC, Ericsson, HP, International Computers, Nixdorf, Olivetti, Philips, Siemens and Unisys. X/Open's North American offices are in San Francisco.

X/Open's purpose is to integrate evolving de facto and international standards in order to achieve a user-driven and open environment, or Common Application Environment. X/Open's XPG defines the specification, and VSX defines the testing and verification procedure.

DRAFT