GeoSpatial Data Development for TMDL in the Trinity Basin

By Jóna Finndís Jónsdóttir, Kimberley M. Davis, and David R. Maidment
Center for Research in Water Resources
University of Texas at Austin

Prepared for the Total Maximum Daily Load Team
Texas Natural Resource Conservation Commission
Austin, Texas

November 1999

Table of Contents

This report summarizes the results of a geospatial data development project carried out for the Total Maximum Daily Load Team of the Texas Natural Resources Conservation Commission by the Center for Research in Water Resources of the University of Texas at Austin. The report has three chapters and two appendices. Each of the chapters relates to one of the tasks in the original research contract, and follows the format of stating the task as it appeared in the original contract, and then the procedure by which the task was accomplished. Appendices A summarizes the procedures for data acquisition and processing using ArcInfo and ArcView, and Appendix B presents the ArcView scripts created for this project. The scope of application of this project is the Trinity river basin, Texas.

1. Task (c) Reconcile point location data layers

1.1. Task description

The TNRCC shall provide the performing party with a set of Water Quality point location data layers (e.g. municipal and industrial wastewater discharge points, Water Quality Segment endpoints, USGS flow gages, TNRCC water quality monitoring stations) pertinent for use in TMDL modeling. The performing party shall establish a process for the reconciliation of these data layers with stream network hydrography at 1:100,000 scale (i.e. EPA River Reach File 3, National Hydrographic Dataset). Products of this process shall be event-theme data layers with the pertinent point information attributed to hydrographic features. This process shall be performed for all Texas river and coastal basins.

1.2. Building a river network

1.2.1. Compiling and editing the original RF3 river network

A river network was built for the Trinity basin, based on EPA River Reach File 3. Procedure of preliminary preparation is provided in Appendix A. In order to use the river network of RF3 in this project, it had to be simplified and made topologically coherent, e.g. a network with graphic double-line features like braided rivers, and water bodies like bays and lakes, is not suitable for a processing of data and depiction of watershed hydrography. The process is semi-automatic, as follows: reach types R (regular reaches), S (start reaches), and T (terminal reaches) are selected from the RF3 network (for the Trinity, no reaches of type T showed up), and USGS centerline coverage, which depicts single line "transport" paths, is added to the network to replace water bodies and double lines.

Although the centerline coverage usually coincides with the simplified RF3 coverage, there are often small gaps where a centerline and a RF3 line should meet, due to imprecision and limits of resolution in the digitization process. To find these dangling

points, a script named Nodes (Appendix B), which identifies dangling nodes, can be used. Then by using the editing mode in ArcView the two lines are connected. This is a semi-automated process, in that the anomalies are detected automatically, but connected manually.

In low-relief terrain, the digitization procedure resulting in RF3 sometimes misinterprets channels and drainage swales, or creates numerous vestigial or nonfunctional links. These must be detected by appeal to the real watershed. For that reason, even after integrating in USGS centerline coverage and resolving dangling nodes, the process leaves some features like loops and unconnected lines, which have to be repaired in order to get a single line network. Those can be hard to correct based upon using the line coverage alone. Therefore Digital Raster Graphics (DRG's), digitized topographic maps of Texas, obtained from TNRIS, were added to the Arc View project. By superimposing the RF3 network on USGS 7.5-quad images, the individual RF3 links are examined manually to identify and correct network topology. In addition, unimportant and/or artifact links are detected and removed from the network. This process, though expedited by the GIS overlay and object-manipulation of the ArcView GUI, is nonetheless tedious and relies upon the ability of the operator to appraise significant hydraulic connections and upon the familiarity of the operator with the basin.

Figures 1.1 through 1.3 show the process of creating a "clean" river network.

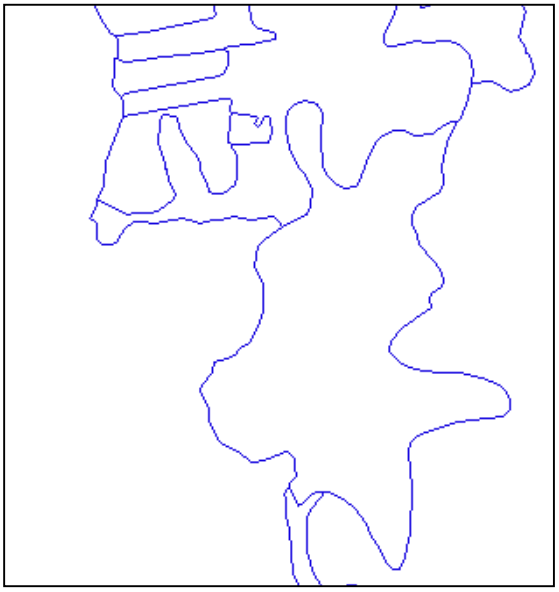


Figure 1.1 The RF3 coverage for a small area of the Trinity River Basin

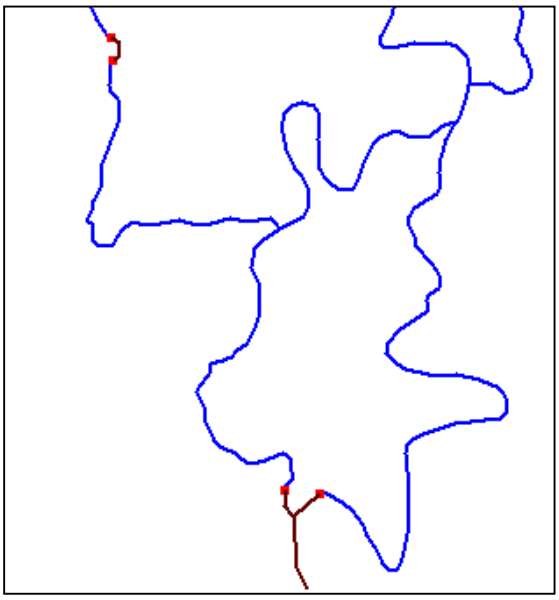


Figure 1.2 The RF3 reach types R and S, the centerline coverage and dangling nodes identified

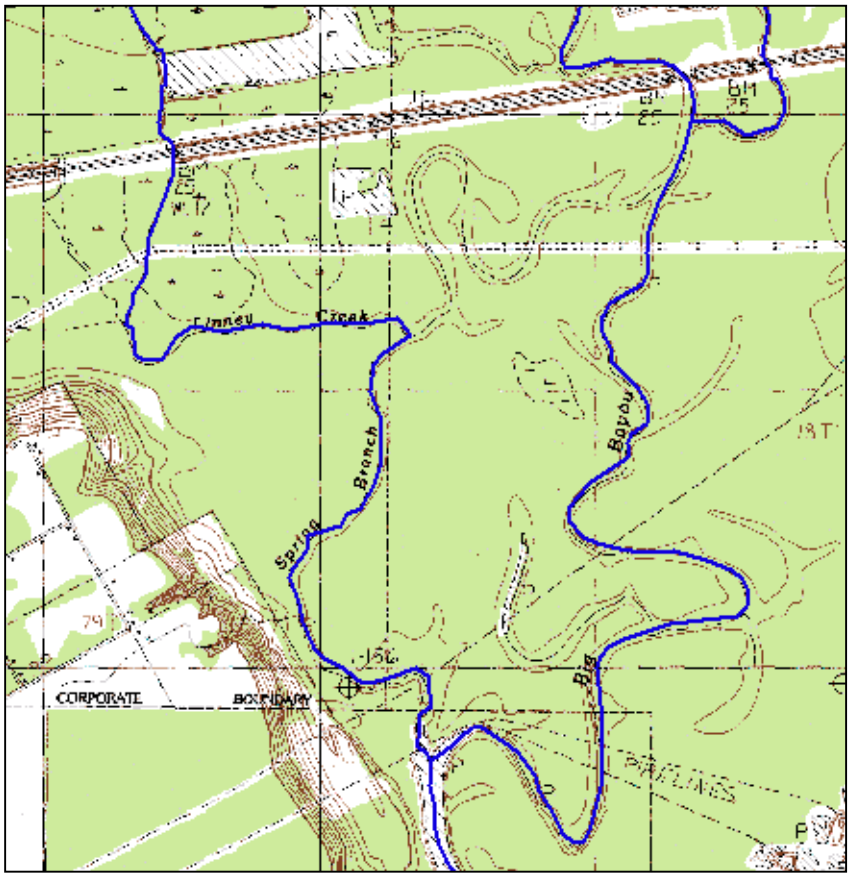


Figure 1.3 The edited "clean" version of the RF3 coverage, superimposed on a topographic map.

The process of "cleaning" a river network requires a lot of work with DRGs, therefore a helpful script "addtopo" (Appendix B) was created in order to view DRGs, to make it easier to add the topographic maps to the view, i.e. by eliminating the need to look up the name of each map. By clicking on a map grid cell, the associated map is added to the view and the bordering quadrangle map collars are trimmed so that adjacent maps can be viewed without obstruction.

The procedure is as follows. Click "Add theme". To search for the right file, it is easier to use the "Hot Link" button capability in Arc View. In order to use this option, a reference grid is needed. For this purpose, a 7.5 minute grid, "Txmesh" was built, each cell of which corresponds to one topographic map. In the quads attribute table, a field has to be added connecting the cells to the file names of the corresponding maps. The script "addtopo" responds to the click on a grid cell by searching for the corresponding file, which is then displayed over the grid as shown in Figure 1.4.

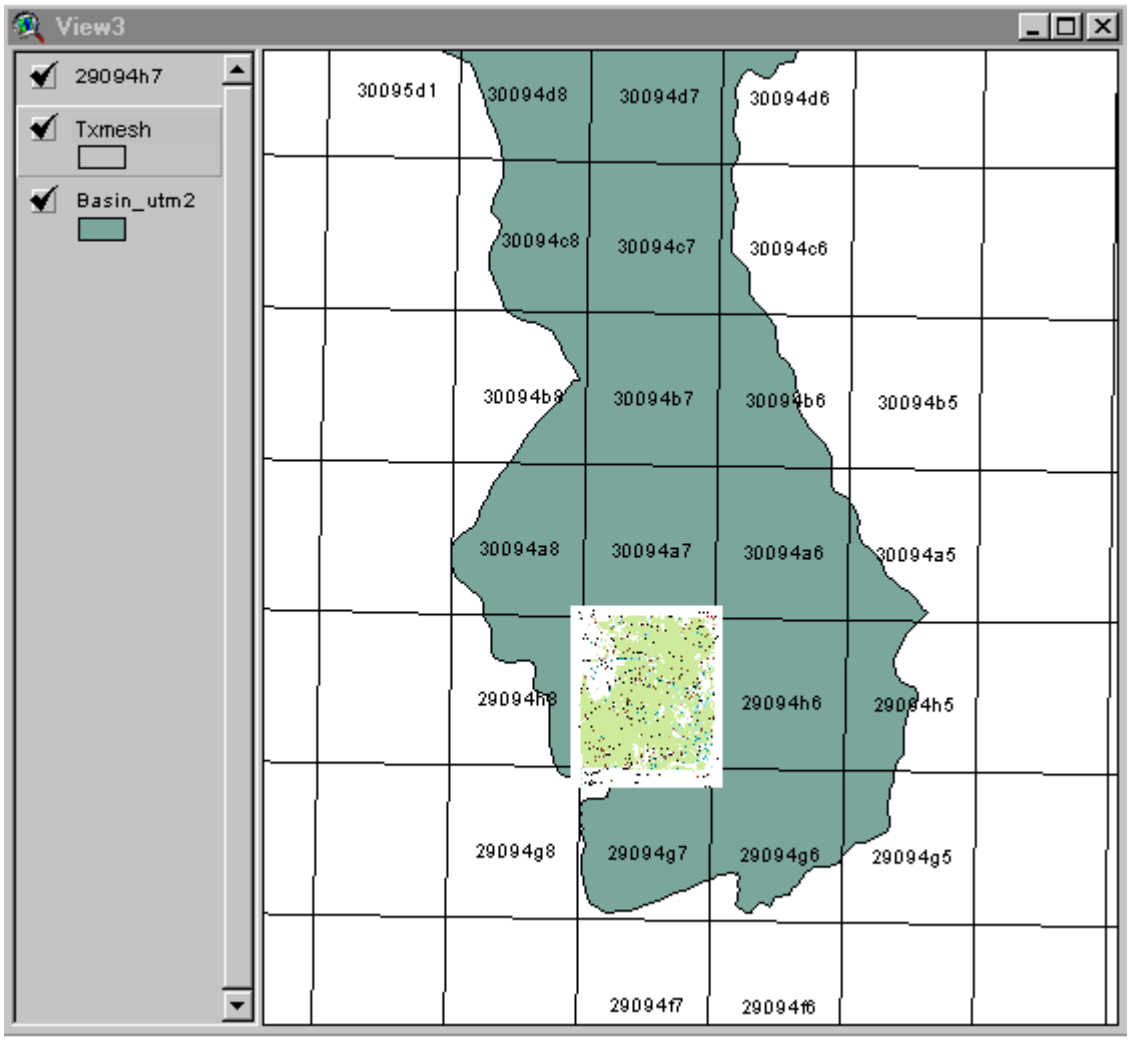


Figure 1.4 Addtopo usage

If the script does not find the map, it displays the name of the file it is looking for so that the right CD can be inserted. Then "addtopo" adds the specified map to the view and zooms in.

When starting Arcview, the "Hot Link" button is dimmed. To be able to use it, the right theme has to be active, then "theme properties" has to be clicked and the topic "Hot Link" selected as shown in Figure 1.5.

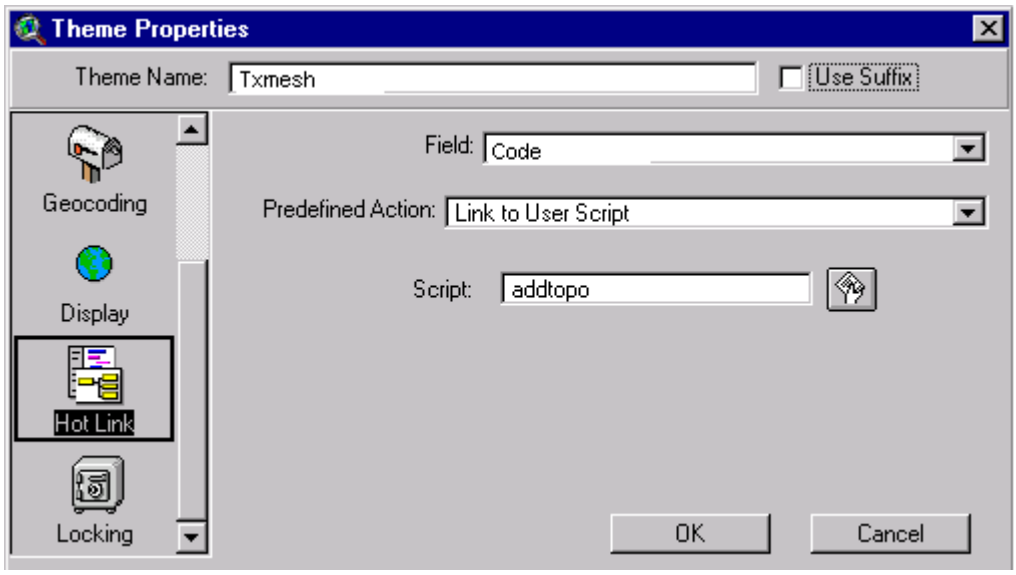


Figure 1.5 Activating the “Hot Link” button

The name of the field with the filenames is then selected in the field cell. In the "Predefined Action" cell, "Link to User Script" is selected if the user wants to use his/her own script. Finally the desired script in the script cell is picked and the OK button is clicked. Those instructions as well as the script "addtopo" and downloadable copy of "txmesh" are stored on the website:

http://www.ce.utexas.edu/stu/jonsdoj/research/hotlink.html

The river network based on the RF3 file and centerline coverage for the Trinity River Basin has now been reconciled with topographic maps (DRGs) as well as discharge points, water right locations, surface water quality monitoring stations and USGS gage stations. This means that most gaps have been closed and reaches have been added when their association with one or more point data was obvious, according to the DRG's. The result is a "cleaned" network, which is appropriate for further processing.

1.2.2. Checking connectivity

One extremely important characteristic of a network is that all the pieces of it have to be connected to each other in some way. To that end, ESRI’s ArcInfo was used to clean up the river network and trace the connectivity until the whole Trinity River was flowing out

into the Gulf of Mexico. Details of how to check the connectivity are presented in Appendix A.

A pitfall of this cleaning procedure is that the river was inadvertently coarsened too much when default cleaning tolerances were used. As a result, large sweeping bends in the river like the one in Figure 1.6 began to look like the one in Figure 1.7 and none of the other analysis steps would work until this was corrected.

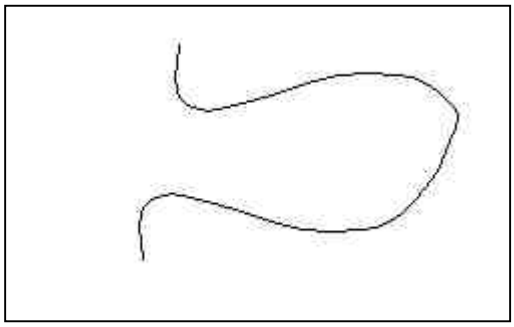


Figure 1.6 A sweeping bend in a river

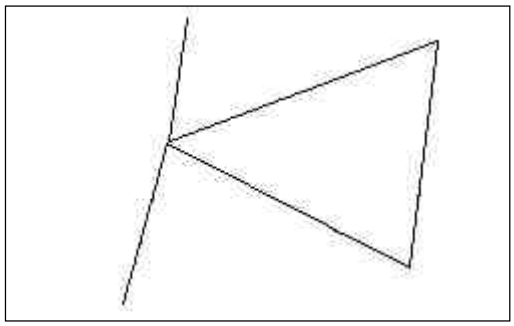


Figure 1.7 Result of cleaning: too coarse river network

The source of the error was a tolerance value that was too large in the Arc Info "Clean" command. The tolerance specified the minimum distance between vertices along a curvilinear feature. Because of the spatial sprawl of this dataset, the large default tolerance value seemed to be appropriate. The RF3 data is so detailed, though, that the tolerance based on areal extent was eradicating a lot of the fine information. The tolerance calculated by Arc Info used the default equation: 1/100,000th the extent of the data in the x or y direction, whichever is smaller. This value was approximately 130 meters for the Trinity River basin. Setting this value to 10 meters corrected the errors.

1.2.3. Eliminating closed loops

Another extremely important characteristic of a network is whether closed loops are allowed. In the case of utility networks, loops are a desirable design feature because they provide for changing flow patterns that keep pipes from clogging up with sediment. In the case of river networks, loops are a problem because the flow direction is difficult to determine, it depends on ground elevation, volumetric flow, and river depth. Almost

every river divergence is caused by something in the streambed (like a rock outcrop) that resists erosion, so the river splits to go around it. Others are caused in very flat terrain when the water (in its attempt to flow downhill) spreads out and finds two paths at the same elevation. At the 1:100,000 scale there are only a few divergences that appear, 18 were located in the entire Trinity Basin. Details for the detection of closed loop are presented in Appendix A.

These loops were manually removed from the network. This was facilitated by use of the script "AddTopo". In order to determine where to break the loops the USGS Digital Raster Graphics (DRGs) were used to see what was "on the ground." In most cases, one of the stream segments forming the loop was intermittent, so it would be broken it at its upstream end as seen in Figure 1.8.

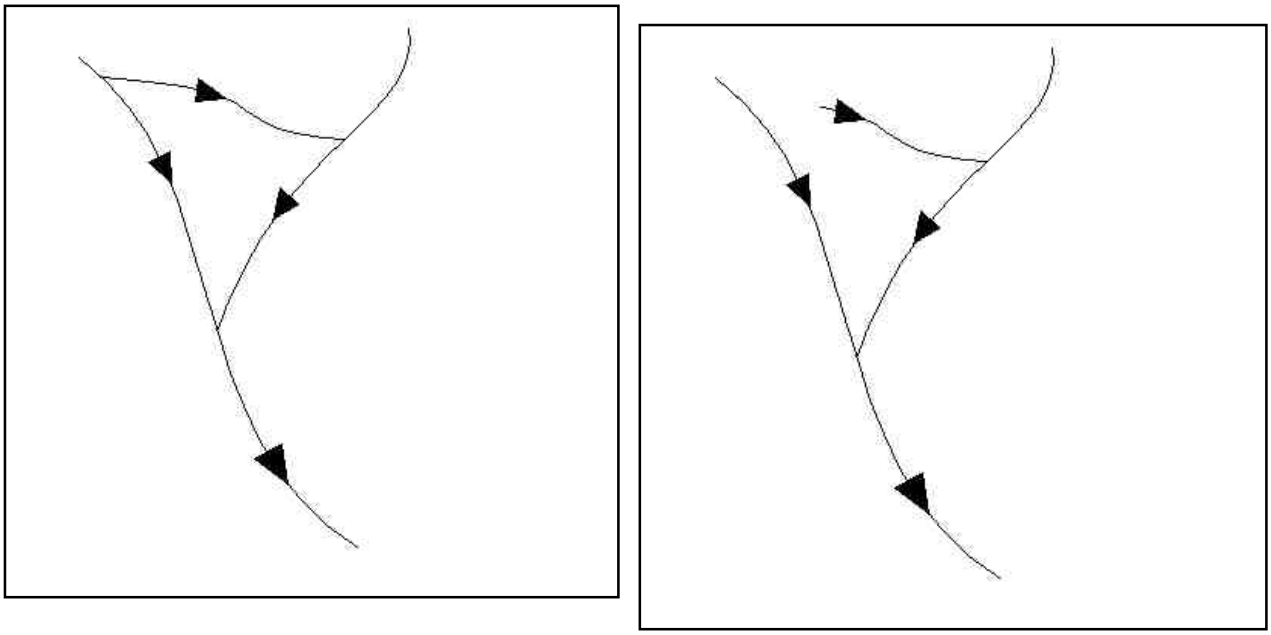


Figure 1.8 Breaking loops

1.2.4. Generating topology

A feature of networks that makes handling, searching, and modeling more efficient is called topology. Topology refers to the relationships that exist between elements of a map, such as: Intersects, Is Left Of, Is Right of, Contains, Is Contained By, etc. Networks have a special set of relationships referred to as Arc-Node Topology. Every piece of the river network is an Arc, and it has a unique identifier, usually an integer number, or the RF3 Reach ID. Every Arc has two ends, or Nodes associated with it. One is designated as the origin, or From Node, the other is the destination, or To Node. This allows each arc to have an orientation, so that things move along the network in the direction the arc is oriented. Each node also has a unique number, so that arcs which share a node are known to join each other. Arcs which share the same number for their To Node converge. An arc whose To Node is the same as another arc's From Node flows into the latter. Topology is established by Arc Info when any changes are made to a set

of arcs. However, not all data are developed in Arc/Info, and ArcView uses algebraic relationships instead of topology to relate features. Accordingly, there are some Avenue scripts that work with ESRI's ArcView Software to assign topology to line themes. Both Arc Info and the ArcView scripts were used in this project to establish Topology for the Trinity River, and the results were equivalent. The topology scripts, however, are much slower. When the topology scripts are be used, the "Labels" script is applied first, then the "Topology" script.

1.2.5. Correcting orientation

The purpose of orienting a river network is to enable queries that look downstream or upstream of a point of interest. To perform an environmental impact statement, it is important to know what is downstream of a proposed site, which segments of the river, and which water users and waste dischargers. To evaluate the causes of water quality problems at a given location, a look upstream can tell which sites and stretches of river are contributing. The computer needs the network to be oriented correctly in order to perform those types of searches.

A script was written that would detect and correct, if necessary, the orientation of river reaches based on the elevation of the nodes. Because most of Texas is quite flat, about ten percent of the reaches in the Trinity Basin (~700 of ~7000) could not be automatically classified and would need to be corrected by hand. This was a daunting task, and there was a better way using a more efficient topology-based search. This concept is still not ruled out completely, because better DEMs are becoming available, and because this approach does not require topology. There are now DEMs of 30 meter resolution available for Texas, and there are specialized data sets for cities that have as small as six inch resolution. Also, there is another network creation goal which DEM work will apply to, and that is the conversion of 2-dimensional river networks to 3-dimensional ones. The DEM can be mined to give river reaches elevation values so that they can be viewed in 3-d as they exist on the surface of the earth. Preliminary work on this step looks promising.

There is a data structure in ArcView called a data dictionary. A data dictionary takes key values (like a node number) and assigns to that key any value the programmer chooses (like a list of arcs containing that node). A script called NetFlip that created a dictionary for From Nodes and a separate one for To Nodes was written. The values stored under each key were the list of arc ID numbers that came from or went to that node. This enabled the computer to memorize, in effect, which arcs came and went from which nodes. Instead of selecting an arc and searching the list of all other arcs until something matched it, this script grabbed an arc, looked at its end, and then looked in the dictionary to find the arcs that also used that end point. This resulted in a great reduction in runtime, cutting it from several hours to approximately two minutes.

Before correcting orientation, some of the reaches point upstream. After correcting orientation, everything points downstream as seen in Figure 1.9.

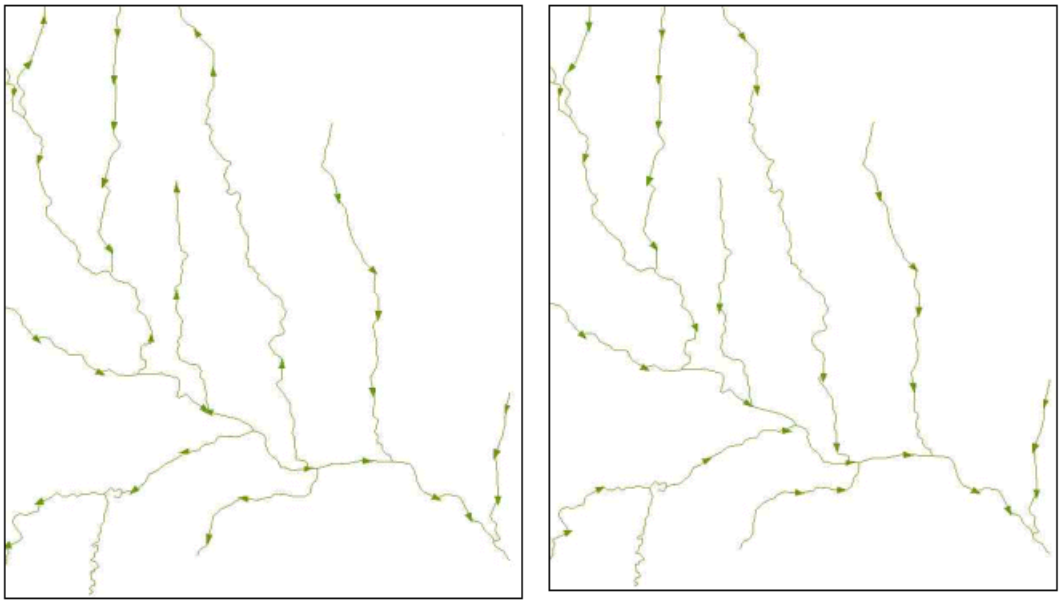


Figure 1.9 Correcting orientation of the network links

1.3. Attaching points

Once a river network has been established, the next thing needed for modeling or analysis is data about conditions along the branches of the river. These are collected at stations, such as the stream gauging stations operated by USGS or the SWQM stations operated by TNRCC, and are represented as single points on a map. The locations of these streams are often reported in the format of their geographic coordinates, or latitude and longitude. Due to small errors in the locations of the points or the streams, the points do not always map exactly on the line representing the stream. These errors must be corrected, and the points must be located exactly on the streams. Even a slight shift off the streamline will keep the computer from recognizing the point as matching to the line. A script, Snap_pnts (code in Appendix B), that would search the vicinity of a point and find the nearest stream was used. It creates a new point exactly on the stream, a "virtual" point, linked by its ID number to the original. This maintains the integrity of the original location data and enables the computer to find the new points associated with the stream network. The delivered layers va_dischpts.shp, va_segmentdp.shp, va_swqm.shp, and va_usgsgages.shp were created by applying the Snap_pnts script to the corresponding original data layers. An example of results is in Figure 1.10.

The black line represents the river. The red point is the original location of the SWQM station. After application of the snap script the purple "virtual point" was created to represent the location of the original point along the line of the river.

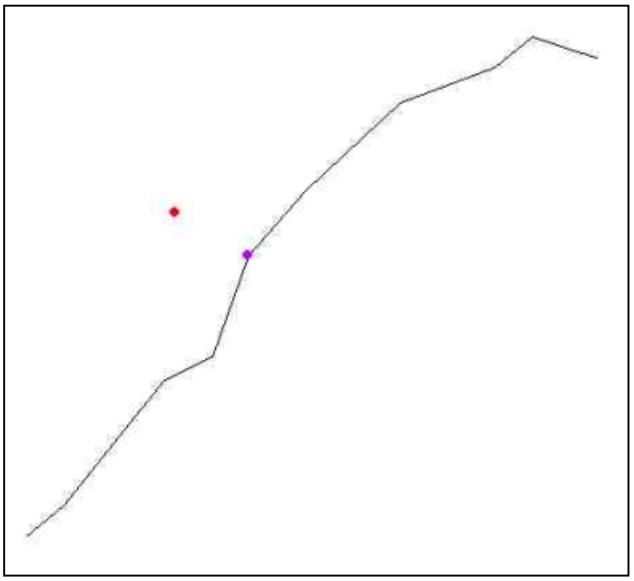


Figure 1.10 Creation of va_swqm.shp by Snapping points to the River

1.3.1. Locating downstream points

Once the points have been accurately located along the network lines, the next step is to attribute the descriptive table with the flow relationships of the points to each other and to the network as a whole. The crucial attributes are the identity of the next downstream point, the distance downstream to the outlet, and the identity of the reach on which the point is located. With this information, incremental distances between points can be determined, and flow tracing can be performed from point to point.

Ordinarily, this process is performed by hand, with operators staring at a map of the river and following downstream until the next point is reached. However, with an oriented stream network this task can be automated. The script that performs this work, called Downstream, builds on the HydroNet.Snap and NetFlip scripts, requiring that the points be located on the network lines and that the network lines have an appropriate topology. From this, the script can analyze each point in turn to determine where it is attached to the network, then add the ID number of the arc and the location in percentage of arc length. It calculates a downstream flow distance for every reach of the network that represents how far it is from the bottom end of that reach to the network outlet. Then the distance from the point to the end of the reach plus the distance to the outlet is assigned to the point as its downstream distance. Next, the script searches downstream to find the nearest point along the flow path and adds its identity to the attribute table.

1.4. Building a virtual network

A useful tool for visualizing the connectivity of points in the network is a skeleton or "virtual" network. It consists only of the points of interest and lines connecting them in flow order. Such a construction is generally useful for visualization purposes only. The

script that creates the virtual network, called VirNetBuilder, relies only on the point table being attributed with downstream point names and downstream flow distances.
An example of a simple virtual network created on the Trinity River is shown in Figure 1.12

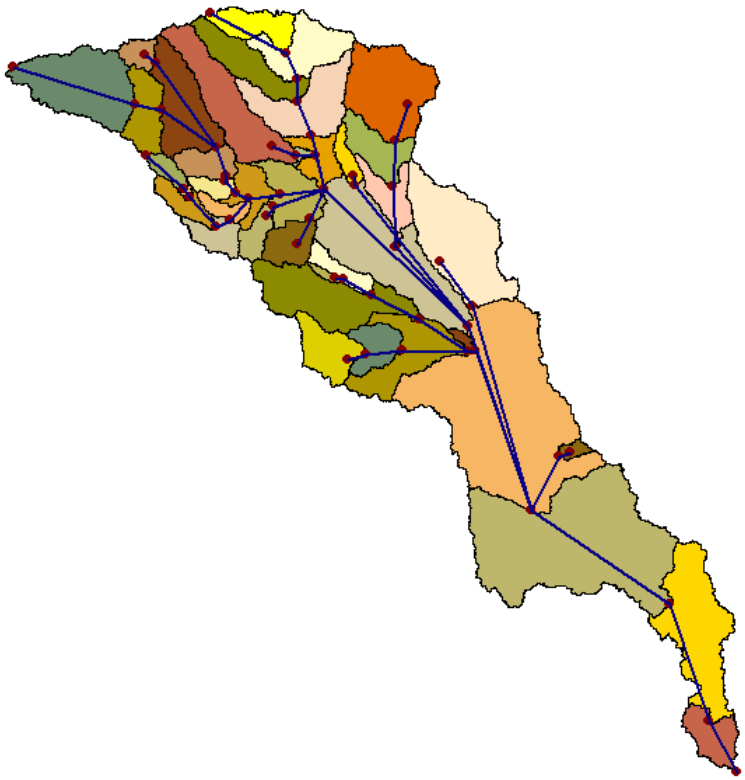


Figure 1.12 A virtual network of the TNRCC water quality segments

The blue lines are the skeleton or “virtual network” connecting the red points in downstream order.

1.5. Dynamic segmentation

Dynamic segmentation is a way of storing information about linear spatial data, without having to store the lines again. It is flexible in the way it references locations and can be used to characterize elements of existing datasets without having to recreate them and change all their attributes. What this means is that the same information can be represented in a smaller amount of space and with less work to generate.

Spatial data are the shapes drawn on the computer screen when a GIS program is in use. Lines are stored as lists of coordinates, drawn out connect-the-dots fashion, or they are stored as a series of directions and distances, like a flight path. Either way, a list of numbers directs the software to represent the line on the screen. The line is stored on a row in a table, other information on that row might be the identification number of the line, its length, and its name if it represents a street or a river. This can be limiting, because if the line is long, certain things about it might change between the beginning

and the end, such as the depth of a stream, or the number of lanes on a road. With the original data table, only one value can be entered for each property of the line from its beginning to its end. To account for changes, the line has to be broken into separate pieces. But this increases the complexity of the data stored and takes up more memory. With dynamic segmentation, it is possible to store the changing properties of the network in an "event table" without breaking the lines and without making a new copy of them. An example of how a GIS stores and displays data is in Figure 1.13.

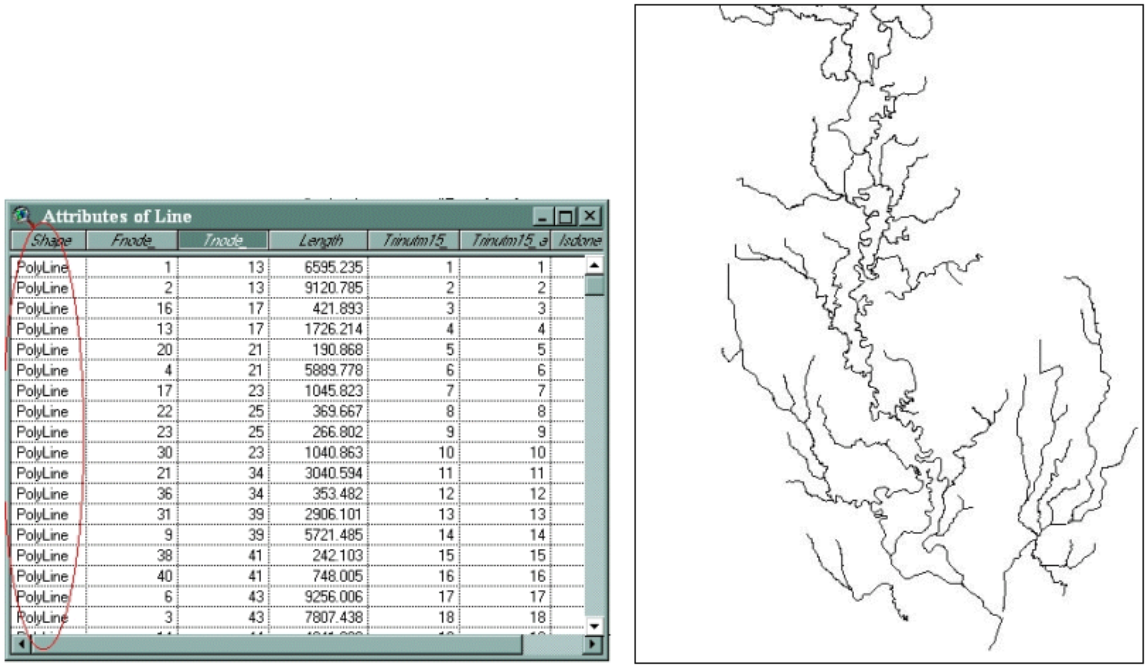


Figure 1.13 An example of how GIS stores and displays data

The entries in the table are the tabular view for the lines to the right. Each table row is one line on the picture. In the circle above, the word "PolyLine" is not stored in the computer's memory, but rather the list of points which when connected draw out the line on the screen.

Each row in the table represents one of the shapes on the screen. From beginning to end, that line is assigned the properties stored in its row in the table. Once this data is stored, however, an event table can be built on top of it by dynamic segmentation, and data can be stored as shown in Figure 1.14.

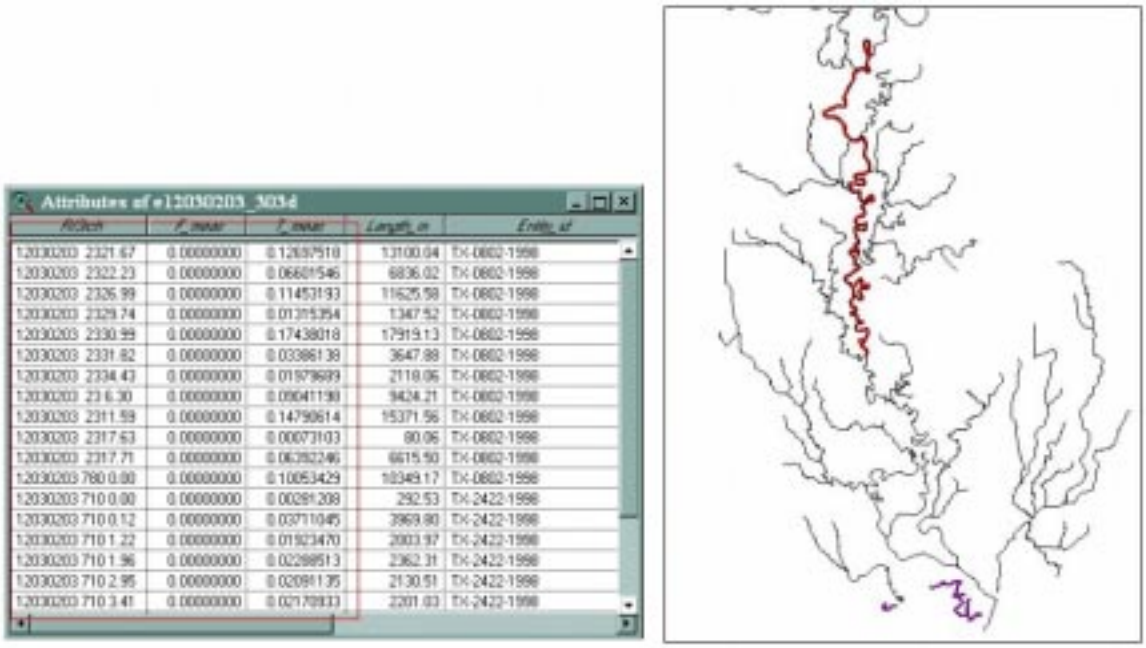


Figure 1.14 Dynamically segmented lines

The red and purple lines in Figure 1.14 are represented by the entries in the event table shown here. The black lines are the same ones from Figure 1.13. The event table knows which table it is built on, so the shapes from that table do not have to be stored in the event table. Instead, what the event table stores are references to the original shapes, namely the identity of the shape (in this case, the RF3 RCHID), the starting point, and the ending point. This is similar to indicating that a bus runs down Interstate 35 from Exit 219 to Exit 230.

The Reach Indexing Tool (RIT) developed by the Research Triangle Institute was used to index part of the Trinity River (reference located in Appendix A). That is, certain parts of the river network were selected and stored as dynamic segments of the river according to their TMDL ID numbers. This highlights the parts of the river that are of concern to the EPA and TNRCC for TMDL development, and ensures that future work will be consistent with any changes made to the original RF3 network.

2. Task (e) Digital delineation of watershed drainage areas

2.1. Task description

The performing party shall establish a reliable, reproducible process for digital delineation of drainage areas, based on 30-meter (1:24,000 scale) Digital Elevation Model (DEM) data and a 1:100,000 scale hydrograph network. This process shall be performed for the drainage areas of the TNRCC Trinity River Basin Water Quality Segments.

2.2. Delineating the watersheds

A preliminary watershed delineation was made with 90 m DEMs and used as a guide for the finer resolution delineation. The 30 m DEMs were compiled for the whole Trinity Basin, merged and clipped to the basin boundaries (HUC-boundaries buffered with 10 km). A river network for the Trinity Basin along with adjacent rivers (rivers that fell onto the 10 km buffer) was burned onto the 30 m DEMs, processed in Arc Info and resulted in watersheds for the 41 TNRCC Trinity River Basin Water Quality Segments.

The river network used for this task was built for the Trinity basin, based on EPA River Reach File 3. The RF3 river network was "cleaned"; loops representing braided rivers and lakes were exchanged for a single line network and connectivity assured. Description of the network as well as the procedure of building it is in task (c).

2.2.1. Preliminary watershed delineation with 90m DEM

90m DEMs, sometimes called 1-Degree DEMs are available for most of USA in one-degree latitude by one-degree longitude maps. They are based on cartographic sources, 1:24,000 scale through 1:250,000 scale maps, and photographic sources. The DEMs elevations are referenced horizontally on the geographic coordinate system of the WGS 72 or WGS 84, while the elevations are in meters relative to the National Geodetic Vertical Datum of 1929. Spacing between points is 3-arc seconds or approximately 90m (changes with latitude).

The 90m DEMs were downloaded from the USGS EROS website, they were merged, clipped and projected to TSMS Albers projection. This elevation model was used with the RF3 based river network for delineation of the watersheds of the TNRCC management segments, using CRWR-PrePro03 in Arc View, a layout of the resulting watersheds is shown in Figure 2.1. Details of this preliminary watershed delineation with 90m DEMs are listed in Appendix A.

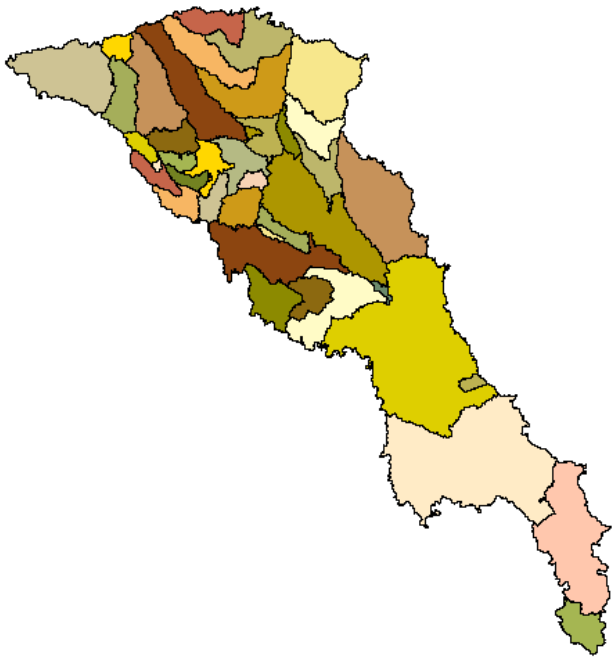


Figure 2.1 Watersheds delineated from 90m DEMs

2.2.2. Watershed delineation with 30m DEM

The 90m DEM delineation gives a good coarse definition of the watersheds of the TNRCC water quality segment drainage areas, however since 30m DEMs became available a finer delineation could be accomplished. A detailed description of how the 30m DEM delineation was done can be seen in Appendix A, while a description of the main steps follows here.

30m DEMs, or 7.5-Minute DEMs are available for all Texas through the National Elevation Database (NED) program. They are derived from digitized map contours and a scanning of NAPP or NHAP program photographs. Each map is 7.5-minute quadrangle with 30m spacing between points; therefore the number of points in a profile varies with latitude. They are generated by using a UTM Cartesian coordinate system as a base for the profiles, while elevations are either in meters or feet referenced to the National Geodetic Vertical Datum of 1929.

The 30 m DEMs became available in May 1999. The data is provided on CDs as Arc Info grids, where each grid spans a one-degree by one-degree area. The grids were merged and clipped to the basin boundaries in a similar way as the 90 m grid. This grid is however of a lot finer resolution, resulting in roughly 75 million cells for the Trinity River Basin, which made all processing slower and required a lot of disk space. The product is shown in Figure 2.2, for greater detail a closer look is taken at one TNRCC segment, the Lower West Fork Trinity River, segment 841.

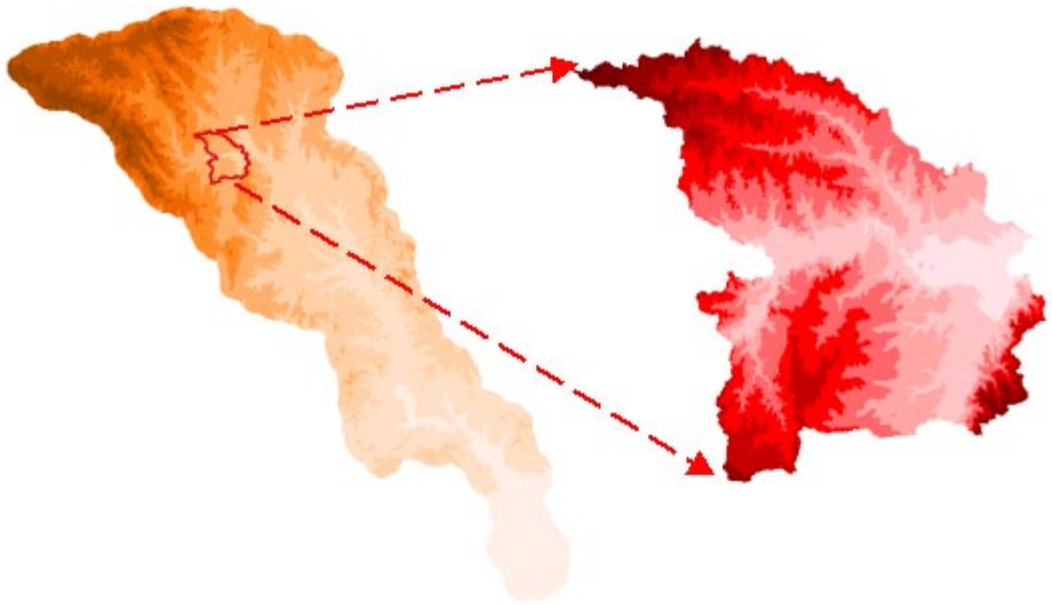


Figure 2.2 30m DEM's for the Trinity River basin

The original grid is a floating point grid with height values in meters. The size of a floating point grid is however a lot greater than an integer grid. Therefore in order to preserve accuracy but decrease the file size, the grid was multiplied by one hundred, i.e. height values were changed to centimeters and converted to an integer grid.

Even though the size of the files were reduced in this way, their size caused difficulties in processing. Arc View cannot handle such large files and CRWR-PrePro could therefore not be used when delineating watersheds from these data. The next choice was to use the built-in functions in Arc Info for watershed delineation using DEMs. The file size proved however to be larger than Arc Info could handle in the delineation process, therefore, the basin was broken down in three parts. When deciding on how to break it down in a convenient way the watersheds from the first delineation (from 90 m DEMs) were used as a guide as shown in Figure 2.3.

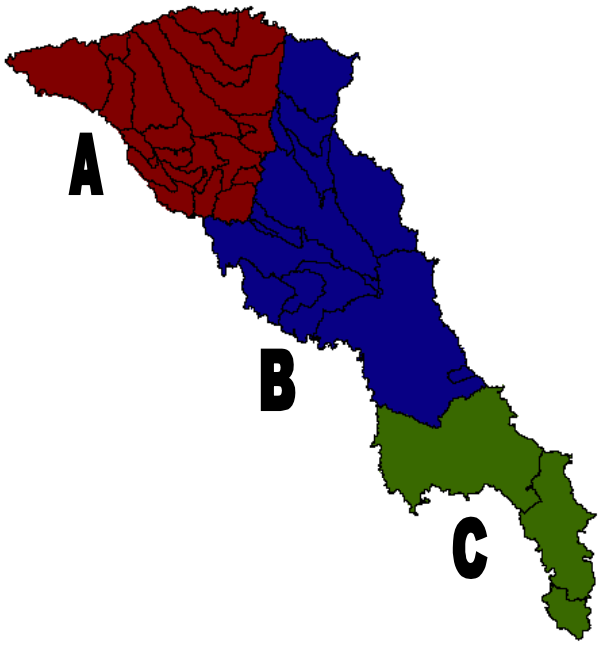


Figure 2.3 Splitting of the Trinity basin for watershed delineation using 30m DEMs

Now each part of the basin was processed separately where the first task was to burn the river network, along with upper-watershed rivers from adjacent basins, into the DEM. A shapefile containing the simplified rivers of the Trinity basin (and the adjacent basin upper-watershed rivers that lie within the 10 km buffer around the Trinity) was converted to a grid, laid on top of the DEM grid and the height value of all cells in the DEM, except for those that belong to a river, was increased by one kilometer. This is done in order to ensure the consistency of delineated rivers with the detailed river network produced earlier; the process of burning actually creates a deep virtual canyon were the river lies and ensures that all the "water" that comes into the river will stay within that canyon and therefore in the real river. This is shown in Figure 2.4.

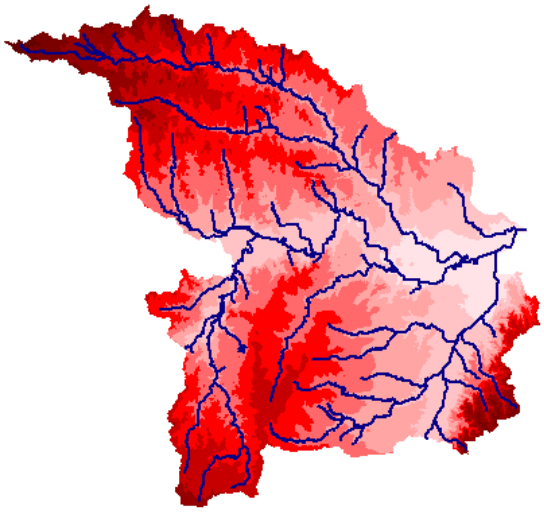


Figure 2.4 The river network burned in the 30 m DEM, TNRCC water quality segment 841

The water in each of the cells in the basin will run off to one and only one cell. The direction in which it will go depends on the steepest descent since water always flows down the steepest hill. Figure 2.5 shows that each cell in the grid is surrounded by eight neighboring cells and a value is given to cell depending on in which direction the steepest descent is, i.e. in what direction the arrow points. The cells in the basin were given values based on the height values in the burned 30m DEM and a "flow direction" grid was produced.

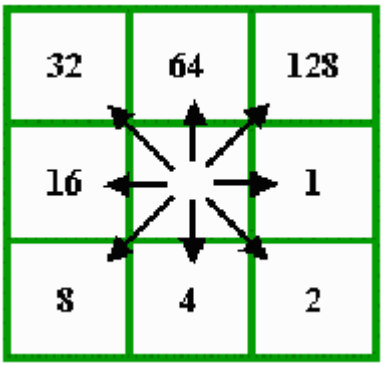


Figure 2.5 Eight direction pour point model

The flow direction grid, shown in Figure 2.6, along with a grid containing the river cells corresponding to the TNRCC segments was then processed further. The segment grid had values assigned according to the number of segment it belonged to, Arc Info was therefore able to find all the cells draining to a certain segment number and assign the segment a watershed composed of all the corresponding cells.


Figure 2.6 The flow direction grid for segment 841 with the corresponding segment grid on top

When this had been done for all three parts of the Trinity Basin, the resulting watersheds were merged again. When the three sub-basin watersheds were merged, some single cells showed up with the value 0 instead of a segment number and they were separate polygons. This had to be corrected; therefore the "gridcode" field of the attribute table of

the watershed file was edited with the number of a neighboring segment instead of the value 0. Then in Arc Info a function “dissolve” was used to dissolve the zero-value watershed cells with the neighboring watershed. The final product was the 41 TRNCC water quality segment watersheds layer, shown in Figure 2.7.

Figure 2.7 The TNRCC water quality segment drainage areas

3. Task (f) Integrated geospatial database compilation

3.1. Task description

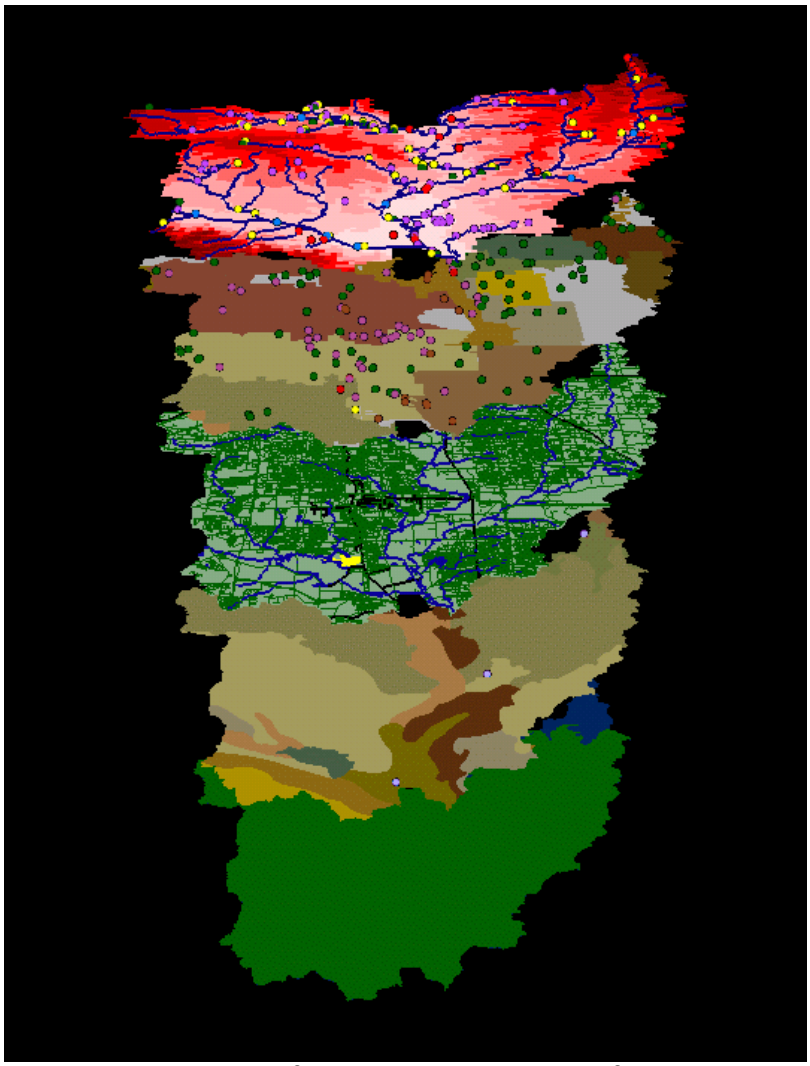
The performing party shall compile an integrated geospatial database for the Trinity River Basin, segregated by TNRCC Water Quality Segment drainage area, with data layers provided in a common map projection as specified by the TNRCC.

3.2. The geospatial database

The geospatial database will be used as a water quality management tool along with water quality models. Approximately 50 data layers relevant to water quality management were therefore compiled, some of them were downloaded from the TNRCC ftp site, some were built at CRWR and others were downloaded from various sources. All the data layers were projected in the same projection, Albers Equal Area projection with Texas State Mapping System parameters which has the following description:

Texas State Mapping System	
Datum	**NAD83**
Spheroid	**GRS80**
Projection	**Albers Equal Area**
Map Units	**meters**
Central Meridian	**100°W (-100.0000)**
Reference Latitude	**31° 10´ N (31.166667)**
First Standard Parallel	**27° 25´ N (27.416667)**
Second Standard Parallel	**34° 55´ N (34.916667)**
False Easting	**1000000**
False Northing	**1000000**

In Figure 3.1 is a sample of the database for the TNRCC segment 841, i.e. Lower West Fork Trinity River.



The layers in the database can be divided in five categories:

Hydrologic and surface water management layers

Geopolitical information and regulatory data layers

Census Tiger files

Environmental background data

Groundwater aquifers

Figure 3.1 A sample of the geospatial database for the TNRCC segment 841

3.2.1. The hydrologic and surface water management layers

- 30-meter (1:24,000) digital elevation models
- 30-meter flow direction grids
- 30-meter flow accumulation grids
- Surface water quality segments
- River network, based on Rf3
- Point locations of wastewater dischargers

- Water quality monitoring stations
- USGS flow gage locations
- Surface water rights diversion points
- National sediment inventory stations
- Locations of dams/hydraulic structure
- Hydrologic Unit Code boundaries
- Watershed data management stations / areas of coverage

A few of those layers for the TNRCC segment 841 are in Figure 3.2.

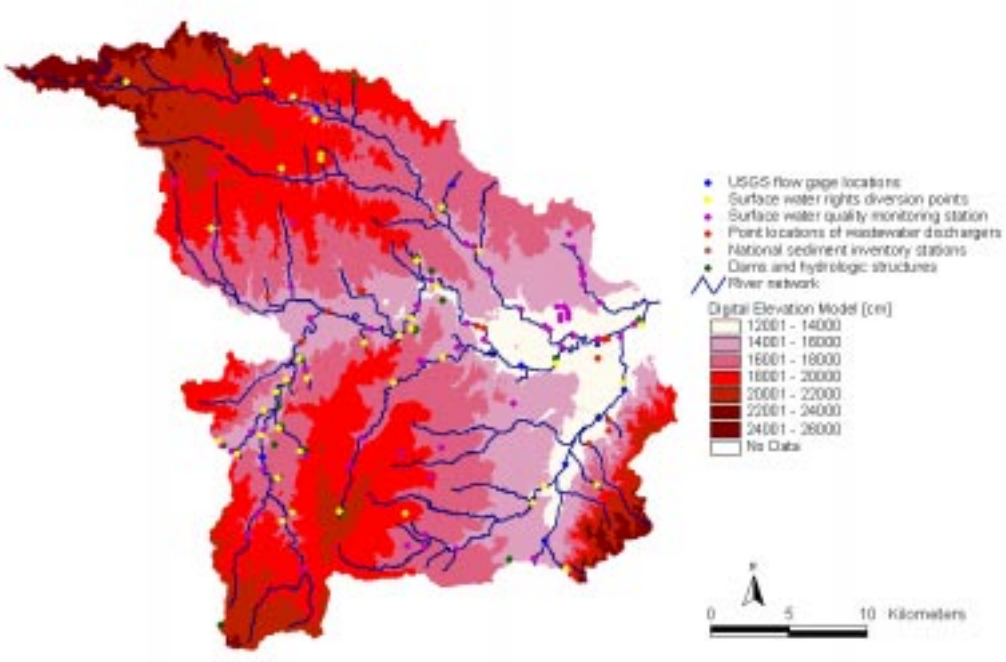


Figure 3.2 Some hydrologic and surface water management layers for segment 841

3.2.2. The geopolitical information and regulatory data layers

- Counties
- Municipal areas
- Superfund sites
- Councils of government
- Eco-Regions
- Federal congressional districts
- TX state house of representatives districts (75th legislature)
- TX state house of representatives districts (76th legislature)
- TX state senate districts
- TNRCC service regions

- TNRCC class B land application sites
- Solid waste landfill locations
- Permitted industrial and hazardous waste sites
- Toxic release inventory sites
- Public water supply sites

A sample of those datalayers is in Figures 3.3 and 3.4

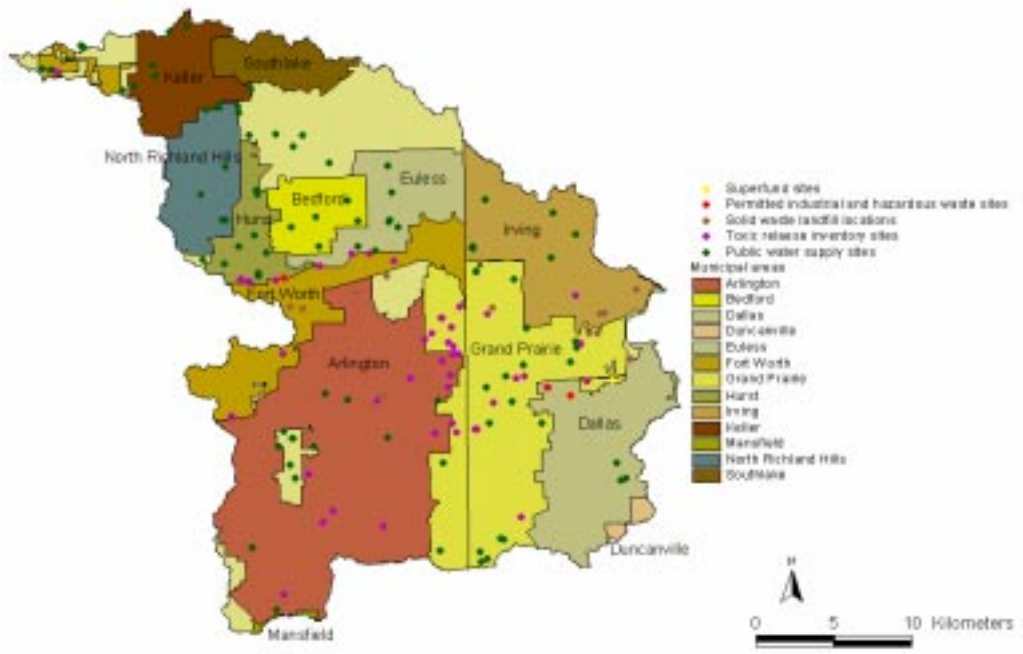


Figure 3.3 A few geopolitical information and regulatory data layers for segment 841

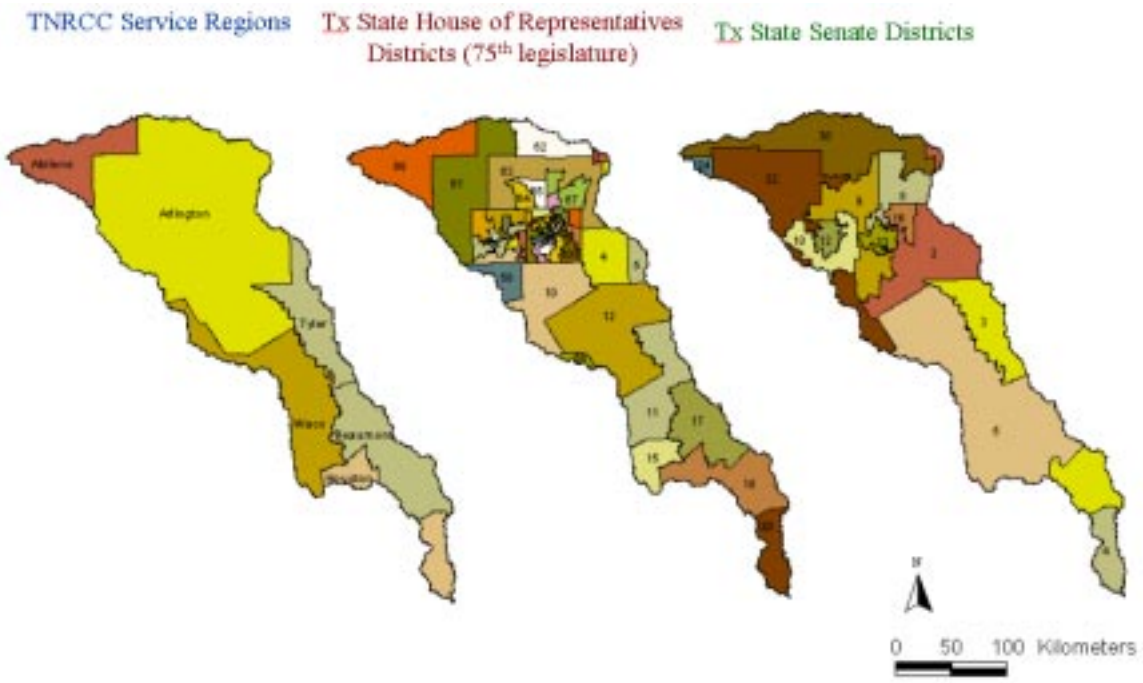


Figure 3.4 A few more geopolitical information and regulatory data layers

3.2.3. The Census Tiger files

- Airport landing strips
- Point landmarks
- Area landmarks
- Non-visible boundaries
- Geographic names info system
- Roads
- Railroads
- Streams & shorelines
- Transmission & power lines

In Figure 3.5 are some of the layers for TNRCC segment 841.

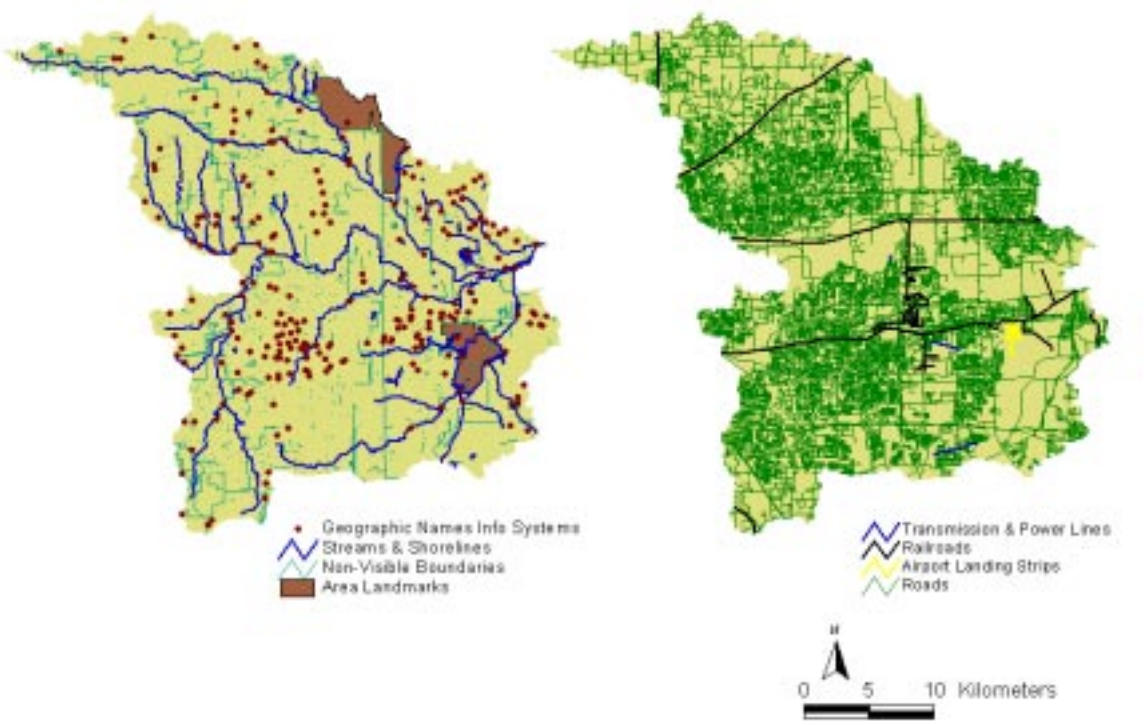


Figure 3.5 The Census Tiger files for TNRCC water quality segment 841

3.2.4. The environmental background data

- STATSGO soils coverage
- SSURGO soils coverage
- Vegetation
- Coastal wetlands habitat land use
- Air quality monitoring stations
- NWS weather stations / areas of coverage
- NCDC precipitation data

A sample of those layers are in Figure 3.6.

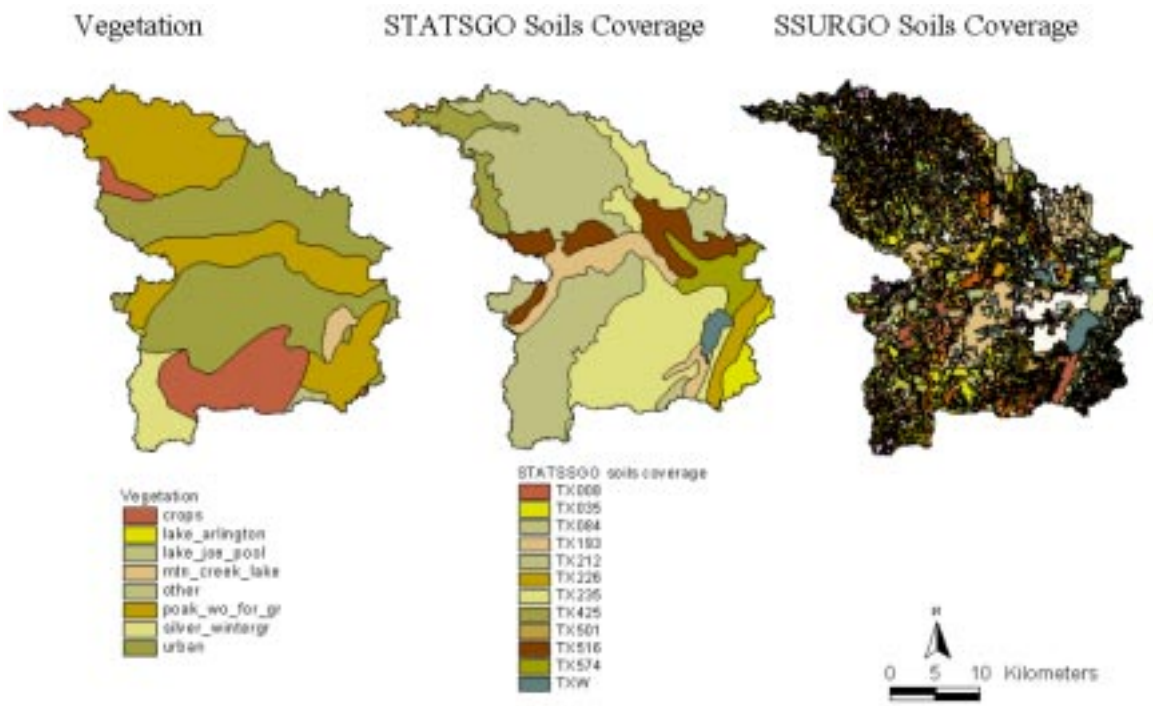


Figure 3.6 Environmental background data for TNRCC segment 841

3.2.5. Groundwater aquifers

- Gulf Coast Aquifer
- Trinity Aquifer
- Carrizo-Wilcox Aquifer
- Nacotoch Aquifer
- Queen City Aquifer
- Sparta Aquifer
- Woodbine Aquifer

The area of those aquifers within the Trinity Basin is in Figure 3.7.

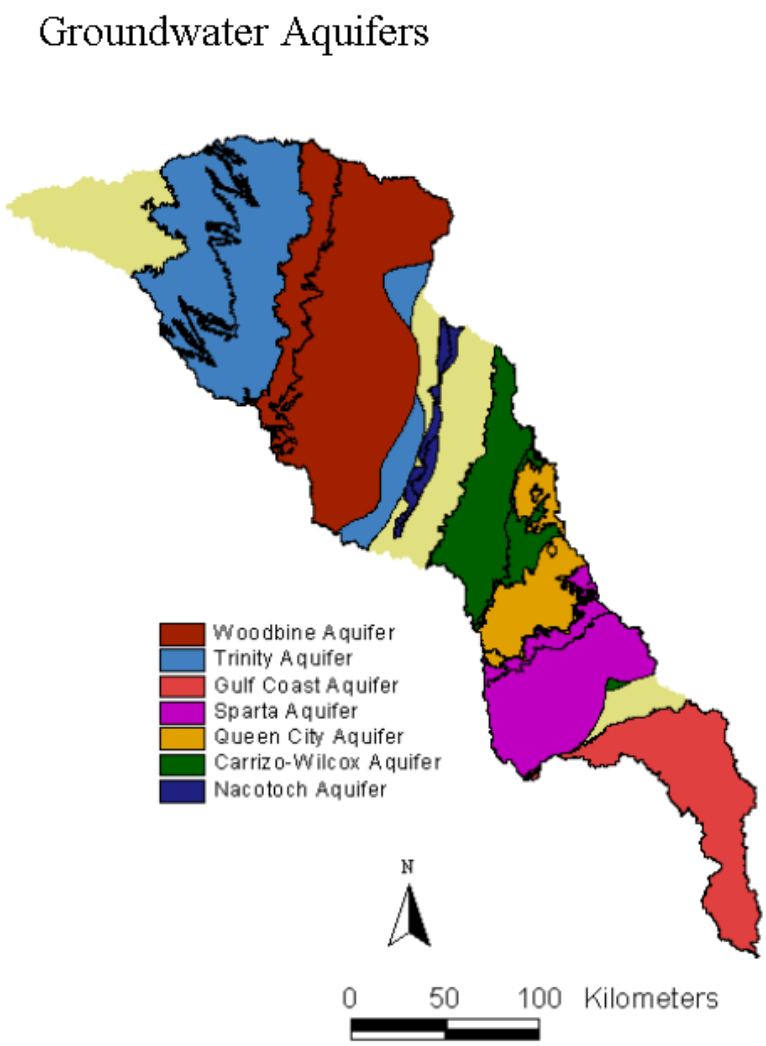


Figure 3.7 Groundwater aquifers

All these layers were assembled and projected using the Texas State Mapping System - Albers (TSMS-Albers) map projection. When the TNRCC water quality segment watershed outlines had been delineated and approved, the watershed polygon layer was used to clip out of the data layers the data that correspond to the each segment. All the layers for each segment were assembled together in one folder carrying the number of the segment and sent to the TNRCC on CDs.

The script used for clipping shapefiles is called Clip.ave and is in Appendix B. Grids were however clipped in Arc Info, details of the clipping procedure are in Appendix A.

APPENDIX A
Procedure details

Preliminary preparation of RF3 coverage:

Download rf3 shapefiles from EPA BASINS web site
Identify 8 digit Hydrologic Unit Code in the basin and download:
http://www.epa.gov/OST/BASINS/STATES/TX/
Double click on each file to unzip with WinZip

Merge individual shapefiles with mergethemes script in Arc View
Add Theme, all shapefiles
In project window, select scripts, new, load script
Select **mergethemes.ave (Appendix B)**
Compile and run, make output shapefile **rf3**

Convert to a coverage
Arc : shapearc **rf3 rf3**
Arc : project cover **rf3 rf4 geo2alb.txt**

Delete temporary grid
Arc : kill rf3 all

Rename final cover
Arc : rename **rf4 rf3**

Steps for cleaning the river network:

(if the data are already in Arc Info Coverage format, skip to step 2)
1. *Convert shapefiles to coverages:*
Arc: shapearc [path\filename.shp][see note] [path\coveragename] line
note: the pathname is not necessary unless the shapefile or coverage are in different directories from the one you are working in

2. *Clean the coverage:*
Arc: clean [**coveragename**] [**clean_coveragename**] # 10[see note] line
note: 10 is the suggested number to avoid the coarsening problem discussed below. For river networks with a small spatial extent, a smaller number may be required. It is not recommended to raise this number if the accuracy of the 1:100,000 hydrology is to be preserved.

3. *Check the connectivity:*
Arc: arcplot
Arcplot: display 9999
(move and resize the display window as needed)
Arcplot: mape [**clean_coveragename**]
Arcplot: arcs [**clean_coveragename**]
Arcplot: trace direction **upstream downstream** *
(click on the outlet point of the river network then wait for it to display on the screen)

(make the text window active by clicking on it)
Arcplot: 9 (press enter)

4. *View the results*:

Arcplot: linesymbol 2
Arcplot: readselect **upstream**
Arcplot: readselect **downstream** keep
Arcplot: arcs [**clean_coveragename**]
(at this point, all the arcs that are connected to the outlet should be showing up in a different color.)
(If everything is connected, this step is done. If not, the set of connected arcs can be displayed in ArcEdit and the gaps closed.)

Steps for detecting closed loops:

1. *Convert shapefiles to coverages*
Arc: shapearc [**path\filename.shp**] [**path\coveragename**] line

2. *Build polygons:*
Arc: build [**coveragename**] poly

3. *Display polygons*:
Arc: arcedit
Arcedit: ec [**coveragename**]
Arcedit: ef poly
Arcedit: drawe poly fill
Arcedit: draw

(These polygons can be used as guidance for breaking the loops in ArcEdit or can be viewed in ArcView for the same purpose)

Preliminary watershed delineation with 90m DEMs

To get a 30m DEMs grid clipped to the basin

Download DEMs
90m DEMs are available at http://edcwww.cr.usgs.gov/doc/edchome/ndcdb/ndcdb.html. It is good to choose ftp via graphics, then the right one-degree quad sheets can be selected for downloading by clicking a map. For doing this is, a map of the counties for the area is convenient, using the county map as a guide, download all necessary DEMs in compressed (gzip) format and unzip using Winzip.

Convert USGS DEMs to ArcInfo grids
Arc : demlattice **DEMname DEM1**

Merge all the DEMs
Arc : Grid
Grid : **Grid1 =** merge (**DEM1, DEM2, …**)

The DEMs are floating point grids and have to be converted to integer grids to reduce the file size.
Grid : **Grid2** = int (**Grid1**)

Get basin boundary and build frame
Add theme j:/texas/state_coverages/tx_basins
Theme/query on basin name

Theme/covert to shapefile your selected basin
Arc : shapearc **Basin Basin**
Arc : build **Basin**
Arc : project grid **Basin Basin_alb**
Arc : buffer **Basin_alb Frame** # # 10000 #

Resize DEM extent with setwindow
Grid : mapex **Frame**
Grid : setwindow **Frame**
Grid : **Grid3** = **Grid2**

Select value cells in the Frame area
Grid : **dem_geo** = selectpolygon (**Grid3, Frame,** inside)
Grid : quit

Project the grid to TSMS Albers
Arc : project grid **dem_geo dem_albers** geo2albers.txt

Kill the temporary grids
Arc : kill **DEM1 …** all
Arc : kill **Grid1** all
Arc : kill **Grid2** all
Arc : kill **Grid3** all
Arc : kill **dem_geo** all

Watershed delineation in Arc View using CRWR-PrePro03

CRWR-PrePro is a convenient tool to use in Arc View when dealing with watershed delineation; PrePro stands for PreProcessor of GIS data for hydrologic models.

Prepro03.apr is an Arc View project file that contains the Scripts, Menus and Buttons for running CRWR-PrePro and is downloadable from http://www.ce.utexas.edu/prof/olivera/prepro/prepro.htm

Using CRWR-PrePro the watersheds of the TNRCC water quality segment for the Trinity River Basin were delineated. The data used were the 90 m DEM, the "cleaned" river network and the downstream most points of each TNRCC water quality segment; the products were 41 watersheds, which were later used as guides for finer resolution delineation.

Watershed delineation using 30m DEMs

To get a 30m DEMs grid clipped to the basin

Copy all the right DEMs to a directory.
Arc : copy e:/…/**dem9934 dem9934**

Multiple by 100 to the protect accuracy of the DEMs, i.e. values are changed from meters to centimeters.
Grid : **mult9934** = 100 * **dem9934**

The DEMs are floating point grids and have to be converted to integer grids to reduce the file size.
Grid : **int9934** = int(**mult9934**)

Merge all the DEMs
*Grid : **dem_geo** = merge(**int9934**, **int9834**, **int9734**, …)*

Project the grid to TSMS Albers
Arc : project grid **dem_geo dem_albers** geo2albers.txt

Kill the temporary grids
Arc : kill **mult9934** all
Arc : kill **int9934** all
Arc : kill **dem_geo** all

Get basin boundary and build frame
Add theme k:/state/tx_basins
Theme/query on basin name
Theme/covert to shapefile your selected basin
Arc : shapearc **Basin Basin**
Arc : build **Basin**
Arc : project grid **Basin Basin_alb**
Arc : buffer **Basin_alb Frame** # # 10000 #

Resize DEM extent with setwindow
Grid : mapex **Frame**
Grid : setwindow **Frame**
Grid : **Grid3** = **dem_albers**

Select value cells in the Frame area
Grid : **dem30m** = selectpolygon (**Grid3, Frame,** inside)
Grid : quit

Watershed delineation in Arc Info

Identify the adjacent rivers that fall on the 10km thick frame by HUCs and downloaded from the EPA website: http://www.epa.gov/OST/BASINS/STATES/TX/

Double click on each file to unzip with WinZip
Arc : shapearc 11130201 11130201
Arc : project cover 11130201 11130201_alb geo2alb

*Add all the rf3 files along with the **a_rf3trin** file and convert to shapefile*

Merge individual shapefiles with mergethemes script
In project window, select scripts, new, load script
Select mergethemes.ave
Compile and run, make output shapefile rf3_frame

Convert shapefile to a grid
Select the part of **rf3_frame** that is within the **frame**.
Open the **rf3_frame** attribute table, select table, start editing, select edit, add field and populate the new area, one, with the number 1 for all records. Select theme, convert to grid, **riverframe**, with analysis properties same as dem_30m and pick the field one for cell values.

*Burn the **dem_30m** with the **riverframe***
Grid: setcell **dem_30m**
Grid: setwindow **dem_30m**
Grid: **demstr** = **riverframe** * **dem_30m**
Grid: **demplus** = **dem_30m** +100000
Grid: **burndem** = merge (**demstr** , **demplus**)
Grid: buildvat **burndem**

For the delineation, make a grid that only encompasses the river lines that make up the segments and has their number as a value:
Using the **segsfy99** coverage a shapefile, **segmentdp** with the downstream points of each segment is made. The **a_rf3trin** arcs are broken where ever a segment starts or ends according to **segsfy99** and a field is added in the **a_rf3trin** with the segment number. The arcs that correspond to the segments are selected and converted to the grid **seggrd** with analysis properties same as **dem_30m** and pick the field with the segment number for cell values.

If basin is big like the Trinity Basin, the watershed has to be divided in two or three parts, here it was divided in three parts, a, b and c by using the former defined boundaries by 90m DEM watershed delineation.

For each watershed a b and c do:
In Arc View: Watersheds from 90m delineation selected, and converted to shapefiles, **water_a**, **water_b** and **water_c**
Arc: shapearc **water_a water_a**
Arc: clean **water_a**
Arc: build **water_a**
Arc: buffer **water_a frame_a** # # 10000 #
Arc: grid
Grid: setcell **dem_30m**
Grid: setwindow **frame_a**
Grid: mapex **frame_a**
Grid: burn_a = selectpolygon (**burndem**, **frame_a**, inside)
Grid: fill **burn_a fill_a** # # **fdr_a**
Grid: **fac_a** = flowaccumulation (**fdr_a**)
Grid: **wsheds_a** = watershed (**fdr_a**, **seggrd**)

When this has been done for all three parts of the basin there is time for merging.
Select the right watersheds in ArcView, again with the 90m watersheds as a guide and skip the small watersheds that showed up because of the frame and save them as **wsgrd_a**, **wsgrd_b** and **wsgrd_c**
Grid: wsgrd = merge (**wsgrd_a**, **wsgrd_b**, **wsgrd_c**)
Arc: gridpoly **wsgrd wspoly3**
Arc clean **wspoly3** # # 30

Now some single cells show up with the value 0 instead of segment number and they are separate polygons. Fix it…
Edit the attribute table of **wspoly3** and add the number of the neighboring segment instead of the value 0 in the **gridcode** field. Then…
Arc: dissolve wspoly3 wspoly33 grid-code poly
Arc: copy wspoly33 wspoly
If everything looks good
Arc: copy wspoly33 watersheds

Clipping grids in Arc Info

Using a shapefile "801.shp" consisting of the outline of segment 801, to get a grid clipped to the outline of the watershed.
Arc: shapearc 801.shp 801
Arc: build 801
Arc: grid
Grid: dem = selectpolygon(dem_30m, 801, inside)
Grid: fac = selectpolygon(fac_a, 801, inside)
Grid: fdr = selectpolygon(fdr_a, 801, inside)

Research Triangle Institute's Reach Indexing Tool (RIT)

The RIT is an ArcView project file designed for use with RF3 data under contract with the EPA for agencies involved in TMDL development and water quality management. The project file, sample data, Documentation, and User Guide are Copyright 1999 Research Triangle Institute. They are freely available from Research Triangle Institute, 3040 Cornwallis Road, Research Triangle Park, North Carolina 27709 USA Telephone: 919-541-6000

APPENDIX B

Avenue Scripts

Mergethemes

```
'MergeThemes

theView=av.GetActiveDoc
theThemes=theView.GetThemes

if (theThemes.Count<2) then
	MsgBox.Error("Must have at least two themes in a view to merge.","")
	exit
end

themesToMerge=List.Make
while (true)
	t=MsgBox.Choice(theThemes,"Choose themes in view to merge:"+NL+"(Click Cancel to
end):","Merge Themes")
	if (t<>Nil) then
		themesToMerge.Add(t)
	else
		break
	end
end

if ((themestoMerge=Nil) or (themesToMerge.Count<2)) then
  MsgBox.Error("Not enough themes to merge.","")
  exit
end

checkType=themesToMerge.Get(0).GetFTab.FindField("Shape").GetType
for each i in 1..(themesToMerge.Count-1)
	t=themesToMerge.Get(i)
	if (checkType<>t.GetFtab.FindField("Shape").GetType) then
		MsgBox.Error("Theme feature type mismatch -- unable to merge.","")
		exit
	end
end

outFName=av.GetProject.MakeFileName("theme","shp")
outFname=FileDialog.Put(outFname,"*.shp","Output Merged Shapefile")
if (outFName=Nil) then
	exit
end

fieldList=List.Make
for each f in themesToMerge.Get(0).GetFTab.GetFields
	if (f.GetName="Shape") then
		continue
	else
```

```
                fCopy=f.Clone
                fieldList.Add(fCopy)
        end
end

shapeType=themesToMerge.Get(0).GetFTab.FindField("Shape").GetType
if (shapeType=#FIELD_SHAPELINE) then
        outClass=POLYLINE
elseif (shapeType=#FIELD_SHAPEMULTIPOINT) then
        outClass=MULTIPOINT
elseif (shapeType=#FIELD_SHAPEPOINT) then
        outClass=POINT
elseif (shapeType=#FIELD_SHAPEPOLY) then
        outClass=POLYGON
else
        MsgBox.Error("Invalid shape field type.","Merge Themes")
        exit
end
mergeFTab=FTab.MakeNew(outFName,outClass)

if (fieldList.Count>0) then
        mergeFTab.AddFields(fieldList)
end

for each t in themesToMerge
        av.ShowMsg("Merging"++t.GetName)
        inFTab=t.GetFTab
        if (inFTab.GetSelection.Count=0) then

                theRecordsToMerge=inFTab
                numRecs=inFTab.GetNumRecords
        else
                theRecordsToMerge=inFTab.GetSelection
                numRecs=theRecordsToMerge.Count
        end
        for each rec in theRecordsToMerge
                av.SetStatus((rec/numRecs)*100)
                newRec=mergeFTab.Addrecord
                inField=inFTab.FindField("Shape")
                outField=mergeFTab.FindField("Shape")
                mergeFTab.SetValue(outField,newrec,inFTab.ReturnValue(inField,rec))
                if (fieldList.Count>0) then
                        for each f in fieldList
                                fName=f.GetName
                                inField=inFTab.FindField(fName)
                                if (inField<>Nil) then
                                        outField=mergeFTab.FindField(fName)
                                        aValue=inFTab.ReturnValue(inField,rec)
                                        mergeFTab.SetValue(outField,newRec,aValue)
                                end
                        end
                end
        end
end

av.ClearMsg
```

```
av.ClearStatus

if (MsgBox.YesNo("Add shapefile as theme to view?","MergeThemes",true).Not)then
	exit
end

viewList={}
for each d in av.GetProject.GetDocs
	if (d.Is(View)) then
		viewList.Add(d)
	end
end

viewList.Add("")
addToView=MsgBox.ListAsString(viewList,"Add Theme to:","Merge Themes")

if (addToView<>Nil) then
	if (addToView="") then
		addToView=View.Make
		addToView.GetWin.Open
	end

	mergeTheme=FTheme.Make(mergeFTab)
	addToView.AddTheme(mergeTheme)

	addToView.GetWin.Activate
end
```

Nodes

```
' Nodes.ave
' version: 1.1.
' date   : 6/1998
'
' This script draws active theme nodes and vertices. The node errors are distinguished.
' The theme must have line or polygon topology. It is reccomended to associate this script
' with popup on right mouse button to enable run it during editing process.
' (The nodes are drawn only in actual View extent without saving them.)
'
' Color coding:  BLUE  - vertices
'               GREEN - regular nodes
'               CYAN  - pseudo nodes
'               RED   - dangle nodes
'
' author : Stepan Kafka, Kutna Hora District Council, Czech republic
' e-mail : oukh@kh.cesnet.cz
'
'-----------------------------------------------------------------------------------
'Example of update script:
'ActiveThemes = av.GetActiveDoc.GetActiveThemes
'if (ActiveThemes.Count <> 0) then
'  theTheme = ActiveThemes.Get(0)
'  if (theTheme.Is(FTheme)) then
'    S = theTheme.GetFTab.FindField("Shape")
'    if (S <> nil) then
'      SELF.SetEnabled((S.GetType = #FIELD_SHAPELINE) or (S.GetType = #FIELD_SHAPEPOLY))
'      exit
'    end
'  end
'end
'SELF.SetEnabled(false)
'-----------------------------------------------------------------------------------


'--- Symbol settings for Node representation -----
'- vertex
SVert = av.GetSymbolWin.GetPalette.GetList(#PALETTE_LIST_MARKER).Get(1).Clone
SVert.SetColor(Color.GetBlue)
SVert.SetSize(4)

'- Regular Node
SNode = av.GetSymbolWin.GetPalette.GetList(#PALETTE_LIST_MARKER).Get(1).Clone
SNode.SetColor(Color.GetGreen)
SNode.SetSize(6)

'- Pseudo Node
SPseudo = av.GetSymbolWin.GetPalette.GetList(#PALETTE_LIST_MARKER).Get(1).Clone
SPseudo.SetColor(Color.GetCyan)
SPseudo.SetSize(6)

'- Dangle Node
SDangle = av.GetSymbolWin.GetPalette.GetList(#PALETTE_LIST_MARKER).Get(1).Clone
SDangle.SetColor(Color.GetRed)
```

```
SDangle.SetSize(6)
'-----------------------------------------------------------------------------------------

'--- Initializing
theView = av.GetActiveDoc
thePrj = theView.GetProjection
theSelMode = theView.GetSelectMode
theView.SetSelectMode(#GRAPHICS_SELECT_NORMAL)
theTheme = theView.GetActiveThemes.Get(0)
theTable = theTheme.GetFTab
D = theView.GetDisplay
theShape = theTable.FindField("Shape")
NodeList = {}

'---- Selecting the shapes
OldSel = theTable.GetSelection.Clone
DExt = D.ReturnVisExtent
theTheme.SelectByRect(DExt, #VTAB_SELTYPE_NEW)
CurrSel = theTable.GetSelection.Clone
theTable.SetSelection(OldSel)

'--- Vertices drawing and Nodes collecting ------
D.BeginClip
for each rec in CurrSel
  theLines = theTable.ReturnValue(theShape, rec).AsPolyLine.Explode
  for each L in theLines
    thePoints = L.AsMultiPoint.ReturnProjected(thePrj)
    D.DrawMultiPoint(thePoints, SVert)
    NodeList.Add(thePoints.AsList.Get(0))
    NodeList.Add(thePoints.AsList.Get(thePoints.Count-1))
  end
end

'--- Nodes processing -------------------------
AllNodes = NodeList.Count-1
av.ShowMsg("Searching nodes...")
av.ShowStopButton
while (NodeList.Count > 0)
  OverPos = 0
  thePoint= NodeList.Get(0)
  NodeList.Remove(0)
  Nodes = NodeList.Count-1
  if (av.SetStatus((AllNodes-Nodes)/AllNodes*100).not) then
    av.SetStatus(100)
    av.ShowMsg("Cancelled by operator.")
    exit
  end
  C = 0
  while (C <= Nodes)
    if (thePoint.Intersects(NodeList.Get(C))) then
      NodeList.Remove(C)
      Nodes = Nodes - 1
      OverPos = OverPos + 1
    else
      C = C + 1
    end
```

```
  end
  if (OverPos > 1) then
    D.DrawPoint(thePoint, SNode)
  elseif (OverPos = 1) then
    D.DrawPoint(thePoint, SPseudo)
  else
    D.DrawPoint(thePoint, SDangle)
  end
end
D.EndClip
av.SetStatus(100)
av.ClearMsg
theView.SetSelectMode(theSelMode)
av.PurgeObjects
```

Addtopo

```
'Name:  addtopo
'
'Title:  Adds a topographic map to a view (DRGs), cuts the edges off and zooms in...
'
'Topics:  Views
'
'Autor: Jona Finndis Jonsdottir  jonafinndis@mail.utexas.edu, 2/22/1999
'
'Instruction: - replace the Txmesh_utm15.shp name with the name of your mesh
'             - change h:/data/ to the path of where the DRGs are

theVal = SELF
theView = av.GetActiveDoc
if (not (theVal.IsNull)) then
  theVal2 = "h:/data/O" + TheVal + ".tif".AsString

lookupname = "Attributes of Txmesh_utm15.shp"

lookuptab=av.getproject.finddoc(lookupname)
if (lookuptab=nil) then
  msgbox.info("Can't find Attribute Table of Txmesh_utm15.shp","")
  exit
end

lookupvtab=lookuptab.getvtab

theField = lookupvtab.FindField("Cd_name")

found = false
p = theView.GetDisplay.ReturnUserPoint
for each t in theView.GetActiveThemes
 if ((t.HasAttributes) and (t.GetHotField <> nil)) then
   recs = t.FindByPoint(p)
   for each rec in recs
     found = true
     theVal3 = t.ReturnValueString(theField.GetName, rec)
     end
 end
end
if (not found) then
 System.Beep
end


if (not (File.Exists(theVal2.AsFileName))) then
MsgBox.Info("Insert " +theVal3+ " CD","" )
end

  if (File.Exists(theVal2.AsFileName)) then
          'Create the SourceName...
          theSrc = SrcName.Make(theVal2)

          'Use the SourceName to make a theme...
```

```
        aTheme = Theme.Make(theSrc)

        'Add the theme to the view...
        theView.AddTheme(aTheme)

        'Set a new name for the theme...
        aTheme.SetName(theVal)

        'Change the extent of the DRG, i.e. cut the edges off.
        r = Rect.MakeEmpty
        r = r.UnionWith(aTheme.ReturnExtent)
bottom1 = r.GetBottom
top1 = r.GetTop
left1 = r.GetLeft
right1 = r.GetRight
height1 = r.GetHeight
width1 = r.GetWidth
heighta = height1*0.105
heightb = height1*0.050
widtha = width1*0.0615
widthb = width1*0.0615

'heighta = height1*0.102305072
'heightb = height1*0.0470882918
'widtha = width1*0.0654828683
'widthb = width1*0.0673380772
bottom2 = (bottom1 + heighta)
top2 = top1 - heightb
left2 = left1 + widtha
right2 = right1 - widthb
r2 = rect.makeXY(left2, bottom2, right2, top2)
if (r2 <> NIL) then
  aTheme.GetImgSrc.SetClipExtent(r2)
end

        'Draw the theme...
        aTheme.SetVisible(true)

        'Make txmesh unactive
        for each t in theView.GetActiveThemes.clone
        t.SetActive( false )
        end

        aTheme.SetActive(true)

        'Zoom
        av.GetProject.SetModified(true)
        theThemes = theView.GetActiveThemes
        r = Rect.MakeEmpty
        for each t in theThemes
                r = r.UnionWith(t.ReturnExtent)
        end

if (r.IsEmpty) then
 return nil
elseif ( r.ReturnSize = (0@0) ) then
```

```
  theView.GetDisplay.PanTo(r.ReturnOrigin)
 else
  theView.GetDisplay.SetExtent(r.Scale(1.1))
 end
   else
            MsgBox.Warning("File "+theVal2+" not found.","Hot Link" )
   end
end
```

Snap_pnts

```
'*****************************************************************************
'Snap_pnts
'description: locate the point coverage to the corresponding stream network. If
'       the point is out of tolerance and cannot be snapped, the point will be
'       highlighted. The output coverage is stored as "Virtualpnts".
'Copyright to: Richard Gu 1/14/99
'The code can be copied or modified as long as this title is kept.
'Significant Corrections:  Kim Davis --June 8,1999
'*****************************************************************************
msgbox.info("Welcome to Snap Tool! This tool will create a new point coverage called [Virtualpnts],
which represents the points snapped. A message box will ask you to input a SNAP_DISTANCE, which is
the tolerance value used for snapping.","")

SNAP_DISTANCE=msgbox.input("Enter a value of distance to snap:","SNAP_DISTANCE","500")
sMsg="Snap Tool"
av.GetProject.SetModified(true)
theView = av.GetActiveDoc
ThmList=TheView.GetThemes

problemlist=list.make

if(ThmList.count=0)then
  Msgbox.Info("No themes were found in the View:"++TheView.GetName, sMsg)
  Exit
end

'*****make a list for selection*****
LineThm=Msgbox.ChoiceAsString(ThmList, "Select a line theme.", sMsg)
if(LineThm.is(FTHEME))then
  LineFtab=LineThm.getFtab
  TheClassName=LineFTab.GetShapeClass.GetClassName
  if((TheClassName = "polyline").Not) then
    Msgbox.Error("Selected theme is not a polyline theme", sMsg)
    Exit
  end
else
  Msgbox.Error("Selected theme is not a polyline theme", sMsg)
  Exit
end
LineFtab.SetEditable(true)

Pntthm=Msgbox.choiceasstring(thmlist, "Select a point theme.", sMsg)
if(PntThm.is(FTHEME))then
  PntFtab=PntThm.getFtab
  TheClassName=PntFTab.GetShapeClass.GetClassName
  if((TheClassName = "point").Not) then
    Msgbox.Error("Selected theme is not a point theme", sMsg)
    Exit
  end
else
  Msgbox.Error("Selected theme is not a point theme", sMsg)
  Exit
end
```

```
PntFtab.SetEditable(true)
PntWorkList=List.make

newthmname="Virtualpnts"
thedir=av.getproject.getworkdir
newfilename=fn.merge(thedir.asstring,newthmname)

newftab=ftab.makenew(newfilename,point)
newthm=ftheme.make(newftab)
pntfld=pntftab.findfield("Shape")
pntfields=pntftab.getfields
pntfields=pntfields - {pntfld}

newfields=pntfields.deepclone

newftab.addfields(newfields)
theview.addtheme(newthm)
newftab.seteditable(true)
newpntfld=newftab.findfield("Shape")

for each rec in pntftab
  pntWorkList.add(rec.clone)
end


Total=pntWorkList.count
Lmt_work=Total-1

RTheme=LineThm
'Inpnt=thepnt
RFTab=Rtheme.getFtab
RShape=RFtab.FindField("Shape")


for each item in pntftab
  thepntrec=item.clone
  Inpnt=pntftab.returnvalue(pntfld, thepntrec)
'  msgbox.info("Inpnt"++inpnt.asstring,"")

if(InPnt=nil)then
  Msgbox.info("No point is selected","")
  exit
end
LineFtab=LineThm.getFtab

'*****interface to snap operation*****
theGraphics = theView.GetGraphics

TolCircle=Circle.Make(InPnt, SNAP_DISTANCE.AsNumber)
'gCircle = GraphicShape.Make(TolCircle)
'aSymbol = VectorFill.Make
'aSymbol.SetStyle(#VECTORFILL_STYLE_HATCH)
'aSymbol.SetAngle(45)
'aSymbol.SetColor(Color.GetYellow)
'aSymbol.SetOLColor(Color.GetRed)
'gCircle.SetSymbol(aSymbol)
```

```
'theGraphics.Add(gCircle)

LineThm.SelectByShapes({TolCircle},#VTAB_SELTYPE_NEW)
recs=LineFTab.GetSelection

'Msgbox.info("recs has"++recs.count.asstring++"records","")

theshpV=nil

if(recs.Count <> 0)then
'  Msgbox.info("Record number"++recs.count.asstring,"Debug")
  DISTANCE=SNAP_DISTANCE.AsNumber
  for each lnrec in recs
    RshpV=RFtab.ReturnValue(RShape,lnrec)
    dist=InPnt.distance(RshpV)
'    MsgBox.Info("dist="++dist.asstring+nl+"DISTANCE="++DISTANCE.AsString,"Debug")
    if(dist<=DISTANCE)then
      DISTANCE=dist
      theshpV=RshpV
    end        'if(dist<DISTANCE)
  end          'for each lnrec in recs
  temppnt=Inpnt.clone
  FOUND=temppnt.snap(theshpV,SNAP_DISTANCE.asnumber)
  if(FOUND=true)then
    newpnt=temppnt
'    msgbox.info("snapped.","")
  else           'if (Found=true)
'*****************************************************************************
  ThePntV=InPnt
  TheShpV=RshpV

'Msgbox.info("SnapLinPnt is being executed.","")

PntList=TheShpV.AsMultiPoint.AsList
ccx=ThePntV.getX
ccy=ThePntV.getY
Clength=0.0
DistList=List.Make  'Dists between each potential snap point and original point.
NPntList=List.Make  'Potential snap-to points.
ClenList=List.Make  'Dists between from node to each potential snap-to points.
if(PntList.count=2)then
  aax=PntList.get(0).getX
  aay=PntList.get(0).getY
  bbx=PntList.get(1).getX
  bby=PntList.get(1).getY
  TLength=(((bbx-aax)^2)+((bby-aay)^2))^(0.5)
  AC=((bbx-aax)*(ccx-aax))+((bby-aay)*(ccy-aay))
  CLength=AC/TLength  'A.C=A.abs*C.abs*Cos(AC), Clength=C.abs*Cos(AC), A.abs=TLength
  NewX=aax+((bbx-aax)*(CLength/TLength))
  NewY=aay+((bby-aay)*(CLength/TLength))
  NewPnt=Point.Make(NewX,NewY)
  PFound=true
else               'if(PntList.count=2)
  pcnt=PntList.Count-2
  PFound=False
  '--Start computation loop
```

```
OldAC=nil
TLength=0.0
for each idx in 0..pcnt
'   Msgbox.info("processing...","")
  aax=PntList.get(idx).getX
  aay=PntList.get(idx).getY
  bbx=PntList.get(idx+1).getX
  bby=PntList.get(idx+1).getY
  AC=((bbx-aax)*(ccx-aax))+((bby-aay)*(ccy-aay))  'chkingDotProductOf A->C->
  BC=((aax-bbx)*(ccx-bbx))+((aay-bby)*(ccy-bby))  'chkingDotProductOF B->C->
  LAC=((ccx-aax)*(ccx-aax)+((ccy-aay)*(ccy-aay))) 'chking Dist between A and C 8/1
  if (LAC < 5) then
    PFound=True
    DistList.Add(LAC.Clone)
    NPntList.Add(PntList.get(idx).Clone)
    CLenList.Add(Clength.Clone)
  end              'if(LAC < 5)   8/1

  ABL=(((bbx-aax)^2)+((bby-aay)^2))
  TLength=TLength+(ABL.Sqrt)
  if((AC<0) or (BC<0))then  'segment AB doesn'tContain Point C
    if(OldAC=nil)then
      OldAC=AC
    else     'if(OldAC=nil)
      if((OldAC*AC)<0)then  'Angle changes from <90 to >90, the end point contains thePntV
        NewX=aax
        NewY=aay
        NewPnt=Point.Make(NewX,NewY)
        ACLength=(((ccx-aax)^2)+((ccy-aay)^2)).sqrt
        if(Not (ACLength.IsNull))then
          DistList.Add(ACLength.Clone)
          NPntList.Add(NewPnt.Clone)
          ClenList.Add(Clength.Clone)
          PFound=True   '------set
        end       'if(Not (ACLength.IsNull))
      else        'if((OldAC*AC)<0)
        OldAC=AC
      end         'if((OldAC*AC)<0)
    end           'if(OldAC=nil)
      Clength=Clength+(ABL.sqrt)
  else  'if((AC<0) or (BC<0)) => Segment AB contains point C
    ACLength=(((ccx-aax)^2)+((ccy-aay)^2)).sqrt
    NewX=aax+((bbx-aax)*AC/ABL)
    NewY=aay+((bby-aay)*AC/ABL)
    CLength=Clength+(AC/(ABL.sqrt))   'AC=A dot C
    CosA=AC/(ABL*ACLength)
    SinA=(1-(CosA^2)).sqrt
    Dist=ACLength*SinA
    if(Not (Dist.IsNull))then  'Zye 7/8/97
      DistList.Add(Dist.Clone)
      NewPnt=Point.Make(NewX,NewY)
      NPntList.Add(NewPnt.Clone)
      CLenList.Add(Clength.Clone)
      PFound=True  '-------------set
    end
  end       'if((AC<0) or (BC<0))
```

```
   end 'for each idx
 end 'if(PntList.count=2)

if(PFound.Not)then
  newpnt=nil
else
  if(PntList.Count>2)then
    Nmatch=NPntList.Count
    Ndx=0
    MinDist=DistList.Get(0)
    NewPnt=NPntList.Get(0)
    CLength=ClenList.Get(0)
    if(Nmatch>1)then
      for each i in 1..(Nmatch-1)
        tmpDist=DistList.Get(i)
        if(TmpDist<MinDist)then
          MinDist=tmpDist.Clone
          Clength=ClenList.Get(i).Clone
          NewPnt=NPntList.Get(i).Clone
        end  'if(TmpDist<MinDist)
      end  'for each i in 1..(NMatch-1)
    end  'if(NMatch>1)
  end  'if (PntList.Count>2)
' MsgBox.info("Return results?","")
' return({NewPnt,Clength,TLength})  'Normal return point.
' MsgBox.info("Returned","")
end 'if(PFound.not)
'*****************************************************************************
    if(newpnt=nil)then
      problemlist.add(Inpnt)
    end
'    msgbox.info("Snap is not successful.","")
end
else
' MsgBox.info("Nil record has been found","")
  problemlist.add(Inpnt)
  newpnt=nil
end

'******end of snapping******


if(NewPnt=nil)then
  continue
else

'---this sentence is new

  newpntrec=newftab.addrecord
  newftab.setvalue(newpntfld, newpntrec, newpnt)

for each fldrec in pntfields
  fldvalue=pntftab.returnvalue(fldrec, thepntrec)
  newfld=newftab.findfield(fldrec.asstring)
  newftab.setvalue(newfld, newpntrec, fldvalue)
end
```

```
  av.setstatus(item/total*100)

  recs.ClearAll
' theGraphics.Empty
end

End  '-------end of the main (for each item..) loop

pntthm.selectbyshapes(problemlist,#VTAB_SELTYPE_NEW)

av.setstatus(100)
pntFTab.SetEditable(false)
system.beep
msgbox.info("The process is complete.","")
Exit
```

NetFlip

```
'****************************************
' Name: NetFlip.Dictionary
' History:  Developed by Kim Davis and Francisco Olivera CRWR-UT
' Date:  30 MAR 1999
' Self: Tool
' Returns:  None
' Description:  Detects arcs in a network that point upstream and
'               flips them so that they point downstream.  Switches
'               FromNode and ToNode values to reflect new direction.
' Requires: Active Polyline Theme with valid Arc-Node Topology
' Dictionary Structure:  Key = To/From Node Number
'                        Value = List of ArcIDs
'****************************************
sMsg="Hydronet NetFlip"
av.UseWaitCursor
'*** Selecting the HydroNet polyline theme

TheProject=av.GetProject
TheView=av.getActiveDoc
TheThemes=theView.GetActiveThemes
TheDisplay=TheView.GetDisplay

if(TheThemes.count=0)then
  Msgbox.Error("No themes were found in View:"++TheView.GetName, sMsg)
  Exit
end
if (TheThemes.count > 1) then
  LineThm=Msgbox.ChoiceAsString(TheThemes, "Select a polyline theme.",
sMsg)
  if(LineThm=nil)then
    exit
  end
else
  LineThm=TheThemes.Get(0)
end
if(LineThm.is(FTHEME))then
  LineFtab=LineThm.getFtab
```

```
  TheClassName=LineFTab.GetShapeClass.GetClassName
  if((TheClassName = "polyline").Not) then
    Msgbox.Error("Selected theme is not a polyline theme", sMsg)
    Exit
  end
else
  Msgbox.Error("Selected theme is not a polyline theme", sMsg)
  Exit
end
LineFtab.SetEditable(true)

'** find proper from and to node items (due to the possibility that the
files
'** could have been processed on a workstation and/or ArcInfo.
LineTnode=LineFtab.FindField("Tnode_")
LineFnode=LineFtab.FindField("Fnode_")
  if(LineTnode=nil)then
    LineTnode=LineFtab.FindField("Tnode#")
    LineFnode=LineFtab.FindField("Fnode#")
  end
  if(LineTnode=nil)then
    LineTnode=LineFtab.FindField("To_node")
    LineFnode=LineFtab.FindField("From_node")
  end
if(LineFnode=nil)then  'if ToNode/FromNode not in the field, exit
  msgbox.error("From/To Node items missing in the polyline theme:
"+LineThm.AsString+nl++nl+"Run HydroNet.Topology to correct the data
structure.", sMsg)
  exit
end

FNDict=Dictionary.Make(LineFTab.GetNumRecords)
TNDict=Dictionary.Make(LineFTab.GetNumRecords)
NewList={}
LineShape=LineFtab.FindField("Shape")
IsDone=LineFtab.FindField("IsDone")
if(IsDone=nil) then
  IsDone=field.make("IsDone",#FIELD_SHORT,2,0)
  LineFTab.addFields({IsDone})
end

'**********get user-clicked reach********
p = theView.GetDisplay.ReturnUserPoint
op = #VTAB_SELTYPE_NEW
  if (LineThm.CanSelect) then
    LineThm.SelectByPoint(p, op)
  else
    msgbox.error("Cannot select features from"++LineThm.AsString, sMsg)
    exit
  end
TheDisplay.Flush

'**************populating the dictionaries*********
av.ShowStopButton

for each arc in LineFTab
  progress = (arc.clone/LineFTab.GetNumRecords) * 100
```

```
  doMore = av.SetStatus( progress )
  if (not doMore) then
    break
  end
  LineFTab.SetValue(IsDone,arc.clone,0)
  TNno=LineFTab.ReturnValue(LineTNode,arc.clone)
  FNno=LineFTab.ReturnValue(LineFNode,arc.clone)
  test=TNDict.Get(TNno)
  if (test<>nil) then
    NewList=test.Add(arc.clone)
  else
    NewList={arc.clone}
  end
  TNDict.Set(TNno,NewList)
  test=FNDict.Get(FNno)
  If (test<>nil) then
    NewList=test.Add(arc.clone)
  else
    NewList={arc.clone}
  end
  FNDict.Set(FNno,NewList)
end
av.SetStatus(100)
av.clearstatus

'**************checking over the dictionary***********
'DebugList=TNDict.ReturnKeys
'MsgBox.ListAsString(DebugList,"To Node Dictionary", "Debug")
'For each nodeno in DebugList
'  MsgBox.ListAsString(TNDict.Get(nodeno),"List for
node:"++nodeno.asstring,"Debug")
'end
'DebugList=FNDict.ReturnKeys
'MsgBox.ListAsString(DebugList,"From Node Dictionary", "Debug")
'For each nodeno in DebugList
'  MsgBox.ListAsString(FNDict.Get(nodeno),"List for
node:"++nodeno.asstring,"Debug")
'end

'********************working with the selection********
TheBitmap=LineFtab.GetSelection
MatchID=TheBitmap.GetNextSet(-1) 'finding the outlet and setting
variables
FNMatch=LineFtab.ReturnValue(LineFNode,MatchID)
LineFTab.SetValue(IsDone,MatchID,1)

'********Removes MatchID from the FNDict and TNDict
NewList=FNDict.Get(FNMatch)
  if (NewList.Count=1)  then
    FNDict.Remove(FNMatch)
  else
    otherList=NewList-{MatchID}
    FNDict.Set(FNMatch,otherList)
  end
TNMatch=LineFtab.ReturnValue(LineTNode,MatchID)
NewList=TNDict.Get(TNMatch)
  if (NewList.Count=1)  then
```

```
    TNDict.Remove(TNMatch)
  else
    otherList=NewList-{MatchID}
    TNDict.Set(TNMatch,otherList)
  end

workinglist={}

'*************big work loop*************
av.ShowStopButton
'chk = -1       'control variable for while loop
'while (chk<>0)
doMore = True
chk = 0
while (doMore)
  progress = (chk/LineFTab.GetNumRecords)
  doMore = av.SetStatus( progress )

'********Checks for correctly oriented arcs (flowing into MatchID)
  Into=TNDict.Get(FNMatch)
  if (Into<>nil) then
    TNDict.Remove(FNMatch)
    for each Item in Into
      ArcID=Item.clone
      FNno=LineFTab.ReturnValue(LineFNode,ArcID)
      LineFTab.SetValue(IsDone,ArcID,1)         'sets IsDone to
indicate arc points downstream
'********Removes arc from FNDict so it won't detect itself in
Headtest(below)
      NewList=FNDict.Get(FNno)
      if ((NewList.Count)=1)  then
        FNDict.Remove(FNno)
      else
        otherList=NewList-{ArcID}
        FNDict.Set(FNno,otherList)
      end
'********test to see if ArcID is a head reach (does anything connect to
FNno)
      Headtest1=TNDict.Get(FNno)               'arcs that flow into
head of ArcID
      Headtest2=FNDict.Get(FNno)               'arcs that originate at
head of ArcID
      if ((Headtest1<>nil)or(Headtest2<>nil)) then  'something is
connected to head of ArcID
        workinglist.add(ArcID)
      end
    end       'for each item in into
  end       'if (Into<>nil)
'********checks for incorrectly oriented arcs (flowing out of MatchID)
  Outof=FNDict.Get(FNMatch)
  FNDict.Remove(FNMatch)
  if (Outof<>nil) then
    for each Item in Outof
      ArcID=Item.clone
      workinglist.Add(ArcID)
      TNno=LineFTab.ReturnValue(LineTNode,ArcID)
      newList=TNDict.Get(TNno)
```

```
      if (NewList.Count=1) then
        TNDict.Remove(TNno)
      else
        otherList=NewList-{ArcID}
        TNDict.Set(TNno,otherList)
      end

    end
  end  'if outof<>nil
'*******flips arcs with IsDone=0 and sets new MatchID
  if(workinglist.IsEmpty.Not)then
    workinglist.RemoveDuplicates
    MatchID=workinglist.Get(0)
    workinglist.Remove(0)
     if(LineFtab.ReturnValue(IsDone,MatchId)=0)then
      TheArc=LineFtab.ReturnValue(LineShape,MatchID)
      FnodeV=LineFtab.ReturnValue(LineFNode,MatchID)
      TnodeV=LineFtab.ReturnValue(LineTNode,MatchID)
      NewArc=TheArc.Flip                          'flips arc and sets
new values in table
      LineFtab.SetValue(LineShape,MatchID,NewArc)
      LineFtab.SetValue(LineTnode,MatchID,FnodeV)
      LineFtab.SetValue(LineFnode,MatchID,TnodeV)
      LineFtab.SetValue(IsDone,MatchID,1)
    end              'if(LineFtab.ReturnValue(IsDone,MatchID)=0)
    FNMatch=LineFTab.ReturnValue(LineFNode,MatchID)
    TNMatch=LineFTab.ReturnValue(LineTNode,MatchID)
    chk = chk + 1              'ends while loop when list is emptied
  else
    doMore = False
  end                'if(workinglist.depth>0)
end                  'while(chk<>0)
'********************housecleaning*****************
TheProject.SetModified(True)
LineThm.ClearSelection
LineThm.SetVisible(false)
TheDisplay.Flush
LineThm.SetVisible(true)
'av.setStatus(100)
'av.ClearMsg
'TheDisplay.Flush
```

Downstream

```
'*************************************************************************
*******
'Script:  DownstreamPoints
'Requirements:  A point theme whose features are all snapped onto those
'               of a line theme.  The line theme must have correctly
assigned
'               from nodes and to nodes.
'Results:  New fields in the point attribute table.
'            AscArc= an embedded foreign key named for an ID field in
the line table
'            DSPoint= ID of the next downstream point
'            DSDist= Flow distance from the point to the outlet
'            PcntDist= Distance along the Associated Arc in % length,
with head=0%
'          New field in the line attribute table.
'            DSDist= Flow distance from downstream end of that arc to
the outlet
'Input:  Select the outlet arc of the polyline theme and then activate
the script.
'Type:  Button or Tool (recommend Button)
'Author:  Kim Davis  6-9-99
'*************************************************************************
********
sMsg="Downstream Point Finder"

'** Selecting the polyline and point themes
TheProject=av.GetProject
TheView=av.getActiveDoc
TheThemes=theView.GetThemes
TheDisplay=TheView.GetDisplay

if(TheThemes.count=0)then
  Msgbox.Error("No themes were found in View:"++TheView.GetName, sMsg)
  Exit
end
if (TheThemes.count >= 2) then
  LineThm=Msgbox.ChoiceAsString(TheThemes, "Select a polyline theme.",
sMsg)
  if(LineThm=nil)then
    exit
  end
  if(LineThm.is(FTHEME))then
    LineFtab=LineThm.getFtab
    TheClassName=LineFTab.GetShapeClass.GetClassName
    if((TheClassName = "polyline").Not) then
      Msgbox.Error("Selected theme is not a polyline theme", sMsg)
      Exit
    end
  else
    Msgbox.Error("Selected theme is not a polyline theme", sMsg)
    Exit
  end
  PointThm=Msgbox.ChoiceAsString(TheThemes, "Select a point theme.",
sMsg)
```

```
  if(PointThm=nil)then
    exit
  end
  if(PointThm.is(FTHEME))then
    PointFtab=PointThm.getFtab
    TheClassName=PointFTab.GetShapeClass.GetClassName
    if((TheClassName = "point").Not) then
      Msgbox.Error("Selected theme is not a point theme", sMsg)
      Exit
    end
  else
    Msgbox.Error("Selected theme is not a point theme", sMsg)
    Exit
  end
else
  MsgBox.Error("Must have at least two themes in the view (1 Point and 1
Polyline).",sMsg)
end

'** find proper from and to node items (due to the possibility that the
files
'** could have been processed on a workstation and/or ArcInfo.
LineTnode=LineFtab.FindField("Tnode#")
LineFnode=LineFtab.FindField("Fnode#")
  if(LineTnode=nil)then
    LineTnode=LineFtab.FindField("Tnode_")
    LineFnode=LineFtab.FindField("Fnode_")
  end
  if(LineTnode=nil)then
    LineTnode=LineFtab.FindField("To_node")
    LineFnode=LineFtab.FindField("From_node")
  end
if(LineFnode=nil)then  'if ToNode/FromNode not in the field, exit
  msgbox.error("From/To Node items missing in the polyline theme:
"+LineThm.AsString+nl++nl+"Run HydroNet.Topology to correct the data
structure.", sMsg)
  exit
end

'** Getting the ArcID field to use as key between tables
LineFields=LineFtab.GetFields
LineID=MsgBox.Choice(LineFields, "Select the Field
from"++LineThm.GetName++"to be used as a key between the point and line
themes.", sMsg)
if (LineID = nil) then
  exit
end

'** Getting the PointID field to write into DSPoint
PointFields=PointFTab.GetFields
PointID=MsgBox.Choice(PointFields, "Select the Field
from"++PointThm.GetName++"to be used as a unique ID.", sMsg)
if (PointID=nil) then
  exit
end

'**Setting up Field Items
```

```
LineFtab.SetEditable(true)
PointFtab.SetEditable(true)
LineShape=LineFtab.FindField("Shape")
LnDSDist=LineFtab.FindField("Dsdist")          'the field listing the
distance from the arc outlet downstream to the network outlet
PointShape=PointFtab.FindField("Shape")
AscArc=PointFtab.FindField(LineID.AsString)   'the key field which
relates a point to its associated arc (the arc it is located on)
PtPcnt=PointFtab.FindField("Percentdist")    'the field storing the
percent distance along the arc that the point is located at
DSPoint=PointFtab.FindField("Dspoint")       'the field storing the id
of the next downstream point
PtDSDist=PointFtab.FindField("Dsdist")

'**Adding Field Items that did not exist
NewFieldList={}
if(LnDSDist=nil) then
  LnDSDist=field.make("DSDist", #FIELD_FLOAT,12,2)
  LineFTab.AddFields({LnDSDist})
end
if(AscArc=nil) then

AscArc=field.make(LineID.AsString,LineID.GetType,LineID.GetWidth,LineID.
GetPrecision)
  NewFieldList.Add(AscArc)
end
if(PtPcnt=nil) then
  PtPcnt=field.make("PercentDist",#FIELD_FLOAT,6,2)
  NewFieldList.Add(PtPcnt)
end
if(DSPoint=nil) then

DSPoint=field.make("DSPoint",PointID.GetType,PointID.GetWidth,PointID.Ge
tPrecision)
  NewFieldList.Add(DSPoint)
end
if(PtDSDist=nil) then
  PtDSDist=field.make("DSDist",#FIELD_FLOAT,12,2)
  NewFieldList.Add(PtDSDist)
end
PointFTab.AddFields(NewFieldList)

'**grabbing the selection
TheBitmap=LineFtab.GetSelection
if (TheBitmap.Count <> 1) Then
  MsgBox.Error("You must select the outlet arc before using the DSPoint
Tool!",sMsg)
  exit
end

'**Making Dictionaries
FNDict=Dictionary.Make(LineFTab.GetNumRecords)
TNDict=Dictionary.Make(LineFTab.GetNumRecords)
PointDict=Dictionary.Make(PointFTab.GetNumRecords)
ArcDict=Dictionary.Make(PointFTab.GetNumRecords)

'**************populating the dictionaries*********
```

```
av.ShowStopButton
Total=LineFTab.GetNumRecords
av.ShowMsg("Building Node Dictionaries")
for each arc in LineFTab
  progress = (arc.clone/Total) * 100
  doMore = av.SetStatus( progress )
  if (not doMore) then
    break
  end
  TNno=LineFTab.ReturnValue(LineTNode,arc.clone)
  FNno=LineFTab.ReturnValue(LineFNode,arc.clone)
  test=TNDict.Get(TNno)
  if (test<>nil) then
    NewList=test.Add(arc.clone)
  else
    NewList={arc.clone}
  end
  TNDict.Set(TNno,NewList)
  FNDict.Set(FNno,arc.clone)
end
av.SetStatus(100)
'av.ClearStatus

'**locating the points along the arcs and building the point and arc
dictionaries
ProblemList={}
av.ShowStopButton
Total=PointFTab.GetNumRecords
av.ShowMsg("Building Arc Dictionary")
For each Pt in PointFTab
  progress = (Pt.clone/Total) * 100
  doMore = av.SetStatus( progress )
  if (not doMore) then
    break
  end
  AbsLoc=PointFTab.ReturnValue(PointShape,Pt)    'the absolute
coordinates of the point (x,y)
  ArcList=LineThm.FindByPoint(AbsLoc)                  'should be a list
of one arc
  If (ArcList.Count<1) then
    ProblemList.add(Pt)
    continue                                          'error--point is not
on a line
  else
    pcnt=101
    for each ArcID in ArcList                    'gets the feature(s)
returned by the "findbypoint" request
      ArcShape=LineFTab.ReturnValue(LineShape,ArcID)
      PcntDist=ArcShape.PointPosition(AbsLoc)
      if (PcntDist<pcnt) then        'if a point snaps to a node, this
loop will associate it with the
        pcnt=pcntDist              'most downstream arc coming from the
node
        chosen=ArcID.clone
      end
    end  'for each ArcID in ArcList
    LineIDVal=LineFTab.ReturnValue(LineID,chosen)
```

```
    PointFTab.SetValue(AscArc,Pt,LineIDVal)
    PointFTab.SetValue(PtPcnt,Pt,pcnt)
    test=PointDict.Get(chosen)
    if (test<>nil) then
      NewList=test.Add(Pt.clone)
    else
      NewList={Pt.clone}
    end
    PointDict.Set(chosen,NewList)
    ArcDict.Set(Pt.Clone,chosen)
  end   'if (List.Count<1)
end    'for each point in pointftab
av.SetStatus(100)

'**checking over the PointDict
'ListofKeys={}
'ListofKeys=PointDict.ReturnKeys
'For each Key in ListofKeys
'  Val=PointDict.get(Key)
'  MsgBox.ListAsString(Val,"Values in"++Key.AsString++"from
PointDict.","Debug")
'end

'**setting up for the next loop
WorkArc=TheBitmap.GetNextSet(-1)              'finding the outlet and
setting variables
WorkList={}
WorkList.Add(WorkArc)
LineFTab.SetValue(LnDSDist,WorkArc,0)

'**Loop for working with arcs to set LnDSDist**
av.ShowStopButton
Total=LineFTab.GetNumRecords
av.ShowMsg("Building DSDist Field")
counter = 0
While (WorkList.IsEmpty.Not)
  counter=counter + 1
  progress = (counter/Total) * 100
  doMore = av.SetStatus( progress )
  if (not doMore) then
    break
  end
  WorkArcShape=LineFTab.ReturnValue(LineShape,WorkArc)
  Length=WorkArcShape.ReturnLength                  'the length of
workarc
  FlowDist=LineFTab.ReturnValue(LnDSDist,WorkArc)        'the flow
distance from the downstream end of workarc to the outlet
  TotDist=Length + FlowDist                     'total distance from
head of workarc to outlet
  FNVal=LineFTab.ReturnValue(LineFNode,WorkArc)  'value of workarc's
from node
  UpsList=TNDict.Get(FNVal)                      'List of arcs that flow
into workarc
  if (UpsList <> nil) then
    For each Upstream in UpsList
      LineFTab.SetValue(LnDSDist,Upstream,TotDist)  'Sets the dist from
outlet of stream to outlent of network
```

```
    end
    WorkList=WorkList+UpsList
  end
  WorkList.RemoveObj(WorkArc)
  if (WorkList.isempty) then
    break
  else
    WorkArc=WorkList.Get(0)
  end
End   'While (WorkList.IsEmpty.Not)
av.SetStatus(100)
av.ClearStatus

'**Loop for seeking downstream points and writing attributes to the
table**
PtList={}
CompList={}
av.ShowStopButton
av.ShowMsg("Attributing Points with DS Properties")
Total=PointFTab.GetNumRecords
For each Pt in PointFTab
  progress = (Pt.clone/Total) * 100
  doMore = av.SetStatus( progress )
  if (not doMore) then
    break
  end
  PtShape=PointFTab.ReturnValue(PointShape, Pt)
  ArcID=ArcDict.Get(Pt.Clone)
  PtList=PointDict.Get(ArcID)
  Found=-2                              'setting control variable for
while loop
  If (PtList.Count > 1) then            'downstream point may be on the
same arc
    PtListClone=PtList.deepClone
    PtListClone.RemoveObj(Pt)           'keeps Pt from being compared to
itself
    BaseVal=PointFTab.ReturnValue(PtPcnt,Pt)   'the location of Pt in %
of arc length
    Next=100
    For Each i in PtListClone
      PctVal=PointFTab.ReturnValue(PtPcnt,i)
      If ((PctVal>BaseVal)and(PctVal<Next)) then
        Next=PctVal
        Found=i.clone
      end
    end   'For Each i in PtListClone
  End  'If (PtList.Count>1)
  TNVal=LineFTab.ReturnValue(LineTNode,ArcID)
  While (Found < -1)                    'loop for when downstream point
is on next arc
    DSArc=FNDict.Get(TNVal)
    if (DSArc=nil) then          'no more downstream arcs--outlet
      Found = -1
    else
      CompList=PointDict.Get(DSArc)   'List of points on next downstream
arc
      if (CompList=nil) then        'no points on this arc
```

```
        TNVal=LineFTab.ReturnValue(LineTNode,DSArc)
        continue
      else
        Next=100
        For each PtRec in CompList
          PctVal=PointFTab.ReturnValue(PtPcnt,PtRec)   'the location of
PtRec in % of arc length
          if (PctVal<=Next) then
            Next=PctVal
            Found=PtRec.clone
          end
        End     'For each PtRec in CompList
      End    'if (CompList=nil)
    End   'if (DSArc=nil) then
  End  'While (Found<-1)
  If (Found<0) then
    FoundID=0
  else
    FoundID=PointFTab.ReturnValue(PointID,Found)
  End    'if found<0
  ArcShape=LineFTab.ReturnValue(LineShape,ArcID)
  PointFTab.SetValue(DSPoint,Pt,FoundID)
  Flowdist=LineFTab.ReturnValue(LnDSDist,ArcID)
  ThisPcnt=PointFTab.ReturnValue(PtPcnt,Pt)
  ThisPiece=(100-ThisPcnt)/100 * ArcShape.ReturnLength
  Flowdist=Flowdist + Thispiece
  PointFTab.SetValue(PtDSDist,Pt,FlowDist)
End  'For Each pt in pointftab
av.SetStatus(100)
av.ClearStatus
PointFTab.SelectByShapes(problemlist,#VTAB_SELTYPE_NEW)
LineFTab.SetEditable(False)
PointFTab.SetEditable(False)
Exit
```

VirNetBuilder

```
'*************************************************************************
*******
'Script:  VirNetBldr
'Requirements:  A point theme with fields DSDist, and DSPoint such as is
'               created with my DownstreamPoints Script.
'Results:  A theme of the virtual network (points of interest and their
connection
'             to each other.)
'Input:  Activate the tool and then click on the outlet reach of the
line theme.
'Type:  Button or Tool (recommend Button)
'Author:  Kim Davis  6-24-99
'*************************************************************************
********
sMsg="Virtual Network Builder"
```

```
’** Selecting the point theme
TheProject=av.GetProject
TheView=av.getActiveDoc
TheThemes=theView.GetThemes
TheDisplay=TheView.GetDisplay

if(TheThemes.count=0)then
  Msgbox.Error("No themes were found in View:"++TheView.GetName, sMsg)
  Exit
end
if (TheThemes.count > 1) then
  PointThm=Msgbox.ChoiceAsString(TheThemes, "Select a point theme.",
sMsg)
  if(PointThm=nil)then
    exit
  end
else
  PointThm=TheThemes.Get(0)
end
if(PointThm.is(FTHEME))then
  PointFtab=PointThm.getFtab
  TheClassName=PointFTab.GetShapeClass.GetClassName
  if((TheClassName = "point").Not) then
    Msgbox.Error("Selected theme is not a point theme", sMsg)
    Exit
  end
else
  Msgbox.Error("Selected theme is not a point theme", sMsg)
  Exit
end

’** Getting the PointID field that is the source of DSPoint
PointFields=PointFTab.GetFields
PointID=MsgBox.Choice(PointFields, "Select the ID Field
from"++PointThm.GetName++"whose values correspond to those in the
DSPoint field.", sMsg)
if (PointID=nil) then
  exit
end

’**Setting up Field Items
PointFtab.SetEditable(true)
PointShape=PointFtab.FindField("Shape")
DSPoint=PointFtab.FindField("Dspoint")       ’the field storing the id
of the next downstream point
DSDist=PointFtab.FindField("Dsdist")

If ((DSPoint=nil)or(DSDist=nil)) then
  MsgBox.Error("DSPoint or DSDist fields not found
in"++PointThm.GetName,sMsg)
End

’**making the new ftab
LineID=Field.Make("ID",PointID.GetType,PointID.GetWidth,PointID.GetPreci
sion)
LineLen=Field.Make("ActLength",#FIELD_FLOAT,12,2)
```

```
myDir=TheProject.GetWorkDir.AsString
myFileName=Filename.Merge(myDir,"Virnet.dbf")
myFTab = FTab.MakeNew( myFileName, POLYLINE )
myFTab.AddFields({LineID,LineLen})
LineShape=myFTab.FindField("Shape")

'**creating a list by which to look up index values from DSPoint Values
av.ShowStopButton
Total=PointFTab.GetNumRecords
av.ShowMsg("Building Point List")
DSDict=Dictionary.Make(Total)
for each Item in PointFTab
  progress = (Item.clone/Total) * 100
  doMore = av.SetStatus( progress )
  if (not doMore) then
    break
  end
  DSID=PointFTab.ReturnValue(PointID,Item)
  DSDict.Add(DSID,Item.Clone)
end

'**Making the network
av.ShowStopButton
Total=PointFTab.GetNumRecords
av.ShowMsg("Building Virtual Network")
for each Item in PointFTab
  progress = (Item.clone/Total) * 100
  doMore = av.SetStatus( progress )
  if (NOT doMore) then
    break
  end
  DSPtID=PointFTab.ReturnValue(DSPoint,Item)
  if (DSPtID <> 0) then          'DSPtID=0 for outlet points
    USShp=PointFTab.ReturnValue(PointShape,Item)
    USPtID=PointFTab.ReturnValue(PointID,Item)
    DSPtIndex=DSDict.Get(DSPtID)
    DSShp=PointFTab.ReturnValue(PointShape,DSPtIndex)
    ln=(Line.Make(USShp,DSShp)).AsPolyLine

'MsgBox.Info("USPt="++USShp.AsString+nl+"DSPt="++DSShp.AsString+nl+"l="+
+l.AsString,"Debug")
    USD=PointFTab.ReturnValue(DSDist,Item)
    DSD=PointFTab.ReturnValue(DSDist,DSPtIndex)
    ActDist=USD-DSD
    FIndex=myFTab.AddRecord
    myFTab.SetValue(LineShape,FIndex,ln)
    myFTab.SetValue(LineID,FIndex,USPtID.clone)
    myFTab.SetValue(LineLen,FIndex,ActDist)
  end   'if(DSPtID <> 0)
end    'for each item in PointFTab
av.ClearStatus
myFTab.setEditable(FALSE)
myFTheme=FTheme.Make(myFTab)
TheView.AddTheme(myFTheme)
```