

Automating Contour-Based Route Projection for
Preliminary Forest Road Designs using GIS

Luke W. Rogers

A thesis submitted in partial fulfillment of the
requirements for the degree of

Master of Science

University of Washington

2005

Program Authorized to Offer Degree:

College of Forest Resources

University of Washington
Graduate School

This is to certify that I have examined this copy of a master's thesis by

Luke W. Rogers

and have found that it is complete and satisfactory in all respects,
and that any and all revisions required by the final
examining committee have been made.

Committee Members:

Peter Schiess

Gerard Schreuder

Robert McGaughey

Date: _____

In presenting this thesis in partial fulfillment of the requirements for a master's degree at the University of Washington, I agree that the Library shall make its copies freely available for inspection. I further agree that extensive copying of this thesis is allowable only for scholarly purposes, consistent with "fair use" as prescribed in the U.S. Copyright Law. Any other reproduction for any purposes or by any means shall not be allowed without my written permission.

Signature _____

Date _____

University of Washington

Abstract

Automating Contour-Based Route Projection for
Preliminary Forest Road Designs using GIS

Luke W. Rogers

Chair of the Supervisory Committee:

Professor Peter Schiess

Management and Engineering Division, College of Forest Resources

By evaluating alternative routes in the office using a pegging routine, days or even weeks can be saved of valuable field time and ultimately, a better design can emerge. Initial road design in forested landscapes often includes pegging roads on large-scale contour maps with dividers and an engineer's scale. An automated GIS based road-pegging tool (PEGGER) was developed to assist in initial road planning by automating the road pegging process. PEGGER is an extension for the commonly available GIS software ArcView®. PEGGER imports topography as digital contours. The user identifies the origin of the new road, clicks in the direction they want to go and PEGGER automatically pegs in road at a specified grade. Through the use of PEGGER, many alternatives can be quickly analyzed for alignment, slope stability, grades and construction cost using standard GIS functionality. The resulting cuts and fills are then displayed in ROADVIEW, a road visualization package for ArcView®. This paper looks at the algorithm used, evaluates it's usefulness in an operations planning environment and suggests additional methods which might be incorporated into PEGGER to further assist the forest engineer.

Table of Content

Preface.....	iii
Introduction.....	1
Background.....	2
The Importance of Road Design.....	3
The Importance of Visualization	3
Existing Road Design Models	4
Existing Forest Visualization Software	8
Objectives	9
Automate Manual Processes.....	9
“Route Projection” or “Pegging”	10
Utilize High-Resolution Topographic Models	10
Functional Requirements	12
Methods.....	14
ArcView.....	14
Pegging	15
Analytical Description	19
Survey	22
Visualization	24
Analytical Description	25
Discussion	27
Ease of Use	27
Proliferation of Software	27
Applications to Management	28
Limitations	30
Conclusions.....	33
Bibliography	34
Appendix A – PEGGER Software Manual.....	39
Appendix B – PEGGER Avenue™ Code.....	54

List of Figures

Figure 1 - A Daratech study summarized the top nine GIS firms' market share based on worldwide GIS revenue (software only) in 2001.....	14
Figure 2 - The simple PEGGER user interface in ArcView GIS 3.	17
Figure 3 – The simple PEGGER setup dialog where users specify the contour theme, elevation attribute, contour interval, and attribute preferences.....	18
Figure 4 - The simple PEGGER toolbar where users can change the grade of the desired road segments and the name of the road.	18
Figure 5 - Flow chart of the PEGGER design and decision process for route location on a vector based contour dataset.	20
Figure 6 - Locating a new route with PEGGER.	21
Figure 7 - The PEGGER survey dialog where users can specify survey parameters to either match field procedures for comparison purposes, or maximize topographic input into RoadEng with the densify function.	22
Figure 8 - The PEGGER survey function allows users to generate survey data for RoadEng so that it mimics field procedures or so that it maximizes topographic input. Three different survey densities are shown above with the one of the left most closely representing typical field survey methods.	23
Figure 9 - A PEGGER surveyed road in RoadEng. The digital survey can be imported into RoadEng using the Criterion .pol unit survey format.....	24
Figure 10 - ROADVIEW visualization of a route located with PEGGER.	25
Figure 11 - Downloads of the PEGGER software from the Rural Technology Initiative Website from 2002 to 2004 with projected downloads for 2005.....	28
Figure 12 - A dirty and well-used field map produced from a road designed using PEGGER. The road grade and stationing information can be used to locate the proposed road in the field with just the map and a clinometer.....	29
Figure 13 - At shallow grades, PEGGER creates very long segments of road which may not represent the topography properly.	31
Figure 14 - Care must be taken when pegging across stream valleys. Here, the red road was located using PEGGER, however, the road would likely be constructed more like the green road. This difference in road length can significantly affect the grade of the road.	32

Preface

As an undergraduate forest engineering student at the University of Washington I had the opportunity to spend two summers working for Washington's largest forest products company. At Weyerhaeuser, I was impressed by the scale and depth of their geographic information system, but perplexed as to why the field foresters and engineers had relatively little access to these tools. Any request for information from the GIS specialists came back as hard-copy maps, even though the Company was willing to install desktop GIS software on the engineers' computers.

In my training as a forest engineer I learned how valuable tools like geographic information systems could be in guiding management decisions. I was exposed to many tools including clinometers, relaskopes, laser range-finders, geographic information systems, optimization programs, and much more. I soon became very interested in how all of these tools could work together to provide more informed decisions about how to better manage the forest and the necessary systems that support forest management.

It was while working for those two summers that I realized there was a large disconnect between university research tools, optimization routines, and the needs of the field forester. This could not have been more apparent than in the manner in which roads were initially designed and subsequently located in the field. Typically, the engineer would look at a 1:4800 scale map (printed from a GIS), briefly glance at a stereo photo set, and then head to the field to locate a route in a trial and error method. While these engineers were very good at doing this, based on their many years of experience, I thought there might be a better way. I thought it would be valuable to take advantage of desktop GIS software and provide some simple tools to help the forest engineer with initial route location. With that concept in mind, and a generous funding offer from the newly created UW Rural Technology Initiative, I started a masters program to design a tool that I would call PEGGER.

Acknowledgements*

Dedication*

Introduction

It can be assumed that since the dawn of forest management, forest road engineers have sought means to more efficiently analyze alternative road locations. Where survey teams once blanketed the forest landscape to locate prime route locations, are found only solitary forest engineers verifying and flagging their chosen road location. This shift in forest road design methodologies likely began with the first contour maps and continues to be transformed today by remote sensing technologies like Landsat and LiDAR and the proliferation of desktop Geographic Information System (GIS) products. It is now possible for a forest manager to analyze many different road location alternatives over a large geographic area in a minimal amount of time. Time consuming field work has been reduced to verification of the chosen alternative and marking it in the field.

This research looks at the differences in forest road planning techniques and the existing software products that have been developed to assist in forest road location. A computer program is presented that automates initial forest road location through the use of a Geographic Information System and digital terrain data. Using PEGGER, forest planners can quickly analyze many road location alternatives and, by taking advantage of standard GIS functionality, evaluate environmental and economic opportunities.

A companion program, ROADVIEW, has been developed to assist in communicating forest road design concepts to those outside of the forestry profession. Using ROADVIEW, forest planners can quickly show the topographic modifications of a planned forest road and evaluate visual impacts associated with alternative road locations.

Background

There are three broad categories of planning in forest management, strategic, tactical and operational (Sedjo 1987; Kent, Bare et al. 1991; Schiess and O'Brien 1995; Boyland 2003). Information exchange between these three tiers of forest management planning is critical in developing a management plan. Strategic planning looks at large areas with aggregated data typically over long time horizons. A large forest products company may have one strategic plan for their entire forest land-base with the goal of providing consistent shareholder returns year after year. Calculating sustained yield occurs at the strategic planning level.

Tactical planning encompasses a diverse range of activities and geographic areas but generally is considered to be associated with a specific management block or ownership. Harvest targets are usually handed down from the strategic level to the tactical level without regard for the specifics of where the harvest volume will come from. It is the tactical plan that harbors more specific information on stand volumes, road systems, and management priorities.

Operational plans are derived from information handed down from the tactical level and then composed into the specific details of when, where, and how. The tactical plan may indicate that a stand is ready for harvest and it is the job of the operations planner to develop a specific harvest plan with road systems, harvest techniques, and conservation priorities.

At all three levels of the forest management decision making process forest road planning is required. At the strategic level, planners need to know roughly how much road will be constructed, maintained or abandoned each year for proper cost accounting and depreciation. At the tactical level, planners need to know if specific areas of the forest are reachable by road, which segments of road will need maintenance, where new roads will

be constructed and how much it will cost. At the operational level, planners need to know site-specific information to properly build, maintain, or abandon roads and develop detailed plans that can be implemented by construction personnel, loggers, and foresters.

Recognizing that office-designed preliminary route locations can save forest managers time and money and with the advent of computers, researchers and forest management consultants have produced myriad software packages to assist in the strategic, operational and tactical aspects of forest road planning.

The Importance of Road Design

The road design and construction process is the most expensive and time consuming portion of a harvest operations plan (Epstein, Sessions et al. 2001). It is not surprising then, that so many road design tools and optimization models have been built to assist with the development of transportation plans. It has been demonstrated that “judicious and appropriate use of forest engineering tools can enable the designer to develop a far superior harvest plan (Schiess and O'Brien 1995).” Part of the design process then must be to “evaluate a sufficient number of alternative routes to locate a final route” that meets operational and environmental targets (Akay, Karas et al. 2004).

Recognizing that office-designed preliminary route locations can save forest managers time and money and with the advent of computers, researchers and forest management consultants have produced myriad software packages to assist in the strategic, operational and tactical aspects of forest road planning.

The Importance of Visualization

According to Barry (1998) one of the factors that have driven the investment in realistic forest visualization tools is public scrutiny and tighter guidelines that mandate more

accurate planning. Barry goes on to say that “as a result, visual impact assessments--and their "before and after" simulations--have become key elements of the submission and approval process [of resource management plans].” McGaughey has identified the importance of visual design in harvest operations planning: “These [visualization] applications were designed to meet the needs of visual management specialists responsible for designing the shape of harvest units and scheduling harvest activities to minimize the overall visual impact on forested landscapes (McGaughey and Twito 1988).”

Given the increased level of public scrutiny associated with timber resource management in recent years, visualization tools are becoming invaluable in communicating forest operations design to the laity. Any road design tool to be developed must include a visualization component to communicate a proposed route location and analyze the aesthetic impact associated with its construction. The availability of free visualization tools like EnVision (McGaughey 2000) and forest management tools like the Landscape Management System (McCarter 2001) that integrate well with each other make it practical to develop a road design tool that can complement those existing products.

Existing Road Design Models

Models have been created for all levels of forest management planning. There are many forest management related models that look at road design, transportation scheduling, harvest scheduling, water quality, wildlife habitat, visual impacts, carbon sequestration, growth prediction, cable yarding, grapple skidding, helicopter logging, and others. This discussion will focus on forest road design and visualization software and where those models fit into the hierarchical structure of forest management planning.

There are very few strategic forest road planning tools. However, the Forest Service developed the linear programming model FORPLAN (1985) to address the multiple use

problem on the Nation's Federal lands. While FORPLAN was useful for analyzing broad National scale issues, it did not address roads directly and a critical evaluation by Kent et al. (1991) recommended that FORPLAN be redesigned or replaced by "smaller, hierarchically-based, easier-to-interpret models." Anderson and Nelson (2004) have created a vector-based road network model that is "specifically concerned with creating road networks that are suitable for strategic planning." They state in their work that an operational plan must be devised (from the strategic plan) and validated before any forest road is approved for construction. A model developed in Chile called PLANEX is able to strategically plan landing locations, harvest settings and access routes on areas as large as ten thousand hectares relying on coarse raster data to locate the transportation network (Epstein, Sessions et al. 2001).

At the tactical level, most existing models are focused around optimization of harvest scheduling and road networks including SNAP (Sessions and Sessions 1988; Sessions and Sessions 1992), NETWORK (Sessions 1987), and the University of Washington Timber Harvest Planning System (UWTHPS) (Schiess and O'Brien 1995). These models are complex and typically unused by the majority of forest operations planners. The necessary datasets, lack of site-specific control, and time required to setup these systems has left them without many users.

At the operational level many road design packages have been developed likely starting in the early 1970's with programs that ran only on mainframe computers and had no user interface (Kobayashi 1973). In 1974 what is thought to be the first forest road design software program for the "modern desk-top calculator" was introduced (Burke 1974). An attached digitizer was used to input topographic information into the program which was then processed by analytical routines. The results were then displayed on a plotter for evaluation and adjustment in an iterative process.

Since 1974 many others have introduced road design software packages for the desktop computer. Of these road design packages (RoadEng, AutoCAD, ROADPAC, F.L.R.D.S., TRACER, ROUTES...) only one has given the user the ability to quickly look at alternative road locations at varying scales, ROUTES (Reutebuch 1988). Traditional road design software relies on survey data collected in the field to generate terrain models and very detailed engineered road location and construction plans. Others have taken a more holistic approach and looked at optimization of road locations for a particular set of topographical, environmental or economical constraints (Thompson 1988; Cha, Nako et al. 1991; Xu 1996). All these programs have relied on a high degree of training on the part of the user and few of the non-commercial packages have matured into an easy to use software package.

ROUTES was developed to automate the road pegging process. Using a large-scale contour map (1in = 400ft) and a digitizer, the user could digitize the contours and use the digitizer puck to locate the road. While the user interface was primitive consisting of high and low pitch beeps from the digitizer puck to signal that the user was “on-grade”, the program worked well and kept track of such things as grade, road length and stationing. ROUTES reliance on a digitizer, its HP 9000 code base and the general lack of a graphical user interface (GUI) left the program without many users.

TRACER is a PC-based stand-alone program for locating forest roads. TRACER takes a linear programming and heuristic approach to locate a vertical alignment with the lowest total costs while conforming to environmental constraints (Akay, Karas et al. 2004). While TRACER is an excellent tool for analyzing many alternatives it does have its limitations. TRACER took 15 minutes to complete an analysis on 55 hectares. It did produce a solution with a 25% cost reduction over another chosen feasible route but may be too “black box” for many forest managers. Reisinger and Davis (1985) suggest in reference to forest planning software that “while these models are often mathematically elegant from a research standpoint, forest industry managers find them operationally

unworkable.” Finding the balance between optimal and workable appears to be the challenge facing software authors.

Both TRACER and ROUTES are useful tools, however as clearly identified by Dürrstein (1992) all road design software packages must include several key components to be most efficient for the end user. According to Durrestein: “Summarizing the existing experiences and demands of the user, computer-aided detailed road planning [software] must fulfill the following conditions:

- suited for standardized hardware (e.g. the widespread Personal Computer);
- modular structure corresponding to traditional planning methods for utilizing the engineer’s experience in planning a road;
- simple handling based on clear screen menus which avoid intensive training in the use of computer systems and increase acceptance by the forest engineer;
- integrated input of usual field data;
- direct interactions between the user and system during the planning procedure;
- data transfer to standard graphic packages (e.g. AUTOCAD).”

It may be more appropriate to identify opportunities to integrate the forest managers experience into forest road planning software rather than attempt to find optimal solutions with “black box” models. Watson and Hill (1983) define a Decision Support System (DSS) as an “interactive system that provides the user with easy access to decision models and data in order to support semi-structured and unstructured decision-making tasks” and suggest that a DSS is essentially an analytic tool to improve the effectiveness in making decisions where the manager’s judgment is still essential. Reisinger further states that “the key to successful implementation of a DSS lies in creating a user-friendly environment in which a manager/planner can interactively obtain answers to practical questions about the large-scale, complex operations they manage.” Thompson (1988) states in reference to his own road spacing model that “the model should be viewed as a tool used to enhance good judgment.”

Existing Forest Visualization Software

Another important consideration in forest road design is the visual impact to the landscape. While current software packages allow you to see where your road will be located and the geometry associated with its construction, they do not show where the built road will be taking land out of production. Integration with existing visualization software is important for communicating design of proposed forest roads. There is the visual impact of the cut and fill slopes and the narrow strip of trees that must be removed within the road right of way. By being able to quickly visualize alternatives and compare the results, aesthetics can be taken into consideration when designing forest roads.

In the early 1970's, around the same time as the first route projection routines were being programmed into mainframe computers, three-dimensional visualization routines were developed. These early 3D perspective terrain tools were primitive, slow, and ran on mainframe computers (for a more detailed discussion of forest visualization software see McGaughey and Twito 1988). It probably wasn't until the introduction of the PC-based Preliminary Logging and Analysis System and its VISUAL and SLOPE components in the late 1980's that visualization became accessible to the forestry community.

Currently, there are a few visualization packages including EnVision, the Stand Visualization System (SVS), the World Construction Set, Visual Nature Studio (VNS), SmartForest, Virtual Forest, and most geographic information systems. Of these packages only EnVision (McGaughey 2000) allows for the landscape wide integration of forest stand data into the visualization, runs on a PC, and is available free of charge.

Objectives

The primary objective of this work is to create a preliminary road design tool that forest operations professionals will use on a regular basis to supplement their existing toolkit of aerial photography, paper based maps, clinometers, compasses, geographic information systems and global positioning (GPS) units. This tool must integrate into existing desktop software products, incorporate the knowledge of the road engineer, have a simple user interface, take little or no training to operate, and be simple and easy to use. The tool must be interactive and empower the road engineer with transparent software logic rather than over-power them with “black box” optimization technologies. The success of a road design tool should not only be measured by the optimization of a particular set of criteria or the cost savings associated with a particular design solution, but by the number of individuals and organizations using the software.

Automate Manual Processes

With the overwhelming popularity of Geographic Information Systems (GIS) in natural resource management it is appropriate to explore opportunities to integrate traditional road design techniques into the GIS. With the availability of free 10-meter digital elevation data for the United States and the continually decreasing cost of LiDAR data it is possible to extend the road pegging technique to include a more detailed analysis. Others (Becker and Jaeger 1992; Dürrstein 1992; Dvorscák and Hrb 1992) have taken this approach and integrated CAD and GIS. However, the availability of GIS software at the time was limited to main frames or large centralized computers. As a result access by engineers and forest land managers was probably infrequent which left these models primarily with governmental and educational users.

Forest engineers increasingly have more access to desktop GIS software. One of the many useful things about GIS and software in general is its ability to automate simple and complex tasks. Identifying opportunities to automate manual processes in the forest

road design process, while incorporating the engineers knowledge, may hold the most promise for a useful tool. Many engineers utilize aerial photography and route projection on a paper map (or pegging) to identify initial route locations. This process is largely repetitive and time consuming and can be efficiently automated within a geographic information system.

“Route Projection” or “Pegging”

Traditional methods for designing a forest road system consisted largely of aerial photo interpretation and field reconnaissance. More recently, forest engineers have used large-scale contour maps to select preliminary routes with dividers, a process known as route projection or “pegging”. According to Pearce (1960), “Route projection is the laying out of a route for a road on a topographic map or aerial photo. The route defines the narrow strip of land within which the field preliminary survey is made.”

Utilizing contour maps, engineers can quickly evaluate multiple route locations in the office and then focus their field work within those areas. In combination with aerial photography, this trial and error method of initial paper based road location has proven itself as a cost effective method for preliminary design and analysis by avoiding expansive field investigations.

Utilize High-Resolution Topographic Models

All route location software relies on topographic models that are commonly referred to as digital elevation models (DEM) or digital terrain models (DTM). Methods for storing the data vary from contour lines, to cell-based raster datasets, to triangulated-irregular-networks or TINs. Most route location software relies on some form the raster data model in which each cell in the data is assigned a particular elevation value. The appropriate cell size to use for forest route location is specific to the particular software application and geography of the area. Many have suggested that a cell size of between 1.0 and 3.0

meters is sufficient for operational route location (Coulter, Chung et al. 2001; Akay, Karas et al. 2004; Krogstad and Schiess 2004).

Digital elevation models can also be stored as contour lines in the GIS. While there is no cell size associated with line features in a GIS, there is the concept of a reference scale. USGS digital elevation models are generated from the 1:24,000 topographic quad sheets that were photogrammetrically derived. Carson and Reutebuch (1997) have shown the limits of using USGS DEM's for forest operations planning. The Washington State Department of Natural Resources has developed contour data products from aerial photography at a scale of 1:4800. It has been shown that those maps were a significant improvement over the USGS product, particularly for forest road location and skyline profile analysis (Schiess and Rogers 1999; Schiess and Rogers 2000; Schiess and Arntzen 2001; Krogstad and Schiess 2004).

While USGS 10 meter digital elevation products and 1:24,000 contours are limited in their use for forest operations planning, Light Detection and Ranging or LiDAR is being recognized as a reliable data source for high-resolution elevation models. Referring to a digital elevation model with a RMSE of 29 cm Pereira (1999) stated that "a [digital elevation model] derived from laser measurements with an average density of 4 points per m² has sufficient quality to represent the terrain relief for the purpose of road planning and design." Forest cover plays a large role in the accuracy of LiDAR generated elevation models. It has been shown however, even under the densest forest canopies that mean errors are less than 0.5 meters (Reutebuch, McGaughey et al. 2003).

In the forest industry and academic institutions LiDAR has been used to test its applicability for aiding forest road design. Some work done on Capitol State Forest in Washington has validated that LiDAR is appropriate for forest road design and can be used to accurately earthwork (Coulter, Chung et al. 2001). Reutebuch states that in

general, the LiDAR DTM was found to be extremely accurate and potentially very useful in forestry (Reutebuch, McGaughey et al. 2003).

LiDAR is one of the fastest growing remote sensing technologies with many different platforms, formats, data products and uses. It is expected that the accuracy of LiDAR generated topographic products will only increase as its use becomes more widespread.

Functional Requirements

This work was funded by the University of Washington College of Forest Resources Rural Technology Initiative program. The mission of the Rural Technology Initiative is to “empower the existing infrastructure to use better technology in rural areas for managing forests for increased product and environmental values in support of local communities.” The mission is essentially to identify relevant technologies at the University level and then translate those technologies into meaningful tools for applied use in rural communities. Staff at the Rural Technology Initiative suggested that any tool that was to be used by Washington’s non-industrial forestland owners and forestry consultants would have to be inexpensive and simple to use in order to be successful as a tool outside of academia.

It was recognized early in the design process that integrating a road design product into an existing software package could be beneficial to both the developer and the users. Utilizing standard functions built into all commercial GIS products as the backbone of a road design tool would allow the developer to focus on the functionality of the tool itself and not the creation of the application necessary to support it. For this reason, it was decided that the tool should be designed as an extension to an existing off-the-shelf GIS software package. The existing GIS software should be relatively inexpensive and available across multiple platforms for maximum success.

As suggested by Dürstein (1992), Schiess (1995), and others, forestry tools should be modular in their design and behave as individual tools in a toolkit so that the experienced forestry professional can select the appropriate device for their particular needs. From a functionality standpoint this means that any tool should be able to communicate well with existing and future tools. Implementing a forest road design tool within a GIS framework ensures that this condition is met as a GIS is designed as a tool for the import, manipulation, display, and export of myriad data formats.

With the importance of visualizations in the communication of forest management plans to the public, it is sensible to enable any new forest road design tool to take advantage of the existing forest visualization tools that are available. The concept then should be to locate a road using the GIS based tool, and then virtually construct that road into the topographic model. The new topographic model can then be visualized in any of the existing forestry visualization software packages or within the GIS itself.

Methods

ArcView

The Environmental Systems Research Institute or ESRI is the largest GIS software producer in the world and owns more than one-third of the GIS software market (Figure 1) (Thrall and Campins 2004). Since 1996 ESRI's ArcView® 3 GIS software has been the most used GIS software in the world with over 500,000 copies sold worldwide. The popularity of the ArcView software makes it the logical choice for the development of a route location program.

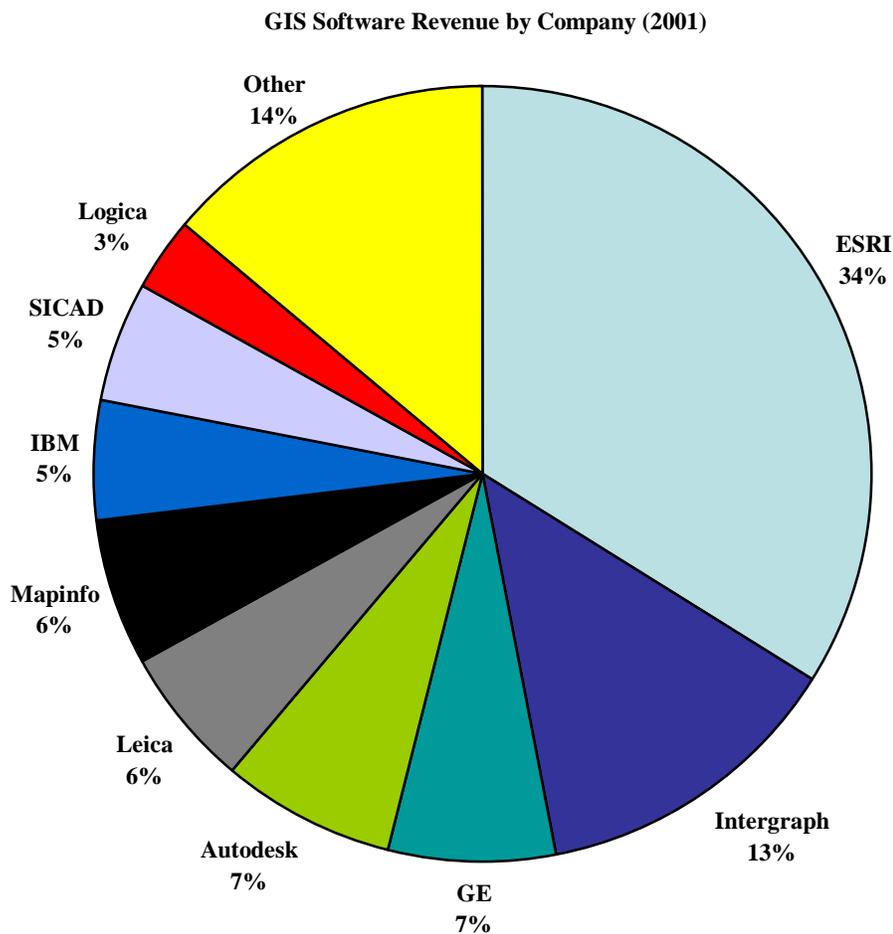


Figure 1 - A Daratech study summarized the top nine GIS firms' market share based on worldwide GIS revenue (software only) in 2001.

ArcView GIS provides the avenue programming language to write add-ons and scripts which can be packaged and installed as extensions. With ArcView it is easy to integrate custom content into the help system to make a professional product which seamlessly integrates with ArcView. If a user knows how to use ArcView they will be able to use an ArcView extension. Another reason to use ArcView and build this as an extension to ArcView rather than as a stand alone product was to take advantage of the data management and display functionality that is designed into any GIS. It also allows for the integration of other tools and datasets inherently. This allowed for more design time to be spent on ease of use and functionality rather than designing a new piece of software from scratch.

Pegging

With the growing availability of LIDAR and IFSAR data, locating roads in the office is becoming a more realistic and practical exercise. Within the GIS framework many tools exist to locate geographic features, examine spatial relationships among natural elements and act as a foundation for a decision support system. Watson and Hill (1983) define a decision support system as an “interactive system that provides the user with easy access to decision models and data in order to support semi structured and unstructured decision making tasks.” It is with the intention of providing an initial decision support system that PEGGER was developed.

PEGGER is an ArcView GIS extension that automates the route projection (“road pegging”) process for use by engineers and forest planners. One of the goals of the PEGGER project was to make the program as usable as possible for as many people as practical. One of the problems with technology is training users to use the software. Forestry professionals responsible for fieldwork have been slow to adopt new technology into their work largely due to the complexity of the software and the time commitment of training. The PEGGER program was designed to avoid these common pitfalls, requiring

no training, minimal setup time and a simplified user interface. Included with the software are detailed help files and a complete tutorial with sample datasets.

PEGGER imports topography as digital contours much like using a paper contour map. Standard tools available within ArcView GIS allow the user to import the contours from Shapefiles, ESRI coverages, AutoCAD dwg and dxf, and Microstation dgn files. In addition to importing data as digital contours, users can use the ArcView Spatial Analyst extension or other publicly available tools to convert USGS digital elevation models or LiDAR elevation models to contours. Contours were chosen as the preferred method of storing topographic surface information primarily because in using a vector based dataset, the user was not required to have either the ArcView Spatial Analyst or 3D Analyst Extensions. In addition, every attempt was made to automate the paper-based route projection process and avoid the perception of a black box technology. By simply automating a manual routine, it is possible that forest engineers and planners will have more confidence in their results.

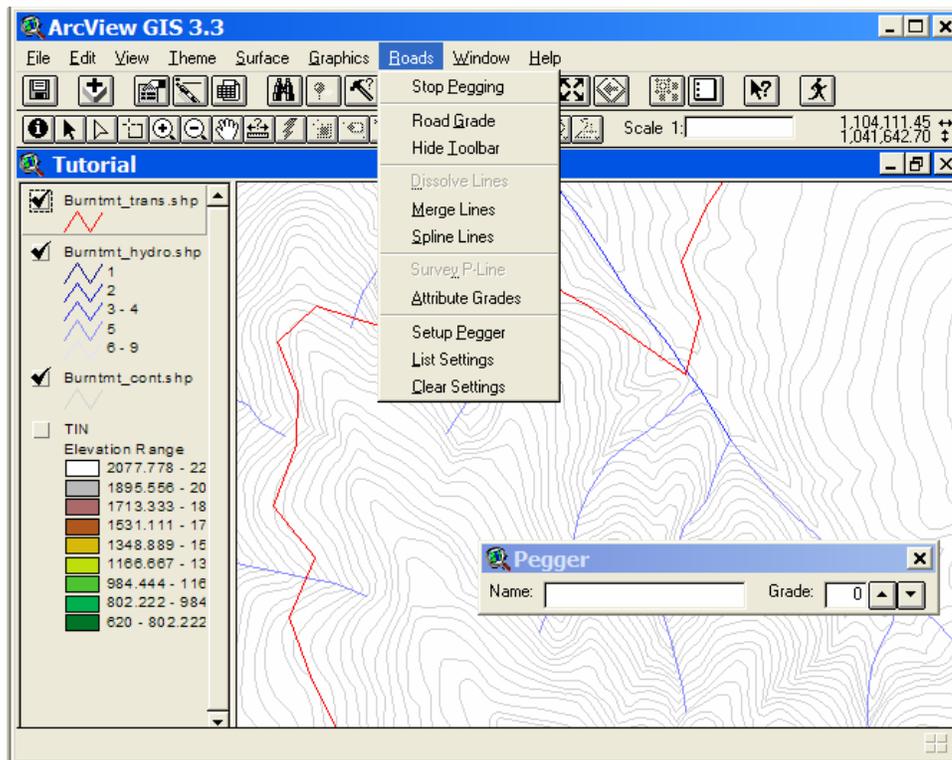


Figure 2 - The simple PEGGER user interface in ArcView GIS 3.

Once digital contours have been imported into ArcView the user must supply a few parameters, the road theme they would like to edit, the contour theme they would like to use as well as confirm the detected contour interval (Figure 3). In addition to the contour and road themes the user can have any number of other layers available in the GIS such as soils, slope classes, streams, wetlands, unstable slopes and property lines. Optionally, the user can maintain attribute information about the grade of the pegged segment and a road name.

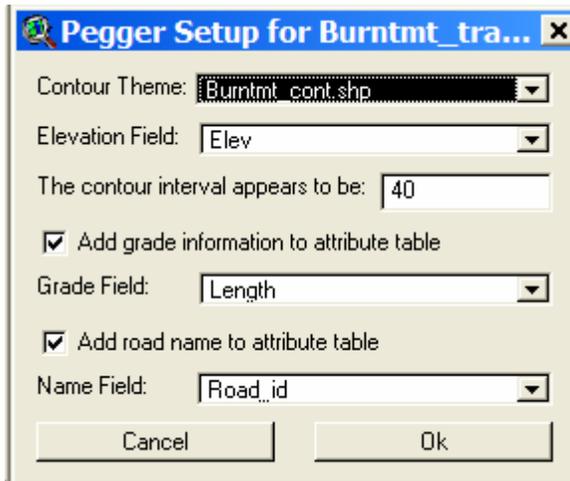


Figure 3 – The simple PEGGER setup dialog where users specify the contour theme, elevation attribute, contour interval, and attribute preferences.

The next step is to locate the desired beginning and/or endpoints of the new road given operational parameters. Using standard tools available in the GIS (ruler and identify) the user can estimate the necessary grade for the road. To start a road the user shift-clicks on the location where they wish to begin and enters the desired grade. To “peg” the road the user only has to click in the general direction they wish to go in order to project the route into the GIS. Successive clicks peg in additional segments of road from contour to contour as fast as the user can press the mouse buttons. Grade changes can be accomplished by using the Roads pull-down menu, using the PEGGER toolbar (Figure 4) or by right clicking the mouse and selecting Increase or Decrease Grade.

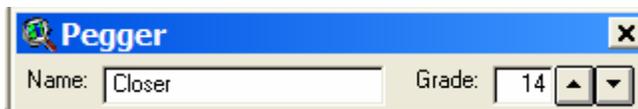


Figure 4 - The simple PEGGER toolbar where users can change the grade of the desired road segments and the name of the road.

If the road fails to reach the desired end point, the previously pegged segments can be quickly deleted and a new grade can be tried. This method of trial and error that used to

mean changing the divider spacing and erasing undesirable segments from the map can now be accomplished in the GIS in a fraction of the time.

Analytical Description

PEGGER works by identifying contour lines that meet a specific set of criteria (Figure 5). Every projected route segment must begin and end on a contour line. To project a segment the user enters a desired grade and PEGGER looks for a point on an adjacent contour line at a distance computed by:

$$d = ci / (g / 100)$$

where d = the distance,

ci = the contour interval, and

g = the desired grade.

NOTE: For pegging on paper maps, the distance would need to be multiplied by the map scale (ie: 1/4800) to get the appropriate divider width.

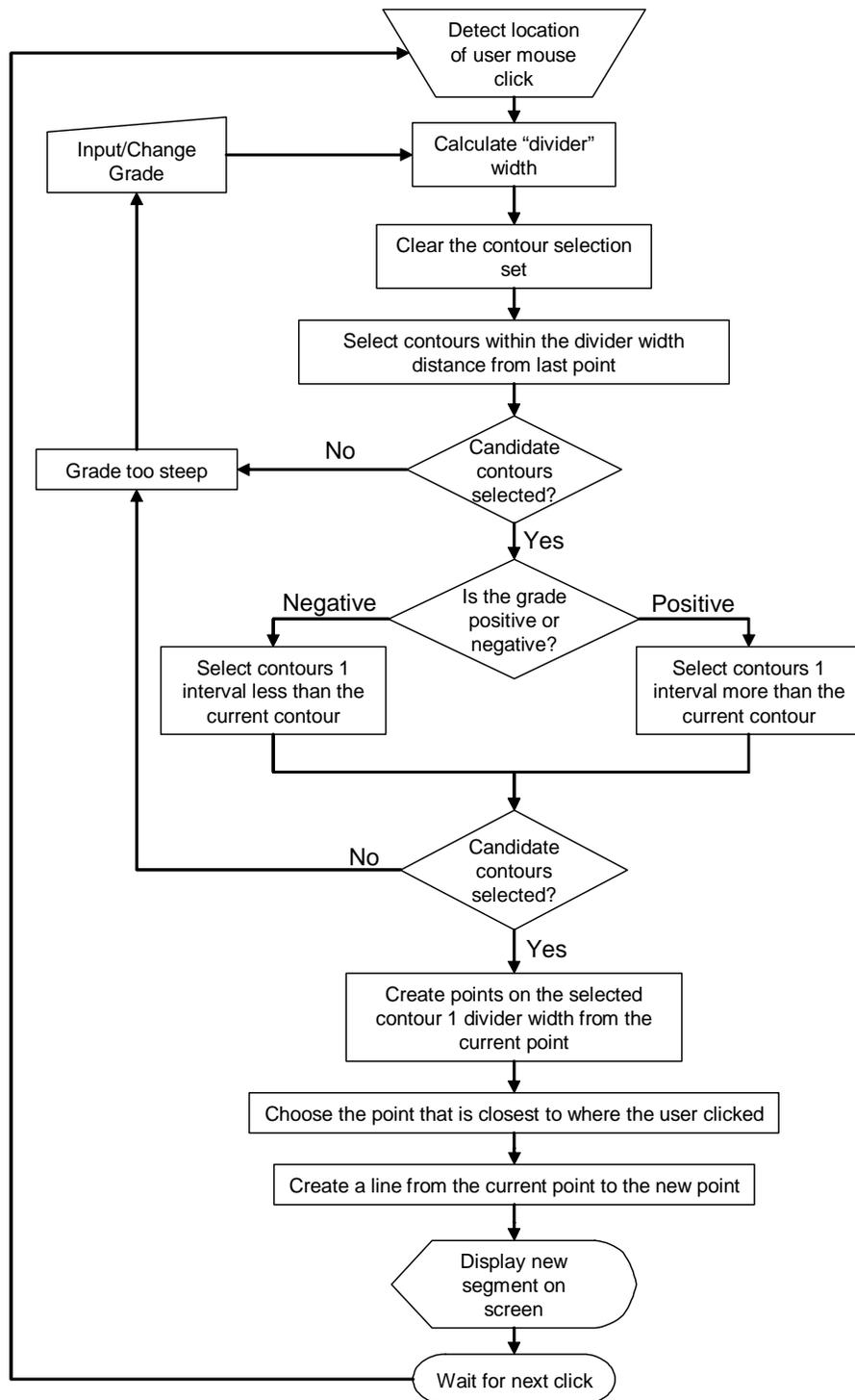


Figure 5 - Flow chart of the PEGGER design and decision process for route location on a vector based contour dataset.

If a point is found, a new route segment is created in the GIS (Figure 6). If a point is not found, the user is notified that the desired grade is not feasible and potential solutions are proposed. Unlike ROUTES, which allowed for a grade tolerance (+/- some tol), PEGGER gives an exact solution in the GIS.

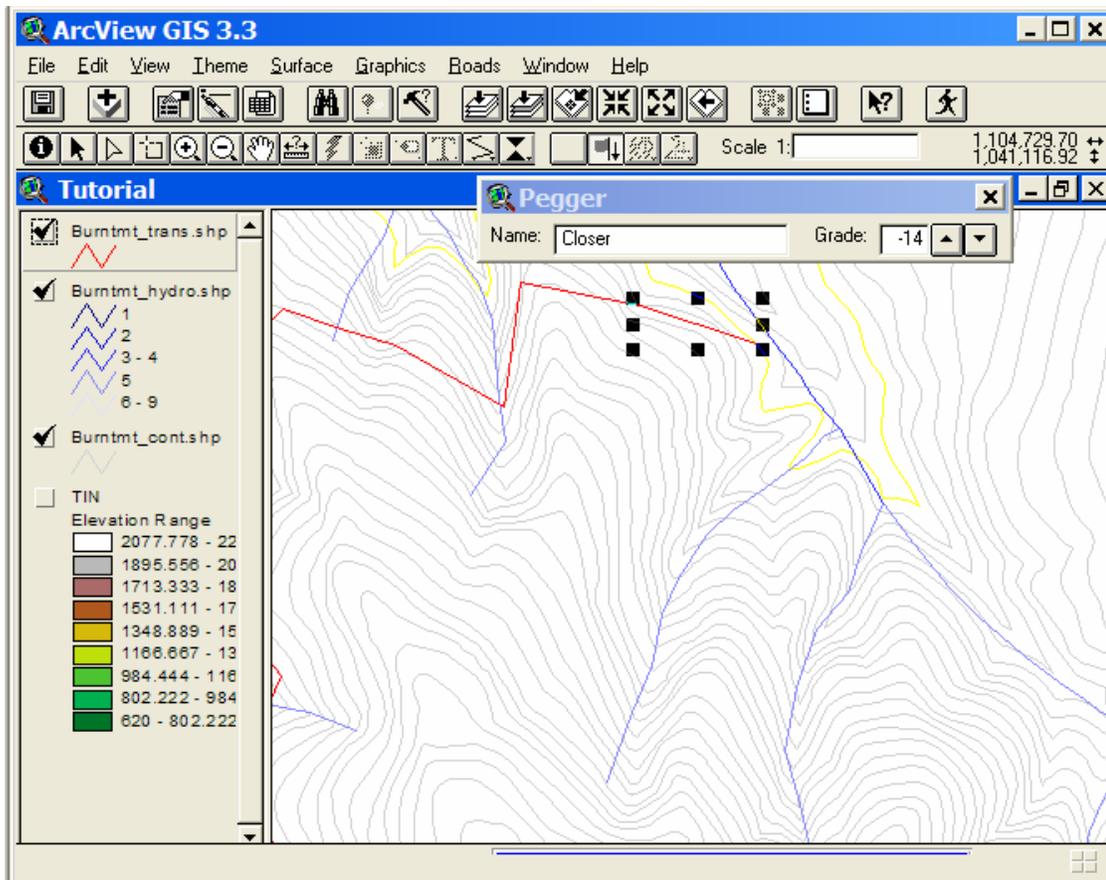


Figure 6 - Locating a new route with PEGGER.

After a desirable route location has been found the user can merge the segments into one long road, dissolve adjacent segments based on a common attribute or spline to smooth sharp corners (much like a finalized design). An attempt was made to produce tangents and curves from the initial design but ArcView's lack of a true curve feature type made the possibility impractical.

Survey

PEGGER is designed as a preliminary route location tool that can inform a more detailed analysis. Using the ArcView extensions Spatial Analyst or 3D Analyst, users can digitally survey a preliminary location and export the survey to RoadEng. The survey technique used by PEGGER closely mimics the methods used by field crews when surveying a P-line with a Criterion laser range finder. Survey parameters such as the elevation model to use, the number and density of side-shots, and the distance between turning-points can be specified in the simple survey dialog shown in Figure 7. An option to densify the survey and gather additional topography ground points as shown in Figure 8, allows the user to replicate field procedures or maximize topographic input into RoadEng. This option allows direct comparison between field based and PEGGER digital surveys for accuracy assessments and research.

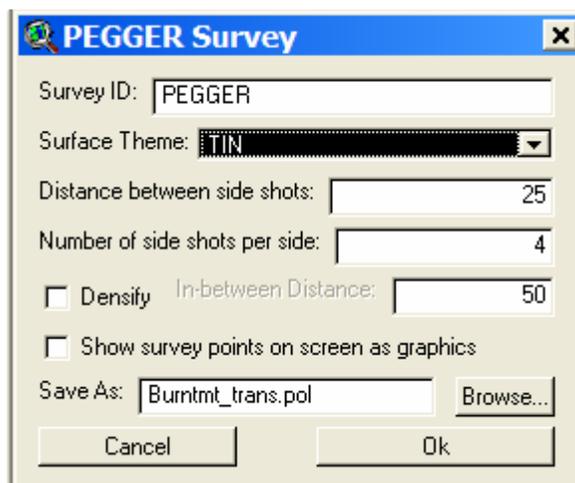


Figure 7 - The PEGGER survey dialog where users can specify survey parameters to either match field procedures for comparison purposes, or maximize topographic input into RoadEng with the densify function.

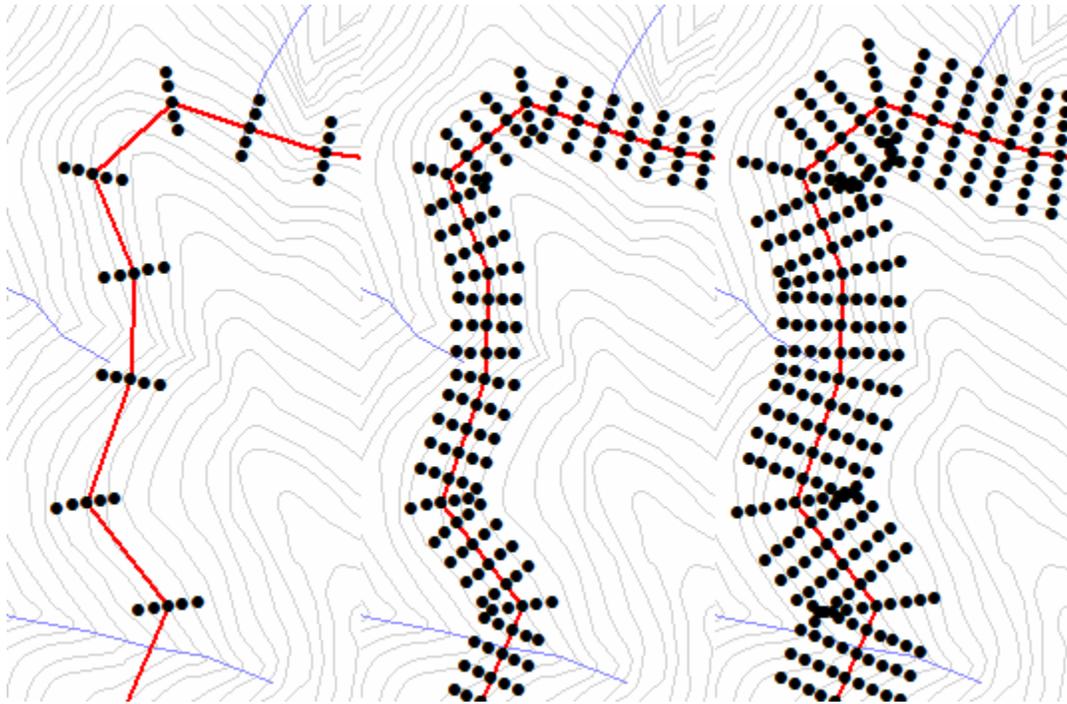


Figure 8 - The PEGGER survey function allows users to generate survey data for RoadEng so that it mimics field procedures or so that it maximizes topographic input. Three different survey densities are shown above with the one of the left most closely representing typical field survey methods.

PEGGER is designed as a tool to quickly evaluate many alternative routes. PEGGER was not designed to optimize or even suggest optimal route locations. However, it has been shown that once a route location has been chosen RoadEng can be used to produce an optimal design (Herald 2002). RoadEng can produce a final road design that considers earth work, horizontal alignment, vertical alignment, super-elevation, and materials. By quickly pegging multiple routes and analyzing them in RoadEng (Figure 9), a preferred alternative can be selected based on environmental, economic, or visual concerns.

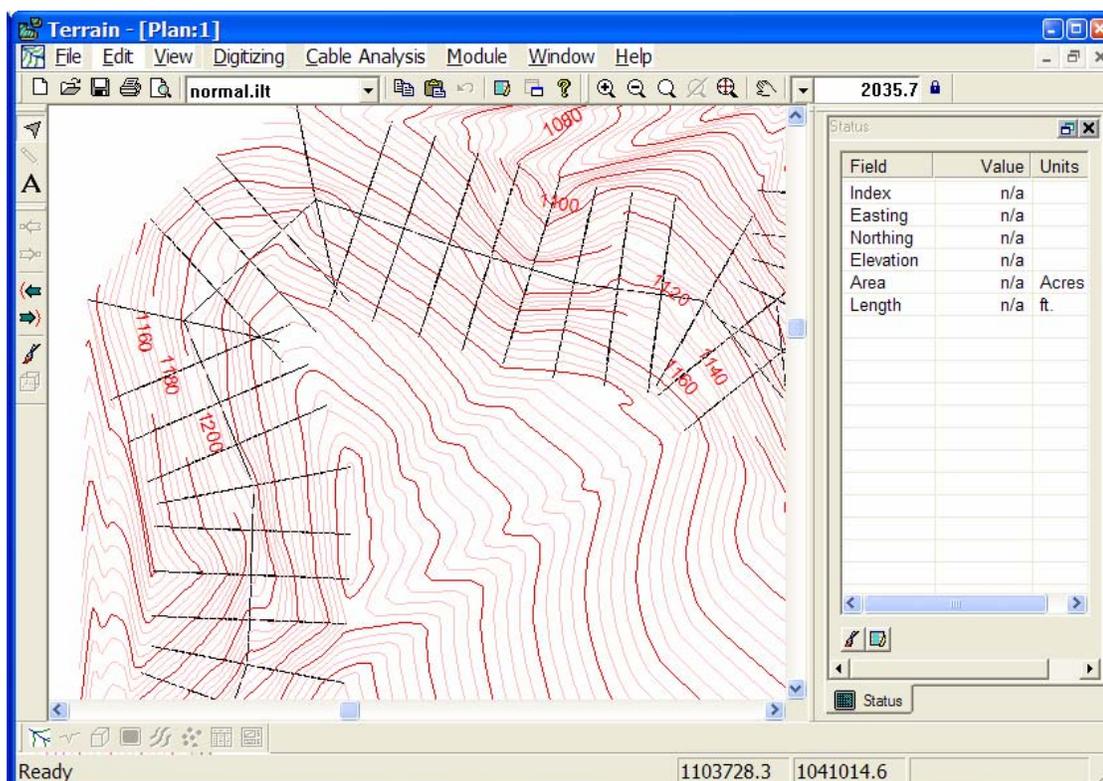


Figure 9 - A PEGGER surveyed road in RoadEng. The digital survey can be imported into RoadEng using the Criterion .pol unit survey format.

Visualization

Complementing PEGGER is a companion program ROADVIEW that takes the preliminary route location generated by PEGGER and creates an approximate 3-dimensional model of the road's cuts, fills and running surface. Using the 3-D model and a visualization program such as EnVision, professionals can look at the road as it might be constructed and effectively communicate with non-forest professionals regarding scenic and environmental impacts. Utilizing the Landscape Management System (LMS) in combination with EnVision can produce realistic representations of the visual impacts associated with right-of-way clearing and road construction.



Figure 10 - ROADVIEW visualization of a route located with PEGGER.

Analytical Description

ROADVIEW works by constructing 4 surfaces based on the location of an existing or pegged road. First a grid is constructed that represents the distance from the road centerline. Second, a road surface grid is constructed from the road using elevation data from the digital elevation model. Third, both cut and fill grids are created for the entire length of the road. Using these 4 grids, a cell-by-cell analysis chooses the appropriate surface based on the order in which those grids overlay. The value of each cell in the visualization grid can be calculated by the following pseudo-code:

```
Select Case
  Case ([DISTANCE TO ROAD] < w)
    [ROAD GRID]
  Case ([CUTSLOPE] < [DEM])
    [CUTSLOPE]
  Case ([FILLSLOPE] > [DEM])
    [FILLSLOPE]
  Else
    [DEM]
End Select
```

where w = road width,

[####] = a grid

Currently, ROADVIEW does not adjust the road location or vertical alignment to balance cuts and fills or construct a full-bench segment. ROADVIEW simply visualizes the road as if it were built exactly as represented in the GIS. While the method is fairly crude, it quickly gives the user an idea of how a particular road will look when constructed. Any future work on ROADVIEW should attempt to visualize roads based on templates for balanced cut/fill and full-bench sections in different side-slope conditions.

Discussion

Ease of Use

To ensure the software was intuitive a usability test was conducted. All users had some familiarity with ArcView 3 GIS. Users were asked to download and install the PEGGER ArcView extension and then follow the included tutorial to learn how to use the software. Monitoring of the users behavior and suggestions made directly by the users resulted in a streamlined and improved design. Since PEGGER's introduction in 2001 many comments have been received from users. Most have commented on the programs ease of installation, well documented tutorial, and simple user interface (personal communication with Dudek 2004; personal communication with Khan 2004; personal communication with Romberg 2005).

There were two main issues identified from users. First, they were looking for the installed program under the Microsoft Windows start menu even though ArcView extensions are not accessed that way. In order to address this behavior there was a program folder set up to appear in the start menu for easy access to the program after installation. Second, the help documentation did not include information for PEGGER because the ArcView help was separate so the PEGGER help documentation was integrated with the ArcView help documentations files. These simple changes to the installation and user interface have made the program much more intuitive and user-friendly.

Proliferation of Software

Downloads

Since PEGGER was first published as an ArcGIS extension in 2001 many users from all over the globe have downloaded and successfully used it. As of April 2005 there have

been over 670 downloads of the software from the Rural Technology Initiative website at <http://www.ruraltech.org/tools/pegger>.

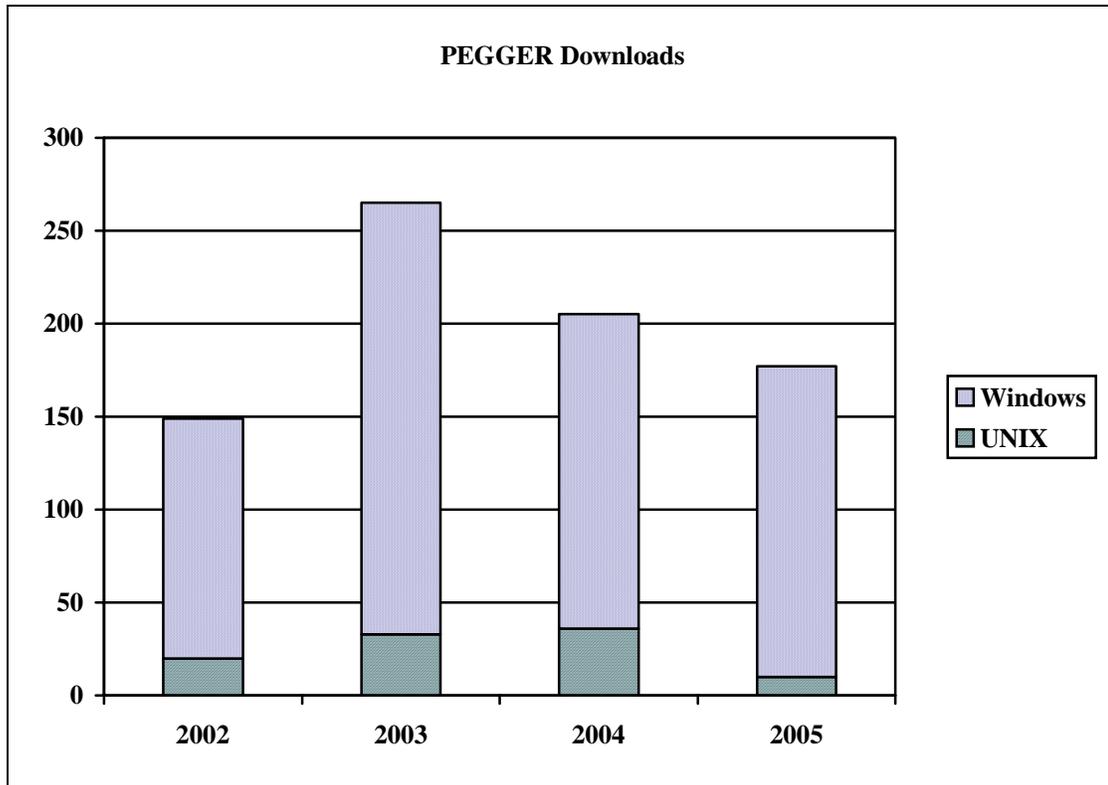


Figure 11 - Downloads of the PEGGER software from the Rural Technology Initiative Website from 2002 to 2004 with projected downloads for 2005.

Applications to Management

Pegging roads in the office before heading to the field can save considerable time. It has been shown that by using PEGGER, feasible preliminary routes can be identified for larger areas in less time than was possible using manual methods alone. In addition to analyzing more alternative routes in less time PEGGER was extended by analyzing side slope data as roads were pegged. By using slope information from a digital elevation model and the distance to the nearest rock-pit, a rough cost estimate could be calculated for each alternative road design (Schiess and Tryall 2002; Schiess and Tryall 2003).

Another application of PEGGER is the ability to create field maps once a route has been chosen. PEGGER keeps track of grade information as a road is pegged. Using the dissolve feature of PEGGER along with the grade information, adjacent segments of pegged road can be dissolved together to create lengths of road that have the same grade. Pegged roads can then be printed on a contour map with grade and length information as shown in Figure 12.

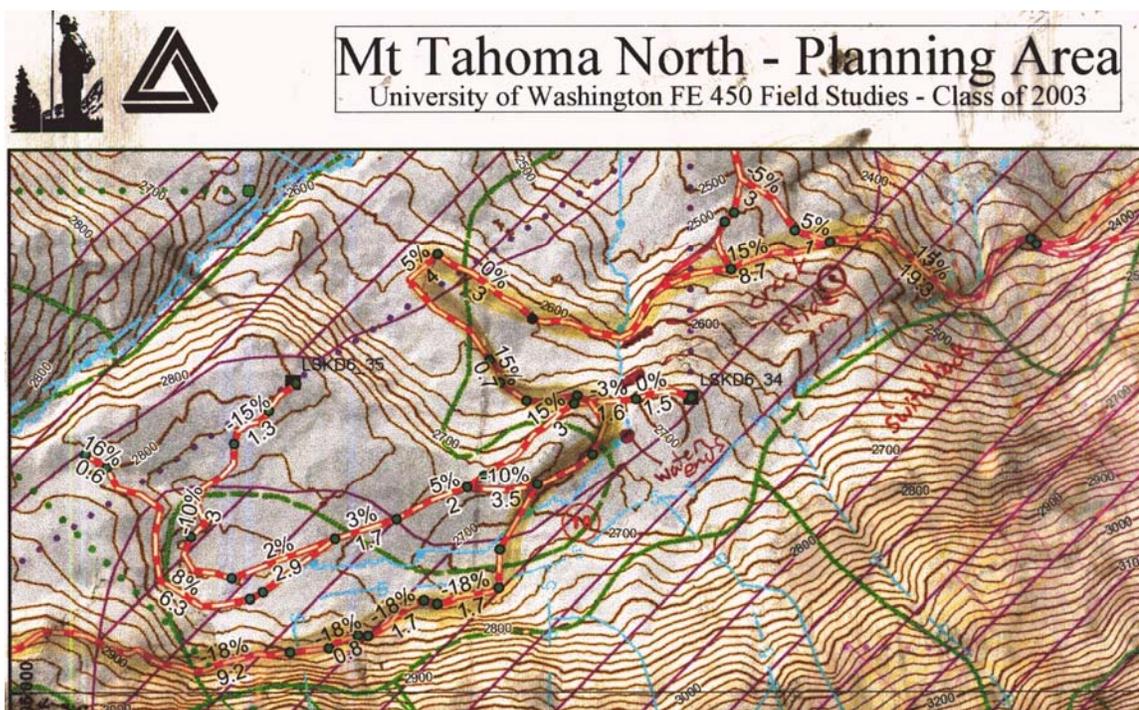


Figure 12 - A dirty and well-used field map produced from a road designed using PEGGER. The road grade and stationing information can be used to locate the proposed road in the field with just the map and a clinometer.

The advantage of using maps like the one above are that the only tool needed to locate the PEGGER designed road is a clinometer. Reading the grade and distances off of the map, roads can be flagged in by simply starting at a known point and pacing the required distance at the indicated grade. Placing flagging along the way, a single person can locate many miles of road in a single day. This method has been used by the University of

Washington Forest Engineering Capstone students very successfully over the past few years and is considered to be their preferred method for forest road grade-line location.

Limitations

The PEGGER program relies on digital topographic information to identify potential road locations. To be of value to the forest professional, the topographic information must accurately represent the actual ground conditions. Steve Reutebuch (1988) noted about ROUTES that “the accuracy of the 30-meter (USGS) DEM’s available at the time were insufficient for accurate route projection.” With the availability of 10-meter digital elevation data and the current popularity of LIDAR data, route projection has become more feasible but discrepancy between the data and actual field conditions should be expected.

The PEGGER program is a tool for quickly identifying possible route location alternatives given grades specified by the user. The tool does not evaluate additional environmental and economic constraints that must be considered by the forest professional such as soil types, hydrology, property lines and slope classes. The GIS provides a framework where these analyses can be implemented but it is outside the scope of the PEGGER program.

The algorithm that PEGGER uses to identify a segment is dependant on the contour interval of the dataset, and the desired grade of the road. At steeper grades, this works very well as pegged segments are short. However, as the grade of interest decreases, as shown in Figure 13, PEGGER must make the pegged segments longer and longer. At a 1% grade on a 20 foot contour interval dataset, the segment length becomes 2000 feet. At such long segment lengths, the topography is not being accurately represented. Therefore, at shallow grades, it is necessary for the engineer to use standard GIS functionality to

locate roads manually and only rely on pegged segments as a guide for freehand placement of a preliminary route location.

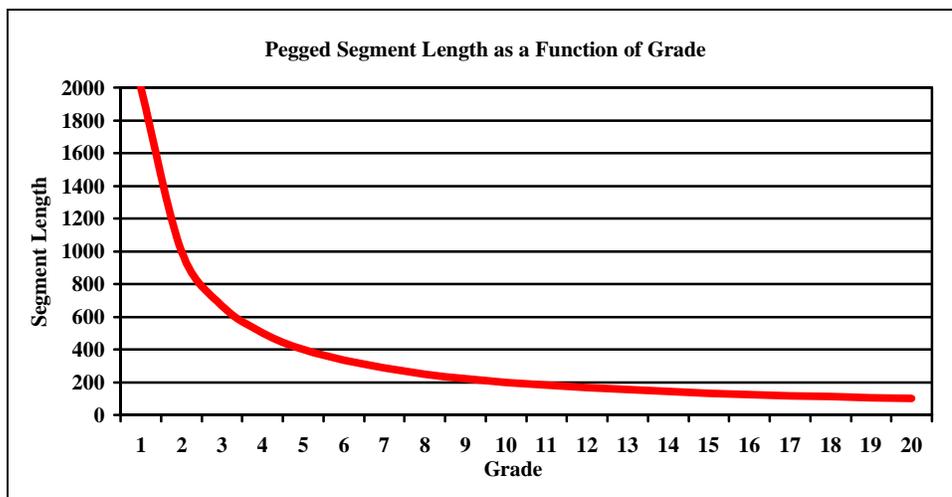


Figure 13 - At shallow grades, PEGGER creates very long segments of road which may not represent the topography properly.

Users must also be careful crossing incised stream valleys (or draws) when using PEGGER. Figure 14 demonstrates the problems that can be encountered when using PEGGER in steep mountainous terrain where incised stream valleys are common. The straight red segments were placed using PEGGER with a -5% grade. These segments, because of the long segment length, skip over the topography of the incised stream. At the highest point, the road is over 100 feet above the stream! In reality, this road would be built more like the curvy green segments that more closely follow the topography of the stream valleys. The road that was initially pegged at -5% grade when built would come out to something more like -3.7%. This difference in grade can be considerable depending on site specific conditions. While it may be appropriate in this situation for a more relaxed grade through the turns, if the grade were critical it would be important to look out for these circumstances.

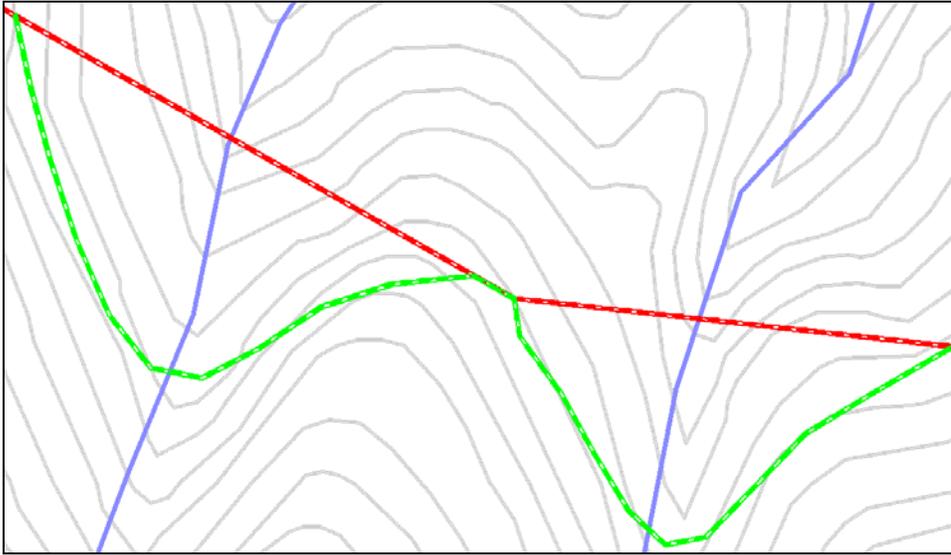


Figure 14 - Care must be taken when pegging across stream valleys. Here, the red road was located using PEGGER, however, the road would likely be constructed more like the green road. This difference in road length can significantly affect the grade of the road.

Conclusions

While computerized route location has been used by forest professionals for many years, it has never become a widely used technology to evaluate initial road locations. With PEGGER, the forest planner can quickly evaluate route locations within a GIS framework, giving the planner access to additional GIS functionality. PEGGER was designed with simplicity and minimal investment cost as primary objectives. Through the use of a carefully designed user interface and extensive tutorial, a typical user can be locating roads in a few minutes on their own PC taking full advantage of forest technology. The automation, rather than optimization, of route location gives the forest planner more confidence in their designs since it incorporates their knowledge in the process.

In the past few years, this tool has been used by hundreds of forestry professionals from all over the world. As shown by the number of downloads of the product, and the responses received from users, PEGGER is another valuable tool that forest planners can add to their toolbox.

Bibliography

- (1985). FORPLAN version 1. Washington, D.C., U.S. Dept. of Agriculture, Forest Service, Land Management Planning Systems Section.
- Akay, A. E., I. R. Karas, et al. (2004). Using High-Resolution Digital Elevation Model for Computer-Aided Forest Road Design. Geo-Imagery Bridging Continents, Istanbul, Turkey, The International Society for Photogrammetry and Remote Sensing.
- Anderson, A. E. and J. Nelson (2004). "Projecting vector-based road networks with a shortest path algorithm." Canadian Journal of Forest Research **34**(7): 1444-1457.
- Becker, G. and D. Jaeger (1992). Integrated design, planning and evaluation of forest roads and logging activities using GIS-based interactive CAD-systems. Computer Supported Planning of Roads and Harvesting Workshop, Feldafing, Germany, International Union of Forestry Research Organizations.
- Berry, J. K., D. J. Buckley, et al. (1998). Visualize Realistic Landscapes: 3-D Modeling Helps GIS Users Envision Natural Resources. GeoWorld.
- Boyland, M. (2003). Hierarchical Planning in Forestry. Vancouver, B.C., University of British Columbia, Canada: 7.
- Burke, D. (1974). Automated analysis of timber access road alternatives. Pacific. U. P. N. F. a. R. E. Station, USDA Forest Service. **GTR-PNW-27**: 1-40.
- Carson, W. W. and S. E. Reutebuch (1997). A Rigorous Test of the Accuracy of USGS Digital Elevation Models in Forested Areas of Oregon and Washington. 1997 ACSM/ASPRS Annual Convention & Exposition, Seattle, Washington, American Congress of Surveying and Mapping (ACSM); American Society for Photogrammetry & Remote Sensing (ASPRS).
- Cha, D. S., H. Nako, et al. (1991). "A Computerized Arrangement of Forest Roads Using a Digital Terrain Model." Journal of the Faculty of Agriculture **36**(1-2): 131-142.
- Coulter, E. D., W. Chung, et al. (2001). Forest road earthwork calculations for linear road segments using a high resolution digital terrain model generated from LIDAR data. First Precision Forestry Symposium, University of Washington, College of Forest Resources, Seattle, Washington.

- Dudek, S. (2004). Personal communication with Sebastian Dudek a Resource Information Systems Analyst with Rayonier. Hoquiam, WA.
- Dürrstein, H. (1992). Detailed Road Planning Using Microcomputers. Computer Supported Planning of Roads and Harvesting Workshop, Feldafing, Germany, International Union of Forestry Research Organizations.
- Dvorscák, P. and M. Hríb (1992). Development and Present State of Utilizing Computing Technique in Projecting of Forest Roads in Slovakia. Computer Supported Planning of Roads and Harvesting Workshop, Feldafing, Germany, International Union of Forestry Research Organizations.
- Epstein, R., J. Sessions, et al. (2001). PLANEX: A System to Identify Landing Locations and Access. 11th International Mountain Logging and Pacific Northwest Skyline Symposium, Seattle, Washington, USA, University of Washington.
- Heralt, L. (2002). "Using ROADENG system to design an optimum forest road variant aimed at the minimization of negative impacts on the natural environment." Journal of Forest Science **48**(8): 361-365.
- Kent, B., B. B. Bare, et al. (1991). "Natural Resource Land Management Planning using Large-Scale Linear Programs: The USDA Forest Service Experience with Forplan." Operations Research **39**(1): 13-27.
- Khan, B. (2004). Personal communication with Babar Khan, Geomatics Engineering Department, Regional Power Inc. Toronto, Ontario, Canada.
- Kobayashi, H. (1973). A Study on Automatic Processing in Forest Road Design Mainly Concerning the Earthwork, Government Forest Experimental Station. Tokyo, Japan. **249**.
- Krogstad, F. and P. Schiess (2004). The Allure and Pitfalls of Using LiDAR topography in Harvest and Road Design. Joint Conference of IUFRO 3.06 Forest Operations in Mountainous Conditions and the 12th International Mountain Logging Conference, Vancouver, B.C., Canada.
- McCarter, J. B. (2001). Landscape management system (LMS): background, methods, and computer tools for integrating forest inventory, GIS, growth and yield, visualization and analysis for sustaining multiple forest objectives. College of Forest Resources. Seattle, WA, University of Washington. **Doctor of Philosophy**.

- McGaughey, R. J. (2000). EnVision - Environmental Visualization System. Seattle, WA, USDA Forest Service, Pacific Northwest Research Station: EnVision is designed to be a full featured rendering system for stand- and landscape-scale images.
- McGaughey, R. J. and R. H. Twito (1988). VISUAL and SLOPE: Perspective and quantitative representation of digital terrain models. USDA, USDA Forest Service. **PNW-GTR-214**: 1-26.
- Pearce, J. K. (1960). Forest Engineering Handbook. B. o. L. M. United State Department of the Interior: 220.
- Pereira, L. and L. Janssen (1999). "Suitability of laser data for DTM generation: a case study in the context of road planning and design." ISPRS Journal of Photogrammetry and Remote Sensing **54**(4): 244-253.
- Reisinger, T. W. and C. J. Davis (1985). A Map-Based Decision Support System for Operational Planning of Timber Harvests. Winter Meeting of the American Society of Agricultural Engineers, Chicago, Il, American Society of Agricultural Engineers.
- Reutebuch, S. (1988). ROUTES: A Computer Program for Preliminary Route Location. U. D. o. A. Pacific Northwest Research Station, Forest Service. **PNW-GTR-216**: 18.
- Reutebuch, S., R. J. McGaughey, et al. (2003). "Accuracy of a high-resolution digital terrain model under a conifer forest canopy." Canadian Journal of Remote Sensing **29**(5): 527-535.
- Romberg, W. (2005). Personal communication with Wes Romberg, a Forester at Pacific Forest Management, Inc. Port Hadlock, WA.
- Schiess, P. and A. Arntzen (2001). Assessment of Operational Feasibilities for Implementing the OESF Conservation Strategy. Seattle, WA, University of Washington.
- Schiess, P. and L. M. O'Brien (1995). The Application of Geographic Information Systems to Forest Operations: The Integration of Cable Setting Design into GIS. 2nd Brazilian Symposium on Timber Harvesting and Forest Transportation, Salvador, Bahia, Brazil.

- Schiess, P. and L. W. Rogers (1999). A Watershed and Transportation Plan for the North Hoodspert Planning Area. Seattle, WA, University of Washington.
- Schiess, P. and L. W. Rogers (2000). A Thinning and Access Strategy for Accelerated Stand Habitat Creation - Burnt Mountain. Seattle, WA, University of Washington.
- Schiess, P. and J. Tryall (2002). Transportation & Road Management Requirements to Facilitate Habitat Restoration in the Tyee South Planning Area. Seattle, WA, University of Washington: 106.
- Schiess, P. and J. Tryall (2003). Developing a Road System Strategy for the Tahoma State Forest. Seattle, WA, University of Washington.
- Sedjo, R. A. (1987). FORPLAN, an evaluation of a forest planning tool: proceedings of a symposium, November 4-6, 1986, Denver, Colorado / Thomas W. Hoekstra, A.A. Dyer, Dennis C. Le Master, technical editors. Fort Collins, Colo., U.S. Dept. of Agriculture, Forest Service, Rocky Mountain Forest and Range Experiment Station.
- Sessions, J. (1987). A Heuristic Algorithm for the Solution of the Variable and Fixed Cost Transportation Problem. The 1985 Symposium of System Analysis in Forest Resources, University of Georgia, Athens.
- Sessions, J. and J. B. Sessions (1988). SNAP - A Scheduling and Network Analysis Program for Tactical Harvest Planning. International Mountain Logging and Pacific Northwest Skyline Symposium.
- Sessions, J. and J. B. Sessions (1992). Tactical Harvest Planning Using SNAP 2.03. Computer Supported Planning of Roads and Harvesting Workshop, Feldafing, Germany, International Union of Forestry Research Organizations.
- Thompson, M. (1988). "Optimizing spur road spacing on the basis of profit potential." Forest Products Journal **38**(5): 53-57.
- Thrall, G. I. and M. Campins (2004). Mapping the Geospatial Community. Geospatial Solutions. **14**: 46-52.
- Watson, H. J. and M. M. Hill. (1983). "Decision support systems or what didn't happen with MIS." Interface **13**(5): 81-88.

Xu, S. (1996). Preliminary planning of forest roads using ARC GRID. Department of Forest Engineering. Corvallis, OR, Oregon State University: 112.

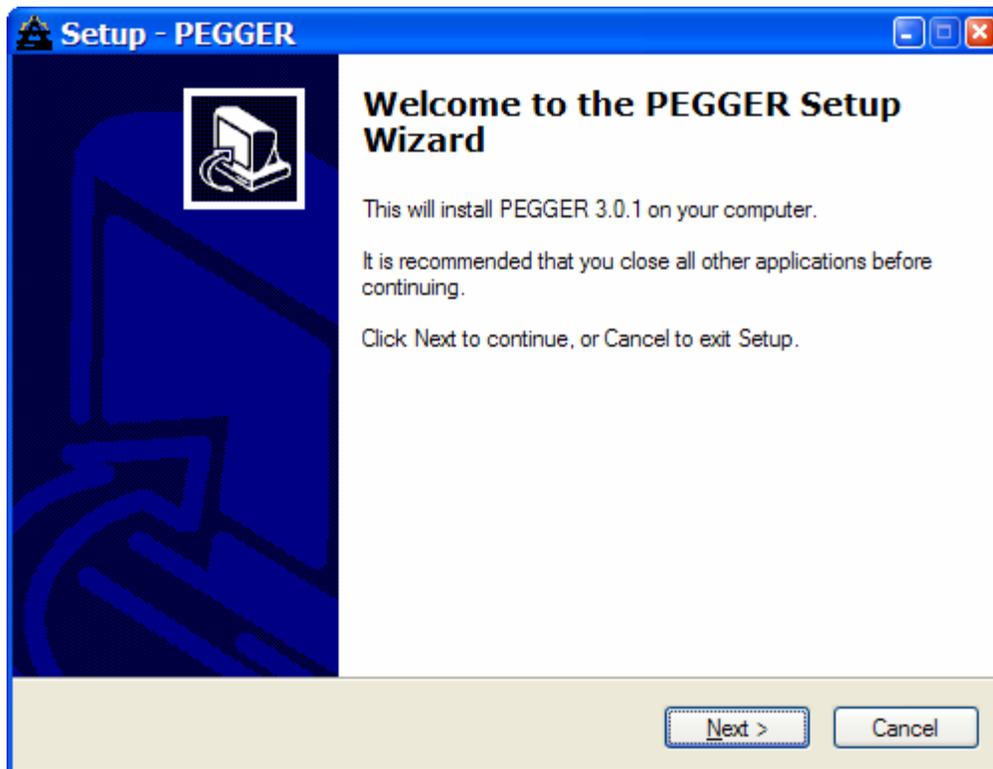
Appendix A – PEGGER Software Manual

About this Tutorial

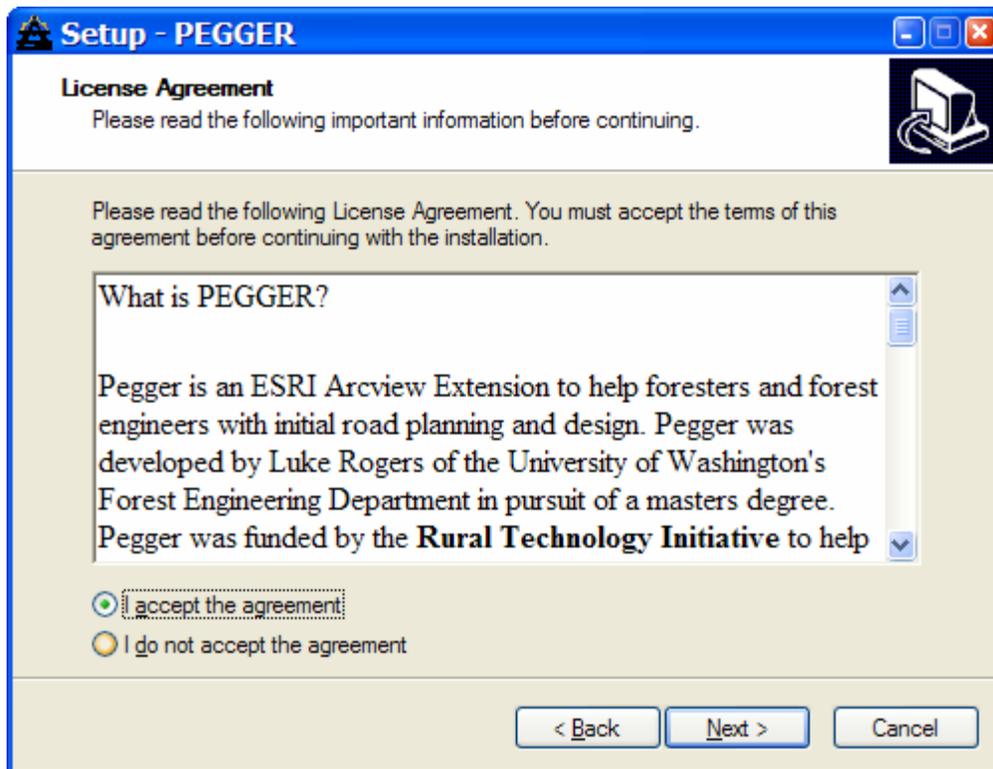
This tutorial is designed to help new users install and run PEGGER in ArcView 3 GIS. While PEGGER will work on all ArcView 3 platforms, this tutorial was written specifically for Microsoft Windows. Users of other platforms will still find this tutorial valuable, but will have to make the appropriate translations when necessary.

Installing PEGGER

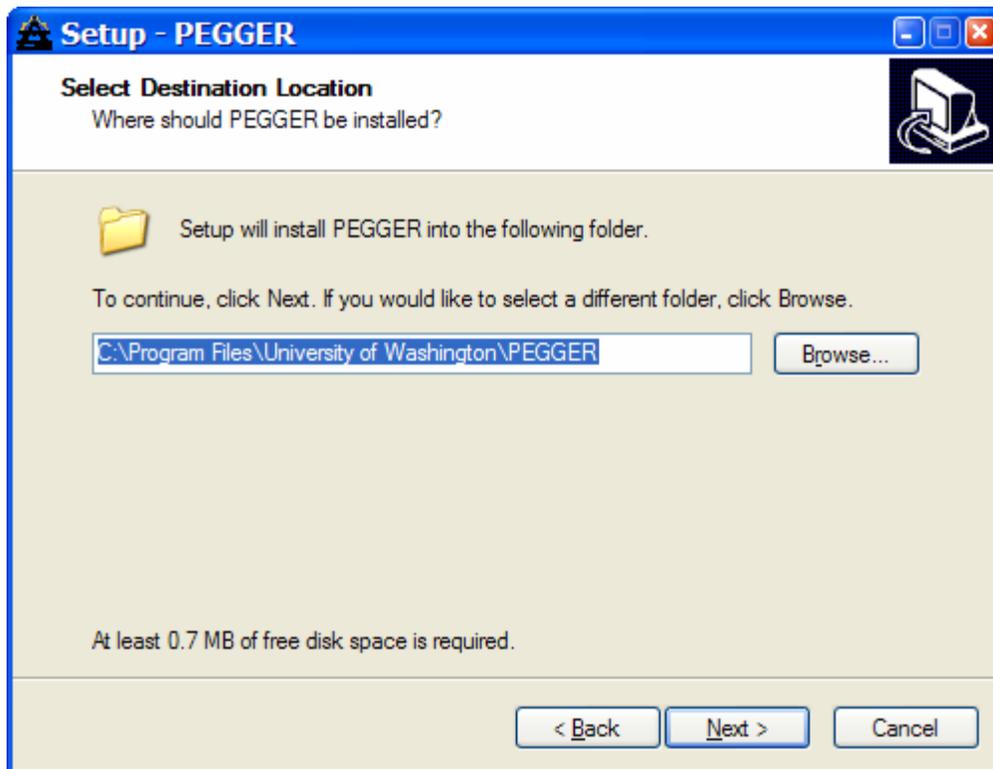
PEGGER can be downloaded from the Rural Technology Initiative website at <http://www.ruraltech.org/tools/pegger>. Download the setup.exe file or run it directly from the web to start the installation. On the first page of the setup program, select Next to continue.



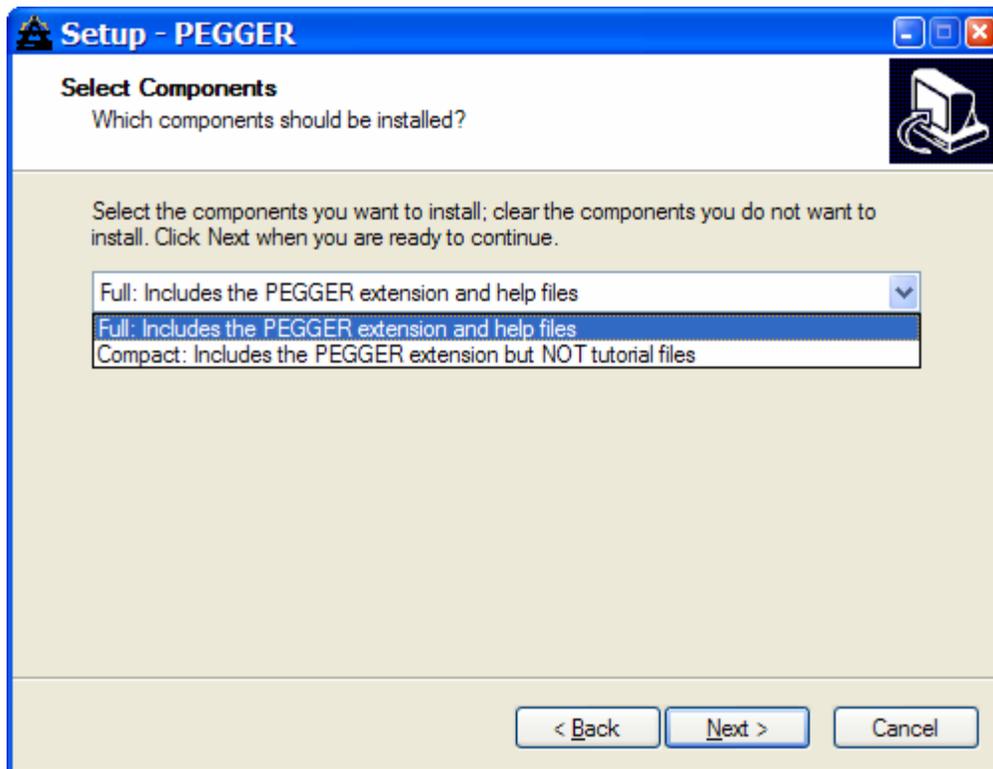
Read the terms of the license agreement and select 'I accept the agreement' if you wish to install the software. Select Next to continue.



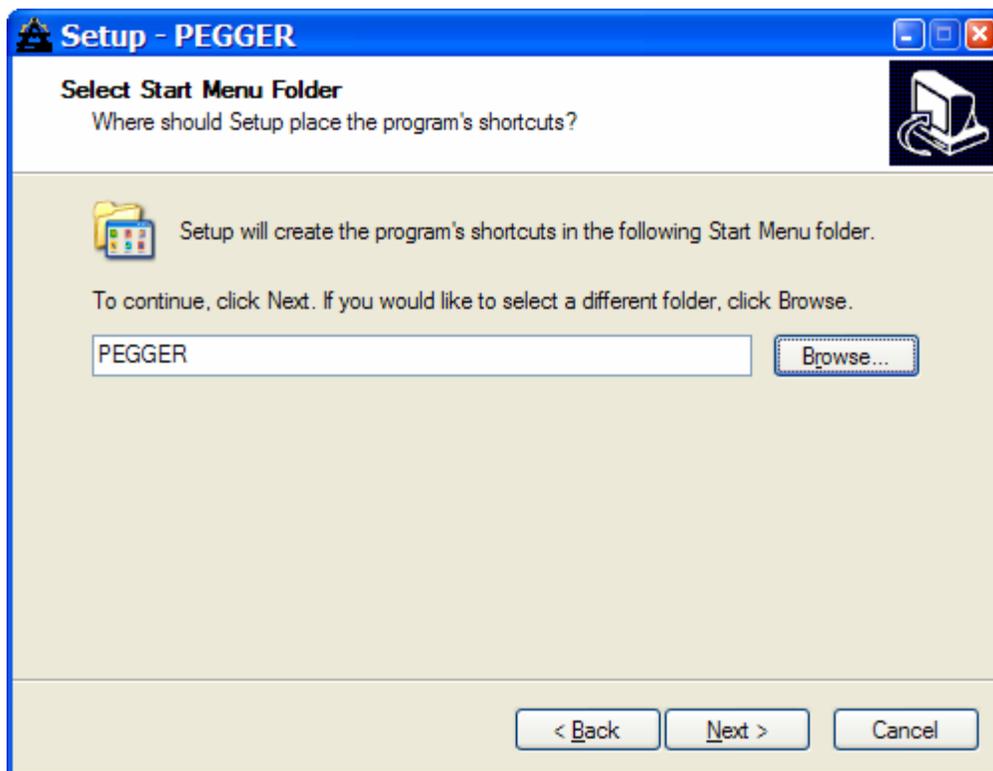
Select a location for Pegger to be installed. Most of the files that Pegger installs will be in the ArcView installation directory so less than 1 MB of files will be placed in the installation directory chosen here. Accept the default location and select Next to continue.



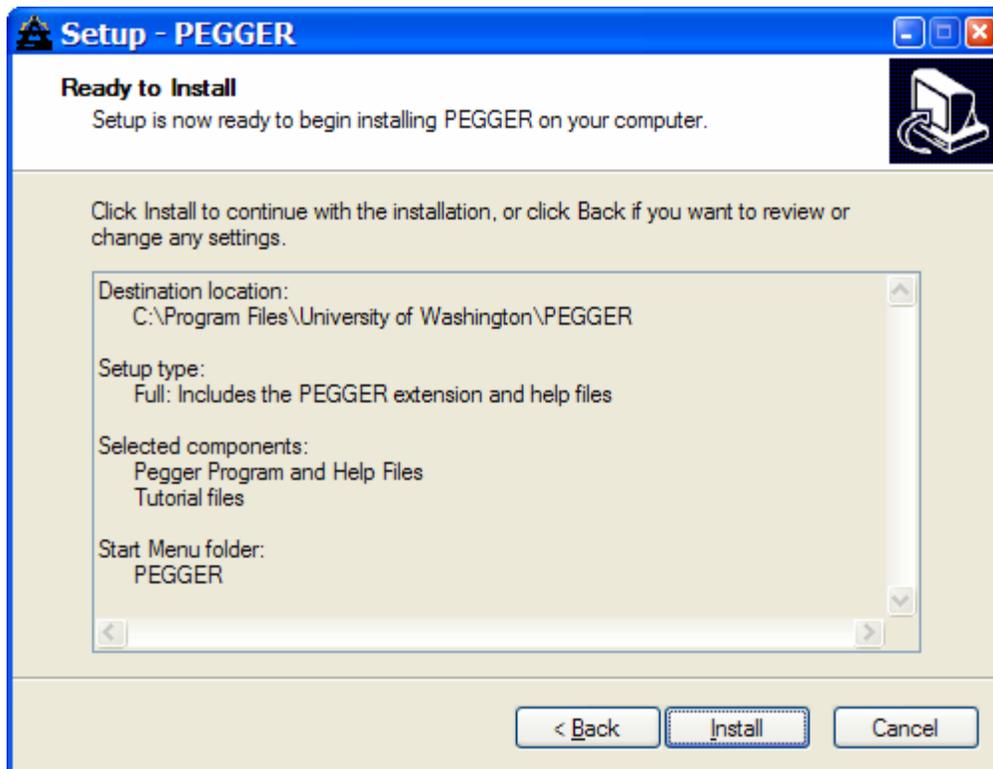
All installations will include the Pegger ArcView extension and the integrated help files. If you do not wish to install the tutorial, select Minimal. The Compact installation includes the Pegger tutorial with only the three shapefile datasets for those who do not have Spatial or 3D Analyst. The Full installation includes the Pegger tutorial and all the tutorial datasets: three shapefiles, a GRID elevation model, and a TIN for those who have Spatial or 3D analyst and wish to use the Survey P-Line function with RoadEng.



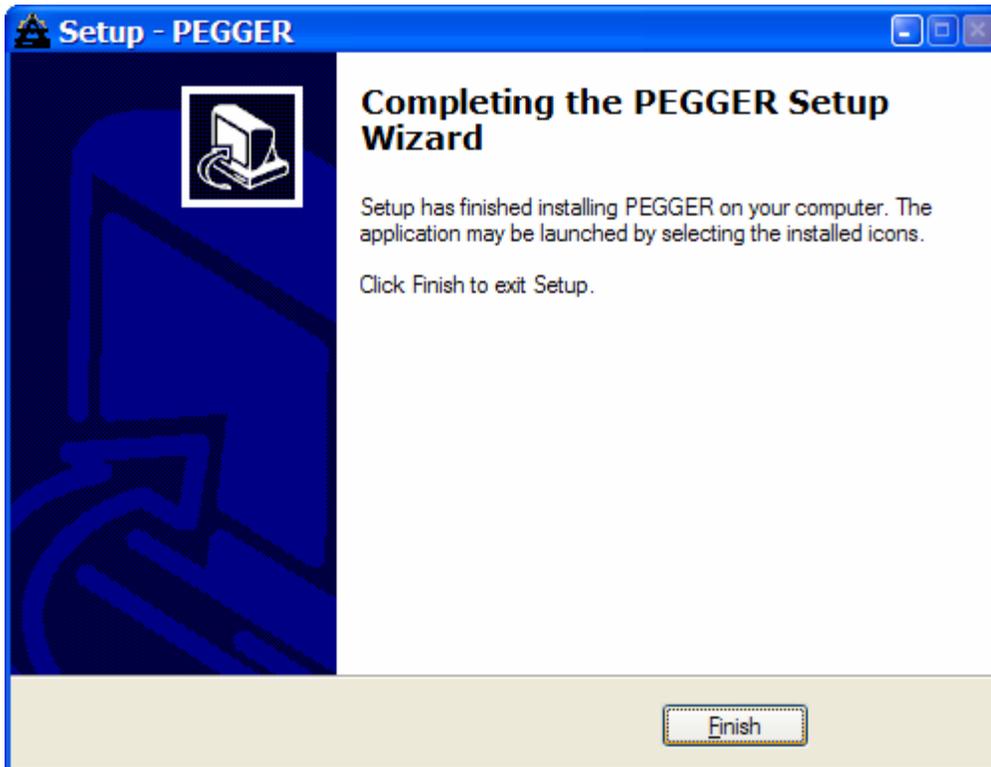
Select a folder to place the program shortcuts in the start menu. Either the default location or the ESRI\ArcView GIS 3.x folder work well.



Review the installation settings and select Install to complete the installation.



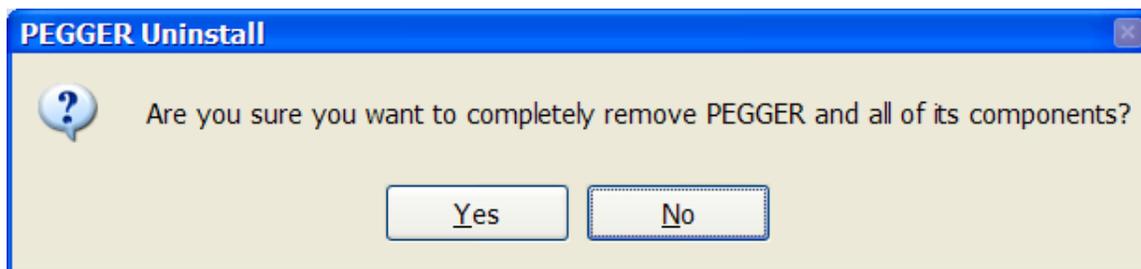
A completion screen will appear if you have installed Pegger successfully.



You should now have a new item under your start menu called Pegger with shortcuts to the Pegger Tutorial, the Pegger Help file and the un-installation program depending on which installation components were selected.

Uninstalling Pegger

To uninstall Pegger, use either the Uninstall Pegger function in the PEGGER folder under the Start menu or use Add/Remove Programs in the Control Panel. Select Yes when prompted to remove the software.



Finding Help

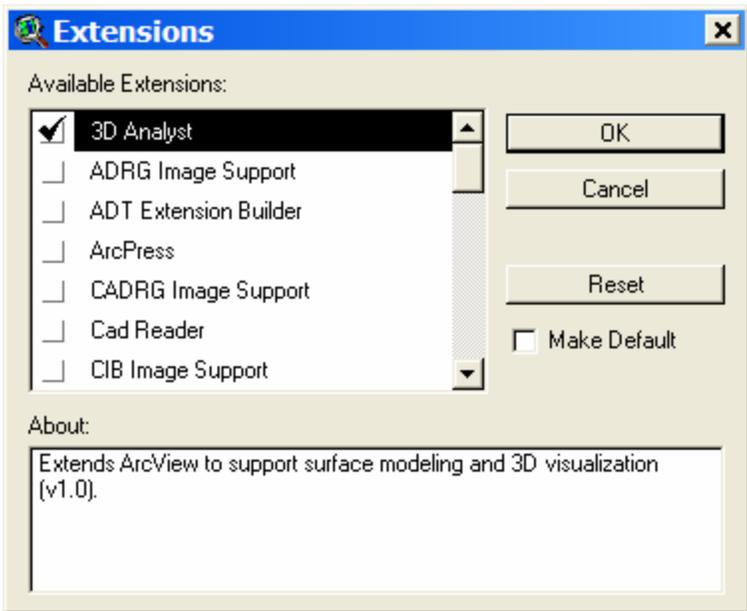
Installed with Pegger are a complete set of help files that are integrated into the ArcView help system. In addition, a shortcut to the Pegger help can be found in the PEGGER folder under the Start menu. A good place to start learning about Pegger is in the table of contents under Extensions > Pegger > Getting Started.

All of the tools, dialogs and menus that are part of the Pegger extension have context based help that can be viewed in the status bar at the bottom of the ArcView application window. In addition using the View Help tool:  and clicking on any Pegger control will bring up the appropriate page in Pegger Help.

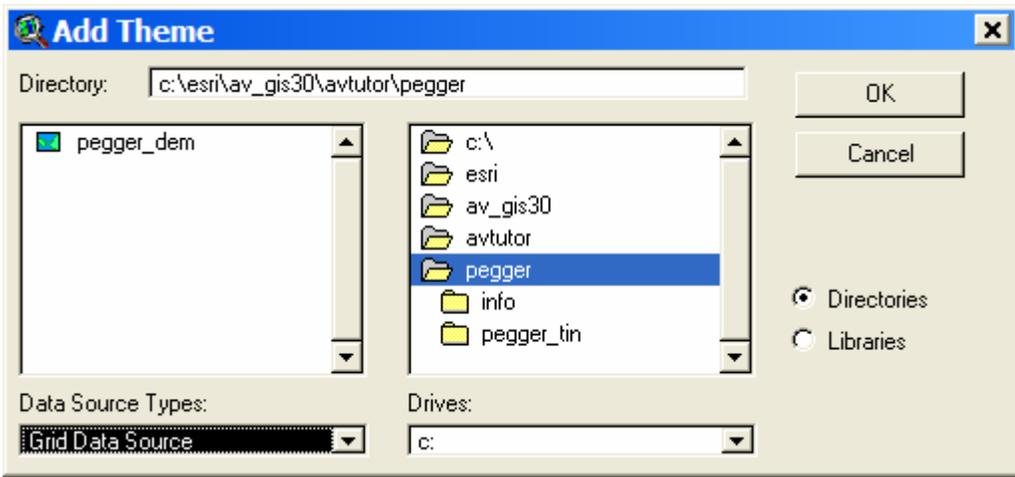
Using Tutorial.apr

Included with the Full or Compact Pegger installation are a tutorial.apr project file, pegger_trans.shp, pegger_cont.shp and pegger_hydro.shp shapefiles. If you have ArcView Spatial Analyst or 3D Analyst and chose the Full installation option, then you may also have a pegger_dem GRID and a pegger_tin TIN.

The tutorial project file opens with the three shapefiles already symbolized and added to a view. If you have Spatial Analyst or 3D Analyst you can also add the DEM or TIN to the view. First, enable the Spatial Analyst or 3D Analyst Extension by selecting Extensions under the File menu and selecting 3D or Spatial Analyst.

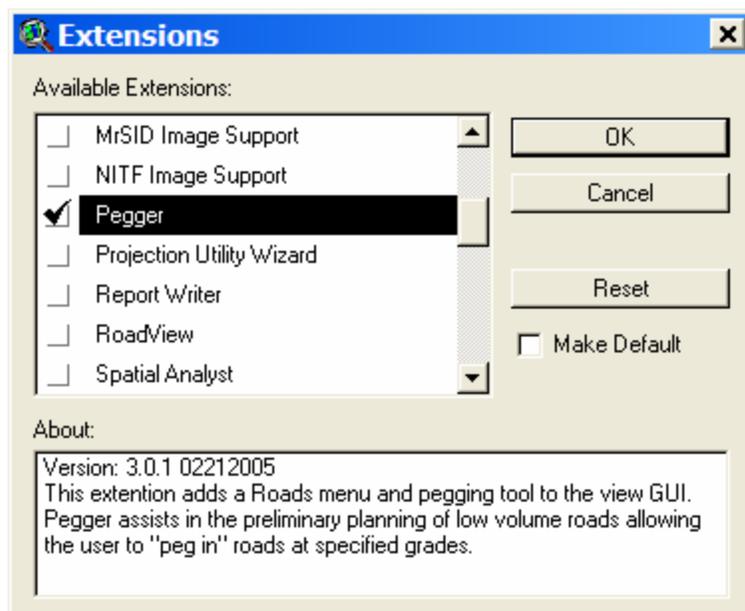


Then, using the Add Theme Button:  browse to the AVTUTOR\PEGGER directory, select Grid Data Source or Tin Data Source for the Data Source Type, and select pegger_dem or pegger_tin.



Activating the Extension

To Activate the Pegger extension you must select it in the Extensions dialog box. To open the Extensions dialog box, select Extensions under the File menu. Select Pegger and any other extensions you wish to use and OK the Extensions dialog box. If you want Pegger to be available each time you open ArcView, select pegger and then check the Make Default box on the right.



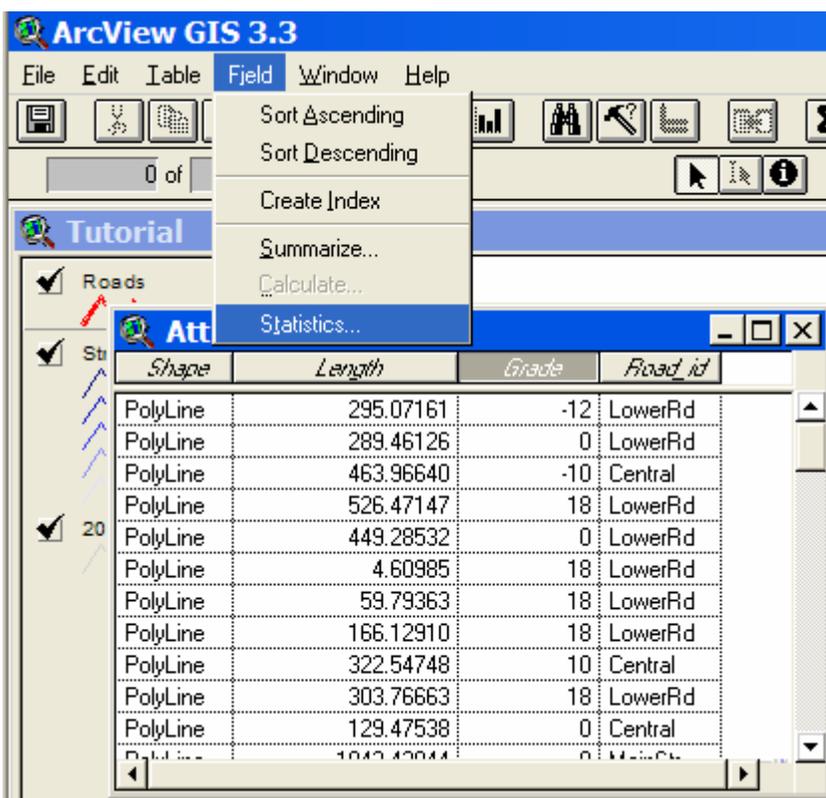
The road theme

Grade attribute

Pegger can keep track of grade information as roads are pegged. In order for Pegger to keep track of grade information there must be a numeric field in the attribute table that can store the grade information. If you want to keep track of grade information as you peg, check to make sure an attribute already exists that will work, or add a new attribute to the table.

Open the attribute table for the road theme by making the road theme the active theme (select it in the view legend), and selecting Theme > Table or by using the Open Theme Table button: 

Look at the fields in the table to determine if there is an appropriate field for storing grade information. In this case, the Grade field looks like it will work, but it is always good to check to make sure it is a numeric field. Select the Grade field and look under the Field menu. If the statistics menu item is not grayed out, then it is a numeric field (it will be grayed out if it is a text field).



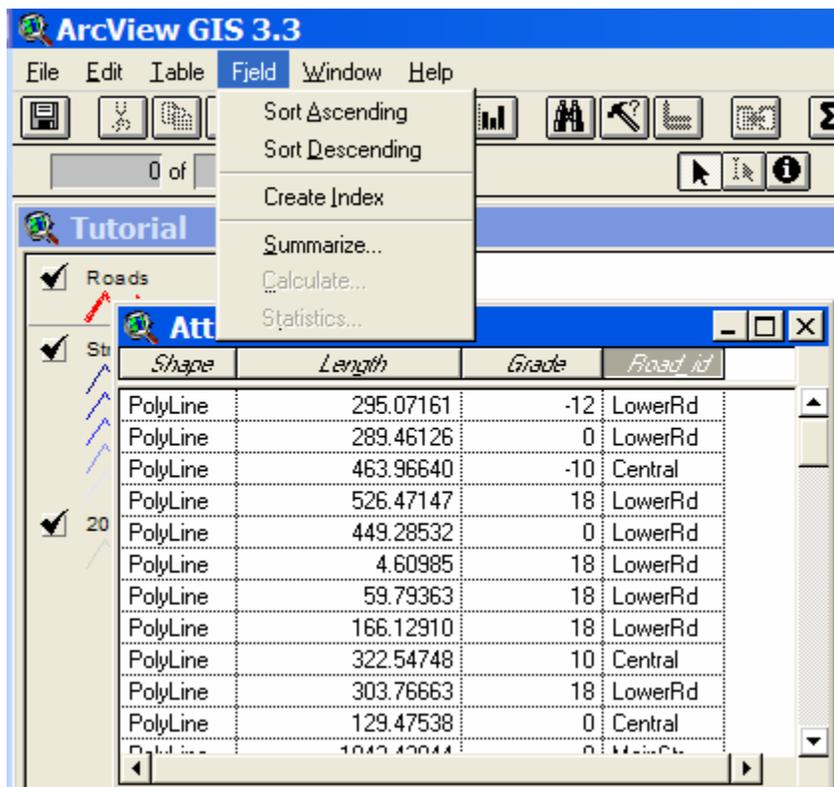
If you want to add a new field to the table to store grade information, you will need to start editing the table, and then add a new field of type number. If you would like more information, lookup “editing a table” in the ArcView help file index.

Name attribute

In addition to storing grade information as roads are being pegged, road name information can also be saved. In order for Pegger to keep track of road name information there must be a text (a.k.a. string) field in the attribute table that can store the road name information. If you want to keep track of road name information as you peg, check to make sure an attribute already exists that will work, or add a new attribute to the table.

Open the attribute table for the road theme by making the road theme the active theme and selecting Theme > Table or by using the Open Them Table button: 

Look at the fields in the table to determine if there is an appropriate field for storing road name information. In this case, the Road_id field looks like it will work, but it is always good to check to make sure it is a text field. Select the Road_id field and look under the Field menu. If the statistics menu item is grayed out then it is a text field.



If you want to add a new field to the table to store road name information, you will need to start editing the table, and then add a new field of type string. If you would like more information, lookup “editing a table” in the ArcView help file index.

The contour theme

Elevation attribute

Pegger uses a contour theme as a representation of the ground surface. While this could have been done on a raster dataset or on a Tin, both of those options would have required access to either the Spatial or 3D Analyst Extension. To make the program as simple and inexpensive to use as possible, contours are used.

In order to use a contour theme with Pegger, an numeric attribute in the contour theme must store the elevation values of the contours. Check to make sure that the contour

theme has a numeric elevation value field by opening the table and looking at the themes attributes.

If the contour theme attribute table does have an elevation field but the values are stored as strings rather than numbers, they will need to be converted before being used with Pegger. For more information, lookup “elevation data” in the ArcView help file index.

To be continued....

Appendix B – PEGGER Avenue™ Code

The following ArcView 3 Avenue code comprises only the parts of PEGGER that deal with route projection, surveying, and other critical aspects of the program. Much of the code behind PEGGER is not directly related to the pegging task and is not included here.

```
' Pegger.PeggingTool.Apply
'
' Created By: Luke Rogers
'             In pursuit of a Masters of Science
'             Forest Engineering
'             College of Forest Resources
'             University of Washington
'             Box 352100
'             Seattle, WA 98115
'
'             luke@nwgeospatial.com
'
'             October 17th, 2000
'
' Description: respond to user click and initiate pegging process
'
' Calls:      Pegger.PeggingTool.StartRoad(theView,theTheme,thePoint)
'            Pegger.PeggingTool.Peg(thePoint,theLastPoint)
'
' Returns:    nothing
'
'*****
theView = av.GetActiveDoc
theTheme = theView.GetEditableTheme
av.ClearMsg
theDict = theTheme.GetObjectTag
theLastPoint = theDict.Get("LastPoint")

If ((System.IsShiftKeyDown) or (theLastPoint = nil)) then
    shiftDown = TRUE
else
    shiftDown = FALSE
end

' Get the point off the display when user clicks
thePoint = theView.GetDisplay.ReturnUserPoint

' If the point they selected is outside of the extent of
' the contour theme then return a warning
theDict = theTheme.GetObjectTag
theContTheme = theDict.Get("ContTheme")
theExtentRect = theContTheme.ReturnExtent
if (theExtentRect.Contains(thePoint).Not) then
    MsgBox.Warning("You cannot peg in a road where no contours exist",
        "PeggingTool.Apply")
    return nil
end
```

```

'*****
' If the shift key is down then it is a new road
if (shiftDown = TRUE) then
  av.ShowMsg("Starting new road at:"++thePoint.AsString)
  theStartRoadStatus = av.Run("Pegger.PeggingTool.StartRoad",
                              {theView, theTheme, thePoint})

  return nil
else
  ' Figure out which point to go to
  theGoToPoint = av.Run("Pegger.PeggingTool.Peg",
                        {thePoint, theLastPoint})
end
'*****
' If there is no goto point then exit and error
if (theGoToPoint = Nil) then
  return nil
else
  theDict.Set("LastPoint", theGoToPoint)
  theTheme.SetObjectTag(theDict)
end
'*****
' Make the line
theGradeLine = PolyLine.Make({{theLastPoint, theGoToPoint}})

' Add the line to the table
if (theTheme <> nil) then
  theFTab = theTheme.GetFTab
  theShapeField = theFTab.FindField("Shape")
  theFTab.BeginTransaction
  rec = theFTab.AddRecord
  theFTab.SetValue(theShapeField, rec, theGradeLine)
  ' Add the grade if setup to do so
  attributeGrades = theDict.Get("AttributeGrades")
  if (attributeGrades) then
    theGrade = theDict.Get("Grade")
    theGradeField = theDict.Get("GradeField")
    theFTab.SetValue(theGradeField, rec, theGrade)
  end
  ' Add the name if setup to do so
  attributeName = theDict.Get("AttributeName")
  if (attributeName) then
    theName = theDict.Get("Name")
    theNameField = theDict.Get("NameField")
    theFTab.SetValue(theNameField, rec, theName)
  end
  theTheme.GetFTab.EndTransaction
  theTheme.GetFTab.GetSelection.ClearAll
  theTheme.GetFTab.GetSelection.Set(rec)
  theTheme.GetFTab.UpdateSelection
end

av.GetProject.SetModified(true)
return nil

```

```

' Pegger.PeggingTool.StartRoad
'
' Created By:  Luke Rogers
'              In pursuit of a Masters of Science
'              Forest Engineering
'              College of Forest Resources
'              University of Washington
'              Box 352100
'              Seattle, WA 98195
'
'              lwrogers@u.washington.edu
'
'              October 17th, 2000
'
' Description: Start a new pegging segment
'
' Calls:       nothing
'
' Returns:     set object tag for the point that was just
'              clicked so that a new road can be started
'
'*****
'MsgBox.Info("Starting new road", "PeggingTool.StartRoad")
theView = SELF.Get(0)
theTheme = SELF.Get(1)
thePoint = SELF.Get(2)
theDict = theTheme.GetObjectTag
theContTheme = theDict.Get("ContTheme")
theElevItem = theDict.Get("ElevItem")
theGrade = theDict.Get("Grade")
theInterval = theDict.Get("ContInterval")
theCurrentElev = theDict.Get("CurrentElev")
theVTab = theContTheme.GetFTab
theShapeField = theVTab.FindField("Shape")
theElevField = theVTab.FindField(theElevItem.AsString)
'*****
' If we are on a contour then get the elevation from the contour,
' otherwise we will have to use the method below of making polygons

' Clear the selection
theContTheme.ClearSelection
theVTab.UpdateSelection

' Try to select a contour by using the point the user just clicked
if (theContTheme.CanSelect) then
    theContTheme.SelectByPoint(thePoint, #VTAB_SELTYPE_NEW)
end

' See if there is a selection (should only be one if any)
if (theVTab.GetSelection.Count = 1) then
    theSelection = theVTab.GetSelection.GetNextSet(-1)
    theCurrentElevation = theVTab.ReturnValue(theElevField,theSelection)

```

```

' Clear the selection
theContTheme.ClearSelection
theVTab.UpdateSelection

' Lets see if we can snap to the contour for extra precision
if (thePoint.Snap(theVTab.ReturnValue(theVTab.FindField("Shape"),
theSelection), 10)) then
    av.ShowMsg("Snapped to the Contour")
else
    av.ShowMsg("Failed to snap to Contour, location approximate")
end

' Update the CurrentElev item in the dictionary and set the object
' tag on the theme
theDict.Set("CurrentElev", theCurrentElevation)
theDict.Set("LastPoint", thePoint)
theTheme.SetObjectTag(theDict)
av.GetProject.SetModified(true)

' We found the elevation so we are done
'MsgBox.Info("Found elevation using point"+NL+
'Elev:++theCurrentElev.AsString, "PeggingTool.StartRoad")
return "POINT"
end

if (theVTab.GetSelection.Count = 0) then
    MsgBox.Info("Must be on a contour before you can peg in a grade.",
    "PeggingTool.StartRoad")
    return "ZERO"
elseif (theVTab.GetSelection.Count > 1) then
    MsgBox.Info("More than one contour selected."+NL+
    "Zoom in to eliminate ambiguous selection", "PeggingTool.StartRoad")
    return "ZERO"
end

MsgBox.Error("Start Road Instance Not Handled",
"PeggingTool.StartRoad")
return nil

```

```

' Pegger.PeggingTool.Peg
'
' Created By:  Luke Rogers
'             In pursuit of a Masters of Science
'             Forest Engineering
'             College of Forest Resources
'             University of Washington
'             Box 352100
'             Seattle, WA 98195
'
'             lwrogers@u.washington.edu
'
'             October 17th, 2000
'
' Description: Given the coordinates of the users click and
'             the last point this module will return a new
'             point on grade in the direction of the click
'
' Calls:      nothing
'
' Returns:    theGoToPoint (new point on grade)
'
'*****
thePoint = SELF.Get(0)
theLastPoint = SELF.Get(1)
theView = av.GetActiveDoc
theTheme = theView.GetEditableTheme
theDict = theTheme.GetObjectTag
theContTheme = theDict.Get("ContTheme")
theElevItem = theDict.Get("ElevItem")
theGrade = theDict.Get("Grade")
theInterval = theDict.Get("ContInterval")
theCurrentElev = theDict.Get("CurrentElev")
theContFTab = theContTheme.GetFTab
theShapeField = theContFTab.FindField("Shape")
theElevField = theContFTab.FindField(theElevItem.AsString)

'*****

' Make the grade circle
' if the grade is 0 then use a 10% grade circle for segment length
if (theGrade = 0) then
  theRadius = theInterval / (10 / 100)
else
  theRadius = theInterval / (theGrade / 100)
end

' If the radius is less than the snap tolerance then no segment
' will be made so send a warning
if ((theTheme.IsSnapping) or (theTheme.IsInteractiveSnapping)) then
  if ((theTheme.GetSnapTolerance > theRadius.abs) or
      (theTheme.GetInteractiveSnapTolerance > theRadius.abs)) then
    MsgBox.Warning("At least one Snap tolerance is greater than "+

```

```

        " the current"+NL+"segment, a line may not be "+
        "added to the theme until its"+NL+"length is "+
        "greater than the current snap tolerance."+NL+
        "The current segment length is:"++
        theRadius.AsString, "PeggingTool.Peg")
    end
end

theCircle = Circle.Make(theLastPoint, theRadius.Abs)

if (theCircle.IsNull) then
    return nil
end

' Clear the current contour selection
theContTheme.GetFTab.GetSelection.ClearAll

' Get the contour themes vtab bitmap
theBitmap = theContFTab.GetSelection
'*****
' Select the contour to intersect the grade circle with
' first select using the grade circle
theContFTab.SelectByShapes ( {theCircle}, #VTAB_SELTYPE_NEW )

if (theContFTab.GetSelection.Count = 0) then
    MsgBox.Warning("A"++theGrade.AsString+"% grade is too steep."+NL+
        "Lessen the grade to continue pegging", "PeggingTool.Peg")
    return nil
end

' Now select using elevation attribute
if (theCurrentElev <> nil) then
    ' build the query string for the contours on either side
    if (theGrade > 0) then
        theNextContour = theCurrentElev + theInterval
    elseif (theGrade < 0) then
        theNextContour = theCurrentElev - theInterval
    elseif (theGrade = 0) then
        theNextContour = theCurrentElev
    end
    theQuery = "("++theElevItem.AsString+"] ="++
        theNextContour.AsString++)"

    theBitmap = theContFTab.GetSelection
    theContFTab.Query(theQuery, theBitmap, #VTAB_SELTYPE_AND)
    theContFTab.UpdateSelection
    if (theContFTab.GetSelection.Count = 0) then
        MsgBox.Warning("A"++theGrade.AsString+"% grade may be too "+
            "steep."+NL+"Lessen the grade to continue "+
            "pegging", "PeggingTool.Peg")
        return nil
    end
end
else

```

```

    MsgBox.Error("The current elevation is null", "PeggingTool.Peg")
    return nil
end
'*****
' Select the proper point to go to
' Intersect the circle with the contour
' Make a list to hold the points
thePointList = List.Make

for each theSelection in theContFTab.GetSelection
    theContShape = theContFTab.ReturnValue(theShapeField, theSelection)
    theMultiPoint = theContShape.PointIntersection(theCircle)
    thePointList.Merge(theMultiPoint.AsList)
end

' Get the one closest to where we want to go
theDistance = 9999999
for each pnt in thePointList
    if (thePoint.Distance(pnt) < theDistance) then
        theDistance = thePoint.Distance(pnt)
        theGoToPoint = pnt
    end
end
'*****
' Set the elevation for the next contour
theDict.Set("CurrentElev", theNextContour)
theTheme.SetObjectTag(theDict)

theView.GetDisplay.Flush
theView.GetGraphics.Empty

av.GetProject.SetModified(true)

return theGoToPoint

```

```

' Pegger.RoadMenu.Dissolve
'
' Created By: Luke Rogers
'             In pursuit of a Masters of Science
'             Forest Engineering
'             College of Forest Resources
'             University of Washington
'             Box 352100
'             Seattle, WA 98195-2100
'
'             lwrogers@u.washington.edu
'
'             December 11th, 2004
'
' Description: Dissolve roads based on grade or any other item.
'
' Calls:      nothing
'
' Returns:    merges adjacent features together based on a common
'             attribute
'
'*****
' Table summary to merge shapes
'Get the theme
theView = av.GetActiveDoc
theTheme = theView.GetActiveThemes.Get(0)
if (theTheme = nil) then
    MsgBox.Warning("Can only dissolve grades on the first active theme"
                  , "RoadMenu.Dissolve")
    return nil
end
if (theTheme.GetFTab.GetShapeClass.GetClassName <> "PolyLine") then
    MsgBox.Warning("Can only dissolve grades on PolyLine features.",
                  "RoadMenu.Dissolve")
    Return nil
end

theFTab = theTheme.GetFTab
'*****
'Get the fields
theFieldList = theTheme.GetFTab.GetFields
theValidFields = List.Make
for each theField in theFieldList
    if ((theField.IsTypeNumber) or (theField.IsTypeString)) then
        theValidFields.Add(theField)
    end
end
'*****
'Get the dissolve field
theDissolveField = MsgBox.List(theValidFields, "Select the field to "+
                               "dissolve on:", "Select Dissolve Attribute")

if (theDissolveField = nil) then

```

```

    return nil
end
'*****
'Get the summary fields
theOtherFields = theValidFields.DeepClone
theDissolveFieldIndex = theValidFields.FindByValue(theDissolveField)
theOtherFields.Remove(theDissolveFieldIndex)

theSummaryList = List.Make
NumTypes = {"by Average", "by Sum", "by Minimum Value",
            "by Maximum Value",
            "by Standard Deviation", "by Variance"}
StrTypes = {"by First", "by Last", "by Count"}
for each theField in theOtherFields
    if (theField.IsTypeNumber) then
        for each type in NumTypes
            theSummaryList.Add((theField.AsString++type))
        end
    elseif (theField.IsTypeString) then
        for each type in StrTypes
            theSummaryList.Add((theField.AsString++type))
        end
    end
end
end

theListToSummarize = MsgBox.MultiListAsString(theSummaryList,
        "Select additional summary fields if desired:",
        "Select Dissolve Summaries")
if (theListToSummarize = nil) then
    return nil
end
'*****
'Parse the chosen summary fields
theSummaryFieldList = List.Make
theSummaryTypes = List.Make

for each theKey in theListToSummarize
    theTokens = theKey.AsTokens(" ")
    theField = theFTab.FindField(theTokens.Get(0))
    theSummaryFieldList.Add(theField)
    theValue = theTokens.Get(2)
    if (theValue="Sum") then
        theSummary = #VTAB_SUMMARY_SUM
    elseif (theValue="Average") then
        theSummary = #VTAB_SUMMARY_AVG
    elseif (theValue="Minimum") then
        theSummary = #VTAB_SUMMARY_MIN
    elseif (theValue="Maximum") then
        theSummary = #VTAB_SUMMARY_MAX
    elseif (theValue="Standard") then
        theSummary = #VTAB_SUMMARY_STDEV
    elseif (theValue="Variance") then
        theSummary = #VTAB_SUMMARY_VAR

```

```

elseif (theValue="First") then
  theSummary = #VTAB_SUMMARY_FIRST
elseif (theValue="Last") then
  theSummary = #VTAB_SUMMARY_LAST
elseif (theValue="Count") then
  theSummary = #VTAB_SUMMARY_COUNT
end
theSummaryTypes.Add(theSummary)
end
'*****
'Add the shape field to the summary
theShapeField = theFTab.FindField("Shape")
theSummaryFieldList.Add(theShapeField)
theSummaryTypes.Add(#VTAB_SUMMARY_AVG)
'*****
'Select location to store new shapefile
theNewShapefile = FileDialog.Put("Dissolve".AsFileName, "*.shp",
                                "Save dissolve shapefile as...")
if (theNewShapefile = nil) then
  return nil
end
'*****
'Run the dissolve
dissolveFTab = theFTab.Summarize(theNewShapefile, Shape,
                                theDissolveField, theSummaryFieldList, theSummaryTypes)
'*****
' Explode multi-part shapes into single-part
theShapeField = dissolveFTab.FindField("Shape")

'Start editing the FTab
dissolveFTab.StartEditingWithRecovery

' Get every selected record after another
For Each record In dissolveFTab
  theLine = dissolveFTab.ReturnValue(theShapeField, record)
  theLine = theLine.ReturnConnected

  ' If the selected feature contains more than one part, explode it
  If (theLine.CountParts > 1) Then
    theLines = theLine.Explode

    ' Take every children...
    For Each aLine In theLines
      newRecord = dissolveFTab.AddRecord
      dissolveFTab.SetValue(theShapeField, newRecord, aLine)

      ' ...and give it the attributes of its ancestor
      For Each aField In dissolveFTab.GetFields
        If (aField.AsString <> "Shape") Then
          dissolveFTab.SetValue(aField, newRecord,
                                dissolveFTab.ReturnValue(aField, record))
        End
      End
    End
  End
End

```

```
End

    ' Remove the ancestor
    dissolveFTab.RemoveRecord(record)
End
End

'Stop editing the FTab
dissolveFTab.StopEditingWithRecovery(TRUE)

dissolveFTab.CreateIndex(theShapeField)
theView.AddTheme(FTheme.Make(dissolveFTab))

return nil
```

```

' RoadMenu.MergePolyLines
'
' Created By:  Luke Rogers
'              In pursuit of a Masters of Science
'              Forest Engineering
'              College of Forest Resources
'              University of Washington
'              Box 352100
'              Seattle, WA 98115
'
'              luke@nwgeospatial.com
'
'              October 17th, 2000
'
' Description: Merge selected features into single feature
'
' Calls:      nothing
'
' Returns:    nothing
'
'*****
theView = av.GetActiveDoc
theTheme = theView.GetEditableTheme
theFTab = theTheme.GetFTab
if (theFTab.GetSelection.Count = 0) then
    MsgBox.Info("No polylines selected.", "RoadMenu.MergePolyLines")
    return nil
end
if(theFTab.GetShapeClass.GetClassName <> "PolyLine") then
    MsgBox.Warning("We can only spline PolyLine features.", "")
    return nil
end
'*****
' Make the list to hold the points
theList = List.Make

' Begin transaction so we can undo
theFTab.BeginTransaction

For Each selrec in theFTab.GetSelection
    thePolyLine = theFTab.ReturnValue(theFTab.FindField("Shape"),selrec)
    thePoints = thePolyLine.AsList
    for each pnt in thePoints
        theList.Add(pnt)
    end
    theFTab.RemoveRecord(selrec)
end

newpolyline = PolyLine.Make(theList)
newrec = theFTab.AddRecord
theFTab.SetValue(theFTab.FindField("Shape"), newrec, newpolyline)

' End the undo

```

```
theFTab.EndTransaction

theFTab.GetSelection.ClearAll
theFTab.GetSelection.Set(newrec)
theFTab.UpdateSelection

theView.Invalidate

return nil
```

```

' Pegger.RoadMenu.Survey
'
' Created By: Luke Rogers
'             In pursuit of a Masters of Science
'             Forest Engineering
'             College of Forest Resources
'             University of Washington
'             Box 352100
'             Seattle, WA 98115
'
'             lwrogers@u.washington.edu
'
'             March 1st, 2003
'
' Description: Performs a "digital survey" of a selected
'             pegged road or all roads if none are selected.
'             The digital survey creates a "UNIT SURVEY"
'             format .pol file that can be directly imported
'             into the ROADENG road engineering package. This
'             functionality will only be enabled if three
'             conditions exist: Spatial or 3D Analyst must
'             be enabled, a valid surface theme must be
'             in the current view, and only one feature can
'             be selected in the active road theme.
'
' Calls:      Pegger.RoadMenu.Survey.Function.Azimuth
'
' Returns:    None
'
' Dependencies: Spatial Analyst or 3D Analyst
'
'*****
theView = av.GetActiveDoc

'Make sure there is only 1 active theme
if (theView.GetActiveThemes.Count <> 1) then
  MsgBox.Warning("There must be only 1 active theme to survey",
                "RoadMenu.Survey")
  return nil
end

theTheme = theView.GetActiveThemes.Get(0)

'Make sure the theme is a PolyLine
if (theTheme.GetFTab.GetShapeClass.GetClassName <> "PolyLine") then
  MsgBox.Warning("We can only survey PolyLine features.",
                "RoadMenu.Survey")
  return nil
end

'Make sure there is only 1 selected road
if (theTheme.GetFTab.GetSelection.Count <> 1) then
  if (theTheme.GetFTab.GetSelection.Count > 1) then

```

```

    MsgBox.Warning("We can only survey 1 feature at a time",
                  "RoadMenu.Survey")
elseif (theTheme.GetFTab.GetSelection.Count = 0) then
    MsgBox.Warning("No selected road found to survey",
                  "RoadMenu.Survey")
end
return nil
end
end
'*****
' Get the dictionary and the contour theme
theDict = theTheme.GetObjectTag
theSurveyId = theDict.Get("SurveyId")
theSurfaceTheme = theDict.Get("SurfaceTheme")
theSideShotDist = theDict.Get("SideShotDist")
theSideShotNum = theDict.Get("SideShotNum")
densify = theDict.Get("Densify")
theDensity = theDict.Get("DensifyDistance")
drawOnScreen = theDict.Get("DrawOnScreen")
theOutFile = theDict.Get("SurveyFile")
'*****
'Set additional parameters from dictionary variables
if (theSurfaceTheme.GetClass.GetClassName = "STheme") then
    theSurface = theSurfaceTheme.GetSurface
    theSurfaceType = "STheme"
else
    theSurface = theSurfaceTheme.GetGrid
    theSurfaceType = "GTheme"
end

theOutFile = theOutFile.AsFileName
theSsDistance = theSideShotDist.AsNumber
theNumSs = theSideShotNum.AsNumber
'*****
'Get the selected road
theFTab = theTheme.GetFTab
theShapeField = theFTab.FindField("Shape")
theBitmap = theFTab.GetSelection
theRec = theBitmap.GetNextSet(-1)
thePolyLine = theFTab.ReturnValue(theShapeField, theRec)
'*****
'Densify survey points?
if ((densify) and (theDensity.IsNumber)) then
    thePointList = thePolyLine.ReturnDensified(theDensity.AsNumber)
    thePointList = thePointList.AsMultiPoint.AsList
else
    thePointList = thePolyLine.AsMultiPoint.AsList
end
'*****
'Make a list to store the survey points
theOutList = List.Make
'*****
' Number the side shots
theNumPoints = thePointList.Count

```

```

theSsId = theNumPoints
'*****
' Initialize the status bar
av.ClearMsg
av.ClearStatus
av.ShowMsg("Surveying " + theSurveyId + "...")
av.SetStatus(0)
'*****
'Loop through the points and get FS, BS and SD information
i = 0
for each pt in thePointList
  ' Update the status bar
  progress = (i/theNumPoints) * 100
  av.SetStatus(progress)
  theOutString = ""
  pt1 = thePointList.Get(i)
  'Draw the survey points on the screen
  if (drawOnScreen) then
    theView.GetGraphics.Add(GraphicShape.Make(pt1))
  end
  x1 = pt1.GetX.SetFormat("d.dd")
  y1 = pt1.GetY.SetFormat("d.dd")
  if (theSurfaceType = "STheme") then
    z1 = theSurface.Elevation(pt1).SetFormat("d.dd")
  else
    z1 = theSurface.CellValue(pt1, Prj.MakeNull).SetFormat("d.dd")
  end
'*****
  if (i = 0) then ' Starting point
    ' Get the next point
    pt2 = thePointList.Get(i + 1)
    x2 = pt2.GetX.SetFormat("d.dd")
    y2 = pt2.GetY.SetFormat("d.dd")
    if (theSurfaceType = "STheme") then
      z2 = theSurface.Elevation(pt2).SetFormat("d.dd")
    else
      z2 = theSurface.CellValue(pt2, Prj.MakeNull).SetFormat("d.dd")
    end
    ' Starting Reference
    theOutList.Add(theSurveyId.AsString + ",SR," + x1.AsString + "," +
      y1.AsString + "," + z1.AsString)

    ' Starting sideshots perpendicular to line
    for each n in 1 .. theNumSs
      d = theSsDistance * n
      theAzimuth = av.run("Pegger.RoadMenu.Survey.Function.Azimuth",
        {x1,y1,x2,y2})
      xSs = x1 + (d * ((theAzimuth + 90).AsRadians.Sin))
      ySs = y1 + (d * ((theAzimuth + 90).AsRadians.Cos))
      ptSs = Point.Make(xSs, ySs)
      if (theSurfaceType = "STheme") then
        zSs = theSurface.Elevation(ptSs).SetFormat("d.dd")
      else

```

```

        zSs = theSurface.CellValue(ptSs,
Prj.MakeNull).SetFormat("d.dd")
    end

    'Draw the survey points on the screen
    if (drawOnScreen) then
        theView.GetGraphics.Add(GraphicShape.Make(ptSs))
    end

    'Get the azimuth of the point from the Station
    theAzimuth = av.run("Pegger.RoadMenu.Survey.Function.Azimuth",
        {x1,y1,xSs,ySs})

    theOutList.Add(theSurveyId.AsString + ",SD," + i.AsString +
        "," + theSsId.AsString + "," +
        theAzimuth.AsString + "," + zSs.AsString +
        "," + d.AsString)

    ' Increment the side shot Id
    theSsId = theSsId + 1

    theAzimuth = av.run("Pegger.RoadMenu.Survey.Function.Azimuth",
        {x1,y1,x2,y2})
    xSs = x1 + (d * ((theAzimuth - 90).AsRadians.Sin))
    ySs = y1 + (d * ((theAzimuth - 90).AsRadians.Cos))
    ptSs = Point.Make(xSs, ySs)
    if (theSurfaceType = "STheme") then
        zSs = theSurface.Elevation(ptSs).SetFormat("d.dd")
    else
        zSs = theSurface.CellValue(ptSs,Prj.MakeNull).SetFormat("d.dd")
    end
    end
    'Draw the survey points on the screen?
    if (drawOnScreen) then
        theView.GetGraphics.Add(GraphicShape.Make(ptSs))
    end

    'Get the azimuth of the point from the Station
    theAzimuth = av.run("Pegger.RoadMenu.Survey.Function.Azimuth",
        {x1,y1,xSs,ySs})

    theOutList.Add(theSurveyId.AsString + ",SD," + i.AsString +
        "," + theSsId.AsString + "," +
        theAzimuth.AsString + "," + zSs.AsString +
        "," + d.AsString)

    ' Increment the side shot Id
    theSsId = theSsId + 1

end

'Starting foresight
theAzimuth = av.run("Pegger.RoadMenu.Survey.Function.Azimuth",
    {x1,y1,x2,y2})

```

```

theOutList.Add(theSurveyId.AsString + ",FS," + i.AsString +
              ", " + (i + 1).AsString + ", " +
              theAzimuth.AsString + ", " + z1.AsString +
              ", " + pt1.Distance(pt2).AsString)
'*****
' Last point so get perpendicular sideshots and no foresight
elseif (i = (thePointList.Count - 1)) then
  ' Get the last point
  pt0 = thePointList.Get(i - 1)
  x0 = pt0.GetX.SetFormat("d.dd")
  y0 = pt0.GetY.SetFormat("d.dd")
  if (theSurfaceType = "STheme") then
    z0 = theSurface.Elevation(pt0).SetFormat("d.dd")
  else
    z0 = theSurface.CellValue(pt0, Prj.MakeNull).SetFormat("d.dd")
  end

  ' Backsight
  theAzimuth = av.run("Pegger.RoadMenu.Survey.Function.Azimuth",
                    {x1,y1,x0,y0})
  theOutList.Add(theSurveyId.AsString + ",BS," + i.AsString +
                ", " + (i - 1).AsString + ", " +
                theAzimuth.AsString + ", " + z1.AsString +
                ", " + pt1.Distance(pt0).AsString)

  ' Ending sideshots perpendicular to line
  for each n in 1 .. theNumSs
    d = theSsDistance * n
    theAzimuth = av.run("Pegger.RoadMenu.Survey.Function.Azimuth",
                      {x1,y1,x0,y0})
    xSs = x1 + (d * ((theAzimuth + 90).AsRadians.Sin))
    ySs = y1 + (d * ((theAzimuth + 90).AsRadians.Cos))
    ptSs = Point.Make(xSs, ySs)
    if (theSurfaceType = "STheme") then
      zSs = theSurface.Elevation(ptSs).SetFormat("d.dd")
    else
      zSs = theSurface.CellValue(ptSs,
Prj.MakeNull).SetFormat("d.dd")
    end

    'Draw the survey points on the screen?
    if (drawOnScreen) then
      theView.GetGraphics.Add(GraphicShape.Make(ptSs))
    end

    'Get the azimuth of the point from the Station
    theAzimuth = av.run("Pegger.RoadMenu.Survey.Function.Azimuth",
                      {x1,y1,xSs,ySs})

    theOutList.Add(theSurveyId.AsString + ",SD," + i.AsString +
                  ", " + theSsId.AsString + ", " +
                  theAzimuth.AsString + ", " + zSs.AsString +
                  ", " + d.AsString)

```

```

' Increment the side shot Id
theSsId = theSsId + 1

theAzimuth = av.run("Pegger.RoadMenu.Survey.Function.Azimuth",
                    {x1,y1,x0,y0})
xSs = x1 + (d * ((theAzimuth - 90).AsRadians.Sin))
ySs = y1 + (d * ((theAzimuth - 90).AsRadians.Cos))
ptSs = Point.Make(xSs, ySs)
if (theSurfaceType = "STheme") then
    zSs = theSurface.Elevation(ptSs).SetFormat("d.dd")
else
    zSs = theSurface.CellValue(ptSs,
Prj.MakeNull).SetFormat("d.dd")
end

'Draw the survey points on the screen?
if (drawOnScreen) then
    theView.GetGraphics.Add(GraphicShape.Make(ptSs))
end

'Get the azimuth of the point from the Station
theAzimuth = av.run("Pegger.RoadMenu.Survey.Function.Azimuth",
                    {x1,y1,xSs,ySs})

theOutList.Add(theSurveyId.AsString + ",SD," + i.AsString +
               "," + theSsId.AsString + "," +
               theAzimuth.AsString + "," + zSs.AsString +
               "," + d.AsString)

' Increment the side shot Id
theSsId = theSsId + 1

end
'*****
else
' Intermediate point so get foresight and backsight
' Get the last point
pt0 = thePointList.Get(i - 1)
x0 = pt0.GetX.SetFormat("d.dd")
y0 = pt0.GetY.SetFormat("d.dd")
if (theSurfaceType = "STheme") then
    z0 = theSurface.Elevation(pt0).SetFormat("d.dd")
else
    z0 = theSurface.CellValue(pt0, Prj.MakeNull).SetFormat("d.dd")
end
' Get the next point
pt2 = thePointList.Get(i + 1)
x2 = pt2.GetX.SetFormat("d.dd")
y2 = pt2.GetY.SetFormat("d.dd")
if (theSurfaceType = "STheme") then
    z2 = theSurface.Elevation(pt2).SetFormat("d.dd")
else

```

```

    z2 = theSurface.CellValue(pt2, Prj.MakeNull).SetFormat("d.dd")
end

' Backsight
' Get the azimuth backsight
    theAzimuth = av.run("Pegger.RoadMenu.Survey.Function.Azimuth",
        {x1,y1,x0,y0})
theOutList.Add(theSurveyId.AsString + ",BS," + i.AsString + "," +
    (i - 1).AsString + "," + theAzimuth.AsString +
    "," + z1.AsString + "," +
    pt1.Distance(pt0).AsString)

' Sideshots bisect the angle between backsight and foresight
for each n in 1 .. theNumSs
    d = theSsDistance * n

    ' Get the X & Y components of the vectors
    vBsx = x0 - x1
    vBsy = y0 - y1
    vFsx = x2 - x1
    vFsy = y2 - y1
    ' Get the magnitudes of the vectors
    vBs = ((vBsx^2) + (vBsy^2)).Sqrt
    vFs = ((vFsx^2) + (vFsy^2)).Sqrt
    ' Get the components of the unit vectors
    uvBsx = vBsx / vBs
    uvBsy = vBsy / vBs
    uvFsx = vFsx / vFs
    uvFsy = vFsy / vFs
    ' Add the unit vectors to get bisector
    bvX = uvBsx + uvFsx
    bvY = uvBsy + uvFsy
    ' Get the magnitude of the bisector
    vB = ((bvX^2) + (bvY^2)).Sqrt
    ' Get the components of the bisector unit vector
    uvBx = bvX / vB
    uvBy = bvY / vB
    ' Multiply the components by distance
    cvX = uvBx * d
    cvY = uvBy * d

    ' If the magnitude is small then not a perpendicular side shot
    if (vB > (0.0001 * pt0.Distance(pt2))) then
        ' Add the components to x1, y1
        xSs = x1 + cvX
        ySs = y1 + cvY
    else 'it is perpendicular
        theAzimuth = av.run("Pegger.RoadMenu.Survey.Function.Azimuth",
            {x1,y1,x0,y0})
        xSs = x1 + (d * ((theAzimuth + 90).AsRadians.Sin))
        ySs = y1 + (d * ((theAzimuth + 90).AsRadians.Cos))
    end
end

```

```

ptSs = Point.Make(xSs, ySs)
if (theSurfaceType = "STheme") then
  zSs = theSurface.Elevation(ptSs).SetFormat("d.dd")
else
  zSs = theSurface.CellValue(ptSs,
Prj.MakeNull).SetFormat("d.dd")
end

'Draw the survey points on the screen?
if (drawOnScreen) then
  theView.GetGraphics.Add(GraphicShape.Make(ptSs))
end

' Get the azimuth of the point from the Station
theAzimuth = av.run("Pegger.RoadMenu.Survey.Function.Azimuth",
  {x1,y1,xSs,ySs})

theOutList.Add(theSurveyId.AsString + ",SD," + i.AsString +
  "," + theSsId.AsString + "," +
  theAzimuth.AsString + "," + zSs.AsString +
  "," + d.AsString)

' Increment the side shot Id
theSsId = theSsId + 1

' If the magnitude is small then not a perpendicular side shot
if (vB > (0.0001 * pt0.Distance(pt2))) then
  ' Subtract the components from x1, y1
  xSs = x1 - cvX
  ySs = y1 - cvY
else 'it is perpendicular
  theAzimuth = av.run("Pegger.RoadMenu.Survey.Function.Azimuth",
    {x1,y1,x0,y0})
  xSs = x1 - (d * ((theAzimuth + 90).AsRadians.Sin))
  ySs = y1 - (d * ((theAzimuth + 90).AsRadians.Cos))
end

ptSs = Point.Make(xSs, ySs)
if (theSurfaceType = "STheme") then
  zSs = theSurface.Elevation(ptSs).SetFormat("d.dd")
else
  zSs = theSurface.CellValue(ptSs,
Prj.MakeNull).SetFormat("d.dd")
end

'Draw the survey points on the screen?
if (drawOnScreen) then
  theView.GetGraphics.Add(GraphicShape.Make(ptSs))
end

' Get the azimuth of the point from the Station
theAzimuth = av.run("Pegger.RoadMenu.Survey.Function.Azimuth",
  {x1,y1,xSs,ySs})

```

```

    theOutList.Add(theSurveyId.AsString + ",SD," + i.AsString +
        "," + theSsId.AsString + "," +
        theAzimuth.AsString + "," + zSs.AsString +
        "," + d.AsString)

    ' Increment the side shot Id
    theSsId = theSsId + 1

end

    ' Foresight
    theAzimuth = av.run("Pegger.RoadMenu.Survey.Function.Azimuth",
        {x1,y1,x2,y2})
    theOutList.Add(theSurveyId.AsString + ",FS," + i.AsString +
        "," + (i + 1).AsString + "," +
        theAzimuth.AsString + "," + z1.AsString +
        "," + pt1.Distance(pt2).AsString)

end

    i = i + 1
end
*****
'Make output file by writing list to theOutFile
theLineFile = LineFile.Make(theOutFile, #FILE_PERM_WRITE)
theLineFile.Write(theOutList, theOutList.Count)
theLineFile.Close
*****
'Clear the status bar
av.ClearMsg
av.ClearStatus

return nil

```

```

' Pegger.RoadMenu.Survey.Function.Azimuth
'
' Created By:  Luke Rogers
'              In pursuit of a Masters of Science
'              Forest Engineering
'              College of Forest Resources
'              University of Washington
'              Box 352100
'              Seattle, WA 98195
'
'              lwrogers@u.washington.edu
'
'              December 11th, 2004
'
' Description: Calculates the angle created by two points.
'              Based on original work by David F. Kimball.
'
' Calls:      Nothing
'
' Returns:    Returns the angle (azimuth) formed by the two
'              points as a number from 0 - 360.
'
'*****
'-----
' the script must be passed a list containing either 2 pts or 4 numbers
' Sample script call:  av.Run("ReturnLineSegmentAzimuth",{1,1,6,9})
'                    or   av.Run("ReturnLineSegmentAzimuth",{1@1,6@9})
'-----
if (SELF.Count = 4) then
  x1 = SELF.Get(0)           'xcoord of first (origin) point
  y1 = SELF.Get(1)           'ycoord of first (origin) point
  x2 = SELF.Get(2)           'xcoord of second point
  y2 = SELF.Get(3)           'ycoord of second point
  p1 = Point.Make(x1,y1)
  p2 = Point.Make(x2,y2)
elseif (SELF.Count = 2) then
  p1 = SELF.Get(0)           'first (origin) point
  p2 = SELF.Get(1)           'second point
  x1 = p1.GetX
  y1 = p1.GetY
  x2 = p2.GetX
  y2 = p2.GetY
else
  return nil
end
'-----
' calculate the angle using simple trig:
'-----
h = p1.Distance(p2)         'h = the distance between the pts
dX = x2 - x1                 'dX = difference in xcoords
dY = y2 - y1                 'dY = difference in ycoords
a = 90 - ((dX / h).ACos.AsDegrees) 'a = the angle made by the points
if (dX < 0) then

```

```
    if (dY < 0) then
      a = 180 + a.Negate
    else
      a = 360 + a
    end
  else
    if (dY < 0) then
      a = 180 - a
    end
  end
end
'-----
'return the angle as a number between 0 and 360:
'-----
return a
```