Geography 356 Final Project

This final project is designed to emulate the type of project work you might find in a “real job”. Basically you will be asked to design and implement an application using ArcView and Avenue. You will need to divide up into groups of 2-4 people. Each group will decide on a project, and then begin working on a requirement study as outlined in lecture 3, and chapter 1 of your book. The deliverables for the requirement study are outlined below. Having completed the requirements study, you will generate a prototype interface, and a detailed outline of how the application will work. You will then completely implement the application, and wrap up your work by generating a users’ guide. The final portion of the project will be an oral presentation to the class. This presentation will be based on a final report that will reside on a web site.

1. Requirements Study (1000 words) 15%

A key aspect of the project is to prepare a requirements analysis to facilitate design and production of your application. The requirements analysis will be completed before you generate any Avenue scripts and (with appropriate modifications) will also be included in the user’s guide (described below) to be submitted with the completed application.

1.1: Introduction, Target Users, and Conceptual Level Goals

The requirements analysis should begin with an introduction to your application and its planned capabilities. This introduction should include (1) the "problem" your application is designed to address (e.g., visual analysis of geographic patterns in socioeconomic data depicted with classed choropleth maps), (2) some general information about the mapping and/or geographic analysis problem category that your application is suitable for (e.g., for choropleth maps, discussion of the kinds of data they are appropriate for and the decisions involved in their creation), (3) a discussion of the intended users (e.g., employees of the census bureau with backgrounds in demographics and statistics) (4) description of the specific planned application (e.g., how information will be presented to users, what they will be able to do with it, what kinds of options they will be able to compare, etc.), and (5) guidelines for appropriate use of the application (e.g., ways in which the tools you provide can facilitate pattern analysis, lead to intelligent data classification choices, and/or an understanding of the implications of classification).

1.2: Operational Level Functions

Here, you should list the specific operations you plan to implement and describe how these operations, either individually or together, will meet each conceptual level goal

2. Prototype 15%

2.1: System diagram, explanation, and convention

Prepare a detailed diagram of your application that shows all ArcView Document objects (e.g. Projects, Views, Scripts, Tables, etc.) that you will use along with the scripts to be associated with each and their connection. This diagram can be drawn by hand or with
graphic design tools (e.g., Freehand) but must be neat and carefully labeled. The diagram should be accompanied by a set of detailed (and typed) footnotes that describe the inputs, functions, and outputs of each planned script.

Establish a set of conventions for use throughout the application. Prepare (and attach to your diagram) a complete list of these conventions -- and follow them. Conventions should include (but are not limited to) (a) a standard format for introductory comments in each script, (b) rules for assigning object names so that the kind of object will always be clear and names do not get confused, and (c) a standard system for assigning file names for input, output, and project files.

2.2: Prototype interface

At this stage, you should have made decisions about the components of your application and how they will be organized in a set of scripts. Your next step is to make decisions about the application's interface (e.g., what kinds of buttons and/or menus will be available at which points during use of the application, how users will access data, what aspects of analysis and display will be under user control, etc.

Before proceeding with the bulk of Script writing, you will first build a prototype interface. Proceed with this step by creating an initial project and customizing the interface for the Project, View, and other windows that users will access. This customization will probably involve hiding some system menus, buttons, and tools and adding menus, button, or tools that will access the special features you will ultimately create. Remember to make system options invisible (instead of deleting them) in case you decide later that you want users to have access to them. This prototype should present the set of available controls that you want the user to have at each stage - with dummy scripts (that say "under construction" or something similar) in those places where you will add your own scripts.

3. Implementation - 40%

3.1. Construction

Once you have completed a prototype of your interface, work through the steps that you expect a user to follow thinking about any places where confusion might occur, additional features might be desired, etc. Modify your plans for the application accordingly and begin preparing scripts.

In constructing the full application, you will have the highest probability for success, and the least frustration, if you follow a few simple rules:

Separate distinct operations in different scripts.

List the conventions you established above as comments in the main application Script – and follow them.

Insert at least major section headings as comments as you go so that it is possible for someone else (e.g., me, another group member, or even you 2 weeks later) to figure out what you were trying to do.
Describe, in written form, the complete operation of each script (and how it links to other scripts) before beginning to write it. [This should appear as a comment at the beginning of the script]

Test each script independently before trying to test them together.

Make use of Avenue's script debugging tools to find errors.

If you are baffled about why a script does not work, ask your other group members to help figure it out (fresh eyes often see errors immediately that can remain hidden for the program author indefinitely). If that fails, ask others in the class to take a look, or ask me.

3.2 Structured Testing

Plan your construction schedule to allow enough time to test each script and the entire application thoroughly. Be sure to test the unanticipated! Do things wrong on purpose to see what happens. Enter parameters that are extreme as well as ones you expect users to enter. Try to make the system crash. If you cannot solve a problem you uncover, insert a warning message telling the user what input to avoid.

I recommend that you even try writing a script similar to the one outlined in lecture 3. This script would contain a series of run requests that run your different scripts in different orders to see if you can make the system crash.

4. Documentation 10%

The scripts in your application must be documented sufficiently to allow another person with programming experience to follow the program logic and make modifications or additions. The documentation will consist of a header for each script as outlined in lecture 4 and comment lines that explain what each section of code does.

5. User's Guide – 1500 words - 15%

Along with your application, you will submit a user's guide to help others use your finished product. This user's guide should include the following.

An introduction modified from section 1, your requirements analysis (incorporating suggestions provided to you).

A revised diagram of system operation from section 2.1

A section on input files required (or allowed). Describe the specific data structures required (or allowed) with sufficient detail that the user could create (or locate) his or her own input files for use with the program.

A section covering the processing and output options in detail.

In describing the data processing and output options that your program is capable of, list the file names needed to run the example and describe in a step by step manner what the user will be required to do in order to use the application. For each major processing component discuss the choices that the user will be presented with, the implications of
those choices, and anything else that the user should know in order to make intelligent use of the application. For each kind of response that the user must make, explain the possible responses and the outcome expected with different possible answers [e.g., range of possible data classification options and the likely appearance of a map].

Include visual examples (maps, tables, statistics, etc.) of the different kinds of output as part of the guide. Include several examples for each option that has choices that can be made by the user.

6. Class Presentation – 5%

Your class presentation will consist of a brief description of the requirements study, and then a demonstration of the application. The presentation will be based on a web site that your team will put together. The web site should showcase your work on the project, including your requirement study, screen-shots of your interface, copies of all scripts, and your user’s guide. The website should have a main page that briefly describes your project, and has links to the information noted above.
Important DATES:
Sept. 28. Final project assigned.
Oct. 3. Final project groups and preliminary proposals due.
Nov. 13: Final project update – you should have at least half of your implementation completed by this date. This will be a graded update worth 10% of your lab grade.
Nov. 27. Final project update – you should be practically done with the implementation of your project by this date, with only a few bugs left to work out. This will be a graded update worth 10% of your lab grade.
Nov. 30. User’s guide due. (5:00 pm)
Dec. 4, 5, 7. Class presentations.

Suggestions:
• Start immediately!
• Be thorough in your requirements analysis and prototype.
• Build your application in components [individual scripts], being sure that each component works before going onto the next.
• Select options in such a way that you can fall back on an easier option if the one you wanted to do proves too difficult.
• Rough out your User's Guide as soon as you complete the Prototype, so that you can add necessary details to it as you complete each part of your program rather than trying to do it all in a hurry at the last minute. Grading
• It is possible to do a good job on the requirements analysis, prototype, documentation and users guide even if your application does not work perfectly. These items are worth 60% of the project grade. Be sure to turn in everything that you finish whether it works perfectly or not.