Lab 2: Your First Real Script

While looking over the labs from last years’ class, I decide that I’d changed the lectures so much that I should really write new labs for the class as well. Therefore, this will be the first in a series of new, and hopefully improved labs (although Rob’s labs were pretty good).

Instead of focusing on ArcView in this lab, we’re going to jump right into Avenue. I am going to write a description of a script, and you will have to write it. This may seem pretty overwhelming at first, but if you learn this stuff early in the semester instead of putzing around with basic ArcView stuff, you will end up doing much better on your final projects. If the lab ends up being really difficult, I will extend the deadline a week, but for now I want you to assume that this lab is due one week from the day it is assigned.

The product of this lab will be a working script. I expect this script to be fully documented as outlined in Lecture 4. This does NOT mean you need to write a users guide. Just document the code internally. Here is a brief description of what the script will need to do:

The script that you write will allow the user to turn on and off multiple themes in a view using a dialog box that lists all of the themes and allows you to select the ones that you want to turn on.

Seems pretty simple, huh?

Alright… maybe it seems a little overwhelming, but let’s try to break it down into smaller pieces and go from there. I’ll try to outline the steps in plain English using terms that I’ve defined in the lecture, and you can try to convert that into Avenue script.

One last note… I encourage you to work together on these labs. Just don’t copy one another’s code. If you’re working with someone, make sure that both of you are doing the coding yourselves, and understand why you are doing what you do. I will not accept code that is blatantly copied from another student! If you are copying code and don’t understand the concepts, it’s going to come back and bite you on the tests.

Step 1:

To begin with, you need a project to work with. There are a number of projects that come with the ArcView tutorials that you can use, but feel free to use any project that you have. The only requirement is that it has a view with four or more themes. The ArcView tutorial projects can be found by opening ArcView, and then clicking on file: open project. When the “Open Project” dialog box opens up it should be in the following directory: \esri\av_gis30\avtutor. One of the subdirectories in this directory is arcview. There is a project file in that subdirectory called qstart.apr. There are other project files in other subdirectories under avtutor. Feel free to use any of those as well. Once you’ve chosen a project to work with, use the “save as” command to save the file to your working directory on the j: drive. Now you’re ready to start writing your script! Oh boy! Don’t ya just love my enthusiasm?!!!
Step 2:

Create a new script, and begin your documentation. You can do this by double clicking on the “Scripts” icon in the project window. Once you’ve got your script, rename it “My First Script” (you can do this by going to the scripts:properties menu when you in the “script” document, or by going to the edit:rename menu when you’re in the “project” window). Now begin your documentation. Write your header, and then skim through the following steps. The first sentence of each step tells you what you’ll be doing there. Add these as a series of comments in your script. Once you’ve done this, you’ll have a complete outline of what your script will do.

Step 3:

Get the view. One of the first things that you need to do when you start coding is to figure out which objects you’re going to need, and how to go about getting them. As I mentioned in lecture 6, the application object (represented with the keyword av) is the root of the object tree. Almost all the objects that you will need can be accessed through the application object. You can get the view from the application object in a number of ways. You can either find it by name (as you can find any doc in a project), or you can just assume that it will be the active window and get the active theme (which is a subclass of document). By the way, you’re going to want to reference the view with a variable, so remember to add that variable name and description to the list in your header file.

Compile and run the script to see if you get any errors. If it works correctly, it won’t seem to do anything yet. If you did something wrong, it will give you an error message.

Hints:

1) Taking a look at the object model diagrams for both the application class and the view class might help you here. Remember that you can find help on each class by clicking on its “box” in an object model diagram. 2) The other thing that you might find to be incredible useful is to find a system script that works with a view document (they all fit the pattern View.*).

Step 4:

Get the themes. You now have a variable referencing the view. You need to get a list of the themes from that view and set that list to a variable. See if you can guess the syntax for the request that you need to use to get the themes from a view. Try to compile and run your code again.

Hints:

1) Remember that requests are often composed of actions and properties concatenated together. 2) If your code won’t compile and run properly it may be because you are using the GetActiveDoc request, and a view is not the active document. When you run a script from the script document, it picks the last document that you were in as the active document. Therefore, you need to go to a view, and then go back to the script document in order for the script to run on the view document.
**Step 5:**

*Display the themes in a message box.* See if you can find the message box class. Remember that you call a message box by making requests to the *class* rather than to an object. Use the help to try to determine what request will give you a message box that allows you to display a list of themes as strings, and to select multiple themes. Remember that the message box request should return a list of themes that the user wants to turn on. You will need to reference this list with a variable.

**Hints:**

1) Try searching the help topics to get information about a specific class. Sometimes you won’t find everything you need to know the first time you search. In that case look through what you *did* find for other key words that you might be able to search on.

**Step 6:**

*Loop through all themes in the view.* Now you should have two lists. What your going to want to do is loop through all the themes in the list, make some of them visible, and turn others off. In this step we will just worry about the first step; looping through all the themes in the view. So think about the fact that you are going to have to execute a block of code for each theme in the list of all themes.

**Hints:**

1) Try looking up looping, or something like that in either your book, or in the ArcView help. 2) Remember that when you use the loop your going to need a variable to reference the theme that you are working with in each iteration of the loop. Be sure that you list this variable in your header. 3) Remember to compile and run your code again.

**Step 7:**

*With each iteration of the loop, test to see whether or not the current theme is in the list of themes that should be turned on.* Within the loop, you now need to figure out what to do with the theme that you are looking at. If the theme is in the list, you will want to do one thing with it (make it visible). If it is not in the list you will want to do something else (make sure it is not visible).

There are really two parts to this step. The first is to set up the structure to that allows the code to do one thing in one case and something else when that condition does not exist. The second part is to figure out how to write the conditional statement. I assume you will all be able to do the first part fairly easily. The second part is a little harder, though. Avenue does not give you a request that allows you to ask, “Is object1 in list1?” Instead, you need to try to find the object in the list. If the object cannot be found in the list, the condition should evaluate to false.

**Hints:**

1) To find out how to set up the conditional structure, look up program flow control and conditional statements in lecture 6. 2) Look up the List class in help. Look for methods that search for an object in a list, and then figure out what those methods do if the object isn’t there. 3) Remember that you can compile and run your program to see if it is giving you any obvious errors.
Step 8:

Make themes in the selected list visible, and make the themes not in the list not visible.

This is the last step! You now have your structure that allows one thing to happen if your conditional statement evaluates to true, and something else to happen when it evaluates to false. In one of those cases you should make the theme visible, in the other case you should make it not visible (I know that using the words “visible”/”not visible” may seem awkward, but it’s a hint). In order to do this you’ll need to figure out the request that will do this for you, and then make that request of the individual theme object (not the either list of themes).

Hints:

1) Look up the theme class in ArcView help. You should be able to find the method that you need there. 2) Compile and run your script again. Once it works correctly check all of your documentation. Make sure that all of the variables that you’ve used are in the header.

Conclusion:

This is just one way of accomplishing the task that I’ve set out for you in this lab. You can do it other ways as well. I don’t care how you do it, as long as it works correctly and you document it. I’ve tried to lay out this lab in a way that teaches you to learn to use the resources available to you. If you learn how to find out what you need to know, you will be able to do anything you want. However, if you just learn the syntax that we cover in this class, you’ll be limited in what you can do. Therefore, the key things that you should take away from this lab are: 1) understanding and using the object model diagrams in ArcView help, 2) understanding and using the class descriptions in ArcView Help, 3) working with your colleagues to find solutions to problems.