Using dialog boxes and objects

Chapter 3 – **Creating a dialog box**

- pp. 37-50
- **Exercise 3**

Chapter 4 – **Programming with objects**

- pp. 51-64
- **Exercises 4A& 4B**

Chapter 3 – Creating a dialog box

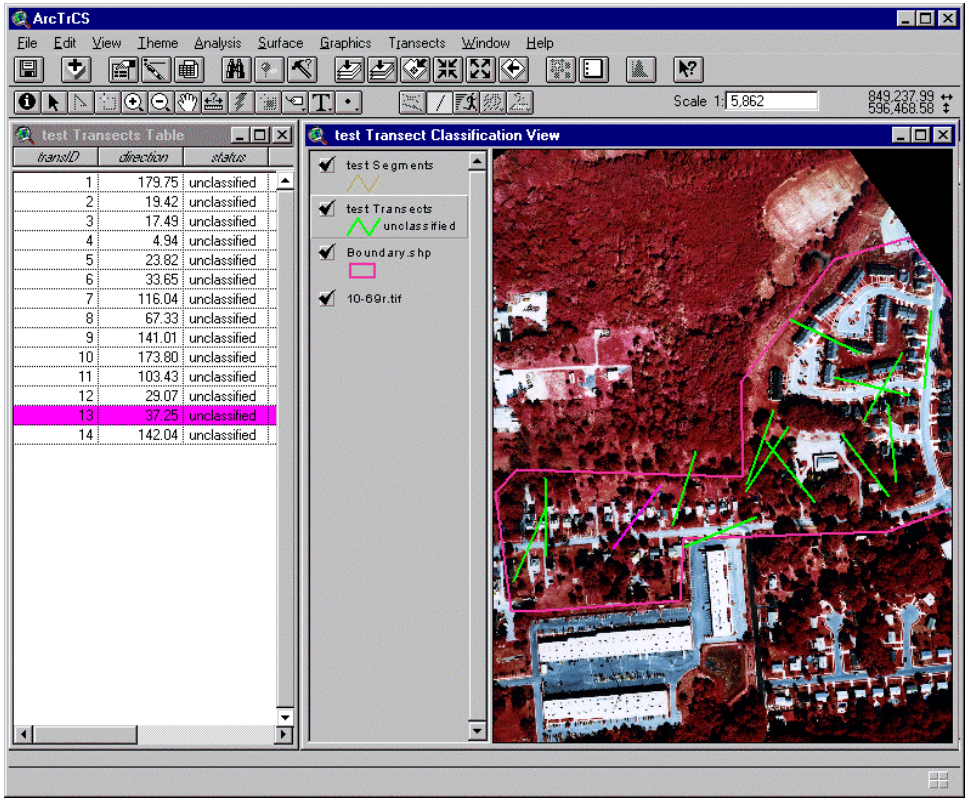
- Using controls to build a form

Chapter 3 – Creating a dialog box

- One of the things that makes **graphical user interfaces** (GUIs) so popular is how **easy** they are to use
  - With **good design**, it becomes quite **natural** for the user to interact with the software
- ArcGIS is a great **example** of this: Users spend a lot of time working with ArcGIS' GUI through the use of their mouse, with some occasional use of the keyboard
- Menus, commands, buttons, and tools call up ArcGIS' capabilities; we make use of **dialog boxes** to input information
  - As **simple** as a **MsgBox**, or as **complex** as … **custom dialog boxes**

Chapter 3 – Creating a dialog box

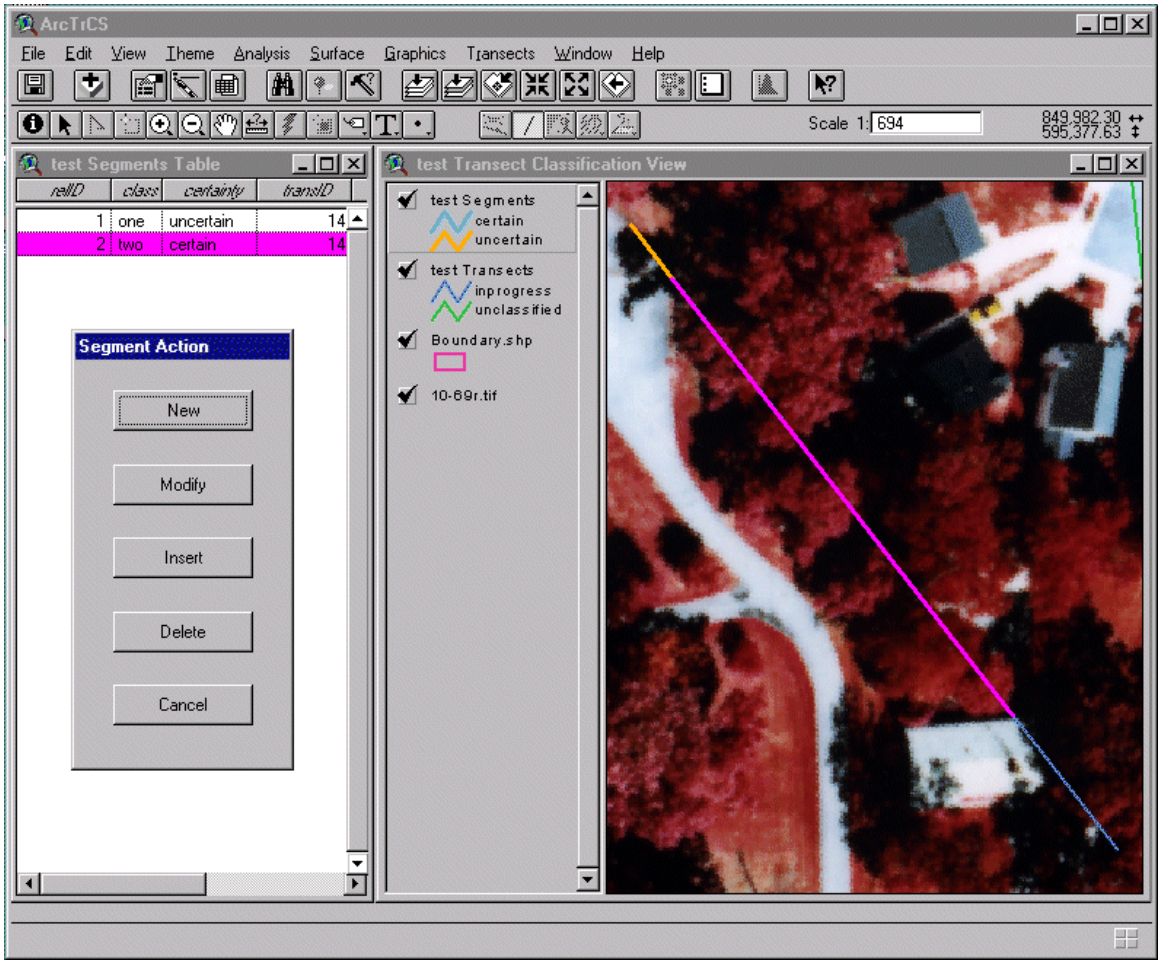
- If we are going to build **custom functionality** for ArcGIS, sometimes it makes sense to build **custom interfaces** too
  - An example: The **ArcView Transect Classification System** (ArcTrCS), a piece of software designed to help users collect land-use land-cover information by placing transects on a digital orthophotograph, and then slice them up into segments that correspond to the extent of different LULCs below the transect:



Tenenbaum, D.E., Cadenasso, M.L., Band, L.E., and S.T.A. Pickett. 2006. Using Transects to Sample Digital Orthophotography of Urbanizing Catchments to Provide Landscape Position Descriptions. *GIScience and Remote Sensing*, 43(4):323-351.

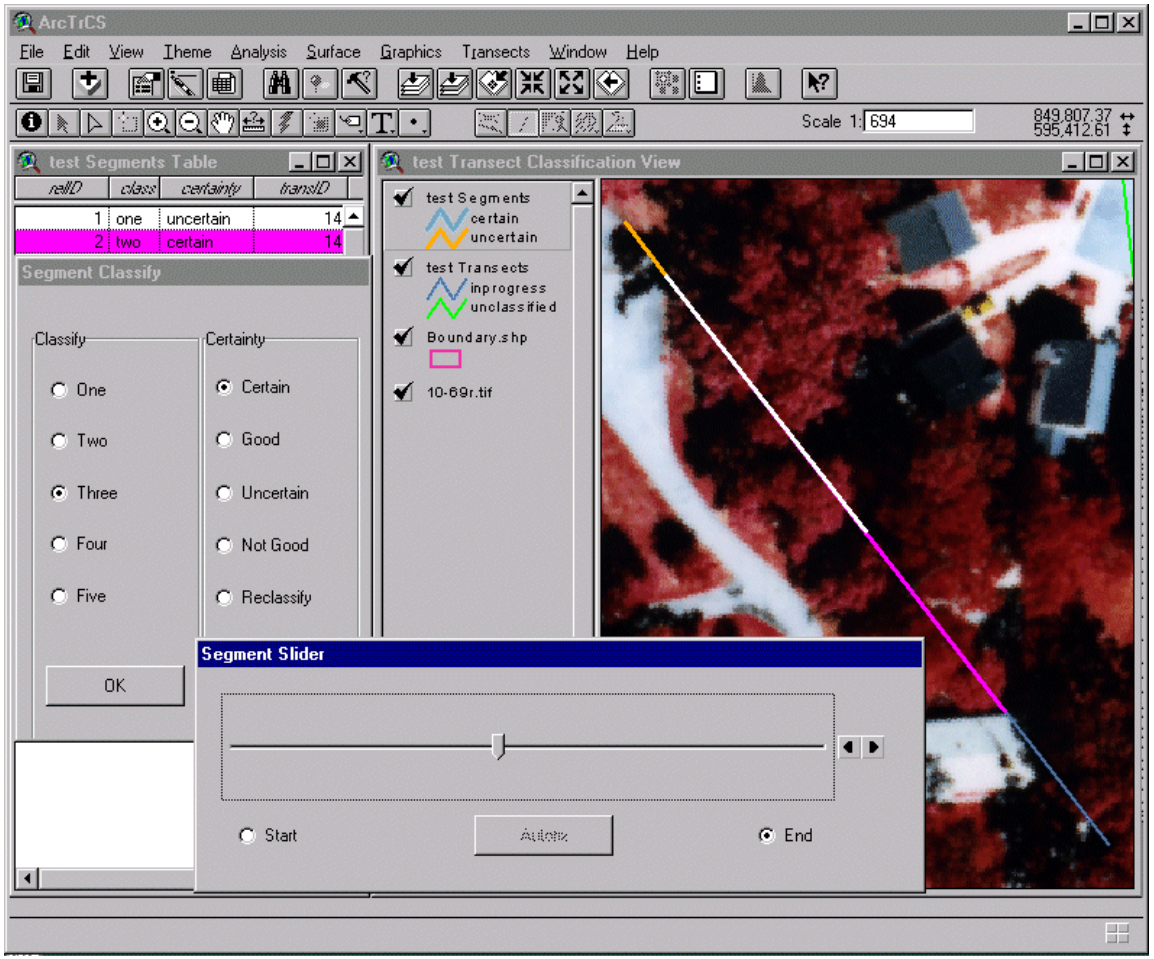
Chapter 3 – Creating a dialog box

- **Segment Action dialog**: Once a user has chosen a transect to edit, this **dialog with five buttons** pops up, allowing the user to choose what they want to do
- Depending on the state of the transect and if a segment is selected, some **options** are **available** and some **are not**



Chapter 3 – Creating a dialog box

- **Segment Classify and slider dialogs**: Once a user has chosen a segment to edit, these **two dialogs** pop up (one with a **slider**, two **radio buttons**, and a button, the other with two sets of five radio buttons), allowing the user to set the spatial extent and attributes of the segment



Using controls to build a form

- You can **build dialog boxes** to suit **whatever purpose** you have in mind, and add **whatever controls** are needed
- In VBA, these are referred to as **Forms**, or **UserForms**
- Within the **Visual Basic Editor**, we can craft a Form by **dragging and dropping controls** onto it
  - Choose from **elements/controls** such as Labels, Textboxes, ComboBoxes, Listboxes, Checkboxes, OptionButtons, ToggleButtons, CommandButtons, Images, and Frames
- Each and every element/control you add to your Form has **many properties**
  - You can set their **initial values** in the **Editor**, and control their state later through **VBA code**

Using controls to build a form

- Once you build the skeleton/appearance of a Form in the Editor, you need to **write code to make it do things** once a user clicks on controls (or performs some other action)
- This is organized into **events** and **event procedures**:
  - An **event** occurs when a **user does something** (performs an action), e.g. a user clicks on a button: Associated with that button's click event is a number of lines of code that performs some action
  - We refer to the code associated with a **given user action** as an **event procedure**, i.e. when a user clicks on a CommandButton, then the CommandButton's click event procedure runs

Chapter 4 – Programming with objects

- Programming with methods
- Getting and setting an object's properties

Chapter 4 – Programming with objects

Recall some of our important OOP terminology:

- Object - **Anything** that can be 'seen' or 'touched' **in the software programming environment** . Objects have attributes (properties) and behaviors (methods)
- Properties - Attributes are **characteristics that describe** objects
  - e.g. *Text.Font = Arial*
- Methods (behaviors) - An object's methods are operations that either the **object can perform** or that **can be performed upon** the object,
  - e.g. *Table.AddRecord*

Chapter 4 – Programming with objects

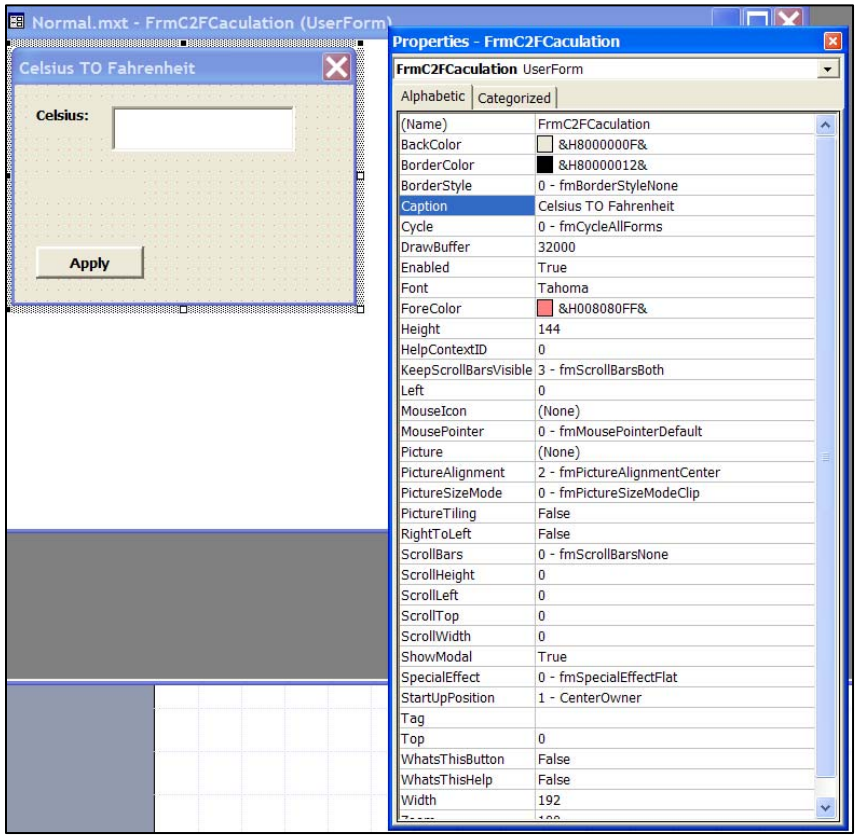
- We work with objects by **setting their properties**, and **calling their methods**, using the **"object dot property"** syntax:
  - *Object.Property*
  - This is also known as **reverse Polish notation**
- We can think of properties a little like variables, in that **they describe an object**, and we can both **get (find out their current value)** and **set (change their value to something else)** them → Properties as **adjectives**, that describe the object
- We can think of methods as **things that objects can do** → Methods as **verbs**, that make the object do something

Programming with methods

- The textbook gives a series of **examples** with a spaceship, which boil down to *Object.Method*, where the **object** is the **spaceship**, and **method** is **something we expect it could do** (*Atlantis.WarpSpeed*, *Atlantis.Shields Down* etc.). Two key things to notice:
  1. The methods in the fictitious example need to be things that the object can do. In real VBA code, the **methods** must be **defined** for an **object of that type**
  2. Some methods have **arguments**, which **specify how** to perform the method *(Atlantis.Shields Down)*, and even multiple arguments, **separated by commas** (*Atlantis.BeamUp Andrew, Thad, Michael)*

Getting and setting an object's properties

- In the Chapter 3 exercise, we will work with the properties of the controls we create, and **set them initially through the Editor interface**, where we can see all the properties of each control
- For example, here's a Form with an InputBox that will convert a value from Celsius degrees to Fahrenheit degrees:



Getting and setting an object's properties

- More important is **getting and setting properties** while our **code is running**
- To **get** a property: *variable = object.property*
- To **set** a property: *object.property = variable*
  - Here, again, is the temperature converter Form example:

```
Private Sub cmdApply_Click()
    strFTemp = (txtCelsius.Text * 9 / 5 ) + 32
   lblF.Caption = "Farenheit:" & strFTemp
End Sub
```

Next Topic:

Control of flow and modularization