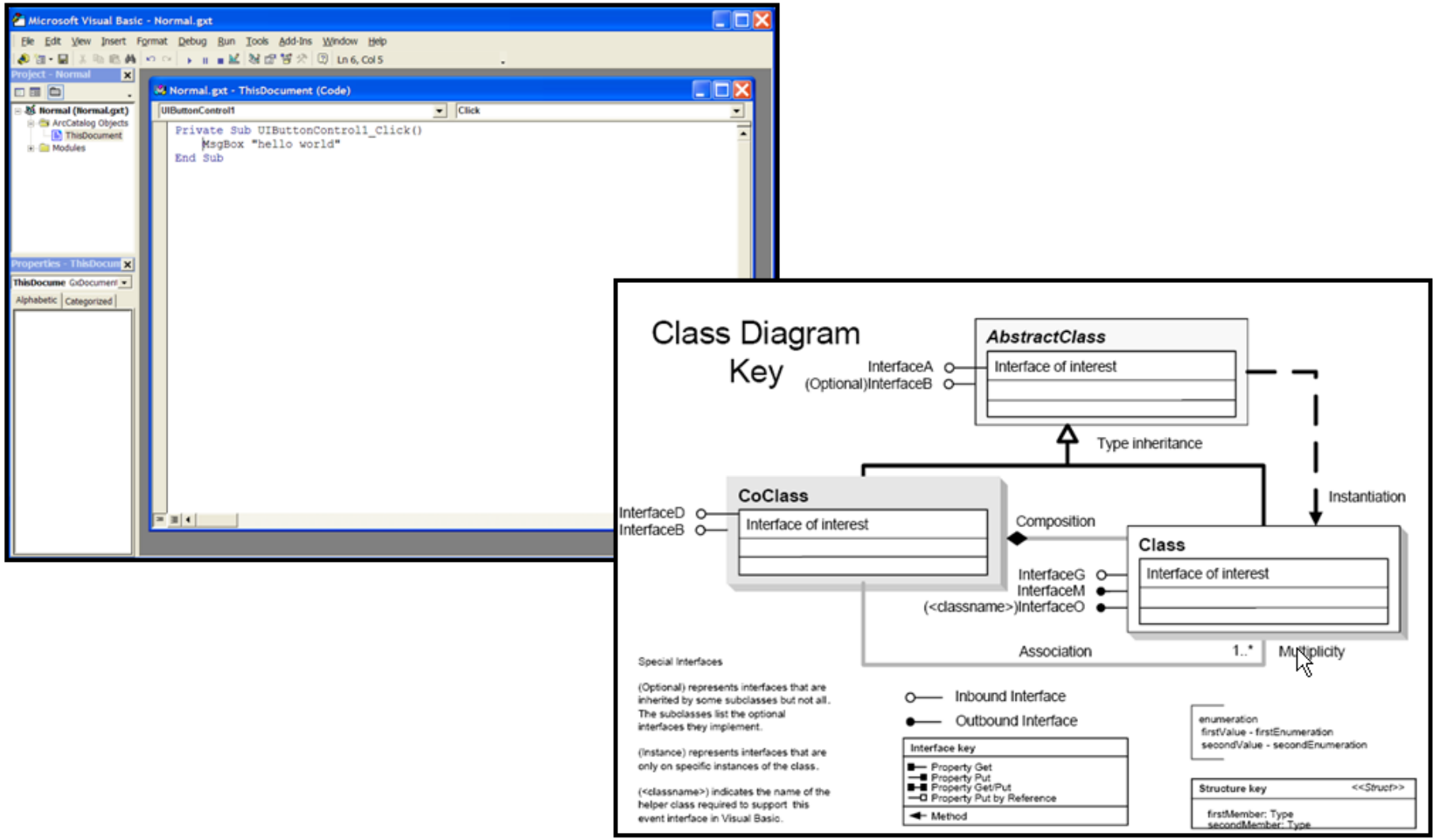
EEOS 472 – Programming for GIScience Applications



EEOS 472 – Programming for GIScience Applications

Course Description:

This course will provide students with an introduction to object-oriented programming, specifically in the context of geographic information science. Students will become familiar with objects and methods, control of flow concepts, code modularization, and debugging techniques. They will learn to customize ArcGIS, by utilizing dialog boxes and building commands, menus, and tools. By learning to navigate object diagrams and interfaces, skill will be developed in using the existing ArcObjects that form the underlying structure of ArcGIS.

EEOS 472 – Programming for GIScience Applications

Course Description:

This knowledge, accompanied by the ability to make new objects, will provide students with the required coding vocabulary to build applications to address a wide variety of GIScience applications. Students are expected to have completed introductory coursework in geographic information science, although no prior experience in programming is required.

EEOS 472 – Programming for GIScience Applications

- Students will be provided with **hands-on experience**, working with Visual Basic for Applications (VBA), which is integrated into the ArcGIS desktop geographic information system (GIS). The **goals** are to help students:

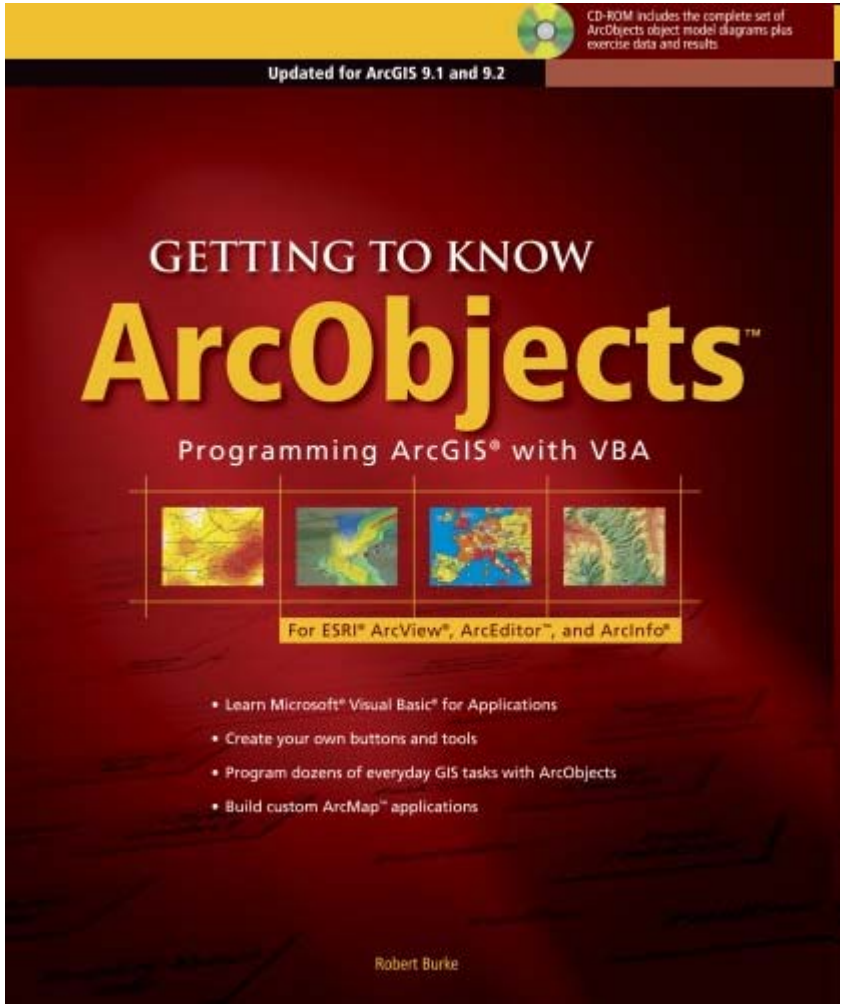
1. Understand the key concepts of **object-oriented programming**
2. Become skilled at **using VBA** to customize ArcGIS
3. Build capability and understanding in the **application of programming techniques** to GIScience applications

EEOS 472 – Programming for GIScience Applications

- Students will be expected to **read chapters before class** and be **prepared to discuss the concepts** each week
  - Class meetings will begin by focusing on the key concepts from the textbook
- This will be followed by **weekly lab exercises** designed to augment the readings and to **sequentially build capability and understanding** of using VBA to customize and program ArcGIS
  - These exercises will use the instructions from the text to provide an opportunity for hands-on learning, combined with supplementary conceptual questions to get you thinking about what you're doing

EEOS 472 – Programming for GIScience Applications

- **Text:** Robert Burke. Getting to know ArcObjects – Programming ArcGIS with VBA. ESRI Press, 2003. ISBN 1-58948-018-X.



EEOS 472 – Programming for GIScience Applications

Date	Topic	Reading and Exercises
UNDERSTANDING VISUAL BASIC FOR APPLICATIONS & ARCOBJECTS		
02/03/11	Object-oriented programming and customizing ArcGIS	•Chapter 1 – Programming with objects, pp. 1-4. •Chapter 2 – Building a custom application, pp. 5-35. **Exercises 2A, 2B & 2C**
02/10/11	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**
02/17/11	Control of flow and modularization	•Chapter 5 – Code for making decisions, pp. 65-76. **Exercises 5A & 5B** •Chapter 6 – Using subroutines and functions, pp. 77-102. **Exercises 6A, 6B, 6C & 6D**
02/24/11	Using loops and debugging code	•Chapter 7 – Looping your code, pp. 103-118. **Exercise 7A & 7B** •Chapter 8 – Fixing bugs, pp. 119-132. **Exercise 8**
03/03/11	Making objects and using interfaces	•Chapter 9 – Making your own objects, pp. 133-145. **Exercises 9A & 9B** •Chapter 10 – Programming with interfaces, pp. 147-170. **Exercises 10A & 10B**
03/10/11	Mid-term Review	N/A
03/17/11	*March Break*	*N/A*
03/24/11	Mid-term Exam	N/A

EEOS 472 – Programming for GIScience Applications

Date	Topic	Reading and Exercises
UNDERSTANDING VISUAL BASIC FOR APPLICATIONS & ARCOBJECTS		
03/31/11	The object model, UML diagrams, and making tools	•Chapter 11 – Navigating object model diagrams, pp. 171-197. **Exercises 11A & 11B** •Chapter 12 – Making tools, pp. 199-225. **Exercises 12A, 12B & 12C**
USING ARCOBJECTS		
04/07/11	Using existing commands and adding layers	•Chapter 13 – Executing commands, pp. 227-238. **Exercise 13** •Chapter 14 – Adding layers to a map, pp. 239-261. **Exercises 14A & 14B**
04/14/11	Map symbology and ArcCatalog	•Chapter 15 – Setting layer symbology, pp.263-293. **Exercises 15A, 15B & 15C** •Chapter 16 – Using ArcCatalog objects in ArcMap, pp. 295-314. **Exercises 16A & 16B**
04/21/11	Displaying and selecting features	•Chapter 17 – Controlling feature display, pp. 315-338. **Exercises 17A & 17B** •Chapter 18 – Working with selected features, pp. 339-357. **Exercises 18A & 18B**
04/28/11	Working with layouts and editing data	•Chapter 19 – Making dynamic layouts, pp. 359-376. **Exercises 19A & 19B** •Chapter 20 – Editing tables, pp. 377-401. **Exercises 20A & 20B**
05/05/11	Final Review	N/A

EEOS 472 – Programming for GIScience Applications

- **Course Web Page:**
  - http://www.faculty.umb.edu/david.tenenbaum/eeos472

- **Lab Exercises:**
  - Instructions are in the text, **supplemental instructions and questions are linked** from the course web page
  - Data will sit in a course directory on the s:\ drive
  - There be will exercises each week, and we will begin working on them during the class meetings
  - Due 1 week later at the beginning of the class meeting

- **Lateness Penalty: -10%** of total mark per weekday
  - Approach your instructor/TA for extenuating circumstances
  - Exercises more than a week overdue will not be accepted

EEOS 472 – Programming for GIScience Applications

Course Evaluation:

Lab assignments	**50%**
Mid-term (Mar. 24)	**25%**
Final exam (TBA)	**25%**

EEOS 472 – Programming for GIScience Applications

- **Class Meetings**:
  - S-3-034
  - Thursdays from 5:30 - 8:00 PM
- **Lab Exercises**:
  - S-3-020 when open and not used by a class
  - S-3-034 anytime 24/7 when not used by a class
- **Instructor Office Hours**:
  - S-1-060
  - Mondays, Wednesdays, Thursdays 1 - 2 PM

David Tenenbaum

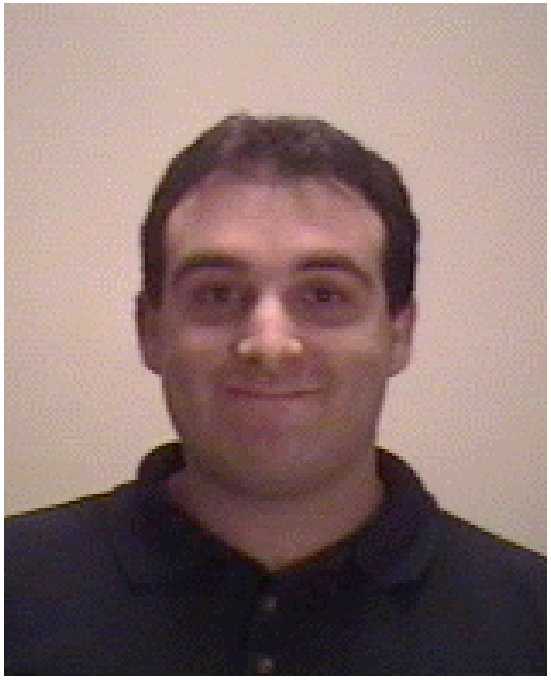
- Hon. B.Sc. at the University of Toronto
  - **Majors**: Physical and Environmental Geography & Environment in Society
- M.Sc. at the University of Toronto
  - **Thesis**: RHESSys-ArcView Integrated Modelling Environment
- Ph.D. at the University of North Carolina at Chapel Hill
  - **Dissertation**: Surface Moisture Patterns in Urbanizing Landscapes
- Canadian Government Lab Visiting Fellow at the Water & Climate Impacts Research Centre
  - **Research**: NAESI - In-Stream Flow Needs

How to Reach Me

David E. Tenenbaum, Ph.D.
Assistant Professor
EEOS, UMass Boston

Office: Science Building S-1-60

Email: david.tenenbaum@umb.edu
Tel: 617-472-7396



Object-oriented programming and customizing ArcGIS

Chapter 1 – **Programming with objects**

- pp. 1-4

Chapter 2 – **Building a custom application**

- pp. 5-35
- **Exercises 2A, 2B & 2C**

Chapter 1 – Programming with objects

- The bigger picture
- Object-oriented programming
- Visual Basic for Applications (VBA)
- ArcObjects

The Bigger Picture

- The (ideal) **course title** for EEOS 472 is **Programming for GIScience Applications**, **not Programming ArcGIS with Visual Basic** … why? A few reasons:
  - Technologies **come and go** (before ArcGIS and VBA there was ArcView 3.X and Avenue, and before that Arc/INFO and AML), **but GIScience** (and the need to work/research in GIScience) **remains**
- The **nuts and bolts** of ArcGIS and VBA are **important**, but the **key concepts** needed to understand them are **more important**
  - Your fluency in ArcGIS/VBA is really a means towards fluency in Programming for GIScience, potentially with other technologies (your **conceptual understanding** will outlive it)

Object-oriented programming

- **Object-oriented programming** (OOP) is a programming approach that uses "objects" to design applications and computer programs
  - This is contrast to **modular or procedural programming** that you might be familiar with, where **lines of code are numbered** and run in sequence
- Even though it **originated in the 1960s**, OOP was not commonly used in mainstream software application development until the 1990s
- Today, **many popular programming languages** (such as Java, JavaScript, C#, VB, .Net, C++, Python, PHP, Ruby and Objective-C) support OOP

Object-oriented programming

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*

Object-oriented programming

Important OOP terminology:

- Class - A **pattern or blueprint** for **creating an object**. It contains **all the properties and methods** that describe the object
- Instance - The **object you create** from a class is called an **instance** of a class
- Distinguishing an **object/instance vs. a class**, examples:
  - Cookie vs. a cookie-cutter
  - Car vs. the blueprint for manufacturing the car

Visual Basic

- The "Basic" part refers to the **BASIC** (Beginners All-Purpose Symbolic Instruction Code) language
  - Visual Basic has **evolved** from the original BASIC language and now contains several hundred statements, functions, and keywords, many of which relate directly to the Windows GUI
- The "Visual" part refers to the **method used to create the graphical user interface** (GUI)
  - Rather than writing numerous lines of code to describe the appearance and location of interface elements, you can simply **add pre-built objects into place on screen**

Visual Basic for Applications (VBA)

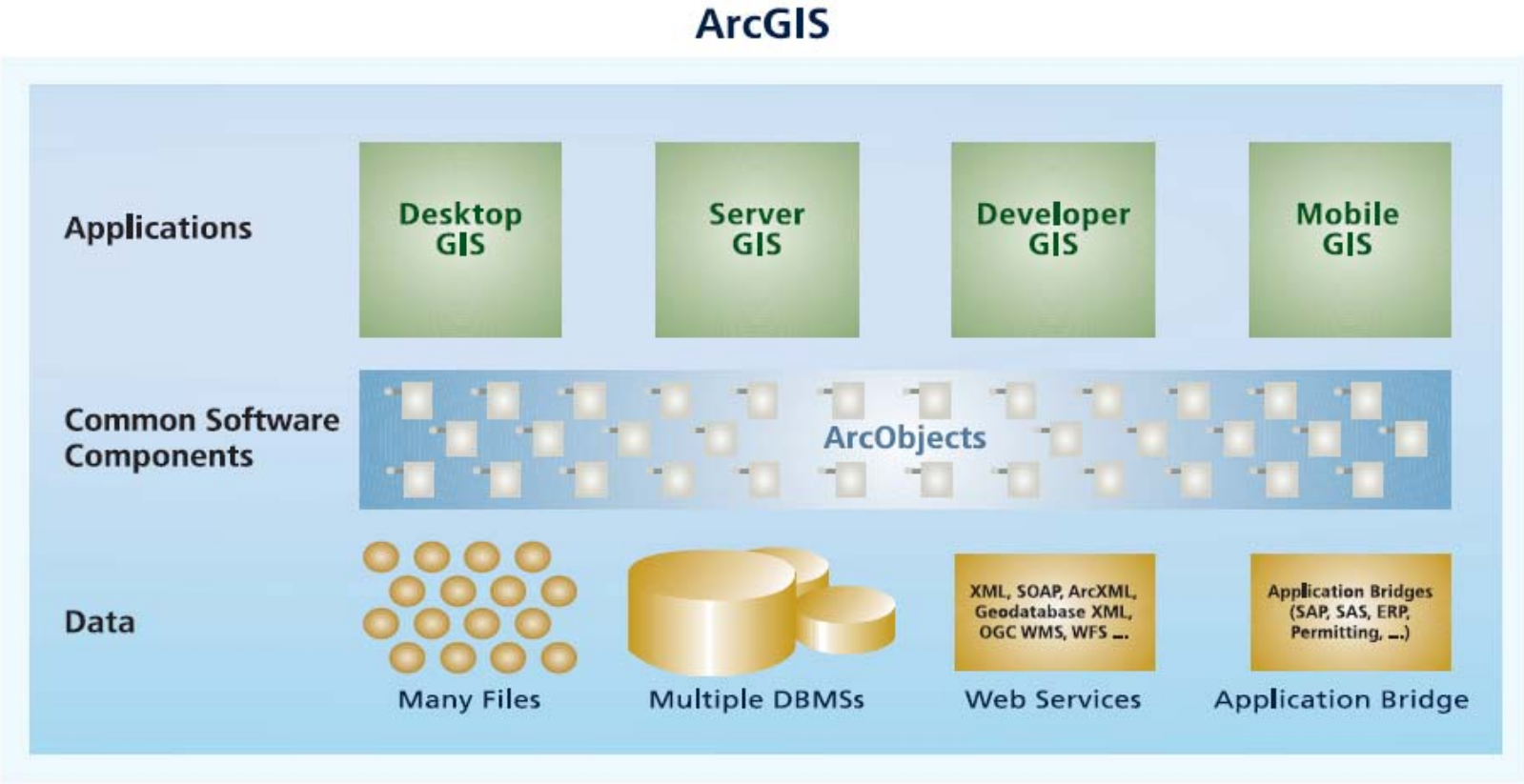
- VBA is the **application edition** of Microsoft's Visual Basic programming language
- VBA can normally only **run code from within a host application** rather than as a standalone application.
  - It can however be used to **control one application from another** using object linked embedding (OLE) automation (for example, embedding objects from the ArcObjects library into a PowerPoint application)
- You can find VBA **built into many** Windows applications (Office suite, AutoCAD, Corel Draw, others)

Visual Basic for Applications (VBA)

- In ArcGIS, we will **interact** with VBA using the **Customize dialog box** and the **Visual Basic Editor**
  - Through the **Customize dialog box**, we can **change the set of controls** that appear in the interface (like menu items, buttons, and controls)
  - Through the **Visual Basic Editor**, we can **work on the VBA code** associated with the controls, so that when someone clicks on a control in the user interface … something happens!
  - For each **event** that might occur in the user interface (e.g. when a user clicks on a control) there is a **particular set of instructions** that are executed. These are **organized into procedures**, which break the code into modular chunks, that can call each other etc.

ArcObjects

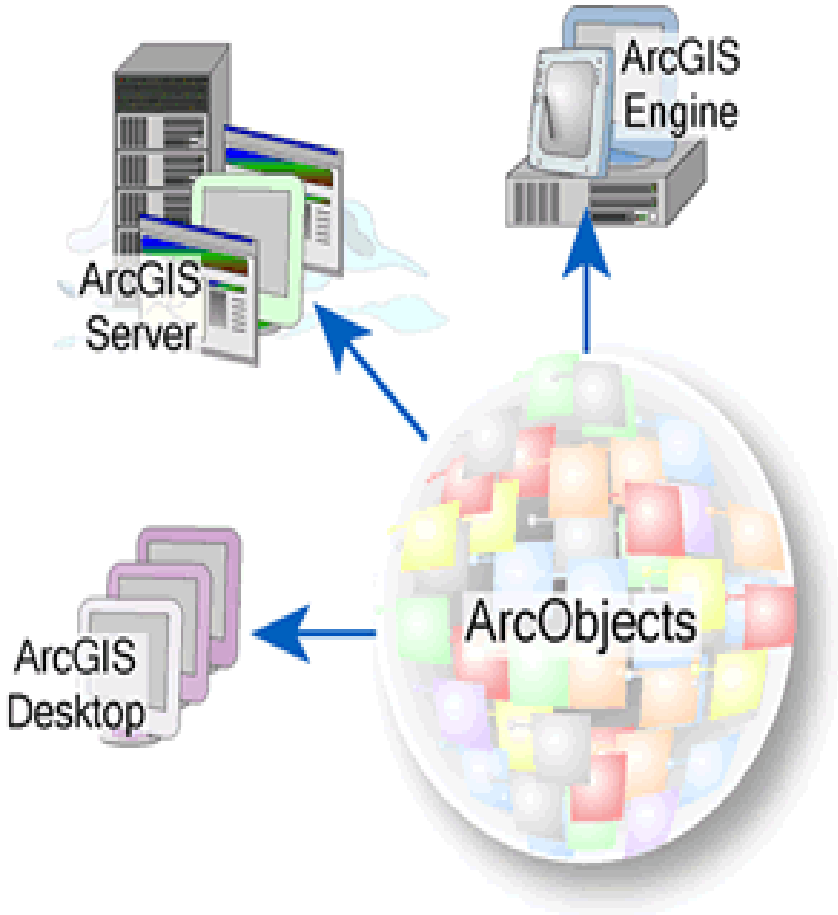
- So where do **ArcObjects** come into the picture?



ArcGIS is a complete GIS software system for authoring, serving, and using geographic knowledge.

ArcObjects

- ArcObjects are **platform independent software components**, written in C++, that provide services to support GIS applications, either on the desktop in the form of thick and thin clients or on a server for Web and traditional client/server deployments
- Because this architecture **supports a number of unique ArcGIS products** with specialized requirements, all ArcObjects are designed and built to support a multi-use scenario



ArcObjects

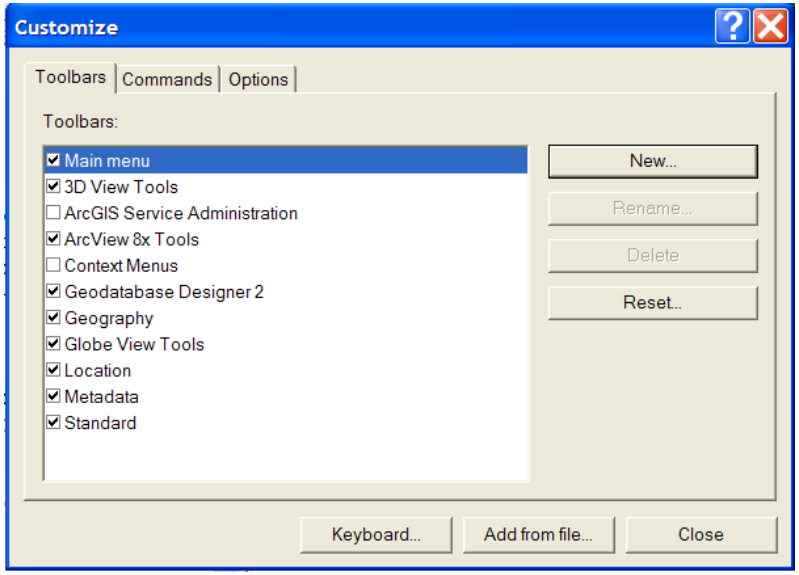
- You can think of ArcObjects as **the stuff that the insides of ArcGIS is made of** … all the structure and functionality that **allows ArcGIS to do what it does**
- ArcObjects are a **set of computer objects** specifically designed for **programming ArcGIS applications**, and they include things that have manifestations you can see onscreen, as well as more abstract data objects
- Anytime you have used ArcGIS, whether or not you knew it, **you have been using ArcObjects** … but what we will learn is how to **work with them in more powerful ways** than just through the user interface

Chapter 2 – Building a custom application

- Organizing commands on a toolbar
- Making your own commands
- Storing values with variables

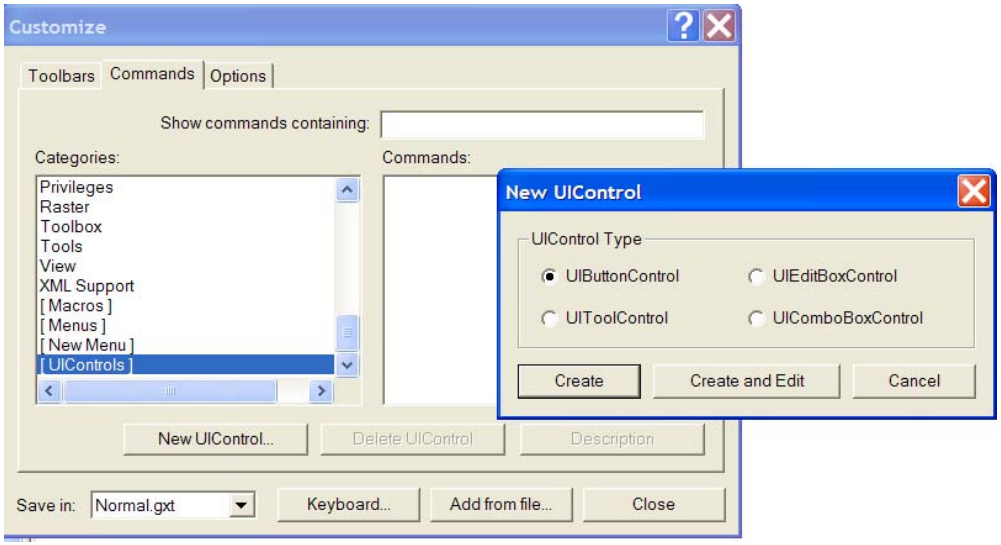
Organizing commands on a toolbar

- The **graphical user interface** (GUI) of ArcMap and ArcCatalog is made up of toolbars, menus, and commands
- Toolbars **contain** commands or menus
- Commands can come as **either buttons or tools**
  - We will learn the **distinction** between these later
- Menus provide **pull-down lists** of commands or of other menus
- To **change** the user interface, the Customize Dialog Box is used:



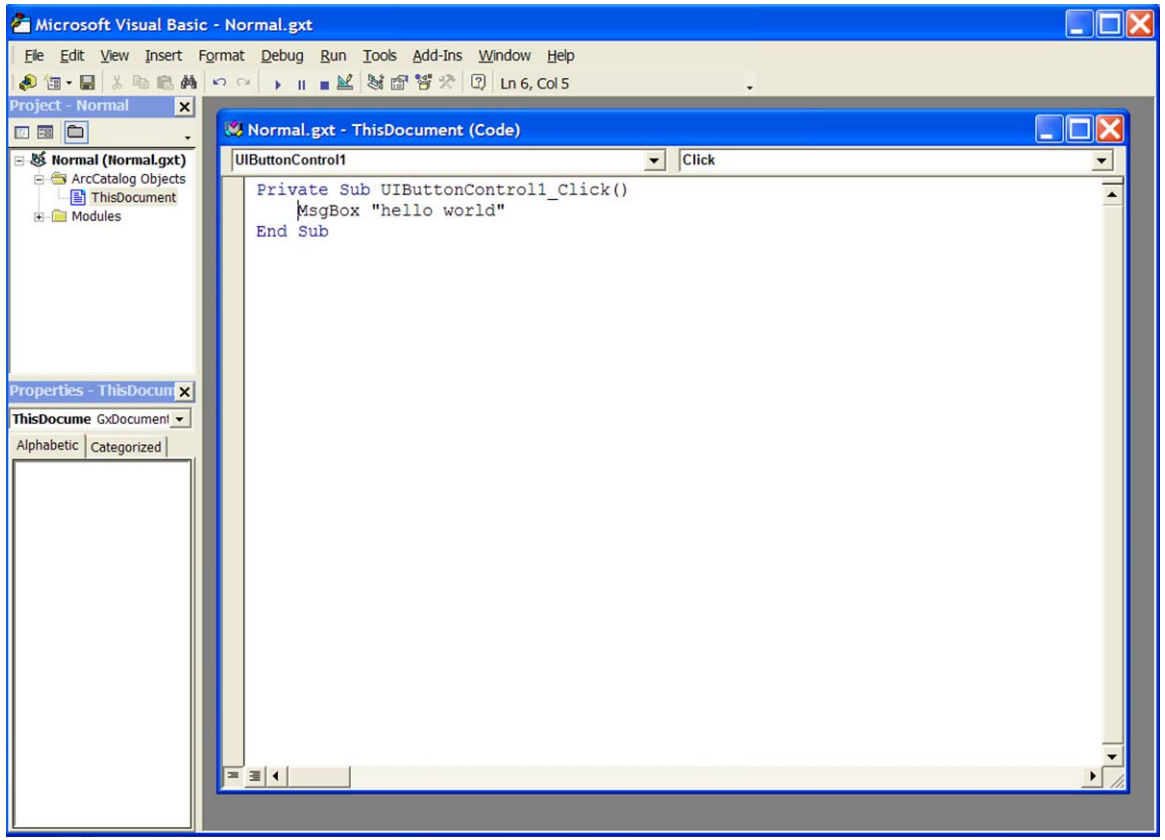
Making your own commands

- Beyond simply shuffling around controls using the Customize dialog box, we **can also make our own controls**, and then **build the code** that makes them do something:
- There are four **types of user interface controls** (in VBA, known as) UIControls:
  - UIButton
  - UITool
  - UIEditBox
  - UIComboBox



Making your own commands

- This is a simple as **setting up your new control**, then using the **Visual Basic Editor** to **write the code** associated with it (and its events):



Making your own commands

- **Where** do these customizations get saved?
- **Projects** are files where your UIControls and VBA code are stored
- There are **three types** of projects:
  - Map documents (.mxd files)
  - Base templates (.mxt files)
  - Normal template (normal.mxt)
- When we save code in a **map document**, it is **only available** when we open that map document
- The **normal template** can be used to store customizations that are **available with any project** that the user opens

Storing values with variables

- What is a **variable**? A **named storage location** that **contains data** that **can be modified** during a program
- Each variable **has a name that uniquely identifies it** within its scope. Usually, a **data type** is specified
  - Some data types: byte, boolean, integer, long, currency, decimal, single, double, date, string, object …

Type (Bytes)	***Sample Value***
String(10+Length)	*Elm Street*
Boolean(2)	*True or False*
Date(8)	*1/1/200 to 12/31/9999*
Byte(1)	*0-255*
Integer(2)	*-21768 to 32767*
Long(4)	*-2,147,483,648 to 2,147,483,648*
Single(4)	*1.401298E-45 to 3.402823E38 positive*
Double(8)	*1.79769313486232E308 maximum*

Storing values with variables

Some rules for naming variables:

- The name **must begin** with a **letter**
- The name **cannot contain periods**
- The name **cannot contain certain characters** (@,$,%,&,#)
- The name **can contain underscore** (_)
- The name can be **no more than 254 characters**
- **Reserved words** will **cause errors** (e.g. For, And, Loop)

Storing values with variables

A convention for naming variables:

- One way to **help keep track** of what **kind of variable** any given variable is makes use of **certain prefixes** for **particular data types**:

Type	***Prefix***	***Example***
String	*str*	*strAddress*
Boolean	*bln*	*blnStatus*
Date	*dat*	*datBirth*
Byte	*byt*	*bytAge*
Integer	*Int*	*intPopulation*
Double	*dbl*	*dblLatitude*

Storing values with variables

- Variables get ‘called into existence’ by **declaring** them:
- In many cases, you will **explicitly declare** them using the Dim statement:
  - *Dim txtGreet as String*
  - *Dim intCount as Integer*
- Another line can be used to **set their value**, once declared:
  - *txtGreet = “Hello”*
  - *intCount = 42*
- In some cases, you will **implicitly declare** them, and set their value simultaneously
  - *txtAlternateGreet = “Howdy”*

Next Topic:

Using dialog boxes and objects