COMPUTER SCIENCE 110 OUTLINE, 2012-2013 K. MACDONALD

This course serves as an introductory course in computer programming using Visual Basic Express 2010 that should prepare students to take further programming courses in high school, university, or community college and to make an informed career choice in this area. Students will learn to create simple Windows programs.

Outline

Topics

Visual Basic Interface
Controls
Toolbox
Properties
Flow Chart
Variables
Data Types
Conditional Statements
Multiple Forms
Writing Expressions
Debugging
Documentation
Animated Objects
Sound

EVALUATION
Tests
3 Minor Projects
Major Project during last 3 weeks

Final Mark
Average of Tests 20%
Minor Projects.....30%
Final Project 25%
Class Mark.....10%
Daily Assignments.....15%

Visual Basic

Visual Basic is used to create applications for Microsoft Windows. It includes tools that allow a programmer to create an application that has features similar to other Windows applications without having to write many lines of code.

Visual Basic is based on the BASIC programming language developed in the 1960s by John Kemeny and Thomas Kurtz at Dartmouth University. BASIC stands for Beginner's All-Purpose Symbolic Instruction Code and was used by Kemeny and Kurtz to teach programming to their students.

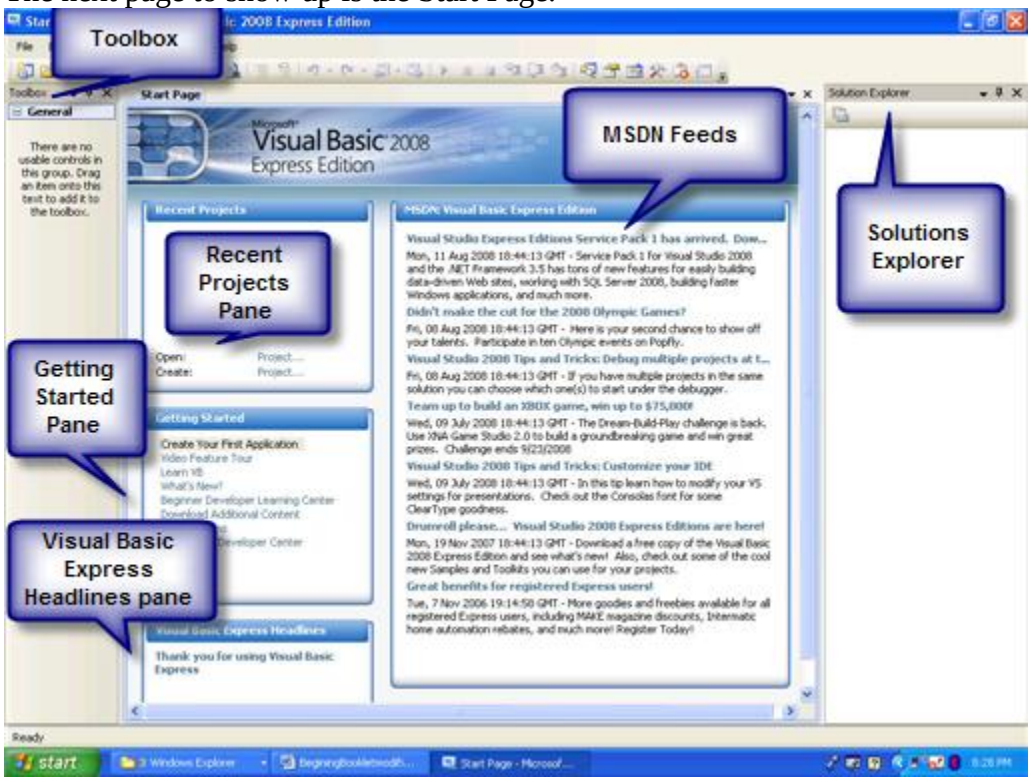
In 1975, Bill Gates and Paul Allen developed a version of BASIC especially for the Altair personal computer. With the success of this new version of BASIC, Gates and Allen founded Microsoft Corporation. BASIC then evolved to QuickBasic, a structured language that made programming easier for the rapidly growing number of personal computer users. In 1985, the Windows GUI (Graphical User Interface) was introduced. In 1992, Microsoft used QuickBasic and a program called Ruby to develop Visual Basic, an ***object-oriented programming*** environment for creating Windows programs.

Visual Basic programs are **event-driven**. An *event* is a way in which the user can interact with an object, such as using the mouse to click on a button in a dialog box. An ***event-driven program*** waits for an event to occur before executing any code, then only code for the current event is executed.

Getting Started

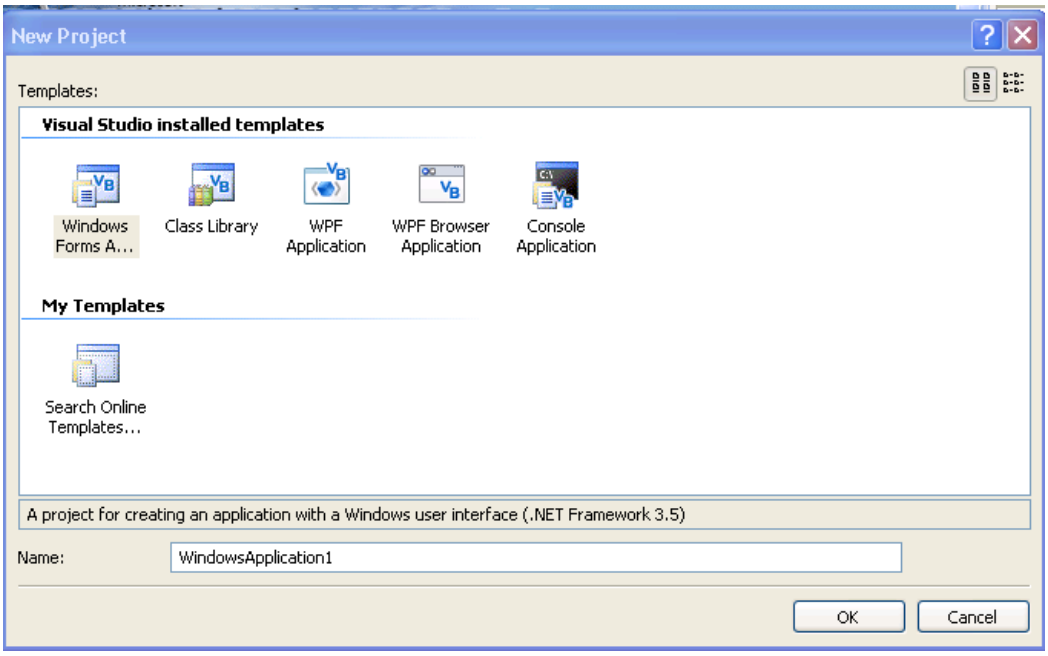
1. **Start Visual Basic** by clicking on the **Microsoft Visual Basic Express Edition on the Program menu or**  **Shortcut**.

2. The next page to show up is the Start Page.

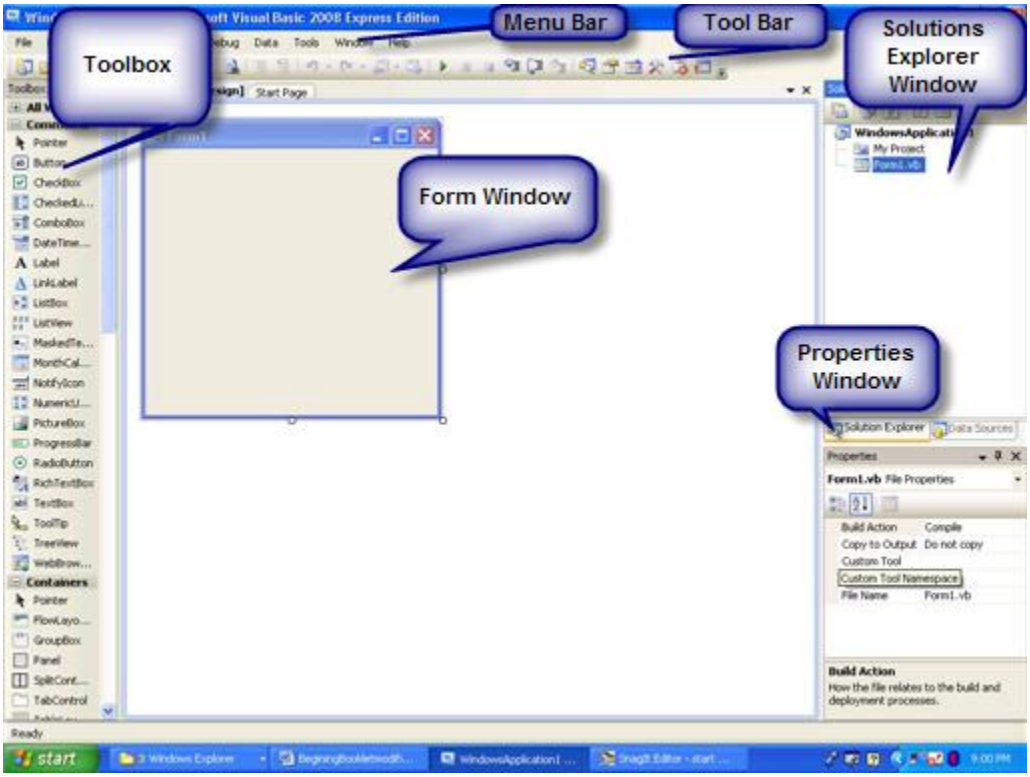


The start page contains the following information:
Toolbox: When expanded the toolbox contains all the controls for creating the applications.
Recent Projects Pane: Lists all the recent projects that have been opened.
Getting Started Pane: Allows the user to get help with projects by hyper linking to programmers or seeing a list of how to articles.
Visual Basic Express Headlines Pane: This is where you receive up to date information about VB from Microsoft if you have a live internet connection.
MSDN Feeds: By default this feed will come from the MSDN VB 2008 Express RSS feeds. This feed can be changed through the tools and options page.
Solutions Explorer: This features lists the files and components in your project.

3. Once you click on Create new project the following templates screen will appear. We will be using the Windows Form Applications for our projects.



4. Your screen should look similar to the picture below. Examine the various parts. If any of them are missing, they can be restored by clicking on View—Then selecting the appropriate part. If the Form Windows does not show, double click on the Form name in the Project Window. Although these Windows can be moved around the screen, it is advisable to leave them in the configuration shown below.

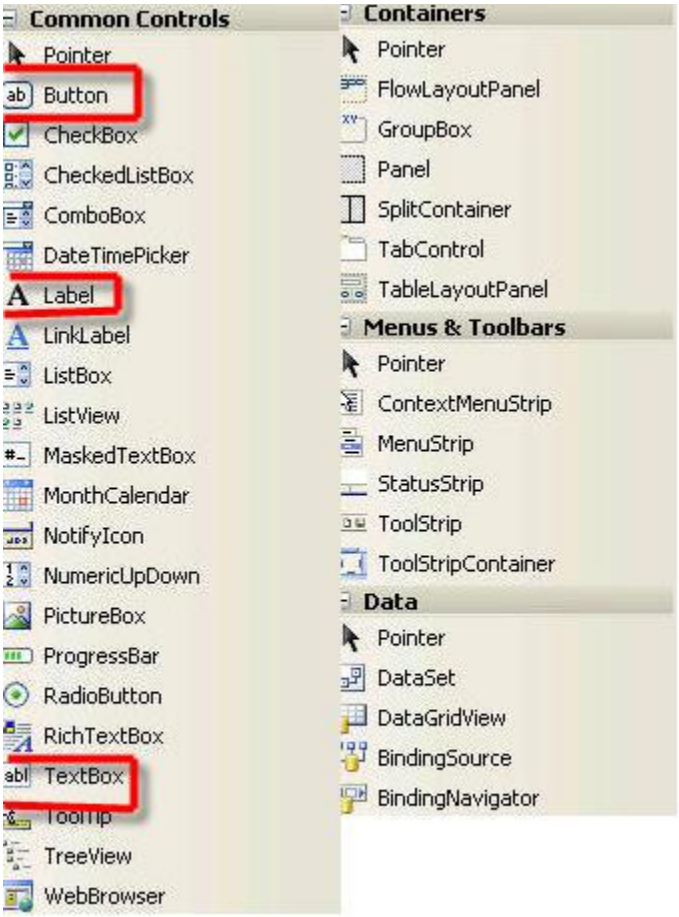


The Form Window

This is where you design the Windows for your program. You can add additional forms by clicking on the New Form icon (or selecting New Form from the File menu).

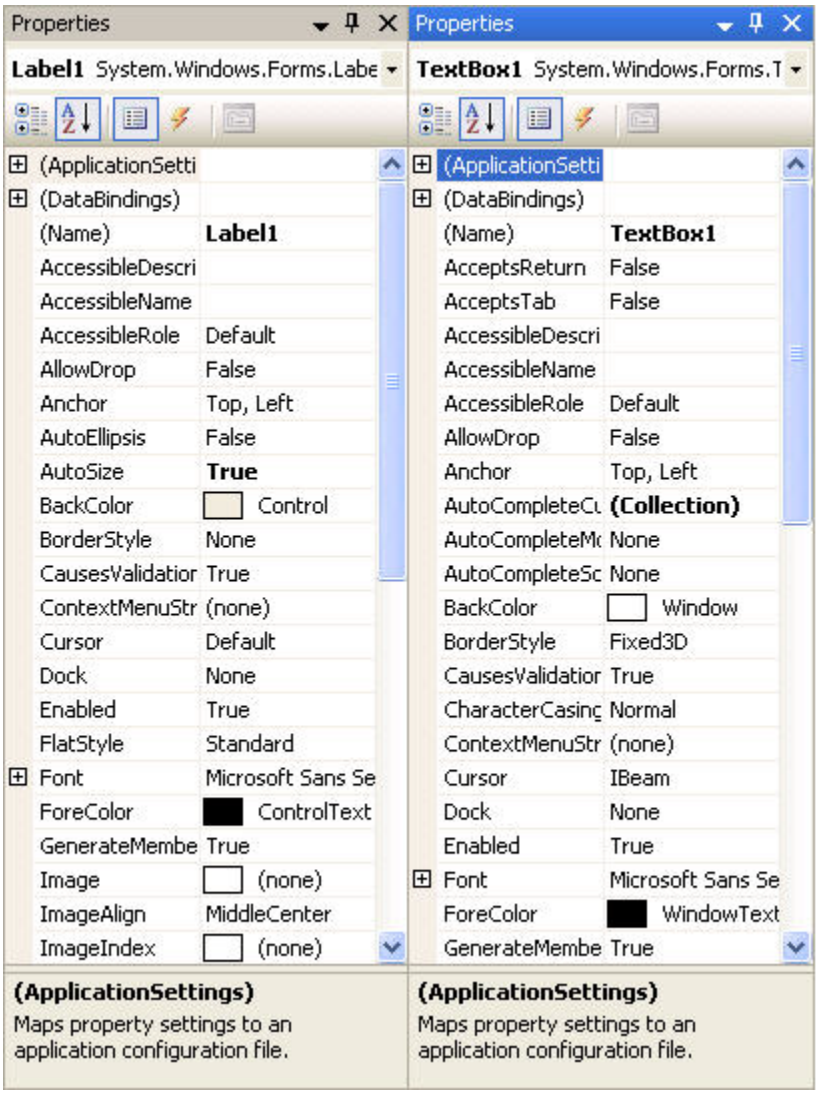
The Solutions Explorer Windows (The Project Window VB6)
Forms are saved as projects. If you need only one form in your program, then that project will contain only a single form. The project window lists the form names used in your project.

The Toolbox As you can see there are many different controls that can be used in a VB project. The most commonly used one are boxed in red.



You can select various tools and put them on the form. This can be done by clicking once on the object and then clicking and dragging across a section on the form, or by double clicking on the tool and then moving it to the point you wish. By clicking on the object on the form, you will get handles which indicate that that tool is the active one and you can then resize it, move it, or set properties on it.

The Properties Window Here is an example of two different objects with some of their properties. Some are the same while others are totally different.



One of the characteristics of an object is that it has a "state." This state is made up of a number of properties. Visual Basic lists the properties of objects in a Properties window. At any one time, the window shows the properties of only one object. First click on the object to select it, then go to the properties window and the properties for that object are displayed. If no object is selected, then the properties for the form are displayed.

Placing the Objects

***Textboxes*—**These are boxes that contain text. A textbox can display numbers, letters, or a mixture of both. You use textboxes to accept INPUT from people running your program.

***Labels*—**You use a label object to tell something to people running your program—in other words, to print something.

***Buttons* –** The button controls are buttons that users click as they are running your program. When a user clicks on one of these buttons, something (an action) happens. The caption of a button, such as "Exit" or "Quit," explains what that action will be.

Naming Conventions

The way you name the controls placed on a form is important because the names are used to refer to the controls in the program code. Meaningful names make understanding the program code easier.

Textbox names start with the prefix "txt". The prefix for labels is "lbl", for Command (Formerly cmd) Buttons, "btn", and for forms "frm". The prefix is followed by one or more words describing the function of the control. No spaces are used. Each new word is capitalized. Here are some examples:

txtLastName
lblBirthPlace
txtUnpaidBalance
btnDisplayPicture
frmDisplay

Here is the list of prefixes:

Prefix Abbreviations for Control Names

Prefix	Control
cbo	Combo Box
chk	Check Box
btn	Button
dir	Directory list box
drv	Drive list box
fil	File list box
fra	Frame
frm	Form
grd	Grid
hsb	Horizontal scrollbar
img	Image
lbl	Label
lin	Line

lst	List box
mnu	Menu
ole	OLE client
opt	Option button
pic	Picture Box
shp	Shape
tmr	Timer
txt	Text box
vsb	Vertical scrollbar

Setting Properties for Forms,Textboxes, Labels, and Commands

Labels
Text—This is the text that the label will display
Name—The name of the label—eg. lblBirthplace—The Name is used to identify the label in code.

Textboxes
Name—The name of the Textbox—eg. txtLastName—used in code
Text—The text that you wish to see displayed, if any. Often this is just deleted as the user will fill in the textbox when the program is run.

Commands / Buttons
Text—Words you wish to see displayed on the command button.
Name—the name of the button—eg btnExit—used in code

Forms
Name—the name of the form—eg. frmTitle—used in code. This name will appear in the Project Window. It will also be used as the filename when you save the form.
Text—This text will appear in the Title Bar of the form.

Steps in Creating a Program

This is a **three-step process**.
1. Select the object and arrange on the form.
2. Set the properties of the form and the objects.
3. Write the code.

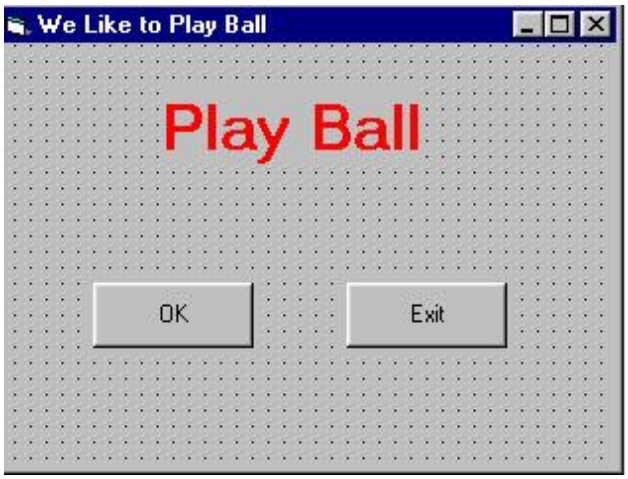
To Write the Code—Attaching Actions to Objects

Double click on the object to get to the Code Window. Notice that Sub and End Sub are already there for you. Insert your code in between. For example, the code for the Exit button would be **Me.Close**. This command should be indented to make your code easier to read.

Double clicking on the form name in the Project Window will return you to the form. Then just double click on the next object you wish to attach code to.

Our First Program

Let's write a program to display "**Play Ball**" in large red letters when we hit a command button called **OK**.



Step 1—Place a **label** on the form, as shown, large enough to hold the words **Play Ball**. Place 2 **buttons** on the form as shown.

Step 2--Set the properties for the form and the two objects as follows:

Form Set **text** to **We Like to Play Ball**
Set **Name** to **frmPlay**

Label Set **text** to **Play Ball**
Set **name** to **lblPlayBall**
Set the **font** to **MS Sans Serif Bold 24 pt**
Set **forecolor** to **red**
Set **visible** to **false**
Set **Autosize** to **true**

1st Button Set **name** to **btnOK**
Set **text** to **OK**

2nd Button Set **name** to **btnExit**
Set **text** to **Exit**

Step 3—Write the code. **Double click** on the **Exit** command. Tab to indent, and insert this code between the sub and end sub commands:

Me.close

Close the code window to return to the form, and **double click** on the **OK** command. Enter this line of code between the sub and end sub commands.

lblPlayBall.Visible = True

Run your program. Test the OK and Exit buttons.

Saving a Project—Select **File, Save Project As**

Select your own folder on the fileserver with your login name on it. For each Visual Program you wish to save, create a new folder within your folder in which to store all the files associated with your project. This is necessary because most programs consist of several files and it is important to keep them together.

You will first be asked to save the form—Yes—The default filename should be frmPlay.—Make sure to put it in a new folder within **your folder**.
Then you will save the project—Yes—Name it **PlayBall** and put it in **the same folder**.