[Introduction to QBASIC 4](#_Toc272150193)

[QBASIC 4](#_Toc272150194)

[**The Editor** 4](#_Toc272150195)

[Loading the editor 4](#_Toc272150196)

[Using the Editor 4](#_Toc272150197)

[**QBASIC Commands** 6](#_Toc272150198)

[**Variables** 6](#_Toc272150199)

[**Numeric Operations** 7](#_Toc272150200)

[Example Program 7](#_Toc272150201)

[**Looping** 7](#_Toc272150202)

[Example Program using a For Loop 8](#_Toc272150203)

[Example Program using a Do Loop 9](#_Toc272150204)

[**Decision Structures** 9](#_Toc272150205)

[**Format of the Program** 9](#_Toc272150206)

[Sample program 10](#_Toc272150207)

[Starting Visual Basic 11](#_Toc272150208)

[**The Visual Basic Programming Environment** 11](#_Toc272150209)

[**Writing a Program in Visual Basic** 13](#_Toc272150210)

[The Controls 13](#_Toc272150211)

[Frequently used Controls 13](#_Toc272150212)

[Arranging Controls in the Form Window 14](#_Toc272150213)

[Setting the Properties for the Controls 14](#_Toc272150214)

[**The Menu and Tool Bar** 16](#_Toc272150215)

[**Entering Code** 17](#_Toc272150216)

[Input and Output 18](#_Toc272150217)

[Input Options 18](#_Toc272150218)

[Output Options 18](#_Toc272150219)

[**Graphics** 23](#_Toc272150220)

[Loading Existing Graphics 23](#_Toc272150221)

[Loading Graphics Created in PaintBrush 23](#_Toc272150222)

[Creating a Background For Your Program 24](#_Toc272150223)

[**Animation** 24](#_Toc272150224)

[**Using Timers** 24](#_Toc272150225)

[Using A Timer For Animation 25](#_Toc272150226)

[Simple Animation 25](#_Toc272150227)

[Advanced Animation 25](#_Toc272150228)

[**Adding Sounds** 26](#_Toc272150229)

[**Using Arrow keys to Move Objects** 28](#_Toc272150230)

[**Modular Programming** 32](#_Toc272150231)

[Passing Parameters 33](#_Toc272150232)

[Designing a Major Program 33](#_Toc272150233)

[**Writing The Code** 33](#_Toc272150234)

[**Common Questions** 34](#_Toc272150235)

[How do I know when two objects collide ? 34](#_Toc272150236)

[How do I get more than one object? 35](#_Toc272150237)

[How do I shoot? 35](#_Toc272150238)

[How do I program Pacman ? 36](#_Toc272150239)

[**Modular Programming** 39](#_Toc272150240)

[Passing Parameters 39](#_Toc272150241)

[Designing a Major Program 39](#_Toc272150242)

[**Writing The Code** 39](#_Toc272150243)

[**Common Questions** 40](#_Toc272150244)

[How do I know when two objects collide ? 40](#_Toc272150245)

[How do I get more than one object? 41](#_Toc272150246)

[How do I shoot? 41](#_Toc272150247)

[How do I program Pacman ? 42](#_Toc272150248)

[Basic Graphics and Animation 19](#_Toc272150249)

[**Basic Graphics** 19](#_Toc272150250)

[Screen Co-ordinates 19](#_Toc272150251)

[Colour Chart **Error! Bookmark not defined.**](#_Toc272150252)

[Colour **Error! Bookmark not defined.**](#_Toc272150253)

[Painting Colours **Error! Bookmark not defined.**](#_Toc272150254)

[**Creating Graphics** 21](#_Toc272150255)

[Drawing 21](#_Toc272150256)

[Example Program 22](#_Toc272150257)

[**Drawing Graphs** **Error! Bookmark not defined.**](#_Toc272150258)

[Redefining Coordinate Systems 45](#_Toc272150259)

[Creating Graphs 45](#_Toc272150260)

[**Common Questions in Visual Basic** 45](#_Toc272150261)

[How do I know when two objects collide ? 45](#_Toc272150262)

[How do I program a board game ? 45](#_Toc272150263)

[Appendix 46](#_Toc272150264)

[Declaring Variables 46](#_Toc272150265)

[QBASIC Assignments 48](#_Toc272150266)

[**Assignment #1** 48](#_Toc272150267)

[**Assignment #2** 49](#_Toc272150268)

[**Assignment #3** 50](#_Toc272150269)

[**Assignment #4** 50](#_Toc272150270)

[**Assignment #5** 51](#_Toc272150271)

[**Assignment #6** 52](#_Toc272150272)

[**Assignment #7** 52](#_Toc272150273)

[**Animation Project** 52](#_Toc272150274)

[**Assignment # 8** 52](#_Toc272150275)

[**Assignment #9** 53](#_Toc272150276)

[**Midterm Practice Problems** 53](#_Toc272150277)

[**QBASIC Major Project** 53](#_Toc272150278)

[Visual Basic Assignments 54](#_Toc272150279)

[**Visual Basic Assignment #1** 54](#_Toc272150280)

[**Example Program** 54](#_Toc272150281)

[Creating the Program 54](#_Toc272150282)

[Changing the Properties 55](#_Toc272150283)

[Entering the Code 56](#_Toc272150284)

[Running The Program 56](#_Toc272150285)

[**Visual Basic Assignment #2** 56](#_Toc272150286)

[**Visual Basic Assignment #3** 57](#_Toc272150287)

[**Visual Basic Assignment #4** 57](#_Toc272150288)

[**Visual Basic Assignment #5** 57](#_Toc272150289)

[**Visual Basic Assignment #6** 57](#_Toc272150290)

[**Visual Basic Major Project** 58](#_Toc272150291)

# Introduction to QBASIC

# QBASIC

## **The Editor**

The QBASIC editor is a program that allows you to enter QBASIC commands and then translate them to machine language so that your program may be executed.

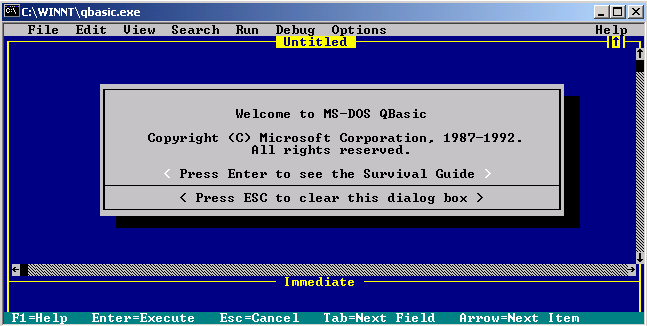


### Loading the editor

When the WINDOWS menu appears find the QBASIC icon by going to START

and then ALL PROGRAMS and then click on MS QUICKBASIC.

The QBASIC editor will then appear on the screen.



Press ESCape to clear the dialog box.

You can now enter your program by typing in the commands.

### Using the Editor

1. Inserting characters

Move the cursor to the location where you want to insert the missing information. Type in the missing information, as you type the text will move to the right to make room for the new information. If you would rather type over the old text, then press INSert to change to the over type mode. To change back to insert mode press INSert again.

2.Deleting characters

1. Use the BACKSPACE key to delete characters to the left of the cursor.
2. Use the DELete key to delete characters at the cursor position.
3. Use CTRL\_Y to delete the entire line.

3.Cursor control keys

1. The ARROW KEYS move the cursor left to right one character at a time or up and down one line at a time.
2. CTRL and the arrow keys move the cursor left to right one word at a time.
3. HOME moves the cursor to the beginning of the current line.
4. CTRL-HOME moves the cursor to the beginning of the program.
5. END moves the cursor to the end of the current line.
6. CTRL-END moves the cursor to the end of the program.
7. ENTER starts a new line
8. PAGE UP moves up one screen worth of commands.
9. PAGE DOWN moves down one screens worth of commands

4. Starting a new program

|  |  |
| --- | --- |
| Using the mouse | Using the keyboard |
| Click FILE  Click NEW | Press ALT-F  NEW is highlighted, press ENTER |

Note: If you were working on another program and you did not save it the computer will remind you to save it. Use TAB to move the various choices and press ENTER to execute one of the options.

5.Running a program

|  |  |
| --- | --- |
| Using the mouse | Using the keyboard |
| Click RUN  Click START | Press ALT-R OR press SHIFT-F5  START is highlighted, press ENTER |

6.Saving a new program

|  |  |
| --- | --- |
| Using the mouse | Using the keyboard |
| Click FILE  Click SAVE AS … | Press ALT-F  move cursor down to SAVE AS…. |
| In the drive window, select U:  Type in the filename of the program filename.bas. | |

You make up the filename; it can be up to 8 characters in length. The filename must end in .bas if you want to find the file again.

The program name will now change from UNTITLED to the name you gave the program.

7. Saving an old program

|  |  |
| --- | --- |
| Using the mouse | Using the keyboard |
| Click FILE  Click SAVE | Press ALT-F  move cursor down to SAVE and press ENTER |

The program will be saved under its current name.

8. Loading an old program

|  |  |
| --- | --- |
| Using the mouse | Using the keyboard |
| Click FILE  Click OPEN | Press ALT-F  move cursor down to OPEN and press ENTER |
| Move cursor to U DRIVE and press ENTER  The filename will read U:\*.BAS. Click on ok and list of files will appear . Move the cursor to the correct file and press ENTER .You can now run or edit a program. | |

9. .Exiting the editor

|  |  |
| --- | --- |
| Using the mouse | Using the keyboard |
| Click FILE  Click EXIT | Press ALT-F  move cursor down to EXIT and press ENTER |

## **QBASIC Commands**

To receive information from the keyboard (input) or to display information on the screen (output) the commands INPUT and PRINT are used.

The format for these commands is as follows:

INPUT “message”, variable name Ex. INPUT “Type in your age”,age%

INPUT “Type in your name”, name$

INPUT answer$

PRINT “any message” Ex. PRINT “HELLO”

PRINT “What is your name?”

## **Variables**

Variables are used to store information that can be changed.

A variable name must start with a letter and end with one of the type-declaration

characters: %, &, !, # or $

|  |  |  |
| --- | --- | --- |
| CHARACTER | VARIABLE TYPE | EXAMPLE |
| % | integer | age% =25 |
| & | long integer | population& = 3562729 |
| ! | single precision real | area! = 15.239 |
| # | double precision real | chemistry# =2.01542984663 |
| $ | string | name$=“Chris” |

Integer- integers from –32,768 to + 32,767

Long integers- integers from -2,147,483,648 to + 2,147,483,647

Single precision real- decimal numbers with accuracy of 7 decimal points

Ex. 2/3=0.6666667

Double precision real- decimal numbers with an accuracy of 16 digits

Example.1/3= .3333333333333333

## **Numeric Operations**

Numbers can be added, subtracted, multiplied, divided, and raised to exponents.

|  |  |  |  |
| --- | --- | --- | --- |
| Symbol | Function | Example | Result |
| + | addition | sum!=1.5 + 4.9 | sum!=6.4 |
| - | subtraction | difference%=7-3 | difference%=4 |
| \* | multiplication | product!=1.5\*2.2 | Product!=3.3 |
| / | division | quotient!=4.5/2.5 | quotient!=1.8 |
| ^ | exponentiation | power%=3^2 | power%=9 |
| \ | integer division | result%=6/4 | result%=1 |
| MOD | remainder | remain%=6MOD4 | remain%=2 |

Other built in functions are:

SIN(n) Sine of n ABS(n) Absolute value of n

COS(n) Cosine of n SGN(n) Sign of n

TAN(n) Tangent of n SQR(n) Square root of n

ATN(n) ArcTangent of n RND Random number

Note: For the trigonometric functions n must be an angle in radians

### Example Program

Ex. Write a program that asks the user for two numbers. Have the program calculate the sum and the product.

Input “ Type in two numbers”, num1% , num2%

Sum% = num1% + num2%

Product% = num1% \* num2%

Print “The sum is ”, Sum%

Print “The product is ”, Product%

End

## **Looping**

One of the greatest powers of the computer is that it can quickly repeat a series of steps. In order to have the computer repeat these steps we use a loop.

In QBASIC there are three main forms of loops:

1. FOR NEXT loop- for a certain number of times the steps are repeated.

2. WHILE WEND loop-while something is true the steps will be repeated.

3. DO LOOP-Repeat the steps as long as a condition is true or until a condition becomes

true.

The format for these loops is as follows :

count% = 1

FOR variable = Start value to end value ex. FOR i% = 1 to 100

Steps to repeat PRINT count%

NEXT variable count% = count% + 1

NEXT i%

WHILE condition ex.While count% <100

Steps to repeat PRINT count%

WEND count% = count% + 1

WEND

DO UNTIL condition DO WHILE condition

Steps to repeat Steps to repeat

LOOP LOOP

DO DO

Steps to repeat steps to repeat

LOOP UNTIL condition LOOP WHILE condition

Ex. DO ‘Print hello 40 times

Print “hello”

Count% = count% + 1

LOOP UNTIL count% > 40

The conditions of the loop statements can use the following operators:

= equals < less than

<> not equals >= greater then or equals

> greater than <= less than or equal

The conditions can be combined with logical connectives AND, OR, NOT.

Ex. WHILE (age% < 10) OR (age% > 70)

UNTIL (count = 50) AND (temperatures% = 35)

WHILE (age% NOT > 10)

### Example Program using a For Loop

Ex. Write a program to find the sum of the first 15 squares.

Sum% = 0

Number% = 1

For i% = 1 to 15

Square% = Number% ^ 2

Sum% = Sum% + Square%

Number% = Number% + 1

Next i%

Print “The sum of the first 15 squares is “, Sum%

End

### Example Program using a Do Loop

Ex. Write a program to determine the square of a number.

Repeat the program until the user decides to quit.

do

Input “ Please type in a number”, number%

Square% = Number% ^ 2

Print “The square of “; number%;” is “; square%

Input “ Would you like to continue ? (y/n) “, response$

Loop until (response$ = “n” )

End

## **Decision Structures**

Another important feature for a programming language to posses is a means for choosing between alternate actions. We would like to be able to perform certain operations in some cases and other operations in other cases.

In QBASIC the main method of deciding between various actions is the IF THEN ELSE statement.

The format is:

IF condition(s) THEN Ex. IF (mark% > 85) THEN

steps to be completed PRINT “Honor roll with distinction”

ELSE ELSE

steps to be completed PRINT “Do more homework”

ENDIF ENDIF

The IF THEN ELSE statement can be extended to deal with more than two possible cases

Ex. IF (number% = 0) THEN

PRINT “ zero ”

ELSEIF(number% = 1)THEN

PRINT “ one ”

ELSEIF(number% = 2) THEN

PRINT ” two ”

ENDIF

## **Format of the Program**

It is good programming practice to type your program in such a way that is easy to follow. The computer could not care less but as a programmer you will find it easier to follow a program that is typed in an organized fashion.

The easiest way of making your program more readable is by an effective use of BLANK LINES. Separate different parts of the program by a single blank line. This will visually group sections together.

Indentation is important when using loops and IF THEN ELSE statements.

Comments are important as they tell you what you are doing in each step of the program.

Comments can be written into the program in two ways:

1. Remark statements. Remark statements start with the command REM and then are followed by any information that you think is useful.

Ex. REM This program will determine the sum of two numbers

1. Using the apostrophe. An apostrophe signals that the rest of the line is a comment.

Ex. DegC! = ((DegF! – 32) \*5)/9 ‘Convert Fahrenheit to Centigrade

### Sample program

REM This program will print letter grades corresponding to students numeric grades.

REM \*\*\*\*\*\*\*read in information\*\*\*\*\*\*\*

INPUT “Please type in your number grade and press ENTER”, grade%

REM\*\*\*\*\*\*\*determine letter grade\*\*\*\*\*\*\*\*\*\*

IF (grade% >= 85) THEN

Letter$= ”A”

ELSEIF (grade% < 85 ) AND (grade% >= 65) THEN

Letter$= “B”

ELSEIF (grade% < 75 ) AND (grade% >65) THEN

Letter$= “C”

ELSEIF (grade% < 65) AND (grade% >= 50) THEN

Letter$= “D”

ELSEIF (grade% < 50)

Letter$= “F”

ENDIF

REM\*\*\*\*\*\*\*\*\*\*\*print grade\*\*\*\*\*\*\*\*\*\*\*\*

PRINT “the letter grade for your mark of”, grade% , “is”, letter$

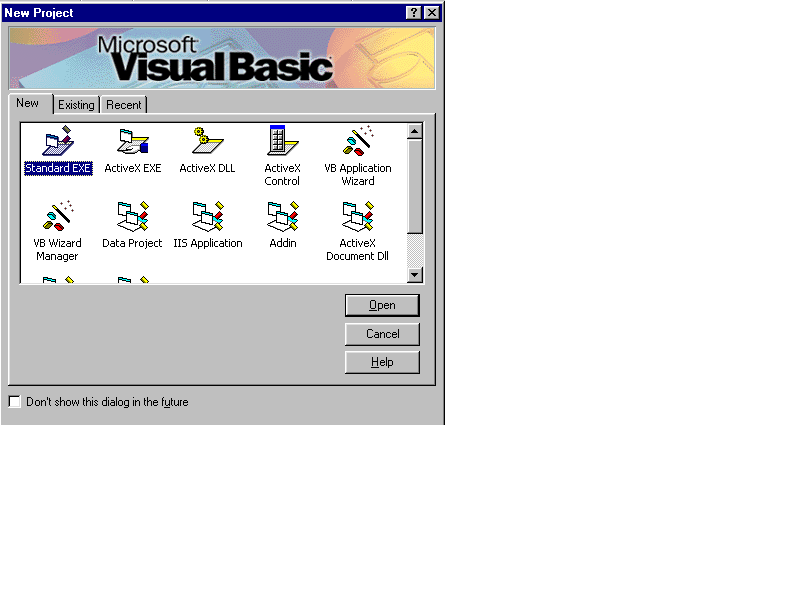
END

# 

# Starting Visual Basic

1. Click on the icon for Visual Basic.

The new project dialog box will appear, as show below. You can start a NEW project or load up already EXISTING projects.

****

2. Click open to start a new project.

A new project will open in the VISUAL BASIC programming environment. This environment includes windows and tools as shown on the next page.

## **The Visual Basic Programming Environment**

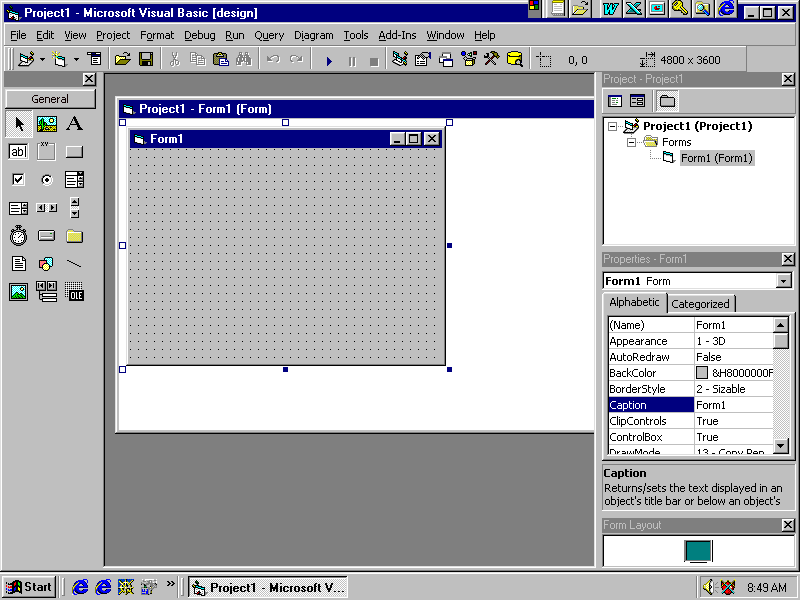
The programming environment contains the following tools to help you construct your programs:

1. menu bar – access commands to EDIT the program, load FILEs , etc
2. toolbar – shortcut buttons to execute commands
3. toolbox - toolbox controls are used to create objects, such as command buttons and labels that will be used by your program.
4. project container window – contains all the parts of the program
5. form layout window – allows you to determine the position of your form when you begin the program
6. project explorer window – lists and allows access to all the files used in the program such as your forms
7. properties window- lists and allows you to change the characteristics of your form or an object on your form such as font size, colour, etc on a form
8. form window - appears as a dotted grid. The grid will help align the different objects that will be used by your program.

Toolbox

Toolbar Project Container Window

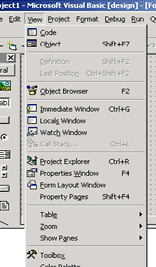
Menu bar Form Window Project Window



Properties Window

Form Layout Window

The form window can be changed in size by using the mouse. This will make it easier to layout the program.



**Restoring Windows**

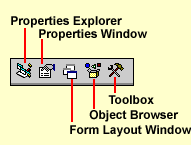
You can restore closed windows in one of two ways:

1. Using View on the Menu bar

Each of the windows listed on the previous screen can be restored by using the View menu. On the Menu bar, click View and select the window you wish to restore from the

drop-down list.

1. Using the Toolbar



On the right side of the Standard toolbar you will find icons for

the windows. If you wish to restore one of these closed windows,

click on the window's icon on the toolbar .

## **Writing a Program in Visual Basic**

There are three steps in writing a program in Visual Basic :



1/ Arrange the controls on the form window

2/ Set the properties for the controls

3/ Write the code that will make the program work

## 

## The Controls

Controls can be added using the tools in the tool box. The toolbox contains tools that can be

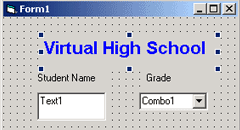
used to add control buttons, menus, labels and geometric shapes and pictures to the program.

To add a control to the form window, click on the icon in the tool box and then use the

mouse to place the object on the form.

### Frequently used Controls

|  |  |  |
| --- | --- | --- |
| Control | Function | Image |
| Command Button | used to request that a specific action be taken immediately in the program Ex. START, END, CALCULATE, SUM | Capture.PNG |
| Label | used to display text on the form. The most common uses of labels are :  1. display help text  2. display screen headings  3. display formatted output such as names, dates and times  4. provide a descriptive label for another object such as a  text box. | Capture.PNG |
| Text Box | used to display text or to get input from a user | Capture.PNG |
| Check Box | more than one option can be selected from a list of options  Ex. Pick the names of countries you would like to visit | Capture.PNG |
| Option Button | only one option may be selected from a list of options  Ex. Pick add , multiply , subtract or divide | Capture.PNG |
| Image Boxes | used to display imported pictures or graphics | Capture.PNG |
| Picture Boxes | used to draw basic images or graphs | Capture.PNG |



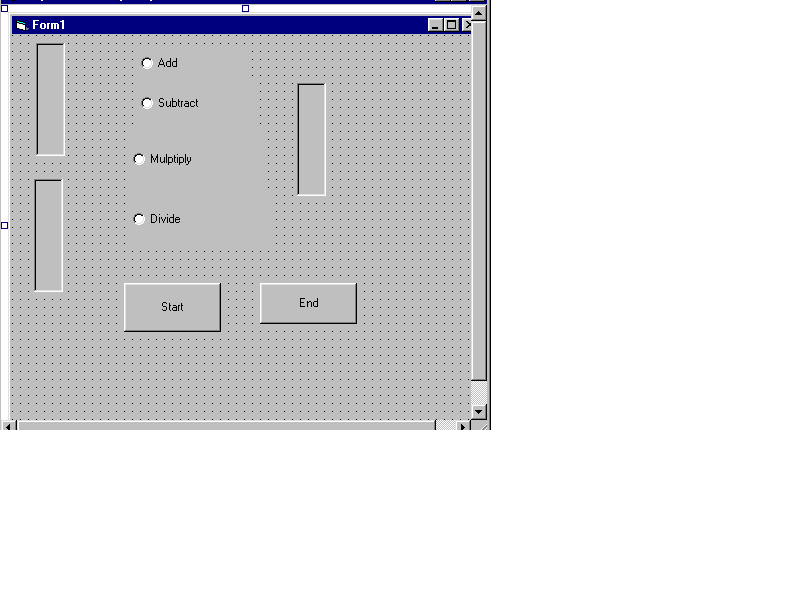
Ex.

Labels

Text box Combo box

### Arranging Controls in the Form Window

Arrange the controls so that the input and output is organized in a way that is easy to read and understand.



This form contains 9 controls :

3 labels to display information

**4 option buttons to choose a mathematical operation**

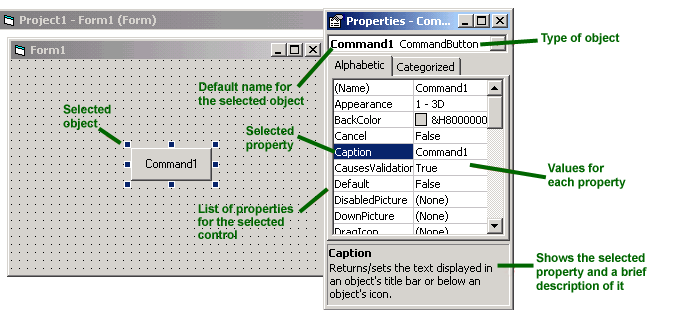
**2 command buttons to start or end the program**

## Setting the Properties for the Controls

The properties window is used to set the properties of each control object. Each type of control has its own list of properties. Properties that can be altered include the text font type, colour and size, captions on buttons, 3-D appearance of buttons and labels, pictures or images that remain visible or an invisible until something happens, etc..

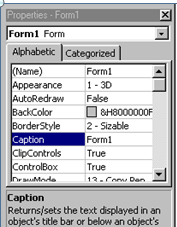
The Properties window lists all the properties that are available for the selected form or object.   
  
The Properties window contains the following:

* A drop-down list box at the top of the window to identify the selected object or from which you may select the desired object
* Two tabs listing the properties either alphabetically or by category
* A description pane at the bottom of the window to show the name of the selected property and a brief description of it



The image shown above highlights some of the available properties for a command button called Command1.

To change the property of a control, click the control on the formwindow. You can now change the properties for that control.



Ex. Change the back colour by clicking on backcolour and then use the mouse

to change the colour.

Each object must have a name. The programming code refers to each object

by its name. Visual Basic assigns a default name to each object when it is

first created. Objects are given default names such as Text1, Command1 or Label2.

As the programmer, you should change the default names to something more

unique. This way, you can avoid mistaking one object for another. The default

name can be changed through the Properties window by selecting the **Name**

property for the selected object. Object names that are assigned with the

Name property cannot exceed 40 characters in length.

The name that you choose for an object should:

* Clearly identify the purpose of the object.
* Identify the Toolbox control that created it.

For example, if you create a text box to hold an address, it will be given the default name Text1 when it is created. A better name for this object would be: txtAddress

The "txt" part of the name refers to the Toolbox control that created it. "Address" identifies the information in the text box. This name can be assigned by you, as the programmer, through the Properties window.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| A good Visual Basic programmer always follows the proper style conventions of the language when naming objects. Proper style convention dictates that each name should begin with a three-letter prefix that identifies the type of control that was used to create the object.   For instance, in the example **txtAddress**:   * The three-letter prefix txt indicates Address was created with a text box control. * Address is the name given to the object by the programmer, clearly indicating the object will hold an address value.   The chart illustrates the Toolbox controls and the three-letter prefixes used in naming these controls when proper style conventions are followed.  In addition to the objects on this table, form objects and menu objects will also be used frequently in Visual Basic. **Form** objects should use the prefix **frm** and **menu** objects should use the prefix **mnu**. | |  |  |  | | --- | --- | --- | | **Object** | **Prefix** | **Example** | | Command button | cmd | cmdStart | | Form | frm | frmPayroll | | Label | lbl | lblName | | Picture box | pic | picRockStar | | Text box | txt | txtAddress | | Combo box | cbo | cboEnglish | | Check box | chk | chkReadOnly | | Data | dat | datBibliography | | Directory list box | dir | dirSource | | Drive list box | drv | drvTarget | | File list box | fil | filSource | | Frame | fra | fraLanguage | | Horizontal scroll bar | hsb | hsbVolume | | Image | img | imgClouds | | Line | lin | linUnderline | | List box | lst | lstPeripherals | | OLE | ole | oleObject1 | | Option button | opt | optFrench | | Shape | shp | shpStar | | Timer | tmr | tmrRunAnimation | | Vertical scroll bar | vsb | vsbTemperature | |

## **The Menu and Tool Bar**

The tool bar provides a shortcut for executing commands.



Displays the controls

Run the program toolbox

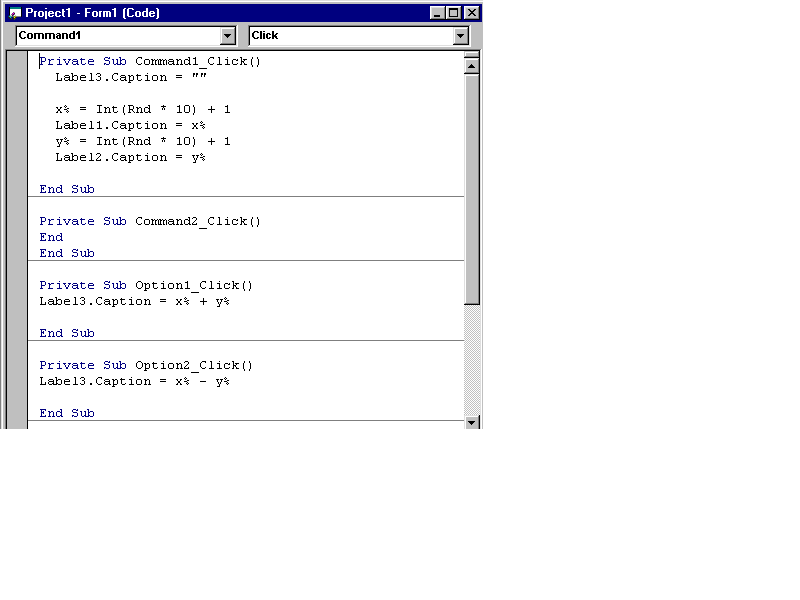
End the program

Form layout window

Displays the properties window

## **Entering Code**

Code is entered into the program using the code window.



To access the code window either double click on the control which activates this part of the program or click the VIEW CODE button in the project window.

All code is entered into separate sub programs. The code will be executed when that command button is pressed or an option button is chosen.

The programming for Visual Basic is essentially the same as it is for QBASIC. The same commands will work in both programs. The difference is in the displaying of information.

Ex. Write a program that will generate two random numbers and print the sum.

QBASIC Visual Basic

X1% = INT(rnd\*10)+1 X1 = INT(rnd\*10)+1

X2% = INT(rnd\*10)+1 X2 = INT(rnd\*10)+1

Sum% = x1% + x2% Sum% = x1 + x2

Print “ Number 1:”,x1% Label1.Caption = x1

Print “ Number 2:”,x2% Label2.Caption = x2

Print “ Sum:”,sum% Label3.Caption = sum%

Note that displaying information involves changing the PROPERTIES of the controls. The format for changing the property of a control is : Controlname.property = value

Ex. Label1.caption = x1%

Image1.width = 100

Picture1.circle(x%,y%),5,14

Text1.text = “Hello, welcome to my program”

### 

## Input and Output

There are a variety of options for getting information from the user or for displaying information on the screen.

### Input Options

|  |  |  |
| --- | --- | --- |
| Control |  | Effect |
| Command button |  | Click the button to execute an option |
| Option buttons |  | The user can choose one option from a variety of options |
| Check boxes |  | The user can choose multiple options from a variety of options |
| Scroll bars |  | The user can move the scroll bar to change the value of a number |
| Max | Maximum value of scroll bar |
| Min | Minimum value of scroll bar |
| Small change | The value of the scroll bar will change by this amount when you click on the bar |
| Hscroll1.value | The value of the scroll bar |
| Combination Boxes |  | The user can choose an option from a list of options |
| Input box | The user can input information in the same manner as QBASIC  Ex. Name$ = Inputbox (“Type in your name”)  You can specify the location of the top left corner of the input box  Ex. Country$ = Inputbox (“Type in the country”, , ,1000,3000) | |

### Output Options

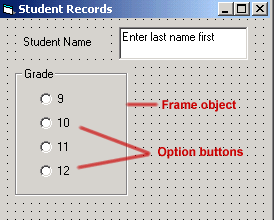
|  |  |  |  |
| --- | --- | --- | --- |
| Control |  | Effect | Example |
| Label |  | Display simple information | Label1.caption =hello”  Label2.caption = x% |
| Textbox |  | Display a long message | Text1.text = “once upon a time…” |
|  | Display more than one line |  |
| Image |  | Display an image | Image1.visible = true |
| Scale an image |  |
| Picture |  | Display a picture | Picture1.visible = true |
| Draw a picture | Picture1.circle(1000,2000),500,RGB(255,0,0) |
| Message box | Display a message on the screen | | MsgBox(“here is a message”) , , “title of message box” |

## Option Buttons

**Option buttons**, sometimes called **radio buttons**, are another way to obtain input from the user. Option buttons are grouped together. They are designed to provide the user with a set of input choices. Only one option button in the group can be selected at a time.

An option button object is created using the OptionButton control in the Toolbox. Properties are then set for this object just as we have done for other objects that we created.

Option buttons are usually coded with a **click** event. The click event procedure is executed when the user clicks on an option button.



A **frame** is a container object. The frame container is used to

group option buttons. An example of a frame containing

option buttons is shown above. In this example, the option

buttons for choosing a grade were placed in a frame with the

caption Grade.

Frames are created using the Frame control in the Toolbox.

A frame must be created first and then the option buttons are

added to it. When adding option buttons to a frame, they are

drawn with the OptionButton control and then dragged to the desired position within the frame. When option buttons are drawn in the frame in this way, both the frame and the option buttons will move together when the frame is dragged to a new location on the form.

When an option button is selected the options value is set to true. The following code could be used to determine which destination has been selected from a choice of 5 options :

If Option1.Value = True Then

destination$ = "Paris"

ElseIf Option2.Value = True Then

destination$ = "Rome"

ElseIf Option3.Value = True Then

destination$ = "London"

ElseIf Option4.Value = True Then

destination$ = "Madrid"

ElseIf Option5.Value = True Then

destination$ = "Riverview "

End If

# Basic Graphics and Animation

## **Basic Graphics**

Pixel- dot on the screen Resolution – number of pixels

### 

### Screen Co-ordinates

Every pixel has an X and Y co-ordinate.

(0,0) (320,0)

(160,100)

(0,200) (320,200)

Each tiny dot on your monitor is actually composed of three smaller dots—one red, one green and one blue. Red, Green and Blue (**RGB**) values are used to define the colours you see on your monitor.

To display a colour, you must vary the brightness of each of the three dots. For example, the colour purple is displayed when the red and blue dots shine brightly and the green dot is dim. To display white, all the dots must be bright and to display black, all the dots must be turned off.

The programmer can set the colour by using RGB values or by using Visual Basic **colour constants**. Using the RGB values allows the programmer to display more colour choices.

Visual Basic has an RGB built-in function. This function can be used at run time, in an assignment statement, to change a colour property. The following is an example:

**Private Sub** Form\_Load()  
     Me.BackColour = RGB(255,0,0)  
**End Sub**

In this statement, red, green and blue are integer values. The values may range between zero (0) and 255 to represent the brightness of the red, green and blue dots, which make up the colour. The above statement changes the background colour of a form to red.

Visual Basic also has predefined constants specifying RGB number values of some of the more standard colours. You can use these values to set the colour of an object instead of using the RGB values. The following is a list of Visual Basic colour constants:

* vbBlack
* vbBlue
* vbCyan
* vbGreen
* vbMagenta
* vbRed
* vbWhite
* vbYellow

The assignment statement below changes the background colour of a form to blue when the form is loaded:

**Private Sub** Form\_Load()  
    Me.BackColor = vbBlue  
**End Sub**

There are methods that can be used to create and manipulate graphics on form and picture box containers when an application is running.   
  
In the next few lessons, we will look at the following methods:

* **Line** —Draws a line or rectangle
* **Circle** —Draws a circle, ellipse or arc
* **Pset** —Sets the colour of an individual point
* **PaintPicture** —Paints a graphic at a specified location
* **Cls** —Clears all graphics

A shape object is created using the Shape control  in the Toolbox. Double-clicking on the Shape object will place a rectangle in the centre of the interface. Select the rectangle and then double-click the Shape property to select a different shape.

The table shows the properties that may be set for the Shape object.

|  |  |
| --- | --- |
| **Property** | **Description** |
| Name | This identifies the object. Good programming style uses the prefix shp with the name. |
| BackColor | This changes the background colour of a shape. This property is ignored when the FillStyle property is set to solid. |
| BackStyle | Can be set to transparent or opaque |
| BorderColor | Changes the colour of the outline of a shape |
| BorderStyle | Can be set to solid, transparent, dash, dot, dash-dot, dash-dot-dot or inside-solid |
| BorderWidth | Changes the thickness of the outline of a shape |
| FillColor | Changes the inside colour of a shape |
| FillStyle | Can be set to solid, transparent, horizontal line, vertical line or upward diagonal, downward diagonal, cross or diagonal cross |
| Shape | Can be set to rectangle, square, oval, circle, rounded rectangle or rounded square |

* **Note: When drawing shapes, the BackColor of the form should be set first because changing the BackColor of the form erases any graphics drawn on the form.**

## **Creating Graphics**

### Drawing

|  |  |  |  |
| --- | --- | --- | --- |
| Shape | Format | Example | Effect |
| Point | picDraw.PSET(x co-ordinate) | picDraw.PSET(5,20) | draws a point at location 5,20 |
| picDraw.PRESET ( X co-ordinate) | picDraw.PRESET(5,20) | erase the point drawn at 5,20 |
| picDraw.PSET STEP (X co-ordinate, Y co-ordinate) | picDraw.PSET STEP (5,10) | draws a point 5 over and 10 down from the last point used |
| Line | picDraw.LINE (start point) – (end point) | picDraw.LINE (5,10) – (20, 100) | draws a line from 5,10 to 20, 100 |
| picDraw.LINE STEP (start point)- STEP (end point) | picDraw.LINE STEP (100,20- STEP (5,-10) | draws a line starting 100 over and 20 down from the previous point and ending over 5 and up 10. |
| Note : If you want one of the points to stay in the same location just erase the STEP command | | | |
| Boxes | picDraw.LINE ( start point) – (end point), color , B | picDraw.LINE (5,10)-(20,100),1,B | draws a box from 5, 10, to 20, 100 and draws it in the colour 1 |
|  | picDraw.LINE (start point)-(end point) , color , BF | picDraw.LINE(5,10)-(20,100),RGB(255,0,0),BF | draws a box from (5,10) to (20,100) and colours it in the colour red |
| Circles | picDraw.CIRCLE (x center, y center), radius,colour | picDraw.CIRCLE (100, 50), 20,RGB(0,255,0) | draws a circle of radius 20 with center 100, 50 in green |
|  | picDraw.CIRCLE STEP (x center, y center), radius,colour | picDraw.CIRCLE STEP ( 2, 5) , 10 ,RGB(0,0,255) | draws a circle of radius 10 and over 2 and down 5 from the previous point in blue. |
| Arcs  Ellipses  and  Wedges | picDraw.CIRCLE(X center, Y center) , radius, color, start, end, aspect | Pi=3.14  picDraw.CIRCLE(100,130),30,RGB(255,0,0),-0,-PI/2 | picDraw.CIRCLE (100,30),30 |
| Start- The beginning point of an arc as an angle in radians  End- The ending point of an arc as an angle in radians  Aspect- The ratio of the x-radius to the y-radius | picDraw.CIRCLE (200, 30),30,RGB(255,0,0), 0,PI | picDraw.CIRCLE(300,30),30,RGB(255,0,0),PI/2, 0 |
| picDraw.CIRCLE(100,130),30,RGB(255,0,0), , ,2 | picDraw.CIRCLE(100,130),30,RGB(255,0,0),,,0.2 |

### Example Program

Draw a house.

picDraw.LINE (150, 220)-(460, 360), 1, BF 'body of house

picDraw.LINE (290, 320)-(330, 360), 4, BF 'red door

picDraw.LINE (370, 250)-(410, 290), 15, BF 'window

picDraw.LINE (200, 250)-(240, 290), 15, BF 'window

picDraw.LINE (150, 220)-(220, 160), 2 'roof

picDraw.LINE (220, 160)-(380, 160), 2

picDraw.LINE (380, 160)-(460, 220), 2

picDraw.LINE (460, 220)-(150, 220), 2

picDraw.PAINT (300, 200), 6, 2 'paint roof brown

END

## **Graphics**

Visual Basic uses TWIPS instead of pixels. There are more TWIPS to a screen than a pixel which gives better resolution. The screen is approximately 10000x10000 twips.

To draw in Visual Basic place a picture control on the form window. You can then draw points, lines and circles.

|  |
| --- |
| Visual Basic |
| Picture1.Pset(x%,y%),RGB(255,0,0) |
| Picture1.Circle (x%,y%),100,RGB(0,0,255) |
| Picture1.Line(x%,y%)-step(100,100),RGB(0,255,0) |

Colour is specified using the RGB function. The numbers in the bracket determine how much Red, Green and Blue to mix to make the colour of your choice.

Ex. RGB(255,0,0) only uses red and since 255 is the largest number this is a deep red

RGB(255,0,255) would mix an equal amount of red and blue producing purple

### 

### Loading Existing Graphics

To load existing graphics simply change the property of the picture. After a picture control is placed on the form change the following properties :

OBJECT PROPERTY SETTING

Picture1 Stretch True

Picture \*

\*Local Disk C: , Program Files , Common Files , Microsoft Shared , Clipart , cagcat50… This will give you some basic pictures from the clipart gallery

### Loading Graphics Created in PaintBrush

Use paintbrush to draw your picture. To start paintbrush click on programs and then accessories. Save your picture under a logical name and then exit paintbrush.

To load the paintbrush picture simply change the property of the picture. After a picture control is placed on the form change the following properties :

OBJECT PROPERTY SETTING

Picture1 Stretch True

Picture \*

\* the name that you used to save your picture.

Save your paintbrush pictures on your drive and then load them from that drive.

### Creating a Background For Your Program

If you would like your program to run with a picture in the background you simply load a picture into the form. You can then place controls on top of this background.

## **Animation**

The **Move** method is used at run time to move a picture box or shape object to a new location. The syntax for the Move method is as follows: In the example:

picHappyFace.Move picHappyFace.Left-20

the picture box object called HappyFace is moved 20 twips to the left of its current position. That means if the picture box is now at 100 twips from the left edge of its container, it will be moved -20 twips and will be repositioned 80 twips from the left edge of its container.

One method of animation involves “floating” a picture across a background.

After loading a picture you can move the image by changing the properties of the picture.

Image 1.top = Image 1.top + 10 moves the image down 10

Image 1.left = Image 1.left + 10 moves the image right 10

If you put this in a loop the image will move across the screen

You can also change the scale or size of an image. After loading an image you can change the width and the height of the image.

Image1. Height = Image1.Height + 100

Image1.Width = Image1.Width + 100

If you put this in a loop the picture will increase in size. Make sure that STRETCH is set to True in the properties for the image.

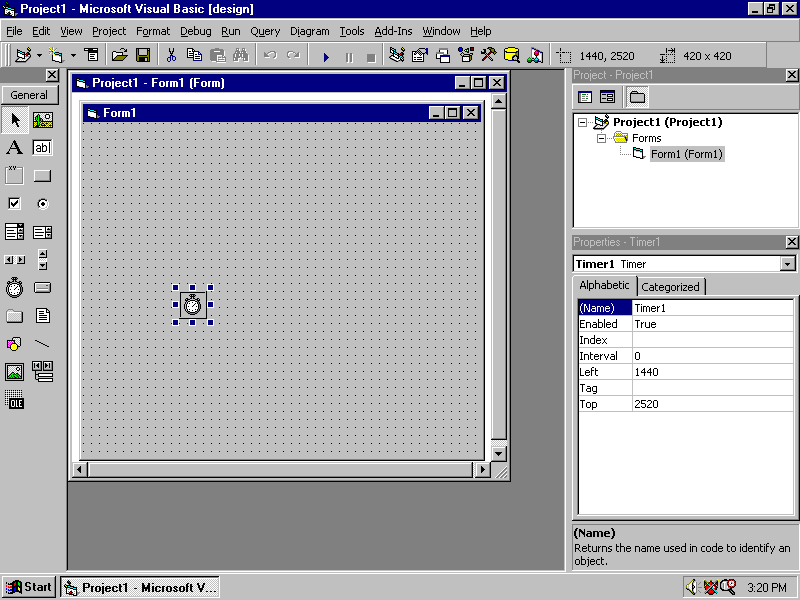
## **Using Timers**

A timer is like an invisible stop watch. It can be used to delay a program or repeat an action after a given interval of time has passed.

The time interval is in milliseconds ( 1/1000 of a second).

Once a timer is enabled , it runs constantly until the program ends or the timer is disabled.

To use a timer, place the timer control anywhere on the form. The timer will not be visible when the program is running. Set the time interval. The code stored under Timer1 will execute once in every specified time interval.



Timers can be used for animation and in designing games.

### Using A Timer For Animation

### Simple Animation

Start a new project. Place an image and a timer on the form. Load any image.

To move an object across the screen you would change the following properties :

Object Property Setting

Timer1 Interval 400

Enabled True

Double-click the timer object on the form and enter the following code :

Image1.left = image1.left + 100

When you run the program the image will move right every 400 milliseconds.

### Advanced Animation

Start a new project. Place two images in the same place on the form. Load an image in one position in image1 and then the same image in a different position in image2.

To alternate between the three images you would change the following properties :

|  |  |  |
| --- | --- | --- |
| Object | Property | Setting |
| Timer1 | Interval | 400 |
| Enabled | True |
| Image1 | Visible | True |
| Image2 | Visible | False |
| Image3 | Visible | False |

Double-click the timer1 object on the form and enter the following code :

if image1.visible = true then ‘ if you see image 1

Image1.visible = false ‘switch to image2

Image2.visible = true

elseif image2.visible = true then

Image2.visible = false

Image3.visible = true

elseif image3.visible = true then

Image3.visible = false

Image1.visible = true

endif

When you run the program the images will alternate every 400 milliseconds.

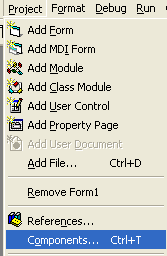
## 

## **Adding Sounds**

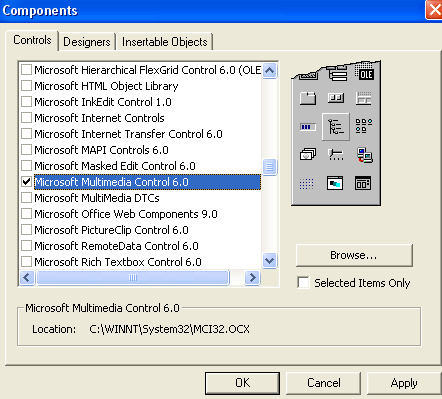
To add sound to your program, you first need to place a Microsoft Multimedia Control 6.0 on your form.

Complete the following steps:

1. Go to Project and Components



2. Find the Multimedia control in

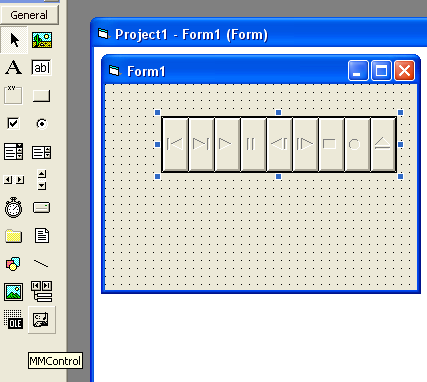


the list of components. Check

the control and press OK. This

will add the control to the

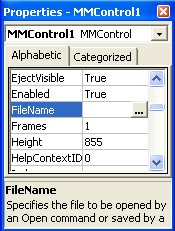
usual collection of controls.



1. Place the control on your

form.

1. In the properties menu , load the wave file that contains the sound you want to play.



1. Now add the following code to play the sound. The code might be placed in a command button.

MMControl1.DeviceType = “WaveAudio”

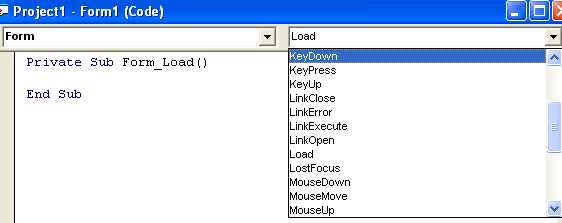
MMControl1.Command=”Open”

MMControl1.From = 0

MMControl1.Command = “Play”

## **Using Arrow keys to Move Objects**

Double click on the form to view the code window. Click the right hand box to view options other than load. Click on the KeyDown option.



Enter the following code :

Private Sub Form\_KeyDown(KeyCode As Integer, Shift As Integer)

*Select Case KeyCode*

*Case vbKeyLeft*

*Image1.left=image1.left-100*

*Case vbKeyRight*

*Image1.left=image1.left+100*

*Case vbKeyUp*

*Image1.top=image1.top-100*

*Case vbKeyDown*

*Image1.top=image1.top-100*

*End Select*

End Sub

This code will move the object left, right, up and down as you press the arrow keys. You may use any keys that have a vb designation.

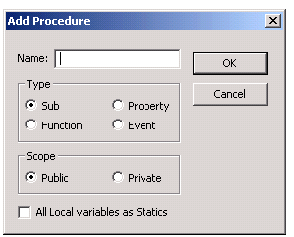
WARNING : This code will only work if all commands on the form are disabled and you have used image boxes, not picture boxes. You can disable a command with the following code : *command1.enabled=false*

Procedures

General procedures may be added to the Code window in two ways:

1. Type **Sub**, followed by the procedure name and then press Enter. Visual Basic then automatically adds an End Sub statement and places the cursor in the body of the procedure.
2. With the Code window open, select **Tools**→**Add Procedure** from the menu bar.

The> Add Procedure command displays a dialogue box, as shown. From the dialogue box, the programmer selects the procedure name, type and scope. After selecting OK, the procedure is placed in alphabetical order in the Code window



A procedure often needs additional data to be able to complete its task. The extra data is enclosed in parentheses at the end of the procedure call. When the procedure is called, the extra data is given, or **passed**, to it.

For example, in the statement

Call GiveClue (intSecretNumber, intGuess)

the GiveClue procedure is called and two values are passed to it—intSecretNumber and intGuess.

A variable or a value that is passed to a procedure is called an **argument**. In our example above, intSecretNumber and intGuess are arguments that will be used by the procedure.

A procedure that requires arguments is declared with **parameters**. The following syntax is used for a procedure with parameters:

**Sub** ProcedureName(ByVal parameter1 As Type, …)  
    statements  
**End Sub**

Where:

* ProcedureName is the name of the procedure.
* ByVal indicates that the parameter is a value parameter.
* parameter1 is the name of the parameter.
* As Type is the data type of the parameter.
* ...statements is the body of the procedure.

It is possible to have multiple parameters. Each parameter must be separated by a comma.

|  |  |
| --- | --- |
| Once again, let's look at the GiveClue procedure:  **Sub** GiveClue(ByVal intFirstNum As Integer, ByVal intSecondNum As Integer)     **If** intFirstNum > intSecondNum **Then**       MsgBox "Too low"     **Else**       MsgBox "Too high"     **End If** **End Sub**  When the GiveClue procedure is called with this statement:  Call GiveClue (intSecretNumber, intGuess)   * The value of the first argument, intSecretNumber, is assigned to the first parameter, intFirstNum. * The value of the second argument, intGuess, is assigned to the second parameter, intSecondNum.   The image shows the two arguments in the Call statement being passed to the parameters declared in the GiveClue procedure.  Capture.PNG |  |

When a procedure is called, the number of arguments passed to it must match the number of parameters that have been declared.   
  
It is important to remember the following points:

The order in which arguments are passed is important, as:

* The order of the arguments must correspond to the order of the parameters.
* For example, the first argument in the procedure call (intSecretNumber) corresponds to the first parameter(intFirstNum) and so on.

Arguments passed by value can be in the form of a constant, a variable, values or expressions.   
  
For example, the GiveClue Call statement could look like any of these:

* Call GiveClue(intSecretNumber, intGuess)
* Call GiveClue(2\*5, 10\*2)
* Call GiveClue(10, intGuess)
* Call GiveClue(10,20)

Variable arguments that are passed by value are not changed by the procedure. For example, examine each line of code and its corresponding comment in green type:

**Sub** Demo( )  
    **Dim** intCounter **As Integer** 'declares variable intCounter  
    intCounter = 1 'assigns value of 1 to intCounter  
    Call ShowCount(intCounter) 'call ShowCount procedure and passes\_  
        copy of intCounter value(1) as argument to ShowCount procedure  
    lblNumber.Caption = intCounter 'Demo procedure displays value of\_  
        intCounter(1) in the label. Value of intCounter(1) is\_  
        unaffected by the ShowCount procedure  
**End Sub**

**Sub** ShowCount(**ByVal** intCounter **As Integer**) 'parameter intCount\_  
        receives value of 1 passed to it  
    intCounter = intCounter + 1 'ShowCount procedure increments its\_  
        intCounter variable by one to make value of intCounter 2  
    MsgBox intCounter 'ShowCount procedure displays value of\_   
        intCounter(2) in the message box  
**End Sub**

A function usually has at least one parameter. This parameter is for data that is required for the function to be able to perform its task. Parameters are ByVal because a function performs a task and returns a single value. The function should not change the arguments that it has been passed.

Functions are useful for verifying user input. For example, an application that asks the user to enter a value between one and ten could use a user-defined Boolean function to determine if the user input falls within the correct range.

The following is the code for a user-defined function called ValidEntry. This function will be used to check that the values entered by the user are within a predetermined range. The function parameters set the lower limit of the range to one and the upper limit of the range to ten.

**Function** ValidEntry(**ByVal** intUserNum **As Integer**, **ByVal** intUpperLimit **As Integer**, \_ **ByVal** intLowerLimit **As Integer**)**As Boolean**  
    **If** intUserNum > intUpperLimit Or intUserNum < intLowerLimit **Then**  
        ValidEntry = False  
    **Else**  
        ValidEntry = True  
    **End If**  
**End Function**

This code uses the ValidEntry function to check the user input:

…  
intGuess = txtUserGuess.Text  
**If Not** ValidEntry(intGuess, 10,1) **Then**  
    MsgBox "Invalid Guess. Please try again."  
**End If**

When working with functions, remember:

* The order of the arguments must match the order of the parameters.
* Only ByVal parameters should be declared in a function.
* A function should not change the arguments that are passed.
* A function returns a single value. This value will be used in another statement such as an assignment statement. For example, in the code shown in the preceding lesson, the Boolean value that is returned becomes the condition of the If…Then statement.

## **Modular Programming**

In order to make a program easier to follow and easier to produce, a large program is broken down into smaller isolated parts.

### Passing Parameters

Variables are sent form the mainline to the subprogram with the Call command.

Ex. CALL drawsquare ( x% , y% , color%) sends the variables x%, y% and colour%

to the function drawsquare.

Sub drawsquare(x%,y%,color%)

Line (10+x%,10+y%)-(20+x%,20+y%), color%,bf

End Sub

Ex. CALL drawsquare ( x% , y% , 4 ) sends the variables x%, y% and the colour 4

to the function drawsquare.

Sub drawsquare(x%,y%,color%)

Line (10+x%,10+y%)-(20+x%,20+y%), color%,bf

End Sub

Warning : The variables must be in the same order in both the Call statement and the function declaration.

# Designing a Major Program

## **Writing The Code**

A major program is best tackled by first planning the project in words. Then group the code into sections. These sections will become your sub programs. Design and test the mainline before starting the subprograms so as to identify any problems with the main engine of the program before committing to the hours of work required to program the sub programs. When you are confident that the program will run probably, write and test each sub program. Get a simple version of the program working before adding any fancy features.

Ex. Write the program for space invaders.

Screen 12

Call TitlePage

Call Instructions

Call Background

Do

Key$ = inkey$

‘draw

Call Ship(sx%,sy%,15) ‘ draw the ship in white

Call Invader (ix%,iy%,4) ‘draw an invader in white

‘pause

Call Pause(0.05)

‘erase

Call Ship(sx%,sy%,0) ‘ draw the ship in background colour

Call Invader (ix%,iy%,0) ‘draw an invader in background colour

‘move

Call MoveShip(sx%,sy%,key$) ‘move the ship according to the pressed key

Call MoveInvader (ix%,iy%) ‘move the invader

Loop Until key$ = “q”

In order to test this mainline we have to write something in each subprogram. For testing purposes, simply write words to describe what the subprogram does.

Ex.

Sub Background

Print “This is the background”

End Sub

When you are satisfied that the program runs the way you intended it to run, you can now add the real code for each subprogram.

## **Common Questions**

### How do I know when two objects collide ?

We use the math formula d = √ ( x2 – x1)2 + ( y2 – y1)2 to detect a collision between two objects.

Ex. Suppose you have a circle and a box that are moving across the screen at the same time. The following code will detect a collision of the two objects.

First determine the center of each object. To do this you have to look at the code for each drawing :

Circle ( 50+cx%, 80+cy%),20,4

|  |  |  |  |
| --- | --- | --- | --- |
|  | X co-ordinate of center | Y co-ordinate of center | Notes |
| Circle | 50 + cx% | 80+cy% | Take the co-ordinates from the center |
| Box | 120+bx% | 60+by% | Find the halfway points of the edges |

Line(100+bx%, 40+by%)-(140+bx%, 80+by%),1,bf

Now use the formula to calculate the distance between the two centers.

D% = sqr ( (50 + cx% - 120 –bx%)^2+(80+cy%-60-by%)^2 ) ‘calculate the distance

If D% < 30 then ‘ if the distance is small enough

lives% = lives% - 1 ‘lose a life or gain points

cx% = int(rnd\*600) ‘put object in new location after the collision

cy% = 0

end if

### How do I get more than one object?

If you want to have full control of each object, then each object should be called separately from the mainline. Simply call the subprogram that creates the object and send the subprogram different variables.

Do

Key$ = inkey$

‘draw

Call Ship(sx%,sy%,15) ‘ draw the ship in white

Call Invader (ix%,iy%,4) ‘draw an invader in white

*Call Invader (ix2%,iy2%,4) ‘draw a second invader*

*Call Invader (ix3%,iy3%,4) ‘draw a third invader*

‘pause

Call Pause(0.05)

‘erase

Call Ship(sx%,sy%,0) ‘ draw the ship in background colour

Call Invader (ix%,iy%,0) ‘draw an invader in background colour

*Call Invader (ix2%,iy2%,0) ‘erase the second invader*

*Call Invader (ix3%,iy3%,0) ‘erase the third invader*

‘move

Call MoveShip(sx%,sy%,key$) ‘move the ship according to the pressed key

Call MoveInvader (ix%,iy%) ‘move the invader

*Call MoveInvader (ix2%,iy2%) ‘move the second invader*

*Call MoveInvader (ix3%,iy3%) ‘move the third invader*

Loop Until key$ = “q”

### How do I shoot?

You will need a variable that determines when you are shooting. When you are shooting you will draw the bullet, erase the bullet and move the bullet. The shoot variable will be set to yes in the subprogram move.

Do

Key$ = inkey$

‘draw

Call Ship(sx%,sy%,15) ‘ draw the ship in white

Call Invader (ix%,iy%,4) ‘draw an invader in white

*If shoot$ = “yes” then*

*Call Bullet (bx%,by%)*

*End if*

‘pause

Call Pause(0.05)

‘erase

Call Ship(sx%,sy%,0) ‘ draw the ship in background colour

Call Invader (ix%,iy%,0) ‘draw an invader in background colour

*If shoot$ = “yes” then*

*Call EraseBullet (bx%,by%)*

*End if*

‘move

Call MoveShip(sx%,sy%,bx%,by%,key$ , shoot$) ‘move the ship according to the

‘pressed key

Call MoveInvader (ix%,iy%) ‘move the invader

*If shoot$ = “yes” then*

*Call MoveBullet (bx%,by%)*

*End if*

Loop Until key$ = “q”

sub MoveShip(sx%,sy%,bx%,by%,key$ , shoot$)

if key$ = “4” then

sx% = sx% - 10

elseif key$ = “8” then

sx% = sx% +10

elseif key$ = “5” then ‘when the user presses the fire button

shoot$ = “yes” ‘ set shoot to yes

bx% = sx% ‘ and place the bullet at the same location as the ship

by% = 0

endif

end sub

### How do I program Pacman ?

Pacman must be programmed using an array. The array stores the maze information in a chart and pacman is allowed to move according to the numbers stored in the chart.

Ex. We can write the program so that pacman can move over the 0’s but not over the 1’s. The 0’s correspond to pathways and the 1’s to walls.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| 1 | 0 | 0 | 1 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 1 |
| 1 | 0 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 0 | 1 |
| 1 | 0 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 0 | 1 |
| 1 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 1 | 0 | 0 | 1 |
| 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |

Creating the maze

The maze is created by storing the 0’s and 1’s in an array.

Dim maze%(6,21) ‘first create the maze. This maze has 6 rows and 21 columns

For i% = 1 to 6

For j% = 1 to 21

Maze%(i%,j%)=0 ‘ set every values to 0

Next j%

Next i%

For i% = 1 to 6

Maze%(i%,1) =1 ‘ set column1 to 1 to represent the wall on the left

Next i%

For j% = 1 to 21

Maze%(1,j%)=1 ‘ set row 1 to 1 to represent the top wall

Next j%

‘continue to set all the values in the maze to 1 to represent the maze

Moving in the maze

Create a sub program to move pacman. The subprogram needs to have the following information :

x% , y% - pacman’s location on the screen

row%, column% - pacman’s location in the maze

Maze%( ) – the values in the maze

Key$ - the key that was pressed by the user

Sub MovePacman (x% , y% , row% , column% , maze%( ), key$)

‘move right

If key$ = “6” and maze%( row%, column% +1) =0 then ‘ if the user chooses right and the space is open

X% = x% + 40 ‘ update position on screen

Column% = column% + 1 ‘ and update position in maze

Elseif key$ = “4” and maze%( row%, column%-1 ) =0 then ‘user chooses to move left

X% = x% - 40

Column% = column% - 1

Elseif key$ = “8” and maze%( row%-1, column% ) =0 then ‘user chooses to move up

y% = y% - 30

row% = row% - 1

Elseif key$ = “2” and maze%( row%+1, column% ) =0 then ‘user chooses to move down

y% = y% + 30

row% = row% + 1

endif

End Sub

Create a sub program to move the ghosts. The subprogram needs to have the following information :

gx% ,gy% - ghost’s location on the screen

grow%, gcolumn% - ghost’s location in the maze

Maze%( ) – the values in the maze

Sub MoveGhost (gx% , gy% , grow% , gcolumn% , maze%( ))

r% = int(rnd\*4)+1 ‘ pick a random movement

‘move right

If r% = 1and maze%( grow%, gcolumn% +1) =0 then ‘ if the ghost chooses right and the space is open

gx% = gx% + 40 ‘ update position on screen

gcolumn% = gcolumn% + 1 ‘ and update position in maze

Elseif r% = 2and maze%( grow%, gcolumn%-1 ) =0 then ‘ghost chooses to move left

gX% = gx% - 40

gColumn% = gcolumn% - 1

Elseif r% = 3 and maze%( grow%-1, gcolumn% ) =0 then ‘ghost chooses to move up

gy% = gy% - 30

grow% = grow% - 1

Elseif r% = 4 and maze%( grow%+1, gcolumn% ) =0 then ‘ghost chooses to move down

gy% = gy% + 30

grow% = grow% + 1

endif

End Sub

## **Modular Programming**

In order to make a program easier to follow and easier to produce, a large program is broken down into smaller isolated parts.

### Passing Parameters

Variables are sent form the mainline to the subprogram with the Call command.

Ex. CALL drawsquare ( x% , y% , color%) sends the variables x%, y% and colour%

to the function drawsquare.

Sub drawsquare(x%,y%,color%)

Line (10+x%,10+y%)-(20+x%,20+y%), color%,bf

End Sub

Ex. CALL drawsquare ( x% , y% , 4 ) sends the variables x%, y% and the colour 4

to the function drawsquare.

Sub drawsquare(x%,y%,color%)

Line (10+x%,10+y%)-(20+x%,20+y%), color%,bf

End Sub

Warning : The variables must be in the same order in both the Call statement and the function declaration.

# Designing a Major Program

## **Writing The Code**

A major program is best tackled by first planning the project in words. Then group the code into sections. These sections will become your sub programs. Design and test the mainline before starting the subprograms so as to identify any problems with the main engine of the program before committing to the hours of work required to program the sub programs. When you are confident that the program will run probably, write and test each sub program. Get a simple version of the program working before adding any fancy features.

Ex. Write the program for space invaders.

Screen 12

Call TitlePage

Call Instructions

Call Background

Do

Key$ = inkey$

‘draw

Call Ship(sx%,sy%,15) ‘ draw the ship in white

Call Invader (ix%,iy%,4) ‘draw an invader in white

‘pause

Call Pause(0.05)

‘erase

Call Ship(sx%,sy%,0) ‘ draw the ship in background colour

Call Invader (ix%,iy%,0) ‘draw an invader in background colour

‘move

Call MoveShip(sx%,sy%,key$) ‘move the ship according to the pressed key

Call MoveInvader (ix%,iy%) ‘move the invader

Loop Until key$ = “q”

In order to test this mainline we have to write something in each subprogram. For testing purposes, simply write words to describe what the subprogram does.

Ex.

Sub Background

Print “This is the background”

End Sub

When you are satisfied that the program runs the way you intended it to run, you can now add the real code for each subprogram.

## **Common Questions**

### How do I know when two objects collide ?

We use the math formula d = √ ( x2 – x1)2 + ( y2 – y1)2 to detect a collision between two objects.

Ex. Suppose you have a circle and a box that are moving across the screen at the same time. The following code will detect a collision of the two objects.

First determine the center of each object. To do this you have to look at the code for each drawing :

Circle ( 50+cx%, 80+cy%),20,4

|  |  |  |  |
| --- | --- | --- | --- |
|  | X co-ordinate of center | Y co-ordinate of center | Notes |
| Circle | 50 + cx% | 80+cy% | Take the co-ordinates from the center |
| Box | 120+bx% | 60+by% | Find the halfway points of the edges |

Line(100+bx%, 40+by%)-(140+bx%, 80+by%),1,bf

Now use the formula to calculate the distance between the two centers.

D% = sqr ( (50 + cx% - 120 –bx%)^2+(80+cy%-60-by%)^2 ) ‘calculate the distance

If D% < 30 then ‘ if the distance is small enough

lives% = lives% - 1 ‘lose a life or gain points

cx% = int(rnd\*600) ‘put object in new location after the collision

cy% = 0

end if

### How do I get more than one object?

If you want to have full control of each object, then each object should be called separately from the mainline. Simply call the subprogram that creates the object and send the subprogram different variables.

Do

Key$ = inkey$

‘draw

Call Ship(sx%,sy%,15) ‘ draw the ship in white

Call Invader (ix%,iy%,4) ‘draw an invader in white

*Call Invader (ix2%,iy2%,4) ‘draw a second invader*

*Call Invader (ix3%,iy3%,4) ‘draw a third invader*

‘pause

Call Pause(0.05)

‘erase

Call Ship(sx%,sy%,0) ‘ draw the ship in background colour

Call Invader (ix%,iy%,0) ‘draw an invader in background colour

*Call Invader (ix2%,iy2%,0) ‘erase the second invader*

*Call Invader (ix3%,iy3%,0) ‘erase the third invader*

‘move

Call MoveShip(sx%,sy%,key$) ‘move the ship according to the pressed key

Call MoveInvader (ix%,iy%) ‘move the invader

*Call MoveInvader (ix2%,iy2%) ‘move the second invader*

*Call MoveInvader (ix3%,iy3%) ‘move the third invader*

Loop Until key$ = “q”

### How do I shoot?

You will need a variable that determines when you are shooting. When you are shooting you will draw the bullet, erase the bullet and move the bullet. The shoot variable will be set to yes in the subprogram move.

Do

Key$ = inkey$

‘draw

Call Ship(sx%,sy%,15) ‘ draw the ship in white

Call Invader (ix%,iy%,4) ‘draw an invader in white

*If shoot$ = “yes” then*

*Call Bullet (bx%,by%)*

*End if*

‘pause

Call Pause(0.05)

‘erase

Call Ship(sx%,sy%,0) ‘ draw the ship in background colour

Call Invader (ix%,iy%,0) ‘draw an invader in background colour

*If shoot$ = “yes” then*

*Call EraseBullet (bx%,by%)*

*End if*

‘move

Call MoveShip(sx%,sy%,bx%,by%,key$ , shoot$) ‘move the ship according to the

‘pressed key

Call MoveInvader (ix%,iy%) ‘move the invader

*If shoot$ = “yes” then*

*Call MoveBullet (bx%,by%)*

*End if*

Loop Until key$ = “q”

sub MoveShip(sx%,sy%,bx%,by%,key$ , shoot$)

if key$ = “4” then

sx% = sx% - 10

elseif key$ = “8” then

sx% = sx% +10

elseif key$ = “5” then ‘when the user presses the fire button

shoot$ = “yes” ‘ set shoot to yes

bx% = sx% ‘ and place the bullet at the same location as the ship

by% = 0

endif

end sub

### How do I program Pacman ?

Pacman must be programmed using an array. The array stores the maze information in a chart and pacman is allowed to move according to the numbers stored in the chart.

Ex. We can write the program so that pacman can move over the 0’s but not over the 1’s. The 0’s correspond to pathways and the 1’s to walls.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| 1 | 0 | 0 | 1 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 1 |
| 1 | 0 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 0 | 1 |
| 1 | 0 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 0 | 1 |
| 1 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 1 | 0 | 0 | 1 |
| 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |

Creating the maze

The maze is created by storing the 0’s and 1’s in an array.

Dim maze%(6,21) ‘first create the maze. This maze has 6 rows and 21 columns

For i% = 1 to 6

For j% = 1 to 21

Maze%(i%,j%)=0 ‘ set every values to 0

Next j%

Next i%

For i% = 1 to 6

Maze%(i%,1) =1 ‘ set column1 to 1 to represent the wall on the left

Next i%

For j% = 1 to 21

Maze%(1,j%)=1 ‘ set row 1 to 1 to represent the top wall

Next j%

‘continue to set all the values in the maze to 1 to represent the maze

Moving in the maze

Create a sub program to move pacman. The subprogram needs to have the following information :

x% , y% - pacman’s location on the screen

row%, column% - pacman’s location in the maze

Maze%( ) – the values in the maze

Key$ - the key that was pressed by the user

Sub MovePacman (x% , y% , row% , column% , maze%( ), key$)

‘move right

If key$ = “6” and maze%( row%, column% +1) =0 then ‘ if the user chooses right and the space is open

X% = x% + 40 ‘ update position on screen

Column% = column% + 1 ‘ and update position in maze

Elseif key$ = “4” and maze%( row%, column%-1 ) =0 then ‘user chooses to move left

X% = x% - 40

Column% = column% - 1

Elseif key$ = “8” and maze%( row%-1, column% ) =0 then ‘user chooses to move up

y% = y% - 30

row% = row% - 1

Elseif key$ = “2” and maze%( row%+1, column% ) =0 then ‘user chooses to move down

y% = y% + 30

row% = row% + 1

endif

End Sub

Create a sub program to move the ghosts. The subprogram needs to have the following information :

gx% ,gy% - ghost’s location on the screen

grow%, gcolumn% - ghost’s location in the maze

Maze%( ) – the values in the maze

Sub MoveGhost (gx% , gy% , grow% , gcolumn% , maze%( ))

r% = int(rnd\*4)+1 ‘ pick a random movement

‘move right

If r% = 1and maze%( grow%, gcolumn% +1) =0 then ‘ if the ghost chooses right and the space is open

gx% = gx% + 40 ‘ update position on screen

gcolumn% = gcolumn% + 1 ‘ and update position in maze

Elseif r% = 2and maze%( grow%, gcolumn%-1 ) =0 then ‘ghost chooses to move left

gX% = gx% - 40

gColumn% = gcolumn% - 1

Elseif r% = 3 and maze%( grow%-1, gcolumn% ) =0 then ‘ghost chooses to move up

gy% = gy% - 30

grow% = grow% - 1

Elseif r% = 4 and maze%( grow%+1, gcolumn% ) =0 then ‘ghost chooses to move down

gy% = gy% + 30

grow% = grow% + 1

endif

End Sub

### Redefining Coordinate Systems

In order to draw a graph you will first want to redefine the coordinate system for your screen. WINDOW maps the screen on to the screen on to a coordinate system of your choice. For example you can define the top left corner to be location (-10,10) and the bottom right corner to be (10, - 10) and then reference all pixels with an x and y value between – 10 and 10. The x values increase towards the right and the y values increase upwards.

### Creating Graphs

A graph is just a series of x and y values the are determined by an equation. Therefore once the values are determined we can just use the PSET command to plot the points.

Ex. Sketch the graph of y = 2x +5

Screen 12

Window (-10,10) – (10,-10) ‘ redefine screen

Line (-10,0)-(10-0), 1 ‘ draw the x and y axes

Line (0,-10)-(0,10), 1

X! = -10 ‘ start with the smallest x value

Y! = 0

For I% = 1 to 20000

Y! = 2\*x! + 5 ‘calculate the value of y

Pset(x!,y!),4 ‘ plot the point

Next I% ‘ move to the next x value

End

## **Common Questions in Visual Basic**

### How do I know when two objects collide ?

We use the math formula d = √ (x2– x1)2 + ( y2 – y1)2 to detect a collision between two objects.

Ex. Determine when a circle (image1) and a square (image2) collide.

|  |  |  |
| --- | --- | --- |
|  | X co-ordinate of center | Y co-ordinate of center |
| Circle | X1 =image1.left + 0.5\*image1.width | Y1 =Image1.top + 0.5\*image1.height |
| Box | x2 = Image2.left + 0.5\*image2.width | Y2 =Image2.top + 0.5\*image2.height |

Now use the formula to calculate the distance between the two centers.

D% = sqr ((x2– x1)^2+(y2– y1)^2 ) ‘calculate the distance

If D% < 1000 then ‘ if the distance is small enough

lives% = lives% - 1 ‘lose a life or gain points

image1.left% = int(rnd\*10000) ‘put object in new location after the collision

image1.top = 0

end if

### How do I program a board game ?

Each piece will have a timer that looks after the movement of the piece.

The subprogram will need to know the current location of the piece and move the piece one square at a time.

|  |  |  |  |
| --- | --- | --- | --- |
| 1 | 2 | 3 | 4 |
| 8 | 7 | 6 | 5 |
| 9 | 10 | 11 | End |

Ex. If the board looked like the table to the right then the code for moving the piece (image1) would look like the following :

If roll > 0 then ‘ keep moving until you have moved once for every number on the dice

If player1location <4 then

Image1.left = image1.left + 1200 ‘ move piece one square to the right

elseIf player1location =4 then

Image1.top = image1.top + 1200 ‘ move piece one square down

elseIf player1location >4 and player1location < 8 then

Image1.left = image1.left - 1200 ‘ move piece one square to the left

elseIf player1location =8 then

Image1.top = image1.top + 1200 ‘ move piece one square down

elseIf player1location >8 and player1location < 12 then

Image1.left = image1.left + 1200 ‘ move piece one square to the right

endif

player1location = player1location + 1 ‘update the player location

roll = roll -1

else

timer1.enabled=false

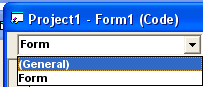
endif

# Appendix

### Declaring Variables

Variables in sub programs can only be changed by that sub program. If you would like a variable to be changed by more than one sub program the variable must be dimensioned. Variables that are going to be used in more than one form must be declared as Global.

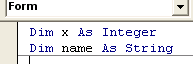
1. Variables used by more than one function.



a) In the code window ,

choose the (General ) option.

b) Dimension any variables that will

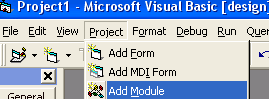


be changed by more than one function.

c) Variables can be dimensioned as

any of the following types :

|  |  |
| --- | --- |
| Data type | Description |
| Integer | Integers from -32,768 to 32,767 |
| Long | Integers from –2,147,483,648 to 2,147,483,647 |
| Single | Decimals from –3.402823E+38 to 3.402823E+38 |
| Double | Decimals –1.79769313486232E+308 to –1.79769313486232E+308 |
| String | Characters |
| Boolean | TRUE or FALSE |
| Date | Holds date and time values |
| Variant | Data of any data type |

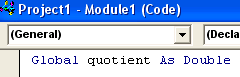


2. Variables used by more than one form.

a) Add a new module

Click the Add Module command on the project

menu and then click Open



b) Declare variables as Global

Variables can be initialized when they are declared.

Ex. Dim highscore as integer

Highscore = 100

Ex. Dim Message as string

Message = “You Win”

Variable names:

* Should be clearly understandable to the reader
* Should be descriptive of the contents of the variable
* Should include a three-letter prefix that describes the type of data the variable will hold

Regardless of the type of variable, the following rules apply when choosing the variable identifier:

1. Variable names must begin with a letter.
2. Variable names can contain letters, numbers and the underscore (\_) character.
3. Variable names can be as long as 255 characters or as short as only one character.
4. Variable names can **not** be Visual Basic keywords. Keywords are words that have a predefined meaning to the Visual Basic compiler. For example, **Double** is a Visual Basic keyword.
5. Variable names can **not** contain spaces, periods and other special characters such as \, \*, ?, etc.

The following are examples of legal variable identifiers:

* intStrikeouts
* strThisIsACrazyVariableNameButItIsLegal
* dblAverageScore
* dblTotalSalary
* dblTaxRate\_2003
* intSum
* intX

**Note:** It is good programming style to precede the identifier with a three-letter prefix indicating the data type. The following table shows the data types and their prefixes:

|  |  |
| --- | --- |
| **Data Type** | **Prefix** |
| Single | sgl |
| Double | dbl |
| Integer | int |
| Long | lng |
| Currency | cur |
| String | str |
| Boolean | bln |

Variable declaration statements reserve space in the computer's memory for a value.

# QBASIC Assignments

## **Assignment #1**

1. A) Use the editor to enter the following program. Type exactly what is written.

INPUT “ Type in your name and press the enter key “ , name$

PRINT “ Hello”, name$

END

Run the program. Be able to describe what the program does.

Save the program on your U: drive as assign1a.bas .

1. B) Open a new file and type in the following program.

PRINT “ My name is Robin Smith “”

PRINT “ I am in Grade 11 ”

PRINT “ My favourite activity is playing soccer”

END

# 

Run the program and see what the program does. Change the information to your name, grade and favorite activity.

Save the program on your U: drive as assign1b.bas .

1. C) Open a new file and type in the following program :

SCREEN 12

PSET ( 100, 200) , 4

LINE STEP ( 0,0) – STEP ( 50,50), 4, BF

END

Use COPY and PASTE to copy the third line of the program and paste it in 3 times before the end command. When you run the program you should see three squares on the screen.

Save the program on your M: drive as assign1c.bas .

If you finish early then either experiment with the last program and see what you can create or read about INPUT and OUTPUT and try assignment #2.

## **Assignment #2**

(1) A) Input 2 numbers , calculate and display their sum, product, quotient and

difference. Test your program with the following numbers :

i) 10 and 2 sum = 12 , product = 20, quotient = 5, difference = 8

ii) 8 and 3 sum = 11 , product = 24, quotient = 2.66666, difference = 5

(1) B) Input 5 numbers and determine the mean (average). Test your program with the

following numbers :

a) 1,2,3,4,5 mean = 3 b) 8,10,11,12,13 mean = 10.8

(1) C) Determine the standard deviation for 5 numbers. To calculate the standard

deviation you must do the following :

i. Input the 5 numbers

ii. Determine the mean

iii. Find the difference between each number and the mean

iv. Square the differences

v. Find the averages of the squares

vi. Take the square root of the average

Test your program with the following numbers : 1,2,3,4,5

|  |  |  |
| --- | --- | --- |
| Number | Difference from mean | Square of difference |
| 1 | -2 | 4 |
| 2 | -1 | 1 |
| 3 | 0 | 0 |
| 4 | 1 | 1 |
| 5 | 2 | 4 |

standard deviation = 1.414

(1) D) The solution to any quadratic equation ax2 + bx + c =0 can be solved using the quadratic formula :

x1 = -b + √ b^2 - 4ac and x2 = -b - √ b^2 - 4ac

2a 2a

Input 3 numbers a,b and c for the quadratic equation and then calculate x1and x2 using the quadratic formula . In other words, translate the formula in to something that computer can understand.

Test your program with the following numbers:

a) a = 1, b = -2 , c = 1 x1 =1 and x2= 1 b) a = 6, b = 1, c = -1 x1 =1/3 and x2= -1/2

## **Assignment #3**

(1) A) Use a loop to find the sum of the first 100 integers i.e. 1 + 2 + 3 + … + 100

(1) B) Modify the previous program to allow the user to type in a starting number and

an ending number and find the sum of all the numbers between them.

ie 17 + 18 + 19 + … + 301

(1) C) Input a starting number, a difference number and the total number of numbers

you want to add. Use a loop to determine the sum of the numbers.

Ex. Start = 10 difference = 3 total = 5

Sum = 10 + 13 + 16 + 19 + 22

Difference of 3

(1) D) Use a loop to calculate n!. Allow the user to input the number n.

Ex. 8! = 1x2x3x4x5x6x7x8

11! = 1x2x3x4x5x6x7x8x9x10x11

Test your program with the following numbers :a) n = 5 , n! = 120 b) n = 8, n! = 40320

WARNING : The program will crash when trying to calculate 8! if your answer is

defined as an integer. Integers in computer science can only be as big as 32,000.

## **Assignment #4**

(1) A) Write a program that will add a series of numbers until the user decides to quit.

Print the sum and the total number of number that were added.

Ex. 5

10

23

quit sum = 38, there were 3 numbers added

(1) B) Write a program that will randomly generate a number from 1 to 11

Use the following code to generate the random number

Randomize timer ‘*this line appears once in the program*

r% = int (rnd \* 11 ) +1 ‘ *use this statement everytime you want a number*

Continue to generate random numbers until the sum of the numbers is greater than 21 or the user decides to quit. Print the numbers, the sum and a message telling the user if they were over or under 21.

## **Assignment #5**

(1) A) Write a program that will allow the user to input two numbers. Give the user the choice of calcutaing the sum or product of the two numbers. Print the answer.

(1) B) Write a program that will print the letter grade when a number grade is entered.

|  |  |
| --- | --- |
| Number grade | Letter grade |
| 90-100 | A |
| 80-89 | B |
| 70-79 | C |
| 60-69 | D |
| 0-59 | F |

(1) C) Write a program that will generate a random number from 1 to 100. Allow the user to guess the number. Tell the user to guess higher or lower until the user guesses the correct number. Tell the user how many guesses it took to get the correct number.

(5) D) Create a game. You will be marked according to the level of difficulty, quality of programming, number of options, different levels and overall impression of your game.

Ex.

* Create a multiple choice quiz game.
* Design Who Wants To Be A Millionaire
* Create a memory game where a number appears for a few seconds and the user has to remember the number. As the game progresses the user has to remember more numbers.
* Design hangman.
* Create a math game where random addition questions appear and the user has determine the answer.

|  |  |
| --- | --- |
| Mark |  |
| 1 | Game works |
| 2 | Includes title page and instructions |
| 3 | Game can be played again |
| 4 | There is a way to win or lose / There is a rating scheme |
| 5 | Game is different every time |

## **Assignment #6**

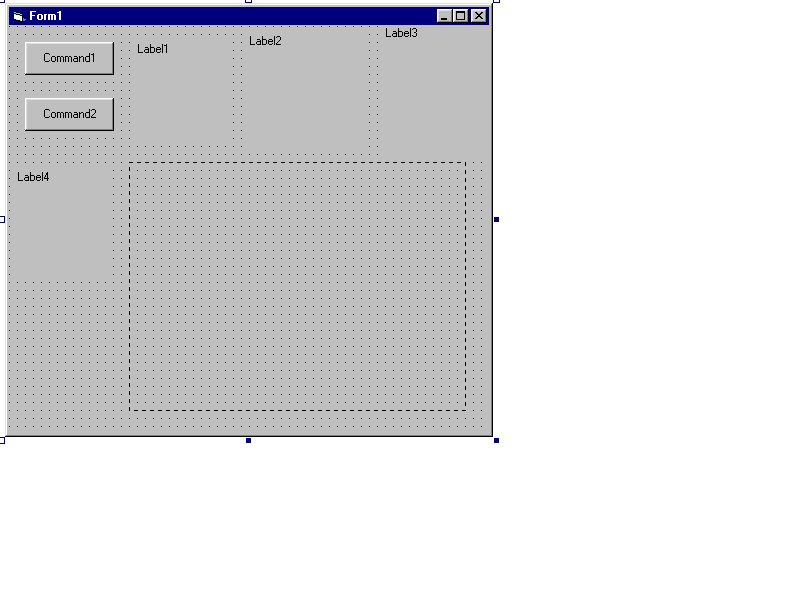
(1) A) Enter the sample program in Visual Basic. Run the program to make sure that it works. Change the properties of the controls to make the output more interesting (change the colours or the font or the picture or …). Save the program on your drive in a folder called Assignment #6.

## **Example Program**

This program will simulate the rolling of 3 dice . If all three dice match the program will print a picture to show that the user has won.

### Creating the Program

1. Click the command button control in the toolbox, and then place the mouse pointer on the form. Position the top left and bottom right corner of the button.
2. Use the same procedure to add another control button, 4 labels and 1 image. The completed form should look the diagram below.



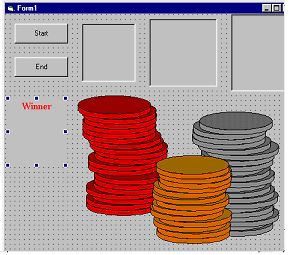
|  |  |  |
| --- | --- | --- |
| OBJECT | PROPERTY | SETTING |
| Command 1 | Caption | Start |
| Command 2 | Caption | End |
| Labels 1,2,3 | Border style | 1-fixed single |
| Alignment | 2-Center |
| Font | Symbol,bold, 72 –point |
| Caption | (none) |
| Label 4 | Caption | Winner |
| Font | Bold,24-point |
| ForeColor | red |
| Image1 | Visible | False |
| Stretch | True |
| Picture | \* |
| \*To load a picture go C:\Program Files\Microsoft Office\CLIPART\PUB60COR\AG00126\_.GIF (or your favourite picture) | | |

### Changing the Properties

Change the properties of the following objects to the indicated settings.

When you are finished the form will look similar to the figure below (you will have a different picture) to the following when all the properties have been changed.

### Entering the Code



1. Double-click the End command button.

Inside the code window you will see : Private Sub Command2\_Click()

End Sub

Type END

1. Double click the Command1 button.

Type the following lines of code :

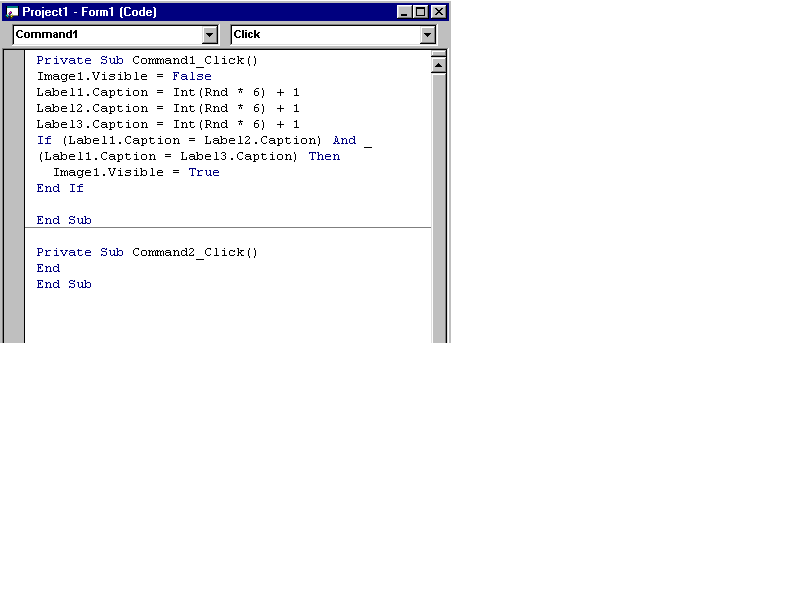


Image1.visible = false

Label1.Caption = int(rnd\*6)+1

Label2.Caption = int(rnd\*6)+1

Label3.Caption = int(rnd\*6)+1

If (Label1.Caption = Label2.Caption) And

(Label1.Caption = Label3.Caption) Then

Image1.Visible =True

End if

### Running The Program

You can now run the program by clicking the start button on the toolbar.

Click the START button to start the program. Click this button as many times as you want to continue running the program. When you want to stop, click the STOP button.

(1) B) Write a program that will make the computer say hello to you when you press a button. You will need 2 command buttons ( SAY HELLO and END) and one text box. In the code for SAYHELLO make text1.text = “Hello”. Your form will look like the following :

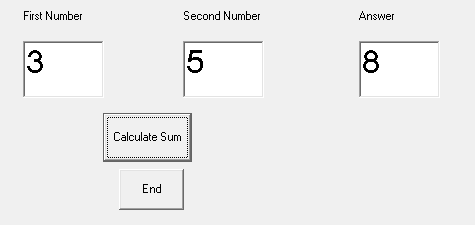


Now add two more command buttons to say hello in

Two more languages Ex. French : Bonjour

Spanish : Hola

(1) C) Write a program that allows the user to enter two numbers and have the computer print the sum. You will need 2 command buttons (SUM and END) and 3 textboxes.



Your form will look like the following :

You can make your program look better if you add labels

to indicate what information is to go into the text boxes.

You can also change the locked property of the 3rd

textbox to true so that the user can not change the answer.

(1) D) Modify your previous program, so that the user can add, subtract, multiply or divide the two numbers.

(1)E)Write a program that will allow the user to input 5 numbers and calculate the mean.

Test your program with the following :

a) 1,2,3,4,5 mean = 3 b) 8,10,11,12,13 mean = 10.8

(1)F.Modify your program to input 5 numbers and calculate the standard deviation. Display the original numbers, the difference from the mean, the difference squared and the final answer in a well organized chart.

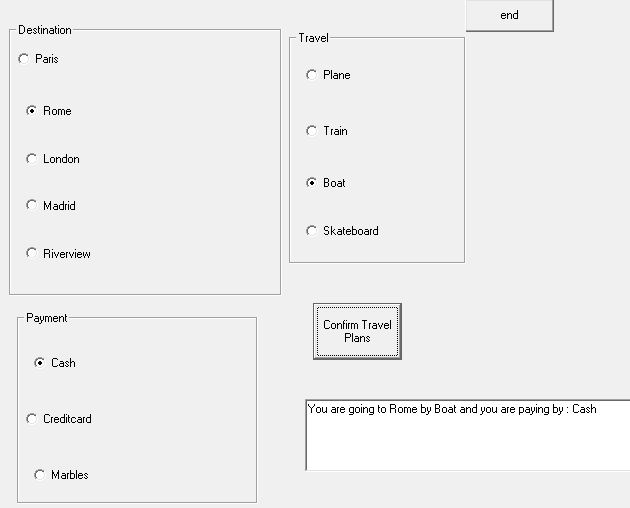
Test your program with the following : 1,2,3,4,5 standard deviation 1.414

Your program should look like the following.



## **Assignment #7**

(1) A) Write a program using option buttons to allow the user to select between 5 different destinations, 4 different modes of transportation and 3 methods of payment. When the user is finished making selections, have the information displayed in a well organized fashion.



Ex.

B)Write a program using check boxes to allow the user to check graduation requirements. When the user has finished checking the boxes the computer should display a list of graduation requirements not yet completed.

(3) C) Create your own program using option buttons and check boxes.

Example : Create a program to design a custom character for a computer game

Create a program for a dating service.

Create a program to design the house of your dreams.

Your program will be marked according to organization, level of difficulty and number of options.

Assignment #5

A.(1)

Write a program to draw a rectangle, triangle, circle and chord. Each object must be painted in a different colour.

B.(1)

Write a program to construct a curved line consisting of at least 6 points. Use the PolyBezier() function.

C.(1)

Write a program to draw Serpinski’sTriangle. See posted sheet for notes on Serpinski’s triangle.

D.(1)

Write a program that will allow the user to input the co-ordinates of two points (x1,y1) and (x2,y2). The program will construct the line that joins the two points. Draw the line on a clearly marked graph (i.e. show the x-axis, y-axis and the scale)

E.(1)

Write a program that will allow the user to specify the center and radius of a circle. The program will draw the circle on a clearly marked graph.

Assignment #6

A.(3)

Write a program that will allow the user to see a variety of different graphs. The program should be able to display the graphs y = x, y=x2 , y = x3 , y =1/x , y = √x and y = ⏐x⏐ .

B.(1)

Modify your program to also display the graphs y = sinx , y = cosx and y = tanx.

Project #1

(10)

Write a program to do something useful with graphics. The program must be related to some school course. Your program will be marked according to organization, level of difficulty and number of options. Challenge yourself.

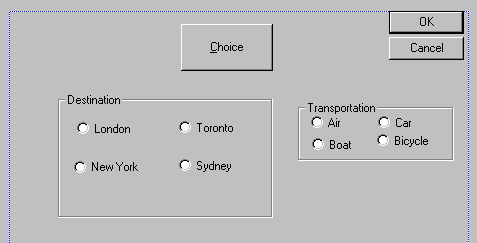
Ex. Write a program that will accept two equations, draw the two lines and calculate the points of intersections ( hint : read about matrices).

Ex. Write a program to determine the area under a curved line. See posted sheet for notes.

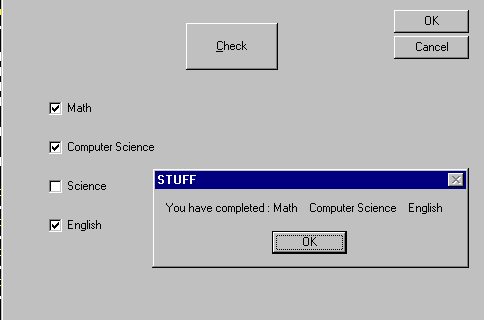
Ex. Write a program that will place a number of small circles on a graph. The user inputs an equation to hit as many of the points as possible. Points are given according to the number of circles hit.

Ex. Write a program to trace the path of a moving object. Calculate maximum height, distance,…

Check the **Group** option for the first radio button in a group



Ex. Create a dialog box to check graduation requirements. Check the courses completed and the press Check to see the message.



Note : To maximize the size of the program screen when running the program change the property WINDOWSTATE on the form to 2.

(1) C) Write a program that will say “Hello” when the user presses a button. You will need 2 command buttons and 1 textbox.

## **Visual Basic Assignment #2**

(1) A) Draw a picture in paintbrush. Go to programs and then accessories. Write a program that will display the picture when the user presses a button.

(1) B) Write a program that will move an object across a background. You can draw the

objects yourself or you can use already existing drawings in Clipart.

(5) C) Use multiple forms to tell a story. On each form have a picture and/or some motion. There should be some words in a textbox to tell the story. When the user presses the button to continue the story load the new form.

Note : To add a form click the ADD FORM command on the PROJECT menu

To load a new form use : LOAD Newformname Ex. Load form2

Newformname.SHOW Form2.show

UNLOAD oldformname Unload form1

## **Visual Basic Assignment #3**

1. A) Write a program that will display 2 random numbers. Give the user 4 mathematical options (add,subtract,multiply and divide) to choose from. Print the result of the calculation. You must use option buttons and you must dimension the variables that will be used in more than one function.

## **Visual Basic Assignment #4**

(4) A) Write a program that uses timers to create animation. There must be at least 4 stages of animation. Be creative.

## **Visual Basic Assignment #5**

(1) A) Write a program that moves an object using arrow keys.

(1) B) Add to the previous program a randomly moving object. Use the distance formula to detect

when your object collides with the randomly moving object.

## **Visual Basic Assignment #6**

(3) A) Write a program that will create a calculator on the screen. The user will click on the numbers and the mathematical operations (+, -, x, /) to have the calculator evaluate simple math problems. You must dimension the variables that will be used in more than one function.

Test your program with one digit numbers. Ex. 5 x 9 = 45

(1) B) Modify your calculator so that it works with more than one digit numbers.

Helpful hints :

* Dimension the following variables :

|  |  |
| --- | --- |
| Variable | What it is used for |
| X | Store the value of the first number |
| Y | Store the value of the second number |
| Number | Record which number you are entering, the first or the second |
| Operation | Store the mathematical operation +., -, \* , ÷ |

* Create the following functions

|  |  |
| --- | --- |
| Function | Actions performed |
| 9 button | Store 9 in x or y |
| \* button | Store \* in operation  Change the value of number |
| = button | Determine the answer based on x, y and operation |
| CLEAR button | Reset x,y, number and operation |

## **Visual Basic Major Project**

The final major project will be marked using the same marking scheme as the QBASIC project.

Write a program that creates something useful. It would be wonderful if you could design something that would have practical applications in another class or real life.

Ex. Create a snowball fighting game. Players would appear on the screen and you would try to hit them with a snowball ( the mouse) . The game ends when you have been hit too many times.

Ex. Design a board game that allows players to follow a specific route. You decide what happens when the players land on certain squares.

Ex. Create battleships , wheel of fortune, hangman, swat the gopher,….

Ex. Mr. Glennan would like someone to design a program that would allow students to learn about different styles of painting. The student could choose which style to explore and the program would show examples of these styles as well as a history of the artist.

Ex. Mrs. Bartlett would like someone to model a program on the computer so that a student could simulate a Chemistry experiment. The student would press the buttons to take measurements or mix chemicals and the results would be shown on the screen.

Ex. Ms. Duffy would like someone to design a program that would allow students to explore the graphs of various functions. The student could change the equation and the computer would display the change in the graph

Ex. Mr. Fogarty would like someone to design a program that displays a Physics formula. The student could change the values of the variables and see the effect on the answer. The program would include a visual representation of what was actually taking place with the formula, i.e. a ball was taking longer to hit the ground because the speed it was thrown at was larger.