Programming with Microsoft[©] Small Basic for Key Stage 3 - Part 1

This is an introduction to programming in a textual language. It is designed to make programming easy, approachable and fun for beginners. It uses the Text Window to introduce the basic concepts such as input, output and selection. The Graphics Window is also explored to produce a simple calculator where students can combine all their knowledge into developing a real working application.

Learning Outcomes

Write and run programs in Small Basic using:

- Sequence
 - o Input & output
- Selection use selection statements to make decisions
 - o Use If...Then
 - o Use If...Then...Else
- Iteration use iteration to repeat a series of instructions a number of times
 - o use For...EndFor loops
 - o use While...EndWhile loop
- Use variables effectively
 - Datatypes
 - One-dimensional array (Part 2)
- Find and correct syntax errors in a program
- Use comparison operators =, <, >, >=, <=, <>
- Use logical operators AND and OR
- Use procedures/sub-routines to break up tasks into subtasks (Part 2)



Vocabulary /keywords /terms Syntax Assignment statement,

Programming comparison operator

environment Logical operator

Intellisense Selection

Variable Loop

String Increment

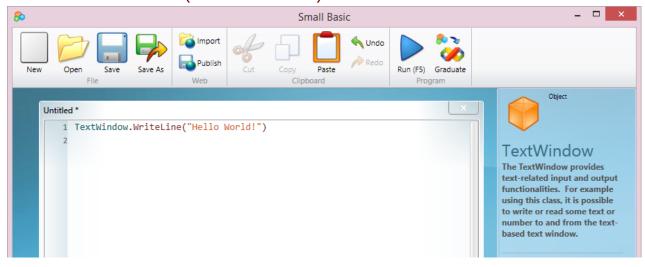
Concatenation Subroutine

Function Text window (console)

Arithmetic operator Graphics window

Getting Started - Hello World!

Writing your first program: - in the program window type the following TextWindow.WriteLine("Hello World!")



The "Hello World" program is the most traditional program that is written by programmers when starting out. — Welcome to the tradition.

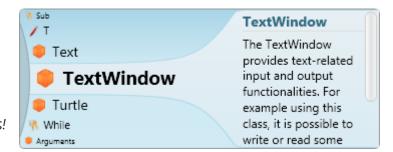


Typing code

As you typed your first program, you might have noticed that a popup appeared with a list of items. This is called "intellisense" and it helps you type your program faster. You can traverse that list by pressing the Up/Down arrow keys, and when you find something you want, you can hit the Enter key to insert the selected item in your

program.

Try and get used to using Intellisense, it will help you write your code out faster and with fewer typing errors!



Writing Small Basic Statements & Expressions

The most common statement in Small Basic is the **assignment** statement. It consists of a variable name, followed by the assignment operator "=", followed by some type of expression. The expression to the right hand side is evaluated, then the variable on the left hand side of the assignment operator is replaced by that value of the expression.

Example:

In maths for example you would see a statement like this:

$$5 \times 4 = 20$$

However in programming, assignment statements would look like this:

Area =
$$length \times width$$

The calculation length x width is executed and the product of the calculation is assigned to the variable Area.

Similarly, when using programming statements such as TextWindow.Read() we would assign the input to a variable:

name = TextWindow.Read()
quess = TextWindow.ReadNumber()



Variables

A variable is used to store different kinds of information, such as text or a number. It can contain different values at different points in time. A variable stored in the computer's memory, we can access the content of the variable when we need to.

Rules for variables

- You should always start variable names with a letter.
- You can use letters, digits, and underscores in the names of your variables.
- You should name your variables so that they describe the values that they store.
- When you name your variables, you should not include certain reserved words, such as If, For, and Then.

Using variables

Let's write a more personal greeting for our program:



In this example, we are asking for the user to type in their name, the word name is the variable and the command TextWindow.Read() reads what the user enters on the screen. So when the user enters their name, it is stored into the variable name.

The next step is to output a simple message greeting the user. We do this by joining the variable to a text string. This is called **concatenation**. (Line 4 in the example above).

TextWindow.WriteLine("Good morning, "+ name) - ^signifies a space (spacebar)



```
H:\Small Basic\Conversation.exe

Please enter your name: Fred
Good morning, Fred
Press any key to continue...
```

When you run the program, it will prompt you to enter your name (in this case Fred) and then output the message "Good morning, Fred".

Strings

Strings are just a set of characters; they can be a mixture of numbers, letters and other characters. For example "Hello World!" is a string and so is "123 Random Road".

We can use the + (addition/plus) symbol to join strings together and also to join to a variable, for example:

```
₿
                                                                  Small Basic
                                                                     Undo Undo
                                                                     Redo
                                     Publish
  New
                  Save
                         Save As
                                                                              Run (F5)
              File
                                                         Clipboard
                                                                                  Progran
Untitled *
      TextWindow.Write("Hello, what is your name: ")
      name = TextWindow.Read()
      TextWindow.WriteLine("Hello, " + name)
    6 TextWindow.Write("What's your favourite food? ")
    7 favFood = TextWindow.Read()
    9 TextWindow.WriteLine("Hello, " + name + " I like " + favFood + " too!")
```

Note the spaces before the closing and opening brackets — this makes it easier to read when the program is run.

```
C:\Users\wmparry\AppData\Local\Temp\tmp8229.tmp.exe

Hello, what is your name: Fred
Hello, Fred
What's your favourite food? Sushi
Hello, Fred I like Sushi too!
Press any key to continue...
```

Running this program, you are prompted to enter your name and your favorite food. The program will then output a message combining both your name and your favourite food.



Datatypes

As you know, we use variables to store information. However, variables store particular types of data. In Small Basic, there are 3 main types of data: numeric, string(text) and Boolean. You should always be aware of the type of data stored in a particular variable.

| Numeric | Integer or decimal | 22 (integer) |
|---------|------------------------------|------------------------|
| | numbers +/- numbers | 35.6(decimal) |
| | | -45 (negative integer) |
| String | A string is just a set of | "abcde" |
| | various characters. It could | "The quick brown fox" |
| | be a word, a sentence, a | "78 Woodbridge Avenue" |
| | paragraph or it could be | "12345-8976" |
| | no characters (null string). | " " – null string |
| Boolean | Can be of either 2 values | True/False |
| | Good for making decisions | |

There are other types of variables in other programming languages, however, Small Basic is happy with the main 3. Later on you will look at Arrays — an array allows you to store lists of values. It's pretty cool.

We do need to be careful not to mix our variable data types. We can only do mathematical operations on numbers. Strings can only work with other strings and Boolean types are really to help us make a decision.



Debugging your program

As you begin to program you will make syntax errors, that's pretty normal.

Syntax in programming languages is similar to spelling, punctuation and grammar in English or French, basically any language has these properties.

Don't be put off by these errors – part of being a programmer is about solving your own programming dilemmas.

The most common errors you will encounter at the beginning with Small Basic are easily fixed.

Common error #1

Omitting the closing bracket in a TextWindow.WriteLine() command.

TextWindow.Write("What's your favourite food? "

Should be:

TextWindow.Write("What's your favourite food? ")

Common error #2

Omitting the closing double quotes (") in a TextWindow.WriteLine() command

TextWindow.Write("What's your favourite food?)

Should be:

TextWindow.Write("What's your favourite food? ")

Common error #3

```
Using an = (equals sign) instead of a + (plus sign).
```

```
TextWindow.WriteLine("Hello, " + name = " I like " + favFood = " too!")
```

Should be:

```
TextWindow.WriteLine("Hello, " + name + " I like " + favFood + " too!")
```

Common error #4

A combination of all of the above.

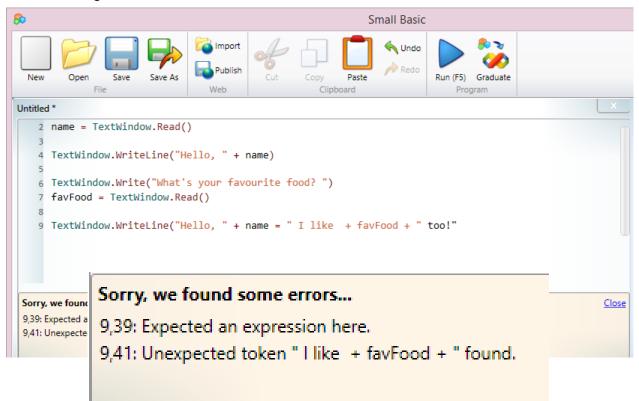
```
TextWindow.WriteLine("Hello, " + name = " I like + favFood + " too!"
Should be:
```

```
TextWindow.WriteLine("Hello, " + name + " I like " + favFood + " too!")
```



Let Small Basic help you debug your program

When you encounter an error when running your code — don't panic! Look at the error message below the code window.



The number at the beginning of the error statement tells you what **line number** the error occurs on, the second number, relates to where on the line the error occurs — in the case of the first error, the problem is on line 9, and the 39^{th} character along.

You can also dbl+click on the error line in the message pane and your cursor will be automatically located to where the problem occurs in your code.

It is good practice to try and debug your own code — it's much more exciting when you fix the problem and your code works!



Activity: Having a conversation with a computer

Stuck for someone to chat to, friends all busy? Write your own program that allows you to have a simple conversation with a computer. For example:

Computer: Hello there, how are you, my name is iBot, what's yours?

You: Fred

Computer: Well, hello Fred, how are you today?

You: Happy

Computer: That's great Fred, I'm glad you are feeling happy today

Computer: What's your favourite colour?

You: Pink

Computer Cool, I like Pink. Pink is my favourite colour too.

Save your program as "computer chat' and run it (F5)

How did it go?

- DID YOUR PROGRAM WORK FIRST TIME?
- WHAT SORT OF ERRORS DID YOU HAVE?
- HOW DID YOU FIX THEM?

If you made a real good job or your conversation, you might want to read about the Turing Test here.

Link: http://news.bbc.co.uk/1/hi/magazine/7670050.stm



Making decisions

So far, our programs have just been executing the instructions in the order that we have written them. This is known as **sequence** in the programming world. Using just sequence doesn't really make our programs very flexible. To allow us to create more complex programs we need to use programming statements that enable us to make decisions. This is called **selection**. This is like a **decision** in a **flow chart**.

Selection

Selection allows our program to carry out different instructions based on the outcome of a question. For example, let's say you want to go to the cinema, however you also want to go to the beach, how do we make our choice? Well, maybe what we choose is based upon the weather forecast.

If it's sunny then let's go to the beach, else, let's go to the cinema.

The statement above can be interpreted as:

If weather = sunny then 'this is the question (or condition)

Go to the beach 'if the sun is out, then it's true, beach

time!

Else 'oh, this means it's raining (sunny = false)

Go the cinema 'so let's go watch a movie instead

Selection Statements:

The basic syntax of a selection statement takes the form of:

IF < condition > THEN

Do the thing that is true here

ENDIF

We use this basic statement if we don't have an alternative option — in this case the code continues to execute after the ENDIF in a sequential order (sequence).

A simple video explanation...watch out for the Orange!



However, it is nice to have a choice, so we can extend our options by using the ELSE statement.

IF <condition> THEN

Do the thing that is true here

ELSE

Do the thing that is false here

ENDIF

Lastly, if we have more than one decision to make, then we can test multiple conditions by using the **Elself** statement.

IF <condition1> THEN

Do this thing if condition 1 is true

ELSEIF < condition 2> THEN

Do this thing if <u>condition 2</u> is true here

ELSEIF < condition 3 > THEN

Do this thing if condition 3 is true

ENDIF

Note: Each IF and ELSEIF in the block is tested sequentially. If the first condition is met then the code within that condition is executed and the IF block is terminated.

Example: High/Low game

IF guess = number THEN

"Guess was correct"

ELSEIF guess > number THEN

"Guess too high"

ELSEIF guess < number THEN

"Guess too low"

ENDIF



Comparison & Logical Operators

It is useful when trying to make decisions to compare values against other values or check whether text entered is what was expected. You should be familiar with these concepts from your maths lessons. There are six comparison operators available to use:

| Operator | Comparison |
|-----------------|--------------------------|
| > | Greater than |
| < | Less than |
| >= | Greater than or equal to |
| <= | Less than or equal to |
| = | Equal to |
| <> | Not equal to |

The result of a comparison operation is a Boolean value – **true** or **false**! E.g. 10 > 5 – evaluates as true whereas "dog" = "cat" is false.

Logical operators again operate on Boolean data types, providing a Boolean result (true/false). There are two operators available to use in Small Basic.

| Operator | Operation | |
|----------|-------------|--|
| AND | Logical AND | |
| OR | Logical OR | |

The AND operator checks to see if two different data types are both TRUE. If both are TRUE then the operator returns a TRUE – if only one data type is true, then it returns FALSE.

Example: Login System

ELSE

"Username or password is incorrect"

ENDIF



Activity: Let's make a simple quiz game

Have you played 2 truths and a lie? You need to think of 3 facts about yourself, 2 of them are true and 1 is a lie. You then get people to guess which the lie.

Write a program where the computer pretends to be you. In your program you should:

- 1. Introduces yourself to your player
- 2. Asks for the player's name
- 3. Tells the player the game rules
- 4. Displays the 3 facts
- 5. Asks the user to select the correct answer
- 6. Checks the answer and tells the user whether they chose the right answer

Plan your solution

- How many variables will you need to store data?
 - Hint: you will need to store the 3 facts, the answer and the guess and maybe one for the player's name.
- How will you check the player's guess is correct?

Hint: you may need to use a selection statement

PseudoCode: Write out your pseudo code first - understand the logic!

You may have something like this

- 1. fact1 = "Write your fact here"
- 2. fact2 = "Write your fact here"
- 3. fact3 = "Write your fact here"
- 4. lie = Fact3 (expressed as a number i.e. 1, 2 or 3)
- 5. output "Hello, I am [name] welcome to my quiz game, enter your name:"
- 6. name = Input name
- 7. output "Hello [name], I am going to tell you some facts about me, 2 of them are true, 1 of them is a lie can you guess which one is a lie?
- 8. output Fact1
- 9. output Fact2
- 10. output Fact3
- 11. output "Enter your answer"
- 12. guess = input



- 13. IF guess = lie THEN
 - a. Output "Well done, you guessed right"
- 14. ELSE
- b. Output "Uh oh, not right, try again
- 15. ENDIF

Small Basic commands - some of the commands you may want to use

- **●** Textwindow.Write()
- TextWindow.WriteLine()
- TextWindow.Read()
- TextWindow.ReadNumber()
- TextWindow.WriteLine("") Add space between lines:
- TextWindow.PauseWithoutMessage() Pause for each statement to be displayed:



Solution: the code

```
Small Basic
                                                                 Undo
             File
                                                     Clipboard
Untitled *
    1 TextWindow.Title ="My quizzy game"
    3 fact1 = "1. I can eat 20 pancakes for breakfast "
    4 fact2 = "2. I am a circus juggler'
   5 fact3 = "3. I am an only child"
   6 \text{ lie} = 3
   8 TextWindow.Write("Hello, my name is Fred, what's your name?")
   9 name = TextWindow.Read()
   10 TextWindow.WriteLine("Cool, let's play a game " + name )
   11 TextWindow.WriteLine("Here are some facts, 2 are true 1 is a lie, can you guess which?")
   12 TextWindow.WriteLine(fact1)
   13 TextWindow.WriteLine(fact2)
   14 TextWindow.WriteLine(fact3)
   16 TextWindow.Write("Enter your guess - is it 1, 2, or 3?")
   17 guess=TextWindow.Read()
   18
   19 If guess = lie Then
       TextWindow.WriteLine("Yay, you got it right")
       TextWindow.WriteLine("Boo, you got it wrong")
   23 EndIf
```

How did it ao?

- Did your program work first time?
- What sort of errors did you have?
- How did you fix them?
- What can you to improve the interactivity of the game?

Extensions:

1. If the user guesses the incorrect fact the first time, the game ends and you have to re-run the program. How can you make the game play again until the user chooses not to?

GOTO - The Goto statement allows branching to a new location in the program. Use a label to set the location to go to.

TextWindow.WriteLine(i)

i = i + 1

Goto start

PROGRAM.END() - Ends the

program.

Hint: You will need to ask the player if they wish to play again, accept a response and depending on their response either start the game again, or end the game.

How did it go?

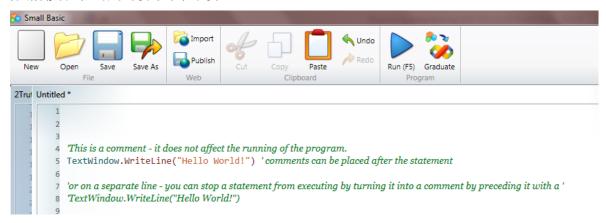
- ☆ DID YOU GET THE GAME TO ALLOW THE PLAYER TO PLAY AGAIN?
- ☆ DID YOU HAVE ANY OTHER PROBLEMS IF YOU GUESSED

 CORRECTLY, DID THE GAME BEHAVE IN THE WAY YOU ANTICIPATED

 IT SHOULD?
- ★ WHAT ELSE DID YOU NEED TO ADD TO MAKE THE GAME PLAY
 CORRECTLY?

Making your code more readable and less forgettable:

It is good practice to make comments in your coding. All major programming allow you to do this in some way. It provides a record of what the statements are doing and allows the programmer to explain how the code was written. To make comments in Small Basic you precede your comment with the '(single apostrophe) character. Comments are recognizable as they are green and italicized in the code window.





The final solution:

```
TextWindow.Title ="My quizzy game"
'set up the facts - facts 1-3 are variables
fact1 = "1. I can eat 20 pancakes for breakfast "
fact2 = "2. I am a circus juggler"
fact3 = "3. I am an only child"
lie = 3 ' set the lie in a variable
'introduce the game
TextWindow.Write("Hello, my name is Fred, what's your name?")
name = TextWindow.Read()
TextWindow.WriteLine("Cool, let's play a game " + name )
'present the facts to the player
restart: ' point at which to start the game if you want to play
again
TextWindow.WriteLine("Here are some facts, 2 are true 1 is a lie,
can you guess which?")
TextWindow.WriteLine(fact1)
TextWindow.WriteLine(fact2)
TextWindow.WriteLine(fact3)
TextWindow.Write("Enter your guess - is it 1, 2, or 3?")
'read the player's input and store into a variable named guess
guess=TextWindow.ReadNumber()
'check the player's guess against the actual lie
If guess = lie Then
  TextWindow.WriteLine("Yay, you got it right")
 TextWindow.PauseWithoutMessage()
 Program.End()
Else
  TextWindow.WriteLine("Boo, you got it wrong")
 TextWindow.Write("Would you like to play again y/n?: ")
 playagain = TextWindow.Read() ' accept the player input or y (yes)
or n (no)
EndIf
If playagain = "y" Then
 Goto restart
  Program.End()
EndIf
```



Repetition, iteration, looping

Objectives:

- * To use a For..EndFor loop to repeat a series of instructions several times
- To use a While...EndWhile condition loop to repeat a series of instructions

As we've seen, in the previous section, selection (conditionals) allow you to run a piece of code once if a condition is true. Loops on the other hand, allow you to run a piece of code multiple times, depending on whether a condition remains true.

For example: while there's food on your plate, keep eating or in an 800m race, run around the track 2 times (to get to the finish line).

For...EndFor loop

Generally a For...EndFor loop is used to run code a specific number of times. To manage this kind of loop, you create a variable that tracks how many times the loop has run. Generally this is called a **Count** Control.

The basic syntax of a For...EndFor statement takes the form of:

For counter = startNumber To endNumber

Repeat these commands (counter)

EndFor

The **counter** is a variable that keeps count of the number of times the instructions inside the loop have been executed. It can be called anything, however, the usual name is just a simple "i".

The **startNumber** is starting number of the loop and the **endNumber** is the total number of times the loop must run. The following example numbers lines in a text window:

For I = 1 to 10

TextWindow.WriteLine(i)

EndFor

Running the program outputs the following:



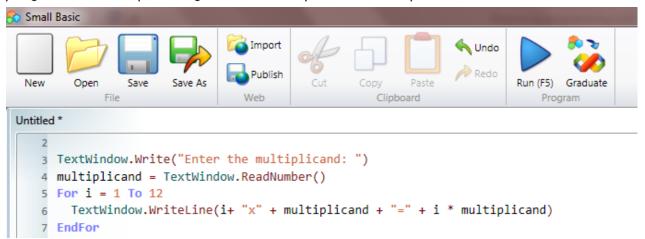
```
Looping in Small Basic

1
2
3
4
5
6
7
8
9
10
Press any key to continue...
```

In this example, the variable (i) contains a value that increases by 1 each time the loop runs — in this case, we asked the program to run 10 times.

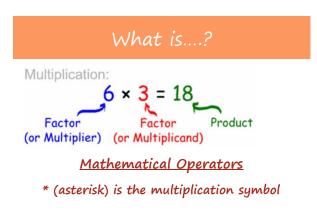
Multiplication tables

We can use this concept to print out multiplication tables. Remember when you had to chant your times tables $1 \times 4 = 4$, $2 \times 4 = 8$, $3 \times 4 = 12$? Let's write a program that outputs any times table up to the multiplier of 12.



Running the program displays the following:

```
Enter the multiplicand: 4
1×4=4
2×4=8
3×4=12
4×4=16
5×4=20
6×4=24
7×4=28
8×4=32
9×4=36
10×4=40
11×4=44
12×4=48
Press any key to continue...
```



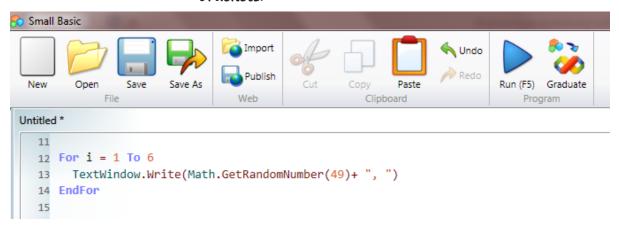


Activity

Lucky Lotto

Write a program that outputs 6 random numbers between 1 and 49. You will need to use a new function.

Math.GetRandomNumber(49) — the number in the brackets tells Small Basic to generate a number between 1 and up to whatever you put in the brackets.



Running the program outputs:

```
Looping in Small Basic

45, 34, 43, 48, 1, 11,

Press any key to continue...
```

While...EndWhile loop

A while loop repeatedly executes the commands until a particular condition stops being true. Basically you are saying: "keep doing this while this condition is true, stop when the condition becomes false". For example, keep washing your face while it is still dirty.

The While...EndWhile loop is useful when the loop count is not known ahead of time. Whereas the For...EndFor loop runs for a pre-determined number of times.

Generally this is known as a Condition Control loop.

Still not sure? Watch this short animation...check out that pineapple!



The basic syntax of a While...EndWhile statement takes the form of:

While condition (is not true)

Repeat these commands

EndWhile

Taking our previous multiplication example, using the While...EndWhile loop:

```
Small Basic
                                                                         Undo U
                                      Publish
New
        Open
                 Save
                         Save As
                                                                                   Run (F5)
  Looping.sb * - C:\Users\wparry\Google Drive\SmallBasic Programs\Looping.sb
         multiplicand = 5
       9 i= 1
      10 While i < 13
            TextWindow.WriteLine(i + "x" + multiplicand + "=" + i*multiplicand)
      11
      12
      13 EndWhile
```

Running the program (F5) outputs:

```
1x5=5
2x5=10
3x5=15
4x5=20
5x5=25
6x5=30
7x5=35
8x5=40
9x5=45
10x5=50
11x5=55
12x5=60
Press any key to continue...
```

Explanation

Firstly we need to set up the condition:

Line 9: sets the value of i to 1

Line 10: the condition of i<13 is the condition

Line 11: Output the times table

Line 12: Increment the value of i by 1
As the loop executes, it checks to see
whether the value of i is less than 13, if
not (while), then it runs the commands
in the loop again. We then set the value
of i to increase by 1 – the code runs
until i becomes 13 and then it stops.
(because 13 is not less than 13).



Activity

Write a program that continually divides a number by 2 until the value is less than 1.

How do I divide?

Mathematical Operators

/ (forward slash) is the division symbol

Possible solution

number = 100

While (number >1)

TextWindow.WriteLine(number)

number = number / 2

EndWhile

Activity

Write a program to implement the function of logging in to a computer. You should set a password and username. Your program should accept the login only if the username AND password are correct.

If either are incorrect a suitable error message should be displayed and a prompt displayed to try and enter again.

- Extend your program to include a loop (repetition), to improve the efficiency of your program.
- Extend your program to offer the option once you've successfully logged in to change your password.



Challenges:

1. Guess the number game - Easy

The computer selects a random number between 1 and 100. The player keeps guessing which number the computer has chosen until they get it right. The computer responds "got it" or "too high" or "too low" after each guess. After the user has guessed the number the computer tells them how many attempts they have made.

2. Create a "higher/lower" guessing game. - Intermediate/Hard

The computer randomly generates a sequence of up to 10 numbers between 1 and 13. The player, after seeing each number in turn has to decide whether the next number is higher or lower. If you get 10 guesses right, you win the game.

Starting number: 12

Higher (H) or Lower(L)? L

Next number: 8

Higher (H) or Lower(L)? L

Next number: 11

You lose!

3. Rock, paper scissors - Intermediate

Make a game of rock, paper scissors against the computer

Algorithm

- 1. Player: Input rock(1), paper(2), scissors(3)
- 2. Computer: generate a random number between 1 & 3
- 3. Compare the player's input with the computer selection
- 4. Display who wins

Use: best of 3 rules & track score

Make sure you display the descriptions e.g. rock Vs scissors (rock wins) etc.

You need to work out the combinations of wins.

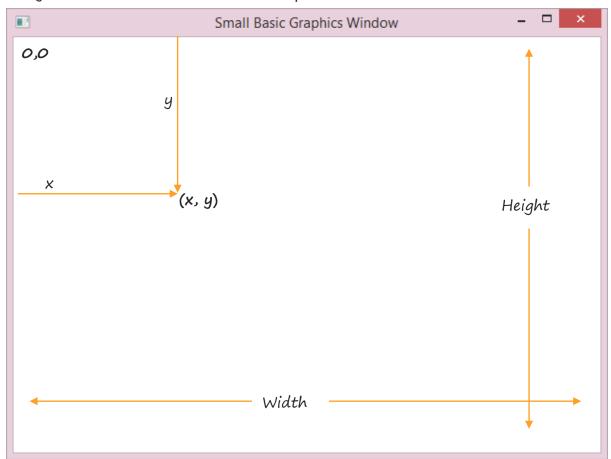


Introducing the Graphics Window

So far we have just been using the TextWindow object; this is a great way of starting out using a textual programming language — it's easy to see the results and you're not so concerned with the visual design aspect. However, not many "apps" are written using the TextWindow — let's face it, they're not very exciting are they?

The GraphicsWindow object is the key to project building in Small Basic. In the GraphicsWindow we can draw lines, shapes and text in many colours. We can interact with the interface by receiving events (mouse clicks, keyboard input) from the user.

The GraphicsWindow has two properties of width and height and we use coordinates, in pixels, to draw objects on to the GraphicsWindow, X(horizontal) axis and Y (vertical) axis. The x axis coordinate increases from left to right and the y axis coordinate increases from top to bottom.



Points in the region are referred to by the two coordinates enclosed in brackets.

Remember to calculate your coordinates from the top left hand corner!



GraphicsWindow Properties

BackgroundColor

BrushColor

FontBold

FontSize

Height

LastKey

LastText

MouseX

MouseY

PenColor

PenWidth

Title

Width

GraphicsWindow Methods

Clear()

DrawEllipse(x,y,w,h)

Drawlmage(image, x,y)

DrawLine(x1, y1, x2, y2)

DrawRectangle(x,y,w,h)

DrawResizedImage(image, x,y,w,h)

DrawText(x,y, text)

DrawTriangle(x1, y1, x2, y2,x3,y3)

FillEllipse(x,y,h.w)

FillRectangle(x,y,h.w)

FillTriangle(x1, y1, x2, y2,x3,y3)

GetColorFromRGB(red, green, blue)

GetRandomColour()

Hide()

Show()

ShowMessage(text, title)

GraphicsWindow Events:

KeyDown

KeyUp

MouseDown

MouseMove

MouseUp

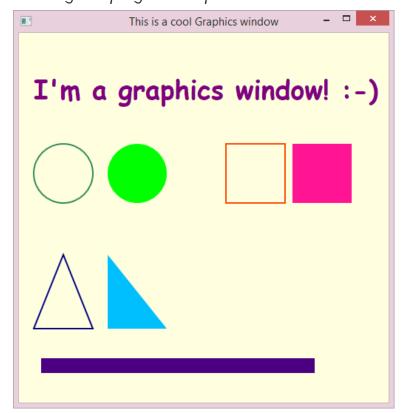
TextInput



Let's try out the GraphicsWindow

```
Untitled *
   1 GraphicsWindow.Title = "This is a cool Graphics window"
   2 GraphicsWindow.BackgroundColor = "LightYellow"
   3 GraphicsWindow.Width = 500
   4 GraphicsWindow.Height = 500
   5 GraphicsWindow.FontSize = 36
   6 GraphicsWindow.FontName = "Comic Sans MS"
   7 GraphicsWindow.BrushColor = "Purple"
   8 GraphicsWindow.DrawText(20, 50, "I'm a graphics window! :-)")
  10 'drawing shapes
  11 GraphicsWindow.PenColor = "SeaGreen"
  12 GraphicsWindow.BrushColor = "Lime"
  13 GraphicsWindow.DrawEllipse(20, 150, 80,80)
  14 GraphicsWindow.FillEllipse(120, 150, 80, 80)
  16 GraphicsWindow.PenColor = "OrangeRed"
  17 GraphicsWindow.BrushColor = "DeepPink"
  18 GraphicsWindow.DrawRectangle(280, 150, 80,80)
  19 GraphicsWindow.FillRectangle(370,150,80,80)
  21 GraphicsWindow.PenColor = "Navy"
  22 GraphicsWindow.BrushColor = "DeepSkyBlue"
  23 GraphicsWindow.DrawTriangle(20, 400,100, 400, 60,300)
  24 GraphicsWindow.FillTriangle(120, 400, 200, 400, 120, 300)
  26 GraphicsWindow.PenColor = "Indigo"
  27 GraphicsWindow.PenWidth = 20
  28 GraphicsWindow.DrawLine(30, 450, 400, 450)
```

Running the program outputs:



Play around with the drawing tools

Try drawing a series of shapes and patterns — look at the colour reference sheet and check out the groovy colours you can use.

Activity:

Draw a stick man!



Controls & Interactivity with the GraphicsWindow

Drawing is pretty cool, however, to create functional or useful applications we need to interact with the interface. We need to use controls such as buttons and text boxes to provide the user with a way to select options or to input text and numbers.

Controls object

This allows you to add 3 controls to the Small Basic program in the GraphicWindow. These controls are:

- © Button
- TextBox
- MultiLineTextBox

You add these controls by using the Controls methods:

- AddButton(caption, x,y)
- AddMultiLineTextBox(x,y)
- AddTextBox(x,y)

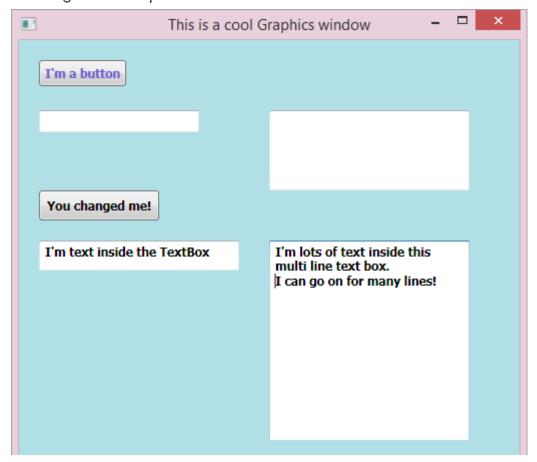
Let's add some controls

Adding controls to the GraphicsWindow is pretty simple:

```
GraphicsWindowControls.sb - E:\SmallBasic Programs\GraphicsWindowControls.sb
    1 GraphicsWindow.Title = "This is a cool Graphics window"
    2 GraphicsWindow.BackgroundColor = "PowderBlue"
    3 GraphicsWindow.Width = 500
     GraphicsWindow.Height = 500
    6 Controls.AddButton("I'm a button", 20,20)
    7 Controls.AddTextBox(20, 70)
    8 Controls.AddMultiLineTextBox(250,70)
   10 'add new controls but give them names
   11 Button1 = Controls.AddButton("Click Me!", 20, 150)
   12 txtBox1 = Controls.AddTextBox(20, 200)
  13 MltBox1 = Controls.AddMultiLineTextBox(250, 200)
  14
   15 'add text to the Textbox and the MultiLineTextBox
  16 Controls.SetTextBoxText(txtBox1, "I'm text inside the TextBox")
17 Controls.SetTextBoxText(MltBox1, "I'm lots of text inside this multi line text box. I can go on for many lines!")
  18 Controls.SetButtonCaption(Button1, "You changed me!")
  20 'control the way the controls look
  21 Controls.SetSize(Button1, 120, 30)
  22 Controls.SetSize(txtBox1, 200,30)
   23 Controls.SetSize(MltBox1, 200, 200)
```



Running this code produces:



Controls Properties

Control properties allow us to determine which control was last used.

- LastTypedTextBox

Controls Events

The control events detects whether an action occurred on a button or a text box – if we clicked on a button for example, we would expect something to happen – this is called an event.

- ButtonClicked
- TextTyped

With some additional code covered later in this module you can do achieve some interesting results using the GraphicsWindow. In the following example, an event is raised when the "Click Me!" button is clicked. SmallBasic detects the event actioned on the button. It then changes the text on the original button, sets the text inside the text box and changes the emoji picture.







If you want to have a go, first you'll need to find 2 pictures that you can use to interchange — both pictures must be saved in the same folder as your SmallBasic project code. You will then need to change the image name at lines 10 & 30.

```
emoji.sb * - H:\Small Basic\emoji.sb
   1 GraphicsWindow.Title = "This is a cool Graphics window"
   2 GraphicsWindow.BackgroundColor = "PowderBlue"
   3 GraphicsWindow.Width = 500
   4 GraphicsWindow.Height = 500
   6 'add controls and give them names
   7 GraphicsWindow.BrushColor = "black"
   8 Button1 = Controls.AddButton("Click Me!", 20, 20)
   9 txtBox1 = Controls.AddTextBox(20, 60)
   10 GraphicsWindow.DrawResizedImage(Program.Directory + "\emoji1.png", 150,200, 300,300)
   12 'add text to the Textbox and the MultiLineTextBox
   13 Controls.SetTextBoxText(txtBox1, "I'm text inside the TextBox")
   15 'controls the way the controls look
   16 Controls.SetSize(Button1, 120, 30)
   17 Controls.SetSize(txtBox1, 200,30)
   19 'controls.buttonclicked - listens out for a button being clicked within the graphics window
   20 Controls.ButtonClicked = ButtonClicked
                                                On line 23 we pass the name of the button through to a variable
   21
   22 Sub ButtonClicked
                                                named 'b'. We can now interrogate the value of 'b; to determine
      b = Controls.LastClickedButton
   23
                                                the next action to be performed.
       If b = Button1 Then
   24
          Controls.SetSize(Button1, 200, 30)
   25
          Controls.SetButtonCaption(Button1, "Ha Ha I've changed")
          Controls.SetTextBoxText(txtBox1, "and I've changed too")
   27
          GraphicsWindow.DrawResizedImage(Program.Directory + "\emoji3.png", 150,200, 300,300)
   28
   29
   30 EndSub
```

Activity

Extend the emoji program so you can flick back to the original emoji picture.



Sub Routines (Sub Procedures)

Often we will want to execute a series of instructions in our program; rather like when we "save" our documents or code — we click on the "save" icon and a series of actions are performed. The code for the save function is written just once, but it is called each time we require it. The save commands have been written in a sub routine and can be reused throughout the execution of the program.

A subroutine is a block of code within the main program that does something specific, it can be called from anywhere in the program. In SmallBasic, the subroutine syntax takes the following form:

Sub <subroutine name>

...instructions

EndSub

In our EMOJI program we used the following subroutine:

```
Controls.ButtonClicked = ButtonClicked
```

```
Sub ButtonClicked
```

EndIf EndSub

```
b = Controls.LastClickedButton
If b = Button1 Then
   Controls.SetSize(Button1, 200, 30)
Controls.SetButtonCaption(Button1, "Ha Ha I've changed")
```

In this example, when SmallBasic hears a button being clicked, it calls the subroutine "ButtonClicked" and executes the lines of code between the SUB and ENDSUB.

After the code within the subroutine has been executed, control passes back to the main program.

GraphicsWindow.DrawResizedImage(Program.Directory + "\emoji3.png", 150,200,300,300)

Executing a subroutine from anywhere in the program

Controls.SetTextBoxText(txtBox1, "and I've changed too")

You can execute a subroutine by calling SubroutineName() – note use of the brackets.

- 1. TextWindow.Write("the time is: ")
- 2. PrintTime()
- Sub PrintTime
- 4. TextWindow.WriteLine(Clock.time)
- 5. EndSub

In the example on the left, PrintTime is the subroutine which will output the time from the computer's system clock and line 2 calls the subroutine.



Colours in Small Basic

Purple Colors

| Lavender | #E6E6FA |
|-----------------|---------|
| Thistle | #D8BFD8 |
| Plum | #DDA0DD |
| Violet | #EE82EE |
| Orchid | #DA70D6 |
| Fuchsia | #FF00FF |
| Magenta | #FF00FF |
| MediumOrchid | #BA55D3 |
| MediumPurple | #9370DB |
| BlueViolet | #8A2BE2 |
| DarkViolet | #9400D3 |
| DarkOrchid | #9932CC |
| DarkMagenta | #8B008B |
| Purple | #800080 |
| Indigo | #4B0082 |
| SlateBlue | #6A5ACD |
| DarkSlateBlue | #483D8B |
| MediumSlateBlue | #7B68EE |

Orange Colors

| LightSalmon | #FFA07A |
|-------------|---------|
| Coral | #FF7F50 |
| Tomato | #FF6347 |
| OrangeRed | #FF4500 |
| DarkOrange | #FF8C00 |
| Orange | #FFA500 |

Pink Colors

| Pink | #FFC0CB |
|-----------------|---------|
| LightPink | #FFB6C1 |
| HotPink | #FF69B4 |
| DeepPink | #FF1493 |
| MediumVioletRed | #C71585 |

Red Colors

| IndianRed | #CD5C5C |
|------------|---------|
| LightCoral | #F08080 |
| Salmon | #FA8072 |



Programming with Small Basic

| DarkSalmon | #E9967A |
|-------------|---------|
| LightSalmon | #FFA07A |
| Crimson | #DC143C |
| Red | #FF0000 |
| FireBrick | #B22222 |
| DarkRed | #8B0000 |

Yellow Colors

| Gold | #FFD700 |
|----------------------|---------|
| Yellow | #FFFF00 |
| LightYellow | #FFFFE0 |
| LemonChiffon | #FFFACD |
| LightGoldenrodYellow | #FAFAD2 |
| PapayaWhip | #FFEFD5 |
| Moccasin | #FFE4B5 |
| PeachPuff | #FFDAB9 |
| PaleGoldenrod | #EEE8AA |
| Khaki | #F0E68C |
| DarkKhaki | #BDB76B |

Green Colors

| GreenYellow | #ADFF2F |
|-------------------|---------|
| Chartreuse | #7FFF00 |
| LawnGreen | #7CFC00 |
| Lime | #00FF00 |
| LimeGreen | #32CD32 |
| PaleGreen | #98FB98 |
| LightGreen | #90EE90 |
| MediumSpringGreen | #00FA9A |
| SpringGreen | #00FF7F |
| MediumSeaGreen | #3CB371 |
| SeaGreen | #2E8B57 |
| ForestGreen | #228B22 |
| Green | #008000 |
| DarkGreen | #006400 |
| YellowGreen | #9ACD32 |
| OliveDrab | #6B8E23 |
| Olive | #808000 |
| DarkOliveGreen | #556B2F |
| MediumAquamarine | #66CDAA |
| DarkSeaGreen | #8FBC8F |



Programming with Small Basic

| LightSeaGreen | #20B2AA |
|---------------|---------|
| DarkCyan | #008B8B |
| Teal | #008080 |

Gray Colors

| Gainsboro | #DCDCDC |
|----------------|---------|
| LightGray | #D3D3D3 |
| Silver | #C0C0C0 |
| DarkGray | #A9A9A9 |
| Gray | #808080 |
| DimGray | #696969 |
| LightSlateGray | #778899 |
| SlateGray | #708090 |
| DarkSlateGray | #2F4F4F |
| Black | #000000 |

White Colors

| White | #FFFFFF |
|-------|---------|
| | |

Blue Colors

| Aqua | #00FFFF |
|-----------------|---------|
| Cyan | #00FFFF |
| LightCyan | #E0FFFF |
| PaleTurquoise | #AFEEEE |
| Aquamarine | #7FFFD4 |
| Turquoise | #40E0D0 |
| MediumTurquoise | #48D1CC |
| DarkTurquoise | #00CED1 |
| CadetBlue | #5F9EA0 |
| SteelBlue | #4682B4 |
| LightSteelBlue | #B0C4DE |
| PowderBlue | #B0E0E6 |
| LightBlue | #ADD8E6 |
| SkyBlue | #87CEEB |
| LightSkyBlue | #87CEFA |
| DeepSkyBlue | #00BFFF |
| DodgerBlue | #1E90FF |
| CornflowerBlue | #6495ED |
| MediumSlateBlue | #7B68EE |
| RoyalBlue | #4169E1 |
| Blue | #0000FF |
| MediumBlue | #0000CD |



Programming with Small Basic

| DarkBlue | #00008B |
|--------------|---------|
| Navy | #000080 |
| MidnightBlue | #191970 |

Brown Colors

| Cornsilk | #FFF8DC | | | |
|----------------|---------|--|--|--|
| BlanchedAlmond | #FFEBCD | | | |
| Bisque | #FFE4C4 | | | |
| NavajoWhite | #FFDEAD | | | |
| Wheat | #F5DEB3 | | | |
| BurlyWood | #DEB887 | | | |
| Tan | #D2B48C | | | |
| RosyBrown | #BC8F8F | | | |
| SandyBrown | #F4A460 | | | |
| Goldenrod | #DAA520 | | | |
| DarkGoldenrod | #B8860B | | | |
| Peru | #CD853F | | | |
| Chocolate | #D2691E | | | |
| SaddleBrown | #8B4513 | | | |
| Sienna | #A0522D | | | |
| Brown | #A52A2A | | | |
| Maroon | #800000 | | | |
| | | | | |

