Getting user input

Up until now we have programmed the computer to display information on the screen with the statement TextWindow.Writeline("Here is some text"). Now however, we would like the computer to respond to some input typed in by the user. We will use the Read() function of the TextWindow.

```
1 TextWindow.WriteLine("What is your favourite colour?")
2 TextWindow.Read()
```

When this program is run we will have an output similar to that below.

```
What is your favourite colour?
green
Press any key to continue...
```

The text window will display the message "What is your favourite colour?" and the computer will wait for you to type some text followed by the <Enter> key, after which the program will end.

What happens to the word "green" after we have typed it in? It has been lost, because we didn't instruct the computer to remember it. If we tell the computer to remember what we type in, then we can begin to write more interesting programs. The computer will be able to respond to our inputs.

Remembering our inputs

To instruct the computer to remember our inputs we will use the assignment operator '=', the equals sign. We will assign a label to our input so we can easily refer to it.

Small Basic's keywords and 2 operators are listed below:

Else
ElseIf
EndFor
EndIf
EndSub
EndWhile
For
Goto
If
Step
Sub
Then
To
While

and the two operators

And
Or

The label can be any word we like (except Small Basic's key words which are listed on the right), so let us choose myColour.

```
1 TextWindow.WriteLine("What is your favourite colour?")
2 myColour = TextWindow.Read()
3 TextWindow.WriteLine("I like the colour" + myColour + " too.")
```

When we run this program the computer will remember our input text by assigning it the label myColour, which it can use later in its response. What is important to note is that we can use this program over and over again; and enter a different colour each time. The computer will always respond accordingly because it will reassign each new input to the label myColour every time the program is run. Run the program three times and vary your input text each time.

```
What is your favourite colour?
red
I like the colour red too.
Press any key to continue...
```

Here the word "red" is assigned the label myColour.

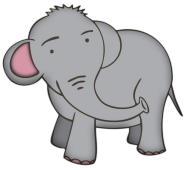
```
What is your favourite colour?
blue
I like the colour blue too.
Press any key to continue...
```

Here the word "blue" is assigned the label myColour.

```
What is your favourite colour?
yellow
I like the colour yellow too.
Press any key to continue...
```

Here the word "yellow" is assigned the label myColour.

In computer science the label myColour is called a variable because its contents can vary, as we have seen. In the previous example the contents of the variable myColour varied from the word "red" to "blue" and finally to "yellow".



Remember that variables are used to temporarily store data in the computer's memory. Each variable has a uniquely identifyable name or label, which enable you to use that data again or even change it.

It is a good tip to make your variable names meaningful, even if they may be a little long. If your variable name contains more than one word you can use a convention called camel case where the first letter of the words after the first word is capitalised (e.g., myNewVariable).

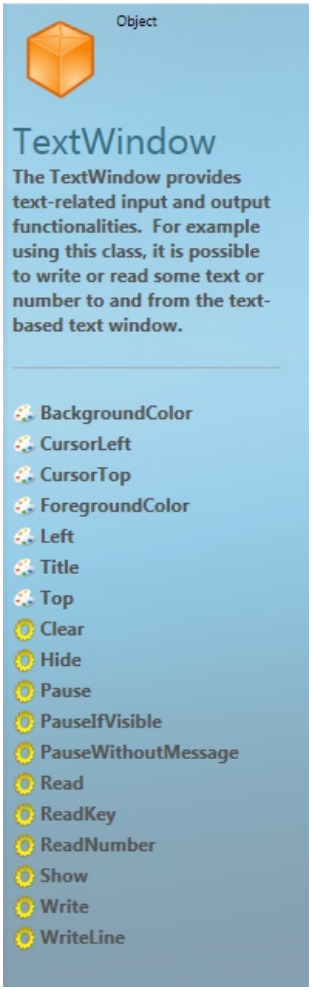
It is important to note that each variable contains just one item, for example the word “red”. When it reassigned to the word “blue” the word “red” has been overwritten and lost. For example, in the next program the contents of the variable myPet contains the word ‘dog’, then is overwritten with ‘cat’ and finally contains the word ‘goldfish’. The other words ‘dog’ and ‘cat’ are lost and no longer stored in the computer’s memory.

```
1 myPet = "dog"
2 TextWindow.WriteLine("myPet is a " + myPet)
3 myPet = "cat"
4 TextWindow.WriteLine("myPet is a " + myPet)
5 myPet = "goldfish"
6 TextWindow.WriteLine("myPet is a " + myPet)
```

Run this program. We see that the contents of the variable myPet has changed three times, from the word “dog” to “cat” and finally to “goldfish”.

A really useful way to learn how to program is to experiment. The intellisense system allows you to explore and try different objects, functions and properties easily. If something doesn't work just try something else.

TextWindow properties & functions



Small Basic comes with an intellisense system, which helps you select the correct objects and functions. When you begin typing the first intellisense window opens. You can use the arrow keys to scroll through the objects until you find the object that you wish to use, eg. TextWindow. Also as you scroll through each object a panel (like the one shown here) will appear on the right of the editor which describes the object and lists its associated functions and properties. After selecting your object (press enter) you should type a full stop after which a new window will appear. The new window displays the functions and properties associated with the selected object. Again a descriptive panel will appear on the right describing the selected function or property. The great thing about the intellisense system is that you can explore different objects and their associated functions and properties to find something useful.

If you use the intellisense system you can see that the TextWindow object has many more functions and properties, not just WriteLine(), Write() and Read(), which we have met before. Let us explore some.

Functions

An object's function is always followed by brackets (), sometimes we will need to put data inside the brackets for the function to work on. Functions differ from properties because they do something, in the same way as verbs differ from adjectives. We have been using the function WriteLine() since the beginning.

Properties

If functions are like verbs then properties are a little like adjectives because they are more descriptive in nature. For example they tend to describe, set the colour or position of an object. Properties are set using the assignment operator '=' equals sign. Try this example.

```
1 TextWindow.Title = "My Window"
2 TextWindow.BackgroundColor = "blue"
3 TextWindow.WriteLine("I'm feeling blue")
```

Can you tell which of these statements contain properties or functions of the TextWindow object?

Title and BackgroundColor are both properties, while WriteLine is a function of the TextWindow object.

Activity 2: functions and properties

Try these activities:

1. Make a TextWindow appear in the top right of your monitor It must have the Title “Here I am.”; inside the TextWindow must be the text “Small Basic is fun.”; and the text must be “red” on a “blue” background.

2. Use the TextWindow object’s function Read() and WriteLine() to assign a variable to create a conversation with the computer. Here’s the first three lines to help you get started.

```
1.  TextWindow.WriteLine(“Hello,  what  is
your name?”)
2. myName = Read()
3. TextWindow.WriteLine(“Hello “ + myName +
“.  What’s the weather like today?”)
```

3. Explore the some of the other properties and functions of the TextWindow and find out what they do. The intellisense system will give you a written description of each one.

4. The Small Basic Calculator

You can use the TextWindow’s WriteLine function as a simple calculator. Copy these simple statements into Small Basic.

```
TextWindow.WriteLine(128 + 23060)
TextWindow.WriteLine(345 - 34.79)
TextWindow.WriteLine(245.62 * 23)
TextWindow.WriteLine(45 / 5)
```

Activity 3: calculations

The purpose of the + (add) and - (subtract) operators are obvious, but what do the * and / operators do? In many (if not all) programming languages * is used for multiplication and / is used for division.

Try these questions.

1. Have a go at making some of your own calculations using * or /.

2. Programming languages can be quirky as well. Try “5” + “3” and “5 + 3” what is the difference between the results? This doesn’t happen in all languages so be careful!

4. Just as in mathematics, brackets are used to show the order of precendence in more complicated expressions such as 3+5*6, which can be interpreted as either (3+6) * 6 or 3+(5 * 6).

For example try this.

```
1 TextWindow.WriteLine(3.14 * (42 - 6))
2 TextWindow.WriteLine((3.14 * 42) - 6)
```

You'll notice that a subtle change in the position of the brackets will change the result significantly.