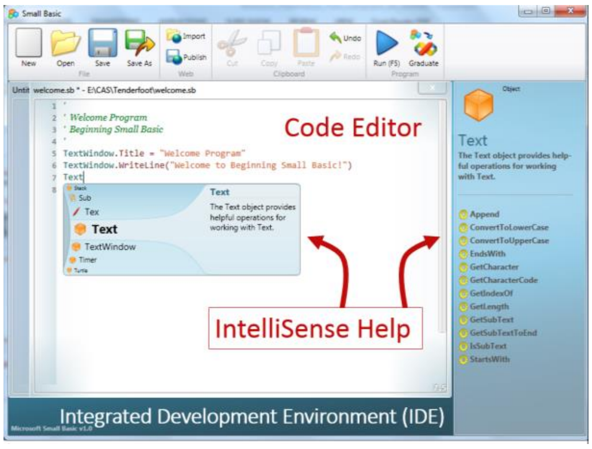
A practical introduction to Small Basic, reinforcing key concepts and constructs.

Preparation required:
Small Basic available on all computers, sample programs available in a shared repository.
Fizz Buzz activity sheet.

Small Basic Basics

Now that students have had some experience of writing text based code we can begin to try to draw everything together. As before, we’ll try to emphasise the 1 big idea, focusing on practical activities to illustrate the 3 constructs and 4 concepts we want to bring out from all the activities. We’ll take the same approach of targeting the exercises to minimise cognitive load and we’ll use another specially designed ‘transition environment’.

Small Basic gives pupils a feel of an industry standard IDE and has been designed to help transition to Visual Basic.NET. It has a code editor where children construct their programs, but one key reason for using it is the ‘IntelliSense’ which is a real help now that syntax becomes more complex. We can also begin to introduce the language of objects, which is common in many programming languages.



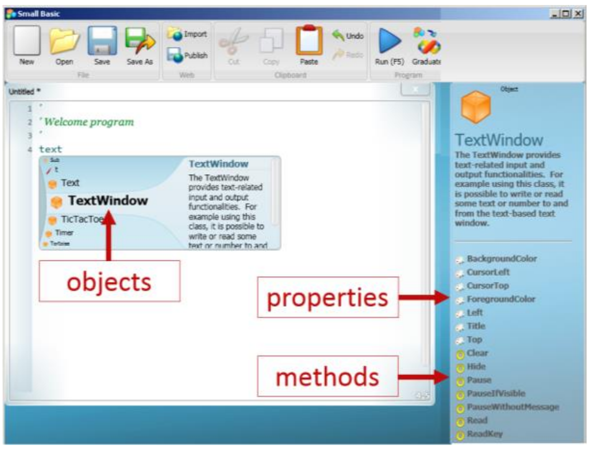
welcome.sb * - E:\CAS\Tenderfoot\welcome.sb

```
'
' Welcome Program
' Beginning Small Basic
'
TextWindow.Title = "Welcome Program"
TextWindow.WriteLine("Welcome to Beginning Small Basic!")
Text
```

Text

The Text object provides helpful operations for working with Text.

As before, work to ensure students can read code, before trying to write it is essential. The presentation steps through the basics. Comments start with an apostrophe, and are highlighted in green. Precision in describing each statement is important. Ask students to read the line highlighted before modelling the answer (on the slide). If we execute the code, a text window opens, with the title property set. All examples are included in the resources, the first being ***01Welcome.sb***.



Untitled *

```
'
'Welcome program
'
text
```

TextWindow

The TextWindow provides text-related input and output functionalities. For example using this class, it is possible to write or read some text or number to and from the text-based text window.

New terminology needs to be outlined at the start, with a particular distinction drawn between properties and methods. Don’t underestimate the extra cognitive load for some students with this new terminology and try to draw an association with concepts they are familiar with from previous work. Compare and contrast properties and methods, perhaps looking for associations with other concepts students are familiar with, such as Nouns and Verbs.

Intellisense lists the objects (and key words) based on what you type and when selected the properties and methods associated with the object are displayed and can be selected. This is a huge help in minimising syntax errors.

When you save a program, a Small Basic project file is created. This is your source code. When you run your program, a further 3 files are created. **SmallBasicLibrary.dll** is called a run-time library. It contains files that help your program run. The **Welcome** file (marked Application if you list the details) is the compiled version which runs and **Welcome.pdb** is a database file with information needed by your program. Three files with the same name can be a source of initial confusion, so it is worth pointing this out. It can be introduced as a quick investigation for pupils. The main point; with the potential to generate many files (if several programs are written), keep each project in a separate folder.

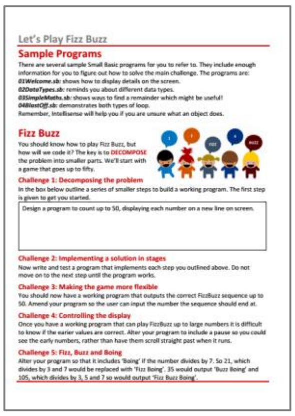
Key Concepts: Data Types

We can use Small Basic to introduce another key idea. We know that a single entity of data can be stored in a variable. Different types of data can be stored and depending on the data type, different things done. So we can add to our list of classroom chants: “There are 4 primitive data types. They are?” and the answer should come back Character, Boolean, Integer and Float. Combinations of characters are known as a string (of characters). A character can be any letter, symbol or digit. Children can struggle to distinguish between a number (made up of digits) and the use of a digit as a character. Phone numbers and vehicle registration plates are good examples to distinguish from integers or floats. You do maths on real numbers.

02DataTypes.sb makes the point about different data types behaving differently. Ask children to predict what will happen. Once they have created their mental model, open and run the code. The result will probably surprise them. Small Basic does not insist on specifying data types, but determines it by evaluating the value assigned to it. So both variables are integers. But… in order to display anything on the screen, it must be converted to a string of characters first. So the + acts to concatenate 3 strings. Uncomment the last line to compare the effect of bracketing the two numbers together. Now the calculation is executed before the conversion to a display string.

03SimpleMaths.sb demonstrates floor division and modulo. When reading the code, insist on precision in the explanation. Use the sample line to model the language required; “Two parameters, the starting number 28, and the divisor 5 are passed to the built in maths function, called remainder, and the result is assigned to the variable LeftOver”. When describing a complex statement, it is often best to read it from right to left.

Once students recognise the pattern in ***04BlastOff.sb*** they nearly always fail to notice it will print a zero before BlastOff. Possibly trace it on the board. The example gives the syntax for a counter controlled loop. An alternative condition controlled loop is shown in the presentation. Having pointed out the printing of zero on the last example, ask if this will print the zero or not? Out by one errors in loops are one of the most common programming mistakes.



Let's Play Fizz Buzz

Sample Programs

There are several sample Small Basic programs for you to refer to. They include enough information for you to figure out how to solve the main challenge. The programs are:
01Welcome.sb: shows how to display details on the screen.
02DataTypes.sb: reminds you about different data types.
03SimpleMaths.sb: shows ways to find a remainder which might be useful!
04BlastOff.sb: demonstrates both types of loop.
Remember, Intellisense will help you if you are unsure what an object does.

Fizz Buzz

You should know how to play Fizz Buzz, but how will we code it? The key is to DECOMPOSE the problem into smaller parts. We'll start with a game that goes up to fifty.

Challenge 1: Decomposing the problem

In the box below outline a series of smaller steps to build a working program. The first step is given to get you started.

Design a program to count up to 50, displaying each number on a new line on screen.

Challenge 2: Implementing a solution in stages

Now write and test a program that implements each step you outlined above. Do not move on to the next step until the program works.

Challenge 3: Making the game more flexible

You should now have a working program that outputs the correct FizzBuzz sequence up to 50. Amend your program so the user can input the number the sequence should end at.

Challenge 4: Controlling the display

Once you have a working program that can play FizzBuzz up to large numbers it is difficult to know if the earier values are correct. Alter your program to include a pause so you could see the early numbers, rather than have them scroll straight past when it runs.

Challenge 5: Fizz, Buzz and Boing

Alter your program so that it includes 'Boing' if the number divides by 7. So 21, which divides by 3 and 7 would be replaced with 'Fizz Boing'. 35 would output 'Buzz Boing' and 105, which divides by 3, 5 and 7 so would output 'Fizz Buzz Boing'.

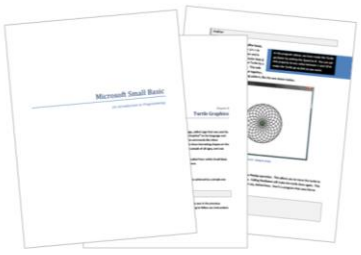
Let’s Play Fizz Buzz

Playing a few rounds of Fizz Buzz with the class first to ensure they all are familiar with the game. Children take it in turns to count up in sequence. If a number is divisible by 3 the child says ‘Fizz’ instead. If it is divisible by 5, the number is replaced with ‘Buzz’. The coding exercise is a classic test of the ability to structure constructs correctly. There is nothing new but, as always, the key to success is to decompose the problem first. If children attempt to write the entire thing, they will probably get lost.

Initially the upper bound could be set in the program to say 50 and the students encouraged to use a For loop. They have the example syntax but should be encouraged to read the Intellisense help. They will need to use it for the extension tasks. The solution will require the use of math.remainder. The key ‘gotcha’ revolves around numbers that are divisible by both 3 and 5, when the output should be ‘Fizz Buzz’.

There is an activity sheet with extensions and solutions to the various stages in the resources.

Taking Thing Further



Microsoft Small Basic

Turtle Graphics

A succinct guide (in the resources) comes with Small Basic. It gives plenty of ideas for simple programs to reinforce the basic ideas. At the risk of repetition, moving on too quickly is probably the biggest pitfall facing inexperienced teachers. Explore some of the shape investigations in the chapter on turtle graphics (Ch8). The preceding chapter on manipulating shapes in the graphics window would be good preparation for the tasks in the next session.