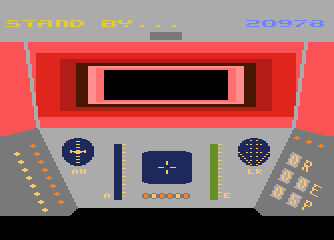
As you’ll have no doubt gathered from my previous ramblings, I grew up in the eighties. Actually that should be the **‘EIGHTIES’** as they were just so damn important. From the Sinclair’s, Commodore’s, the BBC, Atari’s, and then on to the PC. We won’t mention the consoles, oh the consoles!

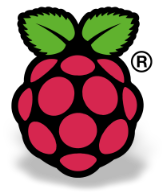
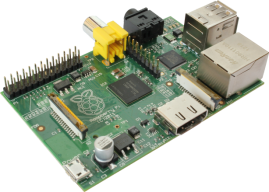
My favourites were of course the BBC due to Elite, the Spectrum as I started delving into machine code but most of all was the Atari 8 BIT series. When Lucasfilm released Ball Blazer and Rescue on Fractulas I was totally blown away. I spent a few months leaning the basics of 6502 machine code on my new Atari 130XE. It actually changed my entire outlook on computing and at the same time changed me as a person.

Up until then I had failed miserably at school, was totally under the impression I was pretty thick and had no idea what to do with myself. The discovery that not only could I program, but that I actually understood the inner workings of, what was at the time, something so very technical, changed my perspective. A year later and I was teaching all age groups to program, employed as a professional computer tutor, at just eighteen.

My confidence boosted, I left Minehead in Somerset and moved to London to make my start in life. The rest as they say is … well actually it’s not history, it is very much the here and now as everything seems to have led to this point.

A few years ago I brought an old BBC Micro down from the loft to show my kids how all this computing lark began. To my surprise my two girls Molly, Gracie and my son David were all intrigued by the BASIC welcome and the Syntax Error response returned by just about every input.

They wanted to know more so we spent a few days learning a few programming commands and playing a few classic games (Lemonade Shop I think). It got me thinking. Wouldn’t it be great to bring back a computer in the same vein? Something that brought access to programming right to the forefront just like it was back in the eighties, sorry, the **‘EIGHTIES’**.

When I started the project I looked for a system that booted straight to BASIC. The Maximite from the Australian tech guru Geoff Graham does this incredibly well. A simple single chip board with display, keyboard, sound and power inputs, it can be setup in minutes and boots to a very BBC like BASIC in about a second.

The Maximite was however superseded by the then new Raspberry Pi as this represented a much more powerful and flexible system. The only downside is that it did not have an accessible version of BASIC directly suited to our needs.

Enter Gordon Henderson, the author of the WiringPi libraries and a cool version of BASIC called RTB (Return To BASIC). RTB is designed specifically to support the workings of the Raspberry Pi. A deal was struck between FUZE and Gordon to produce FUZE BASIC which includes a vast array of enhancements to tailor FUZE BASIC more in line with the requirements of the newly revised IT curriculum so that we could best target the education market with our FUZE Platform.

At FUZE we focus on the language FUZE BASIC to deliver a learning experience far more accessible to broader age and ability group than more complex languages.

Quite simply, BASIC is easier to pick up and learn than just about any other language ever devised. You don’t need to be good at maths, you don’t need to understand the operating system to any great extent and you certainly don’t need to have programmed before.

It is important to add here that the Python, C+ (et al), Java and PHP programmers might scoff at us BASIC students, but I assure you BASIC has something for everyone. Even the most adept coders will find BASIC a great platform to test out ideas and experiment.

First thing then is to get FUZE BASIC.

Visit [www.fuze.co.uk](http://www.fuze.co.uk) and then go to the resources page and download the latest FUZE boot image.

You will need to unzip this file – be warned, unzipped it is 8GB and so requires a 8Gb SD Card to install it on. You will need software to copy the image as it is not as simple as just copying it over. There’s a perfect utility for this called Win32 Disk Imager at <http://sourceforge.net/projects/win32diskimager/>

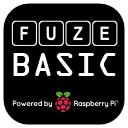
Download and run this, select the unzipped image and the destination SD card and then select write.   
  
This is also perfect for making backups of Boot SD Cards once you have set them up exactly as you want to.

Next boot your Raspberry Pi with your new FUZE image. There isn’t anything special with the boot image over the standard Raspberry Pi one, just a new background and a FUZE BASIC icon.

First off, navigate to the FUZE folder, double click it (enter it) it and then double click the Programs folder. Then create a new folder called MagPi.

Next we need to download a few graphics for our game tutorial.

Please go to [www.fuze.co.uk](http://www.fuze.co.uk) then to the Resources page. You will see a Tab called Tutorials and within this a section called The MagPi Tutorial.

I’m hoping you are up to speed with downloading and saving files. You need to download the six Sprites. These are the Player Ship, the enemy rocks and the ever important bullet.

Download and copy these six files into the MagPi folder we created earlier.

Back to the FUZE desktop, double click the FUZE BASIC icon to start FUZE BASIC.

The Welcome Screen will appear and you’ll be presented with the Ready >\_ prompt.

First off we should get used to the environment. The Ready> prompt means you are in Direct mode. Type in “Hello” and press Enter (not with the quotes). You’ll get and “Equals expected”. That’s exactly what should happen. The computer has no idea what Hello means. Try this;

Type;

**Number1=10 (press enter)**

**Number2=10 (press enter)**

**Answer = Number1+Number2**

I suspect many of you are already more than comfortable with what’s going on here but we need to explain to the newcomers.

**Useful keyboard shortcuts in DIRECT mode**

**Exit** Exit and return to desktop

**Dir** Lists the files in the current folder

**CD name** Change to folder NAME

**CD ..** Go back one folder

**Load name** Load program NAME

**Save name** Save program NAME

**New** Clear current program from memory

**F2** Go to the editor

**Run or F3** Runs the current program

The words “Number1”, “Number2” and “Answer” are just names. They are called variables. Variables are tags we store values in. We could have used any words but generally it is best to use names that make sense. If we write a program using variables like N1 and N2 and A then when we come back to them later we’ll have no idea what everything means, whereas “ShipX” and “ShipY” are obvious. Try and make this a habit. You will appreciate it later.

So, we stored the number 10 in the variable name “Number1” and 10 in the variable “Number2”. We then said that the variable “Answer” = “Number1” + “Number2”.

At this point you should know what the value of “Answer” is. Do you? I hope so or we’re in big trouble!

Type;

**Print Answer (press enter)**

You should see

**20**

If anything else whatsoever happens then something has not gone to plan and you should go back and check where you went wrong. On the basis you did see “**20**” returned we can proceed.

We are currently in “Direct” mode. This is where we can enter commands and expect an immediate response. We can’t do loops or a sequence of commands but we can check variables and enter simple instructions.

It’s not programming though is it? Press **F2** to enter the FUZE BASIC Editor.

You will see a blank screen with a green flashing cursor and a dotted line across the bottom. This is the Editor environment. Here we can enter a list of program instructions that will be saved and executed (**RUN**) whenever we want.

Press **F2** again. This will take you back to Direct Mode. Actually it will ask you for a file name. In this first case don’t bother, just press **F2** again and it will put you in Direct mode again. One last time, press **F2** again and you will be back in the Editor. You get the idea; **F2** takes you between the Editor and Direct mode.

Make sure you’re in the Editor and type in the program as shown.

|  |
| --- |
| **The Editor** (F2 switches between the editor and direct mode) |
| **CYCLE PRINT “Hello MagPi” REPEAT** |

You don’t actually need to worry about capitals or lower case – it’s a great habit to type in commands in capitals but it’s not essential. However the names we give to variables as we did above with “Number1” and “Answer” etc. are set in stone.  
If we assign a variable as “NUMBER1” then we must refer to it as such every time in the future. If we expect the number “numBER1” to return the same result, we’re in for a big surprise. The variable “numBER1” has not been defined so will generate an error.

Ok, enough of the dull stuff. You should at this point be in the Editor with the program as listed above. Press **F3** – if at this point the program hasn’t been saved it will ask you to do so. Just enter a name like “test” and press Enter. The program should then run. “Hello MagPi” should display in a never ending list down the screen. To stop it, hit the **ESC** key.

**Useful keyboard shortcuts in the EDITOR**

**F1** Displays a full list of all the keyboard controls to operate the Editor. Use this as it will have you flying around in no time!

**F2** Go to Direct mode

**F3** Runs the current program

Press **F2** to go back to the Editor and change the **PRINT** line so that it looks like;

|  |
| --- |
| **The Editor** (F2 switches between the editor and direct mode) |
| **CYCLE PRINT “Hello MagPi ”; REPEAT** |

The only difference is we added a space in between MagPi and the quotation mark and added a semi colon to the end of the **PRINT** line. The semi colon tells BASIC to display the next item next to the last one and not on a new line, the space just puts a gap in between. Press **F3** to **RUN** the program again. This time instead of a long list of “Helllo MagPi”’s going down the screen, this time it displays “Hello MagPi ” across the screen.

Again, press **ESC** to exit the program and then **F12** to wipe the current program from memory. All going well you should have a blank screen in the Editor. If not then try pressing **F2** and **F12** until you get there. When you are in Direct Mode you can type “**EXIT**” to exit the program.

Right now we want to be in direct mode with no program in memory. If you type “**NEW**” in direct mode it will clear the memory so when you go into the Editor it will be blank.

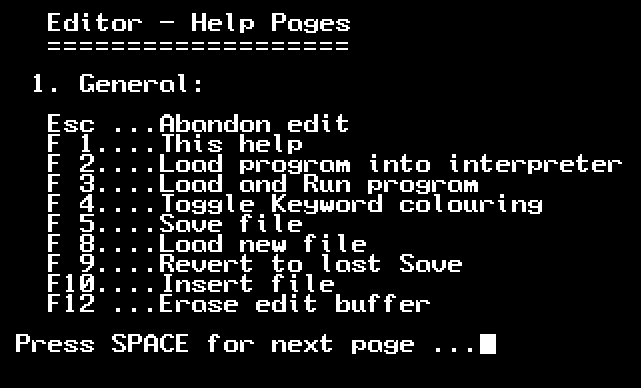
In direct mode type;

**Dir (press Enter)**

Amongst others, you should see a Directory called MagPi (if you did everything above). Type;

**CD MagPi (press Enter)**

This will put us in the same Directory (or Folder) where we saved the Sprites above. When we create our program we want it to be in the same Folder as the Sprites.

Press **F2** to go to the Editor and enter the following Program; If you didn’t use the New command then your previous program might still be there. If so press **F2** for Direct mode, enter **New** and press enter then **F2** to go back to the empty Editor

|  |
| --- |
| **The Editor** |
| **// MagPi Game**  **PROC Setup**  **CYCLE REPEAT**  **END**  **DEF PROC Setup ENDPROC** |

Press **F3** to **RUN** the program. The first time, it will ask you for a file name. Enter “MagPi” and press enter. You don’t need to enter the quotation marks or a file extension. The FUZE BASIC Editor will automatically add the file extension “.fuze”. When you press Enter the program will **RUN** but nothing of any interest will happen as we haven’t *done* anything of interest yet. If you have entered anything incorrectly you may get an error in which case **F2** will take you back to the editor. All going well the screen will just go blank as the program is in an infinite loop (**CYCLE** / **REPEAT**).

Press the **ESC** key and then **F2** to return to the editor.

This is the basic structure of our program. It is important as we progress, that we to try and learn some good habits. When naming Variables a popular method is called Camel Text. ThisIsCamelText. The reason it’s used is because we’re not allowed to use spaces in variable names. Camel Text (notice the humps) makes things readable at a glance.

Most programming languages are case sensitive and FUZE BASIC is no exception although it’s not particularly stubborn about it. Variables name are case sensitive so **CAT**, **cat** and **Cat** would be seen as three different variables. Commands and system variables however are not sensitive. **Print**, **print** and **PRINT** will all work. In homage to the BBC Micro there’s another useful short cut; ? can be used instead of PRINT. Nice ☺

As you write larger programs another variable issue will raise its head. Short, non-related variable names WILL cause you grief later. A 200 plus line program will be very difficult to make sense of if you have used things like “**pbx**” for “**PlayerBulletX**” and just “**X**” for the “**PlayerX**”. Those long names sure do take more time when editing, but man they will save hours later when debugging. Also consider at some point your code might be scrutinised by someone else. You do want to impress them don’t you… let alone make your program legible!

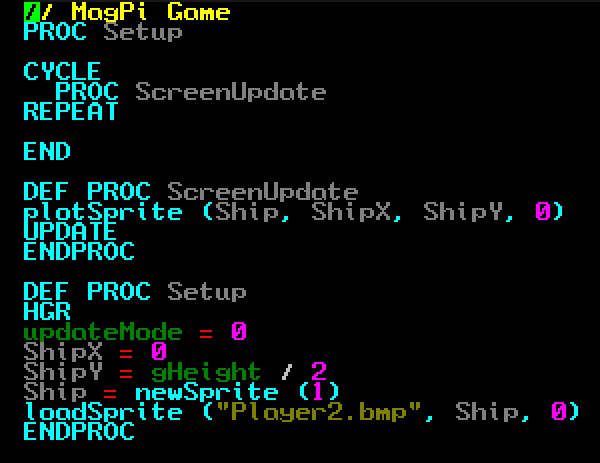
|  |
| --- |
| The Editor |
| **// MagPi Game**  **PROC Setup**  **CYCLE**  **PROC ScreenUpdate**  **REPEAT**  **END**  **DEF PROC ScreenUpdate**  **plotSprite (Ship, ShipX, ShipY, 0)**  **UPDATE**  **ENDPROC**  **DEF PROC Setup**  **HGR**  **updateMode = 0**  **ShipX = 0**  **ShipY = gHeight / 2**  **Ship = newSprite (1)**  **loadSprite ("Player2.bmp", Ship, 0)**  **ENDPROC** |

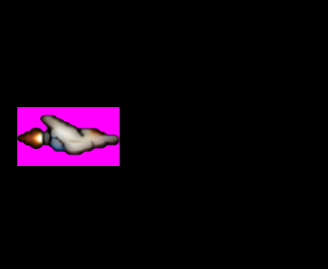
**Time to play a game**

We are going to store our Variables in a **PROCEDURE** called **Setup** along with our Sprites and sound files. This keeps our program nice and tidy. The **CYCLE** and **REPEAT** commands define our main program loop. This is where the main program flow will happen.

Now we need to load the Sprites so we can start having some fun. Go to the editor with **F2** if you’re not already in it.

Edit the code as shown opposite;





When you run the program you might be surprised to see a bright pink box surrounding our space ship. Don’t worry, there’s nothing wrong as we just need to set this colour to transparent.

FUZE BASIC will not draw any colour specified as the transparent colour.

We need this because a sprite graphic is quite simply a box and everything in the box is drawn. So if we have a white background in our sprite it will display a white box which is no good on our black space background. We can stop this by specifying a single colour to be transparent so it is not drawn. Add the ‘setSpriteTrans’ command directly below the loadSprite command as shown below and then run F3 the program again;

|  |
| --- |
| The Editor |
| loadSprite ("Player2.bmp", Ship, 0)  **setSpriteTrans (Ship, 255, 0, 255)** |

Hmm..  
that’s much better!

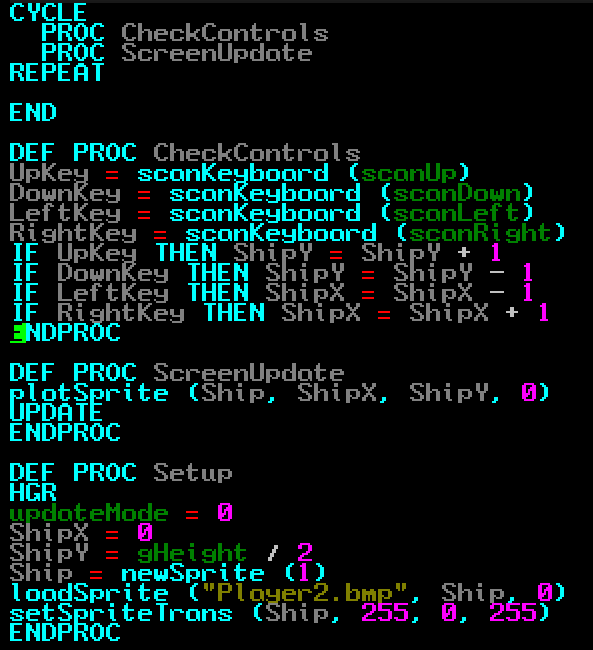
Now that we have something more substantial, let’s take a proper look at what’s going on;

|  |  |
| --- | --- |
| **// MagPi Game**  **PROC Setup**  **CYCLE**  **PROC ScreenUpdate**  **REPEAT**  **END**  **DEF PROC ScreenUpdate**  **plotSprite (Ship, ShipX, ShipY, 0)**  **UPDATE**  **ENDPROC**    **DEF PROC Setup**  **HGR**  **updateMode = 0**  **ShipX = 0**  **ShipY = gHeight / 2**  **Ship = newSprite (1)**  **loadSprite ("Player2.bmp", Ship, 0)**  **setSpriteTrans (Ship, 255, 0, 255)**  **ENDPROC** | Anything displayed after the // is ignored. This allows us to add comments to make the program easy to understand.  Tells the program to jump to the Procedure Setup, run whatever is there and return when it comes to the ENDPROC command.  These three lines define our main program loop. Whatever is between the CYCLE / REPEAT command will be repeated indefinitely.  END signifies the end of the program. When the program executes the command it will return to Direct mode.  This Procedure is going to update everything on the screen at the same time. ENDPROC return back to where it was called from.  The plotSprite draws a sprite (Ship) at X & Y coordinates with the sub ID 0.  When we draw graphics to the screen they are actually drawn to a background screen, then the UPDATE command copies the background screen to the main screen. This keeps everything running smoothly and simplifies games programming.  Our final Procedure is deliberately placed at the end of the program. It is a good habit to keep all the main setup commands in one place so it is easy to find them. Also this will usually end up being quite a big routine so you don’t want it getting in the way at the beginning.  HGR sets High Resolution graphics, updateMode sets the screen update system to manual.  ShipX and ShipY are used to store the x & y coordinates of the player’s ship. ShipY takes a system variable called gHeight which is the height of the screen and divides it by 2 to work out the middle.  Ship = newSprite(1) creates a Sprite ID called Ship with room for just one graphic, the sub ID. Later we will increase the sub ID so we can animate them.  LoadSprite assigns the named graphic to the Ship ID at position 0.  setSpriteTrans sets the colour (Red, Green, Blue) value to bright pink.  ENDPROC signifies the end of the Procedure and returns to where it was called from. |

Ok so now we’re making progress but unfortunately we don’t have space for much more this month. Let’s add one more bit to make it feel like we’re really getting somewhere. Add the following lines to your program so it looks like the listing on the right;

|  |
| --- |
| Change the program so |
| CYCLE  **PROC CheckControls**  PROC ScreenUpdate  REPEAT |

|  |
| --- |
| Change the program so |
| END  **DEF PROC CheckControls**  **UpKey = scanKeyboard (scanUp)**  **DownKey = scanKeyboard (scanDown)**  **LeftKey = scanKeyboard (scanLeft)**  **RightKey = scanKeyboard (scanRight)**  **IF UpKey THEN ShipY = ShipY + 1**  **IF DownKey THEN ShipY = ShipY - 1**  **IF LeftKey THEN ShipX = ShipX - 1**  **IF RightKey THEN ShipX = ShipX + 1**  **ENDPROC**  DEF PROC ScreenUpdate |



Run the program with **F3** – I’m going to be very disappointed if you haven’t worked out what will happen when you press the cursor keys.  
  
Next month we’ll add a few enemies and some fire power. See you then.