ISSUE 01 MAY 2012

The MagPi

A Magazine for Raspberry Pi Users

- RacyPy
- Debian
- Scratch
- Python

Introducing... The Raspberry Pi

A new breed of computer

NO MORE APPLES FOR TEACHER!

QR Code:

http://www.themagpi.com

The MagPi

Over the coming issues, The MagPi will explore the exciting things that can be done with this very special computer.

We will introduce you to the various Raspberry Pi operating systems, how to program in a range of languages, and start your own interesting projects. We aim to help experts and beginners get the most out of the Raspberry Pi hardware and more importantly help build a fun and friendly community for everyone to get involved with.

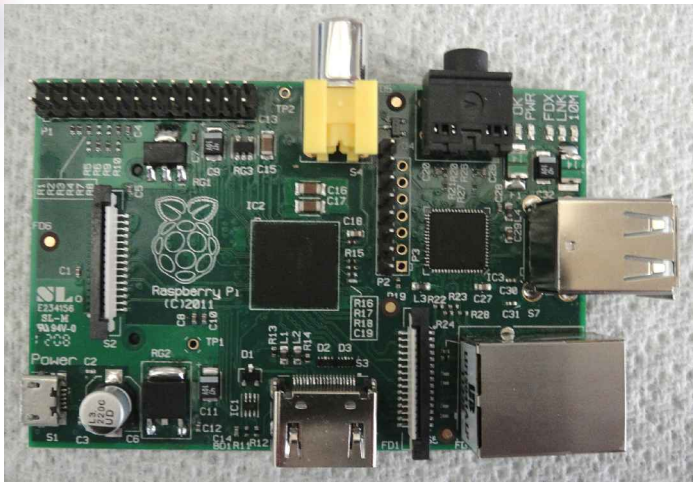
Fresh fruit

If you like your apples to be white and shiny, your apricots made from silicon, and you'd rather talk to your blackberries, then you may be in for a treat...

The Raspberry Pi computer is the size of a credit card, completely silent and costs under £30. The operating system runs from an SD flash card, allowing its personality to instantly be switched by swapping cards. Its potential uses are staggering, and as yet, not fully explored, but it has already been tested as a multimedia player with streaming capabilities, a games machine, an internet browser and a hardware development board. It is intended to be used as an educational device for people of all ages and skill levels.

Interest in the Raspberry Pi is already incredible, and demand has far exceeded expectations. IT professionals, electronics experts and newcomers are all eager to get their hands on the small device which everybody agrees 'is going to be huge'.

Article by Jaseman & Meltwater



Lucky Element 14 Road Test Prize Winner 'GizmoB73' unboxed this late 'R&D' prototype.

Contents

The Dawn Of Affordable Computing

The ZX81 and Spectrum by Sinclair Research, were among the first affordable home computers available during the early 1980's.

Although these 8-bit computers were crude by today's standards, they gave ordinary people the opportunity to write computer programs and games.

The BBC Acorn computer was backed by the British Government for use in education and subsidies were granted to schools. These computers were a little too expensive for most families to have at home.

The 'Electron' was Acorn's attempt at addressing the price issue. They sold a number of units, however despite being more powerful on the hardware side, the range of game titles was significantly smaller than rivals Sinclair and Commodore.

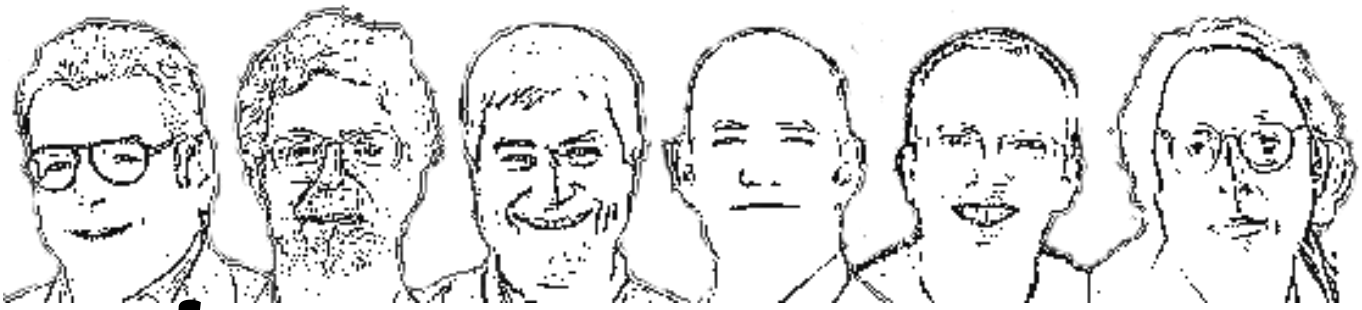
The Fall Of Programming

Home computer users were becoming less interested in writing their own programs.

Slick-looking, professionally made games that were coming out of software publishing companies were compelling.

Ocean Software **for example, boasted an impressive library of hit game titles such as 'Daley Thompson's Decathlon', 'Chase HQ' and 'New Zealand Story'.**

The trend of moving away from programming followed within the education sector. Early software packages like Wordstar, and Lotus 123 were difficult to use before the days of the graphical interface, but typing skills, word processing, spreadsheet and database training continued to be the focus. Programming was seen as a very specialist and niche area of study.



The Raspberry Pi Foundation Trustees

The Pioneers

The story of how the Raspberry Pi computer came to be.

Cambridge and beyond

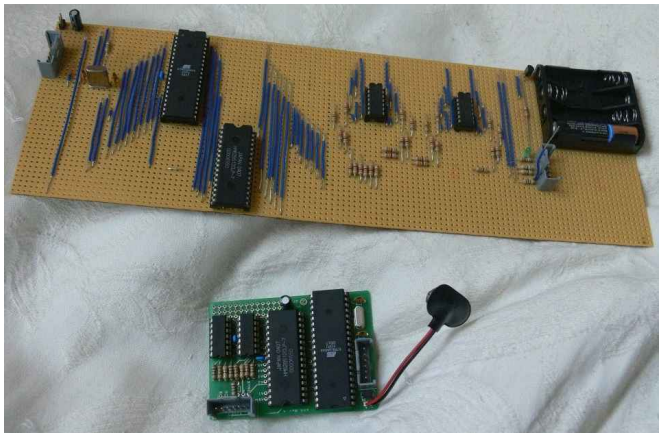
Cambridge University attendees have played a major role in the development of computers in Britain. The story of how these individuals have branched out into various organisations such as ARM Holdings plc, Acorn, Sinclair Research, Element 14, Broadcom and the Raspberry Pi Foundation is a complicated one.

To understand the links between all of these companies and organisations is extremely difficult, due to many mergers, acquisitions, dismemberments, shareholdings, employment movements and so forth, and many a journalist has been beaten trying to unravel exactly who did what, and who now owns it.

What we do know is that the Raspberry Pi project began in 2006. There was a common desire among certain individuals to recapture a sense of the pioneering spirit of computing that came during the 1980's, when affordable personal computers became available to the average hobbyist or budding computer enthusiast. There was a growing concern about the diminishing interest in computer science and an opinion that the ICT curriculum had become too focused upon word processing, spreadsheets and databases.

Initially the plan was to build a simple microcontroller-based computer that booted straight into a Python interpreter prompt (The Pi in the Raspberry Pi name was a reference to the Python programming language).

This cheap device was intended to be used to promote and invigorate the next generation of programmers, and developers.



Early microcontroller-based circuit board

Due to the cost of producing a working system with networking capabilities, device drivers, etc. built into the interpreter, it was decided that it would be easier to use an already freely available operating system (Linux). Using Linux would also allow flexibility in the choice of programming languages and other software that could be used.

The option of using a 'System on a Chip' (SoC) became preferable to a microcontroller computer. Broadcom had developed a range of ARM processors for use in smart phones.

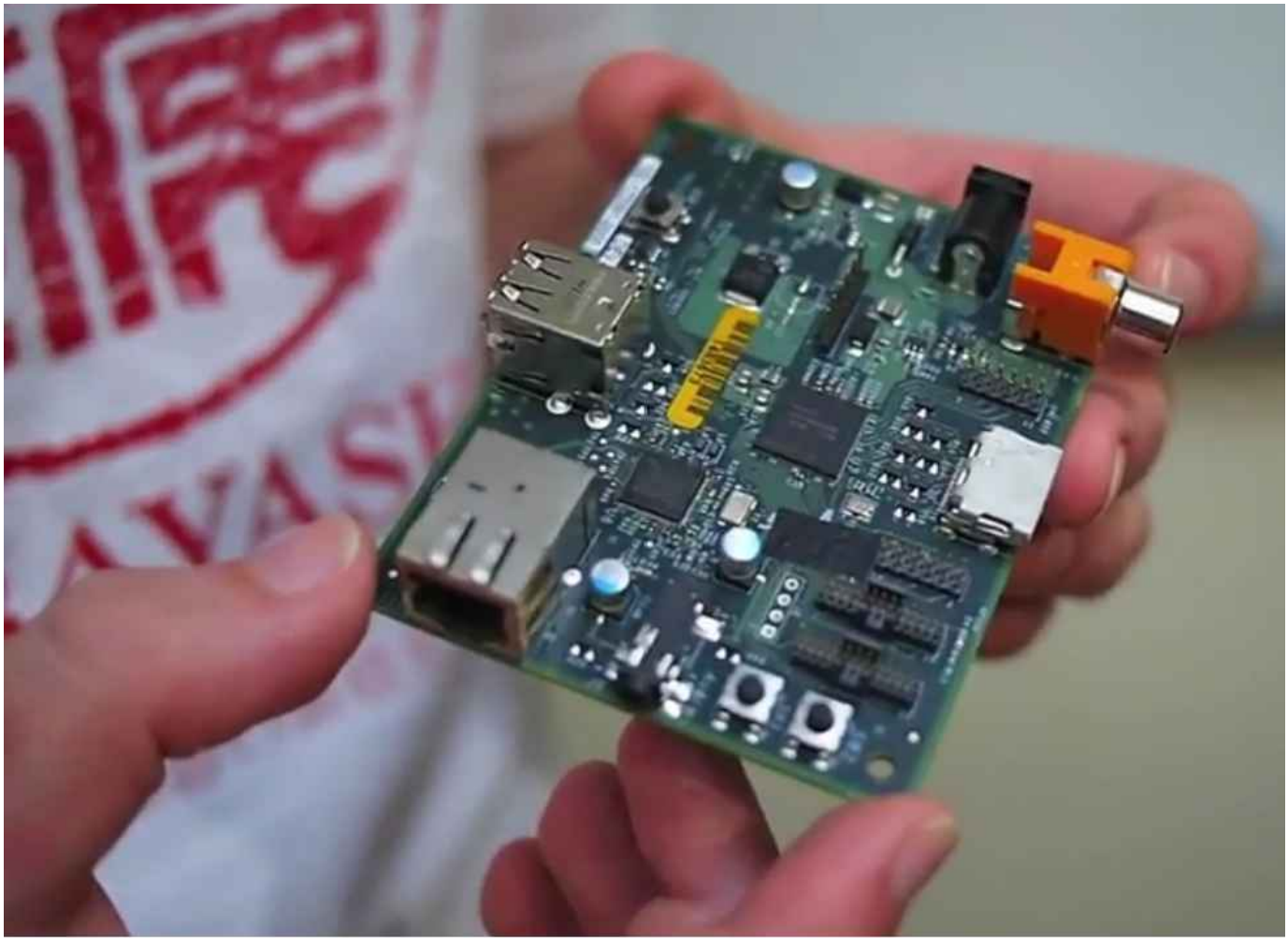
Broadcom employee Eben Upton, and other luminaries such as David Braben (Famed writer of the BBC Micro games 'Elite' and 'Frontier'), Jack Lang, Pete Lomas, Professor Alan Mycroft and Dr Robert Mullins set up a charity called 'The Raspberry Pi Foundation'. The idea they had in mind was to design a SoC board populated with a Broadcom ARM11 processor chip, which could be produced and sold at a very inviting price, to a potential new generation of computer science engineers.

The first prototype boards were the size of a USB memory stick, with 1 USB port at one end and a HDMI video output at the other end. These boards used microSD memory cards to store the Linux operating system. The boards were too small to accommodate a network port, GPIO header, composite video and analogue audio outputs, which would mean that they were unsuitable as hardware development boards.

USB stick sized early version of the Raspberry Pi

It was decided that a credit-card sized board would be sufficient, and The Raspberry Pi Foundation set about designing the board layout and working with the Linux community to develop and refine a number of Linux operating system distributions that would work well with the 700MHz ARM1176JZF-S processor. The initial design had an SD card reader fitted to the underside of the printed circuit board (PCB), however the first prototype card readers protruded slightly outside of the credit card shape.

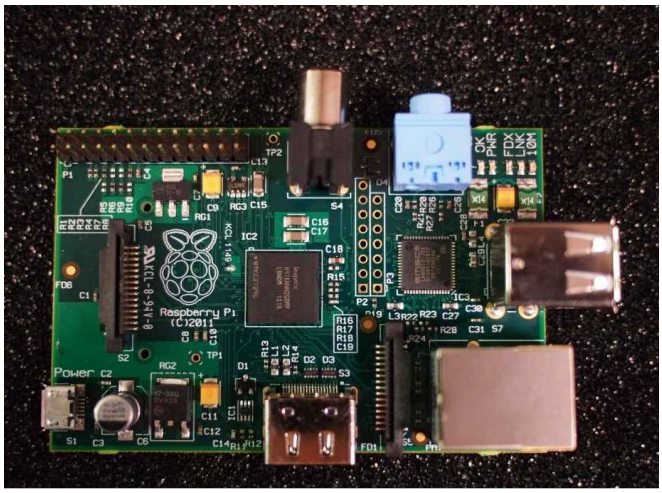
(continued over page...)



Co-founder Eben Upton displays one of the first Alpha boards

In August 2011, fifty alpha boards were produced. The SD card reader had been repositioned to be flush with the edge of the card.

By December 2011, beta boards of the Raspberry Pi were able to demonstrate Full HD 1080p video playback and a ported version of the Quake 3 game, using the onboard Videocore IV graphics processing unit (GPU). Notably the beta boards were fitted with micro-usb power connectors. This meant that it would be possible to use commonly found phone chargers to power the boards.



Raspberry Pi Beta board

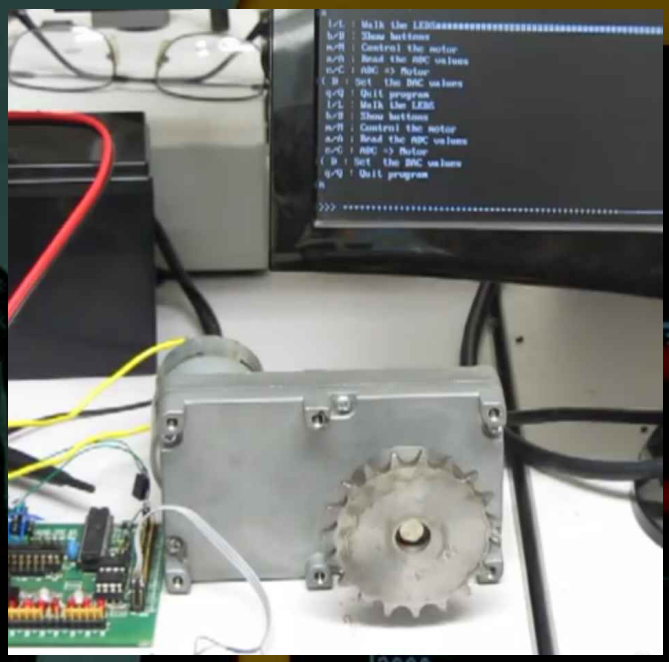
In January, The Raspberry Pi Foundation auctioned some of their prototype Raspberry Pi boards on Ebay and announced that the first 10,000 Raspberry Pi computers were being manufactured in China. They would be sold through industial components suppliers 'RS Components International' and 'Premier Farnell' with additional design engineer community support provided through 'Element 14'.

After a few setbacks, the first Raspberry Pi boards were available for pre-order. The official launch was at 6am on the 29th of February 2012.

Article by Jaseman

Hardware Development and Robotics

Gert van Loo has developed a General Purpose Input/Output (GPIO) expansion board for the Raspberry Pi. The board is able to control flashing light-emitting diodes (LEDs), and connect to servo motors, sensors and other electronic components.

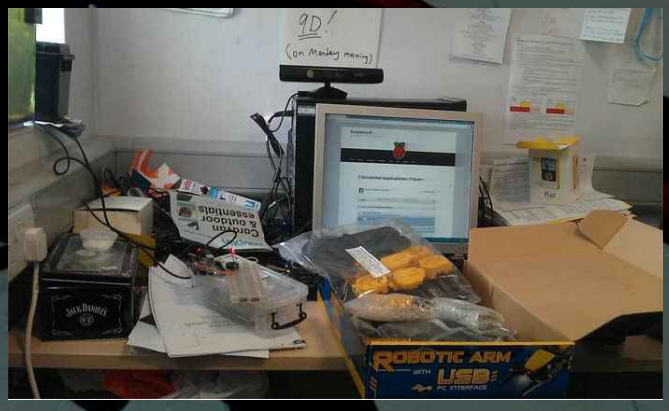


Gertboard & Motor

Enthusiast 'Bodge N Hackitt' is planning to connect a robotic arm to the USB port and GPIO header of a Raspberry Pi. The first part of his series of articles can be found on pages 10-11.

OWI USB Robotic Arm Kit

Over the following months, we will be following these and other home-grown projects.



SKUTTER

By Bodge N Hackitt

In these articles I will be describing some experiments into controlling real world applications using various aspects of the Raspberry Pi computer.

I was inspired to start this project when a member of the Raspberry Pi forum brought to my attention a low cost robotic arm kit that could be controlled via a computer's USB port. The arm reminded me of something called a “skutter” from one of my favourite TV shows, Red Dwarf (Pictured right) that was on when I was still growing up. I thought that, as the Raspberry Pi is so small, has such low power requirements and is still so useful as a computer that I could easily combine the robot arm with a kind of motorized base and have the whole thing run from batteries and powered by a Raspberry Pi which would act as the Skutter's 'brain'

It seemed to me that this would make a fun kind of project which could even have some serious applications. Imagine if you could make an intelligent robot that was able to run about on its wheels and manipulate things in the real world with its robotic arm. Yes, it could be fun just in its own sense but also it could be of tremendous use for people with disabilities. It might even one day be useful as a kind of fix-it bot that could access small spaces where people would find it difficult or dangerous to get to.

In order to accomplish this, the Raspberry Pi has two interfaces which will allow us access to control stuff. The first of these is a kind of standard interface which almost all modern computers have built in. It's called a “USB” or Universal Serial Bus. I will explain how some elements of USB work in the next article but it is worth noting here that this little port that we take for granted is extremely complicated. Fortunately there are some things which will make our work with this easier.

The second of these interfaces on the Raspberry Pi is called the GPIO which stands for General Purpose Input (and) Output. This part of the Raspberry Pi promises to allow the user to easily control all manner of devices but at the time of writing this article the Raspberry Pi is not available.

This means that the first of these experiments will be carried out using a “virtual” Raspberry Pi running on a PC (This is not strictly true, a proper virtual RasPi

would involve programming a simulation of the ARM processor, its memory, storage and all its other associated parts. My “virtual” Raspberry Pi is actually just a minimal installation of Debian-Linux (the operating system) that's running on a program called Oracle Virtual Box. Luckily this set up does give me access to the USB port and this should work exactly the same in this virtual implementation as it would on the real thing. (Sadly, at the moment we can't try things that would use the GPIO as there is currently no straightforward way of making a virtual Raspberry Pi with a virtual GPIO header).

The robotic arm kit called the “OWI Edge” is currently available from Maplin electronics and it uses a simple USB interface to control it.

This kit is sold with a CD ROM with software that will allow you to control it from a Windows PC. The software allows you to remote control the robot arm or to make a sequence of instructions that controls it by setting the time and direction that each joint should move. The design of this robot and software is very limited. The software only allows for the most basic of programming and movement and the arm itself has no feedback. This means that, as it is we have no way of knowing what the arm is doing other than by looking at it ourselves.

A truly useful robot needs to know itself where it is in space. In theory we should be able to say to it “Get the object over “*there*” and the robot will know I am “*here*” and I need to be “*there*”. To get to “*there*” I must move by “*this*” much and in “*that*” direction.

To make this simple robotic arm into a more useful device we will need to do some 'hacking' and make some modifications.

As this project progresses we will cover some ways of adding simple “*feedback*” to our robotic arm so that it will know where each arm part is in the space around it (its environment).

We will also investigate some more advanced methods of giving our robot a kind of sense of sight by using “sonar” and we will consider some further possible applications of machine vision using cameras.

When the actual Raspberry Pi computer becomes available we will be able to start to put things together properly and begin to construct our motorized base.

In the next article I will cover methods of controlling the robotic arm via the USB by writing instructions in the Python programming language. Looking forward to seeing some of you again then.

Bodge N Hackitt.

A Tasty Bit Of RacyPy

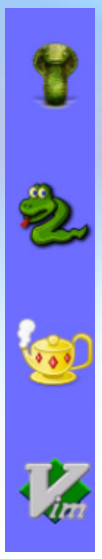
Why not have a play with RacyPy and jump into the world of Linux and programming? RacyPy which uses Puppy, a lightweight Linux distribution, which can be used to try out Linux on any PC without even installing it. This is packaged with a range of tools ready to go, so you are ready to try the tutorials later in this magazine.

What is RacyPy?

RacyPy, is a pre-packaged cd image which when "booted" (run from power up) will load a graphical desktop similar to a MS Windows or Apple Mac computer.

The RacyPy image is setup as a “Live CD” which means it loads directly to memory and nothing is installed.

RacyPy comes with a range of software ready to run, such as:



AbiWord — word processing

Gnumeric — spreadsheets

SeaMonkey — web-browsing

mtPaint — graphic editing

Geany/Vim — programming editors

MPlayer/Ogle/Pmusic — video/DVD/music playback

Python 2.7.2 and 3.1.4 — programming

Plus many other useful utilities. If you decide to use RacyPy more than once you can also setup a USB Memory Device to store your user data on, this will also allow you to install any additional programs which will be loaded every time you reload it.

Thanks to TeamPython and antiloquax for permission to use RacyPy!

Get Ready!

First, obtain the iso image file:

Racy-5PyTeampython.iso (384Mb)

http://goo.gl/UykP4

Depending on the method, you’ll need:

Live CD	Blank CDR & CD/DVD Writer*
Virtual Machine	940Mb hard-drive space This includes: ~300Mb for the Virtual Machine software when installed 384Mb for the iso file 256Mb for temporary RAM space (only needed when running)

If you intend on keeping your settings (which can then be loaded each time), then it is recommended you have a USB flash memory stick of at least 1Gb or more.

Universal USB Installer

http://www.pendrivelinux.com

If you don't have a CD/DVD drive on your system it is also possible to boot and run Linux from a USB memory device (1Gb or more).
Follow the guide from *http://goo.gl/S4VTI* but instead of "Lucid Puppy" select "Racy Puppy". Then press "browse" to Racy-5PyTeampython.iso you downloaded above. If you don't see it, try ensure you have named it with “racy-5” at the start of the filename.

LiveCD Boot

The simplest way to use this image is to "burn" the "image" to a CD. You can use your own software or you can use a free program called ImgBurn, which will ensure the CD is written correctly to allow it to boot.

ImgBurn — http://www.imgburn.com

Once you have installed the software, you can "right click" on the iso file and select "Burn using ImgBurn". Alternatively, run the program and select, "Write Image File To Disc" and select the iso file (File ->Browse for a source file…).

Insert your blank CDR into your CD writer drive and start the write process, by pressing the write button

Once you have a CD, you need to ensure your system will "boot" from it (use the CD to start up instead of your normal operating system). If your computer doesn't do this automatically, often there is a particular key you can press (F2, F8 etc) to select booting from an optical device. Each computer is slightly different so pay particular attention to any messages which flash up when you first switch on your computer.

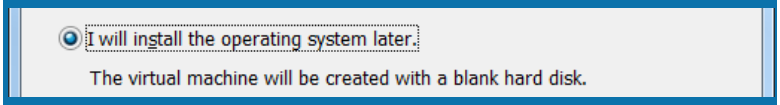
Virtual Machine

A really useful way to try out Linux is to use a virtual machine (VM). A VM is simply a simulated computer which runs on top of whatever operating system you are using.

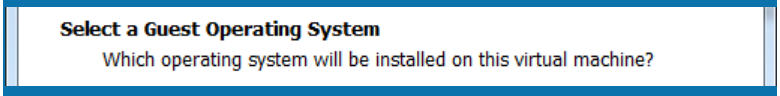
This means you can run this virtual computer in a window alongside your other windows, and everything to do with it is contained within a few files separate to your normal operating system. When you are done, you can simply remove them!

To run RacyPy we recommend using VMPlayer, which can be obtained from *http://www.vmware.com* or directly from *http://goo.gl/8oKED*.

1. Once you have installed VMPlayer, open the program and select "Create a New Virtual Machine.



2. Select "I will install the operating system later".

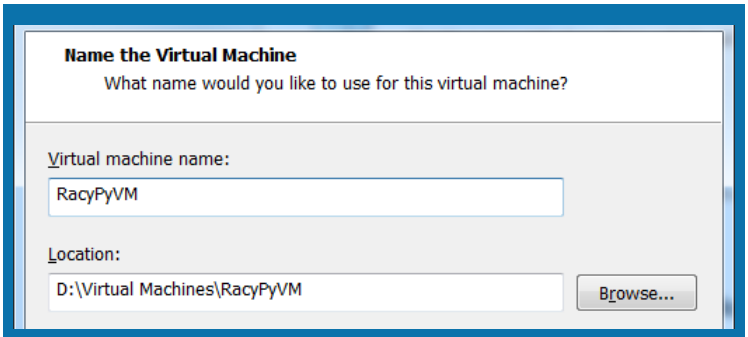


3. Select "Other" and "Other" for the Guest Operating System. We do not need any specific features, so generic options should be fine.

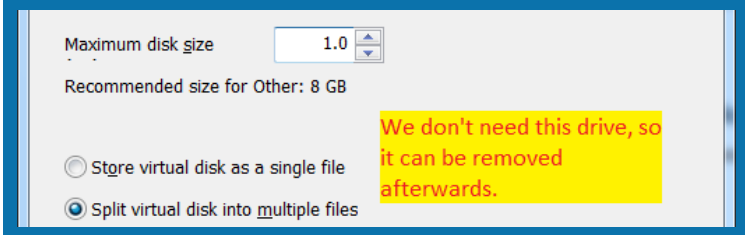
continued over the page...

Got a taste for more?

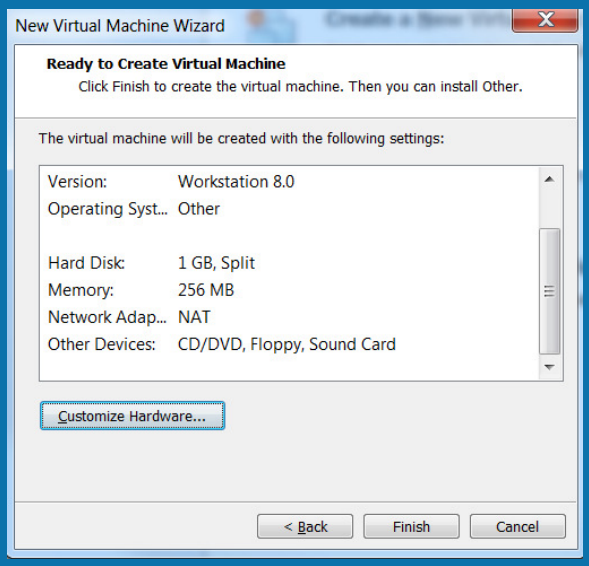
You can install and try out Debian (the official raspberry pi distribution), later on in this issue.



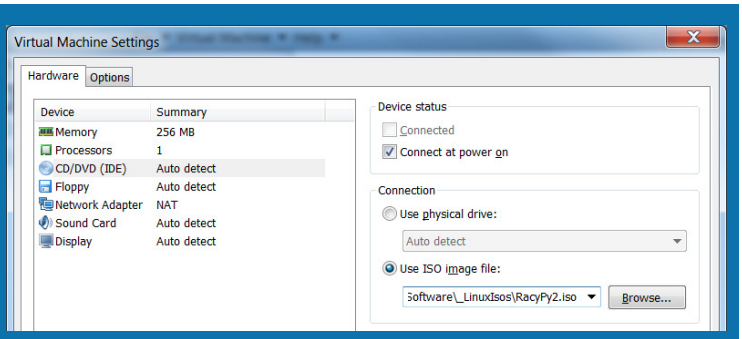
4.Name your Virtual Machine and locate it where you want to. It'll only use up ~2Mb of space when not in use. However, when it is running it'll create a temporary RAM file (default is 256Mb). If you "suspend" the VM (pause it so you can return to it in it's current state) it'll keep that file, otherwise it will be removed when you "power off" the VM.



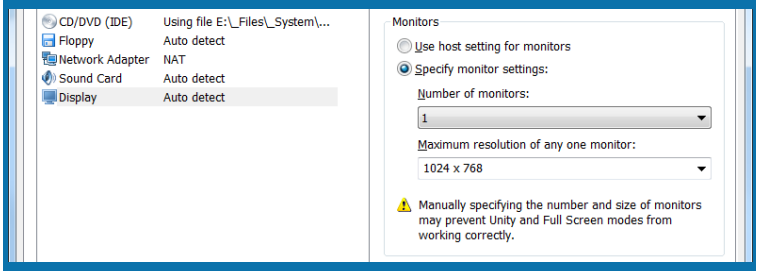
5.Next "Specify Disk Capacity", we don't need a virtual disk, but the wizard asks anyway. Select 1Gb size, and to split the drive, but since we don't use it, it won't take any space up.



6.Finally confirm you want to create the new VM, by pressing Finish. However, don't run it yet, we have a few more tweaks to make.



7.Next we will virtually insert our RacyPy image into our virtual CD/DVD Drive. Right click on your new virtual machine, and select "Virtual Machine Settings...". Select "CD/DVD (IDE)", then "Use ISO image file" and browse to where you have downloaded your RacyPy2.iso image (ensure you keep it where you won't move it. i.e. in the VMs directory). Ensure this is set to "Connect at power on".



8.Finally, to allow Puppy to sit comfortably within a window, we can force the virtual screen to be a fixed size, ideally slightly smaller than the monitor you are using, so for a 1280x1024 monitor, 1024x768 is fine.

Now you can run the new VM, and boot into Puppy!!

VMPlayer Tips:

1. To control the VM, click in the window to capture your mouse and keyboard, then press Ctrl+Alt to release.

2. If you use a USB Flash Memory to store your user data, next time you run the VM it will remind you to re-attach your key if it is missing!

First Boot

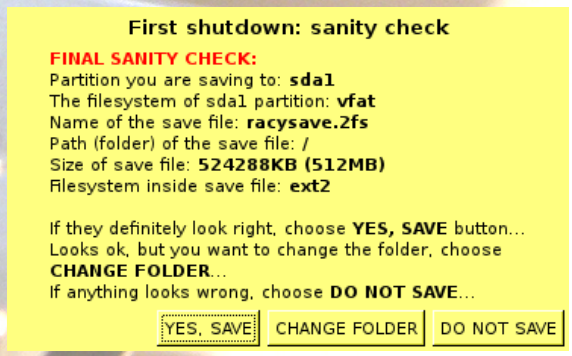
When you boot for the first time, you will be able to configure your setup. Puppy allows you to store all your user data conveniently on a USB memory device so it is available next time you boot (regardless of which machine you are using).

To Keep Your Data

After booting and configuring your setup,

1. Select "Shutdown" and "Power-off computer" from the bottom "Menu" button.
2. You will be prompted by the "First shutdown: save session" window. As recommended, choose to "SAVE TO FILE".
3. We will select the defaults (carefully read the info there for other options).

First shutdown:	***Default Action***
ask fido	Select "administrator"
get ready	Plug the USB memory device in*
choose filesystem	Select "ext2"
save-file name	Leave blank
encryption	Select NORMAL
size save-file	Select 512Mb**
sanity check	Should match below!



4. After the 2fs file is written it may prompt "copy .sfs from CD", this is optional (it will use 379Mb extra on the USB memory device).
5. The next time you boot with the USB memory device attached, it will load up your settings and programs from your last session.***

NOTES:

*If using a VM, you may need to click on the device icon at the bottom of the window (next to VMWare logo). If successful you will see the following icon appear on the desktop.

**The remaining space can then be accessed and used by both Linux and Windows.

***On the 1st boot after this, you may be asked about adding "BootManager:SFS files", press ok for now, we will cover this another time.

Optional "Pets"

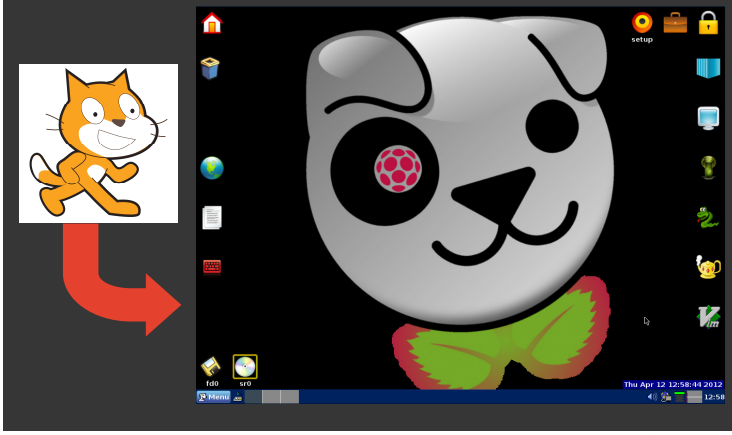
Since this is based on Puppy Linux, there are many packages which are extra easy to install called "pet" files. The files contain everything to install and setup the program.

Installing Pets

Either download the "pet" file directly within RacyPy or put the file on your USB memory device. Click on the file and confirm you want to install. The package manager will tell you when it is installed and where in the menu to find it (i.e. Scratch will be put under "Utilities").

For exampl Scratch, as used in this mag:

Scratch.pet — *http://goo.gl/YQKNw* (33Mb)



More information on using RacyPy and also tips and guides on getting started programming with Python and Scratch visit:
http://raspberrypy.tumblr.com

Article by meltwater

A step-by-step guide to creating a virtual machine of the 'Debian Linux' operating system to run on your Windows PC.

An 'Operating System' (OS) is a set of instructions that help a computer to understand how to control the connected hardware devices.

The 'Boot Sequence' is a list of events that begins when you power up your computer.

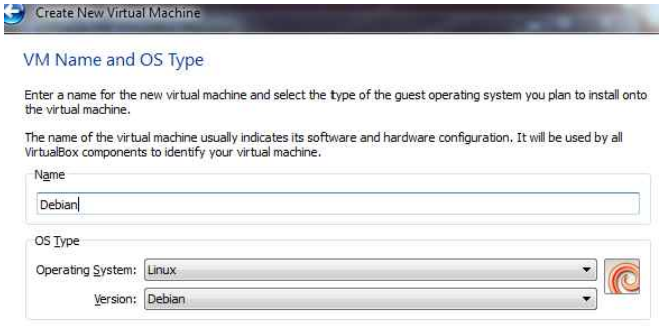
A state of readiness is usually indicated when all the screen activity has stopped and you are confronted with a logon prompt or graphical user interface (GUI). In this example the GUI is called 'LXDE' - Lightweight X11 Desktop Environment.

Creating the Virtual Machine

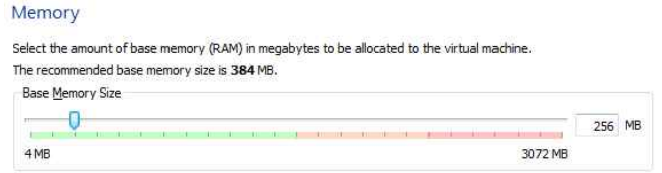
1. Start Oracle VM VirtualBox

2. Click 'New'

3. Click 'Next' to start the 'New Virtual Machine Wizard'



4. Name: Debian O.S.: Linux Version: Debian



5. Memory: 256 MB

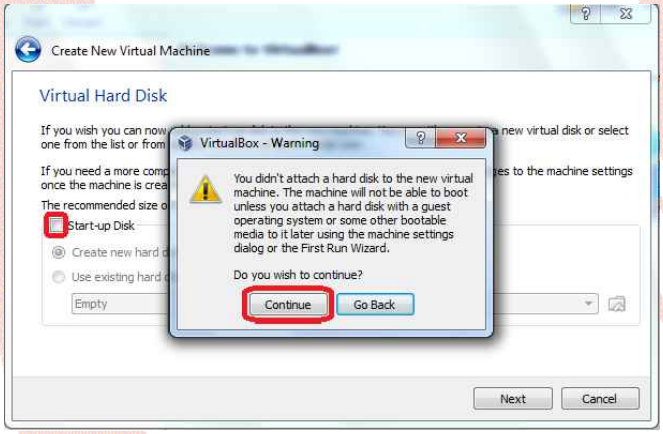
Whilst waiting for your Raspberry Pi, why not try the Linux Debian OS?

Download the Debian LiveCD ISO file:

For Intel Processors (770 MB): http://goo.gl/cOLBP
For AMD Processors (770 MB): http://goo.gl/xPXvs

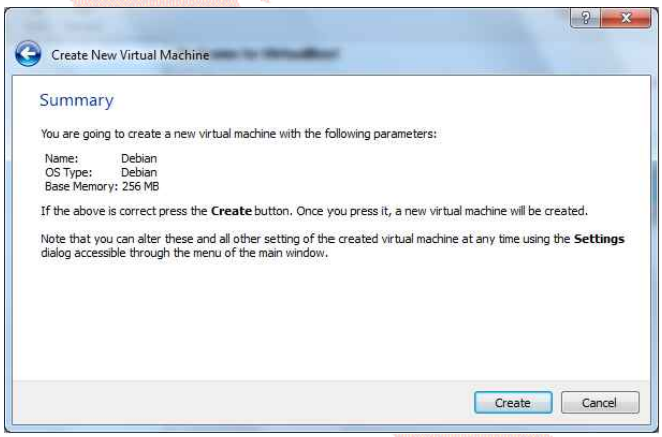
Then download and install Oracle VM VirtualBox:

Use 'Windows Hosts' Version (92 MB): http://goo.gl/gLmjf

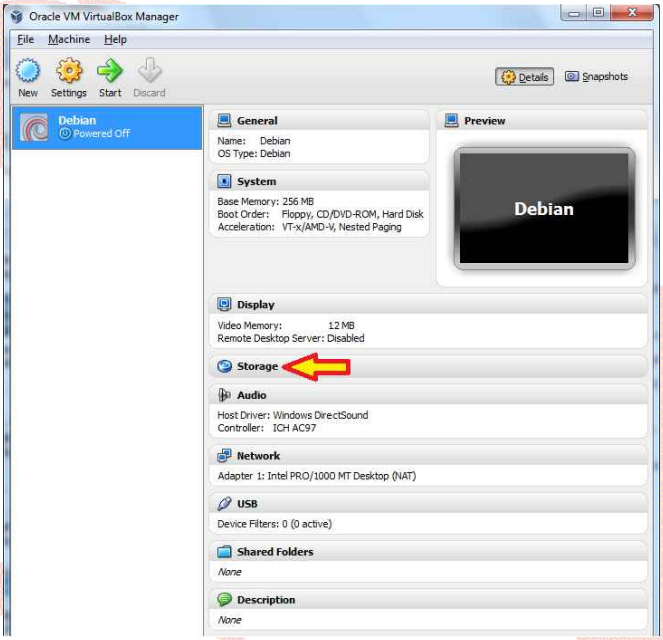


6. Untick Start-up Disk.

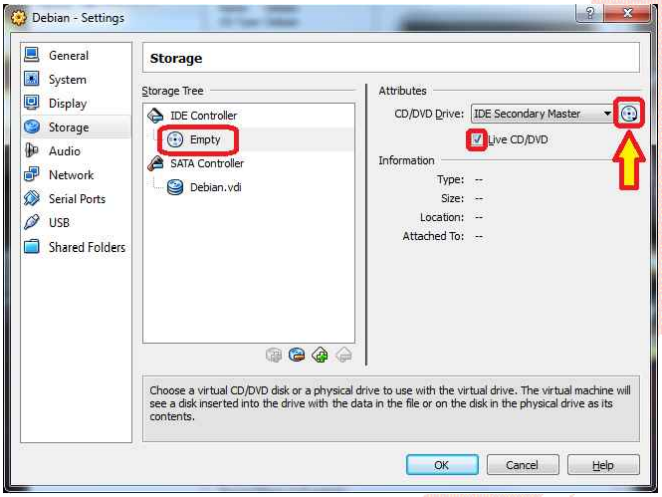
7. Click 'Continue' at the warning



8. Click 'Create'

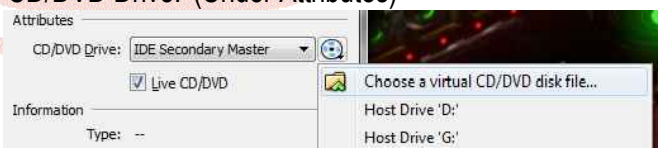


9. In the right hand column of the VirtualBox Manager, click on the word 'Storage'

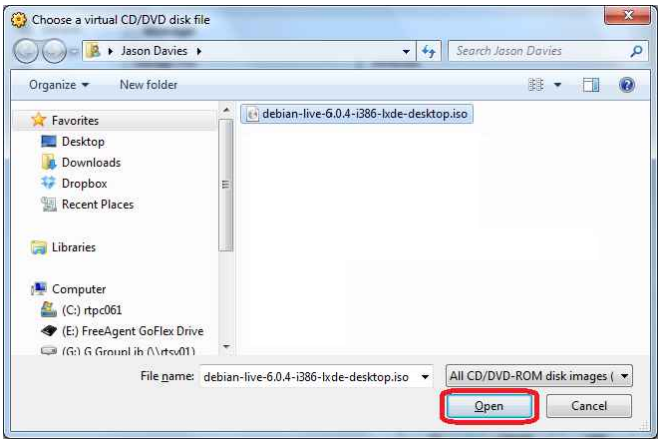


10. Under IDE Controller, click on the CD icon (Empty)

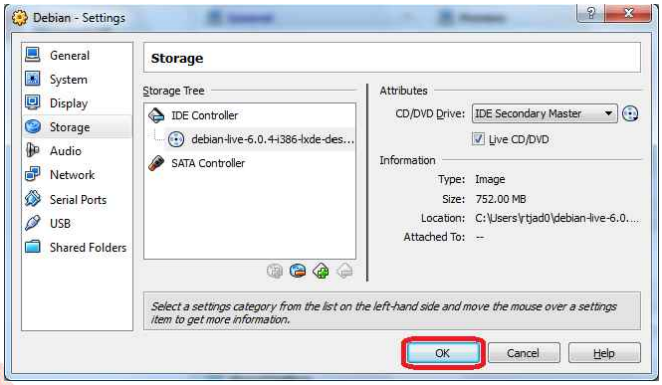
11. Tick Live CD/DVD and click the CD icon next to 'CD/DVD Drive:' (Under Attributes)



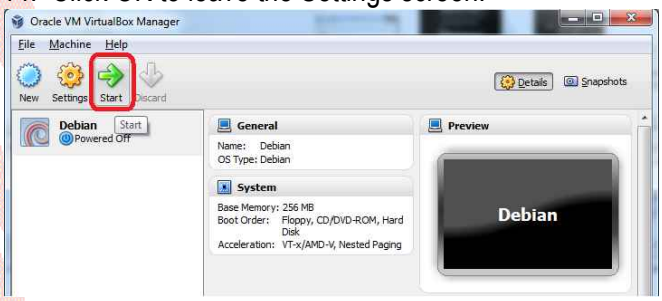
12. Click 'Choose a Virtual CD/DVD disk file...'



13. Browse for the downloaded Debian .iso file and click 'Open'



14. Click OK to leave the Settings screen.



15. Click 'Start' to boot the virtual machine

16. Press ENTER to choose 'Live' option

Wait a moment while Debian boots into the LXDE graphical interface. More to follow in issue 2.

Article by Jaseman

>Programming

Computer programming or 'coding' is a method of getting a computer to perform tasks, such as displaying and processing information, creating games and applications or controlling devices, motors or robots.

The computer can interpret our commands through programming languages, which help to make the processes of instructing the computer easier.

When we 'run' a computer program, the computer starts to carry out our instructions step by step and reacts to 'events'. We can tell the program how it should deal with certain events. For example, your program might be instructed to sense events such as a certain key being pressed on the keyboard, a click of the mouse on a button, and how it should react to these events.

The MagPi will show you how to get started writing your own programs and games, and providing examples and tutorials.

'The Scratch Patch' and 'The Python Pit' are the first programming sections of the magazine (See pages 20 onwards). We will be adding to this as things progress.

Article by Jaseman

'Scratch is a programming language that makes it easy to create your own interactive stories, animations, games, music, and art'

'Python is a remarkably powerful dynamic programming language that is used in a wide variety of application domains.'

'Pygame is a set of Python modules designed for writing games. Pygame adds functionality on top of the excellent SDL library. This allows you to create fully featured games and multimedia programs in the python language.'

'A geometry package providing for both graphical and algebraic input..'

'KidsRuby makes it fun and easy to learn how to program.'

'JavaScript is a prototype-based scripting language that is dynamic, weakly typed and has first-class functions. It is a multi-paradigm language, supporting object-oriented, imperative, and functional programming styles.'

'HyperText Markup Language is the main markup language for web pages.'

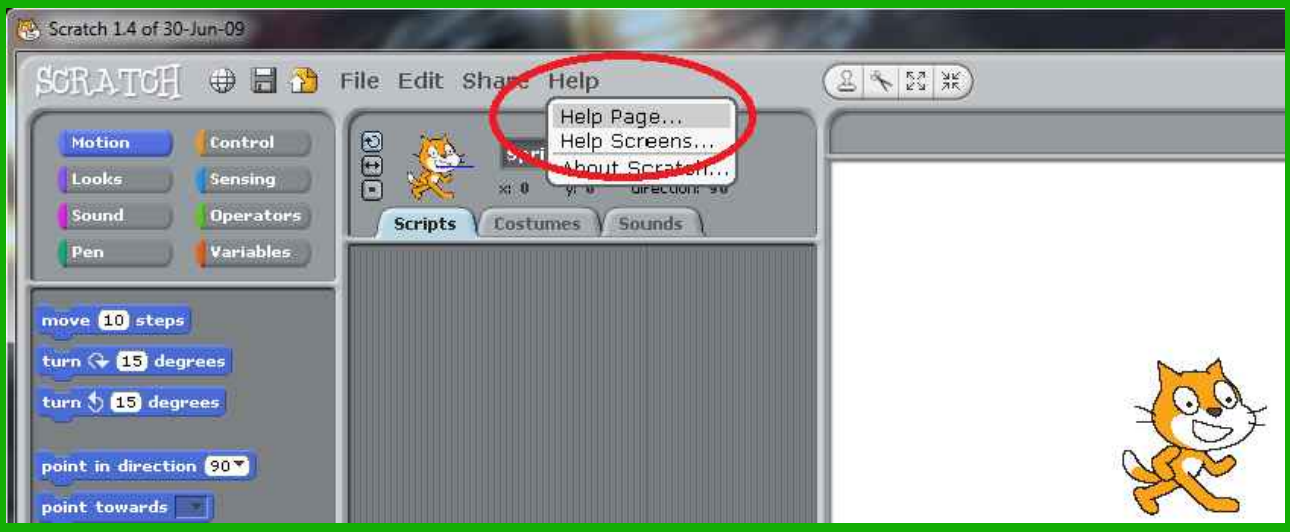
There are many other programming languages available to try. We will try to introduce some of them. Feel free to submit your own programming articles.

THE SCRATCH PATCH

Here at The MagPi, we highly recommend you to try out 'Scratch' - An easy to use programming language for beginners. If you don't already have Scratch, you can download it from the following website:

http://info.scratch.mit.edu/Scratch_1.4_Download

Once you have it installed and running, the best way to get started is to click on the Help pull-down and take a look at the Help Page and Help Screens. Most of the information you will need is already there. Spend some time familiarizing yourself with the scratch interface - It is very intuitive. Then try our Dizzy Cat / Crazy Drum tutorial on the following pages.

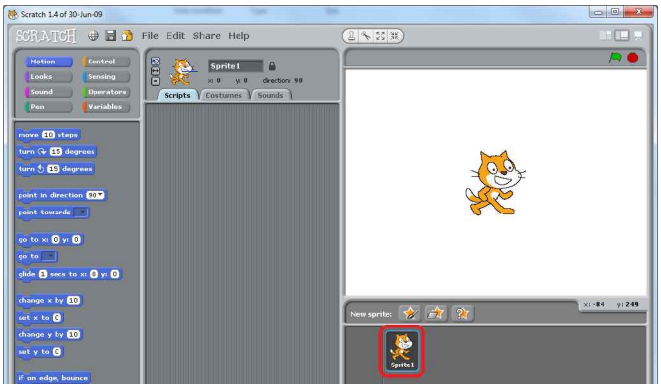


DIZZY CAT / CRAZY DRUMS

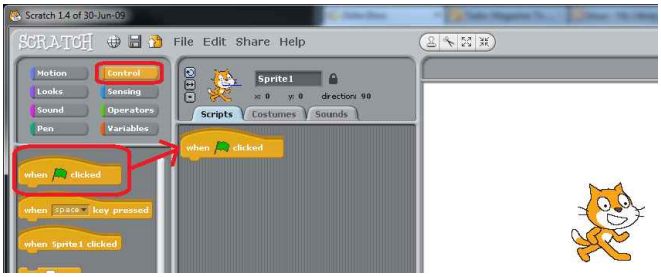
Here is a rather silly little scratch program for you to try:

1. Open Scratch

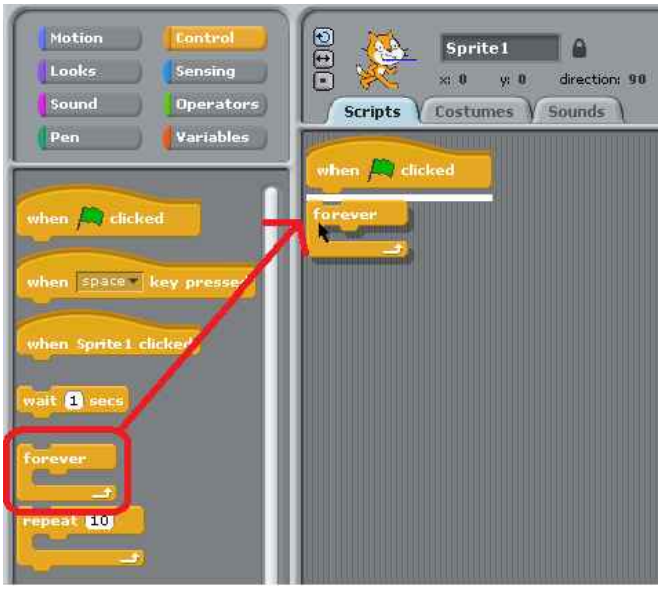
2. Click on the Scratch Cat Sprite 1



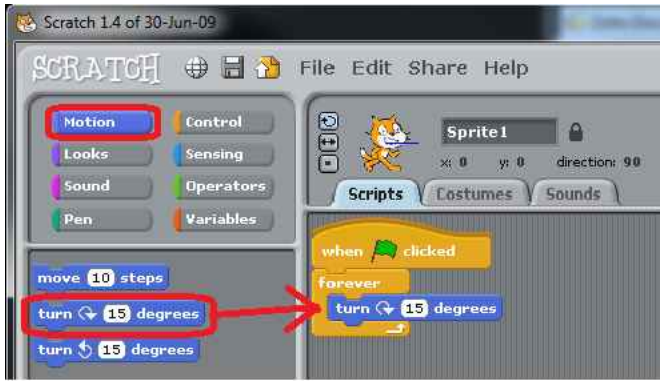
3. Click on Control and drag the 'when flag clicked' block to the Scripts area.



4. Drag the 'forever' block to the scripts area so that it locks in place under the 'when flag clicked' block.

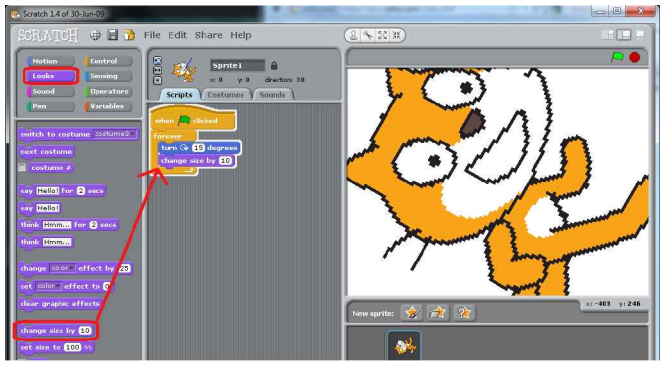


5. Click on 'Motion' and drag the 'turn 15 degrees' block into the middle of the 'forever' block.

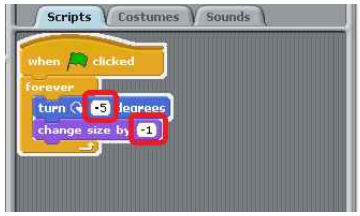


6. Click the green flag to start running the script, and see Scratch Cat spinning.

7. Click on Looks and drag the 'change size by 10' block under the 'turn 15 degrees' block (Also inside the 'forever' block. Watch how the cat grows.

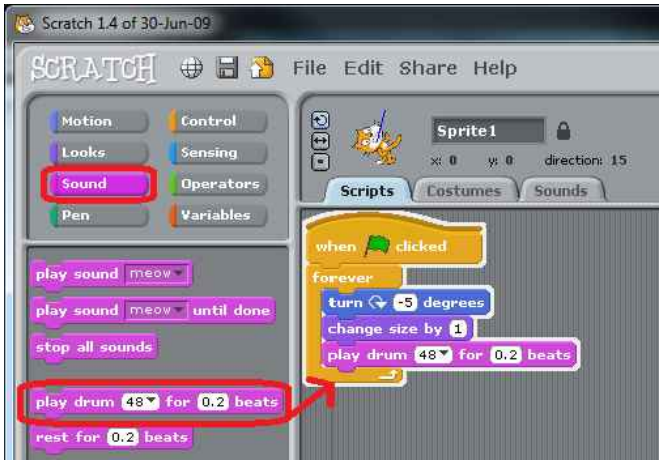


8. Change the '15' in 'turn 15 degrees' to '-5' and change the '10' in change size by 10' to '-1'. See how the cat starts spinning in the other direction and slowly starts to shrink.

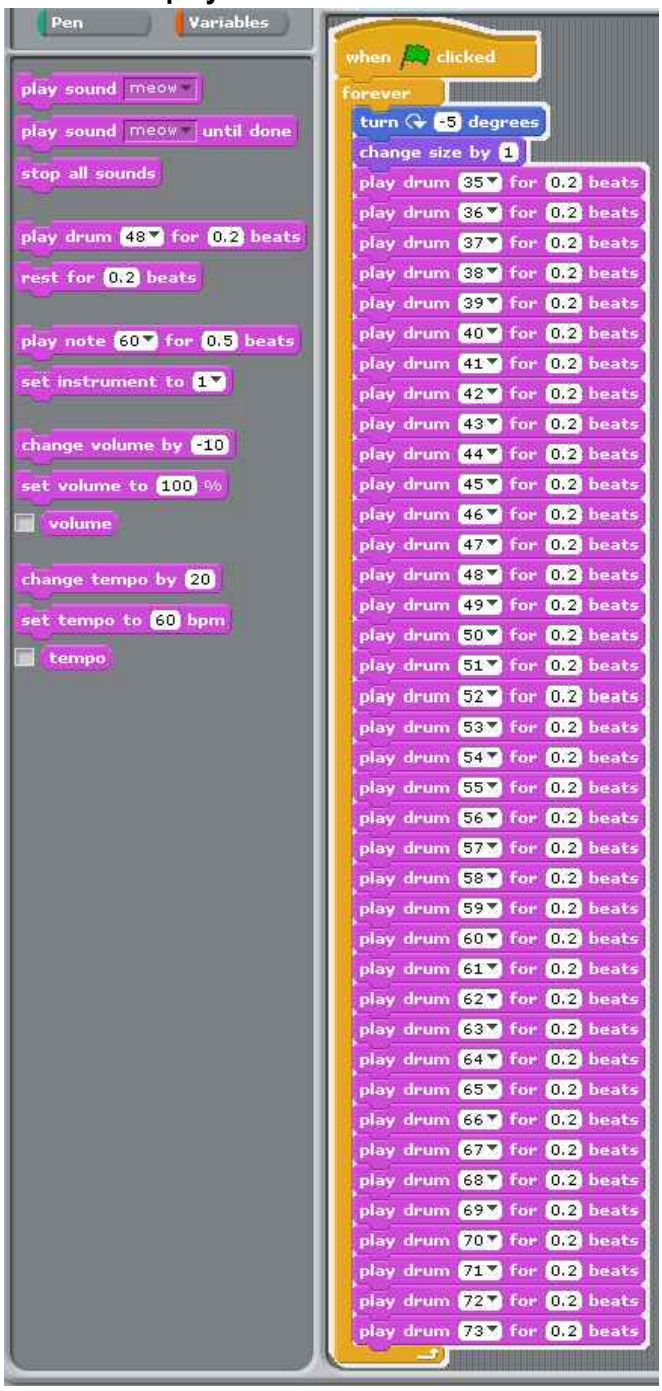


9. Make the cat grow again by changing the value of 'change size by' to '1'.

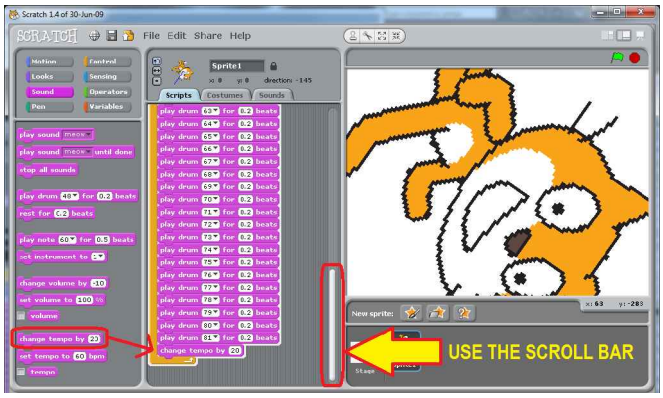
continued over page...



10. Click on Sound and drag the 'play drum 48 for 0.2 beats' into the 'forever' block. Then change the value of play drum from 48 to 35.



11. Keep adding more and more drums and change the number of each one so that the numbers go from 35 all the way up to 81. Listen to the different sounds each drum makes. As the scripts area fills up, you will have to use the scroll bar to the right to scroll up and down.



12. Drag the 'change tempo by 20' block into the 'forever' block. Wait and listen as the drums get faster and faster. When you can't take it anymore, change the value of 'change tempo by' to -20. After a little while, the drum tempo will slow down again - Phew!

13. Press the red spot which is next to the green flag to stop the terrible noise!

Poor cat!

By Jaseman

the python pit™

Here at the Python Pit, you will find Python and Pygame programs to type in. Each month we will bring you examples that will help to demonstrate some of the commands and programming methods to get you started writing your own code.

All of the programs are tested before the magazine goes live, and will have a stamp underneath showing which operating system and language versions were used for the testing.

NOTE: The examples in this issue are written for Python 3. They will work on older versions by making a change to all the print commands: Change print ("TEXT") to print "TEXT" - remove the ( ) brackets.

```
# PUTTING TEXT ONTO THE SCREEN
# By Jaseman - 23rd April 2012


print ("*** Welcome to The Python Pit! ***")
print () # A line space
print ("Brought to you by The MagPi")
```

The # hash symbol means 'comment' Python will ignore anything that is written after the hash - it is a way to add your own headings, notes or reminders to the program.

PYTHON VERSION: 2.6.6 / 2.7.2 / 3.2.2
PYGAME VERSION: N/A
O.S.: Debian / RacyPy / Win7

TESTED!

```
# ARITHMETIC
# By Jaseman - 23rd April 2012


print ("Two plus two equals ",2+2); print ()
print ("Eight minus three equals ",8-3); print ()
print ("Four times two equals ",4*2); print ()
print ("Ten divided by two equals ",10/2); print()
```

Notice how we can put two print commands onto the same line using a ; semi-colon. The second print command gives us a line space between each equation. You will also notice that the answer for the division sum includes a decimal place...

PYTHON VERSION: 2.6.6 / 2.7.2 / 3.2.2
PYGAME VERSION: N/A
O.S.: Debian / RacyPy / Win7

TESTED!

(continued over page...)

In this example we use the 'INT( )' function to remove the decimal places in the answer. The number is rounded to the nearest whole number. INT is short for Integer - Another word for whole number.

```
# SPARE ME THE DETAILS
# By Jaseman - 24th April 2012

print ("Seven divided by three is",7/3); print ()
print ("Seven divided by three is roughly",int(7/3)); print()
print ("Five divided by two is",5/2); print ()
print ("Five divided by two is roughly",int(5/2)); print()
```

Some Python commands only accept whole numbers (For example when referring to color values). Use 'INT( )' if you need to ensure that your result has no decimal places.

PYTHON VERSION: 2.6.6 / 2.7.2 / 3.2.2
PYGAME VERSION: N/A
O.S.: Debian / RacyPy / Win7

TESTED!

Using variables, we can get our program to remember numbers and words that it can use later on. You may remember doing this sort of thing in school.

```
# ALGEBRA
# By Jaseman - 24th April 2012

a = 3
b = 7
c = 4
print ("A is equal to",a); print ()
print ("B is equal to",b); print()
print ("C is equal to",c); print ()
print ("A plus B equals",a+b); print()
print ("A plus B plus C equals",a+b+c); print()
print ("A plus B minus C equals",a+b-c); print()
```

the python pit™

PYTHON VERSION: 2.6.6 / 2.7.2 / 3.2.2
PYGAME VERSION: N/A
O.S.: Debian / RacyPy / Win7

TESTED!

As well as numbers, variables can be words. We call these 'strings' because they are a string of alphanumeric characters strung (or grouped) together.

```
# STRINGS OF WORDS
# By Jaseman - 24th April 2012


a = "If you"
b = "notice"
c = "this"
d = "you will"
e = "is not worth"
f = "noticing!"


print (a,b,c,b,d,b,c,b,e,f); print ()
```

PYTHON VERSION: 2.6.6 / 2.7.2 / 3.2.2
PYGAME VERSION: N/A
O.S.: Debian / RacyPy / Win7

TESTED!

In Python there are many ways of counting. In this example we use a 'FOR LOOP'. The number (n) starts at zero and increases to 10. Note: We have to add a +1 onto the end otherwise the loop will stop at number 9. Try changing the number 10 in the program and run it again to see the results.

```
# COUNTING WITH FOR LOOPS
# By Jaseman - 24th April 2012


for n in range(0,10+1):
    print (n)
```

See how the print command is indented. The indentation is important as this shows Python that the print command is part of the FOR LOOP. Always indent the text as shown in these listings.

PYTHON VERSION: 2.6.6 / 2.7.2 / 3.2.2
PYGAME VERSION: N/A
O.S.: Debian / RacyPy / Win7

TESTED!

It is possible to count in steps too. Try changing the for command as follows:
for n in range(0,100+1,5):

This counts up to a hundred in increments of five. We can even do it in reverse:
for n in range(100,0-1,-5):

(continued over page...)

The WHILE LOOP continues until a certain criteria is met. In this program the number (n) starts off as zero. It keeps increasing by 1 for as long as the number is less than or equal to the value of ten (<= 10).

```
# COUNTING WITH WHILE LOOPS
# By Jaseman - 24th April 2012

n=0
while n <= 10:
        print (n)
        n += 1
```

There are some situations where WHILE LOOPS are more suitable than FOR LOOPS - Usually in less predictable circumstances.

PYTHON VERSION: 2.6.6 / 2.7.2 / 3.2.2
PYGAME VERSION: N/A
O.S.: Debian / RacyPy / Win7

TESTED!

To achieve the incremental steps in the counting, change the n += 1 to a higher number. Counting in reverse could be done like this:

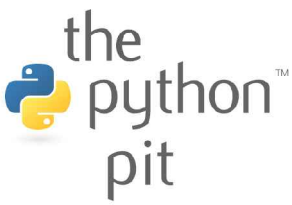
```
n=10
while n >= 0:
        print (n)
        n -= 1
```

```
# LOTTERY NUMBERS
# By Jaseman - 24th April 2012

import random

for n in range(1,6+1): # We want six random numbers
        print (random.randint(1,100)) # Picks a number between 1 and 100
```

We have to import the 'random' library to help Python to generate random numbers. The program makes use of a FOR LOOP so that the print command repeats six times.

PYTHON VERSION: 2.6.6 / 2.7.2 / 3.2.2
PYGAME VERSION: N/A
O.S.: Debian / RacyPy / Win7

TESTED!



If you want more random numbers - change the 6 to a higher number.

You could also change the range in the 'random.randint(,)' if you don't want it between 1 - 100

This time, the WHILE LOOP keeps generating random numbers until it finds a number '13'. The operator '!=" means 'not equal to', in other words keep looping while it's any number but thirteen.

```
# BINGO!
# By Jaseman - 24th April 2012

import random
n = 0

while n != 13: # Keep looping till you get thirteen
        n = random.randint(1,100)
        print (n)
```

PYTHON VERSION: 2.6.6 / 2.7.2 / 3.2.2
PYGAME VERSION: N/A
O.S.: Debian / RacyPy / Win7

TESTED!

Earlier we saw how words can be stored as string variables. This program creates something called a 'STRING ARRAY', which is a list of separate words stored in variables. Each word is allocated a number starting from zero, so the word 'I' is mytext[0], 'Love' is mytext[1], 'My' is mytext[2] and so on.

```
# STRING ARRAYS
# By Jaseman - 24th April 2012

mytext = ["I", "Love", "My","Raspberry","Pi"]

for n in range(0,5):
        print (mytext[n])

print()
```

PYTHON VERSION: 2.6.6 / 2.7.2 / 3.2.2
PYGAME VERSION: N/A
O.S.: Debian / RacyPy / Win7

TESTED!

(continued over page...)

This game will take a bit more typing, but it is a good example of what the Pygame library can do. Don't worry if you do not understand all of the commands. We will be covering these in more detail in the next issue. Note: You will need a mouse connected to your computer to control the bat.

```
# BAT AND BALL

# By antiloquax - 28th April 2012

import pygame
from pygame.locals import *
pygame.init()

# set the width and height of the screen
size = [400, 400]
screen = pygame.display.set_mode(size)

# give the window a title
pygame.display.set_caption("Bat and Ball")

# This makes the normal mouse pointer invisible in graphics window
pygame.mouse.set_visible(0)

# create surfaces for the bat and ball
bat_surf = pygame.Surface((64,12))
bat_surf.fill((0,255,0))
batrect = bat_surf.get_rect()
ball_surf = pygame.Surface((30,30))
ballrect = ball_surf.get_rect()

ball = pygame.draw.circle(ball_surf, (0,0,255),[15, 15], 15)

# set speed of ball
speed = [3, 3]

# puts the bat center of screen, near the bottom
batrect.center = ((size[0]/2), (size[1] - 50))

# make a text object
font = pygame.font.Font(None, 36)
text = font.render("Game Over", True, (255,0,0))
textRect = text.get_rect()
textRect.centerx = (size[0]/2)
textRect.centery = (size[1]/2)

# loop until the user clicks the close button
done=0

# create a timer to control how often the screen updates
clock = pygame.time.Clock()

# main game loop
while done == 0:

    screen.fill((0,0,0))

    # event handling
    for event in pygame.event.get(): # if we click something ...
        if event.type == pygame.QUIT: # if we click close ...
            done=1 # this will cause the loop to finish.
```

```
        # moves bat in accordance with the mouse position
        position = pygame.mouse.get_pos()
        batrect.centerx = position[0]

        # move the ball
        ballrect.left += speed[0]
        ballrect.top += speed[1]

        # collision detection
        if ballrect.colliderect(batrect):
              speed[1] = -speed[1]

        # check if the ball is going off screen
        if ballrect.left < 0 or ballrect.right > size[0]:
           speed[0] = -speed[0]
        if ballrect.top < 0:
           speed[1] = -speed[1]

        # print "Game Over" if the ball leaves screen
        if ballrect.top > size[1]:
           screen.blit(text, textRect)
           pygame.display.flip()
           pygame.time.wait(2000) # 2000 milliseconds pause
           ballrect.top=0; ballrect.left=(size[0]/2) # reset the ball position

        screen.blit(ball_surf, ballrect)
        screen.blit(bat_surf, batrect)


        # set the loop to 60 cycles per second
        clock.tick(60)

        #  update the display
        pygame.display.flip()
```

Make sure to observe UPPER and lower case text, for example:
.font.Font
.Surface
.Clock

Try altering the '# set speed of ball' section:

EG: speed = [5,5]

You can also alter the width and height of the screen.

Let us know how you get on with these programs. We will bring you some more Python and Pygame examples in Issue 2.

the python pit

PYTHON VERSION: 3.2.2
PYGAME VERSION: Pygame 1.9.2a0
O.S.: Win7

TESTED!

Feedback

'Magazine has to be done.'

SN

'I would love to see a magazine, but I'd also like to see it be a community led effort that is free for download.'

Abishur'

'Group effort PDF mag, but it would be nice to see one of those monthly online flash page turn magazines with pictures n stuff.'

monkeyra

'A mag would be great. Especially if there was plenty of programming tutorials/simple apps.'

fastkat

'The magazine is looking great. There is certainly a need for this sort of 'step by step' guide. Keep up the good work!'

Montala

'Hi Guys… I'm education coordinator for the Foundation and (at the moment) the only employee. The MagPi is a great idea!.'

Myra

'Exciting! We're very happy to see you guys doing this, and we're looking forward to seeing what you come up with.

Thanks for letting me know about it.'

Liz

The MagPi

Tandy have generously provided The MagPi with a Domain name.

A great big 'THANK YOU' from The MagPi Team.

Look out for some interesting news about Tandy in next month's issue.

editor@themagpi.com

Other Resources and Weblinks

http://www.themagpi.com
Official website of The MagPi magazine.

http://www.raspberrypi.org
Official home of the Raspberry Pi Foundation.

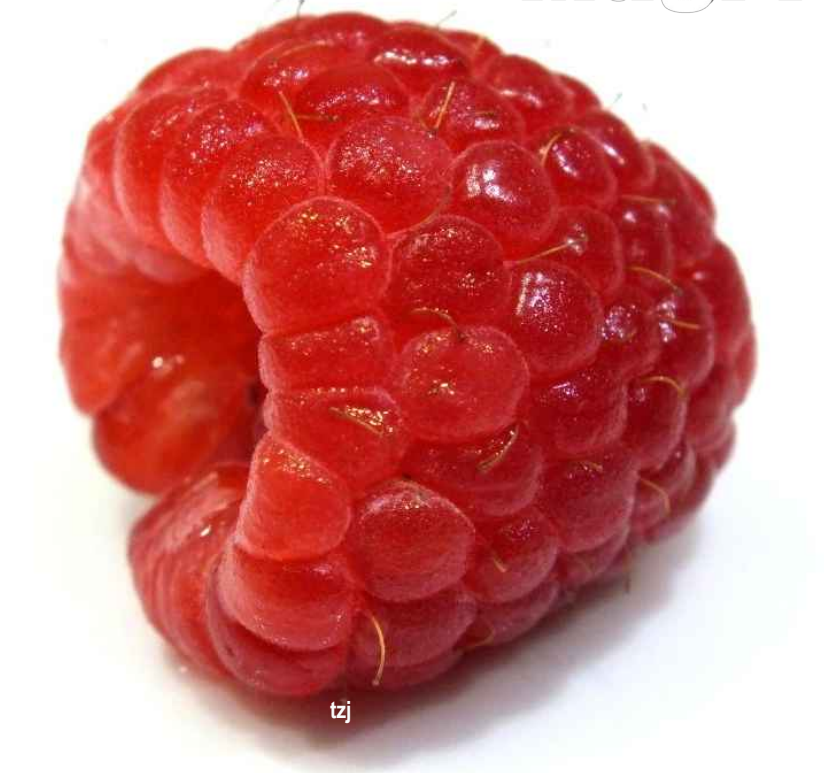
http://elinux.org/R-Pi_Hub
Wiki hub for the Raspberry Pi computer.

http://www.youtube.com/RaspberryPiTutorials
Video tutorials and python / pygame programming examples by Liam Fraser.

http://www.raspberrypi.org/forum/educational-applications/jasemans-python-lessons
Python / Pygame tutorials by Jaseman.

http://www.python.org
Official site for the Python programming language.

http://www.debian.org
Homepage for the Linux Debian Operating System.


tzj

Team:

Ash Stone
Chief Editor / Administrator / Website / Header

Jason 'Jaseman' Davies
Page Designs / Writer

Meltwater
Writer / Editor / Photographer

Chris 'tzj' Stagg
Photographer / Graphics

Bodge N Hackitt
Writer

antiloquax
Python Tutor

Additional help and support:

GizmoB73, Darren Grant, Mike Clements, Russell Davis, Rudi Kuin, Deep Thought, Liz Upton & Myra