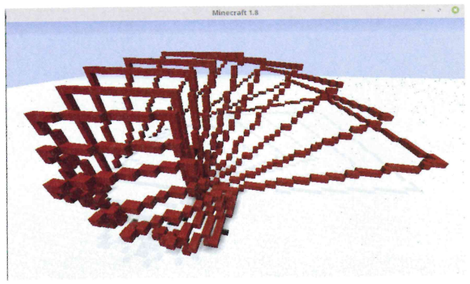
Draw 3D text in Minecraft using Python code

Last issue we hooked directly into Minecraft with Python code. This time we're using Turtle Graphics to draw 3D text

Calvin Robinson is head of computing at an all-through state school. Calvin also works with schools all over London as a computing consultant in education, helping create high-standard computing specs.



What you'll need

- Minecraft **www.mojang.com/games**
- Python **www.python.org**
- McPiFoMo **http://rogerthat.co.uk/McPiFoMo.rar**
- Minecraft Turtle **http://bit.ly/MinecraftTurtle**

Minecraft's 'Creative Mode' is indeed a great mode to get creative with, but it's a time-consuming job placing blocks down into your world one-by-one. That's why we put together a package which incorporates a range of modifications that enable us to execute Python scripts directly into a Minecraft world. McPiFoMo includes MCPiPy by 'fleap' and 'bluepillRabbit' of **https://mcpipy.wordpress.com**, and the Raspberry Jam Mod, developed by Alexander Pruss. We'll also be using Minecraft Turtle, which was put together by Martin O'Hanlon of **http://stuffaboutcode.com**.

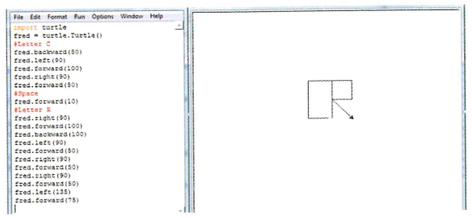
That's a lot of mods, but together they allow us to do with Minecraft on Linux what would usually have only been possible with the Raspberry Pi edition of Minecraft.

As a prerequisite, we assume you've installed McPiFoMo, from last issue's tutorial.

01 Minecraft Turtle Library

Turtle Graphics are bundled with Python, but to get this kind of functionality in Minecraft we'll need to download a custom Minecraft Turtle Library:

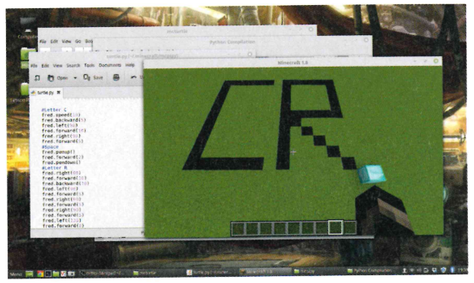
```
cd ~
git clone https://github.com/martinohanlon/minecraft-turtle.git
```



To install it:

```
cd ~/minecraft-turtle
python setup.py install
python3 setup.py install
```

If you've used Turtle Graphics before, you'll have no problem getting to grips with this incarnation.



Above Automate your in-game vandalism/signature with a simple Python script

02 Getting to know Turtle Graphics

Imagine you stuck a paintbrush in the mouth of a real turtle; everywhere that turtle moves, it drags the brush along, drawing a path. That's the basis of Turtle Graphics. You're drawing vector graphics with a relative cursor, across a virtual canvas. We direct the turtle where to go, and it leaves a trail of lines behind it. In Minecraft, those lines are represented by blocks.

03 Third dimension

Traditionally, Turtle Graphics programs can move forward and backward, but not necessarily left or right. We instead rotate left/right in degrees, and move forward/backward to draw our lines in the direction we need. Minecraft Turtle, however, has an additional dimension,

Action	Command	Example
Walk forward:	turtlename.forward(distance)	fred.forward(100)
Walk backward:	turtlename.backward(distance)	fred.backward(100)
Rotate left:	turtlename.left(angle)	fred.left(45)
Rotate right:	turtlename.right(angle)	fred.right(90)
Move up:	turtlename.up(distance)	fred.up(100)
Move down:	turtlename.down(distance)	fred.down(100)
Stop drawing:	turtlename.penup()	fred.penup()
Start drawing:	turtlename.pendown()	fred.pendown()

Above Here's a handy cheat sheet of all the basic commands used in Turtle Graphics

with up and down commands moving toward to the sky/ground accordingly. Pictured above is a list of the basic commands we'll be using to traverse our virtual canvas.

Above Using up/down commands adds another dimension to your lettering

04 Test your turtle

Create a new Python script in the IDLE and input:

```
from mcturtle import minecraftturtle
from mcpi import minecraft
from mcpi import block

#Connect to Minecraft and find the player's current position
mc = minecraft.Minecraft.create()
pos = mc.player.getPos()

#Spawn a new turtle, giving the variable a name of your choice
fred = minecraftturtle.MinecraftTurtle(mc, pos)

#Test your turtle
fred.forward(10)
fred.backward(20)
```

05 Draw your initials

Now we're going to draw out our initials. Some trial and error will be needed. Here's how we drew 'CR':

```
#Draw a sharp letter C
fred.backward(50)
fred.left(90)
fred.forward(100)
fred.right(90)
fred.forward(50)

#Provide a little space, by picking up the pen and moving along
fred.penup()
fred.forward(30)
fred.pendown()

#Draw the letter R
fred.right(90)
fred.forward(100)
fred.backward(100)
fred.left(90)
fred.forward(50)
fred.right(90)
fred.forward(50)
fred.right(90)
fred.forward(50)
fred.left(135)
fred.forward(75)
```

Above Diamonds are forever. Make your custom signature pop by using fancy material blocks

06 Spruce up your turtle

You'll likely want to change the type of block your turtle is drawing with: `fred.penblock(block.DIRT.id)` – where DIRT is a Minecraft block type; others include WOOD, GRASS, WOOL, TNT and DIAMOND.

To check if your pen (or paintbrush) is up or down at any given time, use: `print fred.isdown()`.

You can also change the speed of your turtle, 1 being a turtle, 10 being a hare: `fred.speed(10)`. ■

Position your 3D initials

You may want to change the location of your turtle, before or during the drawing process. To do this, we can use `print` or set commands to alter the position with Minecraft coordinates (x,y,z), just as you would to teleport another player around your world. Pairing these commands with `penup()` and `pendown()` allows you to keep your writing neat, kerning your letters when necessary.

```
#Print your turtle's position
turtlePos = fred.position
print(turtlePos.x)
print(turtlePos.y)
print(turtlePos.z)

#Reassign your turtle's position
fred.setposition(0,0,0)
fred.setx(0)
fred.sety(0)
fred.setz(0)
```

There's also a handy shortcut to send your turtle immediately back to the location it started in: `fred.home()`.