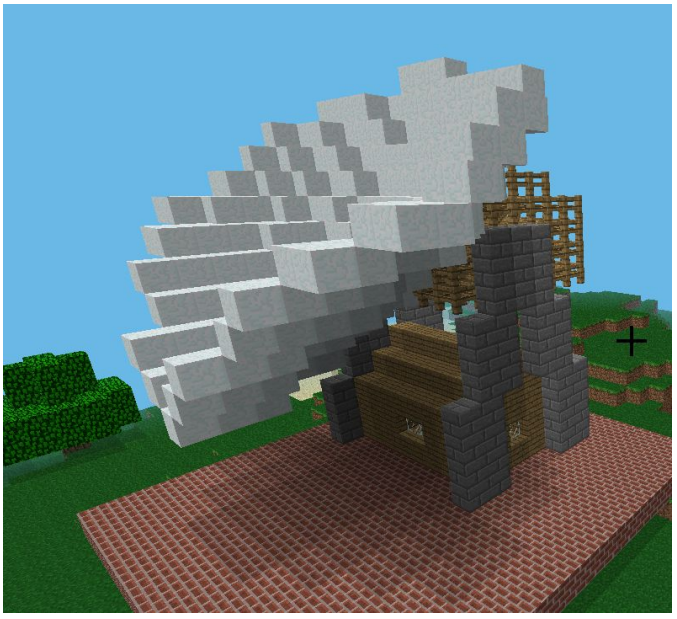
Goonhilly’s Arthur Satellite Dish

In 1962 this satellite dish received the first ever trans-atlantic television signal from the USA to Europe via the Telstar satellite. Still in use today the dish is fully trackable being able to rotate around it’s base at 120 degrees per minute. The ability to track across the sky means that Arthur is suitable to become part of the radio telescope e-Merlin network of telescopes with Jodrell Bank in Cheshire.



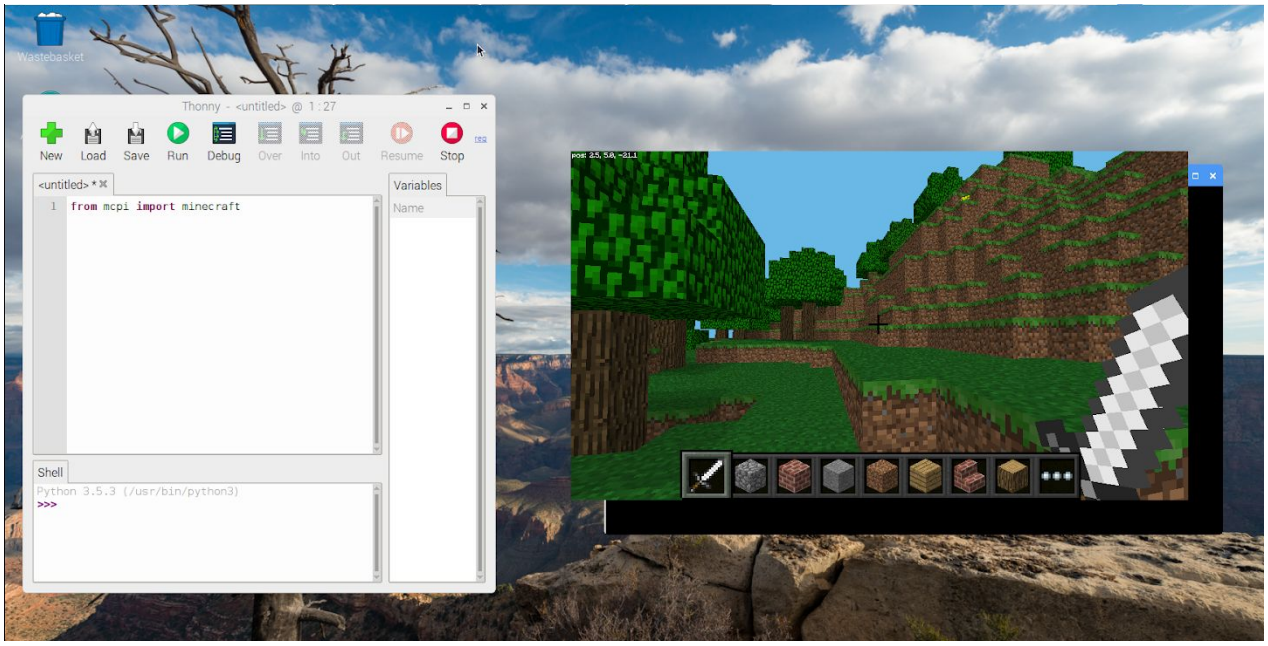
In this exercise you will be recreating Arthur in Minecraft. The model will be almost to the same scale as the real thing and normally would take quite a while to build. But with the power of code you can recreate the model above really quickly once written. In fact you could fill a Minecraft world with Arthurs if you so wished.

Coding Environment

The coding will use Thonny. To open this use the Menu on the left of the screen and under programming click on Thonny.

Menu > Programming > Thonny Python IDE

Open up a new file. A blank page will now open up. This is where the code will be typed and saved and then corrected and amended. Resize the windows so that half the screen width has both windows visible.



Now open Minecraft. Menu > Games > Minecraft. Resize the window to match the view above. Then ‘Start Game’ and ‘Create New’ world. This version of Minecraft is special to the Raspberry Pi. It is in creative mode only and of a limited size. But you can do all kinds of things inside it with code, as you will soon see.

Movement is by the PC type controls. W, A, S, D and the mouse to look around. Jump with Space, fly with double space and down with Shift. The inventory is with E, select a block to close it. ESC leaves the game but **Tab leaves the game running while you regain control of the computer**. Remove blocks with left click, place with right click. Have a quick practice.

Creating a Flat World

There is not an option to create a flat world to build things in this version of Minecraft but we can create one with code.

Click in the blank coding area and then Save the file calling it **clear_world.py inside Goonhilly_Arthur folder**. **This is important** as there is another file in this folder that you will need later.

File > Save > Goonhilly_Arthur > clear_world.py > Save

In the coding area carefully type in the following code.

```
from mcpi import minecraft
from mcpi import block

mc = minecraft.Minecraft.create()

mc.setBlocks(-200, -1, -200, 200, -50, 200, block.GRASS.id)
mc.setBlocks(-200, 0, -200, 200, 50, 200, block.AIR.id)
```

Save this code with CTRL + S (or File > Save) and then run it with the green arrow. Your world should now flatten out to a grass plain after a few seconds.

Did the Code Not Work?

Time to debug your code. Look at the message in the Shell below the coding text box it might give you a clue. Capital letters are important in Python. Third line has lower case m and upper case M in it. It also ends with () two brackets. The capital B is also needed. If you still can't see the problem ask your neighbour and if you are really stuck ask for help. But the problem will be in the typing.

Start Building

So you have the basics of a program to build in Minecraft. The clear_world script you built a world with grass and air blocks replacing everything that was there. So let's **use that program for the base** of a new one.

Save As the clear_world.py as MyName_Goonhilly_Arthur.py.

File > Save As > MyName_Goonhilly_Arthur.py

In front of the two setBlocks line add a hash so they look like this:

```
# mc.setBlocks(-200, -1, -200, 200, -50, 200, block.GRASS.id)
# mc.setBlocks(-200, 0, -200, 200, 50, 200, block.AIR.id)
```

The hash makes anything following it a comment that is ignored by Python when the program is run. **Comments are important in programming** as they enable the human reader understand what the code is going to do. Or as a way of leaving a message of something that needs to be added later. This is what you will now add to your script. **Add** the following code **highlighted in blue**.

```
from mcpi import minecraft
from mcpi import block

# start a link into the game calling it mc (short for minecraft)
mc = minecraft.Minecraft.create()

# find the player position

# place a block under the player feet

# These blocks will be used to make the dish base later
# mc.setBlocks(-200, 0, -200, 200, -5, 200, block.GRASS.id)
# mc.setBlocks(-200, 0, -200, 200, -5, 200, block.AIR.id)

# build the dish
```

The Minecraft world is quite large and if the dish is built out of view it could be difficult to find. So you will need to know where the player is standing so that the dish is built close by. The coordinates of the player are printed on the game screen in the top left. But they can also be read into the game and used.

Add under the '# find player position' the following code which finds the position of the block the player is standing on. The three elements of the position will be called x, y and z.

```
x, y, z = mc.player.getTilePos()
```

Under the next hash tag line add the code to place a block under the player's feet.

```
mc.setBlock(x, y-1, z, block.LAPIS_LAZULI_BLOCK.id)
```

Save the code and run as before.

There should now be a block of Lapis Lazuli under the players feet. The new code works by getting those three numbers from the game and calling them x, y, and z. It is a bit mathy but they are just simple coordinates. X and z are left & right or back & forwards. Y is up and down. So the code to place a block uses the three coordinates and places the Lapis Lazuli one block below the player which is the y-1.

Did the Code Not Work?

Time to debug your code again. Look at the message in the Shell it might give you a clue. Capital letters are important in Python. The Lapis Lazuli name must be in all capital letters. And this time it is setBlock without the s at the end as only a single block is being placed. And don't forget the brackets.

The Satellite Dish Base

Go back to the Lapis Lazuli block on the ground and stand on it by placing the cross hairs directly down and on the block.

Alter the setBlocks lines and remove the hash and alter them to read:

```
mc.setBlocks(x-10, y, z-9, x+8, y+25, z-25, block.AIR.id)
mc.setBlocks(x-9, y-5, z-8, x+6, y, z-24, block.BRICK_BLOCK.id)
```

Save and run the code again. You should now have a big foundation block of bricks on the ground. The foundation has some depth to it (the y-5) so that when you build in a normal world on a slope the foundation should reach right down. And the block.AIR line clears the area of trees and things too close to the dish. Go and take a look to make sure then either return to your Lapis Lazuli block or find a clear patch of ground to continue building Arthur on top of the base.

Did the Code Not Work?

Look at the message in the Shell it might give you a clue. Did you remove the hash at the beginning of the line?

Place Arthur on Top of the Base

To build Arthur yourself would take too long by hand so we have pre-built it and copied the coordinates of all the blocks. There are 194 blocks to place and if you like you can take a look at the file containing them. The file is arthur_blocks.py.

To **your** code now **add at the top the following lines** to enable the code to slow down and another to read the arthur_blocks.py file.

```
from time import sleep
from arthur_blocks import dish_block_positions
```

And below the # build the dish line add this:

```
read_block_data = 0
```

This line sets a variable called read_block_data to zero. It could be called find_block_information = 0 and the code would still work. The name is not important just as long as it makes sense and the same name is used later.

Then add this looping code which reads each item in the arthur_blocks list mentioned before one by one and then every four pieces of data places a block using the data. It carries on reading the data until the value of read_block_data is no longer less than the length (len) of the file containing the data. Meaning when it runs out of data it stops. The first three pieces of data are the x, y and z coordinates and the last is the block identity.

Make sure to **indent the code after the while line**. The Thonny IDE should do it automatically as long as you do not forget the colon at the line end. The comments are for you to understand what is happening.

```
while read_block_data < len(dish_block_positions):

    x_position = dish_block_positions[read_block_data]   # read x position
    read_block_data += 1                          # add one to read_block_data

    y_position = dish_block_positions[read_block_data]   # read y position
      read_block_data += 1

    z_position = dish_block_positions[read_block_data]   # read z position
    read_block_data += 1

    block = dish_block_positions[read_block_data]        # read block ID
    read_block_data += 1

    mc.setBlock(x+x_position, y+y_position, z+z_position, block) # place block

    sleep(0.1)  # A slight pause
```

Save and run again. Arthur should now appear block by block.

If all goes well, why not put a dish in a real Minecraft world. Just find somewhere a bit flat. And while you are there put loads of them into the world!