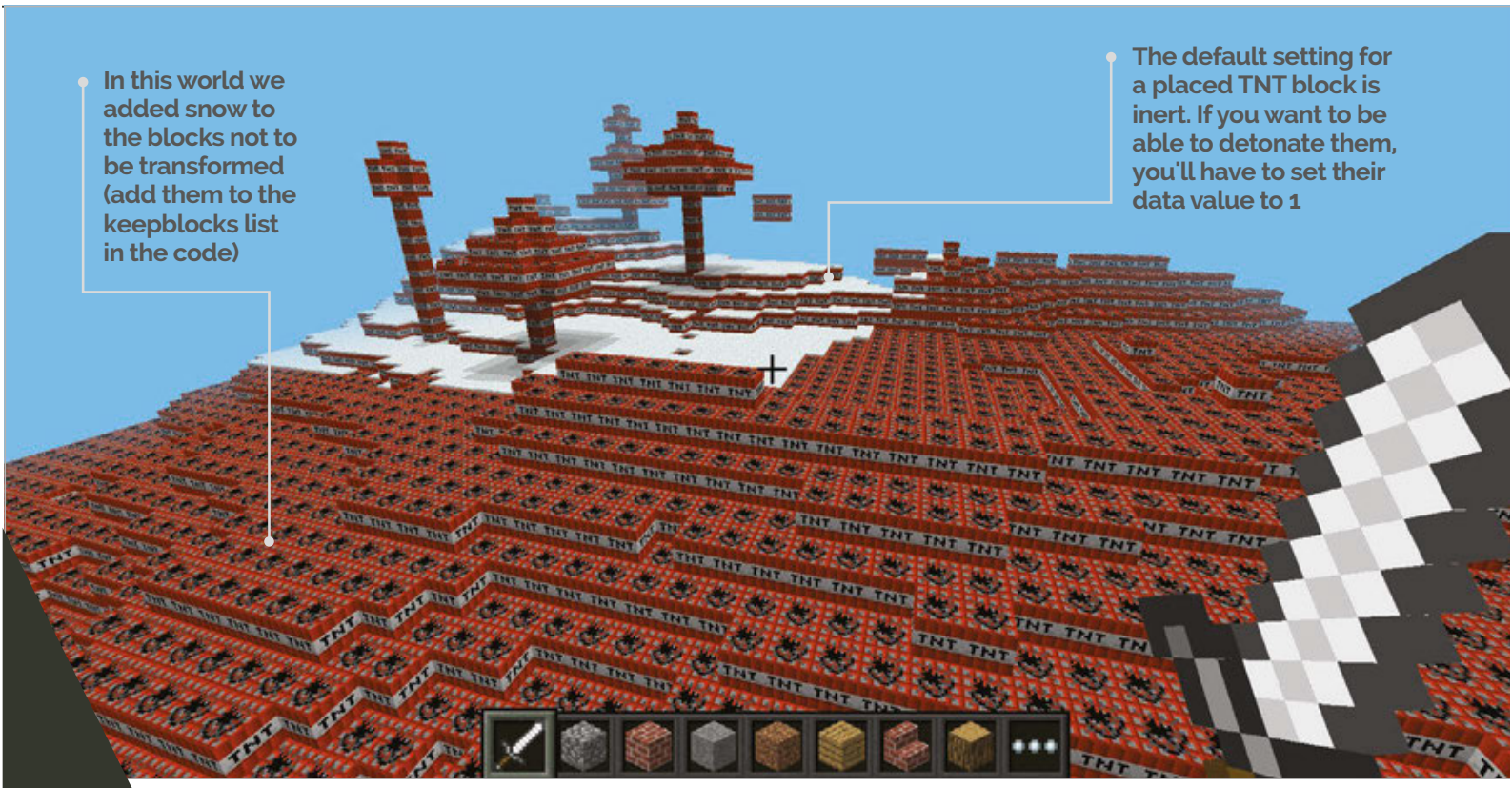
The MagPi ESSENTIALS

[CHAPTER SIX]

TERRAFORMING MINECRAFT

Everyone has their favourite Minecraft block.
What if you could have an entire world made out of them?



You'll Need

- Initial State account **initialstate.com**
- ISStreamer and Python 3 library
- psutil Python 3 library

Imagine fields of gold, fit for King Midas or the dragon Smaug. Or how about a frozen landscape where everything has been turned to ice? Just think what you could do in a world where everything is primed TNT.

Using Python, we can start a terraforming process to remake a Minecraft world to your specifications. Even on a Pi 3, this won't be a quick process: depending on how complex your landscape is, and how much you want to transform, it may take several days. So we'll monitor our progress by uploading data to an Initial State dashboard so that we can keep track of things remotely. If you just want to do the terraforming, there's another version of the code without the Initial State functionality in the same GitHub repository (**terraforming_no_is.py**).

>STEP-01

Generate your world

Before you start coding, you need to create your Minecraft: Pi Edition world and select the block type with which you want to fill your world. This can be any block of your choice, but it has to be a solid block (not ladders or torches). Manipulating the Minecraft ecosystem can be tricky. For example, if you try to turn water directly to lava, you'll probably end up with lakes of obsidian, so you might need an

intermediate step: turn all the water to something inert like wool, then transform it to lava. There may also be some blocks you want to keep – snow or water, for example.

>STEP-02

Get the code

Make sure your Pi is up to date and, if you want to create a remote monitoring dashboard on Initial State, download and install its data streaming library:

```
sudo pip3 install ISStreamer psutil
```

Then download the **is_terraforming.py** code (**magpi.cc/234A3hY**). Note that you'll need to change some of the values to suit your Minecraft environment and to include your Initial State account details.

>STEP-03

Tune the code

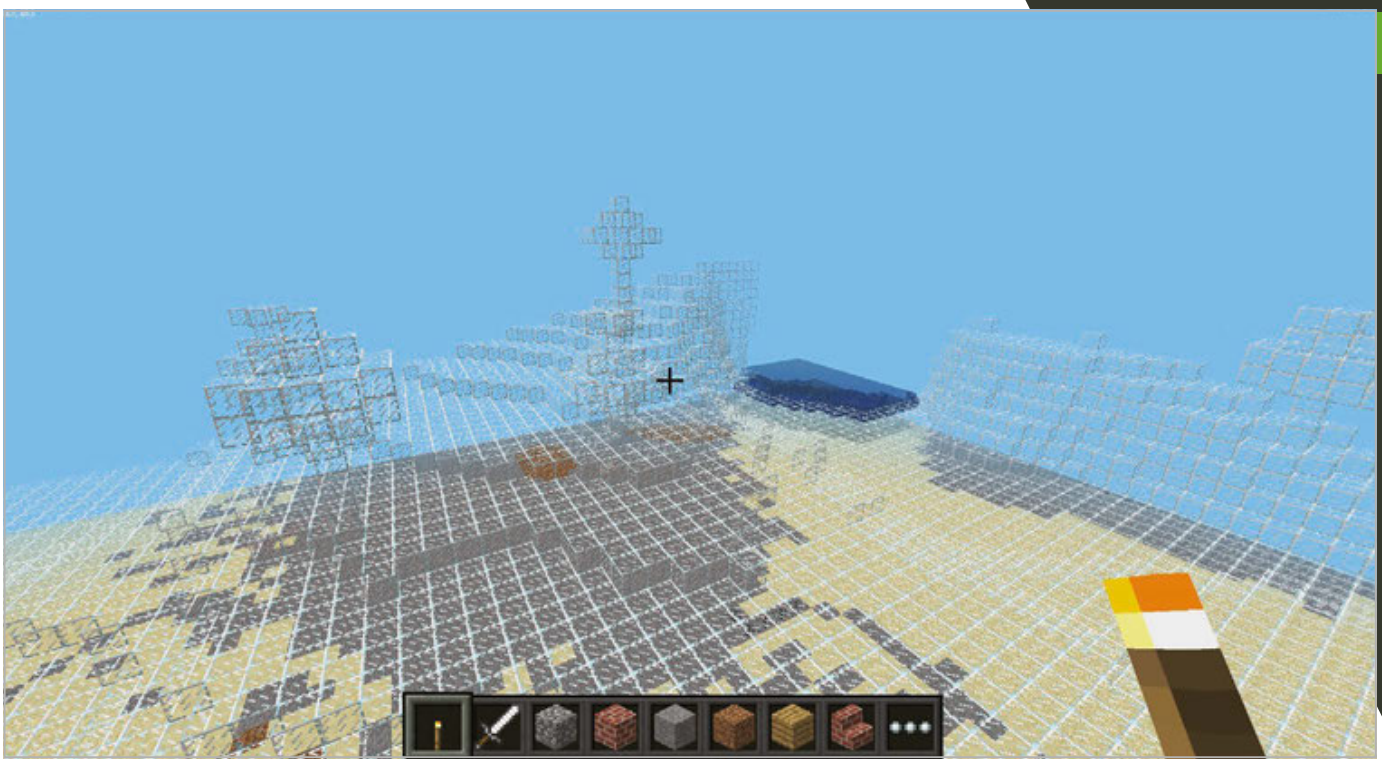
Terraforming can take a long time – we're talking days rather than minutes. However, we can tune our code to speed things up. Explore your world and find the tallest mountain range and deepest valley. Make a note of the height (the third value displayed in the top-left corner of the screen). You can then plug these values into the code.

We've set the default terraforming height range from -3 to 35 on the y axis, but you can make this bigger (this will take longer) or shorter (this will take less time), depending on the size of the geological features in your world.

>STEP-04

Set the speed for the power of your Pi

This code should work on any Pi, but older, less powerful models may struggle if you terraform at full speed. If Minecraft can't keep up with all the changes it's asked to make to the landscape, it may hang. So it's a good idea to pause after a certain number of blocks, to let Minecraft catch up. On a Pi 3, you can comfortably transform 500+ blocks before having to pause, but for a Model B you may need to deal with 50 blocks at a time. You'll probably want to run a few experiments to find the optimum configuration for your setup.



Above You can create some very strange-looking worlds, like this one where everything on the surface is made of glass

>STEP-05

Register for an Initial State account

Initial State allows you to upload live data and plot interesting charts and graphs. A free account lets you stream 25,000 events a month and examine the last 24 hours' worth of data in any bucket. Once you've registered for an account, click on the 'create HTTPS bucket' button (the plus symbol) and give it a suitable name. Then check 'Configure Endpoint Keys' and copy the Bucket Key and Access Key into your version of the code.

>STEP-06

Start terraforming!

If you're using a free account, edit the code and set the `Free_account` variable to True. This will throttle the amount of data sent to Initial State and allow you to record the whole process without exceeding the data cap.

Start your code running and check the console for any errors. You can fly to the corner of your world and should soon be able to see the changes taking place. Once the first data reaches Initial State, you can create a cool dashboard: use the Tiles interface and play around with the different types available.

is_terraforming.py

```
01. import mcpi.minecraft as minecraft # Load libraries
02. from ISStreamer.Streamer import Streamer
03. import mcpi.block as block
04. import time, datetime, psutil
05.
06. for pros in psutil.pids(): # Get the Linux process number for the Minecraft program
07.     if psutil.Process(pros).name() == 'minecraft-pi' and len(psutil.Process(pros).cmdline()) == 1:
08.         pm = psutil.Process(pros)
09. streamer=Streamer(
    bucket_name=":mushroom: Terraforming", bucket_key="<enter here>", access_key= "<eneter here>")
10. Free_account = False # If using a free IS account, set to True to limit data uploads and avoid
    exceeding monthly limit
11. # Function to upload various bits of data to IS
12. def upload_data_to_IS(
    speed,elapsed,blocks_processed, blocks_transformed,cpu,y,x,z,mem,pm,num_blocks):
13.     print('Uploading to Initial State')
14.     streamer.log(":snail: Run Speed",speed)
15.     streamer.log(":jack_o_lantern: Run2 Time since last "+ str(num_blocks) + "blocks",elapsed)
16.     streamer.log(":volcano: Run2 Total Blocks",blocks_processed)
17.     streamer.log(":chocolate_bar:Run2  Blocks transformed",blocks_transformed)
18.     streamer.log(":up: CPU %",cpu)
19.     streamer.log(":arrow_down: Y",y)
20.     streamer.log(":arrow_right: X",x)
21.     streamer.log(":arrow_left: Z",z)
22.     streamer.log(":question: Memory used %",mem.percent)
23.     streamer.log(":question: Minecraft Process memory used %",pm.memory_percent())
24.
25. time.sleep(1)
26. mc=minecraft.Minecraft.create() # Connect to Minecraft
27. keepblocks=[block.AIR.id,block.WATER.id,block.LAVA.id,block.SNOW.id,
                block.WATER_FLOWING.id,block.WATER_STATIONARY]
28. counter = 0 # A bunch of variables to keep track of how many blocks have been processed
29. blocks_processed = 0
30. blocks_transformed = 0
31. blocks_since = 0
32. throttle = 5 # Use this when Free_account is True, to restrict amount of data uploaded
33. num_blocks = 1000 # How many blocks to transform before pausing to let Minecraft catch up
```

Download
magpi.cc/
234A3hY

```
start = time.time()
for x in range(-128,128): # the x-direction
    for y in range(-4,35): # the y-direction (up/down)
        for z in range(-128,128): # the z-direction
            print(x,y,z)
            test = mc.getBlock(x,y,z) # Read a block at x, y, z
            blocks_processed+=1
            blocks_since+=1
            if test not in keepblocks: # Don't transform these blocks (should always contain AIR)
                counter+=1
                if counter > num_blocks:
                    blocks_transformed+=num_blocks
                    counter = 0
                    stop = time.time()
                    elapsed = stop - start # How long since last group of blocks were processed?
                    speed = blocks_since/elapsed # Calculate speed
                    cpu = psutil.cpu_percent() # Read CPU utilisation
                    mem = psutil.virtual_memory() # Read memory usage data
                    if Free_account: # Only bother to throttle if using free IS account
                        if throttle == 0:
                            upload_data_to_IS(
speed,elapsed,blocks_processed, blocks_transformed,cpu,y,x,z,mem,pm,num_blocks)
                            throttle = 5
                        else:
                            throttle-=1
                            print('reducing throttle')
                    else:
                        upload_data_to_IS(
speed,elapsed,blocks_processed, blocks_transformed,cpu,y,x,z,mem,pm, num_blocks)
                    time.sleep(5) # Pause to allow Minecraft to catch up
                    start = time.time()
                    blocks_since=0
                mc.setBlock(x,y,z,block.REDSTONE_ORE.id)

                print('Changing Block: ' + str(test) + ' (counter = ' + str(counter) + ')')
                time.sleep(0.1)
            else:
                print('Not changing Block: ' + str(test) + ' (counter = ' + str(counter) + ')')
```