Raspberry Pi®

CoderDojo Belfast

Project #7: Do Two Things at once

Level: Tricky Stuff

Suppose you wanted to add Project 1 and Project 3 together – write a teleporter, but instead of always teleporting to the position you were at when you start your program, you want to be able to use the magic sword to change the teleport destination. In other words whenever you hit the ground with the magic sword, that point becomes the teleport destination.

Coding skills needed

- Multi-threaded programming - tricky stuff! This is one of the hardest things to get right when writing code.
- But you can do it!

But there's a problem.

In Project 1, we used an infinite loop to watch where the player is

```
while True:
        time.sleep(0.3)
        p = mc.player.getTilePos()import
```

We call it an infinite loop because it just keeps on going as long as the program keeps running.

Then in Project 3, we used another infinite loop to watch the 'block hits' from the magic sword:

```
while True:
    hits = mc.events.pollBlockHits()
```

But if we want to add these together how are we going to do this? Think about it for a minute – how would you write a program that did two infinite loops like this at the same time? You'd have to stop one loop in order to do the other.

Well, you may not be surprised to hear that there is a way to do this. You use something called a Thread.

Weaving with Code

A Thread is a way to do something while the rest of your program is doing something else. Whatever it is you want to do, put it in a function (see Project 2) and then run that function using a thread.

To create a thread we use a standard bit of Python. We create a *class* (more on these in later Projects), and we tell Python that the class is a Thread. Here's our magic sword thread:

```
class SwordThread(Thread):
  # the __init__() method sets the thread up,
  # ready to run.
  def __init__(self):
    Thread.__init__(self)
    # the 'run' method contains the stuff
    # the thread actually does.
    def run(self):
      handle_events(set_start)
```

Whatever is inside the 'run' method is what the thread will do when it runs – you can even put an infinite loop in it and it will happily run away forever alongside the rest of your program. 'handle_events' does just that.

The complete program: teleport2.py

```
#!/usr/bin/env python

import mcpi.minecraft as minecraft
import mcpi.block as block
import time
from sword import handle_events
from threading import Thread

# Let's make a "magic spell" function that sets the teleport start pos.
def set_start(mc,hit):
	global startPos
	startPos = hit.pos
	co_ords = str(start.x) + ", " + str(start.y) + ", " + str(start.z)
	mc.postToChat("Teleport activated at " + co_ords)

# We'll use a thread to run the handle_events function (which contains
# a loop that doesn't finish, so needs its own thread to run in).
class SwordThread(Thread):
	# the __init__() method is used to set the thread up, ready to run.
	def __init__(self):
		Thread.__init__(self)
	# the 'run' method contains the stuff the thread actually does.
	def run(self):
		handle_events(set_start)

# OK let's start the program. Our start position is a global variable.
# One of these days we're going to have to learn how to do without these
# but one thing at a time.
startPos = None

# To use our thread we have to first create it, by the call below...
sword_thread = SwordThread()
# ... and to get it going we have to call its 'start' function.
sword_thread.start()

# The rest is just the same as in Project 1.
mc = minecraft.Minecraft.create()
while True:
	time.sleep(0.3)
	p = mc.player.getTilePos()
	block_below = mc.getBlock(p.x, p.y - 1, p.z)

	if block_below == block.GLASS.id :
		# teleport!
		print("Teleporting")
		mc.setBlock(p.x, p.y, p.z, block.GLOWING_OBSIDIAN)
		mc.setBlock(p.x, p.y+1, p.z, block.GLOWING_OBSIDIAN)
		time.sleep(1)              # give us time to see it!
		mc.player.setPos(start.x, start.y, start.z)   # jump
		# now clear the teleport again
		mc.setBlock(p.x, p.y, p.z, block.AIR)
		mc.setBlock(p.x, p.y+1, p.z, block.AIR)
```

Dojo Challenge:

What else could you do in a separate thread? - How about keeping a score of how far you've walked?