nustem

PyGame Zero

Animate graphics, connect GPIOzero controls, and don't panic about large lumps of code.

Ingredients

Platform:	Raspberry Pi / Raspbian Python3
Components:	Button
Libraries:	PyGameZero GPIOzero

Glossary

ADC	"Analogue Digital Converter" – a circuit which measures an analogue signal and outputs a digital representation of it.

First steps

In Thonny, open `/Digital-Making/examples/pgzero/pgzerotest.py`, and run it.

After a few seconds, you should see a small window with a red background, and a cute little alien graphic moving across it. If you click on the alien, he looks a little unhappy for a few seconds.

Best game ever, right?

Have a look through the code. It should make some sense, but you'll likely have questions. See if you can work out what's going on – this is a good exercise for practising looking through someone else's code for bits which might be useful, and ignoring anything that seems irrelevant or plain too hard.

Some hints:

- Most of the code is functions (the bits which start `def …`) which are called when different events happen. For example, `set_alien_hurt()` is called on line 25.
- What calls `on_mouse_down(pos)`, and what does `pos` represent?
- It's not obvious what calls `update()` and `draw()`. PyGame Zero is hiding a lot of detail from us – they must be called by something happening in `pgzrun.go()`.

PyGame is a large library intended to make building games relatively straight-forward. PyGameZero adds a layer of assumptions and conventions on top of PyGame, which in principle makes it easier to use.

We've used PyGame to display full-screen graphics with (reasonably) high performance, and to use keyboards as multi-button controllers.

Flappy Bird

When you're ready to move on, close the red alien window, then in Thonny open `flappybird/flappybird.py`. This is one of the PyGame Zero example games – run it!

You can use any key to flap.

Northumbria University NEWCASTLE

Challenge: arcade button controller

You'll notice an arcade button connected to the Pi. Let's use it to play the game.

The script `button.py` gives you an example of using GPIOzero to handle button press events. The line:

```
button.when_pressed = button_pressed
```

...sets up a GPIOzero 'handler' to listen for the button being pressed. When that event happens, the function `button_pressed` is called. You could have the button call any function you like.

You'll notice that the PyGameZero code is similarly **event-driven**: when `update()` is called (every 60th of a second), it calls `update_pipes()` and `update_bird()`. `on_key_down()` is called when there's a keypress.

So... how would you add in the arcade button, to work alongside the keyboard?

Start with the `flappybutton.py` script, which sets up the button object for you, and refer back to `button.py`. See if you can puzzle it out. There are a couple of broad hints in `flappybutton.py`.

Key lesson

The important part of all this is: you don't need to follow all the details of some-one else's code to be able to hack on it, modify it, and adapt it to your needs.

The secondary lesson is that it would be really useful if example code was com-mented properly.