Raspberry Pi Projects

Pixel pet

Introduction

Using sensors and output devices is a great way to make your computer programs more interactive. The Raspberry Pi Sense HAT contains a whole set of sensors that can be used to detect movement. In this resource, you will use these to take a digital pet for a walk. You can watch a video tutorial for the project here (https://www.youtube.com/watch?v=gfRDFvEVz-w&rel=0).

What you will learn

By creating your very own interactive pixel pet with code you will learn how to:

- Use variables and lists in Python to display an image on the Sense HAT LED matrix
- Use for loops and while loops to repeat instructions
- Create a function in Python
- Use the Sense HAT's accelerometer to detect movement and trigger some code

This resource covers elements from the following strands of the Raspberry Pi Digital Making Curriculum (https://www.raspberrypi.org/curriculum/):

- Design basic 2D and 3D assets (https://www.raspberrypi.org/curriculum/design/creator)
- Apply basic programming constructs to solve a problem (https://www.raspberrypi.org/curriculum/programming/builder)
- Combine inputs and/or outputs to create projects or solve a problem (https://www.raspberrypi.org/curriculum/physical-computing/builder)

The finished project

What you will need

Hardware

- A Raspberry Pi
- A Sense HAT

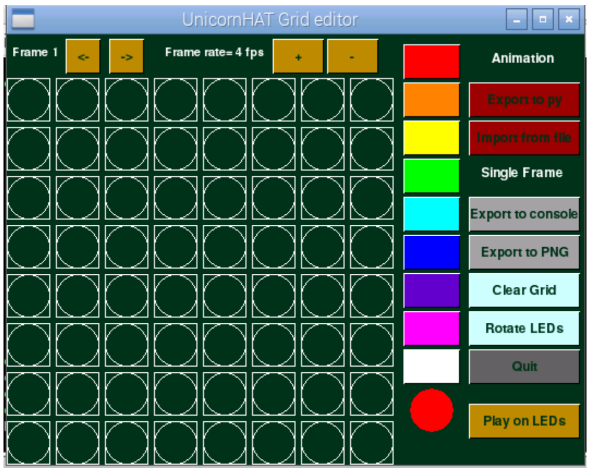
Software

All necessary software is pre-installed on Raspbian.

Draw your pixel pet

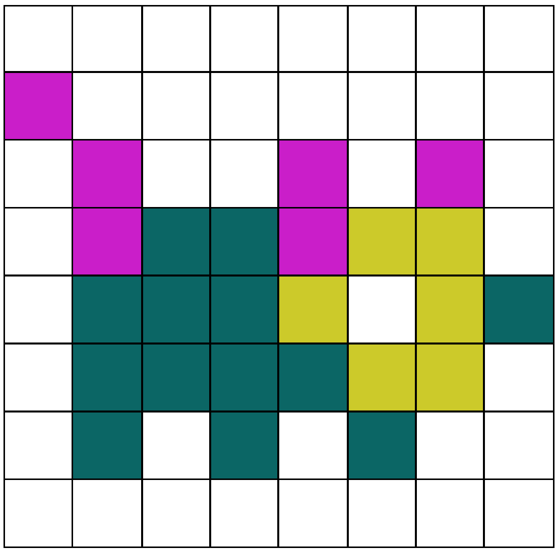
First you'll need to design your pet avatar. Here are some examples (https://www.youtube.com/watch?v=PpHFQXoISWc&rel=0) to give you an idea! You can also print this worksheet (https://projects-static.raspberrypi.org/projects/pixel-pet/14fe1a101a714ac0ebca8f642d445388540425ee/en/resources/printable-worksheet.pdf) to help you design your pixel pet.

1. Open a Terminal by clicking on **Menu**, **Accessories**, and then **Terminal**.
2. Type `cd RPi_8x8GridDraw` and press **Enter**.
3. Next, type `python3 sense_grid.py`. This will run an application which you can use to draw your space pet avatar.

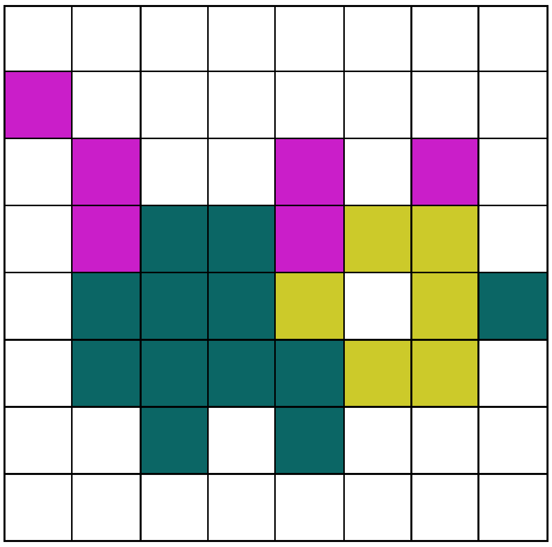


4. Simply select the colour you wish to use from the grid with your mouse pointer, and then select a circle in the grid to change it to that colour.

5. Alternatively, you may wish to draw your picture out on squared paper with coloured pencils, like this:



6. You will need another pet design, preferably one that is very similar to the first, so that we can animate your pet. The image below is almost identical to the one above, but the feet are in a different position:



Later, when you code your animation, you will use these two designs to create the illusion that the pet is walking.

Setting your colours

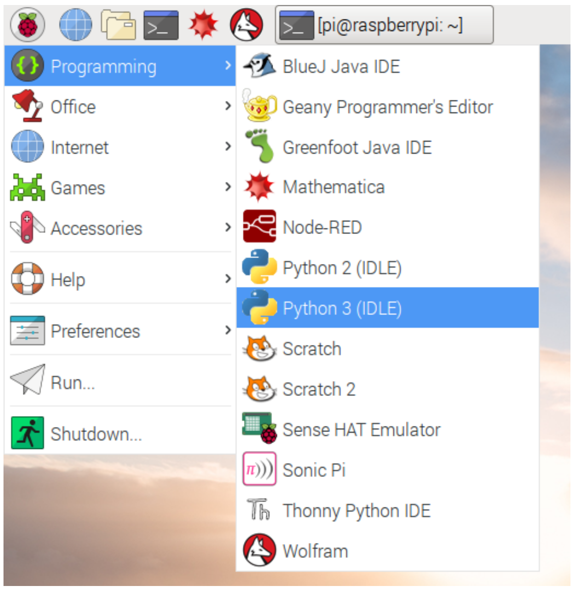
Now that you have your designs represented as letters in a grid or array, you can start to code them in Python.

- Open IDLE and create a new Python file.

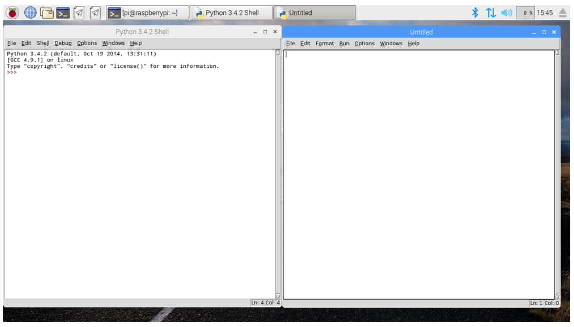
Opening IDLE3

IDLE is Python's **I**ntegrated **D**eve**l**opment **E**nvironment, which you can use to write and run code.

To open IDLE, go to the menu and choose `Programming`.
You should see two versions of IDLE - make sure you click on the one that says `Python 3 (IDLE)`.



To create a new file in IDLE, you can click on `File` and then `New File` in IDLE's menu bar.
This will open a second window in which you can write your code.


[pi@raspberrypi: ~]
Python 3.4.2 Shell
Untitled
15:45
Python 3.4.2 Shell
File Edit Shell Debug Options Windows Help
Untitled
File Edit Format Run Options Windows Help

```
Python 3.4.2 (default, Oct 19 2014, 13:31:11)
[GCC 4.9.1] on linux
Type "copyright", "credits" or "license()" for more information.
>>>
```

- Save this empty file as `space-pet.py`.
- First you need to import all the modules and libraries you will use for this project by typing:

```
from sense_hat import SenseHat
from time import sleep
```

- Next you need to create a `SenseHat()` object to interact with the Sense HAT:

```
sense = SenseHat()
```

  Note that capital letters, full stops, and commas are very important in Python. Your code might not work if you do not include these.

- Create one variable to represent each of the colours you have used in your design. If you use a single letter instead of the full colour name, you will find it easier later on, but make sure you have no duplicates - you can't use `b` for both blue and black!

```
b = (0, 0, 255)
```

You can look up the RGB values of the colours using a colour picker (https://www.w3schools.com/colors/colors_rgb.asp).

These variables are called **tuples**.

Using Python tuples

In the Python programming language, there are several types of data structure (a method of storing data to be accessed by your program) you can use. Two of the most common types of data structure (a method of storing data to be accessed by your program) are lists and tuples.

In Python, a tuple can hold any type of data, and is surrounded by parentheses.

```
my_data = ('Here', 'is', 'my', 'data')
```

The data in a tuple can not be removed, and neither can you add extra data once the tuple has been created. The order of the data in a tuple will always remain the same.

Tuples are very useful for storing data that is going to remain constant throughout your program.

```
red = (255, 0, 0) ## red is always red, so this is constant
edinburgh = (55.9533, 3.1883) ## the latitude and longitude of Edinburgh d
smarties = ('red', 'orange', 'blue', 'green', 'yellow', 'pink', 'violet',
```

You can find out which item is at which position in a tuple by looking at the `index` of the item.

```
smarties[0]
>>> red
smarties[5]
>>> pink
```

You can also loop over the items in a tuple.

```
for color in smarties:
    print(color)
```

I need a hint

Here's a video showing you how to set your colours:

Representing your pet

The pet you designed earlier can now be represented by a Python list.

- Create a list like the one below called `pet1`, and type in the colour of each pixel, using the colour variable names you just created. Note that the list needs to be surrounded by `[` and `]` and each colour pixel must be followed by a comma.

```
pet1 = [
    e, e, e, e, e, e, e, e,
    p, e, e, e, e, e, e, e,
    e, p, e, e, p, e, p, e,
    e, p, g, g, p, y, y, e,
    e, g, g, g, y, w, y, g,
    e, g, g, g, g, y, y, e,
    e, g, e, g, e, g, e, e,
    e, e, e, e, e, e, e, e
    ]
```

- Repeat this for the second pixel pet design, but use a different name, such as `pet2`.

- If you ran your code now, nothing would happen, because so far you have only told the program to store information. To make something happen, you will need to write a command to call on that data and display your colours in the right order on the Sense HAT LED matrix. Add this command below your lists:

```
sense.set_pixels(pet1)
```

- When you've run the program once, you can type directly into the shell to switch back and forth between your two pets.

```
>>> sense.set_pixels(pet2)
```

- If you want to clear the LED matrix after setting the images, you can use the following command:

```
>>> sense.clear()
```

Video help

Here's a video to help you out if you get stuck:

Animate your pet

So far, your pixel pet only changes when you type into the shell. To animate it fully, you will need to switch repeatedly between the pictures with a time delay.

You could write the commands out over and over again but it makes more sense to put them into a loop.

To complete this section you need to create a loop that runs ten times. Within the loop, the pets should be displayed on the Sense HAT, and be switched every half-second.

Have a look at the hints below to learn how to do this.

I need a hint

Here's a video showing you how to animate your pet:

Create a `walking` function

The code you have written that animates your pet now needs to be placed inside a function. This will allow you to trigger the code when an action is performed.

Have a look at the section below if you are unfamiliar with Python function. Then place your loop inside its own named function. In the example below, the function is called 'walking'.

Creating and calling functions in Python

When coding, sometimes you may want to use the same few lines of code multiple times within your script. Alternatively, you may want to have the same few lines of code run every time a certain event occurs, e.g. when a specific key is pressed, or a particular phrase is typed. For tasks like this, you might want to consider using a **function**.

Functions are **named** blocks of code that perform a defined task. Just about the simplest function you can create in Python looks like this:

```
def hello():
    print('Hello World!')
```

You tell Python that you're creating a new function by using the `def` keyword, followed by the name of the function. In this case it is called `hello`. The parentheses after the function name are important.

The colon at the end of the line indicates that the code inside the function will be indented on the next line, just like in a `for` or `while` loop or an `if/elif/else` conditional.

You can **call (run the lines of code within the function)** a function by typing its name with the parentheses included. So to run the example function, you would type `hello()`. Here's the complete program:

```
def hello():
    print('Hello World!')

hello()
```

Have a look at the hints below if you are unsure how to do this.

I need a hint

Here's a video showing the function creation:

Shake to activate

It's time to use the Sense HAT's movement sensors, in particular its accelerometer, to trigger the `walking` function you've created. This will make the project more interactive.
You'll need an infinite loop to begin with. Within the loop, you need to fetch the **raw** accelerometer readings. If the `x`, `y`, or `z` component of the accelerometer reading is above `2`, then your `walking()` function should be called.

If everything works correctly, shaking the Sense HAT will then animate your pixel pet!

Take a look at the hints below for help with writing the script.

I need a hint

Here's a video showing you how the script can be written:

What next?

What else can you do with your pixel pet?

- Why not create a few more pet images, and functions to animate them? You could make it so that when the pet is shaken in one direction, one animation is played, and shaking in a different direction causes a different animation to be played.
- How about changing the speed of the animation, depending on how hard the pet is shaken?
- What other sensors could you use to affect your pet? Could you make it look like it's shivering when the temperature drops?