Week 9: Intro to PyGame

Aims:

to familiarize yourself with the IDEA/ALTER template
use Pygame to display images and draw
to familiarize yourself with the easypg package
to draw moving objects using easypg

1. Download files

1. In a browser, visit `http://www.pygame.org/docs/`. Keep this page open, as you will need to refer to it later.
2. Create a new directory for this week's lab work and download `pygamelab1.zip` into it. You will find this Zip archive in the VLE. Unpack the archive into your directory using the `unzip` command.

2 IDEA/ALTER Template

You've been given a file `template.py`. This is a basic template for Pygame programs, following the IDEA/ALTER concept outlined in Harris' book. Copy this file to a new file called `ex1.py`, open it in an editor and examine it.

Try running `ex1.py`. If the file has execute permission, you should be able to do this by entering `./ex1.py`; otherwise, just enter `python ex1.py`.

Try making some simple changes to this program and observe their effect. For example, try changing window size or the display caption in the 'Display configuration' section, or try specifying a different RGB tuple for the background colour in the 'Entity creation' section.

3 Displaying Images

1. You have been given an image background.jpg. Find out how big this image is (e.g., by viewing with eog or by using the `imginfo` command).

2. Copy `template.py` to a new file `ex2.py`. Edit this file and change the window size to be equal to the image dimensions. Then change the two lines of background creation code in the 'Entity creation' section so that the background object is the image `background.jpg`. You should use `pygame.image.load` to load the image—see the *Image* section of the Pygame docs. You should also convert the pixel format of the loaded image with a line like

   ```
   background = background.convert()
   ```

   (See the *Surface* section of the Pygame docs for more about this.)

3. Run the program and make sure that it displays the image. Experiment with changing the window size; try dimensions smaller than and larger than the image.

4. You have also been given `ship.png` an image of spaceship. Load this with `pygame.image.load` and give the resulting image object the name `ship`. Add another line to convert the pixel format of `ship`, just as you did for the background image.

   Now find the 'Refresh display' section of the program. Underneath the line that blits (i.e., copies) the background onto screen, add a similar line that blits the ship image onto the screen. Use coordinates of (300, 300) to start with.

   Run the program and check that the ship is displayed on top of the background image. Don't worry about the white background of the ship image at moment—we'll get of that next. Try changing the coordinates at which the ship image is blitted and observe the effect on what is displayed by the program.

5. The image `ship.png` is an 'indexed' image, drawing its colours from a limited palette. Pygame allows us to specify that one of these colours should become transparent when the image is displayed.

   Find the code in your program that loads `ship.png` into a Pygame object called `ship` and converts the pixel format. Immediately after this code, add a line that calls the `set_colorkey` method on ship. The argument to this method call should be a tuple containing the red, green and blue values for the colour that should become transparent when the image is displayed. Run the program again and you should no longer see the white background of the ship.

   (For help with using `set_colorkey`, see the *Surface* section of the Pygame docs.)

6. You have been given a third image, `logo.png`—the logo for COMP1551. If you display this image with eog, display or gimp, you will see that it has an 'alpha channel' that already specifies which pixels should be rendered as transparent.

   Beneath the code that loads and converts `ship.png`, add two similar lines that load and convert `logo.png`. Then add a line to the 'Refresh display' section that blits the new logo object onto the screen at some location—(500, 80), for example. What do you notice?

   Read the documentation for the convert method in the *Surface* section of the Pygame docs for an explanation of and solution to this problem. Implement the solution and check that it works.

4 Drawing Things

1. Copy `template.py` to a new file `ex3.py`, then edit this file.

2. Study the *Draw* section of the Pygame docs, then add some lines to `ex3.py` that use some of the functions from `pygame.draw` to draw a simple picture.

Since you are not attempting any dynamic graphics here, the simplest and most efficient approach is to draw your picture directly into the background surface, immediately after the point in the program where it is created.

Where Pygame's drawing functions require a colour, this should be supplied a tuple of three 8-bit integers, representing the red, green and blue components of the colour.

Where these functions require a 'rect', this can be supplied either as a Rect object (see the relevant section of the Pygame docs) or simply as a tuple of four integers, representing the *x* & *y* coordinates of the rectangle's top left corner, its width and its height.

5 Using easypg

`Easypg` is a home-grown package of Python modules that simplify certain aspects of using Pygame. Feel free to use it in your coursework or ignore it, as you wish!

1. To use the package you can enter the command in terminal window

```
export PYTHONPATH=/home/csunix/scpython/lib
```

or better still add the line to your `~/.bashrc` file so you do not need to type it again in future

Useful animated sprites are available in `/home/csunix/scpython/sprites,` as Zip files suitable for using directly with the `easypg Sprite class.`

To check that the path is correct, enter `python` in a terminal window to start an interpreter, then try the following commands at the >>> prompt:

```
import easypg
help(easypg)
```

This will simply list the different modules that are part of `easypg`.

1. Repeat the import and help commands for the module `easypg.colours` and study the documentation that is displayed. Press 'Q' to quit the viewer. Then do the same for `easypg.drawing` and `easypg.utils`.

2. Examine the files `image.py` and `picture.py`. These demonstrate the use of the Window class and various functions from `easypg`. Try running both programs.

6. Creating Moving Objects

1. Copy your `ex2.py` to a new file, `ex5.py`. Then edit this file and strip out the code relating to the logo image. Run the program to make sure that it still displays the background and spaceship images correctly.

2. After the code that creates the ship object, add the following:

```
rect = ship.get_rect()
rect.center = (300, 300)
```

The first of these lines creates a Pygame Rect object representing the position and dimensions of the ship image. This provides a convenient way to move and manipulate the image on screen. The second line has the effect of positioning the ship image so that its centre will be at coordinates (300, 300) when the image is first displayed.

Now find the line that blits your ship image into screen. Replace the coordinates with rect, then run the program again to check that the ship image is still being displayed.

3. Before the while loop, define two new variables vx and vy, representing the *x* and *y* components of the spaceship's velocity:

```
vx = 5 vy = 5
```

Then add the following, immediately after the code that blits the ship image:

```
rect.centerx += vx
rect.centery += vy
```

Try running the program and you will see a problem. Can you see why this has happened?

4. To fix the problem, add the following lines immediately after the code that updates the rect centre coordinates:

```
if rect.left < 0 or rect.right > size[0]:
    vx *= -1
if rect.top < 0 or rect.bottom > size[1]:
    vy *= -1
```

Spend some time trying to figure out what this code does and how it will affect program behaviour. The run the program again to see if you are right.