Pygame - Dev Notes - Graphics - Adding Graphics

- Dr Nick Hayward

A brief intro on adding graphics to sprite objects, background images, &c. in a game window with Pygame.

Contents

Intro

We may now start to add some custom images for our sprite objects. For example, a player object, mobs, projectiles, and a game background.

Add graphics

We can now add images and backgrounds to our game window.

Before we can add our images for the sprites and backgrounds, we need to add some images files to our game's directory structure. For example, we'll normally create an assets folder, and then add any required images, audio, video &c. to sub-directories for use within our game.

So, we may now update our directory structure to include the required assets,

```
|-- game
    |-- assets
        |-- images
            |__ ship.png
```

and so on.

Import assets

In our Pygame code, we'll need to import the Python module for os, which will then allow us to query a local OS's directory structure.

```
# import os
import os
```

Then, we can specify the directory location of the main game file, so Python can keep track of the relative location of this file. For example,

```
game_dir = os.path.dirname(__file__)
```

__file__ is used by Python to abstract the root application file, which is then portable from system to system.

This allows us to set relative paths for game directories, for example

```
# game assets
game_dir = os.path.dirname(__file__)
# relative path to assets dir
assets_dir = os.path.join(game_dir, "assets")
# relative path to image dir
img_dir = os.path.join(assets_dir, "images")
```

We may then import an image for use as a sprite as follows,

```
# assets - images
ship = pygame.image.load(os.path.join(img_dir, "ship.png"))
```

Convert

As we import an image for use as a sprite within our game, we need to use a convert() method to ensure that the image file is of a type Pygame can use natively. If not, there is a potential for the game to perform more slowly. For example,

```
ship = pygame.image.load(os.path.join(img_dir, "ship.png")).convert()
```

In effect, Pygame uses this method, convert(), to create a copy of the image. This image copy is then drawn a lot faster to the Pygame window.

Colour key

For each image that Pygame adds as a sprite, a bounding rectangle will be set with a given colour.

However, in most examples, we want to set the background of our sprite to transparent. This means the rectangle for the image will now blend with the background colour of our game window.

For example,

```
ball.set_colorkey(WHITE)
```

This will now check for white coloured pixels in the image background, and then set them to transparent.

Add game background

We can now add a background image for our game, in effect recreating stars and space. For example, we can add our background as follows,

```
# load graphics
bg_img = pygame.image.load(os.path.join(img_dir, "bg-purple.png")).convert
```

We can also add a rectangle to contain our background image,

```
# add rect for bg - helps locate background
bg_rect = bg_img.get_rect()
```

This basically helps us know where to add our background image, and then subsequently find it as needed within the logic of our game.

We can then draw our background image as part of the game loop,

```
# draw background image - specify image file and rect to load image
window.blit(bg_img, bg_rect)
```

Add game images

Then, we need to add an image for our player's object, projectiles, and mobs.

We can add these as follows

```
# add ship image
ship_img = pygame.image.load(os.path.join(img_dir, "ship-blue.png")).convert()
# ship's laser - projectile
laser_img = pygame.image.load(os.path.join(img_dir, "laser-blue.png")).convert()
# asteroid - mob image
asteroid_img = pygame.image.load(os.path.join(img_dir, "asteroid-med-
grey.png")).convert()
```

To use these new images in our game, we then need to modify the code for each object. For example, for our Player object, we can update our class to include a reference to the ship_img

```
self.image = ship_img
```

However, we can also customise this image by scaling it to better fit our game window.

For example,

```
# load ship image & scale to fit game window...
self.image = pygame.transform.scale(ship_img, (49, 37))
# set colorkey to remove black background for ship's rect - player's object
self.image.set_colorkey(BLACK)
```

We can also update our player object's, a ship for example, rect using a colorkey to ensure the black rect is not visible in the game window.

References

- pygame.image
- pygame.sprite
- Python API - os

Demo

- graphicssprites1.py
- shooter0.4.py
  - add graphics for sprites
    - images for player's ship, ship's laser, and asteroids, &c.
    - set colorkey for rect of sprites
    - set background image for game window...