Pygame - Dev Notes - Sprites - Set Image

A brief intro on using sprites and images with Pygame.

Contents

Intro

As we create an object for a sprite, and then set its bounding rectangle, we may also set an image for this sprite.

Image import

The first thing we need to do is set the location of our images, which we may then import and use within our Pygame window. For example, we might save all our game assets to an appropriately named `assets` directory, which will be located at the root of our game project's directory.

```
|- test-game
    |- assets
        |- images
            |__ green-ball.bmp
    |- basicsprites2.py
```

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
ball = pygame.image.load(os.path.join(img_dir, "green-ball.bmp"))
```

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,

```
ball = pygame.image.load(os.path.join(img_dir, "green-ball.bmp")).convert()
```

In effect, Pygame uses this method, `convert()`, to create a copy of the image. This image copy is then draw 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 background image

We can also set a background image for our game window. For example,

```
# load graphics/images for the game
bg_img = pygame.image.load(os.path.join(img_dir, "bg-lg.png")).convert()
# add rect for bg - helps locate background
bg_rect = bg_img.get_rect()
```

which we set as variables for our defined background image. Then, in the *draw* section of our game loop we may set this background image instead of using the standard colour fill,

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

References

- pygame.sprite

Demo

- basicsprites2.py