Pygame - Dev Notes - Sprites - Intro

A brief intro on using sprites with Pygame.

Contents

- Intro
- Sprites in Pygame
  - create a sprite object
  - sprite rectangles
  - create a sprite group
  - add sprite to group
  - update sprite group
- References
- Demo

Intro

In 2-D computer generated graphics, we may consider a **sprite** as a graphical representation of an image, a character for example, that we may then choose to move and manipulate on the screen.

Whilst a definition of **sprites** was originally limited to 2-D representations that were composited on screen, usually line by line by hardware, this definition has now expanded to include bitmap generated images. This definition also now encompasses images that are parsed from a defined file into a game's frame buffer. i.e. we define an image for rendering in the game window, which is then imported into the game for rendering as a bitmap sprite in the game window.

So, any of our characters may be rendered as **sprites** in the game window.

Sprites in Pygame

Pygame includes support for a single sprite, and sprite groups.

create a sprite object

In Pygame, we may often begin creating a sprite by defining an object for each sprite within the game.

So, we can start by creating a class for a sprite, which we'll call our `Player`

```
# player sprite
class Player(pygame.sprite.Sprite):
    def __init__(self):
        pygame.sprite.Sprite.__init__(self)
        self.image = pygame.Surface((30, 30))
        self.image.fill(WHITE)
        self.rect = self.image.get_rect() #checks images and get rect...
        self.rect.center = (winWidth / 2, winHeight / 2)
        #self.rect.bottom = winHeight
```

This class extends the basic Pygame *sprite* object, which includes many default properties and methods to help us setup and use our sprites in the game window.

We can start our class by defining its constructor, which allows us to define an initialisation method for our class. As you might expect, this will be run each time we create a new object using this class.

There is also a second initialisation call, which is relative to the extended sprite object provided by Pygame. If we don't call this constructor, the sprite object will not be created correctly.

There are also a few basic pre-requisites for creating a sprite in Pygame, including

- `self.image` - basically, how the sprite will look in the game window
- `self.rect` - forms a boundary around, and encloses, each sprite in the game window
  - useful for rendering, moving, collision detection...

sprite rectangles

Each sprite in Pygame has an enclosing rectangle, which acts as the boundary as it is drawn, moved, &c. It also helps Pygame detect and calculate the size of a sprite, where it is located in the game window, and so on.

Each rectangle includes basic properties, which help us define points on the rectangle. For example,

```
      (x, y)         top          topright
           |----------|----------|
           |          |          |
           |          |          |
           |          |          |
     left ----------centre--------- right
           |          |          |
           |          |          |
           |----------|----------|
  bottomleft         bottom        bottomright
```

Each of the above positions on a rectangle may be used to check and set a position value for the Pygame rectangle. For example, we might set a value for the rectangle's `top`. So, we might set the `y` coordinate of a sprite to the bottom of a platform or window by simply setting the `bottom` for our sprite.

create a sprite group

As we start to create our sprites, and update the game window for changes, we can create a sprite group for our game's loop. This group can store all of the sprites we create for our game, assuming we add them explicitly to this sprite group. We may, of course, create as many sprite groups as necessary within the logic of the game.

However, the benefit of this sprite group is that we avoid polluting the game loop itself. In particular, we may now update the game per game loop iteration by simply calling an update to the sprite group in the *update* section of the loop. So, it's a good idea to create and maintain at least one game specific sprite group for all sprites in our game.

We may create a sprite group as follows,

```
game_sprites = pygame.sprite.Group()
```

add sprite to group

We may now create a sprite object, and then add it to our required group. For example,

```
# create player object
player = Player()
# add sprite to game's sprite group
game_sprites.add(player)
```

update sprite group

After creating a sprite, and adding it to the required group, we may call this sprite group as part of the *update* section of the game loop,

```
game_sprites.update()
```

and also in the *draw* section of the game loop,

```
game_sprites.draw(window)
```

where `window` is the defined surface for the game window. These simple additions to the game loop help us abstract logic within the game. Whenever we update a sprite in the game's logic, or create and add a new sprite to the sprite group, we can simply update all sprites in the group using these *update* and *draw* calls in the game loop.

Our new sprite group is rendered, and a sprite is drawn to the centre of the game window.

References

- pygame.sprite

Demo

- basicsprites1.py