Pygame - Dev Notes - Sprites - More Objects

- Dr Nick Hayward

A brief intro on adding more sprite objects to a game window with Pygame.

Contents

Intro

We can add multiple sprite objects to a game window, which may then be added to an existing sprite group or simply added to a new group.

Add extra objects

We can now start to add extra sprite objects to our game window, which are commonly given a collective, generic name of *mob*. We can add the following class Mob to our game,

```
# create a generic mob sprite for the game - standard name is *mob*
class Mob(pygame.sprite.Sprite):
    def __init__(self):
        pygame.sprite.Sprite.__init__(self)
        self.image = pygame.Surface((20, 20))
        self.image.fill(CYAN)
        # specify bounding rect for sprite
        self.rect = self.image.get_rect()
        # specify random start posn & speed of enemies
        self.rect.x = random.randrange(winWidth - self.rect.width)
        self.rect.y = random.randrange(-100, -50)
        self.speed_y = random.randrange(1, 10)

    def update(self):
        self.rect.y += self.speed_y
```

With this class, we can create extra sprite objects, set their size, colour, &c. and then set random x and y coordinates for the starting position of the sprite object. We use random values to ensure that the objects start and move from different positions at the top of the game window, and progress in staggered groups down the window.

Update extra objects

As our enemy objects move down the game window, we need to check if and when they leave the bottom of the game window. We can add the following checks to the update function,

```
def update(self):
    self.rect.y += self.speed_y
    # check if enemy sprite leaves the bottom of the game window - then randomise at the top...
    if self.rect.top > winHeight + 15:
        # specify random start posn & speed of enemies
        self.rect.x = random.randrange(winWidth - self.rect.width)
        self.rect.y = random.randrange(-100, -50)
        self.speed_y = random.randrange(1, 7)
```

So, as each sprite object leaves the bottom of the game window, we can check its position. Then, we may reset the sprite object to the top of the game window. However, we need to ensure that the same sprite object does not simply loop around, and reappear at the same position at the top of the game window. This would become both too easy and tedious for our player.

So, we simply reset our *mob* object to a random path down the window. Hopefully, this will make it slightly harder for our player. We can also ensure that each extra sprite object has a different speed by simply randomising the speed along the y-axis per sprite object.

Show extra objects

We can now create a *mob* group as a container for our extra sprite objects. This group will become particularly useful as we add collision detection later in the game. So, we can update our code as follows, e.g.

```
# sprite groups - game, mob...
mob_sprites = pygame.sprite.Group()
# create sprite objects, add to sprite groups...
for i in range(10):
    mob = Mob()
    # add to game_sprites group to get object updated
    game_sprites.add(mob)
    # add to mob_sprites group - use for collision detection &c.
    mob_sprites.add(mob)
```

We create our *mob* objects, and then simply add them to the required sprite groups. By adding them to the game_sprites group, they will be updated as the game loop is executed. The mob_sprites group will help us easily detect these extra sprite objects when we need to add collision detection, remove them from the game window, and so on.

Modify motion of extra objects

Whilst the above updates work great for random motion along the y-axis, we can add some variation to the movement of an extra sprite object by simply modifying the x-axis. For example, we can modify the x-axis for each extra sprite object to create variant angular motion along both the x-axis and y-axis.

```
# random speed along the x-axis
self.speed_x = random.randrange(-3, 3)
...

self.rect.x += self.speed_x
# check if sprite leaves the bottom of the game window - then randomise at the
top...
if self.rect.top > winHeight + 15 or self.rect.left < -15 or self.rect.right >
winWidth + 15:
    # specify random start posn & speed of extra sprite objects
    self.rect.x = random.randrange(winWidth - self.rect.width)
    self.speed_x = random.randrange(-3, 3)
...
```

So, our mob class may now be updated as follows,

```python
# create a generic extra sprite object for the game - standard name is *mob*
class Mob(pygame.sprite.Sprite):
    def __init__(self):
        pygame.sprite.Sprite.__init__(self)
        self.image = pygame.Surface((20, 20))
        self.image.fill(CYAN)
        # specify bounding rect for sprite
        self.rect = self.image.get_rect()
        # specify random start posn & speed
        self.rect.x = random.randrange(winWidth - self.rect.width)
        self.rect.y = random.randrange(-100, -50)
        # random speed along the x-axis
        self.speed_x = random.randrange(-3, 3)
        # random speed along the y-axis
        self.speed_y = random.randrange(1, 7)

    def update(self):
        self.rect.x += self.speed_x
        self.rect.y += self.speed_y
        # check if sprite leaves the bottom of the game window - then randomise at
the top...
        if self.rect.top > winHeight + 15 or self.rect.left < -15 or self.rect.right
> winWidth + 15:
            # specify random start posn & speed of extra sprite objects
            self.rect.x = random.randrange(winWidth - self.rect.width)
            self.rect.y = random.randrange(-100, -50)
            self.speed_x = random.randrange(-3, 3)
            self.speed_y = random.randrange(1, 7)
```

We've also added a quick check for the motion of our extra sprite object along the x-axis. So, as the sprite exits on either side of the screen, we can simply create a new sprite on a random path down the screen.

References

- pygame.sprite

Demo

- basicsprites4.py
- shooter0.2.py
    - add enemy objects