Pygame - Dev Notes - Sprites - Collision Detection

- Dr Nick Hayward

A brief intro on adding collision detection between sprite objects to a game window with Pygame.

Contents

Intro

Pygame includes support for adding explicit collision detection between two or more sprites in a game window. We can use built-in functions to help us work with these collisions. Please see the Pygame API for further details,

- pygame.sprite

Basic collision detection

We need to add basic collision detection for each time an object hits the player's object at the foot of the game window. To help with this detection, Pygame includes the following function,

```
# add check for collision - extra objects and player sprites (False = hit object is not deleted from game window)
pygame.sprite.spritecollide(player, mob_sprites, False)
```

This function of the sprite object allows us to check if one sprite object has been hit by another sprite object. In this example, we're checking to see if the player sprite object has been hit by another sprite object from the mob_sprites group. The False parameter is a boolean value for the state of the object that has hit. i.e. this determines whether a mob sprite object should be deleted from the game window or not. In this example, the False value means the hit object will not automatically be deleted from the game window.

This command is particularly useful as it returns a *list* data structure containing any of the mob sprite objects that hit the player sprite object. So, we may now update this code as follows, and store this list in a variable

```
collisions = pygame.sprite.spritecollide(player, mob_sprites, False)
```

We may then use this *list* to check if any collisions have occurred in our game window, e.g.

```
...
if collisions:
    # update game objects &c.
    ...
...
```

We can use the above code to simply return a boolean value to check if the *list* collisions is empty or not.

Sprite group collision detection

We can now add collision detection for various groups of sprites. In effect, we have one group of sprites that may be colliding with another, defined sprite group. So, we may use Pygame's collide method for sprite groups, e.g.

```
# add check for sprite group collide with another sprite group - projectiles hitting enemy objects - use True to delete sprites from each group...
collisions = pygame.sprite.groupcollide(mob_sprites, projectiles, True, True)
```

The boolean parameter values of True and True allow us to delete both the hit enemy objects, and the projectile objects that hit them.

Then, as the *list* of collisions is populated, we may create new sprite objects for those that have been hit and deleted. e.g. extra objects that move down the game window.

```
# add more mobs for those hit and deleted by projectiles
for collision in collisions:
    mob = Mob()
    game_sprites.add(mob)
    mob_sprites.add(mob)
```

If we don't create new extra objects, the game window will quickly run out of sprite objects.

References

- pygame.sprite

Demo

- basicsprites6.py
- shooter0.3.py
    - collision detection of single sprite
    - detect group collisions