Pygame - Dev Notes - Sprites - Collision Detection - Better

- Dr Nick Hayward

A brief intro on improving collision detection between sprite objects in 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.

However, we may also add a few improvements to the basic collision detection offered between sprites in a game window.

Add better collision detection

Basic collision detection uses rectangles to detect one sprite colliding with another. This is technically referred to as follows,

- Axis-aligned Bounding Box (AABB)

However, for some sprite images this will often cause an unrealistic effect as the two images collide. In effect, the image does not appear to collide with the other image due to space caused by each respective rectangle. So, as one corner of a rectangle hits another corner a collision will be detected.

```
   _________
  |         |
  |         |
  |        _|______
  |_______|_|      |
          |        |
          |________|
```

As this example shows, unless each sprite's image fits exactly inside the respective bounding box, there will be space left over.

Now, we have a few options for rectifying this issue. We might choose to simply calculate and set a slightly smaller rectangle as the required bounding box. Another option, for example, might simply be to use a circular bounding box for our sprite image.

The benefit of using a rectangle, an *axis-aligned bounding box*, is that the game is able to detect and calculate collisions much faster for a rectangle bounding box.

However, a *circular bounding box* may be slower. This is simply due to the number of calculations the game may need to perform to check radius of one bounding box against another bounding box radius. Thankfully, this rarely becomes a practical issue unless you're trying to work with thousands of potential sprite images.

One other option, and the most precise, is to use *pixel perfect collision detection*. As the name might suggest, the game engine will check each pixel of each possible sprite image to determine if and when they collided. It's particularly resource intensive unless you require such precision.

Add circle bounding box

We can add some *circle bounding boxes* to our sprite images, in particular for the player and mob objects. We can start by adding explicit circles with a fill colour, which help us check the relative position of the circle's bounding box,

```
self.radius = 20
pygame.draw.circle(self.image, RED, self.rect.center, self.radius)
```

The sprite image for the player's object may have a fixed, known size. So, we may set the radius to 20, for example.

Then, we may add some *circle bounding boxes* to the mob objects as well,

```
self.radius = int(self.rect.width * 0.9 / 2)
pygame.draw.circle(self.image, RED, self.rect.center, self.radius)
```

We've used the same basic pattern to add circles, but for the mob objects we may set each circle's radius relative to the sprite image. So, we're setting the radius as 90% of the width of the sprite image, and then returning half of that value.

To be able to use each *circle bounding box*, we need to update the collision checks as well. We can simply update this check for each mob object in the *update* section of the game loop as follows,

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

We've updated the collision check to explicitly look for circle collisions. With this update, we can now remove the explicit drawn circle for each *circle bounding box* for the player and mob object sprites.

For example, we may simply comment out the drawn circle

```
self.radius = int(self.rect.width * 0.9 / 2)
#pygame.draw.circle(self.image, RED, self.rect.center, self.radius)
```

References

- pygame.draw
- pygame.sprite

Demo

- collisionsprites3.py
- shooter0.5.py
  - better collisions and detection
    - change bounding box for player and mob sprite objects
    - change bounding box to circle, and modify radius to fit sprite objects