PyGame Car Crash Tutorial - Part 5

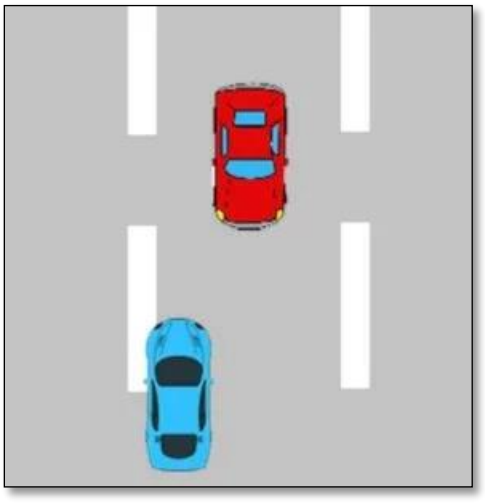
Contents

Time required: 30 minutes

Preview of the Game

Here's a sneak peak of the game that we are going to work on.

CarCrashDemo Video



Car Crash is simple arcade type game. The object is to move your blue car back and forth to avoid the oncoming red cars.

It's time to move it, move it.

enemy.py

We are going to add an update method to the enemy and player classes. The update method will move the sprite at every frame.

We will make one change to the enemy init method, to move the initial position of the enemy car above the program window.

```
12  class Enemy(pygame.sprite.Sprite):
13      """Define the enemy class and methods"""
14
15      def __init__(self):
16          """Construct an enemy object from Sprite class"""
17
18          # Call the constructor of the superclass (pygame.sprite.Sprite)
19          super().__init__()
20
21          # Load enemy car image from file into a variable
22          self.image = pygame.image.load(
23              "./assets/enemy.png").convert_alpha()
24
25          # Get the rectangle area of the player car surface
26          self.rect = self.image.get_rect()
27
28          # Get a random location 40 pixels away from the left and the right.
29          x = randint(40, config.WIDTH - 40)
30
31          # y is -120, the car starts above the program window
32          y = -120
33
34          # Move car to initial position
35          self.rect.move_ip((x, y))
```

This method moves the enemy car down the screen each frame. When it reaches the bottom, it goes back to the top to a random horizontal position.

```
37      # ------------------------- UPDATE ----------------------------------#
38      def update(self):
39          """Update the car's position"""
40          # Move the sprite down SPEED pixels at a time
41          self.rect.move_ip(0, config.SPEED)
42
43          # When the top of the sprite reaches the botton of the surface
44          if (self.rect.top > config.HEIGHT):
45
46              # Get a random location 40 pixels away from the left and right.
47              x = randint(40, config.WIDTH - 40)
48
49              # Move car above the program window
50              y = -120
51
52              # Move car to beginning position
53              self.rect.center = (x, y)
```

player.py

The player's movement is controlled by the keyboard, left and right. Add the update method to the player's class.

```
41  # ---------------------------- UPDATE ----------------------------------#
42      def update(self):
43          """Update the car's position"""
44          # Called each time through the Game Loop
45          # Read the keyboard to see if any keys pressed
46          pressedKeys = pygame.key.get_pressed()
47
48          # Keep the player on the screen
49          # The sprite can't move past the left edge of the surface
50          if self.rect.left > 0:
51
52              # Left arrow key pressed, move left 5 pixels at a time
53              if pressedKeys[pygame.K_LEFT]:
54                  self.rect.move_ip(-5, 0)
55
56          # The sprite can't move past the right edge of the surface
57          if self.rect.right < config.WIDTH:
58
59              # Right arrow key pressed, move right 5 pixels
60              if pressedKeys[pygame.K_RIGHT]:
61                  self.rect.move_ip(5, 0)
```

We define a **move** method for the Player class that controls the movement of the player. When this function is called, it checks to see if any keys are pressed down or not.

The **if** statements check for 2 keys, **LEFT** and **RIGHT**. If the **if** statement proves true, then the **move_ip()** method is called on **Player.rect** moving it in a certain direction. The **move_ip()** takes two parameters, the first representing the distance to be moved in the X direction and second, the distance to be moved in the Y direction.

There are two if statements, as we are testing for two separate event possibilties.

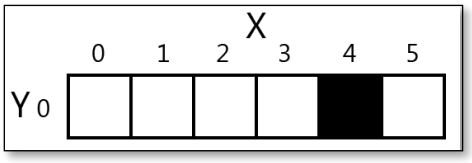
The two if statements, **if self.rect.left > 0:** and **if self.rect.left < config.SCREEN_WIDTH:** ensure that the player isn't able to move off screen.

Animation

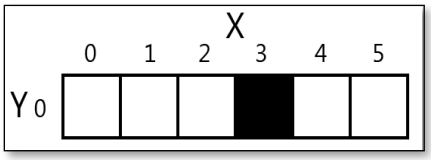
Now that we know how to get the PyGame framework to draw to the screen, let's learn how to make animated pictures. A game with only still, unmoving images is dull. (Sales of the game "Look At This Rock" have been disappointing.)

Animated images are the result of drawing an image on the screen, then a split second later drawing a slightly different image on the screen. Imagine the program's window was 6

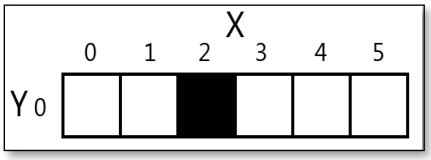
pixels wide and 1 pixel tall, with all the pixels white except for a black pixel at 4, 0. It would look like this:



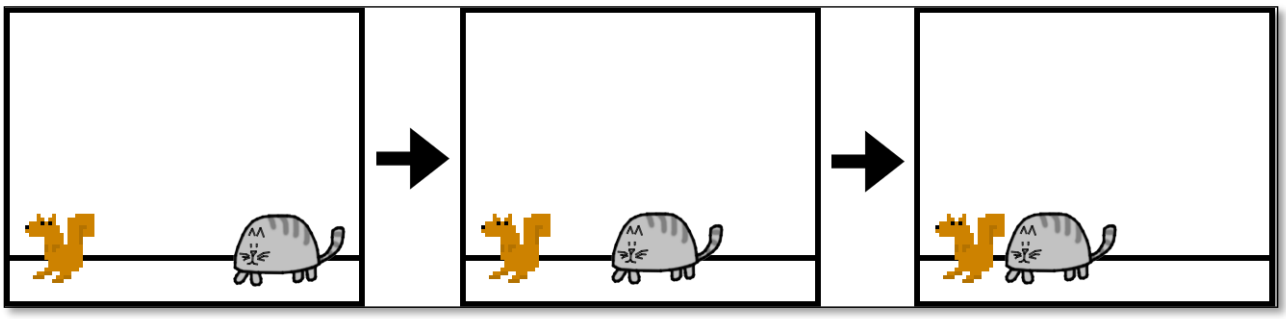
If you changed the image so that 3, 0 was black and 4, 0 was white, it would look like this:



To the user, it looks like the black pixel has "moved" over to the left. If you redrew the window to have the black pixel at 2, 0, it would continue to look like the black pixel is moving left:



It may look like the black pixel is moving, but this is just an illusion. To the computer, it is just showing three different images that each just happen to have one black pixel. Consider if the three following images were rapidly shown on the screen:



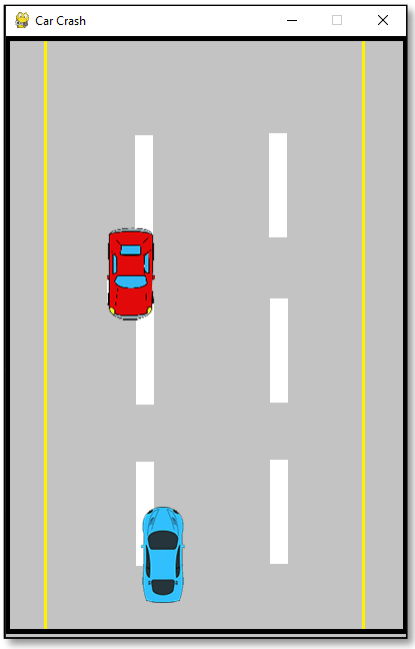
To the user, it would look like the cat is moving towards the squirrel. But to the computer, they're just a bunch of pixels. The trick to making believable looking animation is to have your program draw a picture to the window, wait a fraction of a second, and then draw a slightly different picture.

carcrash_5.py

One change is needed, call the sprites update method from the main program.

```
75  # ---------------------- RUN GAME ----------------------------------------#
76      def game_loop(self):
77          """Start the infinite Game Loop"""
78          while True:
79              self.check_events()
80
81              # ------------------ DRAW ON BACKBUFFER ----------------------#
82              # Draw everything on the backbuffer first
83              # Fill the surface with the background image loaded earlier
84              self.surface.blit(self.background, (0, 0))
85
86              # -------------- UPDATE AND DRAW SPRITES ----------------------#
87              # Run the update method on all sprites
88              self.all_sprites.update()
89
90              # Draw all sprites on the surface
91              self.all_sprites.draw(self.surface)
92
93              # ---------------- UPDATE SURFACE -----------------------------#
94              # From backbuffer, update Pygame display to reflect any changes
95              pygame.display.update()
96
97              # Cap game speed at 60 frames per second
98              self.clock.tick(60)
```

Example run:



The player car moves back and forth with the left and right cursor keys.

The enemy car goes down the road nice and smooth at a random location on the X axis.

Assignment Submission

Zip up the program files folder and submit in Blackboard.