PyGame Tractor Pong Tutorial - Part 4

Contents

PyGame Tractor Pong Tutorial - Part 4 1
Preview of the Game 1
Time to Bounce 2
Assignment Submission 5

Time required: 30 minutes

Preview of the Game

Atari. - the year: 1973 - the date: - November 29th -

That game is called Pong . . . . Then there was Tractor Pong.

Tractor Pong Demo Video

Time to Bounce

1. Save **tractor_pong_3.py** as **tractor_pong_4.py**
2. Let's move things around into different methods. Some of the code that was in here will be moved to different methods.

```
17  class TractorPong:
18      def __init__(self):
19          # Initialize the pygame library
20          pygame.init()
21
22          # Create the game surface (window)
23          self.surface = pygame.display.set_mode((WIDTH, HEIGHT))
24
25          # Set window caption
26          pygame.display.set_caption("Tractor Pong")
27
28          # Only allow these events to be captured
29          # This helps optimize the game for slower computers
30          pygame.event.set_allowed([pygame.QUIT, pygame.KEYDOWN])
31
32          # CLOCK object manages how fast the game runs
33          self.clock = pygame.time.Clock()
34
35          self.load_assets()
```

Create a load_assets method. Some of this is code we have already written, cut and paste it into place.

```
    # ---------------------- LOAD ASSETS ---------------------------------- #
    def load_assets(self):
        # Load the images from the file system into a variable
        ball = pygame.image.load("assets/soccer_ball.png")
        # Convert the image to a PyGame surface
        # This is done to speed up the game
        self.ball = ball.convert_alpha()

        tractor = pygame.image.load("assets/green_tractor.png")
        self.tractor = tractor.convert_alpha()

        # Create a rectangle the same size as the image
        # rect is used to set the location of the image
        self.ball_rect = self.ball.get_rect()
        self.tractor_rect = self.tractor.get_rect()

        # Initial postion of the ball rectangle x random, y/top = 10
        self.set_ball_location()
        self.ball_rect.y = 10

        # Ball speed in pixels for x, y
        self.set_ball_direction()
        self.speed_y = 3

        # Initial location of the tractor
        self.tractor_rect.left = WIDTH // 2
        self.tractor_rect.top = HEIGHT - 90
```

Our game loop is going to be much simpler. We are copying some of the code that used to be here into other methods.

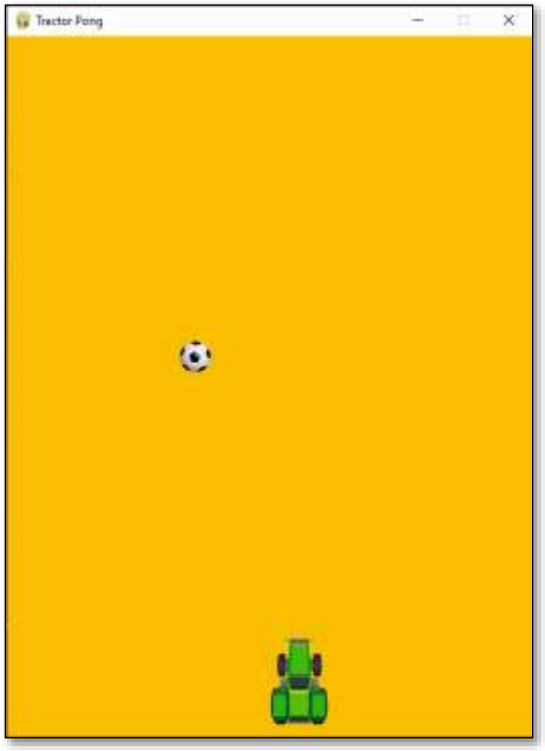
```
# ---------------------------- GAME LOOP ---------------------------------#
    def game_loop(self):
        """Infinite game loop"""
        while True:

            self.check_events()
            self.update_ball()
            self.draw()

            # Cap game speed at 60 frames per second
            self.clock.tick(60)
```

Everything to do with updating the ball will be in this next update_ball method.

```
102     # ------------------------- UPDATE BALL ------------------------------ #
103     def update_ball(self):
104         # Check for collision with left or right wall
105         if self.ball_rect.left <= 0 or self.ball_rect.right >= WIDTH:
106             # Reverse x direction multiply by -1
107             self.speed_x = self.speed_x * -1
108 
109         # Check for collision with top or bottom wall
110         if self.ball_rect.top <= 0 or self.ball_rect.bottom >= HEIGHT:
111             # Reverse y direction multiply by -1
112             self.speed_y = self.speed_y * -1
113 
114         # Move the ball position every frame
115         self.ball_rect.x = self.ball_rect.x + self.speed_x
116         self.ball_rect.y = self.ball_rect.y + self.speed_y
```

All drawing and rendering is in this method. Much of this code was in the game loop.

```
118     # --------------------------- DRAW ----------------------------------- #
119     def draw(self):
120         """Draw everything onto the surface"""
121         # Fill the surface to clear the previous screen
122         # Comment out this line to see why is is necessary
123         self.surface.fill(COUGAR_GOLD)
124 
125         # Draw the ball on the surface
126         self.surface.blit(
127             self.ball,  # What to draw on the surface
128             self.ball_rect,  # Where to draw  on the surface
129         )
130 
131         # Draw the tractor on the surface
132         self.surface.blit(
133             self.tractor,  # Image to draw
134             self.tractor_rect,  # Location to draw the image
135         )
136 
137         # ------------------- UPDATE DISPLAY ---------------------------- #
138         # Copy the surface into video memory
139         pygame.display.update()
```



The ball bounces around the screen off the walls.

Assignment Submission

1. Attach a screenshot showing the operation of the program.
2. Zip up the program files folder and submit in Blackboard.