PyGame Tractor Pong Tutorial - Part 3

Contents

PyGame Tractor Pong Tutorial - Part 3 1
Preview of the Game 1
Time to Bounce 2
Assignment Submission 7

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_2.py** as **tractor_pong_3.py**
2. Modify the following code.

```
"""
Name: tractor_pong_3.py
Author:
Date:
Purpose: The ball moves across the screen
"""

# pip install pygame-ce
import pygame

# Import exit for a clean program shutdown
from sys import exit
from random import randint
from config import COUGAR_GOLD, WIDTH, HEIGHT
```

```
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          # Setup computer clock object to control the speed of the game
33          self.clock = pygame.time.Clock()
34
35          # Load the ball image from the file system into a variable
36          ball = pygame.image.load("assets/soccer_ball.png")
37          # Convert the image to a PyGame surface
38          # This is done to speed up the game
39          self.ball = ball.convert_alpha()
40
41          # Create a rectangle the same size as the ball
42          # rect is used to set the location of the ball
43          self.ball_rect = self.ball.get_rect()
44
45          # Initial postion of the ball rectangle x random, y/top = 10
46          self.set_ball_location()
47          self.ball_rect.y = 10
48
49          # Ball speed in pixels for x, y
50          self.set_ball_direction()
51          self.speed_y = 3
```

Randomization is a way to make a game more interesting. The ball will randomly appear at a different horizontal location and direction.

Add these methods.

```
47  # ------------------------ SET BALL LOCATION -----------------------------#
48      def set_ball_location(self):
49          """Set random initial ball direction along the x axis"""
50          # Randomly determine the initial x coordinate of the ball
51          # along the x-axis (left or right)
52          self.ball_rect.x = randint(20, config.WIDTH - 20)
```

self.ball_rect.x = randint(20, config.WIDTH - 20) - This line sets the x-coordinate of the ball's position (`self.ball_rect.x`) to a random value between 20 pixels (to ensure it's not too close to the edges) and `config.WIDTH - 20` pixels (to ensure it's not too close to the right edge of the screen).

This effectively initializes the ball's position along the x-axis randomly within the game boundaries.

```
54  # ------------------------ SET BALL DIRECTION -----------------------------#
55      def set_ball_direction(self):
56          """Set random initial ball direction along the x axis"""
57          # Randomly determine the initial direction of the ball
58          # along the x-axis (left or right)
59          ball_direction_x = randint(0, 1)
60
61          # If the randomly chosen direction is 0 (left),
62          # set the horizontal speed of the ball to move to the right
63          if ball_direction_x == 0:
64              self.speed_x = 3
65
66          # If the randomly chosen direction is 1 (right),
67          # set the horizontal speed of the ball to move to the left
68          else:
69              self.speed_x = -3
```

ball_direction_x = randint(0, 1) - This line generates a random integer either 0 or 1, representing the initial direction of the ball along the x-axis (left or right).

if ball_direction_x == 0: - This line checks if the randomly chosen direction is 0, indicating that the ball should move to the right.

self.speed_x = 3 - If the ball is meant to move to the right, this line sets the horizontal speed **self.speed_x** to 3, indicating movement towards the right.

else: - If the randomly chosen direction is not 0 (i.e., it's 1), indicating that the ball should move to the left.

self.speed_x = -3 - This line sets the horizontal speed **self.speed_x** to -3, indicating movement towards the left.

```
90      # --------------------------- GAME LOOP ------------------------------ #
91      def game_loop(self):
92          """Infinite game loop"""
93          while True:
94              self.check_events()
95
96              # ------------------ UPDATE BALL ---------------------------- #
97              # Move the ball position every frame
98              self.ball_rect.x = self.ball_rect.x + self.speed_x
99              self.ball_rect.y = self.ball_rect.y + self.speed_y
100
101             # ------------------- DRAW SURFACE ---------------------------- #
102             # Draw everything on the surface first
103             # Fill the surface with Cougar Gold
104             self.surface.fill(COUGAR_GOLD)
105
106             # Draw the ball on the surface
107             self.surface.blit(
108                 self.ball,  # What to draw on the surface
109                 self.ball_rect,  # Where to draw on the surface
110             )
111
112             # ----------------- UPDATE DISPLAY ---------------------------- #
113             # From surface, update Pygame display to reflect any changes
114             pygame.display.update()
115
116             # Cap game speed at 60 frames per second
117             self.clock.tick(60)
118
119
120 # Create game instance
121 tractor_pong = TractorPong()
122 # Start the game
123 tractor_pong.game_loop()
```

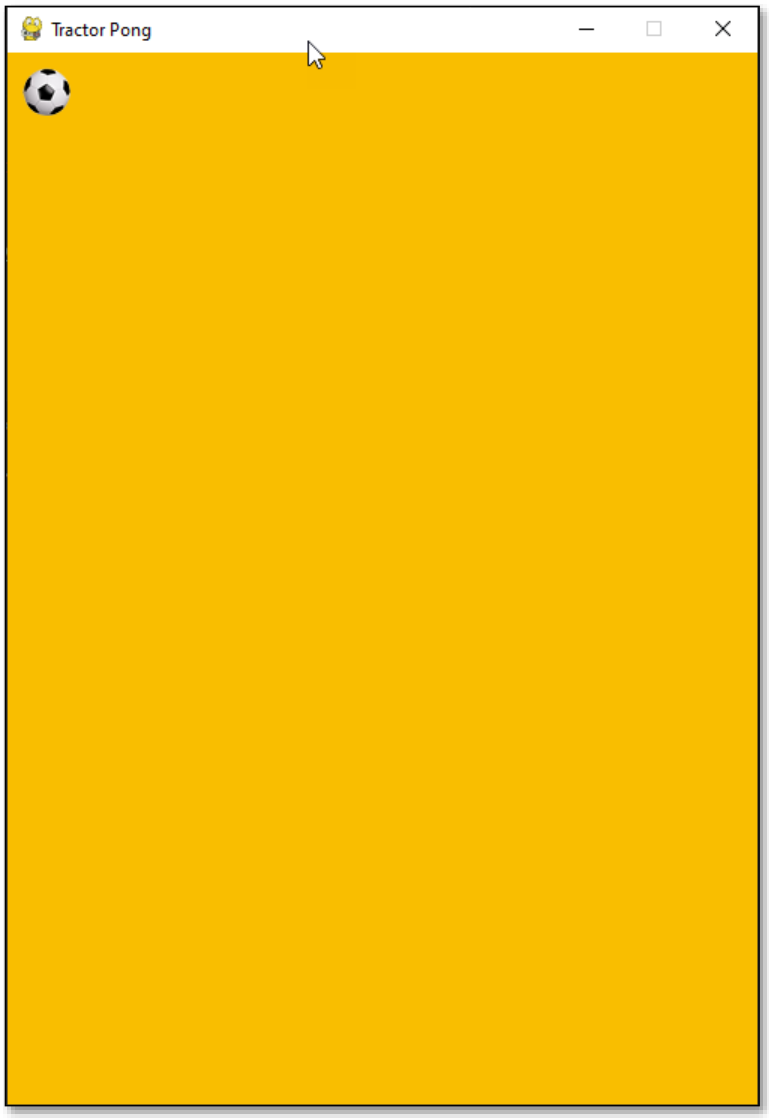
This code updates the position of the ball in the game by adding its current speed along the x and y axes to its current position.

self.ball_rect.x = self.ball_rect.x + self.speed_x - This line updates the x-coordinate of the ball's position (self.ball_rect.x) by adding its current x-axis speed

(self.speed_x). If **self.speed_x** is positive, it moves the ball towards the right; if negative, towards the left.

self.ball_rect.y = self.ball_rect.y + self.speed_y - This line updates the y-coordinate of the ball's position (self.ball_rect.y) by adding its current y-axis speed (self.speed_y). If **self.speed_y** is positive, it moves the ball downwards; if negative, upwards.

These lines update the game's state in each frame, causing the ball to move continuously based on its current speed along the x and y axes.



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.