PyGame Tractor Pong Tutorial - Part 7

Contents

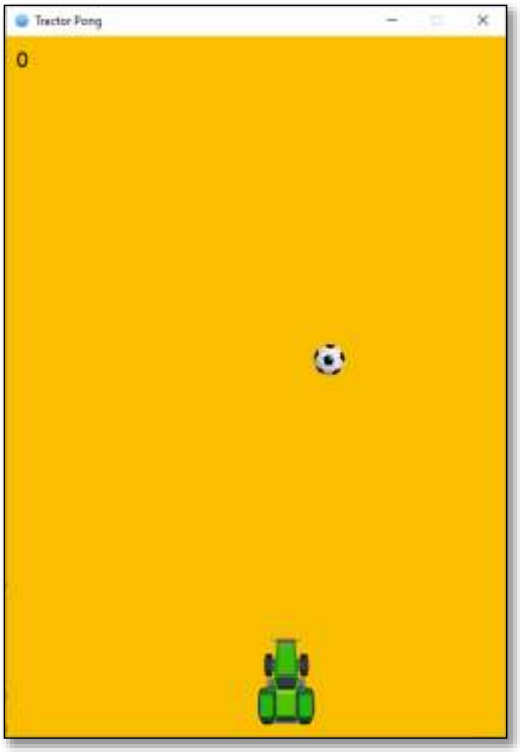
Time required: 30 minutes

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]



Menu Time

The pygame_menu library allows us to create menus.

1. Save **tractor_pong_6.py** as **tractor_pong_7.py**
2. Install pygame-menu-ce

```
# Install pygame-menu-ce
pip install pygame-menu-ce
```

```
 8   # pip install pygame-ce
 9   import pygame
10
11   # pip install pygame-menu-ce
12   import pygame_menu as pm
13   from sys import exit
14   from random import randint
15   from time import sleep
16   from config import COUGAR_GOLD, WIDTH, HEIGHT
```

```
19  class TractorPong:
20      def __init__(self):
21          # Pre initalize mixer with larger buffer size for better performance
22          pygame.mixer.pre_init(
23              44100,  # frequency (Hz)
24              16,  # bit depth
25              2,  # number of channels, 1 mono, 2 stereo
26              4096,  # buffer size, larger to optimize music playback.
27          )
28
29          # Initialize the pygame library
30          pygame.init()
31
32          # Create the game surface (window)
33          self.surface = pygame.display.set_mode((WIDTH, HEIGHT))
34
35          # Set window caption
36          pygame.display.set_caption("Tractor Pong")
37
38          # Only allow these events to be captured
39          # This helps optimize the game for slower computers
40          pygame.event.set_allowed([pygame.QUIT, pygame.KEYDOWN])
41
42          # CLOCK object manages how fast the game runs
43          self.clock = pygame.time.Clock()
44
45          self.load_assets()
46          self.draw_tractor()
47          self.start_music()
48
```

Load assets has changed.

```
49      # ------------------------ LOAD ASSETS ------------------------------ #
50      def load_assets(self):
51          # Load png image, use as program icon
52          ball_ico = pygame.image.load("./assets/blue_ball.png")
53          ball_ico = ball_ico.convert_alpha()
54          pygame.display.set_icon(ball_ico)
55
56          # Load the images from the file system into a variable
57          ball = pygame.image.load("assets/soccer_ball.png")
58          # Convert the image to a PyGame surface
59          # This is done to speed up the game
60          self.ball = ball.convert_alpha()
61
62          tractor = pygame.image.load("assets/green_tractor.png")
63          self.tractor = tractor.convert_alpha()
64
65          # Create a rectangle the same size as the image
66          # rect is used to set the location of the image
67          self.ball_rect = self.ball.get_rect()
68          self.tractor_rect = self.tractor.get_rect()
69
70          # Initial postion of the ball rectangle x random, y/top = 10
71          self.set_ball_location()
72          self.ball_rect.y = 10
73
74          # Ball speed in pixels for x, y
75          self.set_ball_direction()
76          self.speed_y = 3
77
78          # Initial location of the tractor
79          self.tractor_rect.left = WIDTH // 2
80          self.tractor_rect.top = HEIGHT - 90
81
82          # Speed in pixels for the tractor
83          self.tractor_speed = 4
84
85          # Keep track of score
86          self.score = 0
```

Draw Tractor

A method has been added to draw the background and tractor before the tractor startup sound starts.

```
88      # ---------------------- DRAW TRACTOR ---------------------------------- #
89      def draw_tractor(self):
90          self.surface.fill(COUGAR_GOLD)
91          # Draw the tractor on the backbuffer
92          self.surface.blit(
93              self.tractor,  # Image to draw
94              self.tractor_rect,  # Location to draw the image
95          )
96          pygame.display.update()
```

Start Music

This methods loads and starts the background music.

```
98      # ---------------------- START MUSIC ----------------------------------- #
99      def start_music(self):
100         tractor_start = pygame.mixer.Sound("./assets/tractor_starting_up.mp3")
101         tractor_start.set_volume(0.4)  # Set volume to 40%
102         tractor_start.play()
103
104         # Wait until the sound has finished playing
105         while pygame.mixer.get_busy():
106             sleep(0.1)  # wait a bit to reduce CPU usage
107
108         self.ball_hit = pygame.mixer.Sound("./assets/ball.wav")
109         # Set volume for sound effect in range 0.0 - 1.0
110         pygame.mixer.Sound.set_volume(self.ball_hit, 0.5)
111
112         # Load and play background music
113         pygame.mixer.music.load("./assets/tractor_driving.wav")
114
115         # Set volume to 30%, range from 0.0 (mute) to 1.0 (full volume)
116         pygame.mixer.music.set_volume(0.3)
117
118         # Stop any other music from playing
119         pygame.mixer.stop()
120
121         # Play background game music in continious loop from the beginning
122         pygame.mixer.music.play(-1)
123
124         # Create font for scoring
125         self.font_score = pygame.font.SysFont("Verdana", 20)
```

Game Over

This is a brand new method which uses the PyGame Menu library to create a Game Over menu.

```
151     # --------------------------- GAME OVER ------------------------------ #
152     def game_over(self):
153         """Display game over on top of the stopped game"""
154         # Stop background sound
155         pygame.mixer.music.stop()
156         pygame.mixer.music.load("./assets/tractor_driving_game_over.wav")
157 
158         # Play game_over music until the user clicks a button
159         pygame.mixer.music.play(-1)
160 
161         # Define a menu object for the game over screen
162         game_over = pm.Menu(
163             title="Game over",  # Set title menu to "Game over"
164             width=WIDTH,  # Set to width of game surface
165             height=HEIGHT,  # Set to height of game surface
166             # Set the theme of the menu to an orange color scheme
167             theme=pm.themes.THEME_ORANGE,
168         )
169 
170         game_over.add.label(
171             "Tractor Pong",
172             font_size=48,
173             font_color=COUGAR_GOLD,
174             font_name="arialblack",
175         )
176 
177         # Add label to provide space between buttons
178         game_over.add.label("")
179 
180         # Display final score
181         game_over.add.label(f"Score: {self.score}")
182 
183         # Add label to provide space between buttons
184         game_over.add.label("")
```

```
186         # Add a button to the game over menu for exiting the game
187         game_over.add.button(
188             title="Play Again?",  # Button text for play again
189             action=self.restart_game,  # Call main() to start over
190         )
191 
192         # Add label to provide space between buttons
193         game_over.add.label("")
194 
195         # Add a button to the game over menu for exiting the game
196         game_over.add.button(
197             title="Exit",  # Button text
198             action=pm.events.EXIT,  # Exit the game when clicked
199         )
200 
201         # Run the main loop of the game over menu on the specified surface
202         game_over.mainloop(self.surface)
```

There are different themes you can choose for the game_over object. This example uses THEME_ORANGE. You can use any of the following to customize your menu.

```
THEME_BLUE
THEME_DARK
THEME_DEFAULT
THEME_GREEN
THEME_ORANGE
THEME_SOLARIZED
```

Restart game method.

```
204     # ----------------------- RESTART GAME ------------------------------ #
205     def restart_game(self):
206         """Restart the game"""
207         # Stop background sound
208         pygame.mixer.music.stop()
209 
210         # Wait until the sound has finished playing
211         while pygame.mixer.get_busy():
212             sleep(0.1)
213         # Restart the game by calling the main function
214         main()
```

Update Ball

Modify the update_ball method. If the ball hits the bottom, game over.

```
255     # ------------------------- UPDATE BALL ------------------------------ #
256     def update_ball(self):
257         # Check for collision with left or right wall
258         if self.ball_rect.left <= 0 or self.ball_rect.right >= WIDTH:
259             # Reverse x direction multiply by -1
260             self.speed_x = self.speed_x * -1
261 
262         # Check for collision with top or bottom wall
263         if self.ball_rect.top <= 0 or self.ball_rect.bottom >= HEIGHT:
264             # Reverse y direction multiply by -1
265             self.speed_y = self.speed_y * -1
266 
267         # Move the ball position every frame
268         self.ball_rect.x = self.ball_rect.x + self.speed_x
269         self.ball_rect.y = self.ball_rect.y + self.speed_y
270 
271         # Ball hits bottom, player loses
272         if self.ball_rect.bottom > HEIGHT:
273             self.game_over()
```

Check Collision

The check collision method is changed.

```
275     # ---------------------- CHECK COLLISION ------------------------------ #
276     def check_collision(self):
277         """Check for collision between two rects"""
278         # The ball has to be above the tractor to collide
279         # Does the ball collide with the tractor?
280         # If so, reverse the ball y direction [1]
281         if (
282             self.tractor_rect.colliderect(self.ball_rect)
283             and self.ball_rect.bottom < self.tractor_rect.top + 4
284         ):
285 
286             # Reverse y direction
287             self.speed_y = self.speed_y * -1
288 
289             # Randomly change x direction
290             direction = randint(0, 1)
291             if direction == 0:
292                 self.speed_x = self.speed_x * -1
293 
294             # Increase speed by 10% each time the ball is hit
295             self.speed_x = self.speed_x * 1.05
296             self.speed_y = self.speed_y * 1.05
297 
298             # Increase score by 1
299             self.score = self.score + 1
300             pygame.mixer.Sound.play(self.ball_hit)
```

Draw the Score

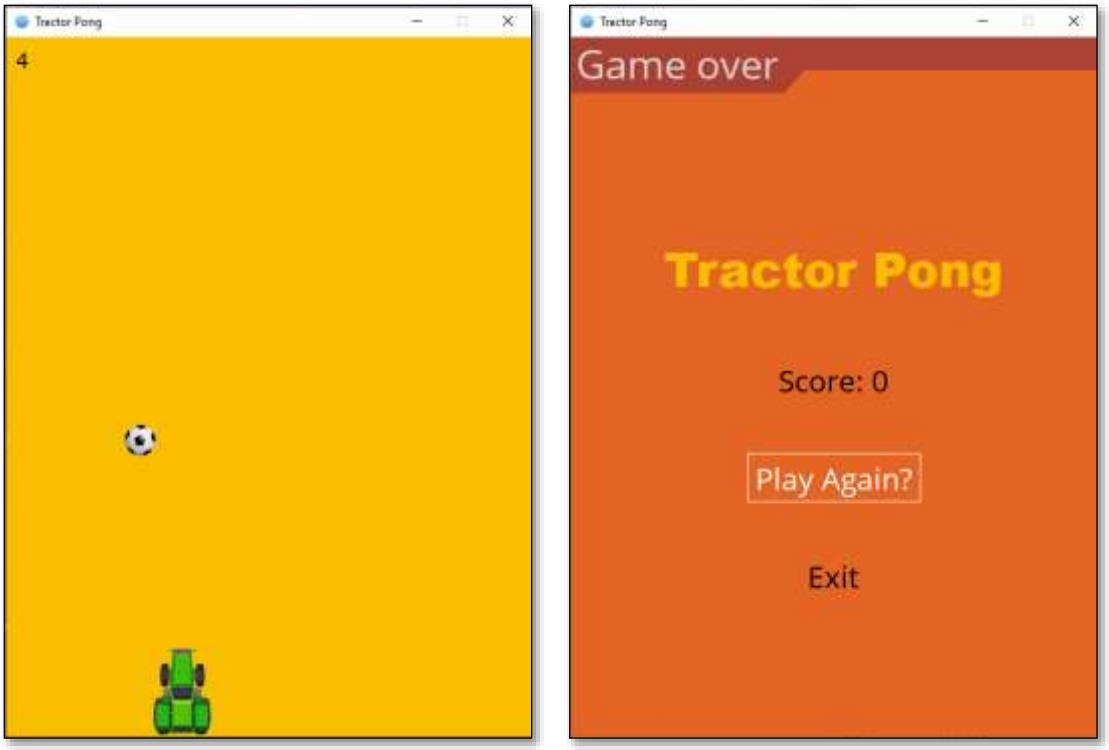
Draw the score on the screen.

```
302     # ------------------------- DRAW ---------------------------------- #
303     def draw(self):
304         # Fill the surface to clear the previous screen
305         # Comment out this line to see why is is necessary
306         self.surface.fill(COUGAR_GOLD)
307 
308         # Draw the ball on the surface
309         self.surface.blit(
310             self.ball,  # What to draw on the surface
311             self.ball_rect,  # Where to draw  on the surface
312         )
313 
314         # Draw the tractor on the surface
315         self.surface.blit(
316             self.tractor,  # Image to draw
317             self.tractor_rect,  # Location to draw the image
318         )
319 
320         # Render score before drawing it on the surface
321         score_display = self.font_score.render(
322             f"{self.score}",  # Score to display
323             True,  # Antialiasing true
324             "black",  # Font color
325         )
326 
327         # Draw score on the surface
328         self.surface.blit(score_display, (10, 10))
329 
330         # --------------------- UPDATE DISPLAY ---------------------------- #
331         # Copy the surface into video memory
332         pygame.display.update()
333
```

Add a main method to allow the game to start over.

```
311 def main():
312     # Initialize program object and start game
313     tractor_pong = TractorPong()
314     tractor_pong.game_loop()
315 
316 
317 main()
```

Example run:



The tractor is King.

What's Next?

- Change the tractor to a different vehicle.
- Change the colors to different RGB colors.
- Add more difficulty levels.
- Keep track of the highest score between games.
- Add more music, change the music
- Change out the images.
- Change the size of the playing field.
- Make the game your own.

Assignment Submission

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