PyGame Tractor Pong Tutorial - Part 7

Contents

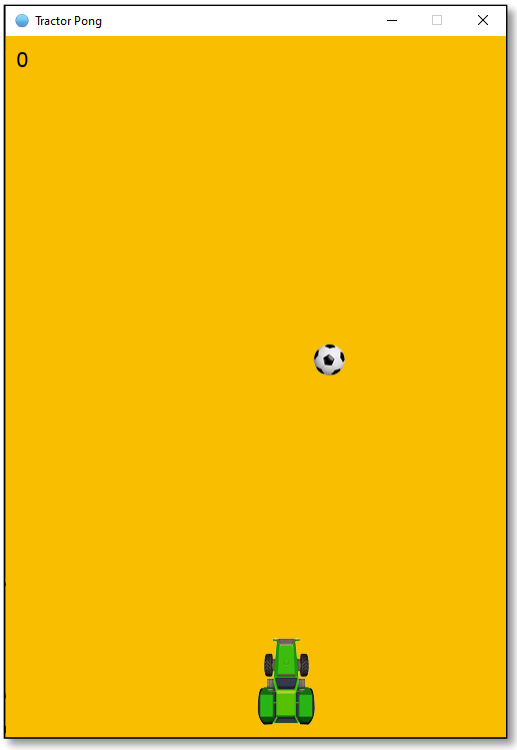
Time required: 30 minutes

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



Collision 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
```

```
7   # pip install pygame-ce
8   import pygame
9   # pip install pygame-menu-ce
10  import pygame_menu as pm
11  from sys import exit
12  from random import randint
13  from time import sleep
14  import config
```

```python
class TractorPong:
    # ------------------- INITIALIZE PYGAME -----------------------------#
    def __init__(self):
        # Pre initalize mixer with larger buffer size for better performance
        pygame.mixer.pre_init(
            44100,  # frequency (Hz)
            16,     # bit depth
            2,      # number of channels, 1 mono, 2 stereo
            4096    # buffer size, larger to optimize music playback.
        )

        # Initialize the pygame library
        pygame.init()

        # Create the game surface (window)
        self.surface = pygame.display.set_mode(
            (config.WIDTH, config.HEIGHT)
        )

        # Set window caption
        pygame.display.set_caption("Tractor Pong")

        # CLOCK object manages how fast the game runs
        self.clock = pygame.time.Clock()

        # Only allow these events to be captured
        # This helps optimize the game for slower computers
        pygame.event.set_allowed(
            [
                pygame.QUIT,
                pygame.KEYDOWN
            ]
        )

        self.load_assets()
        self.draw_tractor()
        self.start_music()
```

```
44  # ---------------------- LOAD ASSETS ------------------------------------#
45      def load_assets(self):
46          # Load png image, use as program icon
47          self.ball_ico = pygame.image.load(
48              "./assets/blue_ball.png").convert_alpha()
49          pygame.display.set_icon(self.ball_ico)
50
51          # Load the images from the file system into a variable
52          self.ball = pygame.image.load(
53              "assets/soccer_ball.png").convert_alpha()
54          self.tractor = pygame.image.load(
55              "assets/green_tractor.png").convert_alpha()
56
57          # Create a rectangle the same size as the image
58          # rect is used to set the location of the image
59          self.ball_rect = self.ball.get_rect()
60          self.tractor_rect = self.tractor.get_rect()
61
62          # Initial postion of the ball rectangle x random, y/top = 10
63          self.set_ball_location()
64          self.ball_rect.y = 10
65
66          # Ball speed in pixels for x, y
67          self.set_ball_direction()
68          self.speed_y = 3
```

Load assets has changed.

```
55  # ---------------------- LOAD ASSETS -----------------------------------
        Codiumate: Options | Test this method
56      def load_assets(self):
57          # Load png image, use as program icon
58          self.ball_ico = pygame.image.load(
59              "./assets/blue_ball.png").convert_alpha()
60          pygame.display.set_icon(self.ball_ico)
61
62          # Load the images from the file system into a variable
63          self.ball = pygame.image.load(
64              "assets/soccer_ball.png").convert_alpha()
65          self.tractor = pygame.image.load(
66              "assets/green_tractor.png").convert_alpha()
67
68          # Create a rectangle the same size as the image
69          # rect is used to set the location of the image
70          self.ball_rect = self.ball.get_rect()
71          self.tractor_rect = self.tractor.get_rect()
72
73          # Initial postion of the ball rectangle x random, y/top = 10
74          self.set_ball_location()
75          self.ball_rect.y = 10
76
77          # Ball speed in pixels for x, y
78          self.set_ball_direction()
79          self.speed_y = 3
80
81          # Initial location of the tractor
82          self.tractor_rect.left = config.WIDTH // 2
83          self.tractor_rect.top = config.HEIGHT - 90
84
85          # Speed in pixels for the tractor
86          self.tractor_speed = 4
87
88          # Keep track of score
89          self.score = 0
```

Draw Tractor

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

```
91  # ---------------------- DRAW TRACTOR ---------------------------------- #
        Codiumate: Options | Test this method
92      def draw_tractor(self):
93          self.surface.fill(config.COUGAR_GOLD)
94          # Draw the tractor on the backbuffer
95          self.surface.blit(
96              self.tractor,       # Image to draw
97              self.tractor_rect   # Location to draw the image
98          )
99          pygame.display.update()
```

Start Music

This methods loads and starts the background music.

```
101    # ---------------------- START MUSIC ---------------------------------- #
           Codiumate: Options | Test this method
102        def start_music(self):
103            tractor_start = pygame.mixer.Sound("./assets/tractor_starting_up.mp3")
104            tractor_start.set_volume(0.4)  # Set volume to 50%
105            tractor_start.play()
106
107            # Wait until the sound has finished playing
108            while pygame.mixer.get_busy():
109                sleep(0.1)  # wait a bit to reduce CPU usage
110
111            self.ball_hit = pygame.mixer.Sound("./assets/whip.wav")
112            self.game_over_snd = pygame.mixer.Sound(
113                "./assets/tractor_driving_game_over.wav"
114            )
115            # Set volume for sound effect in range 0.0 - 1.0
116            pygame.mixer.Sound.set_volume(self.game_over_snd, .5)
117
118            # Load and play background music
119            pygame.mixer.music.load("./assets/tractor_driving.wav")
120
121            # Set volume to 30%, range from 0.0 (mute) to 1.0 (full volume)
122            pygame.mixer.music.set_volume(0.3)
123
124            # Stop any other music from playing
125            pygame.mixer.stop()
126
127            # Play background game music in continious loop from the beginning
128            pygame.mixer.music.play(-1)
129
130            # Create font for scoring
131            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  # ------------------------ DISPLAY 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
157          # Play game_over music until the user clicks a button
158          pygame.mixer.Sound.play(self.game_over_snd, loops=-1)
159
160          # Define a menu object for the game over screen
161          game_over = pm.Menu(
162              title="Game over",      # Set title menu to "Game over"
163              width=config.WIDTH,     # Set to width of game surface
164              height=config.HEIGHT,   # Set to height of game surface
165              # Set the theme of the menu to an orange color scheme
166              theme=pm.themes.THEME_ORANGE
167          )
168
169          # Display final score
170          game_over.add.label(f"Score: {self.score}")
171
172          # Add label to provide space between buttons
173          game_over.add.label("")
174
175          # Add a button to the game over menu for exiting the game
176          game_over.add.button(
177              title="Play Again?",    # Button text
178              action=main             # Call main() to start over
179          )
180
181          # Add label to provide space between buttons
182          game_over.add.label("")
183
184          # Add a button to the game over menu for exiting the game
185          game_over.add.button(
186              title="Exit",           # Button text
187              action=pm.events.EXIT   # Exit the game when clicked
188          )
189
190          # Run the main loop of the game over menu on the specified surface
191          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
```

Update Ball

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

```
232  # ------------------------- UPDATE BALL ----------------------------------#
233      def update_ball(self):
234          # Check for collision with left or right wall
235          if self.ball_rect.left <= 0 or self.ball_rect.right >= config.WIDTH:
236              # Reverse x direction multiply by -1
237              self.speed_x = self.speed_x * -1
238
239          # Check for collision with top or bottom wall
240          if self.ball_rect.top <= 0 or self.ball_rect.bottom >= config.HEIGHT:
241              # Reverse y direction multiply by -1
242              self.speed_y = self.speed_y * -1
243
244          # Move the ball position every frame
245          self.ball_rect.x = self.ball_rect.x + self.speed_x
246          self.ball_rect.y = self.ball_rect.y + self.speed_y
247
248          # Ball hits bottom, player loses
249          if self.ball_rect.bottom > config.HEIGHT:
250              self.game_over()
```

Check Collision

The check collision method is changed.

```
252  # ---------------------- CHECK COLLISION ----------------------------------#
253      def check_collision(self):
254          """Check for collision between two rects"""
255          # The ball has to be above the tractor to collide
256          # Does the ball collide with the tractor?
257          # If so, reverse the ball y direction [1]
258          if self.tractor_rect.colliderect(
259              self.ball_rect
260          ) and self.ball_rect.bottom < self.tractor_rect.top + 4:
261
262              # Reverse y direction
263              self.speed_y = self.speed_y * -1
264
265              # Randomly change x direction
266              direction = randint(0, 1)
267              if direction == 0:
268                  self.speed_x = self.speed_x * -1
269
270              # Increase speed by 10% each time the ball is hit
271              self.speed_x = self.speed_x * 1.05
272              self.speed_y = self.speed_y * 1.05
273
274              # Increase score by 1
275              self.score = self.score + 1
276              pygame.mixer.Sound.play(self.ball_hit)
277
```

Draw the Score

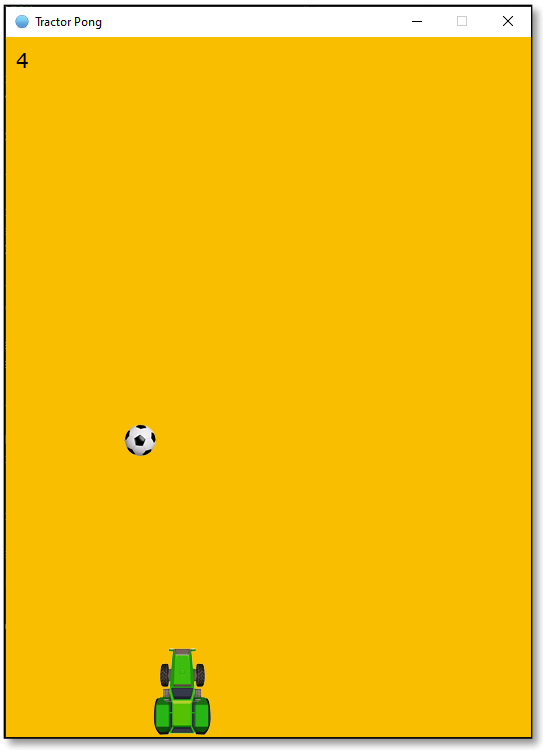
Draw the score on the screen.

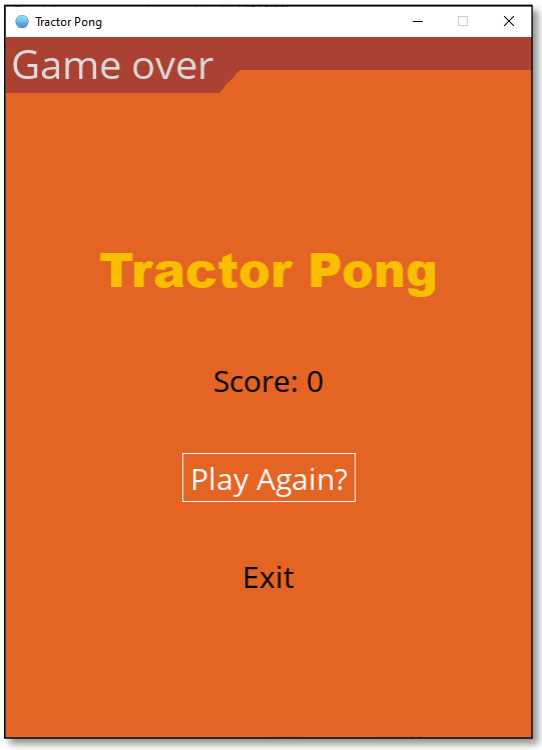
```
278  # -------------------------- DRAW ----------------------------------------#
279      def draw(self):
280          # Fill the display surface to clear the previous screen
281          # Comment out this line to see why is is necessary
282          self.surface.fill(config.COUGAR_GOLD)
283
284          # Draw the ball on the backbuffer
285          self.surface.blit(
286              self.ball,          # Image to draw
287              self.ball_rect      # Location to draw the image
288          )
289
290          # Draw the tractor on the backbuffer
291          self.surface.blit(
292              self.tractor,       # Image to draw
293              self.tractor_rect   # Location to draw the image
294          )
295
296          # Render score before drawing it on the surface
297          score_display = self.font_score.render(
298              f"{self.score}",    # Score
299              True,               # Antialiasing true
300              "black"             # Font color
301          )
302
303          # Draw score on the surface
304          self.surface.blit(score_display, (10, 10))
305
306          # ------- COPY BACKBUFFER INTO VIDEO MEMORY ----------------------#
307          # Copy the backbuffer into video memory
308          pygame.display.update()
309
```

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 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 all tutorials and assignments.
2. Attach screenshots showing the successful operation of each tutorial program.
3. Submit in Blackboard.