PyGame Flappy Bird Tutorial - Part 7

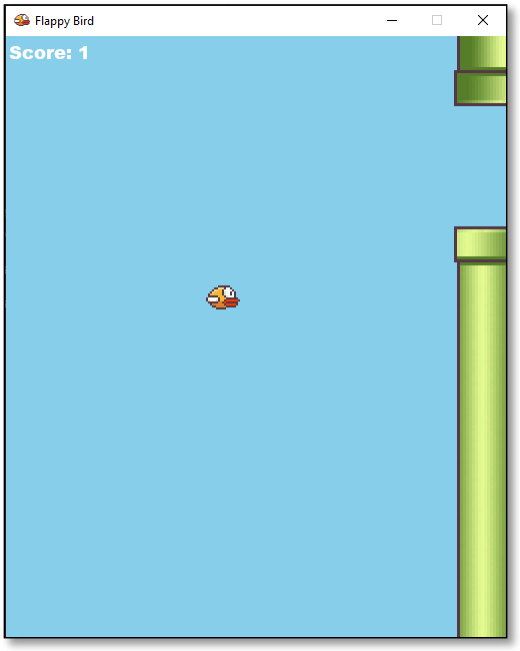
Contents

Time required: 30 minutes

Preview of the Game

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

Flappy Bird Demo Video



Sounds

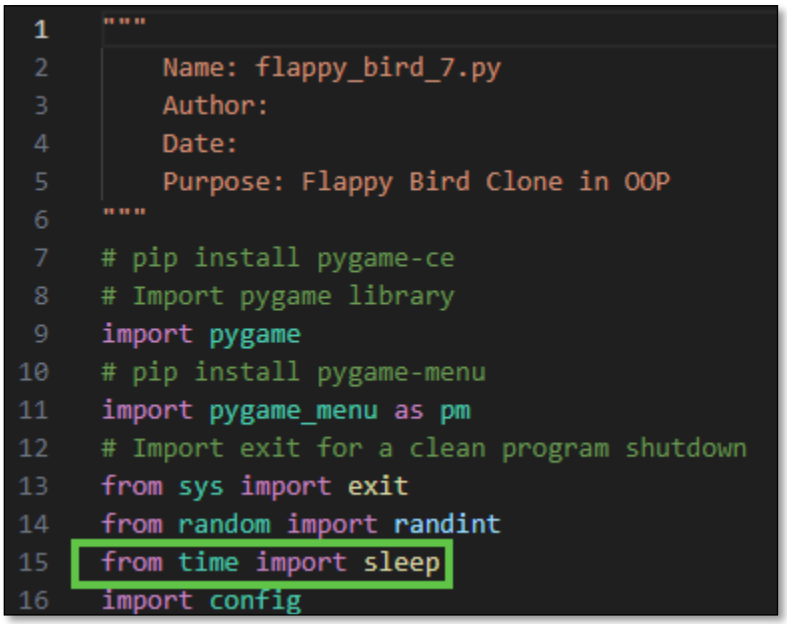
You can use the sounds in the asset file, or create your own.

- https://www.beepbox.co (Create 8 bit songs.)
- https://sfxr.me/ (Create sound effects.)
- https://elevenlabs.io/sound-effects
- https://www.leshylabs.com/apps/sfMaker

Full Game

Time to finish up the game with scoring and sounds.

1. Save **flappy_bird_6.py** as **flappy_bird_7.py**
2. Modify the existing code.
3. Add sleep to pause the game for a moment.



```
"""
    Name: flappy_bird_7.py
    Author:
    Date:
    Purpose: Flappy Bird Clone in OOP
"""
# pip install pygame-ce
# Import pygame library
import pygame
# pip install pygame-menu
import pygame_menu as pm
# Import exit for a clean program shutdown
from sys import exit
from random import randint
from time import sleep
import config
```

4. Initialize the pygame mixer to optimize the sounds.

```
19  class FlappyBird:
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 pygame engine
30          pygame.init()
```

5. Load the background music, set the volume and start the music playing. Setup a few variables.

```
42          self.init_bird()
43          self.init_pipes()
44
45          # Load flappy bird program png icon
46          self.bird_ico = pygame.image.load(
47              "./assets/flappy_bird_ico.png").convert_alpha()
48
49          pygame.display.set_icon(self.bird_ico)
50
51          # Load background music file into memory
52          pygame.mixer.music.load('./assets/flying-minimal.mp3')
53
54          # Set volume to 30%, range from 0.0 (mute) to 1.0 (full volume)
55          pygame.mixer.music.set_volume(0.3)
56
57          # Play in a loop until stopped
58          pygame.mixer.music.play(-1)
59
60          self.score = 0
61          self.game_over = False
62          self.pass_pipe = False
63          self.score_font = pygame.font.SysFont("arialblack", 18)
```

6. Add score counted variable to the end of the init pipes method.

```
112         # Set lower pipe vertical location
113         self.pipe_lower_rect.top = self.pipe_upper_rect.bottom \
114             + self.pipe_gap_size
115
116         # Set score counted to false
117         self.score_counted = False
```

7. Add the same variable to the end of the reset pipes method.

```
# Set initial X off screen to right
self.pipe_upper_rect.left = config.WIDTH
self.pipe_lower_rect.left = config.WIDTH

# New set of pipes, reset score counter
self.score_counted = False
```

8. Add the display score method.

```
154 # ------------------- DISPLAY SCORE -------------------------------------#
155     def display_score(self):
156         """Display the score on the screen"""
157         # Create text image for score display
158         text = self.score_font.render(f"Score: {self.score}", True, "white")
159         self.surface.blit(
160             text,   # Image to display
161             [3, 3]  # x , y to display the image
162         )
```

9. Make some modifications to the display game over menu.

```python
# ------------------------- DISPLAY GAME OVER -----------------------------#
    def display_game_over(self):
        """Display game over menu using the Pygame Menu library"""
        # Stop background sound
        pygame.mixer.music.stop()

        # Play crash sound
        crash = pygame.mixer.Sound('./assets/crash_short.mp3')
        crash.play()
        crash.set_volume(0.3)

        # Wait 2 second while crash plays
        sleep(2)

        # Define a menu object for the game over screen
        game_over = pm.Menu(
            title="Game over",      # Set title menu to "Game over"
            width=config.WIDTH,     # Set to width of game surface
            height=config.HEIGHT,   # Set to height of game surface
            # Set the theme of the menu to an orange color scheme
            theme=pm.themes.THEME_ORANGE
        )

        # Display final score
        game_over.add.label(f"Score: {self.score}")

        # Add label to provide space between buttons
        game_over.add.label("")

        # Add a button to the game over menu for exiting the game
        game_over.add.button(
            title="Play Again?",    # Button text
            action=main             # Call main() to start over
        )

        # Add label to provide space between buttons
        game_over.add.label("")

        # Add a button to the game over menu for exiting the game
        game_over.add.button(
            title="Exit",           # Button text
            action=pm.events.EXIT   # Exit the game when clicked
        )

        # Run the main loop of the game over menu on the specified surface
        game_over.mainloop(self.surface)
```

10. Some game_loop modifications.

```
223   # --------------------------- GAME LOOP ---------------------------------#
224       def game_loop(self):
225           """Infinite game loop"""
226           while not self.game_over:
227               self.check_events()
228               self.detect_collision()
229               # Simulate gravity by moving the bird down
230               # unless the UP key is pressed
231               # Reset gravity to 3 each time through the loop
232               gravity = 3
233
234               # Get list of keys being pressed
235               key_input = pygame.key.get_pressed()
236
237               # If up cursor pressed, move up 5 pixels
238               if key_input[pygame.K_UP]:
239                   gravity -= 5
240
241               # ------------------- INCREASE DIFFICULTY ----------------------#
242               # Adding difficulty relative to score
243               # Increase the speed and decrease the gap of blocks
244               if 5 <= self.score < 10:
245                   self.pipe_move = 5
246                   self.pipe_gap_size = self.bird_rect.height * 4
247
248               elif 10 <= self.score < 20:
249                   self.pipe_move = 7
250                   self.pipe_gap_size = self.bird_rect.height * 3.5
251
252               # ---------------------- SCORING -------------------------------#
253               # If the bird makes it past the pipes, increase score
254               if self.bird_rect.left > self.pipe_upper_rect.right \
255                       and not self.score_counted:
256
257                   # Increase score
258                   self.score += 1
259
260                   # Track whether the current set of pipes have had a score
261                   self.score_counted = True
```

Example run:

A complete game!

What's Next?

There is much more that can be done with this game. Here are some ideas for you to practice and implement on your own.

- Keep track of the score between games.
- Change how often and how many pillars come along.
- Change the pillar image.
- Add some additional audio to the game, such as movement sounds (audio that plays when you move the character)
- Add the concept of multiple Lives or a Health bar.
- Change the colors.
- Change the game to make it 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.