PyGame Flappy Bird Tutorial - Part 7

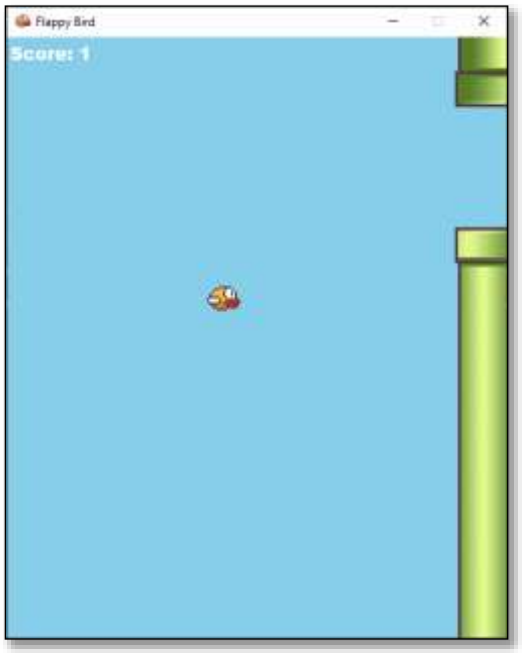
Contents

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
"""

# https://pypi.org/project/pygame-ce
# pip install pygame-ce
import pygame

# https://pypi.org/project/pygame-menu-ce/
# pip install pygame-menu-ce
import pygame_menu as pm

# Import exit for a clean program shutdown
from sys import exit
from random import randint
from time import sleep
from config import WIDTH, HEIGHT, BIRD_X, BIRD_Y
```

Sounds

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()
```

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

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

        # Load flappy bird program png icon
        self.bird_ico = pygame.image.load(
            "./assets/flappy_bird_ico.png"
        ).convert_alpha()

        pygame.display.set_icon(self.bird_ico)

        # Load background music file into memory
        pygame.mixer.music.load("./assets/flying-minimal.mp3")

        # Set volume to 30%, range from 0.0 (mute) to 1.0 (full volume)
        pygame.mixer.music.set_volume(0.3)

        # Play in a loop until stopped
        pygame.mixer.music.play(-1)

        self.score = 0
        self.game_over = False
        self.pass_pipe = False
        self.score_font = pygame.font.SysFont("arialblack", 18)

        # Set the gravity to 3
        # This is how fast the bird falls
        # The higher the number, the faster the bird falls
        # The lower the number, the slower the bird falls
        self.gravity = 3
        self.pipes_speed = 4  # Pipes move faster than background
        self.background_speed = 2  # Background moves slower than pipes

        self.load_background()
        self.init_bird()
        self.init_pipes()
```

Score

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

```
# Set lower pipe vertical location
self.pipe_lower_rect.top = (
    self.pipe_upper_rect.bottom + self.pipe_gap_size
)
# Set score counted to false
# This is used to track if the bird has passed the pipes
self.score_counted = False
```

Add increase difficulty and the **self.score_counted** variable to the end of the reset pipes method.

```
187     # ---------------------- RESET PIPES ---------------------------------- #
188     def reset_pipes(self):
189         """Reset pipes every time they leave the screen"""
190         # Pick a random height for the bottom of the top pipe
191         self.pipe_upper_rect.bottom = randint(
192             50,  # Set minimum random number to 50
193             HEIGHT // 2,  # Set maximum to half the surface height
194         )
195
196         # Set lower pipe top to upper pipe bottom plus pipe gap
197         self.pipe_lower_rect.top = (
198             self.pipe_upper_rect.bottom + self.pipe_gap_size
199         )
200
201         # Set initial X off screen to right
202         self.pipe_upper_rect.left = WIDTH
203         self.pipe_lower_rect.left = WIDTH
204
205         # -------------------- INCREASE DIFFICULTY ------------------------- #
206         # Adding difficulty relative to score
207         # Increase the speed and decrease the gap of blocks
208         if 5 <= self.score < 10:
209             self.pipe_speed = self.pipes_speed + 2
210             self.pipe_gap_size = self.bird_rect.height * 4
211
212         elif 10 <= self.score < 20:
213             self.pipe_speed = self.pipes_speed + 2
214             self.pipe_gap_size = self.bird_rect.height * 3.5
215
216         # Set score counted to false
217         # This is used to track if the bird has passed the pipes
218         self.score_counted = False
```

Add the display score method.

```
239     # ---------------------- DISPLAY SCORE ------------------------------- #
240     def display_score(self):
241         """Display the score on the screen"""
242         # Create text image for score display
243         text = self.score_font.render(f"Score: {self.score}", True, "white")
244         self.surface.blit(
245             text,  # Text surface to display
246             [3, 3],  # x, y Position of text on screen
247         )
```

Update the score.

```
302     # ---------------------- UPDATE SCORE ------------------------------- #
303     def update_score(self):
304         # If the bird makes it past the pipes, increase score
305         if (
306             self.bird_rect.left > self.pipe_upper_rect.right
307             and not self.score_counted
308         ):
309             # Increase score
310             self.score += 1
311
312             # Track whether the current set of pipes have had a score
313             self.score_counted = True
```

Game Over

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)
```

Some game_loop modifications.

```
    # ------------------------- GAME LOOP ------------------------------ #
    def game_loop(self):
        """Infinite game loop"""
        while not self.game_over:
            self.check_events()
            self.detect_collision()
            self.update_score()
            self.update_bird()
            self.update_pipes()
            self.update_background()

            # If the pipes are off the screen, reset them
            if self.pipe_upper_rect.right < 0:
                self.reset_pipes()
            # ------------------ DRAW SURFACE --------------------------- #
            # Filling the surface with the background image
            # clears the previous frame
            # Draw the background
            self.surface.blit(self.background, self.background_rect)
            self.surface.blit(self.background, (self.background_rect.right, 0))

            # Draw bird to the surface
            self.surface.blit(self.bird, self.bird_rect)

            # Draw pipes to the surface
            self.surface.blit(
                self.pipe_lower,  # Source image
                self.pipe_lower_rect,  # Destination location of image
            )
            self.surface.blit(
                self.pipe_upper,  # Source image
                self.pipe_upper_rect,  # Destination location of image
            )

            self.display_score()
            # ----------------- UPDATE DISPLAY --------------------------- #
            # From surface, update Pygame display to reflect any changes
            pygame.display.update()
            self.clock.tick(60)
```

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