PyGame Car Crash Tutorial - Part 7

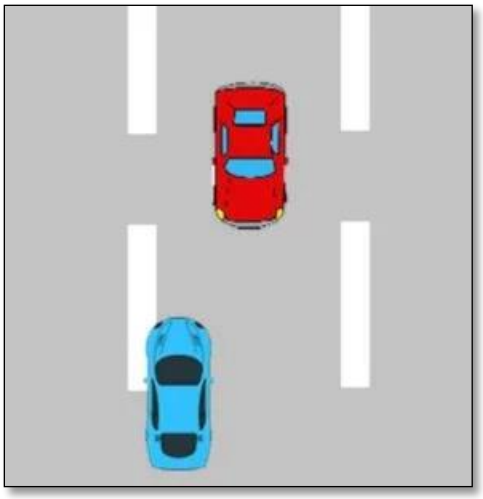
Contents

Time required: 30 minutes

Preview of the Game

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

CarCrashDemo Video



Car Crash is simple arcade type game. The object is to move your blue car back and forth to avoid the oncoming red cars.

Let's finish up our car game with sounds, fonts, and a scoring system.

config.py

```
"""
    Filename: config.py
    Author:
    Date:
    Purpose: Global variables and constants for the entire program
"""
# Import config module into all other modules

# Setup global constants and variables for screen size
WIDTH = 400
HEIGHT = 600

# Global variable for speed across screen
SPEED = 4

# Constant for how much the speed increases each enemy car pass
SPEED_INCREASE = .4

BLACK = (0, 0, 0)
```

enemy.py

To keep score, initialize a score variable in the enemy.py class.

```
14  class Enemy(pygame.sprite.Sprite):
15      """Define the enemy class and methods"""
16  # ------------------------ INITIALIZE ENEMY OBJECT -----------------------#
17
18      def __init__(self):
19          self.score = 0
20          """Construct an enemy object from Sprite class"""
21
22          # Call the constructor of the superclass (pygame.sprite.Sprite)
23          super().__init__()
24
25          self.score = 0
26          self.speed = config.SPEED
27
28          # Load enemy car image from file into a variable
29          self.image = pygame.image.load(
30              "./assets/enemy.png").convert_alpha()
```

We keep track of how many times the Enemy gets to the bottom of the screen. Each time this happens, the player gets one more point.

```python
    # -------------------------- UPDATE ----------------------------------#
    def update(self):
        # Move the sprite down SPEED pixels at a time
        self.rect.move_ip(0, self.speed)

        # When the top of the sprite reaches the botton of the surface
        if (self.rect.top > config.HEIGHT):
            # Get a random location 40 pixels away from left and right.
            x = randint(40, config.WIDTH - 40)

            # Move car above the program window
            y = -120

            # Move car to beginning position
            self.rect.center = (x, y)

            # Increase speed each time the enemy car starts at the top
            self.speed += config.SPEED_INCREASE

            # Increment score every time the player dodges an oncoming car
            self.score += 1
```

car_crash_7.py

We are going to add another library to display our game over screen.

```
pip install pygame-menu
```

```python
"""
    Filename: car_crash_7.py
    Author:
    Date:
    Purpose: Add scoring, sound and a game over menu
"""
# pip install pygame-ce
# Import modules
import pygame
# pip install pygame-menu
import pygame_menu as pm
from sys import exit
from time import sleep
# Import our game classes
import config
import player
import enemy
```

Sound

You can use the sound files in the assets file, or make 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

```python
class CarCrash:
    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()
```

```
48          # Optimize game by only allowing these events to be captured
49          pygame.event.set_allowed(
50              [pygame.QUIT, pygame.KEYDOWN, pygame.KEYUP]
51          )
52
53          # Set window icon
54          window_icon = pygame.image.load('./assets/car.ico')
55          pygame.display.set_icon(window_icon)
56
57          # Create the player and enemy sprites
58          self.create_sprites()
59
60          # Create System font object for score
61          self.font_small = pygame.font.SysFont("arialblack", 20)
62
63          # Load sound file into memory
64          pygame.mixer.music.load('./assets/background_music.wav')
65
66          # Set volume to 30%, range from 0.0 (mute) to 1.0 (full volume)
67          pygame.mixer.music.set_volume(0.3)
68
69          # Play in a loop until stopped
70          pygame.mixer.music.play(-1)
```

Add the display_game_over method.

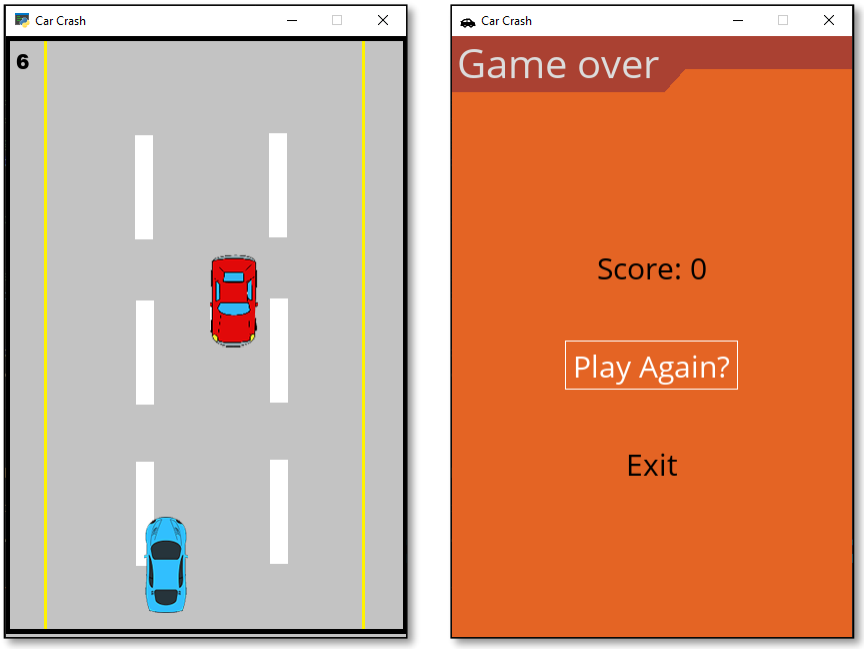
```python
    # ------------------------ DISPLAY GAME OVER -----------------------------#
    def display_game_over(self):
        """Display game over on top of the stopped game"""
        # Stop background sound
        pygame.mixer.music.stop()

        # Play crash sound
        crash = pygame.mixer.Sound("assets/crash.wav")
        crash.play()
        crash.set_volume(0.5)

        # Wait 1 second
        sleep(1)

        # 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.enemy_sprite.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)
```

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
```

```
157  # ---------------------- GAME LOOP ------------------------------------#
158      def game_loop(self):
159          """Start the infinite Game Loop"""
160          while True:
161              self.check_events()
162              self.check_collision()
163
164             # ---------------- DRAW ON BACKBUFFER -------------------------#
165              # Draw everything on the backbuffer first
166              # Fill the surface with the background image loaded earlier
167              self.surface.blit(self.background, (0, 0))
168
169              # -------------- UPDATE AND DRAW SPRITES ---------------------#
170              # Run the update method on all sprites
171              self.all_sprites.update()
172
173              # Draw all sprites on the surface
174              self.all_sprites.draw(self.surface)
175
176              # Render score before drawing it on the surface
177              self.score = self.font_small.render(
178                  str(self.enemy_sprite.score), True, config.BLACK
179              )
180
181              # Draw score on the surface
182              self.surface.blit(self.score, (10, 10))
183
184              # ---------------- UPDATE SURFACE -----------------------------#
185              # From backbuffer, update Pygame display to reflect any changes
186              pygame.display.update()
187
188              # Cap game speed at 60 frames per second
189              self.clock.tick(60)
190
191
192  def main():
193      # Create game instance
194      car_crash = CarCrash()
195      # Start the game
196      car_crash.game_loop()
197
198
199  main()
```

Example run:



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.
- Multiple enemies spawning after set periods of time. (Similar to how we increased speed after a set period of time)
- 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.
- Variations in the shape and size of the "enemies".
- Change the colors.
- Change the game to make it your own.

Assignment Submission

Zip up the program files folder and submit in Blackboard.