Abstract

This project demonstrates a basic 2D car animation created using Python and the Pygame library. The goal of the project is to simulate a car moving smoothly across the screen on a road, with background elements such as trees and buildings to provide a visually appealing environment. The car moves continuously from the left side of the screen to the right and resets its position once it goes off-screen, creating an endless animation loop.

The animation employs simple shapes like rectangles, circles, and polygons to draw the car and background objects. The car features a detailed design, including headlights, taillights, wheels, and a spoiler. The project is a practical example for beginners to understand the basics of game development, Pygame's rendering capabilities, and event handling in real-time applications. This animation can be expanded for use in more complex applications, such as a driving game or educational simulation.

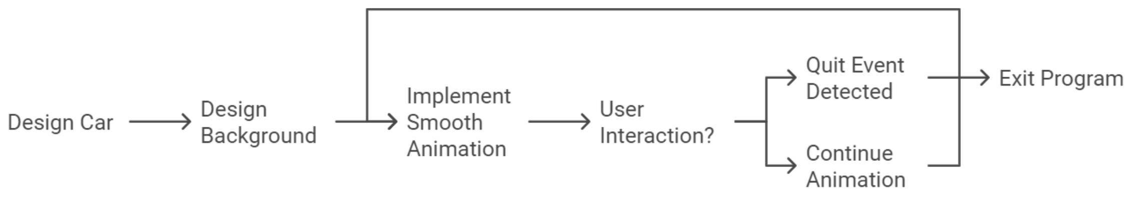
Key features of the project include smooth car movement, background rendering, and a user-friendly interface that continuously updates the display at a frame rate of 60 frames per second. This project showcases the foundational concepts of animation using Pygame and provides a platform for learning and experimentation in game development.

Introduction

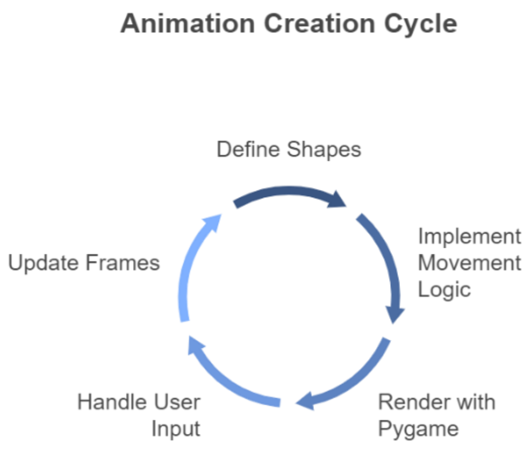
This project is a simple 2D car animation implemented using the **Pygame** library in Python. The animation features a red car that continuously moves from left to right across the screen, passing through a road with buildings and trees as background elements. The car smoothly loops from one side of the screen to the other, giving the impression of infinite motion.

Key Features of the Project:

1. **Car Animation**: The car moves across the screen, with elements such as windows, headlights, taillights, wheels, and a spoiler designed using basic Pygame drawing functions like rectangles, polygons, and circles.
2. **Background Elements**: The road, trees, and buildings are added to create a more realistic environment and provide context to the car's motion.
3. **Smooth Animation**: The movement is controlled using a game loop, and the frame rate is limited to 60 FPS to ensure smooth and consistent animation. When the car moves off the right side of the screen, it reappears from the left, creating a seamless animation effect.
4. **User Interaction**: The program listens for user inputs, particularly the quit event, allowing users to close the window by clicking the exit button.



This project demonstrates how simple geometric shapes and basic movement logic can be combined to create visually appealing animations, making it an ideal starting point for those interested in game development or graphics programming. The Pygame library handles many aspects of game development, including rendering, user input, and frame updates, making it a great tool for beginners



System Specification

Hardware Requirements:

Processor: Intel i3 or above

RAM: 2 GB or more

Hard Disk: 100 MB of free space

Display: 800x600 resolution or higher

Software Requirements:

Operating System: Windows/Linux/MacOS

Programming Language: Python 3.x

Python Libraries: pygame

To install pygame, run : `pip install pygame`

Implementation

The project consists of several components:

1. **Car Drawing**: A custom function draws the car with detailed elements like wheels, windows, headlights, and a spoiler.
2. **Background Elements**: Static background elements such as trees, buildings, and a road are drawn to give a sense of environment to the animation.
3. **Movement Logic**: The car moves across the screen with smooth motion. Once the car reaches the edge of the screen, it resets its position and starts moving again, creating an infinite loop.
4. **Main Game Loop**: The game loop constantly updates the display and checks for user input events like closing the window.

Key Components:

a. Car Drawing Function:

The car is drawn using simple shapes like rectangles and circles to create the body, wheels, and lights.

```
def draw_car(x, y):
    pygame.draw.rect(screen, (200, 0, 0), [x, y, 140, 50], 0, 15)  # Main body
    pygame.draw.polygon(screen, (200, 0, 0), [(x + 20, y), (x + 120, y), (x + 100, y -
25), (x + 40, y - 25)])  # Roof
    pygame.draw.rect(screen, (0, 0, 0), [x + 25, y - 20, 35, 20])  # Windows
    pygame.draw.circle(screen, (255, 255, 0), (x + 135, y + 25), 8)  # Headlights
    pygame.draw.circle(screen, (255, 0, 0), (x + 5, y + 25), 8)  # Taillights
    pygame.draw.circle(screen, (0, 0, 0), (x + 35, y + 50), 20)  # Wheels
```

b. Background Drawing:

The background is drawn using rectangles to simulate the road, buildings, and trees.

```
def draw_background_elements():

  pygame.draw.rect(screen, (50, 50, 50), [0, 450, screen_width, 150])  # Road

  pygame.draw.rect(screen, (0, 255, 0), [600, 300, 50, 100])  # Tree 1

  pygame.draw.rect(screen, (128, 128, 128), [500, 250, 100, 200])  # Building 1
```

c. Main Animation Loop:

The car moves across the screen continuously. Once it exits the right side of the screen, its position is reset to the left, giving the effect of an endless loop.

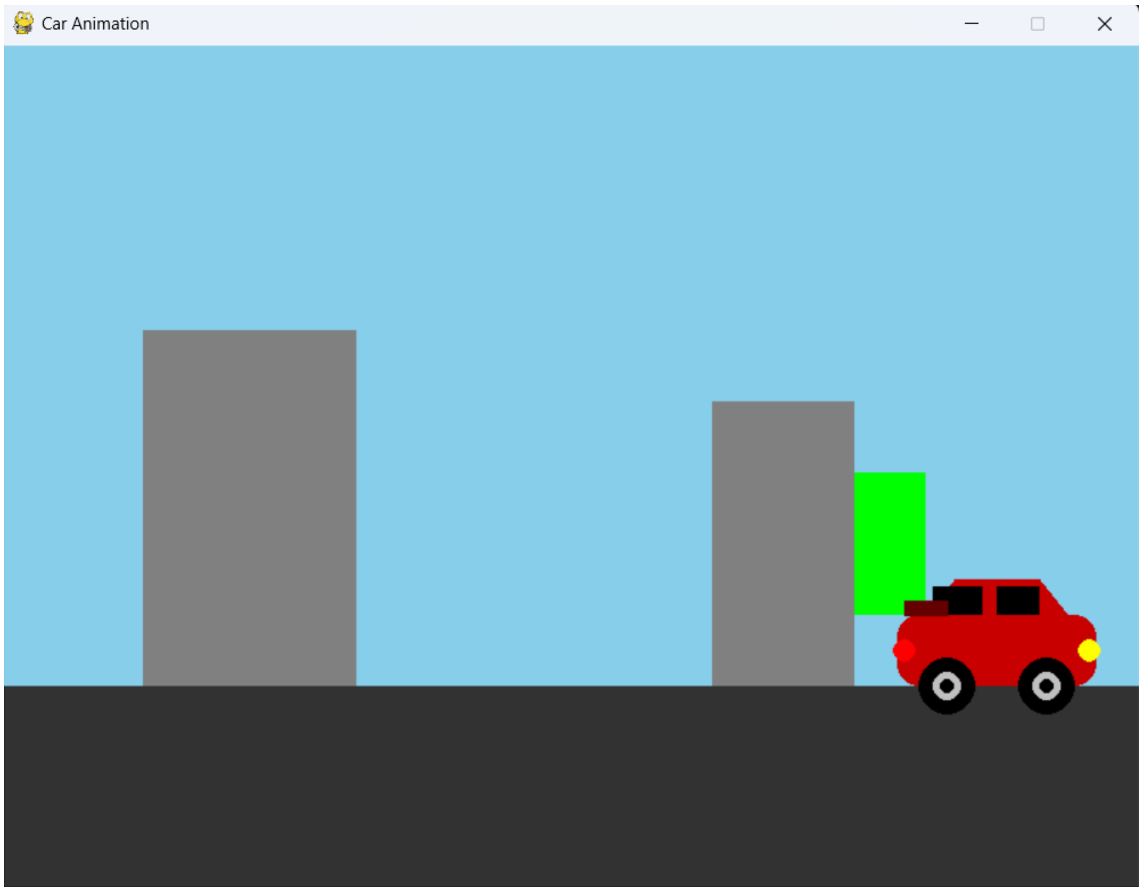
```
        while True:

car_x += car_speed

if car_x > screen_width:

    car_x = -150  # Reset car position
```

Source Code

```
import pygame

import sys


# Initialize Pygame

pygame.init()


# Set up window

screen_width = 800

screen_height = 600

screen = pygame.display.set_mode((screen_width, screen_height))

pygame.display.set_caption(" Car Animation")


# Function to draw an enhanced, detailed car

def draw_car(x, y):

    # Car body

    pygame.draw.rect(screen, (200, 0, 0), [x, y, 140, 50], 0, 15)  # Main body with rounded corners

    pygame.draw.polygon(screen, (200, 0, 0), [(x + 20, y), (x + 120, y), (x + 100, y - 25), (x + 40, y - 25)])  # Car roof


    # Windows

    pygame.draw.rect(screen, (0, 0, 0), [x + 25, y - 20, 35, 20])  # Front window

    pygame.draw.rect(screen, (0, 0, 0), [x + 70, y - 20, 30, 20])  # Rear window


    # Headlights and taillights
    pygame.draw.circle(screen, (255, 255, 0), (x + 135, y + 25), 8)  # Front headlight
    pygame.draw.circle(screen, (255, 0, 0), (x + 5, y + 25), 8)  # Rear taillight

    # Spoiler
    pygame.draw.polygon(screen, (100, 0, 0), [(x + 5, y), (x + 5, y - 10), (x + 35, y - 10), (x + 35, y)])  # Car spoiler

    # Wheels with rims
    pygame.draw.circle(screen, (0, 0, 0), (x + 35, y + 50), 20)  # Front wheel
    pygame.draw.circle(screen, (0, 0, 0), (x + 105, y + 50), 20)  # Rear wheel
    pygame.draw.circle(screen, (192, 192, 192), (x + 35, y + 50), 10)  # Front wheel hubcap
    pygame.draw.circle(screen, (192, 192, 192), (x + 105, y + 50), 10)  # Rear wheel hubcap

    # Rims inside wheels
    pygame.draw.circle(screen, (0, 0, 0), (x + 35, y + 50), 5)  # Front wheel rim
    pygame.draw.circle(screen, (0, 0, 0), (x + 105, y + 50), 5)  # Rear wheel rim

# Function to draw background elements (trees, buildings, road)
def draw_background_elements():
    pygame.draw.rect(screen, (50, 50, 50), [0, 450, screen_width, 150])  # Road
    pygame.draw.rect(screen, (0, 255, 0), [600, 300, 50, 100])  # Tree 1
    pygame.draw.rect(screen, (0, 255, 0), [200, 320, 40, 80])   # Tree 2
    pygame.draw.rect(screen, (128, 128, 128), [500, 250, 100, 200])  # Building 1
    pygame.draw.rect(screen, (128, 128, 128), [100, 200, 150, 250])  # Building 2
```

```
# Main function to run the animation
def main():
    clock = pygame.time.Clock()

    # Initial position and speed for the car
    car_x, car_y = 0, 400
    car_speed = 5  # Speed of the car

    # Main animation loop
    while True:
        screen.fill((135, 206, 235))  # Sky blue background

        # Event handling
        for event in pygame.event.get():
            if event.type == pygame.QUIT:
                pygame.quit()
                sys.exit()

        # Draw background elements
        draw_background_elements()

        # Draw the enhanced car design
        draw_car(car_x, car_y)

        # Move car
        car_x += car_speed
        # Reset car position when it moves off-screen
        if car_x > screen_width:
            car_x = -150  # Reset off the screen

        # Update display
        pygame.display.update()
        clock.tick(60)  # Limit frame rate to 60 FPS

if __name__ == "__main__":
    main()
```

Output



Application

- **Educational Tool**: This project serves as a learning example for beginners in Python and Pygame who want to understand the basics of animation.
- **Game Development**: This simple car animation can be expanded into a more complex game by adding controls for the user to interact with the car or obstacles on the road.
- **Demonstration of Pygame Capabilities**: It showcases how simple shapes can be combined to create more complex objects and animate them in real time.