==================================================================================

**Space Adventure**

**Pygame Space Adventure Game**

==================================================================================

**Table of Contents:**

1. Introduction

2. Installation

3. Usage

4. Classes

5. Sprite Groups

6. Game Loop

7. Scoring and Leveling

8. Controls

9. Sound Effects

10. Health and Lives

11. Game Over

12. Dependencies

==================================================================================

**1. Introduction:**

Space Adventure is a simple space-themed game built using the Pygame library. It involves controlling a spacecraft and shooting down asteroids. The goal is to survive as long as possible, score points, and advance through levels by destroying asteroids.

**2. Installation:**

To run the Space Adventure game, you need to have Python and the Pygame library installed on your system. Follow these steps to install the necessary dependencies:

1. Install Python: Download and install Python from the official Python website (https://www.python.org) based on your operating system.

2. Install Pygame: Open a command prompt or terminal and enter the following command to install Pygame using pip:

pip install pygame

3. Download the Game Assets: Download the game assets, including the background image, spaceship image, asteroid image, explosion sound effect, and any other required files. Place them in the same directory as the game code.

4. Run the Game: Execute the game code using a Python IDE or by running the following command in a command prompt or terminal:

python space\_adventure.py

**3. Usage:**

Once the game is running, you can control the spacecraft using the keyboard arrow keys. Use the left and right arrow keys to move horizontally, the up and down arrow keys to move vertically, and the spacebar to shoot bullets. The goal is to avoid colliding with asteroids while shooting them down to score points and advance through levels.

**4. Classes:**

**4.1 Spacecraft:**

- Description: Represents the player-controlled spacecraft in the game.

- Attributes:

- image: The image representing the spacecraft.

- rect: The rectangular area that encloses the spacecraft image.

- speed\_x: The horizontal speed of the spacecraft.

- speed\_y: The vertical speed of the spacecraft.

- health: The health points of the spacecraft.

- Methods:

- update: Updates the position of the spacecraft based on its speed.

- shoot: Creates a bullet object and adds it to the game.

- take\_damage: Reduces the spacecraft's health by a given amount.

- draw\_health\_bar: Draws the health bar of the spacecraft on the screen.

**4.2 Asteroid:**

- Description: Represents an asteroid that moves down the screen.

- Attributes:

- image: The image representing the asteroid.

- rect: The rectangular area that encloses the asteroid image.

- speed: The vertical speed of the asteroid.

- Methods:

- update: Updates the position of the asteroid based on its speed.

**4.3 Bullet:**

- Description: Represents a bullet shot by the spacecraft.

- Attributes:

- image: The image representing the bullet.

- rect: The rectangular area that encloses the bullet image.

- speed\_y: The vertical speed of the bullet.

- Methods:

- update: Updates the position of the bullet based on its speed.

**5. Sprite Groups:**

The game uses sprite groups to manage and update multiple sprites efficiently. The following sprite groups are used:

- all\_sprites: A group that contains all the sprites in the game, including the spacecraft, asteroids, and bullets.

- asteroids: A group that contains all the asteroid sprites.

- bullets: A group that contains all the bullet sprites.

The sprite groups are used to update and draw all the sprites in the game during the game loop.

**6. Game Loop:**

The game loop is the main loop that runs continuously and handles events, updates the game state, and renders the game graphics. The loop follows these steps:

1. Process Events: Check for and handle user input events such as keyboard input and window close events.

2. Update: Update the position and behavior of all the sprites in the game, including the spacecraft, asteroids, and bullets.

3. Check for Collisions: Detect collisions between the spacecraft and asteroids, and between bullets and asteroids. Perform appropriate actions, such as reducing health, destroying asteroids, and scoring points.

4. Check Level Completion: Check if all asteroids are destroyed. If so, increase the level, spawn more asteroids, and increase their speed.

5. Draw: Draw the background, sprites, score, level, lives, and health bar on the screen.

6. Update Display: Update the display to show the new graphics.

7. Set Frame Rate: Control the frame rate of the game by limiting the loop execution to a specific number of times per second.

8. Repeat: Go back to step 1 and continue the loop until the game is over or the player exits the game.

**7. Scoring and Leveling:**

- Score: The player earns 10 points for each asteroid destroyed by a bullet. The current score is displayed on the screen.

- Level: The game starts at level 1, and the level increases when all asteroids are destroyed. Each level increases the number of asteroids and their speed, providing a higher difficulty level.

**8. Controls:**

- Arrow Keys: Control the movement of the spacecraft (left, right, up, down).

- Spacebar: Shoot a bullet from the spacecraft.

- Close Button: Clicking the close button of the game window will exit the game.

**9. Sound Effects:**

The game includes a sound effect for the explosion of asteroids. The sound effect is played when an asteroid is destroyed by a bullet.It also includes background music for when the game is running.

**10. Health and Lives:**

- Health: The spacecraft has a health attribute representing its health points. The health bar is displayed on the screen, indicating the remaining health of the spacecraft.

- Damage and Game Over: When the spacecraft collides with an asteroid, its health is reduced. If the health reaches zero, the spacecraft loses a life, and the player can continue playing with the remaining lives. If all lives are lost, the game ends, and the player sees the game over screen.

**11. Game Over:**

If the player loses all lives, the game ends. The game over screen is displayed, showing the "Game Over" text in red. The player can exit the game or restart the game by running it again.

**12. Dependencies:**

The Space Adventure game relies on the following dependencies:

- Python 3.x: The programming language used to write the game.

- Pygame: A Python library used for game development. It provides functionality for graphics, sounds, and user input.

Ensure that you have Python and Pygame installed to run the game successfully.