**Project Report**

**Name: Akash Jagtap**

**Roll No. – 05**

**Course – B.Tech Data Science**

**1. Project Title:**

Fruit Slayer Game in Python using Pygame

**2. Objectives:**

* The primary objective of this project is to create a basic, interactive **Fruit Ninja-like game** using **Python** and the **Pygame** library.
* The goal is to make the player "slice" fruits (represented by colored circles) that randomly appear on the screen.
* The game will feature:
  + A simple score system.
  + Fruit slicing interaction through mouse clicks.
  + A countdown mechanism to end the game after a certain number of fruits are sliced.

**3. Features:**

* **Fruit Generation**: Randomly generates fruits that fall from the top of the screen.
* **Mouse Interaction**: The player can slice fruits by clicking the mouse on them.
* **Score System**: Every sliced fruit increases the score, and the total score is displayed on the screen.
* **Game Loop**: The game continues until a specified number of fruits are sliced (5 in this case), after which the game ends and the final score is shown.
* **Start and End Screens**: Displays a start screen where the player can press a key to start the game, and a game-over screen after the game finishes.
* **Restart or Exit Option**: After the game ends, the player can either restart the game or exit the game using a key press.

**4. Technologies Used:**

* **Python**: The programming language used to develop the game.
* **Pygame**: A set of Python modules designed for writing video games. It provides functionalities for handling graphics, events, and sound.
* **Graphics**: Custom shapes (e.g., colored circles) are used to represent the fruits.

**5. Platform Used:**

* The game was developed using **Python 3.x**.
* The platform for development and running the game is a **desktop computer** (can be run on any system with Python and Pygame installed).

**6. Challenges Faced:**

* **Handling Game Timing and Speed**: Initially, managing the speed of the falling fruits and ensuring they disappear off-screen after being missed was tricky.
* **Mouse Interaction**: Detecting mouse clicks on the fruits in a way that was smooth and responsive took some effort, especially with overlapping fruits.
* **Game Reset Logic**: Properly resetting the game variables like score and fruit count after each round of play was a challenge, especially ensuring the state was managed correctly after every restart.
* **Performance**: Ensuring that the game runs smoothly without lag when there are many sprites (fruits) being processed in the game loop.

**7. Functions Used:**

* **pygame.init()**: Initializes all Pygame modules required for game functionality.
* **pygame.display.set\_mode()**: Sets up the display screen size and caption.
* **pygame.time.Clock()**: Controls the game's frame rate (60 FPS in this case).
* **pygame.font.SysFont()**: Loads a system font for displaying text (like the score and game instructions).
* **pygame.event.get()**: Handles the game events (like mouse clicks and key presses).
* **pygame.mouse.get\_pos()**: Retrieves the current mouse position on the screen.
* **pygame.draw.circle()**: Used to draw the fruits (circles) on the screen.
* **pygame.sprite.Sprite()**: Base class for all game objects that are drawn on the screen.
* **pygame.sprite.Group()**: A container for managing multiple sprites.
* **pygame.display.flip()**: Updates the screen to reflect any changes made.
* **pygame.quit()**: Quits the Pygame library and shuts down the game.

**8. Game Output:**

The game consists of the following screens and outputs:

* **Start Screen**: The player is prompted to press any key to start the game.
* **Gameplay**: A black screen with fruits falling down. The score is displayed at the top left. As the player clicks on fruits, the score increases, and the fruit disappears.
* **Game Over Screen**: Once the player has sliced 5 fruits, a game-over screen appears showing the final score, with an option to restart or exit the game.
* **Restart/Exit Options**: After the game ends, the player can either press any key to restart or the ESC key to quit the game.

**9. Feature Enhancement:**

* **Fruit Types**: Introduce different types of fruits with varying sizes, speeds, and point values. For instance, slicing a watermelon could give more points than an apple.
* **Game Difficulty**: Add a difficulty level that increases the speed at which fruits fall, or decrease the time allowed to slice the fruits.
* **Sounds and Effects**: Implement sound effects for slicing fruits and a background music track. Adding visual effects like fruit splashes when sliced would improve the game experience.
* **Leaderboards**: Implement a leaderboard system to track the highest scores over time.
* **Mobile Version**: Optimize the game for touchscreen devices, allowing players to tap on fruits instead of using a mouse.
* **Bonus Fruits**: Add rare fruits that give bonus points or special effects (e.g., a time freeze).

**10. Conclusion:**

The **Fruit Ninja** game project is a simple yet fun demonstration of game development using **Pygame**. It incorporates essential game development concepts such as sprite management, event handling, and game loops. While it’s a basic version, the game is engaging and provides a foundation to add more complex features like power-ups, obstacles, or multiplayer modes. The project successfully highlights the capability of Python for game development and provides an enjoyable experience for players. The challenges encountered and resolved throughout the project helped improve problem-solving and coding skills, and the game can be expanded with more advanced features in the future.