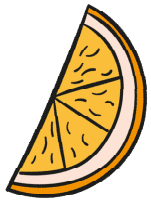
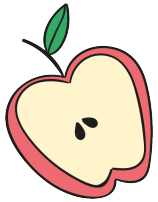
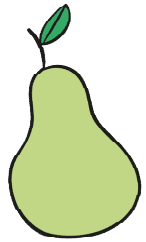
FRUIT-SAMURAI

PROJECT BY:
21BCE201{KAJU PATEL}
21BCE195{HETSI PATEL}

BRIEF

The project is about the Python implementation of a Fruit Samurai game using the Pygame library. The game has a player with three lives who has to cut fruits while avoiding bombs. The game's objective is to score as high as possible before losing all three lives. The game has several entities, including a player, fruits, and bombs.

The code initializes the game window and sets the background, font, and images for the game's various entities. The game loop then starts, and the code tracks the player's lives and score. The game generates fruits at random intervals and random locations, and the player has to cut them by clicking on them with the mouse. If the player misses a fruit or cuts a bomb, they lose a life. The game ends when the player loses all three lives, and a game over screen appears with the final score.

The code also includes functions to draw lives and text on the screen and methods to handle events, such as closing the game window or starting a new game after a game over screen.

PART 01 Libraries used

The code is using the following libraries:

pygame: Pygame is a set of Python modules designed for writing video games. It includes computer graphics and sound libraries designed to be used with the Python programming language. In this code, pygame is used to initialize the game window, load images, play sounds, and handle keyboard events.

sys: The sys module provides access to some variables used or maintained by the interpreter and to functions that interact strongly with the interpreter. In this code, it is used to terminate the game when the user clicks on the close button of the window.

os: The os module provides a way of using operating system dependent functionality like reading or writing to the file system. In this code, it is used to load the font file and create the path for fruit images.

random: The random module is used to generate pseudo-random numbers. In this code, it is used to generate random coordinates, speed, and generate random fruits.

Variables ,colors and window intialization

The code initializes some variables and libraries used in a game.

The variable player_lives keeps track of the number of lives the player has, while the score variable keeps track of the player's score.

The fruits list contains the different types of entities in the game, including fruits like melon, orange, pomegranate, guava, and a bomb.

The pygame library is used to initialize the game window and set its dimensions using WIDTH and HEIGHT variables. The FPS variable controls how often the gameDisplay should refresh.

Different color codes are defined using WHITE, BLACK, RED, GREEN, and BLUE variables.

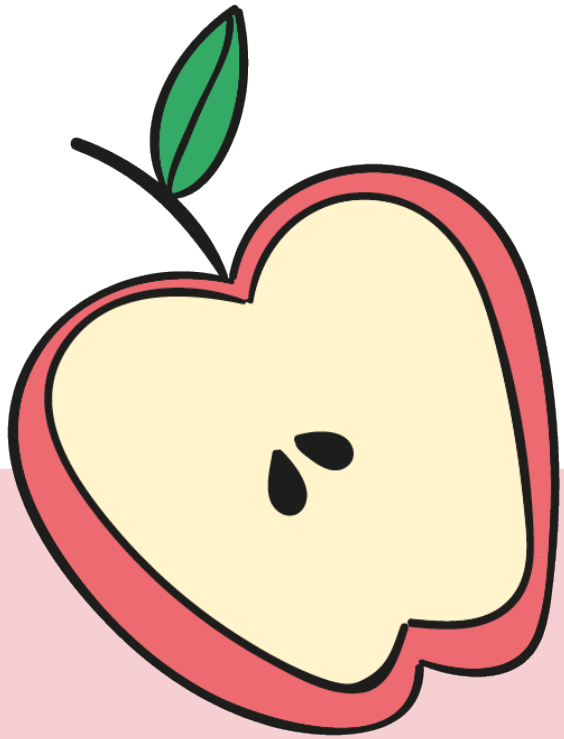
The background variable loads the game's background image, while the font variable sets the font style and size of the game's score display. The score_text variable renders the score display, which will be displayed in the game window.

The lives_icon variable loads the image of the white lives icon, which will be displayed in the game window to indicate the number of lives the player has..

Generalized structure of the fruit Dictionary: generates random coordinates for fruits and stores their attributes in a dictionary called data. The function takes one argument, fruit, which is the name of the fruit. The function first generates the file path of the fruit image by concatenating the string "images/" with the name of the fruit and the string ".png".

Next, the function sets the attributes of the fruit in the data dictionary. The x and y attributes store the x-coordinate and y-coordinate of the fruit, respectively. The speed_x and speed_y attributes determine the speed of the fruit in the x and y directions, respectively. The throw attribute is a Boolean value that determines whether the fruit should be thrown or not. The t attribute is used to manage the time that the fruit has been on the screen. **The hit attribute is used to indicate if the fruit has been hit by the player's cursor or not.**

The function generates random values for the x, speed_x, and speed_y attributes using the random.randint() function. The throw attribute is randomly set to True with a probability of 0.25 using the random.random() function. **If throw is True, it means the generated coordinate of the fruit is outside the gameDisplay, a**nd it will be **discarded.** Otherwise, it will be added to the list of active fruits on the screen..



The generate_random_fruits function generates random fruits at a random position on the screen and stores their information in the data dictionary.

hide_cross_lives(): The hide_cross_lives function draws a red cross icon representing lives at a given (x, y) position on the screen.

draw_text(): The draw_text function is a generic method to draw fonts on the screen. It takes in parameters like display, text, size, x, and y. It sets the font, renders the text with the given parameters and then blits it on the screen at the given coordinates.

draw_lives():The draw_lives function draws the lives of the player using the draw_text function. It takes in parameters like display, x, y, lives, and image. It loops through the range of lives and loads the image of the life (cross icon) and sets the position of each life image. It then blits each image on the screen.

show_gameover_screen():The show_gameover_screen function shows the game over display and the front display. It blits the background image on the screen, displays the game title, the player's score (if the game is not over), and a message to press any key to start the game again. It waits for any key event and then exits the loop when a key is pressed or the game window is closed..

Game loop(main) that runs the Fruit Samurai game. The loop starts by loading and playing the game's background music. It then initializes variables first_round, game_over, and game_running. The game_over variable is used to terminate the game loop if the user cuts more than three bombs.

The loop runs as long as game_running is True. If game_over is True, the loop checks if it's the first round and displays the game over screen if it is. It then resets the game variables player_lives and score.

The loop then checks for events, in this case, only checking for the event of the user closing the game window.

The background is then displayed, followed by the score and the number of remaining player lives. The loop then checks each fruit in the data dictionary to see if it's been thrown by the user. If it has, it updates the fruit's position, checks if the fruit has hit the mouse, updates the score, and generates a new fruit if the current fruit has gone off-screen.

Finally, the loop updates the display, waits for the appropriate time with clock.tick(FPS), and repeats. The loop continues until the user closes the game window or the game is over. After the loop, the game's music is stopped, and pygame is exited.


THANK YOU