

**CS-218**

**DATA STRUCTURES AND ALGORITHMS**

**PROJECT REPORT**

**Fighter Jet**

**Rasib Hassan CS-21071**

**Mazhar Ahmad Khan CS-21096**

**Maaz Mansoor CS-21094**

**Practical Group: G3**

# FIGHTER JET GAME

## Introduction:

Fighter Jet is a high-octane flight simulator game that puts players in the cockpit of a powerful military aircraft. With realistic flight controls and challenging missions, Fighter Jet offers a thrilling gaming experience for players who love fast-paced action and adrenaline. Players will engage in intense air-to-air combat, navigate through dangerous terrain, and complete challenging objectives, all while piloting a state-of-the-art fighter jet. With stunning graphics and dynamic gameplay, Fighter Jet is a must-play for fans of flight simulation games and anyone who loves fast-paced, high-stakes action.



# Mechanism of the Game:

The game mechanics are relatively simple:

**Movement:** You can move your character in two direction to dodge from missile.

**Shooting missile:** You can use missile to defeat enemies.

**Score:** You have a score bar that increases when you give damage to the enemy

In summary, the mechanism of DOOM involves defeating enemies, and exploring the environment to progress and survive.

## How to Run the Game:

- Download Pygame library if not already installed in your machine using command:  
pip install pygame
- Open module main.py and run python file.
- Enjoy the game!

## Controls:

### Movement Keys:

- **UP KEY** to move upward
- **DOWN KEY** to move downward.

### Shooting:

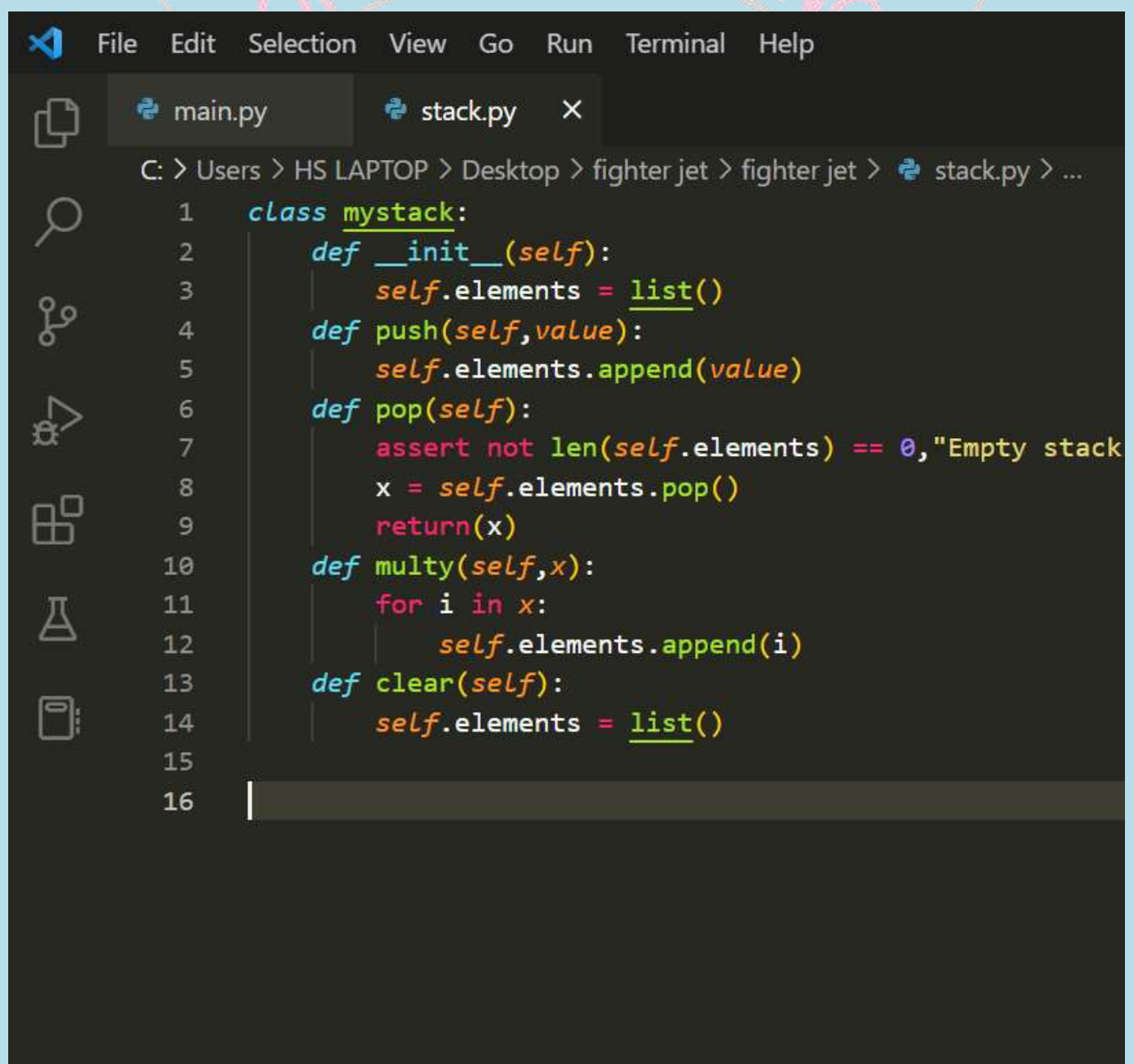
- **SPACE** to fire missile





## Data Structures We Used:

Stack is a linear data structure that follows the Last-In-First-Out (LIFO) principle, where the last element added to the stack is the first one to be removed. It is an abstract data type that consists of a collection of elements with two main operations: push and pop. The push operation adds an element to the top of the stack, while the pop operation removes the element from the top of the stack. Stacks are commonly used in various computer algorithms, programming languages, and computer science applications such as expression evaluation, function calls, and undo/redo operations.



```
File Edit Selection View Go Run Terminal Help
main.py stack.py X
C: > Users > HS LAPTOP > Desktop > fighter jet > fighter jet > stack.py > ...
1  class mystack:
2      def __init__(self):
3          self.elements = list()
4      def push(self,value):
5          self.elements.append(value)
6      def pop(self):
7          assert not len(self.elements) == 0,"Empty stack"
8          x = self.elements.pop()
9          return(x)
10     def multy(self,x):
11         for i in x:
12             self.elements.append(i)
13     def clear(self):
14         self.elements = list()
15
16
```

## Global Variables:



```
File Edit Selection View Go Run Terminal Help
main.py X stack.py
C: > Users > HS LAPTOP > Desktop > fighter jet > fighter jet >
34 data_list = [0, HEIGHT/2, 0, WIDTH/2]
35 Initial.multy(data_list)
36 #Player 1
37 player1 = eval(py_stack.pop())
38 player1X = eval(Initial.pop())
39 player1Y = eval(Initial.pop())
40 player1_change = eval(Initial.pop())
41 #Player 2
42 player2 = eval(py_stack.pop())
43 player2X = eval(Initial.pop())
44 player2Y = eval(Initial.pop())
45 player2_change = eval(Initial.pop())
46 #Bot
47 bot_up = pyInitial
48 bot_down = pyInitial
49 #Bullet of PLayer 1
50 bullet1 = eval(py_stack.pop())
51 bullet1X = pyInitial
52 bullet1Y = pyInitial
53 bullet1_change = 1.5
54 fire1 = True
55 #Bullet of PLayer 2
56 bullet2 = eval(py_stack.pop())
57 bullet2X = pyInitial
58 bullet2Y = pyInitial
59 bullet2_change = 1.5
60 fire2 = True
61 # Score Variable
62 scoreOne = pyInitial
63 scoreTwo = pyInitial
64
```

TECHNOLOG

## Working:

The code is for a 2 player space shooting game using Pygame library in Python. The game window has a fixed width and height of 800 x 600 pixels, and it displays a background image. Background music is played continuously. The game has two players, player 1 and player 2 (a bot). The players and bullets are displayed as images, which are loaded from image files using Pygame's image library.

There are two players and each player has an initial position, movement speed and score display coordinates. The bot randomly moves up or down and the movement is stored in the bot\_up or bot\_down variables. The players move by changing their position on the screen and the bullets move by changing their position in the X direction.

The game has a collision detection mechanism to determine if a player has been hit by the opponent's bullet. If a player is hit, the fire state of the corresponding bullet is set to False and the opponent's score is incremented. The scores of both players are displayed on the screen.

The game runs in a loop that updates the screen and listens for events (e.g. keyboard input to move the players or fire bullets). The game continues until one player reaches a certain score or the game is quit by the player.

## Collision Detection Mechanism:

The collision detection mechanism in the above code is based on the rectangle intersection method. The method checks whether two rectangles overlap by comparing their positions and sizes. If the two rectangles overlap, a collision has occurred. The code calculates the bounding boxes of the two objects (player and enemy) and checks if the rectangles formed by these bounding boxes intersect. If they do, the code returns true, indicating a collision. The code uses the pygame.Rect class to represent rectangles, and the colliderect method of the Rect class to check for an intersection between two rectangles.

## Individual Contribution:

**Mazhar Ahmad Khan (CS-21096):** Making project hierarchy and implementation of main game loop and rendering of sprites.

**Syed Rasib Hassan (CS-21094):** Gave the solution of implementing the data structure stacks for score and use of data structure for enemy score also and

**Maaz Mansoor (CS-21071):** Finding sprites for the game, implementation interface and trouble shooting.



## Future Expansions:

**Multiplayer Mode:** Adding a multiplayer mode where players can compete against each other in air-to-air combat or work together to complete missions.

**Adding AI:** Adding the AI of enemy jets, making them more challenging and realistic to fight against.

**Weapon Upgrades:** Adding the ability to upgrade weapons, making them more powerful and effective against tougher enemies.

**Character Customization** Allowing players to customize their fighter jet, including selecting different skins, equipment, and abilities.

**Boss Battles:** Adding epic air-to-air boss battles that require strategy and teamwork to defeat.

## Conclusion:

In conclusion, the implementation of this shooting game using the Pygame library was a challenging and educational experience. During the project, we utilized our technical skills and knowledge to design and develop a functional solution that meets the requirements. Key features such as sprite rendering, collision detection mechanism and event handling were successfully implemented to improve the overall user experience. Despite facing some challenges, we were able to overcome them through collaboration and determination.

Going forward, there is potential for further improvement in this project. Some areas for future enhancement include adding new features, optimizing performance and improving the user interface. This project has been a valuable learning opportunity that has allowed us to grow as technical professionals, and we are eager to continue learning and advancing in our respective fields.

In addition to the key features mentioned in the previous response, this code also implements collision detection mechanism, which is critical for ensuring that the game play is realistic and enjoyable for the user. This mechanism works by checking if the bullet sprite collides with the enemy sprite, and if a collision is detected, the enemy sprite is removed from the screen. This creates a sense of interaction and challenge for the user as they attempt to eliminate all enemy targets on the screen. The code also uses global variables such as "bullets", "enemies", "player\_rect", and "screen" to keep track of important information and updates throughout the game. These variables play a crucial role in enabling the smooth operation of the game and making it a well-rounded and engaging experience for the player.