Multimedia Systems
S2024 Project

Group No. 15

Student Names:

Peddineni Rupesh chowdary (S20210010173)

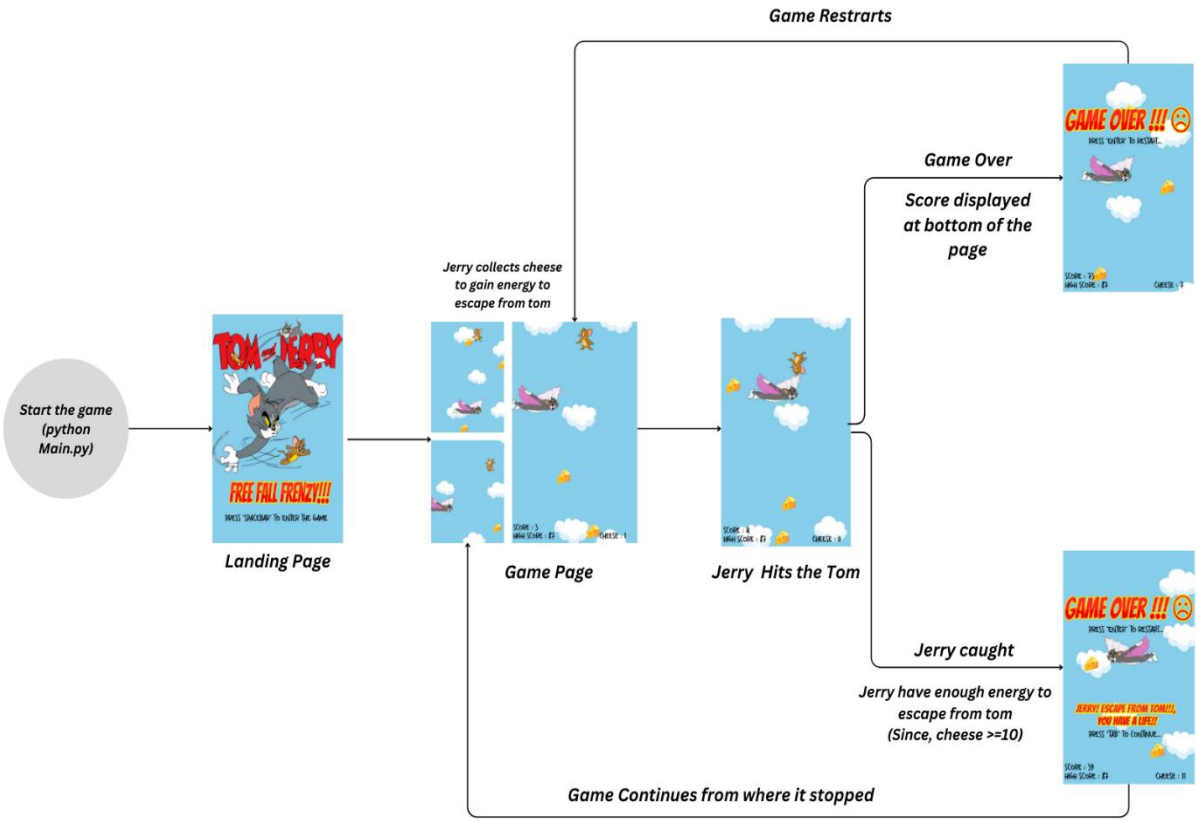
Somisetty Ramtej (S20210010211)

Project Title: *TOM AND JERRY FREE FALL FRENZY*

Introduction:

Welcome to ***'Tom and Jerry Free Fall Frenzy'***! Get ready for some high-flying fun as you join Jerry in his quest to escape Tom's clutches. In this game, Jerry falls from the sky, trying to avoid Tom while collecting cheese for extra lives. But watch out! Tom gets faster as you go down. Can you help Jerry escape? Let's dive into this exciting adventure filled with cheese, challenges, and endless excitement!

How to Play? :



Packages used :

- **Pygame Library**

- Initialization : “pygame.init()” is called to initialize all imported pygame modules. This is necessary before using any other pygame functions

-Screen Setup : “pygame.display.set_mode([WIDTH, HEIGHT])” creates a window with specified width and height for the game.

-**Image Loading** : Various images used in the game (such as clouds, Tom, and Jerry) are loaded using “pygame.image.load()”. These images are scaled and transformed as needed.

-**Rectangles :** “ pygame.rect.Rect()” is used to create rectangles for collision detection and drawing shapes.

-Sound : Sound effects and music are loaded and played using “pygame.mixer.Sound()” and “pygame.mixer.music.load()”

-Event Handling : The pygame.event.get() function retrieves a list of all the events t that have occurred since the last call. This is used to handle user input, such as keyboard keyboard events for controlling Jerry's movement and for quitting the game.

- **Random Module**

-Random Number Generation : “random.randint(a, b)” is used to generate a n random integer between a and b, inclusive. This is used for various purposes such as det determining initial positions of objects, generating random cloud types, and setting enemy spawn positions.

-**Random Choice** : random.choice(seq) is used to choose a random element from a s sequence (seq). This is utilized for selecting cloud images and determining whether to generate one or two clouds

-**Randomized Enemy Movement :** The speed of enemies is determined based on the current score using current_score//15, providing increasing difficulty as the game progresses.

-**Randomized Cloud Placement :** The placement of clouds and cheeses is random within certain constraints. Clouds are positioned at random heights within specified ranges and cheeses are placed randomly within cloud boundaries.

Screen Shots :

- **Draw_Clouds :**

```
# Clouds creation function
1 usage
def draw_clouds(cloud_coordinates_list, images):
    platforms = []
    for j in range(len(cloud_coordinates_list)):
        image = images[cloud_coordinates_list[j][2] - 1]
        platform = pygame.rect.Rect((cloud_coordinates_list[j][0] + 5, cloud_coordinates_list[j][1] + 30), (130, 10))
        screen.blit(image, dest: (cloud_coordinates_list[j][0], cloud_coordinates_list[j][1]))
        # pygame.draw.rect(screen, 'gray', platform)
        platforms.append(platform)

    return platforms
```

- **Draw_Cheese :**

```
1 usage
def draw_cheese(cheese_coordinates_list, cheese_image):
    cheese_platforms_list = []
    for j in range(len(cheese_coordinates_list)):
            image = cheese_image
            platform = pygame.rect.Rect((cheese_coordinates_list[j][0] + 20, cheese_coordinates_list[j][1] + 10),(40, 10))
            screen.blit(image, dest: (cheese_coordinates_list[j][0], cheese_coordinates_list[j][1]))
            # pygame.draw.rect(screen, 'gray', platform)
            cheese_platforms_list.append(platform)

    return cheese_platforms_list
```

- **Draw_Player :**

```
# Player creation function
1 usage
def draw_player(x_pos, y_pos, player_img, direc):
    if direc == -1:
        player_img = pygame.transform.flip(player_img, flip_x: False, flip_y: True)
    screen.blit(player_img, dest: (x_pos, y_pos))
    player_rect = pygame.rect.Rect((x_pos + 30, y_pos + 120), (50, 10))
    # pygame.draw.rect(screen, 'green', player_rect, 3)
    return player_rect
```

- **Draw_enemies :**

```
# Placing Enemies
1 usage
def draw_enemies(enemies_list, tom_img):
    enemy_rects = []
    for j in range(len(enemies_list)):
        enemy_rect = pygame.rect.Rect((enemies_list[j][0] + 10, enemies_list[j][1] + 80), (200, 90))
        # pygame.draw.rect(screen, 'orange', enemy_rect, 3)
        enemy_rects.append(enemy_rect)
        if enemies_list[j][2] == -1:
            screen.blit(tom_img,  dest: (enemies_list[j][0], enemies_list[j][1]))
        elif enemies_list[j][2] == 1:
            screen.blit(pygame.transform.flip(tom_img,  flip_x: 1,  flip_y: 0),  dest: (enemies_list[j][0], enemies_list[j][1]))
    return enemy_rects
```

- **Move_enemies :**

```
1 usage
def move_enemies(enemy_list, current_score):
    enemy_speed = 2 + current_score//15
    for j in range(len(enemy_list)):
        if enemy_list[j][2] == 1:
            if enemy_list[j][0] < WIDTH:
                enemy_list[j][0] += enemy_speed
            else:
                enemy_list[j][2] = -1

        elif enemy_list[j][2] == -1:
            if enemy_list[j][0] > -235:
                enemy_list[j][0] -= enemy_speed
            else:
                enemy_list[j][2] = 1

        if enemy_list[j][1] < -100:
            enemy_list[j][1] = random.randint(HEIGHT, HEIGHT + 500)

    return enemy_list
```

- **Uptade_objects :**

```python
def update_objects(cloud_list, play_y, enemies_list, cheese_list):
    lowest_cloud = 0
    update_speed = 10
    if play_y > 200:    # This is done to move the clouds up...
        play_y -= update_speed   # Until the play_y > 200 , the screen dont move up

        for j in range(len(enemies_list)):
            enemies_list[j][1] -= update_speed

        # This for loop is used to find the bottom most cloud coordinates
        for j in range(len(cloud_list)):
            cloud_list[j][1] -= update_speed
            lowest_cloud = max(lowest_cloud, cloud_list[j][1])

        for j in range(len(cheese_list)):  # this loop is to move the cheeses up...
            cheese_list[j][1] -= update_speed

        if lowest_cloud < 750:  # Screen Height is 800...then creates new clouds
            num_clouds = random.randint(1, 2)  # Randomly generating 1 or 2 clouds, that should be as we go down
            if num_clouds == 1:
                x_pos = random.randint(0, WIDTH - 70)
                y_pos = random.randint(HEIGHT + 100, HEIGHT + 300)
                cloud_type = random.randint(1, 3)
                cloud_list.append([x_pos, y_pos, cloud_type])
                cheese_list.append([(x_pos + 200) % WIDTH, y_pos - 50, 1])
            else:  # The 2 clouds should not overlap...So each cloud is seperated
                x_pos = random.randint(0, WIDTH // 2 - 70)
                y_pos = random.randint(HEIGHT + 100, HEIGHT + 300)
                cloud_type = random.randint(1, 3)
                cloud_list.append([x_pos, y_pos, cloud_type])  # Appending the new cloud coordinates to the existing list

                x_pos2 = random.randint(WIDTH // 2 + 70, WIDTH - 70)
                y_pos2 = random.randint(HEIGHT + 100, HEIGHT + 300)
                cloud_type2 = random.randint(1, 3)
                cloud_list.append([x_pos, y_pos, cloud_type])
                cloud_list.append([x_pos2, y_pos2, cloud_type2]) # Appending the new cloud's coordinates to the existing list

                cheese_list.append([random.randint(x_pos + 20, x_pos2 - 20), random.randint(HEIGHT + 200, HEIGHT + 300), 1])

    return play_y, cloud_list, enemies_list, cheese_list  # Returning the clouds coordinates and updated play_y value
    # We are directly updating the Clouds coordinates list...So no use of returning cloud_list
```

- **Landing_Page(While Loop) :**

```python
start_page = True
run = True
while start_page:   # Intro Page loop
    screen.fill(bg)  # Fill the display screen with bg
    timer.tick(fps)
    # screen.blit(bgImg, (0, 0))
    screen.blit(heading, dest: (0, 10))
    screen.blit(Intro, dest: (0, 50))
    screen.blit(title, dest: (60, 550))
    end_text = font.render( text: 'Press "SPACEBAR" to enter the game', antialias: True, color: 'black')
    screen.blit(end_text, dest: (50, HEIGHT - 120))
    for event in pygame.event.get():    # When Exit is pressed(Red color Cross)...While
        if event.type == pygame.QUIT:   # loop terminates
            start_page = False
            run = False
        if event.type == pygame.KEYDOWN:
            if event.key == pygame.K_SPACE:
                start_page = False

    if not pygame.mixer.music.get_busy():
        pygame.mixer.music.load('assets/Intro.mp3')
        pygame.mixer.music.play()

    pygame.display.flip()
```

- **Exit_Handler :**

```
    # Exit Handler
    for event in pygame.event.get():    # When Exit is pressed(Red color Cross)...While
        if event.type == pygame.QUIT:   # loop terminates
            run = False
        if event.type == pygame.KEYDOWN:
            if event.key == pygame.K_LEFT:
                x_direction = -1

            elif event.key == pygame.K_RIGHT:
                x_direction = 1

            if event.key == pygame.K_RETURN and game_over:  # Pressing Enter key
                game_over = False
                player_x = 240
                player_y = 40
                direction = -1
                y_speed = 0
                x_direction = 0
                score = 0
                Cheese = 0
                total_distance = 0
                enemies_coordinates = [[-234, random.randint( a: 400, HEIGHT - 100), 1]]
                clouds_coordinates = [[200, 100, 1], [50, 330, 2], [350, 330, 3], [200, 670, 1]]
                pygame.mixer.music.play()

            if event.key == pygame.K_TAB and Cheese // 10 >= 1 and game_over:  # when game over..if
                player_y = old_pos_y - 200
                game_over = False
```

- **Game_page(While Loop) :**

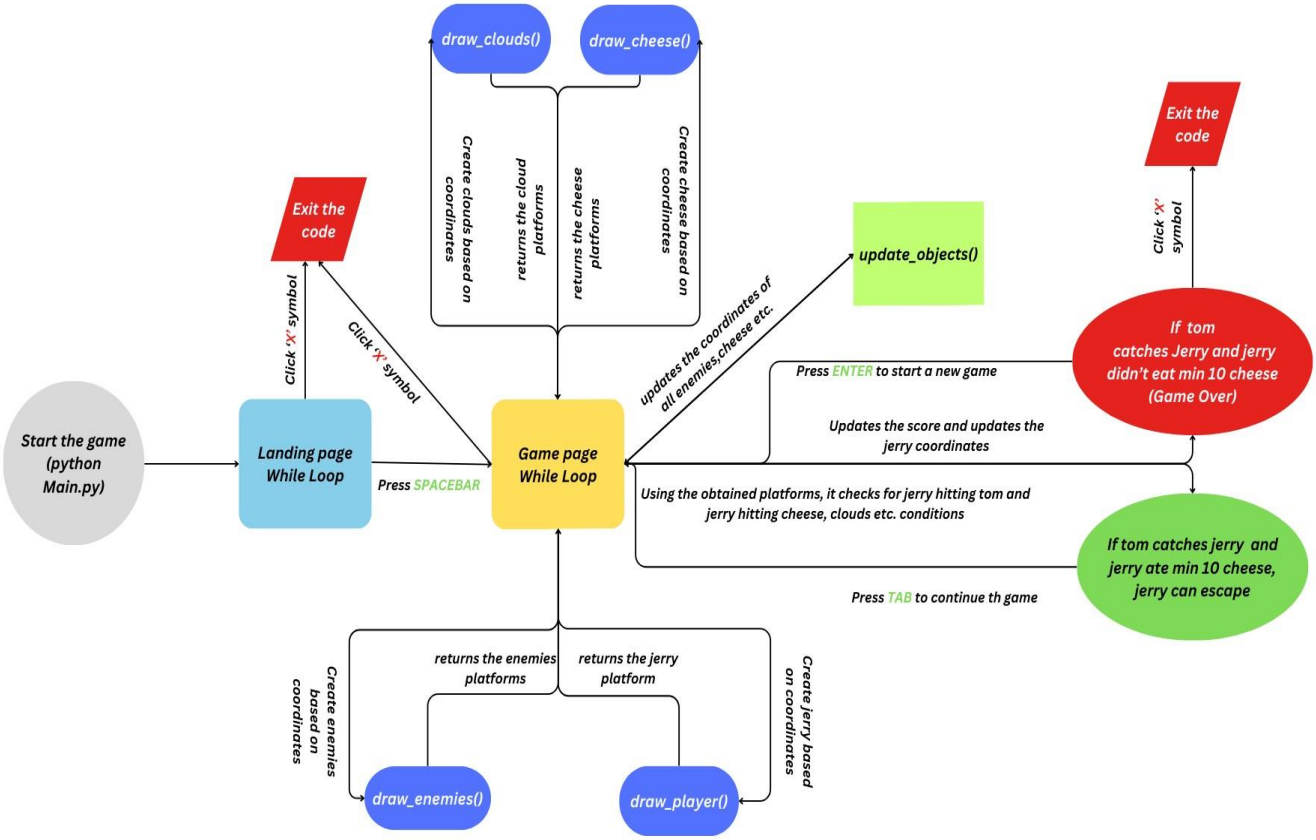
```
while run:  # Game page loop
    screen.fill(bg)  # Fill the display screen with bg
    timer.tick(fps)

    cloud_platforms = draw_clouds(clouds_coordinates, cloud_images)  # Gets back the cloud_platform coordin
    cheese_platforms = draw_cheese(cheese_coordinates, cheese_img)
    player = draw_player(player_x, player_y, jerry, direction)  # Gets back the player_rect
    enemy_boxes = draw_enemies(enemies_coordinates, tom)
    enemies = move_enemies(enemies_coordinates, score)
    player_y, clouds, enemies, cheeses = update_objects(clouds_coordinates, player_y, enemies_coordinates,

    if game_over:  # Enemy should disappear when tom and jerry touch...i.e. catch
        game_over_img = pygame.transform.scale(pygame.image.load('assets/gameover.png'), size: (500, 130))
        end_text = huge_font.render( text: 'Jerry\'s Free Fall Frenzy!!!', antialias: True, color: 'black')
        end_text2 = font.render( text: 'Press "Enter" to Restart...', antialias: True, color: 'black')
        screen.blit(game_over_img, dest: (0, 130))
        screen.blit(end_text2, dest: (100, 250))
        player_y = -300
        if Cheese // 10 >= 1:   # If cheese > 10 then...jerry gets a life
            life_img = pygame.transform.scale(pygame.image.load('assets/life.png'), size: (400, 100))
            end_text3 = font.render( text: 'Jerry! Escape from Tom!!!,You have a life!!', antialias: True, color:
            end_text4 = font.render( text: 'Press "Tab" to Continue...', antialias: True, color: 'black')
            screen.blit(life_img, dest: (40, 500))
            screen.blit(end_text4, dest: (100, 600))

    # Checking whether the jerry rectangle is colliding with any platform rectangle
    for i in range(len(cloud_platforms)):
        if direction == -1 and player.colliderect(cloud_platforms[i]):  # If it is in Free-Fall then change
```

Overall Work Flow:



Conclusion:

In conclusion, we built a vertical scrolling platformer game where the player controls Jerry, collecting cheese while avoiding obstacles represented by Tom. The game utilizes various features of Pygame such as image loading, collision detection, and event handling to create an interactive gaming experience. Players control Jerry's movement with the left and right arrow keys, aiming to collect cheese and avoid collisions with Tom. The game also includes elements like cloud platforms, score tracking, game over conditions, and a high score system, enhancing its gameplay depth and engagement.

Possible Future Extensions:

- **Level Design :**

  Create multiple levels with different layouts, obstacles, and challenges, gradually increasing in difficulty as the player progresses.

- **Multiplayer:**

  Include a multiplayer mode where players can compete against each other or collaborate in completing objectives.

- **Leader Boards** :

  Implement online leaderboards to allow players to compare their scores with others globally and compete for high rankings**.**