Final Project (Mini Games)

Project Description

Overviews

The project is about the two simple arcade games. There are snake arcade game and tic tac toe game. You can choose which game want to play at the user interface that appears when the program starts. At there, you can see three buttons that means which game want to play and quit. When you click the first button, the snake game will start and you can play. When that game is over, you will see another display that shows play again the same game or back to the main menu or quit from game. And, you can see the same logic when you play the tic tac toe game. Game logic is completely different. The same point is that when the game is over, screen that displayed the first game is over will appear again. Moreover, the background music, sound effects, button with colors, hovering colors and smooth user interface can be seen. For the snake game, you can see a snake moving its direction which is controlled by keyboard's key buttons. The food is random spawn and when the snake eat that its body will longer. When the snake hit the borders or collide itself, the game is over. For the Tic Tac Toe Game, you will play with computer as a competitor. You can mark with mouse cursor and if you have done, the computer will mark automatically and randomly after 2 seconds. If the same marks equal in column or in row or cross, that player wins. The purpose of this program is about the pygame, how to make buttons, graphic, sound, how to create beautiful and smooth user interface. The goal is simple arcade games that you can relax and play when you feel stressed out and exhausted.

Set up Instructions

- To run the program, you only need two requirements and set up instructions. Firstly, you need to install Python in your own laptop or computer.

- To download python, you can go to the official python website and can download the relative file that will suit with your device. E.g, window, mac or linux, etc.
- You also need to download pygame library.
- To download that, you can follow the similar instructions that were shown while installing the python.

Project Details

The project is about two simple arcade games**.** Snake game and Tic Tac toe Game. My programs has three main function. They will be snake game function , tic tac toe game function and main menu function.
To get a screen display, graphic, sound, input handling and game loop, I used the popular library 'Pygame' used primarily for game developments.

button.py - C:\Users\Hp\Desktop\MiniGamesByNick\button.py (3.12.0)

File Edit Format Run Options Window Help

```python
class Button():
        def __init__(self, image, pos, text_input, font, base_color, hovering_color):
                self.image = image
                self.x_pos = pos[0]
                self.y_pos = pos[1]
                self.font = font
                self.base_color, self.hovering_color = base_color, hovering_color
                self.text_input = text_input
                self.text = self.font.render(self.text_input, True, self.base_color)
                if self.image is None:
                        self.image = self.text
                self.rect = self.image.get_rect(center=(self.x_pos, self.y_pos))
                self.text_rect = self.text.get_rect(center=(self.x_pos, self.y_pos))

        def update(self, screen):
                if self.image is not None:
                        screen.blit(self.image, self.rect)
                screen.blit(self.text, self.text_rect)

        def checkForInput(self, position):
                if position[0] in range(self.rect.left, self.rect.right) and position[1] in range(self.rect.top, self.rect.bottom):
                        return True
                return False

        def changeColor(self, position):
                if position[0] in range(self.rect.left, self.rect.right) and position[1] in range(self.rect.top, self.rect.bottom):
                        self.text = self.font.render(self.text_input, True, self.hovering_color)
                else:
                        self.text = self.font.render(self.text_input, True, self.base_color)
```

Ln: 1 Col: 0

I made the button function including the click function, hovering colors, update and check for input functions.

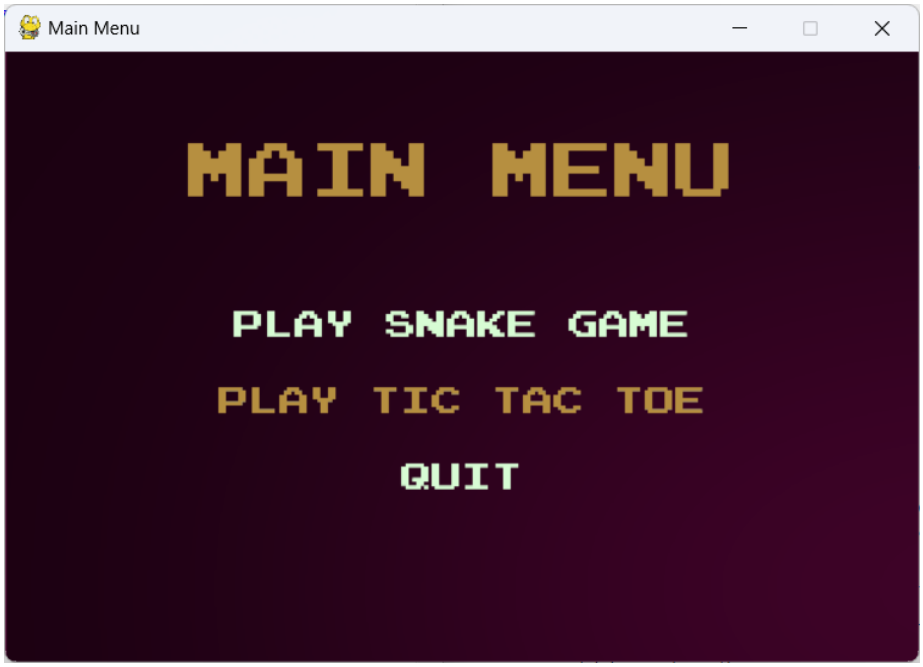
main.py - C:\Users\Hp\Desktop\MiniGamesByNick\main.py (3.12.0)

File Edit Format Run Options Window Help

```
#The Main Function of two games
def main_menu():
    pygame.display.set_caption("Main Menu")

    while True:
        SCREEN.fill("black")
        SCREEN.blit(BG, (0, 0))
        BGMUSIC.play()
        MENU_MOUSE_POS = pygame.mouse.get_pos()

        MENU_TEXT = get_font(40).render("MAIN MENU", True, "#b68f40")
        MENU_RECT = MENU_TEXT.get_rect(center=(300, 80))

        PLAY_SNAKE_BUTTON = Button(image=None, pos=(300, 180), text_input="PLAY SNAKE GAME", font=get_font(2
        PLAY_TICTAC_BUTTON = Button(image=None, pos=(300, 230), text_input="PLAY TIC TAC TOE", font=get_font
        QUIT_BUTTON = Button(image=None, pos=(300, 280), text_input="QUIT", font=get_font(20), base_color="#

        SCREEN.blit(MENU_TEXT, MENU_RECT)

        for button in [PLAY_SNAKE_BUTTON, PLAY_TICTAC_BUTTON, QUIT_BUTTON]:
            button.changeColor(MENU_MOUSE_POS)
            button.update(SCREEN)

        for event in pygame.event.get():
            if event.type == pygame.QUIT:
                pygame.quit()
                sys.exit()
            if event.type == pygame.MOUSEBUTTONDOWN:
                if PLAY_SNAKE_BUTTON.checkForInput(MENU_MOUSE_POS):
                    play_snake()
                if PLAY_TICTAC_BUTTON.checkForInput(MENU_MOUSE_POS):
                    play_tic_tac()
                if QUIT_BUTTON.checkForInput(MENU_MOUSE_POS):
                    pygame.quit()
                    sys.exit()

        pygame.display.update()

if __name__ == "__main__":
    main_menu()
```

Ln: 318 Col: 48

Using those button functions, I made the UI. When you hover the mouse over the three buttons, you can see its color change. When you click a button, it will run its own function.



The UI of main menu function

For the snake game;

main.py - C:\Users\Hp\Desktop\MiniGamesByNick\main.py (3.12.0)

File Edit Format Run Options Window Help

```
#For the Snake Game colors
foodColor = pygame.Color(238, 238, 238)
bgColor = pygame.Color(19, 75, 112)
snakeColor = pygame.Color(80, 140, 155)
fps = pygame.time.Clock()

#for snake properties
snake_speed = 15
snake_position = [100,50]
snake_body = [[100, 50], [90, 50], [80, 50], [70, 50]]

#Define food properties
food_position = [random.randrange(1,(window_x // 10))*10,random.randrange(1,(window_y // 10))*10]
food_spawn = True

#Set initial direction
direction = 'RIGHT'
change_to = direction

#Initial Score
score = 0

#Function to display score
def showScore_for_snake (choice,color,font,size):
    score_font = pygame.font.SysFont(font,size)
    score_surface = score_font.render('Score : '+str(score),True,color)
    score_rect = score_surface.get_rect()
    SCREEN.blit(score_surface, score_rect)
```

Ln: 338 Col: 0

First of all, I define the variables, colors, score and other requirements.

main.py - C:\Users\Hp\Desktop\MiniGamesByNick\main.py (3.12.0)

File Edit Format Run Options Window Help

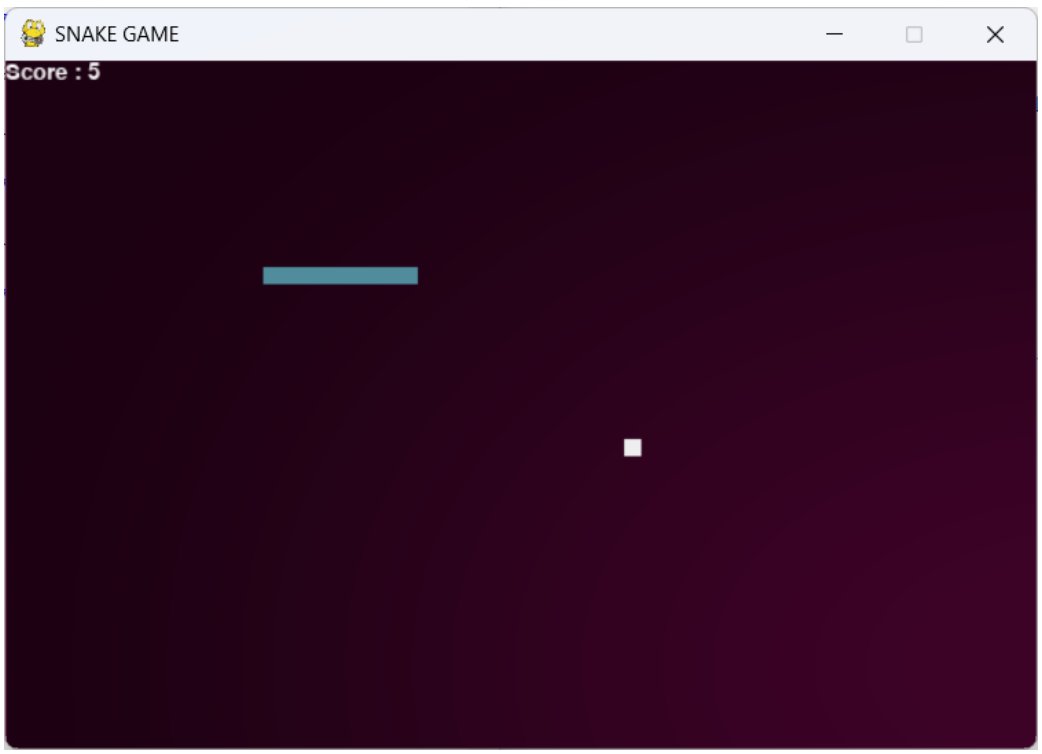
```
#main play snake game function
def play_snake():
    pygame.display.set_caption("SNAKE GAME")
    BGMUSIC.stop()
    SNAKEBGSOUND.play()
    global snake_position,snake_body,food_position,food_spawn,direction,change_to,score

    #Reset game variables
    snake_position = [100,50]
    snake_body = [[100, 50], [90, 50], [80, 50], [70, 50]]
    food_position = [random.randrange(1, (window_x // 10)) * 10, random.randrange(1, (window_y //
    food_spawn = True
    direction = 'RIGHT'
    change_to = direction
    score = 0
    while True:
        for event in pygame.event.get():
            if event.type == pygame.KEYDOWN:
                if event.key == pygame.K_UP:
                    if not direction == 'DOWN':
                        direction = 'UP'
                elif event.key == pygame.K_DOWN:
                    if not direction == 'UP':
                        direction = 'DOWN'
                elif event.key == pygame.K_RIGHT:
                    if not direction == 'LEFT':
                        direction = 'RIGHT'
                elif event.key == pygame.K_LEFT:
                    if not direction == 'RIGHT':
                        direction = 'LEFT'

        #if snake is moving in the direction, change to that direction
        if direction == 'UP':
            snake_position[1] -= 10
        if direction == 'DOWN':
            snake_position[1] += 10
        if direction == 'LEFT':
            snake_position[0] -= 10
        if direction == 'RIGHT':
            snake_position[0] += 10
```

```
        #snake body growing mechanism
        snake_body.insert(0,list(snake_position))
        if snake_position == food_position:
            score += 1
            FOODSOUND.play()
            food_spawn = False
        else:
            snake_body.pop()

        if not food_spawn:
            food_position = [random.randrange(1,(window_x//10))*10,random.randrange(1,(window_y//10))*10]
        food_spawn = True

        #background
        SCREEN.fill(bgColor)
        SCREEN.blit(BG, (0, 0))

        #Draw snake
        for pos in snake_body:
            pygame.draw.rect(SCREEN,snakeColor,pygame.Rect(pos[0],pos[1],10,10))

        #Draw food
        pygame.draw.rect(SCREEN,foodColor,pygame.Rect(food_position[0],food_position[1],10,10))
```

```
        # Game Over conditions
        if snake_position[0] < 0 or snake_position[0] > window_x - 10:
            return game_over_for_snake()
        if snake_position[1] < 0 or snake_position[1] > window_y - 10:
            return game_over_for_snake()

        for block in snake_body[1:]:
            if snake_position == block:
                return game_over_for_snake()

        # Display score
        showScore_for_snake(1, "white", 'assets/font.tff', 20)

        # Refresh game screen
        pygame.display.update()

        # Frame Per Second /Refresh Rate
        fps.tick(snake_speed)
```

Ln: 70 Col: 49

This is the main snake game function. I define the snake initial position, random food spawn position, the direction of the moving snake, speed, that when the snake eats the food, its body will longer, the game over when the snake hit the border or collide itself.



The user interface of the snake game playing

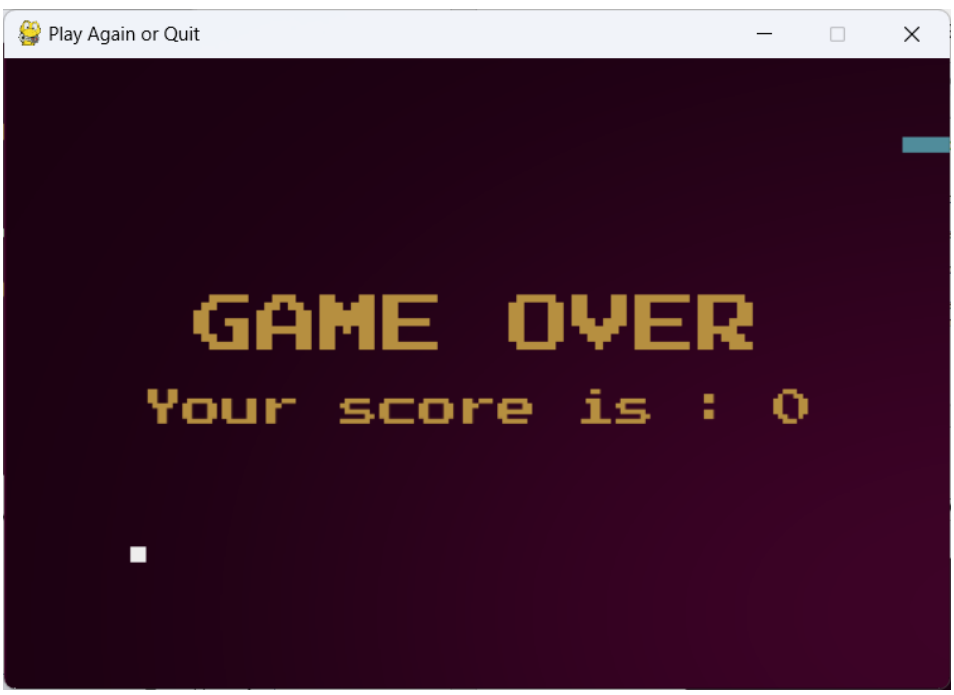
main.py - C:\Users\Hp\Desktop\MiniGamesByNick\main.py (3.12.0)

File Edit Format Run Options Window Help

```python
def game_over_for_snake():
    pygame.display.set_caption("GAME OVER")
    SNAKEBGSOUND.stop()
    HITSOUND.play()
    game_over_surface = get_font(40).render('GAME OVER',True,"#b68f40")
    game_over_surface1 =  get_font(25).render(f'Your score is : {score}',True,"#b68f40")
    game_over_rect = game_over_surface.get_rect()
    game_over_rect.midtop = (300,150)
    game_over_rect1 = game_over_surface1.get_rect()
    game_over_rect1.midtop = (300,210)
    SCREEN.blit(game_over_surface,game_over_rect)
    SCREEN.blit(game_over_surface1,game_over_rect1)
    pygame.display.flip()
    time.sleep(2)
    playagain_or_quit_for_snake()
```

Ln: 338 Col: 0

I made a function when the snake game is over, the game over screen and your score will be appear as below.



The UI of GAME OVER

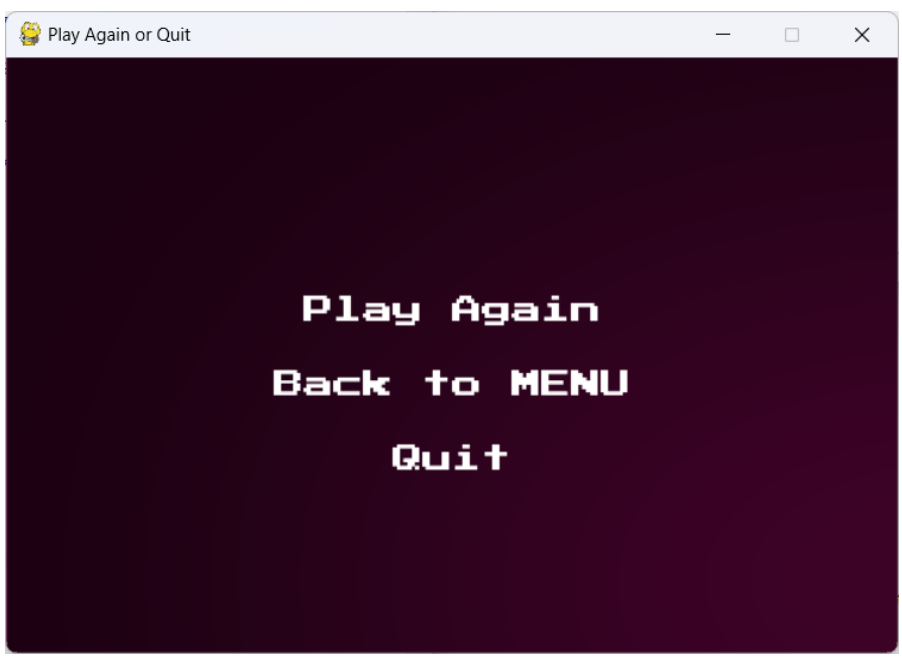
main.py - C:\Users\Hp\Desktop\MiniGamesByNick\main.py (3.12.0)

File Edit Format Run Options Window Help

```
#Playagain or back to main menu or quit
def playagain_or_quit_for_snake():
    pygame.display.set_caption("Play Again or Quit")
    BGMUSIC.play()

    #Ask the user if they want to play again or quit
    while True:
        CHOOSE_MOUSE_POS = pygame.mouse.get_pos()
        SCREEN.fill(bgColor)
        SCREEN.blit(BG, (0, 0))

        PLAY_AGAIN_BUTTON = Button(image=None, pos=(300, 170),text_input="Play Again", font=get_font(20), ba
        MENU_BUTTON = Button(image=None, pos=(300, 220),text_input="Back to MENU", font=get_font(20), base_c
        QUIT_BUTTON = Button(image=None, pos=(300, 270),text_input="Quit", font=get_font(20), base_color="Wh

        for button in [PLAY_AGAIN_BUTTON, QUIT_BUTTON,MENU_BUTTON]:
            button.changeColor(CHOOSE_MOUSE_POS)
            button.update(SCREEN)

        for event in pygame.event.get():
            if event.type == pygame.QUIT:
                pygame.quit()
                sys.exit()
            if event.type == pygame.MOUSEBUTTONDOWN:
                if PLAY_AGAIN_BUTTON.checkForInput(CHOOSE_MOUSE_POS):
                    play_snake()
                if MENU_BUTTON.checkForInput(CHOOSE_MOUSE_POS):
                    BGMUSIC.stop()
                    main_menu()
                if QUIT_BUTTON.checkForInput(CHOOSE_MOUSE_POS):
                    pygame.quit()
                    sys.exit()
        pygame.display.update()
```

Ln: 318 Col: 48

After a few second, you can choose play again that will run the main snake game program and back to menu that will run the main menu function and quit. The UI is as below.



For the tic tac toe game;

```
#For tic tac toe game
#Board setup
RED = pygame.Color(255,0,0)
board = [' ' for _ in range(9)]
player = 'X'
computer = 'O'
current_turn = player
computer_move_time = None  # Timer for computer's move

# Draw the grid
def draw_grid():
    SCREEN.fill("WHITE")
    SCREEN.blit(BG,(0,0))
    pygame.draw.line(SCREEN, main_color, (200, 0), (200, 400), 5)
    pygame.draw.line(SCREEN, main_color, (400, 0), (400, 400), 5)
    pygame.draw.line(SCREEN, main_color, (0, 133), (600, 133), 5)
    pygame.draw.line(SCREEN, main_color, (0, 266), (600, 266), 5)

# Draw the marks (X and O)
def draw_marks():
    for i in range(9):
        x = (i % 3) * 200 + 100
        y = (i // 3) * 133 + 66
        if board[i] == 'X':
            pygame.draw.line(SCREEN, RED, (x - 50, y - 50), (x + 50, y + 50), 10)
            pygame.draw.line(SCREEN, RED, (x + 50, y - 50), (x - 50, y + 50), 10)
        elif board[i] == 'O':
            pygame.draw.circle(SCREEN, secondary_color, (x, y), 50, 10)
```

At first, I define the required variables, make a function to draw a board with scale [3:3] and marks [“0”, “X”]

The UI of board and marks

main.py - C:\Users\Hp\Desktop\MiniGamesByNick\main.py (3.12.0)

File Edit Format Run Options Window Help

```python
# Check for a win or draw
def check_winner():
    win_conditions = [(0, 1, 2), (3, 4, 5), (6, 7, 8), (0, 3, 6), (1, 4, 7), (2, 5, 8), (0, 4, 8), (2, 4, 6)
    for a, b, c in win_conditions:
        if board[a] == board[b] == board[c] and board[a] != ' ':
            return board[a]
    if ' ' not in board:
        return 'Draw'
    return None

# Computer makes a move (random for simplicity)
def computer_move():
    empty_cells = [i for i, mark in enumerate(board) if mark == ' ']
    if empty_cells:
        move = random.choice(empty_cells)
        board[move] = computer

# Display the result
def display_winner(winner):
    SCREEN.fill("white")
    SCREEN.blit(BG,(0,0))
    if winner == 'X':
        text = get_font(25).render("Player Wins!", True, RED)
    elif winner == 'O':
        text = get_font(25).render("Computer Wins!", True, secondary_color)
    else:
        text = get_font(25).render("It's a Draw!", True, "White")
    SCREEN.blit(text, (window_x // 2 - text.get_width() // 2, window_y // 2 - text.get_height() // 2))
    pygame.display.update()
    pygame.time.delay(2000)

# Get the cell index based on mouse position
def get_cell_index(pos):
    x, y = pos
    row = y // 133
    col = x // 200
    return row * 3 + col
```

Ln: 276 Col: 0

Define the function who wins or draws, computer move, display winner and mark the box that player and computer choose.

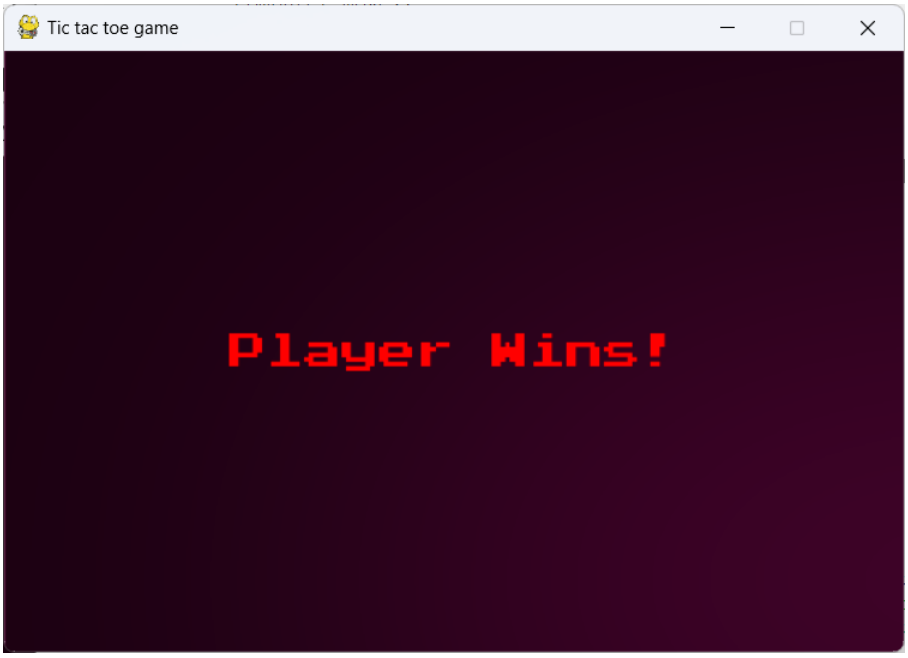
main.py - C:\Users\Hp\Desktop\MiniGamesByNick\main.py (3.12.0)

File Edit Format Run Options Window Help

```python
def play_tic_tac():
    pygame.display.set_caption("Tic tac toe game")
    BGMUSIC.stop()
    SNAKEBGSOUND.play()
    global current_turn, computer_move_time
    game_over = False

    while True:
        draw_grid()
        draw_marks()

        for event in pygame.event.get():
            if event.type == pygame.QUIT:
                pygame.quit()
                sys.exit()

            if event.type == pygame.MOUSEBUTTONDOWN and not game_over and current_turn == player:
                pos = pygame.mouse.get_pos()
                cell_index = get_cell_index(pos)
                if board[cell_index] == ' ':
                    board[cell_index] = player
                    FOODSOUND.play()
                    current_turn = computer
                    computer_move_time = pygame.time.get_ticks()  # Start the 2-second timer

        # Handle computer's move with a 2-second delay
        if current_turn == computer and not game_over:
            if computer_move_time and pygame.time.get_ticks() - computer_move_time >= 2000:  # 2 seconds del
                computer_move()
                current_turn = player
                computer_move_time = None  # Reset the timer

        winner = check_winner()
        if winner:
            SNAKEBGSOUND.stop()
            display_winner(winner)
            board[:] = [' ' for _ in range(9)]  # Reset board
            current_turn = player
            game_over = True
            playagain_or_quit_for_tic_tac_toe()
        pygame.display.update()
```

Ln: 297 Col: 0

This is the main function of the tic tac toe game including who wins.



The display of who wins

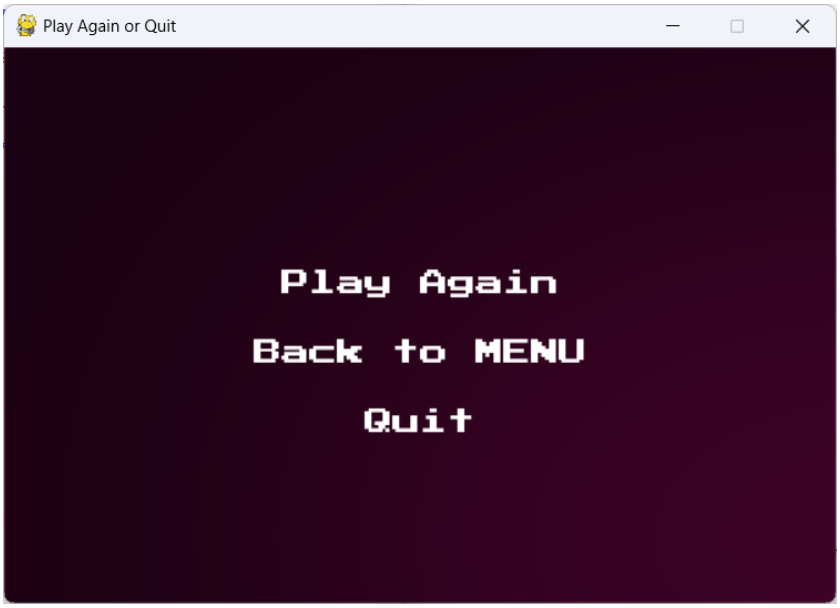
main.py - C:\Users\Hp\Desktop\MiniGamesByNick\main.py (3.12.0)

File Edit Format Run Options Window Help

```python
#play again or quit function
def playagain_or_quit_for_tic_tac_toe ():
    pygame.display.set_caption("Play Again or Quit")
    BGMUSIC.play()
    #Ask the user if they want to play again or quit
    while True:
        CHOOSE_MOUSE_POS = pygame.mouse.get_pos()
        SCREEN.fill(bgColor)
        SCREEN.blit(BG, (0, 0))

        PLAY_AGAIN_BUTTON = Button(image=None, pos=(300, 170),text_input="Play Again", font=get_font(20), base_color="White", hovering_color="#b68f40")
        MENU_BUTTON = Button(image=None, pos=(300, 220),text_input="Back to MENU", font=get_font(20), base_color="White", hovering_color="#b68f40")
        QUIT_BUTTON = Button(image=None, pos=(300, 270),text_input="Quit", font=get_font(20), base_color="White", hovering_color="#b68f40")

        for button in [PLAY_AGAIN_BUTTON, QUIT_BUTTON,MENU_BUTTON]:
            button.changeColor(CHOOSE_MOUSE_POS)
            button.update(SCREEN)

        for event in pygame.event.get():
            if event.type == pygame.QUIT:
                pygame.quit()
                sys.exit()
            if event.type == pygame.MOUSEBUTTONDOWN:
                if PLAY_AGAIN_BUTTON.checkForInput(CHOOSE_MOUSE_POS):
                    play_tic_tac()
                if MENU_BUTTON.checkForInput(CHOOSE_MOUSE_POS):
                    BGMUSIC.stop()
                    main_menu()
                if QUIT_BUTTON.checkForInput(CHOOSE_MOUSE_POS):
                    pygame.quit()
                    sys.exit()
        pygame.display.update()
```

Ln: 318 Col: 48

This is source code of buttons that you can choose play again that will run the main Tic Tac Toe game program and back to menu that will run the main menu function and quit.



The UI of play again or back to menu or quit

Other features

The UI of pygame is not good at all and I customized it by using background image. And, its positions work with (X,Y) position. There is no built-in buttons and I created buttons by using mouse position. I added the font, background music and sound effects.

Conclusion

To summarize, this program is made with three main functions,two game functions and main menu function. And also, it is made up with customize buttons, background, colors, font, background music and sound effects. When one game is over , you can play that again or go back to menu and choose the game that you want to play. If you feel bored playing game, you can competite with your friends by comparing scores or quit and enjoy your life.

References

Collab my ideas with the creative ideas that found on youtube and Generative AI

Enjoy The Game and Life!!!!!