**Documentation for Snake Game**

This documentation provides an overview and instructions for understanding and customizing the Snake Game implemented in Python using the tkinter library.

**Introduction**

The Snake Game is a simple, classic game where a snake moves around the screen, grows when it eats food, and ends when it collides with the walls or itself. This implementation provides a graphical user interface using tkinter.

**Game Features**

• **Graphics**: A graphical grid-based game board implemented using tkinter.Canvas.

• **Snake Movement**: Controlled by arrow keys (Up, Down, Left, Right).

• **Score Tracking**: The score increases as the snake eats food.

• **Game Over**: The game ends when the snake hits the walls or its own body.

**Constructor: \_\_init\_\_(self)**

Initializes the game, including the canvas, score, snake, food, and key bindings.

• **Canvas**: Creates the game window.

• **Snake**: Initializes the snake’s starting position.

• **Food**: Places food randomly on the canvas.

**Methods**

1. place\_food(self)

• Randomly generates a new food position on the game board.

• Ensures food is placed within the bounds of the canvas.

2. change\_direction(self, event)

• Updates the snake’s direction based on arrow key inputs.

• Ensures the snake cannot reverse into itself directly.

3. move\_snake(self)

• Moves the snake in the current direction by adding a new “head” segment and removing the “tail.”

• If the snake eats food, the tail is not removed, allowing the snake to grow.

4. check\_collision(self)

• Detects collisions with the walls or the snake’s own body.

• Returns True if a collision occurs, triggering the end of the game.

5. draw\_elements(self)

• Clears and redraws the canvas to update the positions of the snake, food, and score.

6. run\_game(self)

• The main game loop:

• Moves the snake.

• Checks for collisions.

• Updates the canvas elements.

• Continues the game unless self.game\_over is True.

**Methods**

1. place\_food(self)

• Randomly generates a new food position on the game board.

• Ensures food is placed within the bounds of the canvas.

2. change\_direction(self, event)

• Updates the snake’s direction based on arrow key inputs.

• Ensures the snake cannot reverse into itself directly.

3. move\_snake(self)

• Moves the snake in the current direction by adding a new “head” segment and removing the “tail.”

• If the snake eats food, the tail is not removed, allowing the snake to grow.

4. check\_collision(self)

• Detects collisions with the walls or the snake’s own body.

• Returns True if a collision occurs, triggering the end of the game.

5. draw\_elements(self)

• Clears and redraws the canvas to update the positions of the snake, food, and score.

6. run\_game(self)

• The main game loop:

• Moves the snake.

• Checks for collisions.

• Updates the canvas elements.

• Continues the game unless self.game\_over is True.

**Controls**

• **Arrow Keys**: Use the arrow keys (Up, Down, Left, Right) to control the direction of the snake.

**Game Rules**

1. **Objective**: Navigate the snake to eat as much food as possible, increasing the score and growing the snake.

2. **Food**: Eating food increases the score by 1 point and extends the snake’s body.

3. **Game Over**: The game ends when:

• The snake collides with the canvas boundary.

• The snake collides with its own body.

**Customization**

1. **Game Speed**:

• Adjust the GAME\_SPEED constant to increase or decrease the game’s pace (lower values = faster speed).

2. **Game Size**:

• Modify GAME\_WIDTH, GAME\_HEIGHT, and SQUARE\_SIZE for a different canvas size and segment dimensions.

3. **Colors**:

• Customize the snake, food, or background colors by changing SNAKE\_COLOR, FOOD\_COLOR, and BACKGROUND\_COLOR.

4. **Collision Logic**:

• Enhance the collision logic to include boundaries or wrap-around movement.

**Known Limitations**

• The game does not currently support pausing or restarting. Enhancements can be made by adding buttons or key bindings for these features.

**Further Enhancements**

Consider implementing the following features:

• **Pause/Resume Functionality**: Add a key binding to pause or resume the game.

• **Levels/Difficulty**: Increase the speed as the score increases.

• **Leaderboard**: Save and display high scores using file handling.