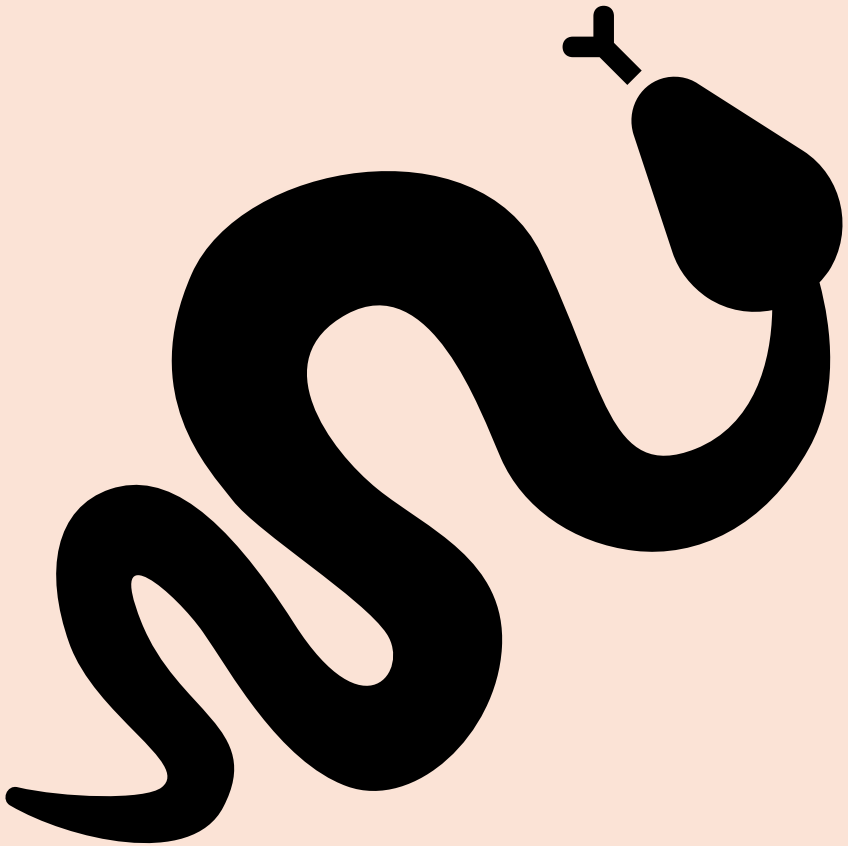
- **Title**: Snake Game Project

- **Name**: Nikunj and melvince



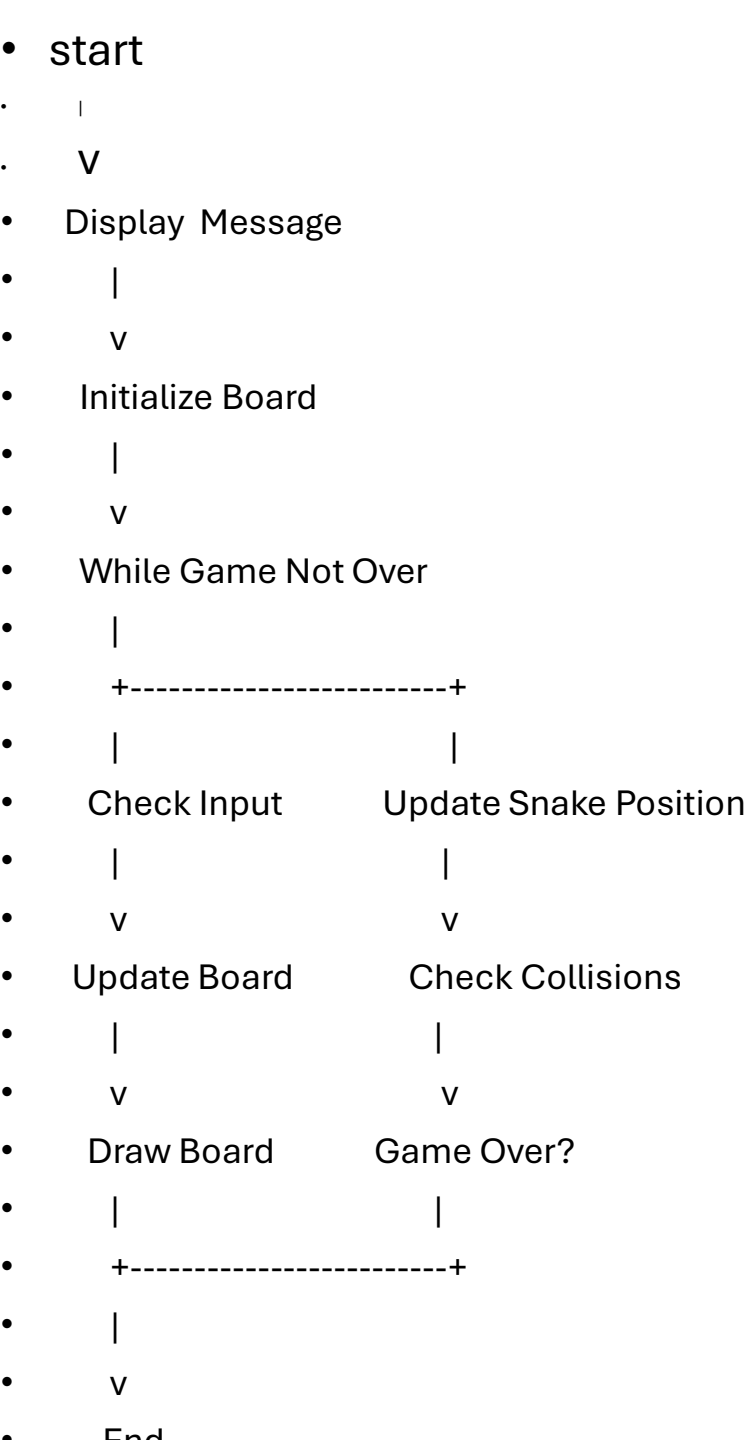
Overview – Design Goals

- **Objective**: Create an engaging Snake game using the C programming language.
- **Main Design Goals**:
  - **User Controls**: Easy-to-use keyboard controls for snake movement.
  - **Smooth Gameplay**: Quick response to player inputs.
  - **Scoring System**: Track the player's score and total accumulated score.
  - **Random Fruit Generation**: Fruit should appear in random locations on the game board.
  - **Collision Detection**: Identify when the snake hits the walls or itself.

Key Engineering Investigations and Findings

- **Game Loop**: The game operates in a continuous loop that keeps it running.
  - The loop checks for player input, updates the snake's position, and refreshes the board.
- **Control Directions**: Implemented simple controls for moving the snake:
  - **'W'**: Move Up
  - **'S'**: Move Down
  - **'A'**: Move Left
  - **'D'**: Move Right

- **Fruit Placement**: Fruits are placed randomly, ensuring they don't overlap with the snake or go outside the board.

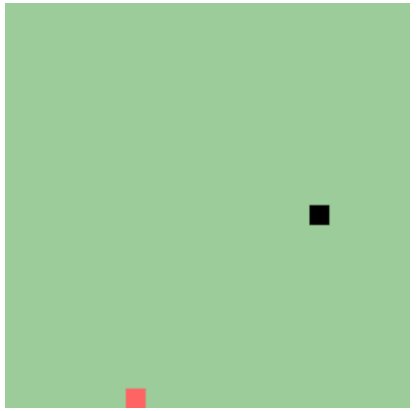

start
v
Display Message
|
v
Initialize Board
|
v
While Game Not Over
|
+-------------------------+
|
|
Check Input
Update Snake Position
|
|
v
v
Update Board
Check Collisions
|
|
v
v
Draw Board
Game Over?
|
|
+-------------------------+
|
v
End

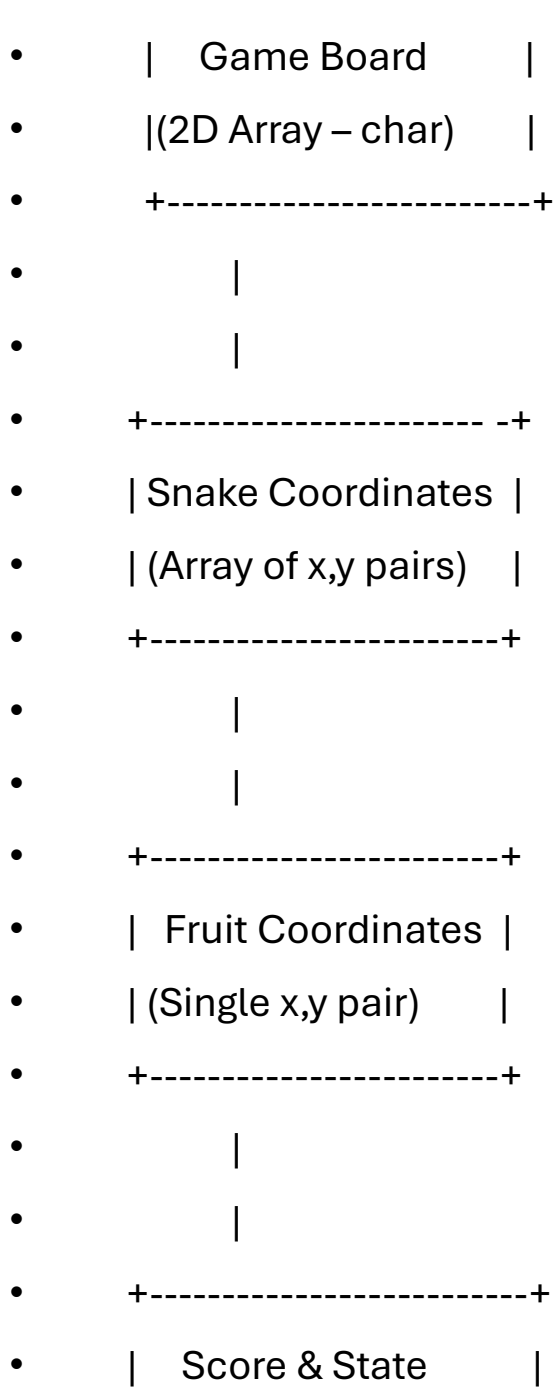
•**Arrays Used**:

•**Snake Coordinates**: Arrays store the x and y coordinates for each segment of the snake.
•This makes it easy to track its movement.

•**Game Board Representation**: A 2D array represents the game board visually, using
• characters to indicate the snake and the fruit.

•**Dynamic Length**:
•The snake grows longer as it eats fruits, which is managed by updating the array that
• holds its segments.




+------------------------+
| Game Board |
|(2D Array – char) |
+------------------------+
|
|
+---------------------- -+
| Snake Coordinates |
| (Array of x,y pairs) |
+------------------------+
|
|
+------------------------+
| Fruit Coordinates |
| (Single x,y pair) |
+------------------------+
|
|
+------------------------+
| Score & State |

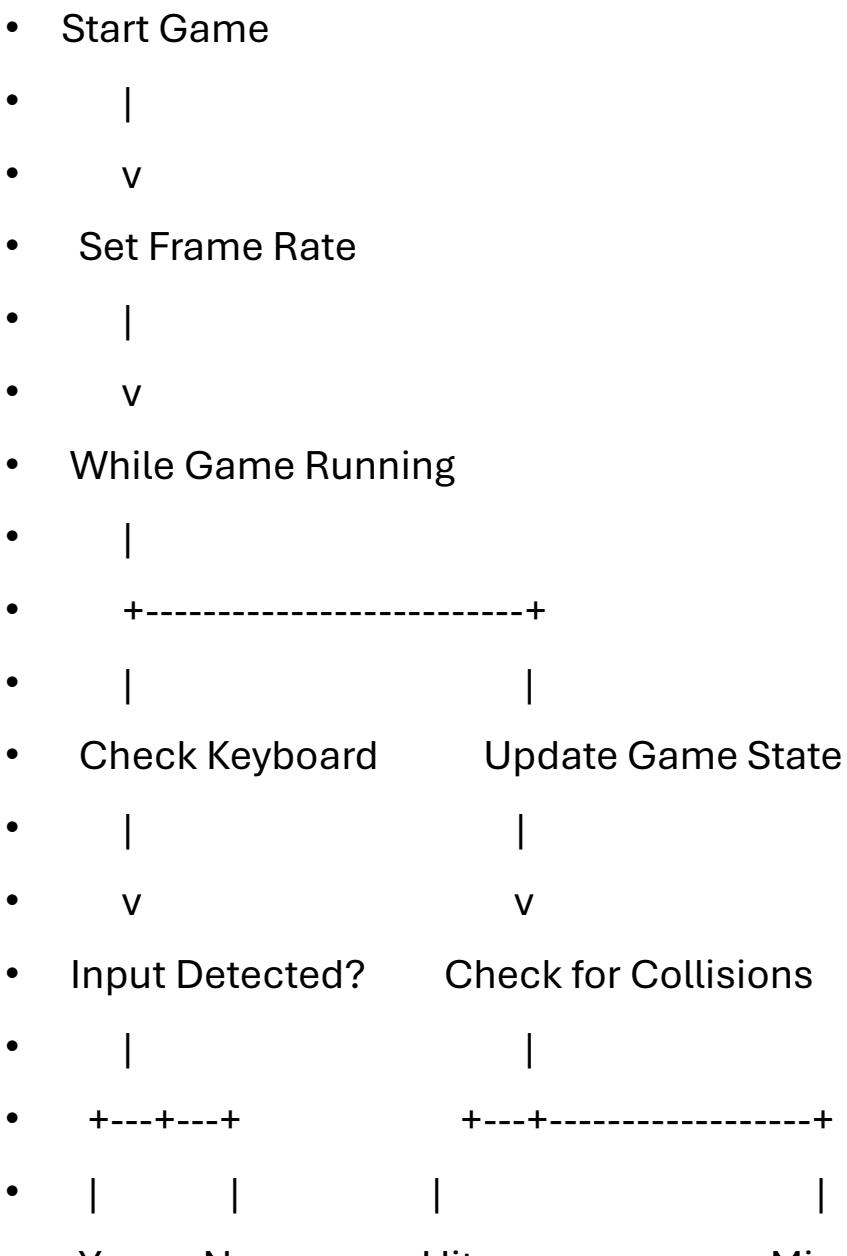
Performance Considerations

•**Game Speed Control**:

•The game manages how quickly it runs to maintain a fun but challenging experience for players.
•A delay is introduced between each move to control the snake's speed.

•**Input Handling**:
•Efficient input functions are used to quickly read user inputs, making the gameplay smooth and responsive.
•This helps in ensuring that player commands are registered without lag.

Flowchart: Game Performance Flow



Completed Prototype Demonstration

- **Demo of the Game:**
    - Welcome screen
    - Active gameplay
    - Game over screen

Potential Future Features

- **Future Enhancements**:
  - **Obstacles**: Add walls or barriers for added challenge.
  - **Difficulty Levels**: Allow players to choose speed or size of the game board.
  - **Power-Ups**: Include special items that provide advantages when eaten.
  - **High Scores**: Keep track of the highest scores for player motivation.
  - **Sound Effects**: Add audio feedback for actions like eating fruit or crashing.

Conclusion and Q&A

- **Conclusion**:
  - This project showcases how to build a simple Snake game in C, including handling user inputs and game mechanics.
  - It provides a strong foundation for developing more complex games in the future.