Terminal-Based Snake Game Documentation

Submitted By:

Zohra Amna 2022-SE-03

Muhammad Noman 2022-SE-31

Submitted To:

Dr. Farah Adeeba

Department of Computer Science

University of Engineering and Technology,

Lahore, New Campus

Table of Contents

Overview

The Terminal-Based Snake Game is a classic arcade game implemented in C programming language using the NCurses library. The game allows players to control a snake that moves around a bordered terminal screen, consuming food pellets to grow longer. The objective is to achieve the highest possible score without colliding with the snake's body or the screen border.

Project Structure

The Snake Game project consists of a single C source file (snakegame.c). This file contains the main game logic, including snake movement, collision detection, score tracking, and user interface rendering.

Source File (snakegame.c)

1. **Description:** The snakegame.c file contains the complete implementation of the Snake Game, including initialization, game loop, user input handling, and rendering.
2. **Technologies:** C programming language, NCurses library.
3. **Functionality:**
   - Initializes the game environment using NCurses library functions.
   - Manages the game state, including the snake's position, size, and score.
   - Implements the game loop, continuously updating the game state and rendering the game screen.
   - Handles user input to control the snake's movement (arrow keys) and pause/resume the game (space key).
   - Detects collisions between the snake and food pellets, as well as collisions with the screen border or the snake's own body.
   - Adjusts the snake's speed based on the player's score.
   - Displays the game screen with ASCII characters representing the snake, food pellets, and game status.
   - Provides visual feedback on the current score and game status.

Setup and Dependencies

1. **NCurses Library:** Ensure that the NCurses library is installed on your system. You can install it using package managers like apt (on Debian-based systems) or brew (on macOS).
2. **Compilation:** Compile the snakegame.c source file using a C compiler such as GCC. Make sure to link against the NCurses library using the -lncurses flag.

Usage

1. **Compilation:** Compile the snakegame.c source file using the provided compilation command.

```
gcc snakegame.c -o snake_game -lncurses
```

2. **Execution:**

   - Run the compiled executable (snake_game) in a terminal window.
   - Use the arrow keys to control the snake's movement.
   - Press the space key to pause/resume the game.
   - Press 'x' to exit the game.

Gameplay

1. **Controls:**

   - Arrow Keys: Control the snake's movement (up, down, left, right).
   - Space Key: Pause/Resume the game.
   - 'x' Key: Exit the game.
2. **Objective:** Eat food pellets to grow longer and achieve the highest score possible.
3. **Rules:**
   - The snake grows longer each time it eats a food pellet.
   - Colliding with the snake's own body or the screen border ends the game.
   - The game status and current score are displayed on the screen.
4. **Scoring:** The player's score increases each time the snake eats a food pellet. The score is displayed in the game interface.

Functionalities

1. Snake Movement

- **Description**: The snake's movement is controlled by the player using the arrow keys (up, down, left, right). The snake continuously moves in the direction specified by the player until another arrow key is pressed or the game is paused.
- **Implementation**: The controlSnake function handles user input and updates the snake's direction accordingly. It utilizes the NCurses library to detect arrow key presses and modify the flow variable representing the snake's direction.

2. Pause/Resume Game

- **Description**: Players can pause and resume the game at any time by pressing the space key. When paused, the game stops updating the snake's position and freezes the game screen.

- **Implementation**: The controlSnake function detects the space key press and toggles the game's pause state. While paused, the function waits for another space key press to resume the game.

3. Score Tracking

- **Description**: The game keeps track of the player's score, which increases each time the snake eats a food pellet. The score is displayed on the game screen and serves as an indicator of the player's progress.
- **Implementation**: The updateScore function updates the current_score variable whenever the snake eats a food pellet. The updated score is then rendered on the game screen for the player to see.

4. Collision Detection

- **Description**: The game detects collisions between the snake and various game elements, including food pellets, the screen border, and the snake's own body. Collisions with food pellets result in the snake growing longer, while collisions with the border or the snake's body end the game.
- **Implementation**: Collision detection is performed within the game loop. The game checks the snake's position against the coordinates of food pellets, the screen border, and the snake's body. If a collision is detected, the appropriate action is taken (e.g., growing the snake or ending the game).

5. Snake Speed Adjustment

- **Description**: The game dynamically adjusts the snake's movement speed based on the player's score. As the player's score increases, the snake's speed gradually changes, adding an element of challenge and excitement to the gameplay.
- **Implementation**: The adjustSpeed function utilizes a time-slice-based approach to adjust the snake's speed at regular intervals. The speed adjustment algorithm increases or decreases the time delay between successive snake movements, resulting in faster or slower gameplay.

6. User Interface Rendering

- **Description**: The game renders the user interface using ASCII characters, displaying the snake, food pellets, game status, and score on the terminal screen. The interface provides visual feedback to the player and enhances the gaming experience.
- **Implementation**: Rendering of the game interface is performed using NCurses library functions. ASCII characters representing game elements are drawn onto the terminal screen, creating a visually appealing and interactive environment for the player.

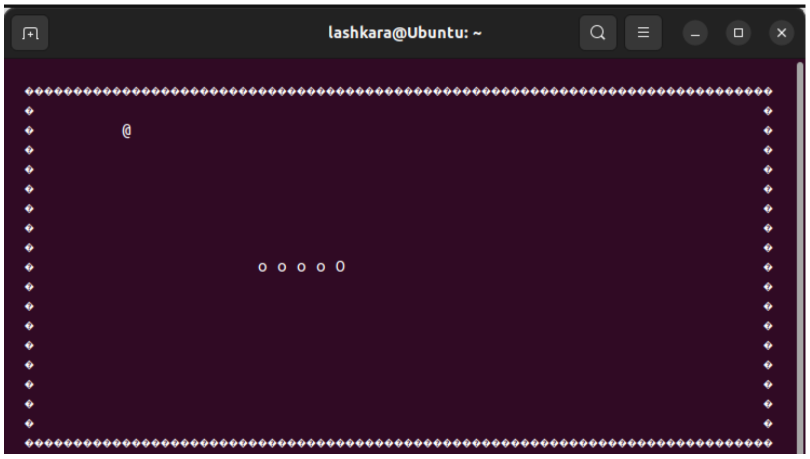
7. Game Over Handling

- **Description**: When the game ends due to a collision with the border or the snake's own body, the game displays a "Game Over" message and allows the player to restart the game. The player's final score is also shown.
- **Implementation**: Game over handling is triggered when a collision is detected within the game loop. Upon detecting a collision, the game displays the "Game Over" message, waits for a brief period, and then restarts the game with a new snake and food pellet arrangement.

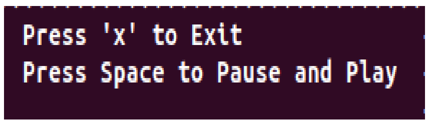
8. Client-Server Communication

- **Description**: We will be trying to add socket to code for client server communication so that the game supports multiplayer functionality through client-server communication using sockets. Multiple players can connect to a central server and play the game simultaneously, with the server managing game state synchronization and updates.
- **Implementation**: The snake_server.c and snake_client.c files implement the server and client components, respectively. The server listens for incoming connections from clients and broadcasts game state updates to all connected clients. Clients receive updates from the server and adjust their local game state accordingly.

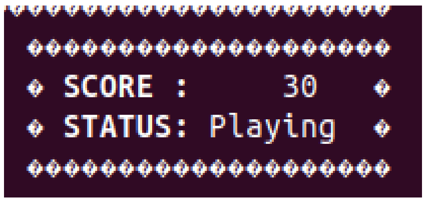
Game Screenshots



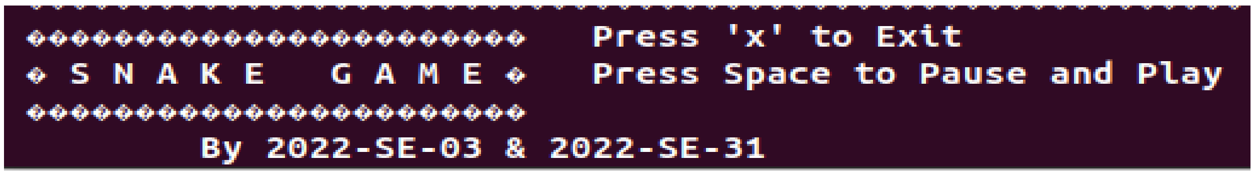
```
@

o o o o 0
```



```
Press 'x' to Exit
Press Space to Pause and Play
```

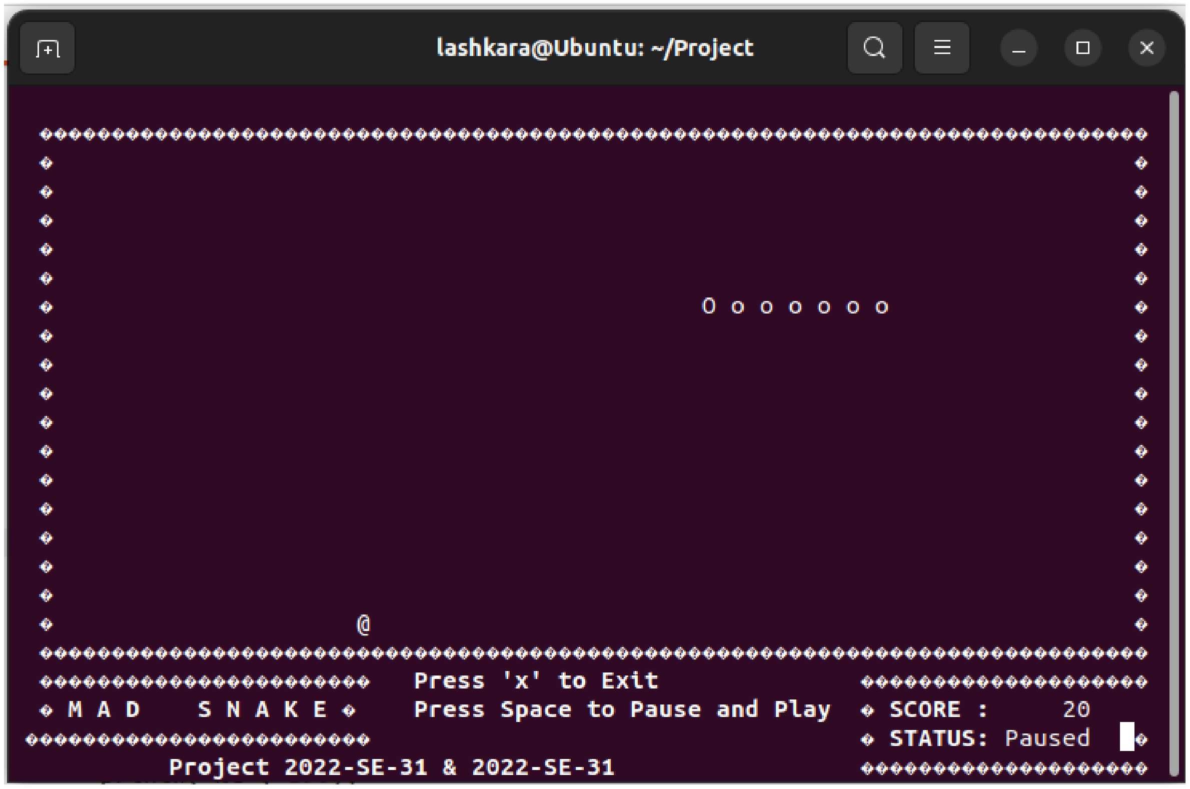


```
◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆
◆ SCORE :     30   ◆
◆ STATUS: Playing  ◆
◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆
```



```
◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆   Press 'x' to Exit
◆ S N A K E   G A M E ◆   Press Space to Pause and Play
◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆
         By 2022-SE-03 & 2022-SE-31
```

Operating System



```
�����������������������������������������������������������������������������
�                                                                           �
�                                                                           �
�                                                                           �
�                                                                           �
�                                                                           �
�                                             O o o o o o o                 �
�                                                                           �
�                                                                           �
�                                                                           �
�                                                                           �
�                                                                           �
�                                                                           �
�                                                                           �
�                                                                           �
�                                                                           �
�                                                                           �
�                     @                                                     �
�����������������������������������������������������������������������������
�����������������������   Press 'x' to Exit              ��������������������
� M A D    S N A K E �    Press Space to Pause and Play  � SCORE :     20
������������������������                                 � STATUS: Paused   �
         Project 2022-SE-31 & 2022-SE-31                 ��������������������
```

Conclusion

The Terminal-Based Snake Game offers a nostalgic gaming experience within a terminal environment. With its simple yet addictive gameplay and ASCII-based graphics, it provides hours of entertainment for players of all ages. Whether you're a beginner programmer looking to learn C or a seasoned developer seeking a fun project, the Snake Game is an excellent choice for honing your skills and creativity. Enjoy the challenge of controlling the snake and mastering the game's mechanics!