Snake Game in C++

A classic Snake game implementation in C++ using linked lists for the snake's body structure.

Features

- **Linked List Implementation**: The snake's body is represented using a custom linked list structure
- **Dynamic Growth**: Snake grows when eating food
- **Collision Detection**: Wall and self-collision detection
- **Score System**: Points awarded for eating food
- **Smooth Controls**: WASD movement controls

Game Rules

- Use **W**, **A**, **S**, **D** keys to move the snake
- Eat the food (*) to grow and increase your score
- Avoid hitting walls or your own body
- Press **X** to quit the game

Technical Details

Data Structures Used

- **Linked List**: Custom implementation for snake body segments
- **Node Structure**: Each segment contains x,y coordinates and pointer to next segment
- **Dynamic Memory Management**: Proper allocation and deallocation of nodes

Key Classes

- `SegmentNode`: Represents individual snake body segments
- `Snake`: Manages the linked list of segments and snake operations
- `SnakeGame`: Handles game logic, rendering, and user input

How to Compile and Run

Prerequisites

- C++ compiler (g++, Visual Studio, etc.)
- Windows OS (uses Windows-specific libraries for console operations)

Compilation

bash

```
g++ -o snake_game main.cpp
```

Running the Game

bash

```
./snake_game
```

Code Structure

```
snake-game/
├── main.cpp        # Main game implementation
├── README.md       # This file
├── Makefile        # Build configuration
└── .gitignore      # Git ignore file
```

Screenshots

```
####################################
#                  #
#                  #
#     O              #
#     o              #
#     o       *      #
#                  #
#                  #
####################################
Score: 30
Use WASD to move, X to quit
```

Features Implemented

- ✅ Linked list data structure for snake body
- ✅ Dynamic memory management
- ✅ Collision detection (walls and self)
- ✅ Food generation and consumption
- ✅ Score tracking
- ✅ Game over conditions
- ✅ Smooth gameplay controls

Learning Objectives

This project demonstrates:

- **Data Structures**: Implementation and usage of linked lists
- **Object-Oriented Programming**: Class design and encapsulation
- **Memory Management**: Dynamic allocation and deallocation
- **Game Development**: Basic game loop and state management
- **Console Programming**: Terminal-based user interface

Future Enhancements

- [ ] Cross-platform compatibility (Linux/Mac support)
- [ ] High score persistence
- [ ] Multiple difficulty levels
- [ ] Sound effects
- [ ] Color-coded snake segments
- [ ] Power-ups and special food items

Contributing

Feel free to fork this project and submit pull requests for improvements!

License