HW-4 Developer Document

Overview

Name: Snake game

Platform: MacOS

The Snake Game is a terminal-based interactive program where players control a snake to collect apples and avoid collisions.

Modules

1. Core Gameplay

Game loop: it provides the main game logic, updates game state, and renders the board.

Collision detection: It ensures that the game ends on self-collision, wall collision (or collisions between snakes in two-player mode).

2. Rendering

The program:

-Renders the game board, snakes, apples, and borders.

-Refreshes the terminal to show updated game state.

3. High Score Management

The program updates in real time and displays the highest score.

Data Structures

1. **Position Structure**

This structure represents the coordinates of a snake segment or an apple on the game board.

2. **Snake Node Structure**

The structure provides the information of snake's body: position and the pointer for the next node.

3. **Snake Structure**

The structure contains the information about head node, tail node, length of the snake and the movement direction.

Algorithms

Apple Placement

Randomly places the apple on the game field and avoid borders.

Collision Detection

It detects if the head of snake going throw boarder, self-body (or second snake's body in the two-players mode) during the game and finish it if the detection is happened.

3. Game Updates

The functions update the snake's position by creating a new head and add a new segment if the snake eats the apple.

If no apple is eaten moving the tail.

Functions

1. Core Gameplay

Snake* createSnake()

The function allocates memory for the snake, sets it's start coordinates and movement direction. It returns the pointer to an initialized Snake structure.

void placeApple(Position *apple)

The function gets the pointer to a Position structure and randomly spawn an apple within the board.

int updateSnake(Snake *snake, Position *apple, int *score)

The function gets the pointer to Snake, pointer to Position (apple), pointer to score.

It creates the new head for the snake and in case the apple is eaten, then the snake is increasing, otherwise the last node disappears. Also it checks collisions. The function is called every frame. It returns 1 if the snake collides with walls or itself, then the game is over and 0 otherwise and the game continue.

int checkSelfCollision(Snake *snake)

The function gets pointer to a Snake and returns 1 if the snake's head collides with its body and 0 otherwise.

int checkCollisionWithSnake(Snake *snake1, Snake *snake2)

The function gets pointer to a Snake and returns 1 if the snake's head collides with another snake and 0 otherwise.

2. Rendering

void display(Snake *snake, Position apple)

The function gets pointer to Snake and Position apple. It Renders the game board, boarders, snake and apples in single-player mode.

void displayTwoPlayers(Snake *snake1, Snake *snake2, Position apple)

The function gets pointers to both Snakes and Position apple. It Renders the game board, boarders, snake and apples in two-players mode.

3. High Score Management

void saveScore(int score)

The function gets score of the game and updates the high score file if the new score is higher.

void ReadScore()

The function reads and displays the high score from file.

4. Input Handling

void disableBufferedInput()

Disables canonical and echo mode for real-time input handling.

void enableBufferedInput()

Restores default terminal input settings.

int kbhit()

It non-blocking function to detect key presses. It returns 1 if a key is pressed and 0 otherwise.

5. Menu and Main Loop

void menu()

Displays the main menu.

void gameLoop()

It Implements the game loop for single-player mode. It is responsible for getting convert user's inputs to the direction and print the message of the game end. Also, there the developer can set the speed of the game with usleep () function.

void gameLoopTwoPlayers()

It implements the game loop for two-player mode. It is responsible for getting convert user's inputs to the direction and print the message of the game end. Also, there the developer can set the speed of the game with usleep() function.

This documentation provides a understanding for developers about code for the Snake Game.