**Semester Project Documentation**

**Project Name:**

**Snake Game**

**Students Details**

|  |  |  |  |
| --- | --- | --- | --- |
|  | Student Name | Student Reg # | Student Degree |
| Student-1 | M Rayan Ur Rehman Khan | 2023281 | FEE |
| Student-2 | Mohammad Abdullah | 2023324 | FEE |
| Student-3 | Mian Abdul Moeez | 2023314 | FEE |
| Student-4 | Malik Osama Feroz | 2023300 | FEE |

**Main Features**

1. *Object-oriented approach: The code utilizes classes and objects to organize data and functionality.*
2. *File handling: It includes file handling using the <fstream> library to read and write high scores.*
3. *Dynamic memory allocation: Dynamic arrays are allocated using new to store snake body coordinates.*
4. *User input and control: Functions like input () and control() handle user input and control snake movements.*
5. *Game mechanics: Various functions handle game mechanics such as checking for collisions, updating snake position, and controlling game over conditions.*
6. *Console manipulation: Functions like console\_set\_handler() and system("cls") are used to manipulate the console for displaying game output.*
7. *Randomization: The srand() function along with rand() is used for generating random fruit positions.*
8. *Visual feedback: The code provides visual feedback through colored console output to highlight important game elements like fruit and snake body.*
9. *Level selection: Users can select the game difficulty level, affecting the speed of the game loop.*
10. *High score tracking: The game tracks and updates high scores, which are stored in a file for persistent storage.*

**Types of Users**

1. *Players: Individuals who play the game.*
2. *Developers: Those who analyze or modify the code for educational or recreational purposes.*
3. *Administrators: Responsible for managing game settings or configurations.*
4. *Reviewers: Users who provide feedback or reviews about the game.*
5. *High score viewers: Individuals interested in viewing or comparing high scores.*
6. *Beginners: People who are new to programming and use the code as a learning resource.*
7. *Enthusiasts: Those who enjoy exploring and experimenting with different aspects of the game.*
8. *Educators: Teachers or instructors who may use the code as a teaching tool for programming concepts.*
9. *Researchers: Individuals studying game development or programming methodologies.*
10. *Casual users: Those who simply enjoy playing the game without much involvement in its development or customization.*

**Requirements Breakdown**

1. *Object-oriented approach:*
   1. *Define a class named data to encapsulate game data and functionality.*
   2. *Implement member variables to store game state such as snake position, fruit position, and game settings.*
   3. *Create member functions to access and modify game data.*
   4. *Ensure data encapsulation by making member variables private and providing public getter and setter functions.*
2. *File handling:*
   1. *Include the <fstream> library for file handling functionalities.*
   2. *Define functions to read and write high scores from/to a file.*
   3. *Handle file I/O errors gracefully, such as when the high score file is not found or cannot be accessed.*
3. *Dynamic memory allocation:*
   1. *Allocate dynamic arrays using the new keyword to store snake body coordinates.*
   2. *Release dynamically allocated memory using the delete [] operator to prevent memory leaks.*
4. *User input and control:*
   1. *Utilize functions from the <conio.h> library to handle keyboard input.*
   2. *Capture user input for controlling snake movement and pausing the game.*
   3. *Ensure input validation to prevent invalid or unintended inputs from affecting the game.*
5. *Game mechanics:*
   1. *Implement functions to control game flow, such as initializing game state, updating snake position, and checking for collisions.*
   2. *Handle scenarios like snake eating fruit, snake colliding with itself, and reaching game over conditions.*
   3. *Control game speed based on the selected difficulty level.*
6. *Console manipulation:*
   1. *Utilize functions like system("cls") to clear the console screen for updating game output.*
   2. *Modify console cursor visibility and position for a better user experience.*
   3. *Use ANSI escape codes for colorizing console output to differentiate game elements.*
7. *Randomization:*
   1. *Use the srand() function along with rand() to generate random fruit positions within the game boundaries.*
   2. *Ensure that generated fruit positions do not overlap with the snake's body.*
8. *Visual feedback:*
   1. *Implement colored console output to visually represent game elements such as snake body, fruit, and game over messages.*
   2. *Choose distinct colors for different game elements to enhance visibility and clarity.*
9. *Level selection:*
   1. *Provide users with options to select the game difficulty level, such as very easy, easy, medium, or difficult.*
   2. *Adjust game speed based on the selected difficulty level to increase or decrease the challenge.*
10. *High score tracking:*
    1. *Create a text file to store and update high scores achieved by players.*
    2. *Read and display high scores to users who want to view the top scores.*
    3. *Update high scores whenever a new record is achieved and save them to the high score file.*

**Features to Codding Matrix**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Sr  no. | Feature Name | Concept Used | Functions Created | Variables / Obj Created | Line of Code Written |
| 1 | Object oriented approach | Classes, inheritance | data(), srand\_fun(), control(), snake\_movement(), input(), set\_level(), game\_over(), rigid\_boundry\_fun(), check\_high\_scores() | Data object, snake movements | Approximately 150 lines |
| 2 | File handling | File I/O, error handling | check\_high\_scores() | Ifstream, ofstream | 30 lines |
| 3 | Dynamic memory allocation | Dynamic array, memory managment | Data, constructor | int body\_x, body\_y | 20 lines |
| 4 | User input and control | Keyboard input | input() | char move, move1 | 40 lines |
| 5 | Game mechanics | Game login, stste managment | control(),  snake\_movement() | int size | 100 lines |
| 6 | Console manupulation | Console output, cursor control | Control\_set\_handler() | handle cord | 20 lines |
| 7 | randomization | Random number generation | srand\_fun(), srand\_check() | int x | 20 lines |
| 8 | Visual feed back | Console coloring, output formatting | control() | bool body | 30 lines |
| 9 | Level selection | User input, difficulty setting | Set\_level | int level | 40 lines |
| 10 | High score tracking | File I/O, data managment | check\_high\_scores | int high\_scores[5] | Approximately 50 lines |