**comprehensive overview of the project, explaining the structure, functionality**

**Project Overview: Treasure Hunt Game**

This project is a terminal-based game where a player navigates through a map to find treasure while avoiding a snake. The game features a dark mode for increased difficulty and an undo functionality.

File Structure and Functionality:

1. main.c

- Contains the main game loop

- Handles command-line arguments

- Initializes the game

- Processes user input

- Calls functions to update and display the game state

2. game.h

- Defines game-related structures (GameState, UndoNode, GameManager)

- Declares constants for game objects (PLAYER, SNAKE, TREASURE, etc.)

- Declares function prototypes for game logic

3. game.c

- Implements game logic functions:

- init\_game: Initializes the game state from a map file

- update\_game: Updates the game state based on player moves

- is\_game\_over: Checks if the game has ended

- undo\_move: Reverts the game state to a previous move

- free\_game: Frees allocated memory

4. display.h

- Declares functions for displaying the game

5. display.c

- Implements display\_map function to render the game board

- Handles the visibility logic for dark mode

6. terminal.h and terminal.c

- Provide functions to handle terminal input (disableBuffer, enableBuffer)

7. random.h and random.c

- Implement random number generation for snake movement

8. newSleep.h and newSleep.c

- Implement a sleep function for timing control

9. game\_utils.h

- Defines utility macros, including IS\_VISIBLE for dark mode

10. makefile

- Manages the compilation process

- Handles conditional compilation for dark mode

Game Functionality:

1. Initialization:

- The game reads a map file specified as a command-line argument

- It creates a 2D char array representing the game board

- Positions of the player, snake, treasure, and lantern are set

2. Game Loop:

- The game continuously:

a. Displays the current game state

b. Waits for user input

c. Updates the game state based on input

d. Checks if the game has ended

3. Player Movement:

- The player can move up (w), down (s), left (a), or right (d)

- Movement is restricted by walls and map boundaries

4. Snake Movement:

- The snake moves randomly after each player move

- It can move in 8 directions (including diagonals)

- The snake cannot move through walls or off the map

5. Dark Mode:

- When enabled, limits the player's visibility to a certain range

- Uses Manhattan Distance to calculate visibility

- The lantern increases visibility range when collected

6. Undo Functionality:

- Allows the player to revert to previous game states

- Implemented using a linked list of game states

7. Win/Lose Conditions:

- The player wins by reaching the treasure

- The player loses if caught by the snake

Memory Management:

- The game uses dynamic memory allocation for the game board and undo states

- All allocated memory is freed at the end of the game or when no longer needed

Compilation and Execution:

- The game is compiled using the provided makefile

- Dark mode can be enabled during compilation with 'make DARK=1'

- The game is run with './treasure <map\_file>'

This project demonstrates key programming concepts including:

- File I/O

- Dynamic memory allocation

- Data structures (2D arrays, linked lists)

- Modular programming

- Makefiles and conditional compilation

- Terminal manipulation in C