```
=== MINESWEEPER PROJECT CONSOLIDATED ===
Generated: 09/27/2025 23:10:01
Files included: minesweeper_core.h, minesweeper_core.cpp,
minesweeper_terminal.cpp, main.cpp, projectTree.txt

================================================
FILE: minesweeper_core.h
================================================
// Game class declaration

#ifndef MINESWEEPER_CORE_H
#define MINESWEEPER_CORE_H

#include <vector>
#include <ctime>

class Minesweeper {
private:
      struct Cell {
            bool isMine = false;
            bool isRevealed = false;
            bool isFlagged = false;
            int adjacentMines = 0;
      };

      std::vector<std::vector<Cell>> grid;
      int rows, cols, totalMines;
      int revealedCells, flaggedMines;
      bool gameOver, gameWon;
      time_t startTime;
      bool gameStarted;

public:
      Minesweeper(int r = 10, int c = 10, int mines = 15);

      // Getters
      int getRows() const;
      int getCols() const;
      int getTotalMines() const;
      int getRevealedCells() const;
      int getFlaggedMines() const;
      int getRemainingMines() const;
      bool isGameOver() const;
      bool isGameWon() const;
      int getElapsedTime() const;

      // Cell queries
      bool isCellRevealed(int r, int c) const;
      bool isCellFlagged(int r, int c) const;
      bool isCellMine(int r, int c) const;
      int getCellAdjacentMines(int r, int c) const;

      // Game actions
      void revealCell(int r, int c);
```

```
      void toggleFlag(int r, int c);
      void resetGame();
      void showMines();

private:
      void initializeGrid();
      void placeMines();
      void calculateAdjacentMines();
      bool isValidCell(int r, int c) const;
      void startGame();
      void revealAllMines();
      void checkWinCondition();
};

#endif
```

```
================================================
FILE: minesweeper_core.cpp
================================================
// Game logic implementation

#include "minesweeper_core.h"
#include <iostream>
#include <random>
#include <algorithm>

Minesweeper::Minesweeper(int r, int c, int mines)
      : rows(r), cols(c), totalMines(mines), revealedCells(0),
      flaggedMines(0), gameOver(false), gameWon(false),
      gameStarted(false) {
      initializeGrid();
      placeMines();
      calculateAdjacentMines();
}

// Getters
int Minesweeper::getRows() const { return rows; }
int Minesweeper::getCols() const { return cols; }
int Minesweeper::getTotalMines() const { return totalMines; }
int Minesweeper::getRevealedCells() const { return revealedCells; }
int Minesweeper::getFlaggedMines() const { return flaggedMines; }
int Minesweeper::getRemainingMines() const { return totalMines -
flaggedMines; }
bool Minesweeper::isGameOver() const { return gameOver; }
bool Minesweeper::isGameWon() const { return gameWon; }

int Minesweeper::getElapsedTime() const {
      if (!gameStarted || gameOver) return 0;
      return static_cast<int>(difftime(time(nullptr), startTime));
}

// Cell queries
bool Minesweeper::isCellRevealed(int r, int c) const {
```

```
      return isValidCell(r, c) && grid[r][c].isRevealed;
}

bool Minesweeper::isCellFlagged(int r, int c) const {
      return isValidCell(r, c) && grid[r][c].isFlagged;
}

bool Minesweeper::isCellMine(int r, int c) const {
      return isValidCell(r, c) && grid[r][c].isMine;
}

int Minesweeper::getCellAdjacentMines(int r, int c) const {
      return isValidCell(r, c) ? grid[r][c].adjacentMines : 0;
}

// Private methods
void Minesweeper::initializeGrid() {
      grid.resize(rows, std::vector<Cell>(cols));
}

void Minesweeper::placeMines() {
      std::random_device rd;
      std::mt19937 gen(rd());
      std::uniform_int_distribution<> rowDist(0, rows - 1);
      std::uniform_int_distribution<> colDist(0, cols - 1);

      int minesPlaced = 0;
      while (minesPlaced < totalMines) {
            int r = rowDist(gen);
            int c = colDist(gen);

            if (!grid[r][c].isMine) {
                  grid[r][c].isMine = true;
                  minesPlaced++;
            }
      }
}

void Minesweeper::calculateAdjacentMines() {
      for (int i = 0; i < rows; i++) {
            for (int j = 0; j < cols; j++) {
                  if (!grid[i][j].isMine) {
                        int count = 0;
                        for (int di = -1; di <= 1; di++) {
                              for (int dj = -1; dj <= 1; dj++) {
                                    int ni = i + di;
                                    int nj = j + dj;
                                    if (isValidCell(ni, nj) &&
grid[ni][nj].isMine) {
                                          count++;
                                    }
                              }
                        }
                        grid[i][j].adjacentMines = count;
```

```
                }
            }
        }
}

bool Minesweeper::isValidCell(int r, int c) const {
      return r >= 0 && r < rows && c >= 0 && c < cols;
}

void Minesweeper::startGame() {
      if (!gameStarted) {
            startTime = time(nullptr);
            gameStarted = true;
      }
}

void Minesweeper::revealCell(int r, int c) {
      if (!isValidCell(r, c) || grid[r][c].isRevealed ||
            grid[r][c].isFlagged || gameOver) {
            return;
      }

      startGame();

      grid[r][c].isRevealed = true;
      revealedCells++;

      if (grid[r][c].isMine) {
            gameOver = true;
            revealAllMines();
            return;
      }

      if (grid[r][c].adjacentMines == 0) {
            for (int di = -1; di <= 1; di++) {
                  for (int dj = -1; dj <= 1; dj++) {
                        revealCell(r + di, c + dj);
                  }
            }
      }

      checkWinCondition();
}

void Minesweeper::toggleFlag(int r, int c) {
      if (!isValidCell(r, c) || grid[r][c].isRevealed || gameOver) {
            return;
      }

      startGame();

      if (grid[r][c].isFlagged) {
            grid[r][c].isFlagged = false;
            flaggedMines--;
```

```
	}
	else {
		grid[r][c].isFlagged = true;
		flaggedMines++;
	}
}

void Minesweeper::revealAllMines() {
	for (int i = 0; i < rows; i++) {
		for (int j = 0; j < cols; j++) {
			if (grid[i][j].isMine) {
				grid[i][j].isRevealed = true;
			}
		}
	}
}

void Minesweeper::checkWinCondition() {
	int totalCells = rows * cols;
	if (revealedCells == totalCells - totalMines) {
		gameWon = true;
		gameOver = true;
	}
}

void Minesweeper::resetGame() {
	revealedCells = 0;
	flaggedMines = 0;
	gameOver = false;
	gameWon = false;
	gameStarted = false;

	for (int i = 0; i < rows; i++) {
		for (int j = 0; j < cols; j++) {
			grid[i][j].isRevealed = false;
			grid[i][j].isFlagged = false;
			grid[i][j].isMine = false;
			grid[i][j].adjacentMines = 0;
		}
	}

	placeMines();
	calculateAdjacentMines();
}

void Minesweeper::showMines() {
	std::cout << "\nMine locations (for debugging):\n";
	for (int i = 0; i < rows; i++) {
		for (int j = 0; j < cols; j++) {
			if (grid[i][j].isMine) {
				std::cout << "(" << i << "," << j << ") ";
			}
		}
	}
```

```
      std::cout << "\n\n";
}

WARNING: minesweeper_terminal.cpp not found

================================================
FILE: main.cpp
================================================
// Menu + launches UIs

#include "minesweeper_core.h"
#include <iostream>
#include <tuple>
#include <ctime>
#include <fstream>
#include <string>

// Forward declarations for terminal functions
void displayGrid(const Minesweeper& game);
bool playTerminalGame(Minesweeper& game);

class MinesweeperMenu {
public:
      static void showMainMenu() {
            std::cout << "\n=== MINESWEEPER - CHOOSE INTERFACE ===\n";
            std::cout << "1. Terminal Version (Classic Text)\n";
            std::cout << "2. Qt GUI (Professional Desktop)\n";
            std::cout << "3. Web Browser (HTML/JavaScript)\n";
            std::cout << "4. SDL2 GUI (Lightweight Graphics)\n";
            std::cout << "5. Dear ImGui (Modern Interface)\n";
            std::cout << "6. Show Project Structure\n";          // ADD
THIS LINE
            std::cout << "7. Exit\n";                            //
CHANGE 6 to 7
            std::cout << "Choose interface: ";
      }

      static void showDifficultyMenu() {
            std::cout << "\n=== MINESWEEPER DIFFICULTY ===\n";
            std::cout << "1. Beginner (9x9, 10 mines)\n";
            std::cout << "2. Intermediate (16x16, 40 mines)\n";
            std::cout << "3. Expert (16x30, 99 mines)\n";
            std::cout << "4. Custom\n";
            std::cout << "5. Back to main menu\n";
            std::cout << "Choose difficulty: ";
      }

      static std::tuple<int, int, int> getDifficultySettings(int choice)
{
            switch (choice) {
            case 1: return std::make_tuple(9, 9, 10);
            case 2: return std::make_tuple(16, 16, 40);
            case 3: return std::make_tuple(16, 30, 99);
            case 4: {
```

```
                int rows, cols, mines;
                std::cout << "Enter rows: ";
                std::cin >> rows;
                std::cout << "Enter columns: ";
                std::cin >> cols;
                std::cout << "Enter number of mines: ";
                std::cin >> mines;

                if (mines >= rows * cols) {
                      std::cout << "Too many mines! Using maximum: " <<
(rows * cols - 1) << "\n";
                      mines = rows * cols - 1;
                }
                return std::make_tuple(rows, cols, mines);
          }
          default: return std::make_tuple(9, 9, 10);
          }
     }

     static void showProjectTree() {
          std::ifstream file("projectTree.txt");

          if (!file.is_open()) {
                std::cout << "\n=== PROJECT STRUCTURE ===\n";
                std::cout << "Error: projectTree.txt not found!\n";
                std::cout << "Make sure projectTree.txt is in the same
directory as the executable.\n\n";
                std::cout << "QUICK STRUCTURE:\n";
                std::cout << "- minesweeper_core.h/.cpp (Game logic)\n";
                std::cout << "- minesweeper_terminal.cpp (Terminal
UI)\n";
                std::cout << "- main.cpp (Menu system)\n\n";
                std::cout << "Press Enter to continue...";
                std::cin.ignore();
                std::cin.get();
                return;
          }

          std::cout << "\n";
          std::string line;
          while (std::getline(file, line)) {
                std::cout << line << "\n";
          }
          file.close();

          std::cout << "\nPress Enter to continue...";
          std::cin.ignore();
          std::cin.get();
     }

     static void startTerminalGame() {
          int choice;
          while (true) {
                showDifficultyMenu();
```

```
                std::cin >> choice;

                if (choice == 5) return;

                std::tuple<int, int, int> settings =
getDifficultySettings(choice);
                int rows = std::get<0>(settings);
                int cols = std::get<1>(settings);
                int mines = std::get<2>(settings);

                Minesweeper game(rows, cols, mines);
                bool returnToMenu = playTerminalGame(game);  // Handle
return value
                if (!returnToMenu) {
                      return; // User chose to quit completely
                }
                // If returnToMenu is true, continue the loop for
another game
          }
     }

     static void startQtGame() {
          std::cout << "\nStarting Qt GUI Version...\n";
          std::cout << "Note: Requires Qt library to be linked.\n";

          int choice;
          showDifficultyMenu();
          std::cin >> choice;

          if (choice == 5) return;

          std::tuple<int, int, int> settings =
getDifficultySettings(choice);
          int rows = std::get<0>(settings);
          int cols = std::get<1>(settings);
          int mines = std::get<2>(settings);

          std::cout << "Qt GUI would start here with " << rows << "x"
<< cols
                << " grid and " << mines << " mines.\n";
          std::cout << "Press Enter to continue...";
          std::cin.ignore();
          std::cin.get();
     }

     static void startWebGame() {
          std::cout << "\nStarting Web Browser Version...\n";
          std::cout << "This will generate HTML/CSS/JavaScript
files.\n";
          std::cout << "Press Enter to continue...";
          std::cin.ignore();
          std::cin.get();
     }
```

```
	static void startSDLGame() {
		std::cout << "\nStarting SDL2 GUI Version...\n";
		std::cout << "Note: Requires SDL2 library to be linked.\n";
		std::cout << "Press Enter to continue...";
		std::cin.ignore();
		std::cin.get();
	}

	static void startImGuiGame() {
		std::cout << "\nStarting Dear ImGui Version...\n";
		std::cout << "Note: Requires ImGui and graphics backend.\n";
		std::cout << "Press Enter to continue...";
		std::cin.ignore();
		std::cin.get();
	}

	static void run() {
		int choice;
		while (true) {
			showMainMenu();
			std::cin >> choice;

			switch (choice) {
			case 1:
				startTerminalGame();
				break;
			case 2:
				startQtGame();
				break;
			case 3:
				startWebGame();
				break;
			case 4:
				startSDLGame();
				break;
			case 5:
				startImGuiGame();
				break;
			case 6:                                  // ADD THIS CASE
				showProjectTree();
				break;
			case 7:                                  // CHANGE from case
6 to case 7
				std::cout << "Goodbye!\n";
				return;
			default:
				std::cout << "Invalid choice!\n";
			}
		}
	}
};

int main() {
	std::srand(std::time(nullptr));
```

```
      MinesweeperMenu::run();
      return 0;
}

================================================
FILE: projectTree.txt
================================================
=== MINESWEEPER PROJECT STRUCTURE ===

=== MINESWEEPER PROJECT STRUCTURE ===

CA_minesweeper/
|-- Header Files/
|   +-- minesweeper_core.h          Game logic interface
|-- Source Files/
|   |-- main.cpp                    Menu system & launcher
|   |-- minesweeper_core.cpp        Game logic implementation
|   |-- minesweeper_terminal.cpp    Terminal UI functions
|   +-- minesweeper_qt.cpp          Qt GUI (planned)
|-- Resource Files/
|   +-- projectTree.txt             This file
+-- Documentation/
    +-- (future: user manual, API docs)

=== IMPLEMENTATION STATUS ===
[X] Core Game Logic     - Complete (mines, reveal, flag, win/lose)
[X] Terminal Interface  - Complete (enhanced with help system)
[ ] Qt GUI Interface    - Planned (desktop application)
[ ] Web Interface       - Planned (HTML/CSS/JavaScript)
[ ] SDL2 Interface      - Planned (lightweight graphics)
[ ] ImGui Interface     - Planned (modern immediate-mode)

=== FILE RESPONSIBILITIES ===
minesweeper_core.h/.cpp:
  - Minesweeper class definition and implementation
  - Game state management, mine placement, cell logic
  - No UI dependencies - pure game logic

minesweeper_terminal.cpp:
  - Terminal display functions
  - User input handling for text interface
  - Game rules and help system

main.cpp:
  - Main menu system
  - Difficulty selection
  - Interface launcher (coordinates between UIs)

=== MODULAR DESIGN BENEFITS ===
1. Separation of Concerns - UI separate from game logic
2. Reusability - Same game logic for all interfaces
3. Testability - Can test game logic independently
4. Maintainability - Fix bugs in one place
5. Extensibility - Easy to add new interfaces
```