**Gem Odyssey Game Documentation**

**Introduction:**

Gem Odyssey is a simple puzzle game developed using C++ and the SFML library. The game involves swapping gems in an 8x8 grid to match three or more gems of the same type, scoring points and clearing them from the board. This C++ code implements a simple match-3 puzzle game using the SFML (Simple and Fast Multimedia Library). The game board consists of an 8x8 grid, and the player can swap adjacent gems to create matches of three or more horizontally or vertically. The code includes various game mechanics such as combo detection, special gems, and destroyers.

**Code Structure:**

**main() Function:**

- Initializes the main game window and sets up the menu.

- Defines menu buttons and handles keyboard input for navigation and menu selection.

- Calls functions based on the selected menu option (Play Game, Credits, How to Play, Exit).

**game() Function:**

- Sets up the game window, including the gem grid, timer, and score display.

- Handles user input for gem selection and swapping using arrow keys.

- Renders and updates the gem grid, score, and timer.

- Utilizes a matrix to represent the gem grid and swaps gems based on user input.

- Includes a countdown timer for the game duration.

**infoMenu() Function:**

- Displays the instructions on how to play the game.

- Provides information about using arrow keys for navigation and the enter key for gem selection.

**credMenu() Function:**

- Displays information about the developers of the game.

- Shows the names and student IDs of the developers.

**Graphics and Text:**

- Uses SFML for rendering graphics and text.

- Loads and displays background images for the main game and information screens.

- Displays buttons with text for the menu options.

- Implements a gem grid using different gem images based on the gem type.

**User Input:**

- Handles keyboard input for menu navigation and gem swapping.

- Arrow keys are used for menu navigation and gem swapping.

- The enter key is used to select gems in the game.

**External Resources:**

- Utilizes external resources such as images for gem types and background textures.

- Requires the SFML library for graphics and window management.

- Uses a custom font for text rendering.

**Error Handling:**

- Checks for font loading errors and exits the program if the font fails to load.

- Checks for background image loading errors and exits the program if the image fails to load.

**Game Mechanics**

**Gem Matching:**

The primary objective of the game is to match gems of the same type either horizontally or vertically. The `Check` function is responsible for detecting and handling these matches. The game loop repeatedly calls `Check` until no more matches are found.

**Special Gems:**

Special gems are created when a match of four or more gems occurs. These special gems, represented by values greater than 9, trigger additional effects when matched. The code includes logic to identify, handle, and detonate special gems.

**Destroyer Gems:**

Destroyer gems are a special type of gem that can eliminate entire rows and columns when matched. The `CheckDestroyer` function identifies and processes destroyer gems. When a destroyer gem is part of a match, it triggers the destruction of its corresponding row or column.

**Combo Detection:**

The code identifies combos of gems, ranging from basic three-gem matches to more complex five-gem matches. Combos result in different scores and may also create special or destroyer gems.

**Score System:**

The code includes functions for swapping gems (`CheckHor` and `CheckVer`), blasting rows and columns (`BlastHor` and `BlastVer`), and handling destroyer gem explosions (`BlastDestroyer`). These functions ensure that the game board is updated dynamically as gems are matched and destroyed.

**User Interface:**

The code includes a basic user interface setup with functions such as `createButton` for creating buttons with specified attributes and `loadTexture` for loading textures from files.

**Dependencies:**

The game relies on the SFML library for graphics and user input. Ensure that the necessary SFML headers and libraries are included in the project for successful compilation and execution.

**Conclusion:**

Gem Odyssey is a simple and visually appealing puzzle game with a straightforward user interface. Players can navigate through menus, read instructions, play the game, and view developer credits. The game provides an engaging experience through gem swapping mechanics and a countdown timer, challenging players to achieve a high score within the time limit.