**Bejewled Blitz**

Bejeweled Blitz is a fast-paced puzzle game that requires players to quickly line up colorful jewels and gems in a limited amount of time. Players attempt to create rows or columns with three or more identical gems by trading adjacent gems; when this is accomplished, the adjacent gems vanish and new gems fall from above.

**Features**

* Time-limited Gameplay
* Score Display After End Of Gameplay
* Top Score Bar
* Instructions Bar
* Credits Bar

**Credits**

* Uma E Rubab 23L\_0928
* Ayesha Khalid 23L\_0667

**How to Play:**

* Use the arrow keys (up, down, left, right) to navigate around the game board.
* The player will 60 seconds for one gameplay.
* Press enter to select the gem which you want to swipe.
* If three or more than three gems are similar then press enter again to cancel them.
* If 4 gems are matched either horizontally or vertically then the player will get the bonus score.
* If 5 gems are matched either horizontally or vertically then the player will get a special gem which will blast the whole board.

**Language and Libraries/ Setup and Installation:**

* Specify the programming language (C++ in this case) and any external libraries or frameworks used in the development.
* SFML library is used for graphics.
* Requirements: C++ compiler (e.g., g++), SFML library.
* Installation: Clone the repository, compile the code, and run the executable.

**About Code**

**Libraries used in the code:**

#include <SFML/Graphics.hpp>

#include <iostream>

#include<string.h>

#include<cstdlib>

#include <fstream>

**Functions used in the code:**

* void displayMainWindow(Font &font)

This function is related to the sfml library.This function is used to design the main menu which includes play,credits,instructions,top scores and exit.

* string getInputName( Font & font)

This function is used to take name of the player. When the player selects play option from main menu then the window "Enter your name" will open. After player inputs his/her name on pressing enter from keyboard this window will close.

* void saveScore(const string& name, int score)

This funtion is used to store the score of players in a file named scoresFile.

* void loadScores(string names[MAX\_PLAYERS], int scores[MAX\_PLAYERS])

This function is used to read the scores which are stored in the file scoresFile.

* void sortAccordingToScore(string names[MAX\_PLAYERS], int scores[MAX\_PLAYERS])

This function is used to arrange the scores in descending order.

* void displayTopScores(Font &font)

This function is used to display top score in main menu.

* void generateGem(int gemMatrix[gridSize][gridSize])

This function uses rand() to generate random gems in the board.

* void drawGemMatrixWindow(RenderWindow& window, Font& font, int gemMatrix[gridSize][gridSize], Texture gemTextures[], int space, int selectedX, int selectedY)

This function drawGemMatrixWindow is used to render or display a grid of gem sprites onto an SFML RenderWindow. It takes several parameters that help it to visualize the gem matrix grid and highlight a selected gem. It's a fundamental part of the game's graphical representation, showing the gems and allowing players to interact with them by visually indicating the selected gem for potential swaps or actions.

* void displayMatrix(int gemMatrix[gridSize][gridSize])

This function is used to display the matrix of 8x8 size.

* void checkForThreeMatches(int gemMatrix[gridSize][gridSize], int& score)

This function is used to check whether three gems are similiar either vertically or horizontally consecutively. If a match is found score increase by factor of 10. (1)

* void checkForFourMatches(int gemMatrix[gridSize][gridSize], int& score, int selectedX, int selectedY)

This function is used to check whether four gems are similiar either vertically or horizontally consecutively. If a match is found score increase by 100. Furthermore, the row and column of the selected gem will also be destroyed if a match of four gems is found. (2)

* void checkForFiveMatches(int gemMatrix[gridSize][gridSize], int& score)

This function is used to check whether five gems are similiar either vertically or horizontally consecutively. If a match is found score increase by 300. Furthermore, the whole table gets destroyed and a new table is generated. (3)

* void fillEmptySpaces(int gemMatrix[gridSize][gridSize])

This function is used to fill the empty spaces created by either of the three functions mentioned above {1,2,3}. As the gem gets destroyed, empty spaces are created which are filled using this function. Furthemore, this function uses rand() to generate new gems at empty spaces. (4)

* void checkAndDestroyMatches(int gemMatrix[gridSize][gridSize], int& score, int selectedX, int selectedY)

This functions combines all four functions mentioned above {1,2,3,4}. This function basically checks and generates new gems if needed.

* bool checkIfHorizontallySwapPossible(int gemMatrix[gridSize][gridSize]) (a)

This function ensures that gems will only move when there will be three or more matching gems horizontally, otherwise the selected gem will move to its original position.

* bool checkIfVerticallySwapPossible(int gemMatrix[gridSize][gridSize]) (b)

This function ensures that gems will only move when there will be three or more matching gems vertically, otherwise the selected gem will move to its original position.

* bool isValidSwap(int gemMatrix[gridSize][gridSize], int cursorX, int cursorY, int newCursorY, int newCursorX)

This function uses two helper functions mentioned above a and b. Via this function we can swap the gems both horizontally and vertically.

* bool intializeGame(RenderWindow &window, Sprite &backgroundSprite, Texture &backgroundTexture,Texture gemTextures[],int gemMatrix[gridSize][gridSize]) { if (!backgroundTexture.loadFromFile(GAME\_BACKGROUND))

It is responsible for initializing the game environment, including the window, background, gem textures, and the gem matrix.

* void handleGameEvents(RenderWindow& window, int gemMatrix[gridSize][gridSize], int& cursorX, int& cursorY, bool& enterPressed)

This function is divided into two parts “Handle game event” and “draw matrix”. Handle game events function handles game events, such as keyboard input for gem swapping. Draw Matrix funtion is responsible for drawing the gem matrix on the game window. It uses the SFML graphics library to draw gem sprites and a bounding box around the selected gem.

* void playGame(Font &font)

This function essentially orchestrates the entire game loop, handling user input, updating game states, rendering visuals, and managing the game's flow until the game ends due to a timeout or an error during initialization.

* void displayTimer(RenderWindow& window, sf::Font& font, int time)

This function essentially sets up the timer text with the remaining time and renders it onto the game window at a specific position, allowing the player to view the countdown during gameplay.

* void displayScore(sf::RenderWindow& window, sf::Font& font, int score)

This function sets up the score text with the player's current score and renders it onto the game window at a specific position, allowing the player to view their score during gameplay.

* void displayRemarks(RenderWindow& window, Font& font, string remarks)

This function sets up the remarks text with the provided message and renders it onto the game window at a specific position, allowing the display of information or messages for the player during gameplay.

* void displayGameEnd(Font& font, int& score)

This function creates a game over screen with a "Game Over" message, displays the player's score, and offers a "Quit" button that, when clicked, closes the game window and potentially triggers actions to return to the main game or exit entirely.

* void showInstructions(Font& font)

This function essentially creates an instructions window with text displaying game instructions and a "Back" button to return to the main game when clicked.

* void showCreditsWindow(Font& font)

This function essentially creates an credits window with text displaying credits and a "Back" button to return to the main game when clicked.