Minesweeper Spiel - Funktionsweise

Das Minesweeper-Spiel wurde in C++ unter Verwendung des Qt Frameworks entwickelt und bietet eine benutzerfreundliche Oberfläche zur Steuerung eines klassischen Spiels. Das Programm besteht aus mehreren Modulen, die verschiedene Funktionen wie Spiellogik, Benutzeroberfläche, Statistikverwaltung und Einstellungen abdecken.

1. Spiellogik (Game) Das Kernmodul des Spiels ist die `Game`-Klasse(Game engine). Die Klasse Game steuert das Spiel auf dem logischen Niveau: verwaltet das Spielfeld, die Spielmechanik wie Minenlegung, Zellenöffnung und -markierung sowie die Kontrolle von Spielendebedingungen. Zudem verwaltet sie den aktuellen Status des Spiels. Abhängigkeiten: Die Game-Klasse wurde so entworfen, dass die Gamelogik und Visualisierung maximal von einander getrennt sind. Das Modul ist aber in "mainwindow" Modul verwendet, damit die beide Spielebene verbunden werden.

2. Benutzeroberfläche (MainWindow) Die Hauptbenutzeroberfläche des Spiels wird in der `MainWindow`-Klasse implementiert. Sie ist für die Anzeige des Spielfeldes und der Benutzerinteraktionen verantwortlich. Die Benutzeroberfläche enthält:
- Buttons für die Steuerung des Spiels
-Statusbar, das über wichtige Spielaspekte informiert
    - Wie viele Minen in der jetzigen Spieleinstellung sind.
    - Wie viele Flagen schon gesetzt wurden.
    - Wie viele Sekunden seit dem Spielbeginn vergangen ist.

-Menu, mit folgenden Aktionen:
    - Einstellungen zur Anpassung des Spiels (Mit Hilfe des Moduls SettingsDialog)
    - Anzeigen von Statistiken(Mit Hilfe des Moduls StatisticsDialog)
    - Anzeigen von Hilfefenster (Mit Hilfe des Moduls HelpDialog)

Das Spielfeld wird als ein GridLayout von `QMinerPushButton`-Instanzen(QMinerPushButton-Klasse) angezeigt, wobei jeder Button eine Zeilen- und Spalteninformation enthält. Die Buttons reagieren auf Mausereignisse, und durch Klicks wird die Spiellogik aktiviert.

3. Anpassung des Spiels (SettingsDialog) Der `SettingsDialog` ermöglicht dem Benutzer, die grundlegenden Spielfeldparameter wie die Anzahl der Minen, die Spielfeldgröße (Länge und Breite) sowie den Schwierigkeitsgrad anzupassen. Der Dialog enthält:
- Ein ComboBox, wo man eine Schwierigkeit wählen kann
- Eingabefelder für die Spielparameter
- Slider zur Anpassung der Einstellungen
- Ein Button zum Bestätigen der Auswahl und Speichern der Änderungen Durch diesen Dialog können die Benutzer das Spiel auf ihre Bedürfnisse zuschneiden. Alle Änderungen werden sofort im Spiel übernommen.

4. Statistiken (StatisticsDialog) Das `StatisticsDialog` zeigt die Statistiken der gespielten Spiele an. Es werden Daten wie die Anzahl der gespielten Spiele, die gewonnenen Spiele, verlorenen Spiele und die schnellste Zeit zum Beenden eines Spiels angezeigt. Die Statistiken werden in einer JSON-Datei gespeichert und beim Starten des

Spiels geladen. Die Tabelle wird dynamisch mit den geladenen Werten befüllt und zeigt die Fortschritte des Spielers. Die statistischen Daten werden in einer `QTableWidget` angezeigt.

5. Hilfefenster (HelpDialog) Das `HelpDialog` zeigt den Spieler alle nötige Informationen über die Aktionen mit den Spielfeldern, wie:
-Rechter Mausklick
-Mittlerer Mausklick
-Linker Mausklick
und zeigt was die Icons von den Feldern eigentlich bedeuten.
Es verfügt auch über ein goBackButton, womit wieder Hauptfenster angezeigt wird.

6. Benutzerdefinierte Buttons (QMinerPushButton) Die `QMinerPushButton`-Klasse ist eine erweiterte Version des `QPushButton` und behandelt spezifische Mausklickereignisse wie Links-, Rechts- und Mittelklicks. Jeder Button repräsentiert ein Feld im Spielfeld und ist mit einer Zeilen- und Spaltennummer verknüpft. Die `QMinerPushButton`-Klasse ermöglicht es, bei einem Klick ein bestimmtes Ereignis (z. B. das Öffnen eines Feldes) auszulösen.

###7. Zellen (Cell-Klasse). Die Klasse Cell repräsentiert eine einzelne Zelle im Minesweeper-Spiel.
Jede Zelle kann verschiedene Zustände haben, wie versteckt, markiert,
vermint oder explodiert. Zusätzlich speichert sie die Anzahl der benachbarten Minen.
Abhängigkeiten: Die Cell-Klasse ist unabhängig und wird nur im Modul Game verwendet, um das Spielfeld zu initialisieren und die Zustände der einzelnen Zellen auf dem Spielfeld zu verwalten.
Technische Entscheidung: Die Cell-Klasse verwendet einfache boolesche Flags, um die Effizienz und Lesbarkeit zu erreichen. Eine komplexere Lösung wie Enums war für dieses Szenario nicht notwendig.