Dokumentacja użytkownika komponentu SaperComp

Wprowadzenie

Komponent `SaperComp` to klasyczna gra w sapera zaimplementowana w bibliotece Swing (Java). Gracz odkrywa pola na planszy, unikając min, i oznacza potencjalne miny flagami. Gra obsługuje trzy poziomy trudności (łatwy, średni, trudny), zapisuje wyniki oraz ustawienia w plikach konfiguracyjnych. Komponent jest konfigurowalny za pomocą klasy `SaperConfig`.

Wymagania

- **Środowisko**: Java 8+ z biblioteką Swing.
- **Zależności**: Standardowa biblioteka Javy (pakiety `javax.swing`, `java.awt`).
- **Pliki konfiguracyjne**: Plik `saperConfig.txt` (zapisywany w katalogu `~/.saper/`) dla ustawień gry oraz `saperScores.txt` dla wyników.
- **Zasoby**: Ikony `bomba2.png` i `flaga2.png` w folderze `/icons/` w zasobach projektu oraz opcjonalny plik konfiguracyjny `saperConfig.txt` w folderze `/`.

Instalacja

1. Skompiluj kod źródłowy komponentu (`SaperComp.java`, `SaperConfig.java`) w projekcie Java.
2. Upewnij się, że ikony (`bomba2.png`, `flaga2.png`) znajdują się w folderze `/icons/` w zasobach projektu.
3. Upewnij się, że plik konfiguracyjny `saperConfig.txt` (jeśli używany) znajduje się w folderze zasobów projektu lub w katalogu `~/.saper/`.
4. Uruchom grę jako aplikację Java.

Użycie

Inicjalizacja gry

Utwórz instancję gry:
SaperComp game = new SaperComp();
lub z callbackiem dla zamknięcia:
SaperComp game = new SaperComp(() -> System.out.println("Gra zamknięta"));

1. Gra automatycznie inicjalizuje GUI i jest gotowa do użycia po wyświetleniu okna.

Rozpoczęcie gry

- Kliknij lewym przyciskiem myszy na dowolne pole, aby odkryć je. Pierwsze kliknięcie jest zawsze bezpieczne (miny są rozmieszczane po pierwszym kliknięciu, omijając kliknięte pole i jego sąsiedztwo).
- Kliknij prawym przyciskiem myszy, aby oznaczyć pole flagą (oznacza potencjalną minę).
- Kliknij przycisk „Reset”, aby zrestartować grę z bieżącymi ustawieniami.
- Po przegranej (odkrycie miny) lub wygranej (odkrycie wszystkich bezpiecznych pól) pojawi się okno dialogowe z opcją restartu lub zamknięcia gry.

Sterowanie

- **Lewy przycisk myszy**: Odkrywa pole.
- **Prawy przycisk myszy**: Oznacza pole flagą lub usuwa flagę.
- **Przycisk „Reset”**: Restartuje grę.

Konfiguracja

Gra korzysta z klasy `SaperConfig` do zarządzania ustawieniami, które można zmieniać programowo lub poprzez plik konfiguracyjny.

Programowa zmiana ustawień

Pobierz obiekt konfiguracyjny:
SaperConfig config = game.saperConfig; // Pole jest publiczne, ale zaleca się użycie gettera, jeśli zostanie dodany

1. Dostosuj ustawienia, np.:
   config.setDifficulty(2); // Poziom trudny
   config.setRows(2, 30); // 30 wierszy dla poziomu trudnego
   config.setCols(2, 25); // 25 kolumn dla poziomu trudnego
   config.setMines(2, 150); // 150 min dla poziomu trudnego
   config.setMaxDisplayNumber(2, 8); // Maksymalna liczba min w sąsiedztwie

2. Zapisz ustawienia do pliku:
   config.saveSettings();

3. Zrestartuj grę, aby zastosować zmiany:
   game.restartCurrentGame();

Konfiguracja przez plik

Ustawienia są zapisywane w pliku `~/.saper/saperConfig.txt` za pomocą metody:
config.saveSettings();

1. Wczytaj ustawienia z pliku podczas inicjalizacji gry (automatyczne przy tworzeniu `SaperComp`):
   config.loadSettings();

Główne metody

- **Konstruktor `SaperComp()`**: Inicjalizuje grę z domyślnymi ustawieniami.
- **Konstruktor `SaperComp(Runnable onExit)`**: Inicjalizuje grę z callbackiem wywoływanym przy zamknięciu.
- **`restartCurrentGame()`**: Restartuje grę z bieżącymi ustawieniami.
- **`closeSaper()`**: Zamyka grę, wywołuje callback `onExit` (jeśli ustawiony) i zwalnia zasoby.
- **`saveScores(String user, String difficulty, String time)`**: Zapisuje wynik gry (gracz, poziom trudności, czas) do pliku `saperScores.txt`.
- **`setTileBackgroundColor(Color color)`**: Ustawia kolor tła nieodkrytych pól.
- **`setRevealedTileColor(Color color)`**: Ustawia kolor tła odkrytych pól.
- **`setNumberColor(int number, Color color)`**: Ustawia kolor liczby min w sąsiedztwie.

Dostępne ustawienia (SaperConfig)

Parametr	Typ	Domyślna wartość	Opis
`player`	String	"SaperPlayer"	Nazwa gracza
`difficulty`	int	0 (Easy)	Poziom trudności (0: Easy, 1: Medium, 2: Hard)
`easyRows`	int	10	Liczba wierszy (łatwy)
`mediumRows`	int	18	Liczba wierszy (średni)
`hardRows`	int	24	Liczba wierszy (trudny)
`easyCols`	int	8	Liczba kolumn (łatwy)
`mediumCols`	int	14	Liczba kolumn (średni)
`hardCols`	int	20	Liczba kolumn (trudny)
`easyMines`	int	10	Liczba min (łatwy)
`mediumMines`	int	40	Liczba min (średni)
`hardMines`	int	99	Liczba min (trudny)
`easyMaxNumber`	int	3	Maksymalna liczba min w sąsiedztwie (łatwy)
`mediumMaxNumber`	int	4	Maksymalna liczba min w sąsiedztwie (średni)
`hardMaxNumber`	int	6	Maksymalna liczba min w sąsiedztwie (trudny)

Mechanika gry

- **Plansza**: Siatka pól (`ROWS` x `COLS`) z losowo rozmieszczonymi minami (`MINES`). Poziomy trudności:
    - **Łatwy (0)**: 10x8 pól, 10 min, maksymalna liczba sąsiednich min: 3.
    - **Średni (1)**: 18x14 pól, 40 min, maksymalna liczba sąsiednich min: 4.
    - **Trudny (2)**: 24x20 pól, 99 min, maksymalna liczba sąsiednich min: 6.
- **Odkrywanie pól**: Lewy kliknięcie odkrywa pole. Jeśli pole zawiera minę, gra się kończy. Jeśli pole jest puste (0 sąsiednich min), odkrywane są sąsiednie pola rekurencyjnie.
- **Flagi**: Prawy kliknięcie oznacza pole flagą, wskazując potencjalną minę. Liczba pozostałych min jest aktualizowana w `minesLeftLabel`.
- **Zwycięstwo**: Gra kończy się wygraną, gdy wszystkie bezpieczne pola (bez min) zostaną odkryte.
- **Czas**: Timer mierzy czas gry w sekundach, wyświetlany w `timerLabel`.
- **Ikony**: Mina (`bomba2.png`) pojawia się po przegranej, flaga (`flaga2.png`) oznacza oznaczone pole.

Ważne uwagi

- **Pierwsze kliknięcie**: Jest zawsze bezpieczne – miny są rozmieszczane tak, aby kliknięte pole i jego sąsiedztwo były wolne od min.
- **Zapis wyników**: Wyniki są zapisywane w formacie `gracz,poziom,czas` w pliku `~/.saper/saperScores.txt`.
- **Konfiguracja**: Plik `saperConfig.txt` jest wczytywany automatycznie przy starcie gry. Jeśli nie istnieje, używane są domyślne wartości.
- **Kolory**: Kolory liczb (1-8) są predefiniowane, ale można je zmienić za pomocą `setNumberColor`. Domyślne kolory:
    - 1: Niebieski
    - 2: Zielony
    - 3: Czerwony
    - 4: Ciemnoniebieski
    - 5: Brązowy
    - 6: Turkusowy
    - 7: Czarny
    - 8: Ciemnoszary
- **Ikony**: Brak ikon (`bomba2.png`, `flaga2.png`) w folderze `/icons/` powoduje wyjątek i zatrzymanie gry.

Przykład pełnego kodu

```
import saper.SaperComp;

public class Main {
    public static void main(String[] args) {
        // Inicjalizacja gry z callbackiem
        SaperComp game = new SaperComp(() -> System.out.println("Gra zamknięta"));
        // Opcjonalna zmiana ustawień
        game.setTileBackgroundColor(new Color(200, 200, 200)); // Jasnoszare tło pól
        game.setRevealedTileColor(new Color(240, 240, 240)); // Bardzo jasne tło odkrytych
pól
        game.setNumberColor(1, Color.MAGENTA); // Magentowy kolor dla liczby 1
        game.saperConfig.setDifficulty(2); // Poziom trudny
        try {
            game.saperConfig.saveSettings();
        } catch (IOException e) {
            System.err.println("Błąd zapisu konfiguracji: " + e.getMessage());
        }
    }
}
```

Czyszczenie zasobów

Aby zamknąć grę i zwolnić zasoby, wywołaj:

```
game.closeSaper();
```

Metoda ta zatrzymuje timer, wywołuje callback `onExit` (jeśli ustawiony) i zamyka okno gry.

Obsługa błędów

- **Brak pliku konfiguracyjnego**: Gra używa domyślnych wartości, jeśli plik `saperConfig.txt` nie istnieje.
- **Błąd wczytywania/zapisywania**: Wyświetlane są komunikaty błędów w konsoli, a gra kontynuuje działanie z domyślnymi wartościami.
- **Nieprawidłowe wartości w pliku konfiguracyjnym**: Np. nieprawidłowy format liczby powoduje komunikat błędu, a wartość pozostaje domyślna.
- **Brak ikon**: Brak plików `bomba2.png` lub `flaga2.png` w folderze `/icons/` powoduje wyjątek `RuntimeException`.

Rozszerzanie komponentu

- **Dodanie nowych poziomów trudności**: Rozszerz klasę `SaperConfig`, dodając nowe zmienne (np. `expertRows`, `expertMines`) i dostosuj metody `getRows`, `getMines` itd.
- **Nowe elementy wizualne**: Dodaj własne ikony lub style pól w metodzie `initGUI`.
- **Dodatkowe statystyki**: Rozszerz metodę `saveScores` o zapis dodatkowych danych (np. liczba flag).

Ograniczenia

- Gra nie obsługuje dynamicznej zmiany poziomu trudności w trakcie rozgrywki – wymaga restartu.
- Brak wsparcia dla pauzy gry lub menu ustawień w GUI.
- Rozmiar pól jest ustalony (40x40 pikseli), co może wymagać dostosowania dla dużych plansz.