Dokumentacja użytkownika komponentu SnakeGame

Wprowadzenie

Komponent SnakeGame to klasyczna gra w węża zaimplementowana w bibliotece Swing (Java). Gracz steruje wężem, który porusza się po planszy, zbiera jabłka, aby zwiększyć wynik, i unika kolizji z przeszkodami (na poziomie trudnym) lub własnym ciałem. Gra obsługuje trzy poziomy trudności (łatwy, średni, trudny) i zapisuje wyniki oraz ustawienia w plikach konfiguracyjnych. Komponent jest konfigurowalny za pomocą klasy SnakeConfig.

Wymagania

- **Środowisko**: Java 8+ z biblioteką Swing.
- **Zależności**: Standardowa biblioteka Javy (pakiet `javax.swing`, `java.awt`).
- **Pliki konfiguracyjne**: Plik `snakeConfig.txt` (zapisywany w katalogu `~/.snake/`) dla ustawień gry oraz `snakeScores.txt` dla wyników.
- **Zasoby**: Opcjonalny plik konfiguracyjny `snakeConfig.txt` w folderze zasobów projektu (`/snakeConfig.txt`), jeśli nie istnieje w katalogu użytkownika.

Instalacja

1. Skompiluj kod źródłowy komponentu (`SnakeGame.java`, `SnakeConfig.java`) w projekcie Java.
2. Upewnij się, że plik konfiguracyjny `snakeConfig.txt` (jeśli używany) znajduje się w folderze zasobów projektu lub w katalogu `~/.snake/`.
3. Uruchom grę jako aplikację Java.

Użycie

Inicjalizacja gry

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

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

Rozpoczęcie gry

- Na ekranie startowym naciśnij klawisz **Enter**, aby rozpocząć grę.
- Po zakończeniu gry (ekran „Game Over”) naciśnij **Enter**, aby zrestartować, lub wybierz „Nie” w oknie dialogowym, aby zamknąć grę.

Sterowanie

- **Strzałka w lewo**: Poruszanie wężem w lewo.
- **Strzałka w prawo**: Poruszanie wężem w prawo.
- **Strzałka w górę**: Poruszanie wężem w górę.
- **Strzałka w dół**: Poruszanie wężem w dół.
- **Enter**: Rozpoczęcie gry (z ekranu startowego) lub restart po przegranej.

Konfiguracja

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

Programowa zmiana ustawień

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

1. Dostosuj ustawienia, np.:
   config.setSnakeColor(new Color(0, 0, 255)); // Niebieski wąż
   config.setBackgroundColor(new Color(255, 255, 255)); // Białe tło
   config.setDifficulty(2); // Poziom trudny
   config.setWidth(2, 800); // Szerokość dla poziomu trudnego
   config.setHeight(2, 600); // Wysokość dla poziomu trudnego

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

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

Konfiguracja przez plik

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

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

Główne metody

- **Konstruktor `SnakeGame()`**: Inicjalizuje grę z domyślnymi ustawieniami.
- **Konstruktor `SnakeGame(Runnable onExit)`**: Inicjalizuje grę z callbackiem wywoływanym przy zamknięciu.
- **`initializeGame()`**: Inicjalizuje stan gry (wąż, jabłko, przeszkody, wynik).
- **`restartCurrentGame()`**: Restartuje grę z bieżącymi ustawieniami.
- **`closeSnake()`**: Zamyka grę, wywołuje callback `onExit` (jeśli ustawiony) i zwalnia zasoby.
- **`saveGameLog()`**: Zapisuje wynik gry (gracz, poziom trudności, wynik) do pliku `snakeScores.txt`.

Dostępne ustawienia (SnakeConfig)

Parametr	Typ	Domyślna wartość	Opis
`player`	String	"SnakePlayer"	Nazwa gracza
`difficulty`	int	1 (Medium)	Poziom trudności (0: Easy, 1: Medium, 2: Hard)
`easyWidth`	int	600	Szerokość planszy (łatwy)
`mediumWidth`	int	600	Szerokość planszy (średni)
`hardWidth`	int	600	Szerokość planszy (trudny)
`easyHeight`	int	600	Wysokość planszy (łatwy)
`mediumHeight`	int	600	Wysokość planszy (średni)
`hardHeight`	int	600	Wysokość planszy (trudny)
`easyDotSize`	int	20	Rozmiar segmentu węża/jabłka (łatwy)
`mediumDotSize`	int	20	Rozmiar segmentu węża/jabłka (średni)
`hardDotSize`	int	20	Rozmiar segmentu węża/jabłka (trudny)
`easyDelay`	int	200	Opóźnienie ruchu w ms (łatwy)
`mediumDelay`	int	140	Opóźnienie ruchu w ms (średni)
`hardDelay`	int	100	Opóźnienie ruchu w ms (trudny)
`snakeColor`	Color	RGB(0, 255, 0) (zielony)	Kolor węża
`appleColor`	Color	RGB(255, 0, 0) (czerwony)	Kolor jabłka
`backgroundColor`	Color	RGB(0, 0, 0) (czarny)	Kolor tła
`obstacleColor`	Color	RGB(128, 128, 128) (szary)	Kolor przeszkód (tylko poziom trudny)

Mechanika gry

- **Wąż**: Porusza się w czterech kierunkach (lewo, prawo, góra, dół) za pomocą strzałek. Każdy segment węża ma rozmiar określony przez `dotSize`.
- **Jabłka**: Pojawiają się losowo na planszy. Zjedzenie jabłka zwiększa wynik o 1 i wydłuża węża.
- **Przeszkody**: Występują tylko na poziomie trudnym (`difficulty = 2`). Generowane losowo (10 przeszkód) przy rozpoczęciu gry.
- **Poziomy trudności**:
    - **Łatwy (0)**: Wąż przechodzi przez krawędzie planszy (zawijanie), brak przeszkód, wolniejszy ruch (`delay = 200 ms`).
    - **Średni (1)**: Kolizja z krawędziami planszy kończy grę, brak przeszkód, średni ruch (`delay = 140 ms`).
    - **Trudny (2)**: Kolizja z krawędziami i przeszkodami kończy grę, szybszy ruch (`delay = 100 ms`).
- **Wynik**: Zwiększa się o 1 za każde zjedzone jabłko. Zapisywany do pliku `snakeScores.txt` po zakończeniu gry.

Ważne uwagi

- **Fokus klawiatury**: Gra wymaga, aby panel gry (`gamePanel`) miał fokus, aby odbierać zdarzenia klawiatury. Kliknięcie myszą na panel przywraca fokus.
- **Zapis wyników**: Wyniki są zapisywane w formacie `gracz,poziom,wynik` w pliku `~/.snake/snakeScores.txt`.
- **Konfiguracja**: Plik `snakeConfig.txt` jest wczytywany automatycznie przy starcie gry. Jeśli nie istnieje, używane są domyślne wartości.
- **Kolory**: Kolory w pliku konfiguracyjnym podaje się w formacie RGB (np. `255,0,0` dla czerwonego). Nieprawidłowe wartości powodują komunikat błędu w konsoli.

Przykład pełnego kodu

```
import Sneake_Game.SnakeGame;
public class Main {
    public static void main(String[] args) {
        // Inicjalizacja gry z callbackiem
        SnakeGame game = new SnakeGame(() -> System.out.println("Gra zamknięta"));
        // Opcjonalna zmiana ustawień
        game.snakeConfig.setSnakeColor(new Color(0, 0, 255)); // Niebieski wąż
        game.snakeConfig.setDifficulty(2); // Poziom trudny
        try {
            game.snakeConfig.saveSettings();
        } catch (IOException e) {
            System.err.println("Błąd zapisu konfiguracji: " + e.getMessage());
        }
    }
}
```

Czyszczenie zasobów

Aby zamknąć grę i zwolnić zasoby, wywołaj:

game.closeSnake();

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

Obsługa błędów

- **Brak pliku konfiguracyjnego**: Gra używa domyślnych wartości, jeśli plik `snakeConfig.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 koloru lub liczby powoduje komunikat błędu, a wartość pozostaje domyślna.

Rozszerzanie komponentu

- **Dodanie nowych poziomów trudności**: Rozszerz klasę `SnakeConfig`, dodając nowe zmienne (np. `expertWidth`, `expertDelay`) i dostosuj metody `getWidth`, `getDelay` itd.
- **Nowe elementy gry**: Dodaj nowe obiekty (np. bonusowe jabłka) w metodzie `drawGame` i obsłudze kolizji.
- **Interakcje użytkownika**: Rozszerz metodę `keyPressed` o obsługę dodatkowych klawiszy (np. pauza gry).

Ograniczenia

- Gra nie obsługuje dynamicznej zmiany poziomu trudności w trakcie rozgrywki – wymaga restartu.
- Przeszkody są generowane tylko na poziomie trudnym i nie można ich dostosować bez modyfikacji kodu.
- Brak wsparcia dla pauzy gry lub menu ustawień w GUI.