Dokumentáció

„Snake” játék mesterséges intelligencia kiegészítéssel Java nyelven

Szeberényi Gergő, Szikszai Márk, Gönczöl Mátyás

A projekt háttere és célja:

Ebben a projektben a "Snake" nevű klasszikus videojáték egy modernebb változatát Java nyelven valósítottuk meg. A “Snake” egy széles körben ismert és kedvelt játék, 1976-ban jelent meg az első változata a Blockade játéktermi gépen. A játék célja, hogy a játékosok minél több almát egyenek meg, ezzel növelve a kígyó hosszát és összegyűjtve a lehető legtöbb pontot.

Az általunk készített alkalmazás biztosít egy felhasználói felületet, ahol a játékosok irányíthatják a kígyót, almát gyűjthetnek, miközben kerülik a kígyó saját testével való ütközést és a játékterület széleit. A projekt magában foglal egy mesterséges intelligencia (MI) módú változatot is, ami érdekességet nyújt a játékosok számára, mivel itt a kígyót a számítógép irányítja. A projekt egy tanulási platformként szolgál a fejlesztői csapat számára az eseményvezérelt programozási paradigma megértésében. Célunk egy szórakoztató, ugyanakkor kihívást jelentő játékélmény biztosítása a felhasználóknak.

A projekt mappaszerkezete

Gyökér Könyvtár:
A projekt gyökérkönyvtára tartalmazza az összes szükséges konfigurációs fájlt, beleértve a **pom.xml**-t a **Maven** projektkezelőhöz és a **rekord.txt**-t, ami a játék rekordjainak tárolására szolgál.

src/main/java: Ez a mappa tartalmazza a Java forráskódokat, beleértve az alkalmazás fő osztályait, mint a **Frame.java**, **SnakeGame.java**, **SnakeGameMI.java**, és **HamiltonianCycleOnGrid**.java.

Target Mappa: A target könyvtár a projekt fordított és csomagolt fájljait tartalmazza, beleértve a futtatható **JAR** fájlokat és az **osztályfájlok**at, amelyek a Java kódból generálódnak.

Osztályok:

1. Frame.java

A **Frame osztály** a "Snake" játék főmenüjének megjelenítéséért felelős. Ez az osztály a JFrame-ből származik, és implementálja az ActionListener interfészt a felhasználói interakció kezeléséhez.

Felületi Elemek:
JPanel panel: Egy panel, amely GridBagLayout-ot használ a komponensek elrendezésére.

JButton gombJatek, **JButton gombGepJatek**: Két gomb a játék indításához és az MI mód választásához.
JLabel cim: Egy címke, ami a "Snake" szöveget jeleníti meg.

Fő Konstruktor - Frame():

- Létrehoz egy új **JFrame** példányt FrameA néven, amely később a játék fő ablakává válik.
- Beállítja a cim JLabel tulajdonságait, mint például a szöveg színe és betűtípusa.
- Mindkét gomb méretét, hátterét, szegélyét és betűtípusát beállítja
- A **gridElrendezes** változó segítségével elrendezési szabályokat állít be a **GridBagLayout** számára.
- Hozzáadja a címkét és a gombokat a panelhez az elrendezési szabályok alapján.
- Az **ActionListener**ek hozzáadása a gombokhoz, amelyek **reagálnak a felhasználói interakciókra:**
- A **gombJatek** ActionListenerje új **SnakeGame** példányt hoz létre, amikor a gombot megnyomják.
- A **gombGepJatek** ActionListenerje új **SnakeGameMI** példányt hoz létre, ami az MI módú játékot indítja el.

A **Frame osztály** a felhasználói felület elsődleges kapcsolódási pontja, amely lehetővé teszi a játékosok számára, hogy kiválasszák a kívánt játékmódot, és elindítsák a játékot.

2. HamiltonianCycleOnGrid.java

A **HamiltonianCycleOnGrid osztály** a "Snake" játék mesterséges intelligenciájának (MI) magját képezi, amely egy komplex algoritmust használ a kígyó mozgásának optimalizálására a játékterületen.

Alapvető Változók:
int ARENA_HEIGHT, **int ARENA_WIDTH**: Ezek a változók határozzák meg a játéktér méretét.
int[] tourToNumber: Egy tömb, ami nyilvántartja a Hamiltoni útvonal minden pontjának indexét.
Food food: Az étel objektum, amely az étel helyzetét és értékét tárolja.

Osztályok és Enumerációk:
enum SnakeDirection: Enumeráció a kígyó lehetséges irányainak meghatározására.
class Snake: Ez az osztály a kígyó jelenlegi állapotát reprezentálja, beleértve annak fejét, farkát, testét és növekedési hosszát.
class Maze: A labirintus generálásáért felelős osztály, amely a Hamiltoni ciklus alapján működik.

Fontos Metódusok:
getPathNumber(int x, int y): Egy adott koordináta Hamiltoni útvonalon belüli indexének lekérdezése.
move_snake_head(SnakeDirection dir): A kígyó fejének mozgatása az adott irányban.
aiGetNewSnakeDirection(): Kiszámítja a kígyó következő lépését az MI algoritmus alapján.
generate(): A Hamiltoni ciklus alapján generálja a labirintust.

A **HamiltonianCycleOnGrid osztály** a kígyó mozgását, az étel helyzetét és a játéktér állapotát kezeli. A Hamiltoni ciklus algoritmus biztosítja, hogy a kígyó mindig egy előre meghatározott útvonalon mozogjon, minimalizálva az összeütközés esélyét.

Ez az osztály kulcsfontosságú a "Snake" játék MI módjának működéséhez, mivel lehetővé teszi a kígyó automatizált navigálását a játéktéren belül.

3. SnakeGame.java

A **SnakeGame osztály** a "Snake" játék alapvető logikáját és a felhasználói felületét valósítja meg egy JPanel-en keresztül.

Alapvető Változók és Konstansok:
int kepernyoSzelesseg, **int kepernyoHossz**, **int meret**: A játéktér méretei és a kígyó mérete.
int jatekMeret: A játéktérben elhelyezhető maximális kígyóelemek száma.
int kesleltetes: A játék frissítési sebessége.
int[] x, **int[] y**: Tömbök a kígyó testének x és y koordinátáinak tárolására.
int kigyoHossz, int **megevettAlmak**: A kígyó jelenlegi hossza és az elfogyasztott almák száma.
int almaX, **int almaY**: Az alma pozíciójának koordinátái.

Konstruktor - SnakeGame():

- Inicializálja a játéktér méretét és hátterét.
- Beállítja a panel fókuszálhatóságát a billentyűzetesemények fogadásához.
- Elindítja a játékot a startGame() metódussal.

Fontos Metódusok:
startGame(): Inicializálja az alapvető játékelemeket és elindítja a Timer-t.
paintComponent(Graphics g): Felelős a játéktér és a kígyó megjelenítéséért.
Rajzolas(Graphics g): A kígyó és alma kirajzolása a játéktéren.
UjAlma(): Új alma helyzetének generálása a játéktéren.
mozgas(): A kígyó mozgásának frissítése.
AlmaTalalat(): Ellenőrzi, hogy a kígyó elfogyasztott-e egy almát.
Utkozes(): Ellenőrzi, hogy a kígyó ütközött-e a falakkal vagy önmagával.
gameOver(Graphics g): Game over képernyő megjelenítése.

Belül Definiált Osztály - MyKeyAdapter:
Figyeli a billentyűzet eseményeket és frissíti a kígyó mozgásának irányát.
Ez az osztály biztosítja a "Snake" játék alapvető funkcionalitását, beleértve a kígyó mozgatását, az alma generálását, és a játék állapotának kezelését.

4. SnakeGameMI.java

A **SnakeGameMI osztály** a "Snake" játék mesterséges intelligenciával (MI) működő változatát valósítja meg, amely számos kulcsfontosságú különbséget mutat a SnakeGame osztályhoz képest.

Alapvető Változások és Kiegészítések:
int kesleltetes: Ez a változó az MI játékmód sebességét szabályozza, ami jelentősen gyorsabb, mint a standard játékmódé.
HamiltonianCycleOnGrid cycle: Az MI algoritmus példánya, amely a kígyó mozgását irányítja.
StartPosition(): A kígyó kezdőpozíciójának beállítása, ami eltér a hagyományos módtól.

Fontos Metódusok:
startGame(): (módosítva) Az MI alapú játék indítása, beleértve az algoritmus inicializálását.
Rajzolas(Graphics g): (módosítva) A kígyó és alma kirajzolása, ahol a kígyó irányítása az MI algoritmus alapján történik.

mozgas(): (módosítva) Automatizált mozgás, ahol az irányt az MI algoritmus határozza meg.
AlmaTalalat(), Utkozes(), gameOver(Graphics g): Ezek a metódusok hasonlóan működnek, mint a SnakeGame osztályban, de az MI kontextusában.

MyKeyAdapter Osztály:
Egy belső osztály, amely lehetővé teszi a felhasználó számára, hogy a sebességet állítsa az MI játékmódban, ezáltal növelve vagy csökkentve a játék nehézségi szintjét.

A **SnakeGameMI osztály** a "Snake" játék egy új dimenzióját nyitja meg, ahol a játékosok megfigyelhetik, hogyan navigál az MI vezérelte kígyó a játéktéren. Az osztály a HamiltonianCycleOnGrid osztályt használja az optimális útvonal meghatározására, biztosítva a kígyó folyamatos és ütközésmentes mozgását.

Eseményvezérelt Programozási Paradigma a projektben:

A projekt kiemelkedően alkalmazza az eseményvezérelt programozási paradigmát a felhasználói interakciók kezelésében. A játékban a kígyó irányítása, az almák elfogyasztása, és **a játék állapotának** (például a játék kezdete, vége, és MI mód aktiválása) **változása eseményeken alapul**. Ezek az események **billentyűzet- és időzítő események**, amelyeket a **Java Swing** könyvtár **ActionListener** és **KeyAdapter osztályainak használatával kezelünk**. Ez a megközelítés lehetővé teszi a játék **dinamikus** és **reaktív** működését, ahol **a felhasználói bemenetek valós időben változtatják meg a játék menetét,** így biztosítva egy interaktív és élvezetes felhasználói élményt.