Game of Life Programozási Feladat Specifikáció

Gáspár Róbert
Neptun kód: K8FD5S

2024. november 19.

Feladat rövid ismertetése

A házi feladat keretében egy klasszikus *Game of Life* szimulációt kell megvalósítani. A *Game of Life* egy sejtautomata, amit John Conway matematikus alkotott meg 1970-ben. A játék egy kétdimenziós rácson játszódik, ahol minden cella két állapotban lehet: él vagy halott. Az idő múlásával a sejtek állapota az alábbi szabályok szerint változik:

- **Túlélés**: Egy élő sejt továbbra is életben marad, ha 2 vagy 3 szomszédos élő sejt veszi körül.
- **Szaporodás**: Egy halott sejt életre kel, ha pontosan 3 szomszédos élő sejt veszi körül.
- **Halál**: Minden más esetben a sejtek meghalnak, azaz halottak maradnak.

A feladat során egy interaktív *Swing* grafikus felhasználói felületet (GUI) kell készíteni, amely lehetőséget biztosít a felhasználónak a rács méretének kiválasztására, a szimuláció elindítására, megállítására, valamint a mentések kezelésére (mentés és betöltés). A program lehetőséget nyújt arra is, hogy a felhasználó fájlba mentse a játék állapotát JSON formátumban, majd később újra betöltse azt. A játék nézete dinamikusan nagyítható és kicsinyíthető lesz, illetve a grid elmozdítható lesz a képernyőn. A felhasználó menükön és billentyűzet használatán keresztül érheti el az összes funkciót, mint például új játék indítása, játék mentése, betöltése vagy a program kilépése.

Use-case-ek

- **Új játék indítása**: A felhasználó a menüben vagy egy kezdő képernyőn elindíthat egy új játékot, ahol megadhatja a grid méretét (pl. 10x10, 50x50 stb.). A grid inicializálása után a felhasználó kattintással élővé tehet vagy kiüríthet sejteket a kezdő állapot beállításához, majd elindíthatja a szimulációt.

- **Játék indítása/leállítása**: A felhasználó egy gomb vagy menüpont segítségével elindíthatja és megállíthatja a szimulációt. A szimuláció során a sejtek állapota folyamatosan változik az előbb említett szabályok alapján, és a képernyőn vizuálisan frissül.

- **Játék mentése**: A felhasználó a menüben vagy egy megfelelő gombbal elmentheti az aktuális játékállást egy JSON fájlba. A mentett fájl tartalmazza a rács méretét, a sejtek állapotát.

- **Játék betöltése**: A felhasználó a menüben kiválaszthat egy korábban elmentett JSON fájlt, amely betölti a játék előző állapotát, beleértve a rács méretét és a sejtek állapotát.

- **Nézet nagyítása/kicsinyítése és mozgatása**: A felhasználó dinamikusan nagyíthatja és kicsinyítheti a grid nézetét, illetve elmozdíthatja azt, hogy egy adott részletet megfigyelhessen. Ezt az egér görgőjével vagy gombok segítségével érheti el.

- **Kilépés**: A felhasználó a menü segítségével kiléphet a programból. Ha a játék nincs elmentve, akkor egy figyelmeztetés jelenik meg, hogy megerősítést kérjen a kilépésről.

Megoldási ötlet vázlatos ismertetése

A megoldás alapvetően a Java *Swing* keretrendszerre épül, mivel a specifikáció megköveteli a Swing GUI használatát. A program főbb részei a következők:

- **Grafikus felhasználói felület (GUI)**: A GUI menükből és panelekből épül fel. A menük biztosítják a felhasználó számára az alapvető funkciókat, mint a játék indítása, mentése, betöltése és kilépése. A grid nézete egy *JPanel* lesz, amelyben az egyes sejtek állapotai jelennek meg. A grid vizuálisan is dinamikusan változik a játék során, a *Graphics* osztály segítségével rajzolva.
- **Rács nézet és transzformációk**: A rács nézete nagyítható és kicsinyíthető, valamint elmozdítható lesz a képernyőn. Ehhez a *Graphics2D* osztály és az affine transzformációk lesznek alkalmazva, hogy a grid mérete és pozíciója dinamikusan módosítható legyen.
- **Gyűjtemény keretrendszer**: A grid belső adatstruktúrájának kezeléséhez egy kétdimenziós tömböt használunk, amelyben a sejtek állapotát tároljuk.
- **JSON fájlkezelés**: A játékállások mentéséhez és betöltéséhez a *Jackson* vagy *Gson* könyvtárakat használjuk, amelyek lehetővé teszik az objektumok JSON formátumban történő szerializálását és deszerializálását. A grid állapota és a játék további adatai egy fájlban lesznek eltárolva, amit a felhasználó betölthet később.
- **Tesztelés**: A program működésének ellenőrzésére *JUnit* tesztelési keretrendszer kerül alkalmazásra. Legalább három osztály és tíz metódus kerül tesztelésre.