Specifikáció

Életjáték

Gutási Ádám

# Specifikáció

A játék az **életjáték** sejtautomata egy implementációja Javában.

## Életjáték

Adott egy négyzetrács, melyen egy négyzet egy cellát reprezentál. Minden cella két állapotban lehet (élő/üres). A következő iterációban azok a cellák maradnak életben, amelyeknek a beállított játékszabályban szereplő számú élő szomszédja van. Azok az üres cellák pedig a következő iterációban életre kelnek, amelyeknek a játékszabályban beállított számú élő szomszédja van. A szomszédok vizsgálata Moore-környezetben történik (azaz egy cellának maximum 8 szomszédja lehet).

## Funkciók

A játék elején az üres játékmezőt látjuk. Lehetőségünk van előző játékmenetet betölteni, vagy teljesen újat indítani. Új játékmenet esetén, mielőtt elkezdődhetne a játék, először be kell állítani a sejtautomata szabályát. ennek a szokásos jelölése *B../S..* (born / survive). pl. a Conway-féle klasszikus Game of Life kódja B3/S23.

Ezekután kijelölhetjük azokat a sejteket, melyeket a kezdőállapotban aktívként szeretnénk elindítani. Ha mindezeket beállítottuk, kezdődhet a szimuláció.

A GUI a következő elemekből áll:

* Játékmező, ami ebben az implementációban egy 50x50-es mezőből áll.
  + A mező alatt továbbá találhatóak **Step, Auto**, illetve **1x, 2x, 3x,** és **4x** feliratú gombok (utóbbiak az automatikus szimuláció sebességének állítására szolgálnak).
* A jelenlegi iteráció száma.
* A szimuláció sebességét (amennyiben automatikus léptetésben van a szimuláció)
* A felhasznált szabály szövegformában, illetve ennek az átállítására szolgáló szövegdoboz, mellette egy OK gomb, ennek bevitelére.
* Régebbi játékállás betöltésére, illetve a jelenlegi állás mentésére szolgáló gombok.

Egy érvényes szabály-, majd kezdőértékek megadása után az **Auto** gombot megnyomva a játék elkezd magától iterálni. Ezután a szimuláció sebessége állítható a **1x, 2x, 3x,** és **4x** gombokkal. Manuálismódba a **Step** gombbal léphetünk.

A játékból a GUI bezárásával, vagy az Exit game gomb megnyomásával léphetünk ki. Ezek azonban nem mentik a játékállást: menteni a felhasználó tud a **Save** **state** gomb megnyomásával.

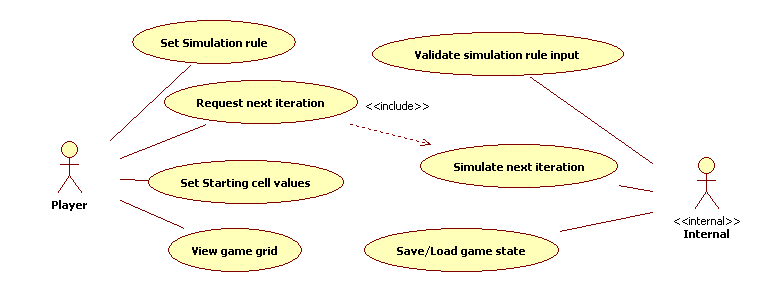
## Megvalósítás

A program Swing-et használ a grafikus tartalmak megjelenítésére. Egy ablakot használ az implementációm, így egy **JFrame**-ben lesz minden elem megtalálható. Három **JPanel** objektum tárolja az elemeket:

* az első a szabály beállításához szükséges elemeket;
* A második a játéktáblát, illetve
* a harmadik az automaikus/kézi léptetést szolgáló gombokat, a jelenlegi populáció és az iteráció számát tartalmazza.

A játéktáblát 50x50 darab **JButton** elemmel oldottam meg, melyek a szimuláció indítása előtt kattinthatóak, ezzel módosíthatóak az általuk reprezentált cellák állapotai. A további gombok szintén **JButton** elemekből állnak, illetve szöveg megjelenítésére **JLabel**, beviteli mezőként pedig **JTextField** elemet használok.

A játék a mentett állást (beleértve a játéktáblát és a hozzátartozó szabályt) egy *gameOfLifeGameState.txt* nevű fájlba menti.



- A játék **use case** diagrammja

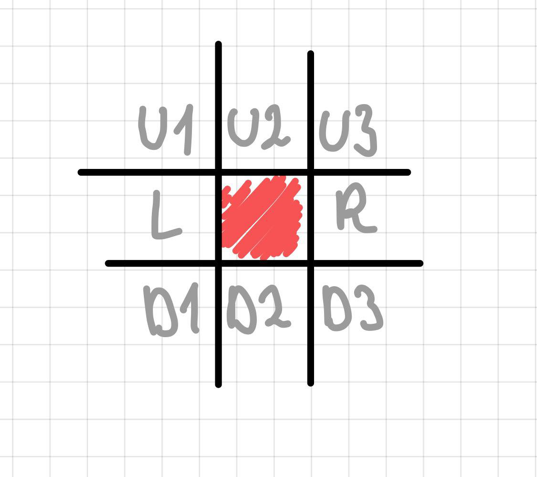
## use-case -ek

|  |  |
| --- | --- |
| Set simulation rule | A játék indítása, majd a cellaértékek beállítása után a játékos megadja azt a szabályt, ami alapján a játék iterálni fog. |
| Request next iteration | Következő iteráció kérése. Ezt a felhasználó a UI **Step** gombjának megnyomásával érheti el. |
| Set Starting cell values | A kezdő cellaértékek beállítására rögtön a játék indításakor van lehetőség. |
| View game grid | A játékmező a program futása során folyamatosan látszik, mindig van lehetőség annak megtekintésére. |
| Validate simulation rule input | Miután a játékos egy szabályt beírt, és az **OK**  gombra nyomott, a belső rendszer validálja azt. Amennyiben hibás bemenetet észlelt, nem engedi a felhasználó számára a rendszer iterálását. |
| Simulate next iteration | A felhasználó kérésére (**Step** gomb), vagy az automatikus iterálás során (**Auto** gomb) végrehajtott műveletsorozat, melyet az Internal irányít. |
| Save/Load game state | A felhasználó által a megfelelő gombok lenyomásával indított folyamat, mely menti/ betölti a játékot. |

## megoldási ötletek, technológiai megoldások

### cellaszomszédok

A cellák szomszédiat a **Grid** osztály segítségével kaphatjuk meg. Ezen szomszédokat egy HashMapben kaphatjuk meg, ahol az értékek maguk a cellák, a kulcs pedig egy String, a következő módón:



A képen a piros cella szomszédjait látjuk, rajtuk a HashMap-ben hozzájuk tartozó kulcs értékét.

### játékmező

A játékmező egy ArrayList<ArrayList<Cell>> kollekcióként van tárolva, ami így egy kétdimenziós listaként is értelmezhető. az egyes sorok a külső, az azon belüli elemek pedig a belső ArrayList -ben vannak tárolva.

### szabály validálása, beállítása

Ahhoz, hogy a játék az iteráció során lépni tudjon, be kell, hogy legyen állítva egy szabály számára, ami alapján születnek és túlélnek a cellák. A szabály beállítása több lépésben történik:

1. A felhasználó beírja a szabályt, majd megynomja az OK gombot.
2. A játék validálja a kapott szabályt a B{1}[0-9]+\\/S{1}[0-9]+ reguláris kifejezés segítségével. (Olyan String-et várunk, ami 1db B betűvel, majd valamennyi számmal 0 és 9 közt folytatódik. Ezután egy / jellel, 1 db S betűvel, végül ismét valamennyi 0 és 9 közti számmal zárul)
3. Amennyiben a szabály érvényes, a bemeneti String átalakul: a betűket elhagyja a program, majd a perjel mentén két int[] tömbbe rendezi őket. Ez a két tömb tárolja el, hogy az adott szabály a cella hány élő szomszédja esetén érvényesül.

### fájlformátum

Több szerializálható osztály is létezik a programon belül. Ezek a mentés gomb megnyomásával kimentésre kerülnek egy egyszerű szövegfájlba, a következő sorrendben:

1. játékmező
2. születési szabály
3. túlélési szabály
4. iteráció száma
5. populáció száma

A fájl beolvasása is ugyanebben a sorrendben történik.