Explorator – Suunnittelu/Toteutusdokumentti

1 Kuvaus

Explorator sovelluksen tarkoituksena on ollut luoda helposti ylläpidettävä polunetsintäalgoritmi sovelluskirjasto. Ohjelmaan kuuluu myös Processing kielellä toteutettu graafinen käyttöliittymä, jolla voi testata polunetsintäalgoritmeja visuaalisesti. Tämän työn puitteissa oli tarkoitus toteuttaa ja visualisoida seuraavat algoritmit: Floyd-Warshall, Djikstra sekä A*. Myöskään käyttöliittymä koodi ei ole tarkoituksella kovinkaan modulaarista, koska ollaan haluttu saada nopeasti valmiiksi toimiva graafinen käyttöliittymä. Näin ollen käyttöliittymässä esiintyvä koodi ei välttämättä ole kaikkein kauneinta mitä maa päällään kantaa.

2 Ohjelman yleisrakenne

Ohjelma koostuu kahdesta suuremmasta moduulista: Käyttöliittymä moduulista, joka on toteutettu Processing ohjelmointi kielellä sekä Explorator Java projektista, johon on käytetty Jdeveloper 11.1.2.4 kehitystyökalua. Processing kieli pystyy lukemaan ja kääntämää Java koodia, joten Java projektista on käännetty explorator.jar moduuli, johon viitataan käyttöliittymässä. Alla on luetelty ohjelmaan tällä hetkellä kuuluvat Processing käyttöliittymä komponentit tiedoston nimittäin git repositoryn juuresta alkaen. Java luokkia ei kuvata, koska niistä on generoitu oma erillinen JavaDoc dokumentaatio. Ainoastaan Java koodin juuripolku on esitelty alla.

Processing:

```
/Code/Explorator/Explorator.pde (GUI koodi)
/Code/Exploraror/Tuple2.pde (GUI koodin käyttämä apu luokka)
```

Java:

```
/Code/src/main/java   (Explorator Java koodit)
/Code/src/main/test   (Explorator Testi luokat)
```

Ajettava ohjelma

```
/Releases/application.linux32 (ei testattu, vaatii JDK 1.7)
```

/Releases/application.linux64 (ei testattu, vaatii JDK 1.7)
/Releases/application.windows32 (toimii)
/Releases/application.windows64 (toimii)

3 Suorituskykytestaus tuloksia

Suorituskykytestaus on suoritettu oheisessa java luokassa
/Code/src/main/test/fi/explorator/PerformanceTests.java

Testissä on luotu verkko, jossa on 2304 solmua. Tällä on yritetty simuloida pelikarttaa, joka olisi resoluutiossa 1280x720 sekä jokaisen solmun koko olisi 20x20 pixeliä. Verkosta n.33% on solmuja, jotka on blockkaavia eli joita ei käydä läpi. Myös verkko on staattinen eli solmujen blokkaavuus ei muutu. Tämä asia jokseenkin suosii Floyd-Warshall algoritmia, jossa suurin osa ajasta menee etäisyysmatriisin laskemiseen ensimmäisellä kerralla. Tämän jälkeen voidaan hakea todella nopeasti matriisista polku solmusta a solmuun b. Alla tulokset teksti muodossa. Polkujen lukumäärä kasvaa 100:Sta 1000:een.

```
Cells in grid: 2304
100 paths resolved using Floyd-Warshall in 34.680996 secs.
100 paths resolved using Dijkstra       in 0.8020005 secs.
100 paths resolved using A*             in 0.09900003 secs.
200 paths resolved using Floyd-Warshall in 0.012000001 secs.
200 paths resolved using Dijkstra       in 1.5400009 secs.
200 paths resolved using A*             in 0.10300003 secs.
300 paths resolved using Floyd-Warshall in 0.018000001 secs.
300 paths resolved using Dijkstra       in 2.2959979 secs.
300 paths resolved using A*             in 0.12900005 secs.
400 paths resolved using Floyd-Warshall in 0.023000002 secs.
400 paths resolved using Dijkstra       in 3.0819905 secs.
400 paths resolved using A*             in 0.18900016 secs.
500 paths resolved using Floyd-Warshall in 0.028000003 secs.
500 paths resolved using Dijkstra       in 3.8199837 secs.
500 paths resolved using A*             in 0.21500024 secs.
600 paths resolved using Floyd-Warshall in 0.035999995 secs.
600 paths resolved using Dijkstra       in 4.532978 secs.
600 paths resolved using A*             in 0.26500013 secs.
700 paths resolved using Floyd-Warshall in 0.042999983 secs.
700 paths resolved using Dijkstra       in 5.5789704 secs.
700 paths resolved using A*             in 0.31299967 secs.
800 paths resolved using Floyd-Warshall in 0.046999976 secs.
800 paths resolved using Dijkstra       in 6.0219646 secs.
800 paths resolved using A*             in 0.35999924 secs.
900 paths resolved using Floyd-Warshall in 0.052999966 secs.
900 paths resolved using Dijkstra       in 6.8549566 secs.
900 paths resolved using A*             in 0.37899905 secs.
1000 paths resolved using Floyd-Warshal in 0.058999956 secs.
1000 paths resolved using Dijkstra      in 8.064934 secs.
1000 paths resolved using A*            in 0.4109986 secs.
```

4 Bugi ja puutelista

- ~~**BUG:** Generoitu ajettava Processing sovellus ei jostain syystä toimi tällä hetkellä.~~
  ~~**Workaround:** Kopioi explorator.jar tiedosto Processing sketchbook kotihakemiston alle seuraavaan hakemistoon: libraries/explorator/library/ Processing sketchbook hakemiston sijainnin näet käynistämällä Processing IDEn ja menemällä File >Preferences. Ylimmäisenä lukee Sketchbook location.~~
  Ratkaisu: explorator.jar piti nimetä erinimellä, koska projekti oli myös Explorator, jolloin processing generoi Explorator.java luokan ja tämä aiheutti jotain name clashausta. Tällä hetkellä ajettavat windows releaset toimivat.

5 Lähteet

http://theory.stanford.edu/~amitp/GameProgramming/AStarComparison.html

http://en.wikipedia.org/wiki/A*_search_algorithm

http://en.wikipedia.org/wiki/Floyd%E2%80%93Warshall_algorithm

http://en.wikipedia.org/wiki/Dijkstra%27s_algorithm

http://en.wikipedia.org/wiki/Priority_queue

https://www.cs.auckland.ac.nz/software/AlgAnim/dijkstra.html

http://www.geeksforgeeks.org/greedy-algorithms-set-6-dijkstras-shortest-path-algorithm/