Viikko 1.

16.1. Perjantai

En oikein tiennyt perjantaina, että mistä aloittaisin projektini tekemisen. Päätin kuitenkin lukea aluksi kurssin sivulla olevan artikkelin http://theory.stanford.edu/~amitp/GameProgramming/AstarComparison.html. Luin myös Tietorakenteet ja algoritmit -kurssin materiaalia aiheesta A*-algoritmi ja näillä materiaaleilla pääsin hyvin alkuun. Aloitin projektini ohjelmoinnin toteuttamalla ensin Dijkstran algoritmin, jonka avulla sain tietää jonkin lyhyimmän polun A → B. Tämän päälle rakensin analysointia, jonka avulla esimerkiksi kartasta

. . . . B
. . . . .
A . . . .

voitiin muodostaa verkko ArrayListan avulla. Tein myös luokan Solmu ja PriorityQueue<Solmu> pitämään verkkoani kasassa ja auttamaan löytämään tietyn solmun naapurit tehokkaasti. Dijkstra onnistui hyvin ja löysi polun. Lauantaina aion laajentaa Dijkstran A*-algoritmiksi.

17.1. Lauantai

Koska sain Dijkstran toimimaan, oli nyt aika laajentaa siitä A*. Käytännössä tämä tarkoitti, että Dijkstran algoritmiin lisättiin heurestiikka. Tulin siihen tulokseen, että käytän projektissani Manhattan etäisyyttä heurestiikan muodostamiseen.

Kohtasin kuitenkin ongelmia toteutuksessa, koska verkkoni oli yksiulotteinen taulukko, joka kuitenkin käyttäytyi kuin yksiulotteinen taulukko, joten etäisyyksien laskeminen ei ollut koordinaatistossa niin yksinkertaista. Tein kuitenkin pari apumetodia ja ongelma oli ratkaistu, mutta kohtasin uuden ongelman; A*-algoritmini ei toiminut kuten piti ja koodi alkoi olla rumaa ja vaikeasti hallittavissa, joten aion ottaa sen ensimmäiseksi prioriteetikseni.

Refaktoroin koodin kunnolla ja tein monta uutta luokkaa. Se selvensi paljon koodini rakennetta ja toi paljon uusia ideoita siitä, että miten aion projektini lopullisen vaiheen toteuttaa. Seuraavaksi aion saada A* toimimaan täysin.

18.1. Sunnuntai - 20.1. Tiistai

Koska A* algoritmini ei toiminut, niin päätin aloittaa testien kirjoittamisen. Kirjoitin paljon testejä siitä, että algoritmini Dijkstra-versio edes toimii. Testit olivat pääosin koordinaattien muutosten testailuja, karttojen polkujen löytämistä suljetuissa tiloissa, kartan analysointia, sekä polkujen oikeellisuutta.

Testeista tuli mielestäni melko kattavia ja voin luottaa, että A* läpäisee samat testit ja kun saan sen toimimaan, niin pystyn kirjoittamaan vielä kattavampia testejä, kuten lyhintä polkua jne.

Päätin jättää A* toteutuksen keskiviikolle.

21.1. Keskiviikko

Halusin saada A* toimimaan joten päätin lukea aiheesta vielä hieman enemmän. Tajusin, että minulla olivat solmut olleet väärässä prioriteetissa ja siksi A*-algoritmini toimi koko ajan kuten

Dijkstran algoritmi. Sain ongelman korjattua ja testit menivät läpi, mutta huomasin seuraavan ongelman: A*-algoritmini käy liian paljon solmuja läpi, vaikka löytääkin aina lyhyimmän polun. Päätin hakea tietoa aiheesta ja törmäsinkin ongelmaani; A*-algoritmini heurestiikka oli turhan löysä ja tarvitsin tiukemman prioriteetin solmuilleni verkossa. Tämä johti siihen, että vaikka Manhattan etäisyys oli oikein ja etäisyys lähtösolmusta, niin kekoon meni useita solmuja samoilla prioriteeteilla. Ratkaisu tähän oli muuttaa heurestiikkaa tiukemmaksi ja kun sen sain tehtyä, toimi A* erittäin hyvin. Seuraavassa kuvassa (saattaa vaatia zoomausta) esimerkki algoritmini toiminnasta. Se toimii siis erittäin tehokkaasti jo nyt verrattuna Dijkstraan, koska solmuja ei tarvitse analysoida niin paljon.

```
* = käyty ja analysoitu
. = ei käyty
© = optimaalinen polku (lyhin)
A -> B, jossa A sijaitsee vasemmassa alakulmassa ja B oikeassa yläkulmassa

Dijkstra:

[*, *, *, *, *, *, *, *, *, *, *, *, *, ©]
[*, *, *, *, *, *, *, *, *, *, *, *, ©, ©]
[*, *, *, *, *, *, *, *, *, *, *, ©, ©, *]
[*, *, *, *, *, *, *, *, *, ©, ©, ©, *, *]
[*, *, *, *, *, *, *, *, *, ©, ©, *, *, *]
[*, *, *, *, *, *, *, *, ©, ©, *, *, *, *]
[*, *, *, *, *, *, *, ©, ©, *, *, *, *, *]
[*, *, *, *, *, *, ©, ©, *, *, *, *, *, *]
[*, *, *, *, *, ©, ©, *, *, *, *, *, *, *]
[*, *, *, *, *, ©, *, *, *, *, *, *, *, *]
[*, *, *, ©, ©, ©, *, *, *, *, *, *, *, *]
[*, *, *, ©, *, *, *, *, *, *, *, *, *, *]
[*, ©, ©, ©, *, *, *, *, *, *, *, *, *, *]
[*, ©, *, *, *, *, *, *, *, *, *, *, *, *]
[©, ©, *, *, *, *, *, *, *, *, *, *, *, *]

A*

[., ., ., ., ., ., ., ., ., ., ., ., ., *, ©]
[., ., ., ., ., ., ., ., ., ., ., ., *, ©, ©]
[., ., ., ., ., ., ., ., ., ., ., *, ©, ©, *]
[., ., ., ., ., ., ., ., ., ., *, ©, ©, *, .]
[., ., ., ., ., ., ., ., ., *, ©, ©, *, ., .]
[., ., ., ., ., ., ., ., *, ©, ©, *, ., ., .]
[., ., ., ., ., ., ., *, ©, ©, *, ., ., ., .]
[., ., ., ., ., ., *, ©, ©, *, ., ., ., ., .]
[., ., ., ., ., *, ©, ©, *, ., ., ., ., ., .]
[., ., ., ., *, ©, ©, *, ., ., ., ., ., ., .]
[., ., ., *, ©, ©, *, ., ., ., ., ., ., ., .]
[., ., *, ©, ©, *, ., ., ., ., ., ., ., ., .]
[., *, ©, ©, *, ., ., ., ., ., ., ., ., ., .]
[*, ©, ©, *, ., ., ., ., ., ., ., ., ., ., .]
[©, ©, *, ., ., ., ., ., ., ., ., ., ., ., .]
```

22.1. Torstai

Päivä meni JavaDocien kirjoittamisessa. Seuraavaksi aion vielä testata A*-algoritmini oikeellisuutta ja sen jälkeen laajentaa sen toimimaan kuvien kanssa.

1. Viikko kokonaisuudessaan

Opin viikon aikana käytännössä Dijkstran algoritmin ja sen toiminallisuuden täydellisesti sekä teorian toiminnallisuuden takana. Myös A* tuli erittäin tutuksi ja nyt ymmärrän, että miksi sen sanotaan olevan vain Dijkstran laajennos. Törmäsin myös muutamaan A*-algoritmin tehostamiseen liittyvään artikkeliin. Ensi viikko menee kuitenkin kokonaan omien tietorakenteiden toteuttamisessa, liikkuvan maalin tarkastelussa, algoritmin tehostamisessa sekä kuvien analysoinnin toiminnallisuuden toteuttamisessa. Hieman epäselvää kuitenkin vielä on, että miten toteutan liikkuvan maalin haun tehokkaasti.