Määrittelydokumentti

Kristian Wahlroos
wakr@cs.helsinki.fi

Aiheeksi työlleni olen ottanut A*-algoritmin toteutuksen. A* tulee työssäni esille niin, että ohjelmani analysoi ensin sille annettavan kuvan jossa on maali, lähtö ja staattiset esteet piirrettynä. Sitten ohjelma suorittaa itse A*-algoritmin hakemalla lyhimmän polun lähdöstä maaliin liikuttaen hahmoa samalla. Lisähaasteena työssäni on, että esimerkiksi keskellä kuvaa on liikkuvia esteitä, joita liikkuvan hahmon on väisteltävä tehokkaasti ja jopa liikkuva maali johon hahmoni on yritettävä päästä tehokkaasti.

Käytettävästä algoritmeista esille nousee A* ja tietorakenteista minimikeko, koska A* sisäinen toteutus luottaa siihen A* ollessa Dijkstran algoritmin laajennos heurestiikalla. Toisaalta jos löydän jonkun paremman ja nopeamman tietorakenteen A* varten kuin minimikeko, niin aion toteuttaa sen.

Ohjelmani ratkaisee ongelman ”Mikä on lyhin polku A:n ja B:n välillä?”. Käytän A*-algoritmia siksi, koska se on hyvin suosittu erilaisten tietokonepelien joukkojen liikkumisessa ja se on hyvinkin tehokas. Jos työni onnistuu hyvin, niin algoritmi tulee todennäköisesti käyttöön indie-henkiseen peliin kaverilleni.

Syötteenä ohjelmani saa kuvan, joka on piirretty käsin esimerkiksi .jpg-muodossa. Tämän kuvan väriarvot luetaan ja niiden perusteella määräytyy painot eri väreille ja alueille; musta väri tarkoittaa läpäisemätöntä aluetta, sininen vettä jonka läpi on hidasta kulkea, vihreä on maali ja valkoinen aluetta jonka läpi voi kulkea vapaasti. Piirtoalustan keskelle tulee liikkuva este tai monia esteitä, joten siihen on aina jätettävä hieman tilaa.

Tavoitteena oleva aikavaativuus on A*-algoritmille $O((|V| + |E|) \log |V|)$, jossa V = solmujen määrä ja E = kaarien määrä, tai parempi, jos löydän minimekoelle paremman vaihtoehdon. Tilavaativuus on $O(|V|)$, joka on myöskin sama kuin Dijkstrassa. Aikavaativuudet voivat nousta, koska liikkuvan esteen väistäminen/kiinniottaminen vaatii useita eri A* suorituksia, jotta väistettävän/haettavan sijainti aina päivittyisi, mutta tämä ongelma tulee esille paremmin vasta toteutuksen aikana. Toisaalta aikavaativuudet voivat myös laskea, jos löydän tai keksin tavan kehittää tehokkaampi ratkaisu algoritmille.

Lähteinä aion käyttää monipuolisesti Internettiä, koska ongelma on todella yleinen tietokonepelien liikkumisen tekoälyn keskuudessa.

Lähteitä:

- http://www.policyalmanac.org/games/aStarTutorial.htm
- http://fportfolio.petra.ac.id/user_files/04-021/MICEEI2010.pdf
- http://theory.stanford.edu/~amitp/GameProgramming/AStarComparison.html
- http://heyes-jones.com/astar.php