# A* reitinhaku labyrintissa

## Toteutusdokumentti

Käydään läpi toteutus luokkakohtaisesti. Jokaisesta luokasta esitellään tärkeimmät toiminnot ja tarpeen mukaan analysoidaan toteutuksen tehokkuutta.

Ennen luokkien tarkastelua kannattaa tutustua avoimen ja suljetun listan periaatteeseen. Avoimessa listassa ovat ne ruudut, jotka voivat mahdollisesti osua lyhyimmän reitin varrelle. Se on siis käytännössä lista, jossa on ne ruudut joita vertaillaan ja joista valitaan edullisin vaihtoehto (eli suurimman prioritettin omaava ruutu, eli se ruutu, jonka f-arvo on pienin). Suljetussa listassa on ne Ruudut, jotka on jo käyty läpi, eli ne joita ei oteta enää uudestaan käsittelyyn. Avoimen ja suljetun listan periaate on otettu ”*A* Pathfinding for Beginners*” tutoriaalin esimerkistä: http://www.policyalmanac.org/games/aStarTutorial.htm

1. **Ruutu**

A*-algoritmin labyrintti muodostuu ruuduista. Luokka pitää sisällään labyrintillä sijaitsevan (yhden) ruudun kaikki ominaisuudet. Luokka pitää sisällään etäisyyden alkuun (g-arvo), arvioidun etäisyyden maaliin (h-arvo), niiden summan (f-arvo), tiedon vanhemmasta ja siitä onko ruutu jo käsitelty.

Luokassa on myös metodi, jolla voidaan vertailla kahden ruudun arvoja toisiinsa, jota käytetään siinä tilanteessa kun halutaan tietää kummalla ruudulla on suurempi prioriteetti, eli kumman f-arvo on pienempi.

2. **Keko**

Luokka toteuttaa tietorakenteen minimibinäärikeko. Keko voi sisältää vain 'Ruutu'-luokan ilmentymiä. Keko toimii A*-algoritmin avoimena listana ja se on toteutettu taulukkorakenteena. Indeksointi toteutetussa keossa alkaa 1:sta ja ruutujen määrästä pidetään kirjaa.

Keko luokassa voidaan tehdä seuraavia asioita:

-luo keko: suuruus on labyrintin korkeus*leveys.

-lisää kekoon: lisättävälle ruudulle etsitään oikea paikka. Etsintä saattaa jatkua keon korkeuden verran eli aikavaativuus operaatiolle on O(log n)

-poista pienin: pienin on aina keon päällimmäisenä, joten aikavaativuus on O(1). Sen jälkeen keko voi olla kuitenkin epätasapainossa, koska viimeinen alkio on nostettu sen tilalle juureksi. Sen siirtäminen oikealle paikalleen on aikavaativuudeltaan O(log n).

-onko tyhjä: katsotaan onko ruutujen määrä 0, aikavaativuus O(1)

Luokkaan on toteutettu myös testaus, jossa lisätään Ruutuja listaan 'random' arvoilla ja sitten ne tulostetaan käyttämällä poistaPienin -metodia. Testattu toimivaksi, sillä arvot ovat aina tulostettu pienimmästä suurimpaan.

3. **AtahtiReitinhaku**

AtahtiReitinhaku-luokka pitää sisällään A*-algoritmin ja käyttää hyväkseen Keko-luokkaa tietorakenteenaan. Konstruktorina on labyrintti ja tärkeimpänä metodina on itse A* algoritmi. Heuristinen arvio maaliin lasketaan ruudun ja maalin x koordinaattien sekä ruudun ja maalin y koordinaattien erotusten itseisarvon summana.

A* -algoritmi käyttää avointa listaa (Keko) ja toistaa omaa algoritmiaan kunnes avoin lista on tyhjä. Matkalla tarkistetaan ollaanko maaliruudussa ja jos ollaan, niin merkataan paras polku korvaamalla jokaisen polulle osuvan ruudun '.' -merkki 'o' -merkillä. Tämä tehdään pakittamalla maalista alkuun ja kysymällä ruudulta mikä sen vanhempi on. Mikäli maalia ei löydy, niin palautetaan ”Ei ratkaisua”.

4. **Tulkki**

Tulkitsee labyrintin, eli laskee ja pyytää tarkistamaan sen korkeuden ja leveyden, jotta staattiselletaulukolle saadaan sopivat "mitat". Käytetään Testi -pääohjelmassa, sillä mikäli labyrintti ei täytä vaatimuksia, niin sitä ei kasitellä pidemmälle.

Tulkitseminen koostuu kahdesta osasta. Ensimmäinen osa laskee parametrinä annetun labyrintin korkeuden ja leveyden sekä tarkistaa ovatko kaikki rivit saman pituisia. Mikäli rivit ovat yhtä pitkiä, niin alustetaan labyrintti[korkeus][leveys]. Tällöin ollaan tarkastetttu, että labyrintti on symmetrinen ja samalla ollaan käyty läpi labyrintti riveittäin ja laskettu jokaisen rivin pituus. Aikavaativuus riippuu labyrintin muodosta ja siitä kuinka kauan rivi.length() vie aikaa suhteessa rivien määrän laskemiseen.

Tulkitsemisen toisessa osassa labyrintti skannataan ja talletetaan 2-uloitteiseen taulukkoon mikäli labyrintti on sisällöltaan kelvollinen. Labyrinttia luetaan siis merkki kerallaan ja tarkistetaan, että kaikki vastaan tulevat merkit ovat sallittuja merkkejä. Aikavaativuus on O(n).

*Muuta: O*hjelmaa voisi kehittää vielä niin, että etsimisen välivaiheita tulostetaan näytölle. Myös kaikki läpikäydyt ruudut ruudut voisi merkitä esim merkillä 'k', jolloin nähdään missä A*-algoritmi on labyrintillä käynyt, pinon geneerisyys, graafinen käyttöliittymä.

# Lähteet


**http://www.policyalmanac.org/games/aStarTutorial.htm**
**http://www.cs.helsinki.fi/u/floreen/tira2013/tira.pdf**
**http://www-cs-students.stanford.edu/~amitp/gameprog.html#Paths**
**http://heyes-jones.com/astar.php**
**http://en.wikipedia.org/wiki/A*_search_algorithm**