Aiheenmäärittely

Labyrinth- ohjelman tarkoituksena on löytää nopein reitti labyrintista ulos. Käyttäjä saa eteensä tyhjän labyrintin eli "maa-alueen." Käyttäjä asettaa laudalle esteitä (seiniä), joiden läpi ei voi liikkua, luoden näin oman labyrintin. Tämän jälkeen pelaajan tehtävänä on asettaa labyrintin uloskäynti sekä lähtöpiste, josta ohjelma lähtee liikkeelle.

Ohjelman on tarkoitus etsiä näitä tietoja hyväksi käyttäen jokin nopeimmista reiteistä ulos labyrintista. Kun ohjelma on löytänyt nopeimman reitin, antaa se käyttäjälle tiedon reitin käyttämistä "maapaloista" värjäämällä kuljetun polun.

Algoritmina käytän A*- algoritmia, joka toimii seuraavasti:
Ohjelma arvottaa pelin "maapalat" etäisyyden mukaan eli asettaa niille heuristiset arvot. Tämän jälkeen ohjelma ottaa avoimelta listalta *maapalan,* jolla on pienin kokonaisarvo ja “tarkastelee” sitä. (Aluksi listalla on vain aloituspiste, joten se on ensimmäinen tarkasteltava *maapala*)

Ohjelma alustaa tarkasteltavan *maapalan* naapureidensa “vanhemmaksi” eli parent- solmuksi, sillä tarkasteltava solmu on juuri se solmu, joka on johdattanut naapurisolmuun. Naarureille asetetaan myös kokonaisarvo, joka lasketaan seuraavasti: *heuristinen arvo + liikkumiskustannus(10) + (vanhemman kokonaisarvo - vanhemman heuristinen arvo)*. Tämän laskettuaan naapurit lisätään avoimelle listalle eli listalle, johon kuuluvat maapalat, joita ei ole vielä tarkasteltu. Tarkasteltu alkio puolestaan lisätään suljetulle listalle, johon kuuluvat kaikki jo tarkastellut alkiot.

Algoritmi tarkistaa vain ylä- ja alapuolella olevat naapurit sekä oikealla ja vasemmalla puolella olevat naapurit. Labyrintissa voi siis liikkua ainoastaan vertikaalisesti tai horisontaalisesti. Jokaisen tarkasteltavan *maapalan* kohdalla tehdään yllä mainitut toimenpiteet, joten ohjelmalla on kokoajan tieto siitä, mitkä maapalat on jo tutkittu ja mitkä maapalat odottavat vielä vuoroaan. Näiden listojen avulla algoritmin tehokkuus saadaan O(|E|) aikaan eli kaikki solmujenväliset siirtymät käsitellään korkeintaan kerran nopeimman reitin löytämiseksi.

Tietorakenteet:

Tietorakenteina käytän ohjelmassani taulukoita, minimikekoa sekä linkitettyä listaa, jotka sisältävät labyrintin maapaloja. Näillä maapaloilla on tieto omista vanhemmistaan "parenteista", joita pitkin kulkien lopullinen reitti voidaan saada lopun löydyttyä esiin. Toteutan itse avoimen listan sekä suljetun listan toiminnallisuudet eli toteutan listalle lisäämistoiminnon, listalta poistamistoiminnon, sekä listalta etsimistoiminnon.

Avoimena listana käytetty minimikeko vaatii myös Heapify- metodin toteutuksen, joka takaa, että minimikeon ehdot toteutuvat keossa.

Labyrinttina toimii 2D- taulukko, joka sisältää verkon solmut (maapalat.)