Projektisuunnitelma: Labyrintti

Henkilötiedot:

Henrik Mäntylä 788429
Automaatio- ja systeemitekniikka
2019
25.02.2022

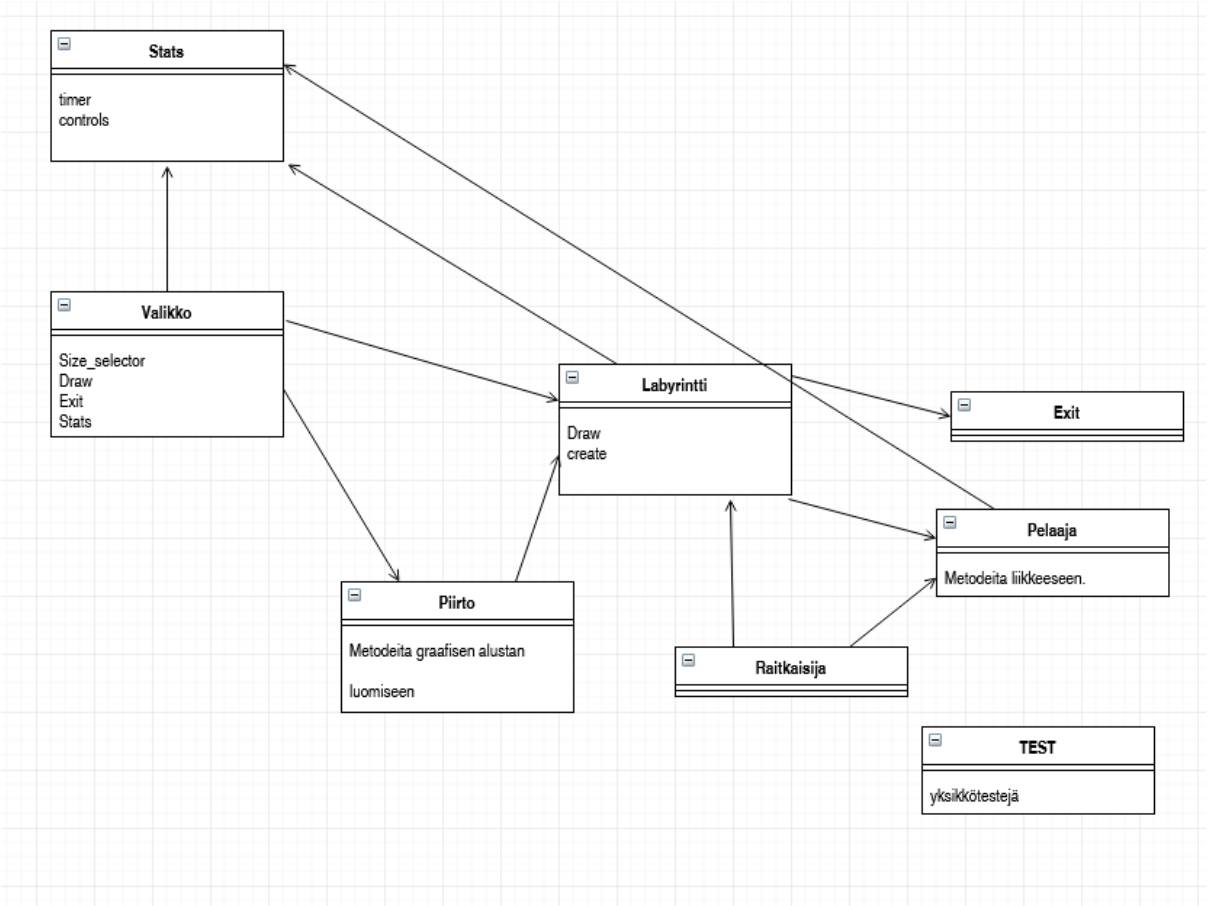
Yleiskuvaus ja vaikeustaso:

Lähden tekemään vaativan vaikeustason labyrinttiä. Kuvaus on sama mitä projektiaiheen alla. Tarkennuksena siihen, lähtökohtaisesti lähden tekemään weave-tyyppistä labyrinttiä. Sekä vaativan työn työn lisäominaisuuksista valitsen järkevän pisteytyksen ja tuloslistan, sekä tehokkaat luonti- ja ratkaisualgoritmit.

Käyttötapauskuvaus ja käyttöliittymän luonnos:

Koska kyseessä on labyrintti ideana on, että graafinen käyttöliittymä on simppelin näköinen, mutta haastava ratkaista. Ohjelman käynnistyessä esiin tulee valikko mistä valitaan labyrintin koko, tämän jälkeen esiin tulee haluttu labyrintti ja hiiri siirtyy labyrintin keskelle, josta sen on päästävä pois. Näkyviin tulee myös aika, joka lähtee samantien liikkeelle, sekä nappi mistä voi luovuttaa. Hiirtä ohjataan labyrintissä nuolinäppäimillä. Kun labyrintti on suoritettu, joko käyttäjän tai tietokoneen toimesta, esiin tulee taas alkuvalikko, jossa näkyy myös viimeisimmän yrityksen tulokset.

Ohjelman rakennesuunnitelma:



Tässä vaiheessa vielä simppeli. Todennäköisesti tulee paljon muutoksia.
Luokat ja niistä nopeat selitykset:

Valikko: Hoitaa alkuvalikon, jossa näkyy edellisen pelin tulostaulukko, sekä kaksi nappulaa exit ja labyrinthin koon valitsin. Kysyy tulokset stats luokalta. Tarvitsee piirtoluokkaa ilmestyäkseen ja lähettää labyrintille käskyn valmistella halutun koon mukainen labyrintti.

Stats: Ottaa ylös haluttuja tietoja pelin kulusta ja lähettää ne eteenpäin.

Piirto: Huolehtii graafisesta alustasta.

Labyrintti: Rakentaa labyrintin, ja sijoittaa sinne pelaajan sekä ulospääsyn. Kutsuu ratkaisijaa, jos luovutus nappia on painettu.

Pelaaja: Huolehtii pelaajan liikkumisesta labyrintissä

Seinä: Läpipääsemätön este.

Ratkaisija: Ratkaisee luovutetun pelin.

Exit: Maali

TEST: tänne kaikki yksikkötestit.

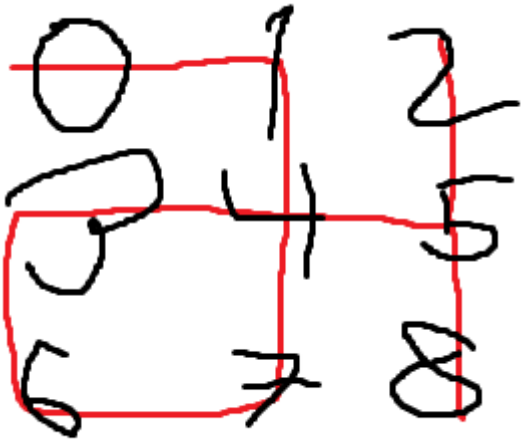
Tietorakenteet:

Käytetään verkkoa labyrintin pohjan mallintamiseen. Jokainen ruutu on liitetty aina 1-4 naapuriinsa. Voimme myös laittaa erilaisia painoarvoja ruutujen välille. Verkkoa on helppo kuvata matriisina.

Kun käymme verkon läpi jollakin algoritmilla saamme siitä puu rakenteen. Esimerkiksi Primin algoritmistä saamme MST:n.

Tiedostot ja tiedostoformaatit:

Oletan että vaatimuksena on mahdollisuus tallentaa ja tuoda erilaisia labyrinttejä tiedostoista.
Voimme tuoda ja tallentaa suoraan esimerkiksi MST joka kuvaa jo meidän labyrinttiä.
Esimerkkimuoto matriisista:
[[0, 1, 0, 0, 0, 0, 0, 0, 0],
[1, 0, 0, 0, 1, 0, 0, 0, 0],
[0, 0, 0, 0, 0, 1, 0, 0, 0],
[0, 0, 0, 0, 1, 0, 1, 0, 0],
[0, 1, 0, 1, 0, 1, 0, 1, 0],
[0, 0, 1, 0, 1, 0, 0, 0, 1],
[0, 0, 0, 1, 0, 0, 0, 1, 0],
[0, 0, 0, 0, 1, 0, 1, 0, 0],
[0, 0, 0, 0, 0, 1, 0, 0, 0]]
Matriisin ylin rivi kuvaisi ruudun 0 linkkejä jne.
Joka kuvaisi, jotain tämän kaltaista labyrinttiä.



Algoritmit:

Ainakin kaksi kappaletta tarvitaan labyrintin luomiseen sekä sen ratkaisemiseen.
Luomiseen on hyviä esimerkkejä jo tarjotussa linkissä ja weblog.jamisbuck.org on lisää hyviä ajatuksia labyrintin luomiseen. Esimerkiksi voimme käyttää hänen growing tree algoritmia,

jonka ideana on aloittaa jostain ruudusta ja tutkia sen naapureita, jos ne ovat tyhjiä luodaan linkki. Kunnes saavutaan paikkaan, josta ei voida enään luoda linkkejä, sitten palataan takaisin päin kunnes löytyy ruutu jonka kaikkia naapureita ei ole vielä käyty läpi. Weave ominaisuus saataisiin lisäämällä siihen, jos naapurissa on reitti 90 asteen kulmassa ja siitä suoraan eteenpäin on tyhjä ruutu voimme jatkaa sinne.

Ratkaisemiseen löytyy myös paljon vaihtoehtoja antamastanne nettisivulta. Voimme esimerkiksi testata wall follower algoritmia ja jos se ei toimi testaamme jotain muuta vaihtoehtoa.

Testaussuunnitelma:

Tärkeimmät testattavat ovat varmaan algoritmit, jotka vastaavat labyrintin rakentamisesta ja ratkaisemisesta. Weave ominaisuuden kanssa saattaa varmaan tulla ongelmia joten siihen kannattaa kiinnittää huomiota. Myös statistiikkaa ylläpitävää luokkaa täytyy testata, että se toimii.

Kirjastot ja muut työkalut:

Ainakin PyQt, ja mahdollisesti random. En osaa yhtään sanoa ennen kuin aloitan itse koodia. Randomia saatan haluta käyttää labyrintin linkkien painoarvoihin.

Aikataulu:

Alustavasti lähden tekemään projektia kuten tähän mennessä tämän kurssin muitakin tehtäviä, eli kerran viikossa varaan yhden päivän tämän kurssin tehtäville. Jos aikaa on noin kymmenen viikkoa ja yhdessä päivässä tekisin noin 8h töitä siitä tulisi noin 80h töitä. Jos lopussa näyttää siltä, että tarvitaan lisää pystyn varmaan löytämään lisää aikaa jostakin.

Aluksi aloitan labyrintin muodostamisesta ja graafisesta ikkunasta. Sitten yritän saada pelaajan ja exitin sisään, jotta saadaan toimiva peli. Sitten ratkaisualgoritmi. Lopuksi mukaan aloitusnäyttö ja statistiikat. Jos jää aikaa voin lisätä tai muokata asioita nopeammaksi.

Ekaan palautukseen vähintään labyrintin muodostaminen ja siihen graafinen liittymä.

Toiseen palautukseen toimiva peli ja ratkaisualgoritmi.

Kirjallisuusviitteet ja linkit:

Think Labyrinth: http://www.astrolog.org/labyrnth/algrithm.htm
https://tarjotin.cs.aalto.fi/cs-a1141/OpenDSA/Books/CS-A1141/html/GraphIntro.html
https://weblog.jamisbuck.org/2011/1/27/maze-generation-growing-tree-algorithm
https://weblog.jamisbuck.org/2011/3/4/maze-generation-weave-mazes.html