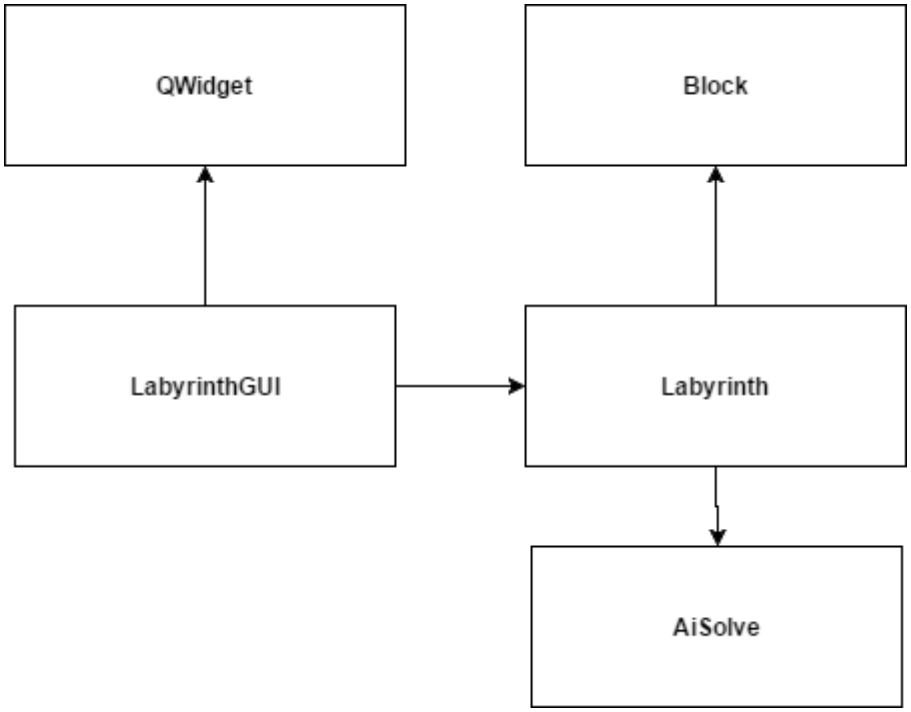
Tekninen suunnitelma
Labyrintti

Jani Hyrkäs. 348348

Ohjelman rakennesuunnitelma



LabyrinthGUI luo pelille ikkunan ja napit joilla voi käyttää Labyrinth luokkaa. LabyrinthGUI on siis sananmukaisesti graafinen käyttöliittymä Labyrinth luokalle, ja yhdistää useat napit luokan metodeihin. Luokka perii pääasiassa luokasta QWidget tarvitsemansa välineet graafiseen esittämiseen. LabyrinthGUI tulee myös todennäköisesti käyttämään useita muita PyQt:n tarjoamia luokkia.

Labyrinth luokka generoi labyrintin määriteltyjen parametrien eli koon ja mahdollisen vaikeuden mukaan, joka todennäköisesti perustuu aloituspaikan ja maalin väliseen etäisyyteen. Generoitu labyrintti koostuu alustavasti pelkistä palikoista joilla on seinä kaikkiin suuntiin. Sen jälkeen se luo reitin annetun algoritmin mukaan. Toinen tapa olisi tehdä rakennusalgoritmille oma luokka.

-current_position: Pelinappulan sen hetkinen paikka labyrintissä. Huomioi x-, y- sekä z-akselin. Päivittyy aina kun pelaa liikuttaa onnistuneesti pelinappulaa.

-generate_labyrinth: Luo labyrintin.

Block luokka sisältää labyrintin tarvitsemat erilaiset palikat joissa on joko kulkuyhteys yhteen tai useampaan ilmansuuntaan, taikka seinä. On myös mahdollista, että palikka on eri tasolla verrattuna muihin eli on silta tai toinen kerros. Eli yhdestä palikasta on mahdollista liikkua viiteen muuhun palikkaan.

AiSolve on pelin automaattista selvittämistä varten oleva luokka, joka pääasiassa sisältää erilaisia algoritmeja labyrintin selvittämistä varten.

Käyttötapauskuvaus

Avattaessa ohjelma tuo esiin ikkunan jossa on tyhjä tila labyrintille, ohjaus labyrinttiä varten sekä osio jossa voi asettaa labyrintille erilaiset parametrit. Tässä LabyrinthGUI osio aktivoituu luodakseen tarvittavan ikkunan ja sille kaikki napit jotka ovat vastaavasti linkitetty niiden omiin toimintoihin. Sen jälkeen käyttäjä voi joko ladata olemassa olevan labyrintin taikka generoida uuden, valitsemillaan parametreilla. Ohjelman ikkunassa on tekstikenttä johon käyttäjä voi kirjoittaa haluamansa labyrintin koon antamalla x- ja y-akselin. Käyttäjä sen jälkeen painaa ”Generoi labyrintti” nappulaa joka sitten luo Labyrinth luokassa labyrintin käyttämällä Block luokkaa palikkoihin. Generoitu labyrintti sen jälkeen kuvataan graafisesty LabyrinthGUI luokassa ja esitetään ohjelman ikkunassa. Käyttäjä on asetettu satunaiselle paikalle labyrintissä. Käyttäjä pystyy liikuttamaan pelinappulaansa joko graafisesti ikkunan nappuloilla, taikka näppäimistön nuolinäppäimillä. Mikäli pelaaja ei halua pelata loppuun asti, hän painaa ikkunassa olevaa automaattisen ratkaisun nappia, joka aktivoi AiSolve luokan algoritmin Labyrinth luokan kautta, ja vie pelinappulan maaliin asti. Halutessaan pelaaja voi ikkunassa olevalla tallennus napilla tallentaa sen hetkisen labyrintin myöhempää varten.

Algoritmit

Tärkein algoritmi tulee olemaan labyrintin luomiseen tähtäävä. Tämä tulee toimimaan niin että pelaajan aloituspaikka ja labyrintin maali arvotaan annetusta labyrintin alueesta. Näille pisteille määritellään minimi etäisyys. Tämän jälkeen ohjelma arpoo aloituspaikasta jonkin suunnan josta se avaa seinän, katon taikka lattian mikäli se sijaitsee sillä hetkellä jo toisessa kerroksessa. Ohjelma siirtyy sen jälkeen avattuun palikkaan, kunhan se ei sijaitse labyrintin ulkopuolella, ja toistaa aikaisemman vaiheen. Ohjelma toistaa tätä niin kauan kunnes saapuu maalipisteeseen. Tämän jälkeen ohjelma etsii satunnaisen palikan joka on jo avatun palikan vieressä, mutta jossa itsessään ei ole avattua seinää, ja lähtee tästä paikasta toistamaan ylempää kaavaa, kunnes törmää joko ulkorajaan tai palikkaan jossa on avattu seinä. Tätä seinää ohjelma ei tietenkään avaa. Tätä etsimisvaihetta ohjelma toistaa niin kauan, kunnes kaikki palikat on käyty läpi.

Toinen ohjelman algoritmi olemaan AiSolve:n käyttämä, jonka avulla pelaaja voi automatisoida labyrintin ratkaisemisen. Helpoin tapa ohjelmoida ratkaisu on laittaa pelinappula noudattamaan oikean- tai vasemmankäden sääntöä, jossa pelinappula kääntyy yhteen määriteltyyn suuntaan aina kun mahdollista. Mikäli siihen suuntaan jatkaminen ei ole mahdollista, pyrkii pelinappula kulkemaan suoraan, tahi viimeisenä vaihtoehtona kääntyy vastakkaiseen suuntaan kuin on määritelty. Ohjelma noudattaa tätä siihen asti, kunnes on selvittänyt labyrintin. Muita algoritmeja tähän löytyy lukuisasti muun muassa Wikipediasta, ja niitä tulee todennäköisesti lisättyä, kun tärkeämmät prioriteetit ohjelman kannalta on saatu tehtyä.

Tietorakenteet

Labyrintin Block oliot varastoidaan kolmiulotteiseen listaan perustuen paikkaa x- ja y-akselilla, sekä z-akseli kahden kerroksen verran. Alustavasti tämä tuntuu helpoimmalta toteuttaa, vaikka se voikin suuremmilla labyrinteillä olla aikaa vievä, varsinkin labyrintin luomisessa riippuen käytetystä

algoritmista. Yksi mahdollisuus olisi käyttää Numpy:n kaltaista ulkoista kirjastoa kätevää taulukoimista varten.

Aikataulu

Ensimmäinen viikko, 17.3 – 24.3: Tässä vaiheessa pyritään toteuttamaan LabyrinthGUI ilman yhteyttä Labyrinth luokkaan. Eli koodataan ikkuna sekä siihen tarvittavat nappulat. Tämän lisäksi luodaan palikat annetun labyrintin kokoparametrin mukaan graafisesti ikkunaan ja niille seinät.

Toinen viikko, 25.3 – 31.3: Luodaan Labyrinth luokka ja sen käyttämä kolmiulotteinen lista. Tässä vaiheessa luodaan myös yksinkertainen Block luokka, jotta sitä voidaan käyttää Labyrinth luokan listassa. Lista alustetaan näillä palikoilla joilla on pelkät seinät. LabyrinthGUI yhdistetään Labyrinth luokkaan ja piirretään käyttäen Labyrinth luokan listaa.

Kolmas viikko, 1.4 – 7.4: Lisätään Labyrinth luokkaan luomisalgoritmi ja testataan että se käy koko kolmiulotteisen listan läpi. Tämä piirretään ohjelman ikkunaan ja tarkistetaan että se toimii eri parametreilla sekä luo toimivan labyrintin. Lisätään Block luokkaan tarvittavia osia.

Neljäs viikko, 8.4 – 14.4: Lisään pelinappulan sekä sille tarvittavat osat Block sekä Labyrinth luokkaan. Luon pelinappulan liikuttamiseen tarvittavan koodin sekä yhdistän LabyrinthGUI ohjelmaikkunan liikkumiseen tarkoitetut napit Labyrinth luokan koodiin. Testaan liikkumisen toimivuutta sekä miten pelinappula olisi paras graafisesti esittää.

Viides viikko, 15.4 – 21.4: Lisään labyrintin sekä pelinappulan tallennus ja lataus mahdollisuuden Labyrinth luokkaan, ja yhdistän nämä ohjelmaikkunan nappuloihin. Alan työstää sitä mahdollisuutta, että peliruutu liikkuu pelinappulan mukana.

Kuudes viikko, 22.4 – : Alan hioa ohjelmaa fiksummaksi, varsinkin käyttöliittymän kannalta, sekä pyrin parantamaan ulkoasua. Samoin alan lisäillä tuloslistaa ajan sekä todennäköisesti käyttäjän komentomäärän suhteen. Nämä tulevat olemaan jokaiselle labyrintille omat, ja tulevat tallentumaan labyrinttien kanssa. Ne tulevat olemaan ohjelmaikkunan kautta nähtävissä jonkinlaisessa muodossa.

Yksikkötestaussuunnitelma

Tämä projekti pohjautuu loppuvaiheessa vahvasti graafiseen käyttöliittymään labyrintistä, ja sen testaus toimii aika lailla pelkästään kokeilemalla säätää ikkunan kokoa sekä esimerkiksi kokeilla, että kaikki implementoidut napit tekevät mitä niiltä odotetaan.

generate_labyrinth metodi on yksi keskeinen asia projektissa, ja sen lopputulosta on monimutkaista tarkastella ilman jonkinlaista graafista toteutusta, kuten merkeillä taikka PyQt:n graafisilla osilla. Mutta kun tämä graafinen osio on toteutettu, voi aluksi verrata, että Labyrinth kolmiulotteisen listan oliot vastaavat graafisen esityksen palikoita siten että kaikki ovat oikeilla paikoilla. Sen jälkeen täytyy itse läpäistä labyrintti siten että näkee että sillä on ratkaisu eri parametreillä.

Automaattisen ratkaisun algoritmia voi testata myös sen jälkeen, kun graafinen osio on toteutettu. Tätä voisi aluksi testata niin että tarkistaa että algoritmi pystyy läpäisemään erilaisia generoituja labyrinttejä. Mikäli se pystyy siihen, niin voisi tarkastella sen tehokkuutta esimerkiksi niin että merkitsee ratkaisijan kulkeman reitin graafisesti, ja tarkastelee että se vastaa suunniteltua.

Kirjallisuusviitteet ja linkit

https://en.wikipedia.org/wiki/Maze_solving_algorithm - Erilaisia labyrintin ratkaisemiseen käytettäviä algoritmeja. Alustavasti käytetään oikean- tai vasemmankäden algoritmia, myöhemmin mahdollista lisätä muita.

http://www.astrolog.org/labyrnth/algrithm.htm - Täältä tutkin erilaisia luomisalgoritmeja kolmiulotteisille labyrinteille. Alustavasti päädyin käyttämään ”Hunt and kill” tapaista algoritmia.

http://pyqt.sourceforge.net/Docs/PyQt5/ - Tätä käytän PyQt kirjaston tutkimiseen ja kuinka se erottuu C++ Qt versiosta oleellisesti. Tuolta tulen tukimaan lukuisia eri luokkia ja miten niitä voi mahdollisesti käyttää. Tässä vaiheessa on vaikea sanoa mitä kaikkia tulen käyttämään koska PyQt ja Qt ovat hyvin laajoja.