1. Henkilötiedot

   Ohjelmoinnin peruskurssi Y2, projektityön dokumentti

   Jani Hyrkäs, 348348, EST, 12.5.2017

2. Yleiskuvaus

   Tässä projektissa on luotu yksinkertainen PyQt:n avulla graafisesti toimiva labyrinttipeli. Labyrintit peli luo satunnaisesti ja labyrintit ovat kolmiulotteisia. Peli asettaa pelaajan ennalta määritettyyn paikkaan ja pelaajan on tarkoitus ohjata pelinappula ennalta määritettyyn paikkaan.

   Pelaajalla on mahdollisuus antaa pelin suorittaa labyrintti loppuun, näyttäen algoritmin läpikäymät palikat. Pelaaja voi myöskin tallentaa senhetkisen labyrintin ja avata se myöhemmin halutessaan.

   Lisäominaisuuksia ei ole joten projekti täyttää vain helpon tehtävän vaatimukset. Eli toisin kuin alun perin suunniteltu, peli ei sisällä pistelistaa eikä ylimääräisiä algoritmeja joita pelaaja voisi valita. Labyrintin kokoa ei myöskään pysyt pelaaja määrittämään

3. Käyttöohje

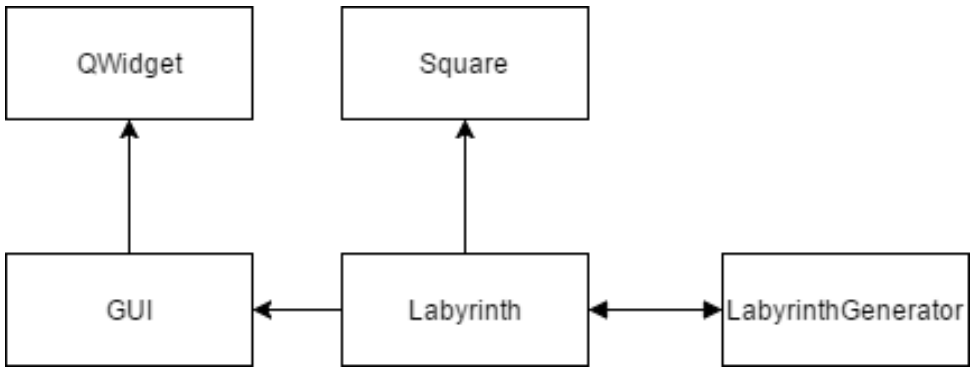
   Ohjelman käynnistyy suorittamalla main.py tiedoston, joka avaa graafisen peli-ikkunan. Pelialue on aluksi tyhjä sillä peli ei alustavaa labyrinttiä, vaan pelaaja joutuu painamaan ’Generate labyrinth’ nappia, joka sitten luo satunnaisen labyrintin peli-ikkunaan. Mikäli labyrintti on jotenkin puutteellinen, tulee ikkunan oikeaan alakulmaan viesti jossa tästä ilmoitetaan, ja pyydetään generoimaan uusi labyrintti.

   Pelinappula sijoitetaan labyrintissä vasempaan yläkulmaan, ja maali löytyy ylemmän tason oikeasta alakulmasta. Seinät ovat pelissä mustia kuutioita, pelaaja sininen kuutio ja maali on purppura. Tämän lisäksi pelissä on tikapuita joiden avulla voi siirtyä ylemmän ja alemman tason välillä, menemällä näiden kohdalle ja painamalla oikealla alakulmassa olevaa ’Use ladder’ nappia. Pelaaja voi myöskin vaihdella näkymää tasojen välillä painamalla ’Change floor’ nappia.

   Huomioitavaa on että pelinappula ei piirry tikapuiden päälle ikkunassa, joten on pidettävä huomiota mihin liikkuu. Pelissä voi liikkua joko neljän liikkumanapin avulla jotka sijaitsevat vasemmassa alakulmassa, taikka näppäimistön nuolinäppäimillä.

   Alhalla keskellä sijaitsevat tallennus, lataus sekä automaattinen ratkaisu -napit. Automaattinen ratkaisunappi selvittää tien maaliin ja sijoittaa pelaajan sinne, sekä merkkaa läpikäymänsä napit sinisellä.

4. Ohjelman rakenne



GUI luo ohjelmalle ikkunan ja siihen napit jotka ovat liitetty Labyrinth luokkaan. Sen lisäksi se määrittää eri osa-alueille paikat ikkunassa. Suurimman osan ominaisuuksista luokka perii PyQt:n luokasta QWidget.

Labyrinth hoitaa itse labyrintin piirtämisen peli-ikkunaan, sekä päivittää kolmiulotteista matriisia joka koostuu Square -luokista. Kolmiulotteinen matriisi sekä peli-ikkuna päivittyy GUI luokasta tulevilla nappien liitoksilla, joihin pelaaja antaa syötteen. Labyrinth luokka myöskin sisältää 'automatic_solver' metodin joka hoitaa pelin automaattisen ratkaisemisen ja päivittää samalla kolmiulotteisen matriisin.

Square luokka on jokaiselle yksittäiselle pelipalikalle kuuluva luokka, joka pitää sisällään palikoiden tärkeimmät tiedot, kuten sisältääkö se pelaajan, onko palikka seinä, sisältääkö se tikapuut, onko se maalialue ja palikan x-, y- sekä z- arvon. Square luokka pystyy myöskin päivittämään sisältämiään tietoja erilaisilla metodeilla.

LabyrinthGenerator generoi vaaditun labyrintin kolmiulotteisen matriisin ja palauttaa sen luokalle Labyrinth. Sisältää siis algoritmin jolla labyrintti luodaan.

5. Algoritmit

Ohjelmalla on kaksi oleellista algoritmia. Eli algoritmi joka generoi labyrintin kolmiulotteisen matriisin, sekä algoritmi joka automaattisesti ratkaisee labyrintin pelaajan senhetkisestä paikasta.

Generointi algoritmin koodi noudattaa hieman "Randomized Prim's' algoritmia [1].

Aluksi generoidaan kolmiulotteinen labyrintti koostuen Square luokista, jotka alustetaan kaikki seiniksi. Sen jälkeen lisätään pelaaja sille kuuluvalle palikalle, josta sitten poistetaan seinä, ja jonka vieressä olevat seinät lisätään erilliseen listaan johon lisätään ja poistetaan seiniä sitä mukaan kun labyrinttiä generoidaan. Pelaajan palikan vierailtu arvo myöskin muutetaan True arvoon.

Seuraavaksi otetaan satunnaisesti kyseisetä listasta yksi seinä, josta tarkistetaan kaksi ympärillä olevaa vastakkaista palikkaa. Mikäli näiden molempien palikoiden vierailtu arvot ovat True, muutetaan alkuperäinen seinäpalikan vierailtu arvo myöskin muotoon True, ja se poistetaan listasta.

Mutta mikäli vain yksi näistä vastakkaisista palikoista on True, muutetaan alkuperäinen seinäpalikka avoimeksi väyläksi, muutetaan sen vierailtu arvo muotoon True, ja sitten vastakkaisen palikan kaikki viereiset seinät lisätään listaan, joka koostuu vierailemattomista seinistä.

Tätä toistetaan niin kauan kunnes lista on tyhjä.

Labyrintin ratkaisualgoritmi noudattaa periaatetta jossa se pyrkii aina kääntymään vasemmalle itsestään katsoen ja liikkumaan siihen suuntaan. Mikäli se ei tässä onnistu, se valitsee uuden suunnan ja pyrkii liikkumaan sinne. Tätä algoritmi toistaa siihen asti kunnes se tunnistaa että senhetkinen palikka on sama kuin maalipalikka.

6. Tietorakenteet

Labyrintin Square oliot varastoidaan kolmiulotteiseen matriisin perustuen paikkaa x- ja y-akselilla, sekä z-akselilla. Alustavasti tämä tuntui helpoimmalta toteuttaa, vaikka sen läpikäyminen voikin suuremmilla labyrinteillä olla aikaa vievä, varsinkin labyrintin luomisessa.

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

7. Tiedostot

   Ohjelma käyttää ainostaan labyrintin kolmiulotteisen matriisin tallentamiseen tiedostoa. Tähän on yksinkertaisesti käytetty Pythoniin sisältyvää Pickleä, joka luo ja lukee käsittelemänsä datan save_file.pickle tiedostoon.

8. Testaus

   Suurin ongelma ohjelmassa oli sen kaatuminen ilman virheilmoituksia. Tätä varten lisäsin joitakin testejä joiden avulla pystyin seuraamaan mitä kaikkea ohjelma kerkesi suorittamaan ennen kaatumistaan, ja jotta pystyisin korjata ongelmakohdat.

   Ohjelmaa työstin tekemällä pieniä osia tarkistaen että ne toimivat kunnolla, ennen kuin lisäsin uusia ominaisuuksia ohjelmaan.

   Graafiset osat testasin liittämällä ne johonkin toimintoon ja tarkastin olivatko kytkennät onnistuneita.

   .

9. Ohjelman tunnetut puutteet ja viat

   Labyrintin generoimiseen toteuttamani algoritmi ei ole täydellinen, sillä joskus seinistä koostuvaan listaan jää pyörimään seinä tai seiniä joita se ei kykene onnistuneesti käymään läpi. Tämä aiheuttaa jatkuvan while() -loopin. Olen tällä hetkellä kiertänyt ongelman pistäen algoritmin tarkistamaan milloin se käy samoja seiniä läpi epäonnistuneesti, ja pyytämällä näin käydessä käyttäjää generoimaan uuden labyrintin. Tehokkaiden ja toimivien algoritmien luonti on minulle vielä hieman vaikeaa.

   Pelaajan edetessä pelissä nappulaa liikuttamalla, pelin reagoivuus hidastuu ja jostain syystä kuluu enemmän aikaa ennen kuin pelinappula liikkuu kentällä sen jälkeen kun pelaaja on antanut syötteen. En ole vielä ihan varma mistä tämä johtuu, mutta epäilen sen alustavasti liittyvän siihen kuinka pelikenttä saattaa päivittyä ruutuun turhan usein. Pohdin että tätä voisi parantaa jotenkin lisäämällä jonkinlaisen rinnakkaisen suorituksen peliruudun päivitykselle.

   Tällä hetkellä myöskin ’Change floor’ nappulan toiminallisuus välillä ontuu, sillä joskus sitä joutuu painamaan useamman kerran ennen kuin peliruutu päivittyy.

10. 3 parasta ja 3 heikointa kohtaa

    Vaikka graafinen ulkoasu on aika karkea, pidän sitä kuitenkin onnistuneena sillä se toimii tällä hetkellä jotakuinkin fiksusti ja suunnitellusti.

    Heikoimmat kohdat ovat tällä hetkellä hyvin pitkälti samat kuin ohjelman tunnetut puutteet ja viat kohdassa.

    Lisäksi lisäisin heikkoudeksi sen kuinka välillä pelinappula ja tikapuupalikat menevät päällekkäin, ja pelinappula katoaa näkyvistä sen alle käytännössä.

    Tämän lisäksi mielestäni yksi heikko kohta on se kuinka tällä hetkellä en onnistunut lisäämään luotettavasti erikokoisten labyrinttien luomisen.

    Myöskin koodin kommentointi on jäänyt pois kokonaan.

11. Poikkeamat suunnitelmasta

    Suunnitelmasta poiketen monia lisäominaisuuksia jäi lisäämättä. Tämä johtuu pitkälti vähäisestä ajankäytöstä. Toteutusjärjestys noudatti pitkälti sitä mitä olin suunnitellut.

12. Toteutunut työjärjestys ja aikataulu

    Välillä 17.3 -26.3 toteutin aikalailla suunnitelman mukaan projektia. Mutta sen jälkeen läheisen äkillisen sairastumisen ja poismenon johdosta koulumotivaatio oli laskenut paljon, enkä saanut sen johdosta projektia toteutettua suunnitellusti.

    Tämän johdosta suurin osa projektista tuli tehtyä päivämäärän 7.5 jälkeen.

13. Arvio lopputuloksesta

    Oma mielipide toteutetusta projektista jäi lopuksi hieman pettymykseksi sillä puutteita on paljon muun muassa kommentoinnissa sekä algoritmien toteutuksissa. Olen myös sitä mieltä että parempi koordinaattien käyttö ja läpikäyminen olisi tehostanut algoritmeja huomattavasti. Algoritmit olisivat muutenkin päällimmäisenä asiana joita parantaisin.

    Samoin pyrkisin poistamaan pelinappulan liikkumisen hidastumisen jotta pelimukavuus säilyisi pidemmälle mentäessä, sillä tällä hetkellä sen graafista esitystä joutuu hieman odottamaan.

    Mikäli algoritmit saisi toteutettua kunnolla, voisi labyrintin kokoa helposti kasvattaa ja muuttaa käyttäjän halun mukaan. Pienillä muutoksilla olisi myös mahdollista lisätä useampia kerroksia labyrinttiin, tämänhetkisen kahden sijaan.

14. Viitteet

    Mitä kirjoja, nettisivuja tai muuta materiaalia olette käyttäneet? Kaikki lähteet tulee ilmoittaa, vaikka niihin kuuluisivat pelkkä kurssilla käyttämänne oppikirja ja perusluokkakirjastojen API-kuvaus.

    [1] https://en.wikipedia.org/wiki/Maze_generation_algorithm#Randomized_Prim.27s_algorithm
    Täältä sain ohjeistusta labyrintin generoimiseen tarvittavaan algoritmiin.

    https://docs.python.org/3/library/pickle.html

    Labyrintin tallennusta ja lataamista varten tutustuin Python pickle osioon.

    http://pyqt.sourceforge.net/Docs/PyQt4/classes.html

    Vaikka minulla olikin hieman kokemusta Qt:n käytöstä C++ kielen kautta, niin minulla oli kuitenkin tarve tutustua Pythonin versioon siitä.