Henrik Granö, 528 634 26.2.2018

Bioinformaatioteknologian kandidaattiohjelma, vuosikurssi 2016

CS-A1121

Projektin tekninen suunnitelma

Ohjelman rakenne/luokat:

GUI: Pääikkuna (QMainWindow) jossa näytetään valikko sekä peli. Pelin käynnistyttyä vaihdetaan attribuuttia QCentralWidgettiä peli-ikkunaolioon **Gamewidget**.

Gamewidget: Peli-ikkuna joka näytetään pääikkunassa pelin käynnistyttyä. Tähän piirretään kenttä **Game:in** kentän mukaisesti sekä pelaajahahmo.

Metodit:

- **draw_map:** piirtää kentän näkyviin

- **draw_player:** piirtää pelaajahahmon näkyviin

Game: Peliolio, joka sisältää mm. seuraavat attribuutit ja metodit:

Attribuutit:

- Pelikenttä **map**, joka saadaan ladatusta tiedostosta.

- Pelaajahahmo **player**

Metodit:

- Main_loop: metodi joka timerin avulla (QTimer) suorittaa seuraavat asiat:

    - lukee käyttäjältä syötettä

    - päivittää pelitilanteen

    - päivittää graafisen käyttöliittymän pelitilanteen mukaisesti

Map: Pelikenttä, joka alustetaan ladatusta tiedostosta. Tämä toteutetaan kaksiulotteisena taulukkona. Pystyy esim. tarkistamaan mikäli tietty kentän ruutu on tyhjä tai ei metodilla **is_empty**.

Player: Pelaajahahmo, joka pystyy liikkumaan ja hyppäämään metodeilla **move_left**, **move_right** ja **jump**.

Käyttötapauskuvaus:

Käyttäjä käynnistää ohjelman ajamalla main-tiedoston, joka alustaa käyttöliittymän QApplicationin ja **GUI:n** avulla. Kenttä ladataan valitsemalla valikosta "load map"-nappi josta avautuu tiedostodialogi ja sieltä valitsemalla validi kenttätiedosto. Kenttä **Map** alustetaan tämän perusteella ja lisätään peliin eli **Game:iin**. Kun kenttä on onnistuneesti ladattu käyttäjä valitsee valikosta "play"-napin jolloin pääikkuna **GUI** vaihtaa näkymää peliin. Peli alkaa pyöriä **Game**-oliossa ja pääsilmukka rupeaa päivittämään pelitilannetta sekä graafista käyttöliittymää **Gamewidget:iä**. Käyttäjä ohjaa pelaajahahmon kentän läpi loppuun saakka (

näppäimistön käyttö huomataan pääsilmukassa ja näppäinpainallukset johtavat **Player**-luokan liikemetodeihin kutsuun) ja käyttöliittymä **GUI** palaa jälleen valikkonäkymään. Tämän jälkeen ohjelma suljetaan painamalla valikon nappia "Exit", joka on kytketty käyttöliittymän ikkunan sulkemiskomentoon.

Algoritmit:

Ohjelmassa ei varsinaisesti käytetä mitään monimutkaisempia algoritmeja. Pelaajahahmon liikkuminen tapahtuu lineaarisesti eli kun **A**- tai **D**-nappi pysyy alas painettuna, hahmo liikkuu tasaisesti vasemmalle tai oikealle x-koordinaatteja muuttamalla. Hyppy käynnistää tapahtumaketjun jossa hahmo liikkuu määritellyllä tavalla y-koordinaatteja pitkin, kunnes kohtaa pudotessaan esteen ja y-akselin mukainen liike pysähtyy.

Törmäyksentunnistus toteutetaan **Map**-olion is_empty-metodilla, jossa tarkistetaan yksittäisen ruudun tilanne kentän taulukkorakenteesta.

Tietorakenteet:

Ohjelman keskeisin tietorakenne on pelikenttä, joka ladataan kuvatiedostosta. Kuva luetaan pikseli pikseliltä ja lisätään kaksiulotteiseen taulukkorakenteeseen. Tämä on järkevin ratkaisu sillä tietorakennetta täytyy pystyä indeksoimaan. Kun kenttä on luettu, sitä ei tarvitse enää muokata. Tästä kaksiulotteisesta taulukosta tulee kenttäpohja, jossa yksi kuvatiedoston pikseli vastaa yhtä ruutua joka voi sisältää esteen tai pelaajan.

Aikataulu:

Projektin ajankäyttöä on erittäin vaikea arvioida ison projektin tekemisen kokemuksen puutteessa. Seuraavassa kuitenkin etenemissuunnitelma ja arvioitu ajankäyttö:

1. Pääikkuna ja valikko (Lähinnä Qt-totuttelua ja tutkimista. 3h)
2. Itse peliluokka **Game** ja perustoiminnot (10h)
3. Kuvatiedoston lataaminen tietorakenteeseen **Map** (6h)
4. Pelin graafinen käyttöliittymä eli **Gamewidget**, kentän piirtäminen (10h)
5. Pelaajan lisääminen ja liike (6h)
6. Törmäyksentunnistus (3h)
7. Graafisen puolen hiominen: taustakuva, jne. (6h)

Yksikkötestaussuunnitelma:

Peliä on vaikea testata ennen kuin pelin peruskomponentit on toteutettu. Tason lataamista alkuvaiheessa (ilman graafista käyttöliittymää) voi kuitenkin testata tarkastelemalla lopputuloksena syntyvän **Map**-taulukon indeksejä. Tämän lisäksi täytynee testata että ohjelma ei kaadu erikoisten tiedostojen tapauksissa, esim. rikkoutunut tiedosto, väärä tiedostoformaatti, puuttuva hahmon aloitusruutu ja puuttuva maalialue/ruutu. Kun graafinen käyttöliittymä on toteutettu voi kentän lataamista vielä testata manuaalisesti kokeilemalla sen pelaamista. Tällöin tulisi kokeilla kaikkia mahdollisia ruutuja ja aluetyyppejä ja varmistaa että ne toimivat kuin on tarkoitus. Samalla tulee testattua itse piirtäminen ja se että esineet sekä hahmo tulevat oikeille paikoilleen.

Törmäyksentunnistusta ei voi testata muulla tavoin kuin kokeilemalla itse pelissä, eri esteitä kohti. Tähän ongelmaan liittyen pelaajahahmon täytyy myös tunnistaa kentän reunat ja törmätä niihin, ettei pääse ylittämään kentän rajoja ja kaatamaan ohjelmaa.

Kirjallisuusviitteet ja linkit:

http://gameprogrammingpatterns.com/game-loop.html

http://gameprogrammingpatterns.com/component.html

http://doc.qt.io/