29.4.2018

Henrik Granö, 528 634

Bioinformaatioteknologia, sähkötekniikan kandidaattiohjelma

CS-A1121 Ohjelmoinnin peruskurssi Y2

Y2-tasohyppelyprojektin dokumentaatio

1. Yleiskuvaus

Ohjelma on tasohyppelypeli, jossa pelaaja ohjaa hahmoa kentän läpi hyppimällä erilaisten tasojen ja esteiden päälle. Pelin keskeisiä ominaisuuksia ovat graafinen käyttöliittymä, törmäyksen tunnistus sekä kentän helppo luominen kuvankäsittelyohjelmalla.

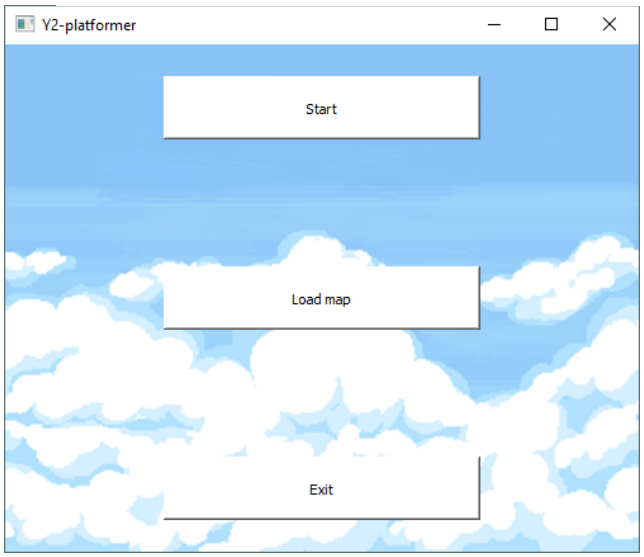
Ohjelma poikkeaa alkuperäisestä suunnitelmasta lähinnä rakenteeltaan. Alkuperäisen suunnitelman luokka Gamewidget, jonka oli tarkoitus hoitaa pelin grafiikat sekä näppäimistösyötteet, on siirretty kokonaisuudessan pääikkunan MainGUI:n sisälle, ja Game-luokan metodi mainloop() eli keskeinen peliä pyörittävä silmukka on myös siirretty MainGUI:n metodiksi.

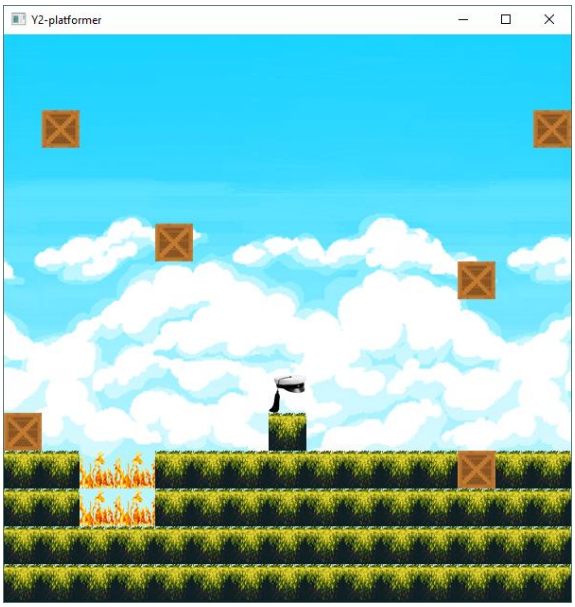
Työ on toteutettu vaikeustaso 3 mielessä.

2. Käyttöohje

Ohjelmaa käynnistettäessä main.py-tiedostolla avautuu pääikkuna, josta voi klikkaamalla valita kolme vaihtoehtoa: pelin aloittaminen "Start", kentän lataus "Load map" ja pelin lopettaminen "Exit". Peliä ei voi aloittaa ennen kentän valitsemista. "Load map"-nappia painettaessa avautuu tiedostoselain, jonka avulla \maps-kansiosta voi etsiä itselleen mieluisan kentän 24-bittisessä BMP-formaatissa. Ohjelma ilmoittaa mikäli tiedoston luku onnistui vai ei. Tämän jälkeen pelin voi käynnistää painamalla "Start"-nappia pääikkunasta. Pelihahmoa ohjataan A- ja D-näppäimillä seuraavasti: A-näppäin siirtää hahmoa vasemmalle, D oikealle. Välilyönti-näppäintä käyttämällä hahmo hyppää, ja Q-näppäimellä pelin voi keskeyttää jolloin näkymä vaihtuu jälleen pääikkunaan (peliä voi jatkaa painamalla "Start":ia uudestaan tai lataamalla uuden kentän). Peli loppuu kun hahmo joko osuu vaaralliseen ruutuun (oletusarvoisesti ruutuun jonka tunnistaa liekeistä) ja pelin voi voittaa ohjaamalla hahmo maaliruutuihin saakka (oletusarvoisesti ruutuihin jotka tunnistaa mustavalkoisesta kuviosta). Pelin lopetettua ohjelma palaa pääikkunaan, jonka voi halutessa sulkea "Exit"-napilla.

Oman minkä tahansa näköisen kentän voi luoda vaikkapa perinteisellä Microsoftin MS Paint-kuvankäsittelyohjelmalla, katso kohta 7: Tiedostot tarkempia ohjeita varten. Muutamia asetuksia voi myös muuttaa settings.txt-tiedostoa muokkaamalla. Katso kohta 7: Tiedostot tarkempia ohjeita varten. Alla vasemmalla kuva pelin päävalikosta, oikealla ruudunkaappaus pelistä.





3. **Ulkoiset kirjastot**

Ohjelma on kirjoitettu ja testattu Pythonin versiolla 3.5.

Projektissa on käytetty PyQt-kirjastoa, ja sitä on testattu versioilla 5.4.1 ja 5.10.0. PyQt mahdollistaa Qt-grafiikkakirjaston käyttöä Python-kielellä, ja sitä on käytetty nimenomaan grafiikoiden tuottamiseen sekä ajastimen (QtCore.QTimer), näppäimistösyötteen (QtWidgets.QMainWindow.KeyPressEvent()), ja äänentoiston (QtMultimedia.QSound) käyttöön.

Tämän lisäksi on käytetty pythonin omia sisäärakennettuja kirjastoja seuraavasti:

os - tiedostopolkujen hakua varten

sys - Qt:n alustamista varten

enum - pikselityyppienumeraation luomiseen

4. **Ohjelman rakenne**

Ohjelman sydämenä toimii GUI.py-tiedostoon toteutettu luokka MainGUI. Tämä luokka hallinnoi kaikki graafiseen käyttöliittymään liittyvät tehtävät. Se alustaa ja pyörittää pääikkunaa, joka perii Qt:stä QWidget()-luokan, joka vuorossaan saa taustakuvan QLabel-luokan kautta sekä pääikkunapainikkeet QPushButton-luokista. Pelikentän latausvalikko, jota kutsuu mainGUI:n metodi mapdialog(), on toteutettu QFileDialog-luokalla. Pelin käynnistyttyä mainGUI hallinnoi ruudun grafiikoiden piirtoa metodilla paintEvent() , näppäimistösyötettä metodilla keyPressEvent() ja pelin main-loop metodia loop(). Kuten ylemmällä mainittu, main-looppi on siirretty MainGUI-luokkaan näppäimistösyötteen toimivuuden vuoksi. Main-looppi pyörii pääikkunan QTimer-ajastinluokalla, joka lähettää signaalin määrätyn aikavälein, ja pyörittää peliä: se tarkistaa pelin ja pelihahmon tilan (onko peli voitettu vai hävitty?), kutsuu hahmon siirtofunktioita ja kutsuu pääikkunan grafiikanpiirtometodia. loop()-metodi pyöritetään QTimer:illa, koska sen suoritusnopeutta on vaivatonta muokata, toisin kuin for- tai while-silmukan tilanteessa. MainGUI myös lukee settings.txt-tiedostoa metodilla readsettings().

Toiseksi isoin luokka on Game-luokka, joka sisältää pelin törmäyksentunnistusmetodit collision(), ycollision, gravity() sekä pelikentän latausfunktion load_map(). MainGUI on riippuvainen tästä luokasta käynnistääkseen pelin, ja sillä on viittaus tähän voidakseen piirtää pelikentän. Game-luokasta on viittaus Player-olioon, joka kuvaa pelaajahahmoa. Tässä luokassa sijaitsee pelaajan liikealgoritmit metodeissa move(), jump(), left() ja right().

Pienemmät luokat ovat Map joka sisältää kaksiulotteisen kenttätaulukon joka kuvaa pelikenttää, PixelType joka toimii enumeraatioluokkana, sekä Coordinates jota käytetään ympäri ohjelmaa kuvaamaan x- ja y-suuntaista liikettä ja sijaintia.

5. **Algoritmit**

Matemaattista laskentaa tarvitaan ohjelman kahdessa osakokonaisuudessa: pelaajaliikkeessä sekä pelikentän indeksoinnissa. Nämä perustuvat hyvin yksinkertaiseen periaatteeseen. Pelaaja aloittaa kentän aloituskohdasta, ja liikettä seurataan offset-muuttujalla joka mittaa pelaajan kuljettua matkaa pikseleissä y- ja x-suuntaisesti. Graafinen käyttöliittymä ja törmäyksentunnistus mittaaavat siis pelaajan sijaintia seuraavalla kaavalla:

location = (player.coordinates.x + player.offset.x, player.coordinates.y + player.offset.y)

Jossa coordinates.x ja coordinates.y muodostavat pelaajan muuttumattoman aloituskohdan, offset.x ja offset.y taas pelaajan liikkumaa matkaa. Pelaaja liikkuu sivuttaissuunnassa noudattaen kaavaa:

player.xspeed += player.xacceleration

player.offset.x += player.xspeed

eli

nopeus = nopeus + kiinteä kiihtyvyysvakio

sijainti = sijainti + nopeus

Eli kun pelaajahahmoa käsketään liikkumaan, ensin nopeusmuuttujaan lisätään kiinteän kiihdytysvakion verran nopeutta, sen jälkeen lisätään sijaintimuuttujaan dynaamisen muutetun nopeusmuuttujan verran siirrettyä matkaa. Hyppääminen ja putoaminen tapahtuu samalla periaatteella:

player.yspeed -= player.yacceleration

player.offset.y += self.player.yspeed

Graafisesta käyttöliittymästä on mainitsemisen arvoista että se poimii pelaajan ympäriltä aina lähimmät ruudut ja piirtää ainoastaan ne eikä koko kenttää, iteroimalla kenttää pelaajan sijainnin ympärillä.

6. **Tietorakenteet**

Keskeisin tietorakenne on pelikenttä, joka on toteutettu yksinkertaisena kaksiulotteisena (muuttuvatilainen) Pythonin listana, joka täytetään pikseli/ruutu/indeksi kerrallaan kenttää luettaessa. Muita luontaisia vaihtoehtoja ei ollut. Tämän lisäksi Coordinates-luokkaa voi luonnehtia yksinkertaiseksi tietorakenteeksi, joka sisältää x- ja y-arvon.

7. **Tiedostot**

Ohjelma käsittelee kahta tiedostotyyppiä: tekstitiedostoa .txt-formaatissa alustaamaan ohjelman virkistystaajuutta ja grafiikkopuolen kuvia, sekä 24-bittistä BMP-kuvatiedostoa pelikentän lataamiseen.

Muuttamalla settings.txt-tiedoston framerate-arvoa voi määrittää pelin virkistystaajuuden eli käytännössä nopeuden, ja muokkaamalla muita riviä voi vaihtaa ladattavien kuvatiedostojen nimiä, jolloin voi muuttaa pelin ulkonäköä omilla kuvilla jotka ovat kooltaan 40x40 pikseliä. Ohjelma ei lue #-merkillä alkavia riviä, sillä ne ovat tarkoitettu kommenteiksi. Jokainen muutettava ominaisuus on tiedostossa esitetty omalla rivillään.

Pelikenttä ladataan kuvatiedostosta omalla algoritmilla ilman ulkoista kirjastoa. Oman minkä tahansa näköisen kentän voi luoda vaikkapa perinteisellä Microsoftin MS Paint-kuvankäsittelyohjelmalla, kunhan noudattaa seuraavia sääntöjä:

- Kentässä voi olla vain yksi pelaaja, jonka aloituskohta merkitään sinisellä pikselillä (varmista RGB-arvot: R: 0, G: 0, B: 255)

- Punainen pikseli tarkoittaa estettä (varmista RGB-arvot: R: 255, G: 0, B: 0)

- Vihreä pikseli tarkoittaa maata/ruohoa (varmista RGB-arvot: R: 0, G: 255, B: 0)

- Valkoinen pikseli tarkoittaa tyhjää tilaa, jossa pelaajahahmo voi liikkua (varmista RGB-arvot: R: 255, G: 255, B: 255)

- Musta pikseli tarkoittaa maalialuetta. Pelaajan ollessa tämmöisessä ruudussa peli on voitettu (varmista RGB-arvot: R: 0, G: 0, B: 0)

- Keltainen pikseli tarkoittaa vaarallista ruutua. Pelaajan koskiessa tämmöistä ruutua peli on hävitty (varmista RGB-arvot: R: 255, G: 255, B: 0)

- Kenttä täytyy tallentaa 24-bittisenä BMP-tiedostona

Ohjelma saattaa käyttäytyä viallisesti joissain tapauksissa, katso kohta 9: Ohjelman tunnetut puuttet ja viat

8. **Testaus**

Ohjelmaa testattiin hyvin säännöllisesti kehityksen aikana, ja juuri kuten alunperin oletettiinkin. Jokaisen toiminnon valmistuttua suoritettiin manuaalinen testi toimivuuden kannalta. Tasolataamisen valmistuttua testattiin erikokoisten tasojen lataaminen ja rikkinäisen kuvatiedoston lataaminen, jotka käytännössä ovat ainoat mahdolliset muuttujat tässä kokonaisuudessa. Törmäyksentunnistus testattiin manuaalisesti ja viilattiin kuntoon sen perusteella, samoin pelin loppuminen. Yksikkötestaus ei ole mahdollista tämäntyyppiselle ohjelmalla joka riippuu täysin käyttäjän näppäimistösyötteestä.

9. **Ohjelman tunnetut puutteet ja viat**

Pelin offset-periaatteeseen nojautuva algoritmi ei ole ideaali, sillä indeksoiminen ja liikkeen seuraaminen koordinaatistolla ja erillisillä sijaintimuuttujilla vaatii sekä ylimääräistä laskentatehoa että vaivannäköä ohjelmoijalta algoritmien luomiseen ja viimeistelyyn. Parempaa olisi jos pystyisi indeksoimaan pelkästään yhdellä koordinaatistolla, joka toimisi pikselitasolla. Ohjelman suunnitelma kuitenkin nojautui alkuperäiseen ruutujärjestelmään ("tile-based") ja tämän liittäminen graafiseen käyttöliittymään joka vaati pikselitason indeksointia osoittautui hankalaksi ongelmaksi, jonka johdosta offset-periaate otettiin käyttöön.

Sama seikka johtaa bugiin jossa peligrafiikan piirtäminen ei toimi saumattomasti kun pelaaja on siirtynyt lähtökohdasta vasemmalle. Tällöin kaikki esineet piirtyvät yhden ruudun verran liian pitkälle vasemmalle, jolloin törmäyksentunnistus ja näyttö eivät täsmää. Tämän lisäksi kaikki grafiikat nytkähtävät tasaisin välein kun liikutaan x-suunnassa.

Viimeisen testauksen aikana ilmaantui uusi ikävä virhe: mikäli pelaaja siirtyy alkupisteestä 3 tai enemmän ruutua alemmas, ohjelma kaatuu. Tämän voi estää suunnittelemalla semmoinen kenttä jossa pelaaja ei pääse liikkumaan alaspäin kuin kahden ruudun verran (yleisesti tasohyppelypeleissä ei muutenkaan siirrytä kovin usein taaksepäin/alaspäin vaan eteenpäin ja ylöspäin). Vika on kriittinen mutta hyvin tiukan aikataulun ja ajanpuutteen vuoksi se täytyy valitettavasti jättää viimeiseen palautukseen. Vian voisi hyvin todennäköisesti ratkaista ottamalla huomioon negatiiviset offset-arvot toisella tavalla grafiikanpiirrossa ja törmäyksentunnistuksessa. Myös yllä mainittu piirtobugi voidaan ratkaista tällä tavalla.

Törmäyksentunnistus ei ole täydellinen, jonka johdosta pelaajahahmo voi joutua osittain esteen sisään liikkuessaan sekä x- että y-suunnassa samanaikaisesti. Tämän voisi välttää lisäämällä metodi / muokkaamalla olemassa olevia törmäyksentunnistusmetodeja siten, että ne mittaavat myös x- ja y-nopeuksien yhteisvektoria ja sen päätymispistettä. Tätä ei välttämättä kuitenkaan tarvitse laskea viaksi vaan ominaisuudeksi, sillä se mahdollistaa esteiden yli kiipeämisen näppärästi.

10. **3 parasta ja 3 heikointa kohtaa**

Parasta:

1. Helppo kentän luominen. Oman kentän voi luoda näppärästi melkein millä tahansa kuvankäsittelyohjelmalla.

2. Suhteellisen sulava liike normaalitilanteissa sekä helppo hahmon liikuttaminen joka tekee pelaamisesta mieluisaa, varsinkin verrattuna aiemman prototyyppiin.

3. Manuaalinen kentän lataus ilman ulkoista kirjastoa joka vaati hiukan perehtymistä BMP-tiedoston rakenteeseen.

Heikkoa:

1. Offset-periaatteen käyttö liikkeen seuraamiseksi ei ole optimaalinen algoritmi ja on turhan monimutkainen.

2. Offset-periaatteen seuraukset eli negatiivisten arvojen vaikutus grafiikan piirtämiseen ja jopa ohjelman kaatumiseen erikoistilanteessa.

3. Offset-periaatteen johdosta koodia on hiukan vaikea lukea ja ymmärtää.

11. Poikkeamat suunnitelmasta

Ohjelma poikkeaa alkuperäisestä suunnitelmasta lähinnä rakenteeltaan. Alkuperäisen suunnitelman luokka Gamewidget, jonka oli tarkoitus hoitaa pelin grafiikat sekä näppäimistösyötteet, on siirretty kokonaisuudessan pääikkunan MainGUI:n sisälle Qt-kirjastoon liittyvien teknisten vaikeuksien vuoksi. Game-luokan metodi mainloop() eli keskeinen peliä pyörittävä silmukka oli myös siirrettävä MainGUI:n sisälle jotta saisi näppäimistösyötteen toimimaan. Pelaajaliike oli aluksi tarkoitus toteuttaa staattisena mutta se on lopulta kuitenkin toteutettu nopeus- ja kiihtyvyysarvoilla napakan ja sulavan liikkeen aikaansaamiseksi.

Ajankäyttöarvio osui hyvin lähelle totuutta, kannatti siis arvioida eri vaiheiden vaativuustasot yläkanttiin. Kuvatiedoston lataaminen tietorakenteeseen ei vaatinut suunniteltua 6 tuntia vaan lähemmäs 2 tuntia, ja ylijääneet 4 tuntia kuluikin helposti törmäyksentunnistukseen. Kaiken kaikkiaan aikaa kului ohjelmaan karkeasti 40 h.

Toteutusjärjestys oli sama kuin suunnitelmassakin

12. Toteutunut työjärjestys ja aikataulu

Toteutusjärjestys yleisellä tasolla:

Ensin pääikkuna, sen jälkeen kentän lataus, peliluokka Game, graafinen käyttöliittymä, liike, törmäyksentunnistus, ja lopulta hiominen. Moduuli, joka toimii perustana toiselle, täytyy luonnolisesti toteuttaa järjestyksessä ennen toista.

Pääikkuna, kentän lataus ja Game-luokka toteutettiin 6-15.3.2018, ja loput aikavälillä 16-29.4.2018. Tämä poikkeaa suunnitelmasta siten, että ajan puute ja kiireen runsaus maalis- ja huhtikuussa yllättivät pahemman kerran ja siirsivät projektin toteutusta viimeisille viikoille.

13. Arvio lopputuloksesta

Ottaen huomioon että kyseessä on ensimmäinen itse luotu peli, on lopputulokseen syytä olla tyytyväinen. Projektia oli mielekästä tehdä ja siihen on käytetty paljon aikaa. Ohjelma on luotu vaatimukset mielessä eikä siinä siinä mielessä ole varsinaisia puutteita, muutamia erityisbugeja lukuun ottamatta. Yleistasolla se on toimiva ja siisti kokonaisuus, mutta konepellin alta paljastuu kankeahko mekanismi joka vaikuttaa pelin stabiliteettiin: mikäli pelaaja haluaa liikkua paljon alas ja taaksepäin, ei ohjelma välttämättä sitä kestä kovin kauan liikealgoritmin johdosta. Kuten aiemmin on kuitenkin mainittu, eivät useimmat muutkaan tasohyppelypelit tue tämmöistä liikettä. Ohjelmaa voisi jatkossa parantaa käsittelemällä negatiiviset sijaintiarvot erikseen, mahdollistaen monipuolisemman käyttökokemuksen. Muuten ratkaisumenetelmät ja luokkajako toimivat hyvin. Ohjelmaan voi suhteellisen helposti tehdä muutoksia kaikkeen paitsi itse liike/koordinaatistomekanismiin: käyttöliittymämuutoksia, pelaajan liikealgoritmeja, muiden kenttätiedostotyyppien tukeminen yms. ei vaadi tolkuttomasti vaivaa. Lisäksi omien grafiikoiden käyttäminen on tehty helpoksi settings.txt-asetustiedoston avulla.

14. Viitteet

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

http://gameprogrammingpatterns.com/component.html

https://en.wikipedia.org/wiki/BMP_file_format

http://doc.qt.io/

kuvat ja äänet:

https://opengameart.org/content/sky-background

http://pixelartmaker.com/art/ac2f6f276831990

https://ansimuz.itch.io/magic-cliffs-environment

http://soundbible.com/tags-jump.html

15. Liitteet

Y2-platformer lähdekoodi: https://version.aalto.fi/gitlab/granoh1/Y2-platformer