5.5.2021

CS-A1121
Ohjelmoinnin peruskurssi Y2

Peli: Tasohyppely

Daniel Nikkari, 653088

Informaatioteknologia

Peli: Tasohyppely

Daniel Nikkari

Yleiskuvaus

Tämän projektin tavoitteena oli luoda pythonilla yksinkertainen tasohyppelypeli, jossa pelaaja ohjaa hahmoa kentän läpi hyppimällä erilaisten esteiden yli, kuten esimerkiksi pelissä Donkey Kong. GUI:n (Graphical user interface) kehittämiseen oli käytössä PyQt5 kirjasto. Ohjelmaan toteutin yksinkertaisen tasohyppelypelin, jossa pelaaja voi nimetä hahmonsa, ja aloittaa pelin painamalla START-nappia. Peli siten luo kartan, jossa on pelaajan hahmon lisäksi NPC (Non-player character) -hahmoja. Pelaajan pitää hyppimällä, tai ampumalla tulipalloja, päästä näiden NPC-hahmojen ohitse koskettamatta niitä. Jos pelaaja koskettaa NPC-hahmoa pelaaja häviää pelin, jolloin peli keskeytyy, ja näytöllä lukee GAME OVER. Tällä näytöllä pelaaja voi päättää joko mennä takaisin menuun tai yrittää uudelleen eli RESTART. Mikäli pelaaja onnistuu päästä kentän loppuun, voi pelaaja aukaista oven rakennuksesta, jolloin pelaaja voittaa pelin, ja pelaajan aika tallentuu tiedostoon, sekä mikäli se on paras aika, niin se tallentuu myös ”Top scorena” menuun näkyville.

Pelin grafiikkoina käytin https://opengameart.org -sivustolta ladattua tasohyppelygrafiikkapakettia, jonka loi Kenney Vleugels (www.kenney.nl). Grafiikkojen lisenssinä on CC0 (http://creativecommons.org/publicdomain/zero/1.0/).

Työni lopullinen toteutus oli keskivaikean ja vaativan tason välistä.

Käyttöohje

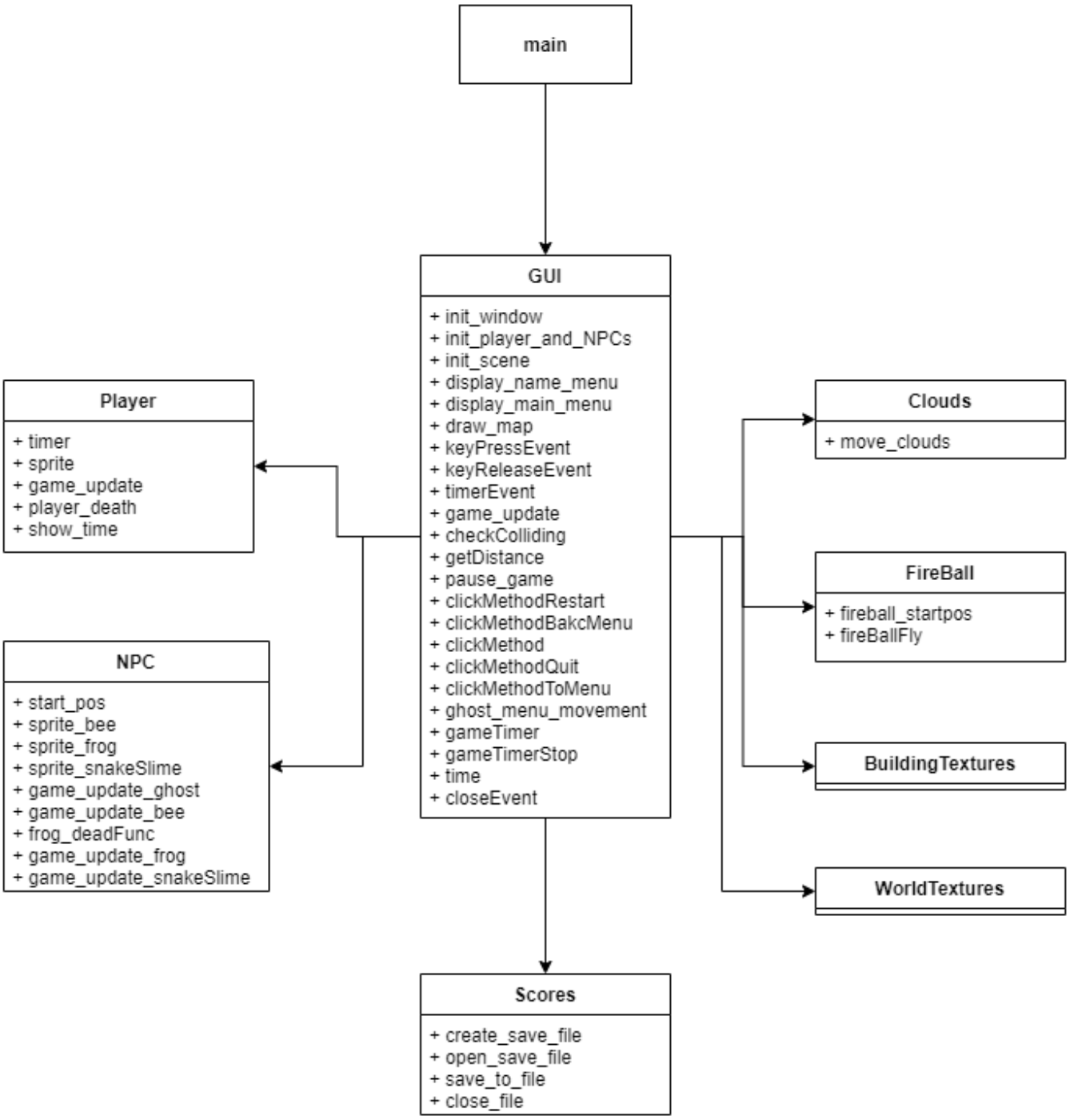
Ohjelma on ohjelmoitu python 3:lla. Ohjelma käynnistetään ajamalla **main.py** tiedosto, jolloin ohjelma aukaisee ikkunan, jossa näkyvät pelin grafiikat. Aluksi pelaaja voi kirjoittaa hahmonsa nimen toiseen pienempään ikkunaan, ja sulkea ikkunan painamalla yläoikealla olevaa X. Tämän jälkeen pelaaja voi painaa ok, jolloin peli menee menuun. Menusta käsin pelaaja voi joko poistua pelistä painamalla QUIT tai aloittaa pelin painamalla START.

Pelaaja voi käyttää hahmonsa liikuttamiseen näppäimiä W, A, D ja S. W:llä pelaaja voi hypätä, A ja D liikuttaa pelaajaa vasemmalle tai oikealle, S:llä pelaaja voi mennä kyykkyyn. Näiden lisäksi on käytössä E, jolla voi ampua tulipalloja, sekä F, jolla voi nostaa asioita maasta tai aukaista oven. Jos pelaaja häviää tai voittaa, pelaaja voi valita joko mennä takaisin menuun tai aloittaa alusta.

Ulkoiset kirjastot

Ulkoisena kirjastona oli käytössä PyQt5-kirjasto.

Ohjelman rakenne



Main, toisin kuin muut .py tiedostot projektissa, sisältää olion sijaan pelkästään funktion, josta ohjelma käynnistetään.

Oliot	Selostus
GUI	GUI luo ja aukaisee ikkunan pelille sekä luo ”scenen”, johon peli piirtää kaikki objektit kuten pelaajan hahmon, jne. Samoin GUI kutsuu eri olioita oikeina hetkinä ja luo kaikki tarvittavat objektit peliin. GUI samoin tahdittaa pelin ajan kulun ajastimella, jossa on 16ms per kuva. Main-funktio kutsuu GUI:ta, jolloin GUI hoitaa loput ohjelman ajamisen suhteen, sekä kutsuu kaikkia muita olioita tarpeen mukaan. GUI:ssa myös tapahtuu törmäyksen tunnistus.
Player	Player-olio luo pelaajan graafisen hahmon, sekä ottaa vastaan signaaleja GUI:lta, kuten keyPressEventtejä, jolloin Player-olio reagoi tähän signaalin perusteella. Player-olio sisältää myös funktiot pelaajan animaatioille.
NPC	NPC-olio luo NPC-hahmoja sen perusteella, millä nimellä GUI kutsuu NPC:tä. Esimerkiksi jos GUI kutsuu ghost nimellä, niin NPC-olio luo ghost hahmon. Samoin GUI kutsuu ajoittain päivityksenä NPC:n eri funktioita. NPC sisältää funktiot NPC:n liikkumiselle ja animaatioille.
Scores	Scores-olio luo tekstitiedoston, johon pelaajan ajat tallennetaan. Scores nimeää tiedostot käyttämällä sen hetkistä päivämäärää ja aikaa.
Clouds	Clouds-olio luo pilviobjekteja, joita se liikuttaa move_clouds funktiolla.
FireBall	FireBall-olio luo tulipallo-objektin, jota se liikuttaa fireBallFly-funktiolla. Fireball_startpos-funktio varnistaa, että tulipallo lähtee oikeasta kohtaa lentämään.
BuildingTextures	BuildingTextures-olio sisältää kaikki rakennuksen tarvitsemat tekstuurit, joita GUI voi tarpeen mukaan kutsua. BuildingTextures ei sisällä funktioita, vaan se pelkästään luo objektin kutsutulle tekstuurille.
WorldTextures	WorldTextures-olio toimii samalla periaatteella kuin BuildingTextures, ja samoin kuin BuildingTextures, ei sisällä funktioita.

Algoritmit

Ohjelma sisältää hyvin yksinkertaisia algoritmeja. NPC-hahmojen liikkuminen perustuu hahmojen x, y-koordinaattien sijaintiin pikselikartalla scenen sisällä. Keskeiset algoritmit hahmojen liikuttamiseen ovat x + dx ja y + dy, jossa dx ja dy on nopeus, jolla x- tai y-koordinaatti muuttuu, sekä x ja y edustaa tämänhetkistä koordinaattia (objektin sijaintia 2D-pikselitasolla). Pelissä on tämän lisäksi törmäyksentunnistus, jossa on käytössä algoritmi sammakon päälle hyppäämiselle, joka tarkastelee mistä suunnasta objektien törmääminen tapahtuu. Se käyttää kahden objektin y-koordinaattien välistä tarkastelua. Itse törmäyksen tunnistus algoritmit tulevat PyQt5:sta. GUI:ssa on funktio, joka laskee kahden objektin pisteen välisen etäisyyden.

$$d = \sqrt{((x_2 - x_1)^2 + (y_2 - y_1)^2)}$$

Tämä algoritmi jäi käyttämättömäksi, mutta olisin käyttänyt sitä mahdollisesti älykkäämpään NPC:n tekoälyyn, joka olisi yrittänyt kävellä pelaaja kohti.

Pelaajan hyppimisessä on käytössä algoritmina $F = m \times (hyppy\ nopeus)^2$, hypyn aikana hyppy nopeutta pienennetään, kunnes se on alle 0. Siten massan etumerkki vaihdetaan negatiiviseen. Kun hyppy nopeus on $-(hyppy\ nopeus) - 1$, niin if-lause lopettaa hypyn ja palauttaa massan arvon takaisin positiiviseksi sekä hyppy nopeuden takaisin sen alkuarvoon.

Tietorakenteet

Pelin tulosten tallentamiseen käytin CSV-tekstitiedostoa. Pistetauluun tallentamisessa käytössä on lista. Tekstuurien noutamiseen tein tietorakenteen, joka sisältää sanakirjan, jossa tekstuuri on yhdistetty avaimeen, jolla sen helposti löytää isosta määrästä tekstuureita. Tässä tein virheen siinnä, että laitoin itse sanakirjan sisälle PyQt5.QtGui.QPixmap. Tämä hidastaa ohjelman toimintaa, sillä kun se etsii jonkin tietyn tekstuurin sanakirjasta, niin se lataa jokaisen muun tekstuurin pixmapin turhaan samalla. Tämän olisi voinut korjata ottamalla QPixmap sanakirjan ulkopuolelle, ja tehdä QPixmap vasta kun sanakirja löytää tekstuurin sijainnin tietorakenteessa. Olisin korjannut kyseisen ongelman, mutta se jäi tekemättä aikataulu syistä, samoin tehtävänanto ei vaatinut tehokasta tietorakennetta, niin oletan ettei se myöskään vaikuta projektin kokonaisuuden arvosteluun.

Esimerkki tekstuurin hakemisesta:

self.grassMid = WorldTextures(”grassMidTex”)	Luodaan WorldTextures olio ja haetaan sanakirjasta tekstuuri avaimella grassMidTex.
self.worldTextures = {…	Etsitään tekstuuri sanakirjasta.
”grassMidTex”: QPixmap(”Textures/	
Ground_Textures/	
grassMid.png”),	
…}	
wanted_texture = self.worldTextures[texture]	Otetaan tekstuuri talteen muuttujaan.
self.setPixmap(wanted_texture)	Asetetaan tekstuuri oliolle.

Tiedostot

Ohjelma käsittelee tekstuureissa PNG-tiedostoja, ja lataa nämä PyQt5 käyttämällä. Ohjelman scores.py luo tekstitiedoston, ja kirjoittaa siihen, sekä sulkee sen lopuksi. Pelissä on käytössä MP3 muotoisia audio tiedostoja pelin musiikille ja ääniefekteille.

Testaus

Ohjelman testaaminen perustui pääosin tutkivaan testaamiseen (exploratory testing). Ajoin ohjelmaa muutosten jälkeen, ja kokeilin koodin toimivuutta. Ongelman löytämiseen käytin myös print-komentoja, joissa käytin aakkos- järjestelmää ”A,B,C..”, jolla yritin löytää ongelman kohdan. Testauksen suunnittelu oli suhteellisen epätarkka, joten sitä oli vaikea käyttää käytännössä. Yritin hyödyntää ohjelmaani yksikkötestausta, mutta en onnistunut saada sitä toimimaan siten, että se olisi parempi kuin tutkiva testaaminen. Samoin ohjelmani palauttaa arvoja harvoista funktioista, sekä olioiden luomisessa yksikkötestiä varten tuli ongelmia, joita en osannut ratkaista. Tulin lopputulokseen, että tutkiva testaaminen on tarpeeksi ohjelman testaamiseen, mutta yksikkötestin käyttö olisi hyvä osata jatkossa.

Ohjelman tunnetut puutteet ja viat

Ohjelmassa on aikaisemmin mainitsemani ongelma sanakirjoissa, jossa kun etsii tiettyä tekstuuria avaimella, sanakirja lataa jokaisen muunkin sanakirjan sisällä olevan tekstuurin QPixmap:llä. Tämän korjaisin ottamalla QPixmap:n ulos sanakirjoista, jolloin tekstuurien löytäminen ja lataaminen olisi nopeampaa. Jälkeenpäin olisin ohjelmoinut kartan rakennuksen tiedostosta käsin toimivaksi, jolloin olisi helpompi luoda uusia karttoja. Samoin loisin tasoja, joiden päälle voi hyppiä, jolloin peliin tulisi hieman enemmän syvyyttä. Lisäksi automatisoitu testaaminen yksikkötestin muodossa puuttuu.

3 parasta ja 3 heikointa kohtaa

Yhtenä erityisen hyvänä osana ohjelmaani pidän pelin tekstuurien ja ääniefektien käyttöä ja animaatioita. Peli näyttää itsessään tosi hyvältä, ja sisältää hauskoja yksityiskohtia. Yksi hyvä, mutta samalla heikko, kohta ohjelmassani ovat sanakirjat. Jos ne korjattaisiin, niin ne toimisivat hyvin tekstuurien hakuun.

Poikkeamat suunnitelmasta

Itse pelin luonne ja olennainen idea on kuin suunniteltu, mutta lisäsin pieniä muutoksia projektin varrella. Vaihdoin hypyn space-näppäimestä W-näppäimeen, sillä olen tottunut pelaamaan FPS pelejä, jolloin 2D-tasolla W-näppäin hypylle tuntui itselleni luontaisemmalta. Tein samoin pieniä lisäyksiä ominaisuuksiin, joita en ollut ennen projektia suunnitellut implikoitavaksi kuten tulipallon. Samoin projektin rakenne muuttui hieman, muttei paljoa sekä UML-mallin olennainen muoto pysyi jotakuinkin samana suunnitelman kanssa.

Suunnittelemani ajankäyttö oli hyvin erilainen oikeasta ajankäytöstäni. Tunti määrältä se vastasi odotuksia, mutta päivämäärissä ja työnjärjestyksessä oli poikkeamia. Tein käytännössä suurimman osan projektia noin neljässä päivässä maaliskuun loppupuolella. Tämän jälkeen hioin koodia hieman silloin tällöin ja lisäsin tarvittavia muutoksia.

Toteutunut työjärjestys ja aikataulu

19.3.2021-22.3.2021	Main-funktio, GUI, Player, WorldTextures ja NPC alustavasti implementoitu. Prototyyppi kentästä luotu, pelaajan hahmo ja sen liikuttaminen implementoitu. Ensimmäiset NPC hahmot lisätty. Ongelmia pelin latausajoissa.
26.3.2021-5.4.2021	Pelin latausaika ongelma selvitetty, checkpoint pidetty. Alustava README kirjoitettu. Perus pisteidentallennus implementoitu sekä lisätty pelin ajastin ja voitto/häviämisehdot. Pelaajan nimeämis- ominaisuus lisätty. Koodia hieman hiottu paremmaksi.
6.4.2021-6.5.2021	Pieniä korjauksia ohjelmaan ja pientä koodin siistimistä sekä kommentointia. Dokumentti kirjoitettu. Ohjelma demo- sekä palautusvalmis.

Suunnitelmasta poikettiin pääosin kaikessa aikataulutuksessa paitsi checkpointissa ja projektin palautuspäivämäärässä.

Arvio lopputuloksesta

Ohjelman laatu kokonaisuudessaan on suhteessa hyvä ottaen huomioon aikaisempi kokemattomuus projektin aihealueesta. Koodi on kommentoitu sekä jäsennelty tarpeeksi siististi, että siitä saa selvää. Samoin muuttujien nimeäminen on mielestäni tarpeeksi selkeä. Itse ohjelman tietorakenteiden, laajennettavuuden ja koodin laadun olisi voinut tehdä paremmin. Sanakirjoissa oli ongelma QPixmapin kanssa, jota en osannut odottaa, ja jonka olisi voinut viellä korjata ottamalla sanakirjoihin pelkästään tiedostosijainit, ja tehdä QPixmap sanakirjan ulkopuolella. Ohjelman laajennettavuus ei ole myöskään paras, sillä ohjelma ei käytä esimerkiksi karttojen luomiseen tiedostoja, vaan ”hard coded” eli kovakoodattua karttaa suoraan ohjelmassa. Tämän kyllä voisi korjata lisäämällä ominaisuuksia, mutta se vaatisi huomattavasti lisää työtä. Ohjelman rakenne soveltuu silti laajennettavaksi, mutta se vaatii huomattavasti työtä.

Luokkajako ohjelmassa on tyydyttävä, siinnä olisi voinut tehdä pieniä muutoksia esimerkiksi WorldTextures suhteen, ja eritellä tekstuureja hieman enemmän omiin luokkiinsa. Tulevaisuudessa olisin parantanut ohjelmaa korjaamalla sanakirjat, lisäämällä hieman modulaarisuutta käyttämällä parempia tietorakenteita ja vähemmän kovakoodaamista. Olisin ohjelmoinut lisää karttoja ja luonnut pari uutta pelaaja hahmoa, jolloin pelaaja olisi voinut valita mieluisensa hahmon. Olisin myös lisännyt paremman pistelistan, josta näkisi aikaisempien pelikertojen pisteet. Tämän lisäksi olisin yrittänyt saada tasojen päälle hyppäämisen korjattua, ja tuonut enemmän Marion tyyppistä tasohyppelypelin ominaisuuksia.

Viitteet

https://gist.github.com/rogerallen/f06ba704ce3befb5459239e3fdf842c7

https://zetcode.com/gui/pyqt5/

https://www.geeksforgeeks.org/making-label-jump-using-pyqt5-in-python/

https://stackoverflow.com/questions/57238032/how-do-i-make-this-pyqt5-character-move-on-arrow-key-input

https://doc.qt.io/qt-6/

https://github.com/equati0n/Super_Mario

https://stackoverflow.com/

https://www.youtube.com/c/AbdullahAghazadah/about

https://codereview.stackexchange.com/questions/143103/example-of-pyqt5-snake-game

https://arcade.academy/_modules/arcade/sprite.html

https://opengameart.org/

www.kenney.nl

Esimerkki kuvia pelistä