Projektityön dokumentti – Ohjelmoinnin Peruskurssi Y2

1. Henkilötiedot

Nimi:	Joel Laitila
Opiskelijanumero:	708182
Koulutusohjelma:	Kauppatieteet
Vuosikurssi:	2018
Päiväys:	2.5.2020

2. Yleiskuvaus

Projektini on tasohyppelypeli, jossa pelaaja ohjaa pingviiniä kentän läpi hyppien erilaisten tasojen päälle. Tavoitteena on päästä vihreän tason päälle, jolloin kenttä on päästy läpi. Jos pingviini tippuu alas tasoilta ja kokoaan pois pelikentältä, pelaajan on aloitettava alusta. Pelissä on toteutettu seuraavat (tekniset) asiat:

- Graafinen käyttöliittymä
- Toimiva törmäyksen tunnistus, jotta pelihahmo ei tipahda pois tasoilta
- Ehto voittamiselle: läpäise kaikki 3 pelikenttää. Yhden pelikentän läpäisee saavuttamalla vihreän tason.
- Ehto häviämiselle: tipahda pois pelikentältä -> täytyy aloittaa alusta
- Yksinkertaiset grafiikat:
    - Pelihahmo on kuva pingviinistä
    - Taustakuvat
    - Tasoina on käytetty suorakulmioita
- Musiikit; taustamusiikki, hyppy, häviäminen ja voittaminen
- Realistinen painovoima ja liikkuvuus
- Helppo laajennettavuus

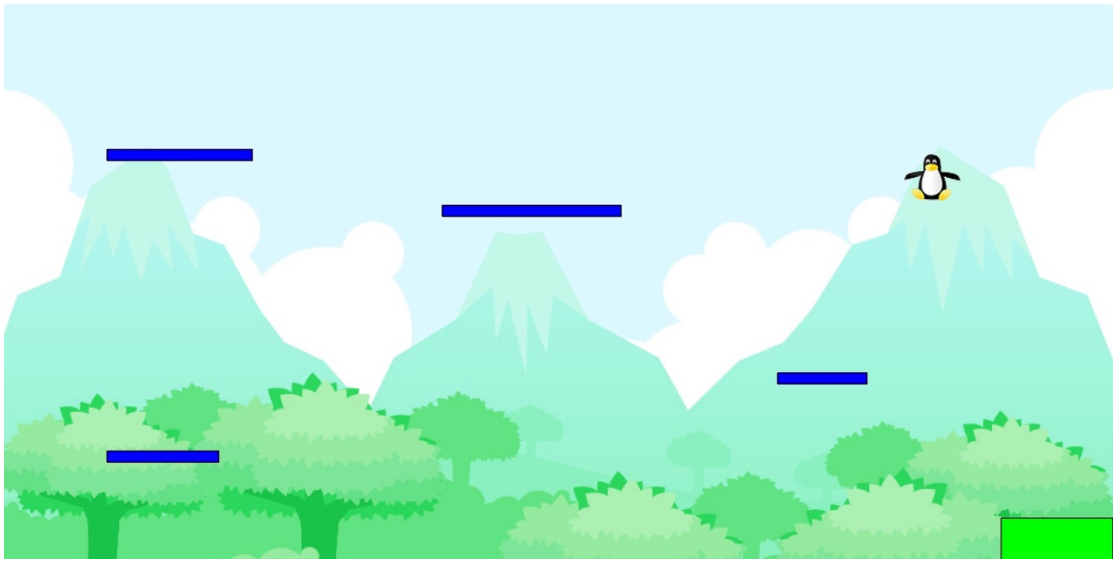
Jätin pelistäni pois kenttäeditorin, joka oli alkuperäisessä suunnitelmassani. Pelin toteutus tuntui itsestäni vaikealta ja opin siitä paljon, ja se sisältää kaikki keskivaikean pelin ominaisuudet. Lisäksi lisäsin muutaman pienen lisäominaisuuden peliin, kuten äänet ja musiikin.

3. Käyttöohje

Ohjelma käynnistetään ajamalla ”main.py” -tiedosto. Peli-ikkuna aukeaa, jossa on ohjeet pelin pelaamiseen, jotka ovat:

- Yritä saavuttaa vihreä taso hyppimällä tasojen päälle (katso alla oleva kuva)
- Käytä nuolinäppäimiä liikkumiseen: vasemmalla ja oikealla näppäimellä liikkuu vaakatasossa ja ylänuolella voi hyppiä
- Yritä olla putoamatta, sillä silloin peli täytyy aloittaa alusta

- Paina välilyöntiä aloittaaksesi
- Jokaisen pelin eri kentän alkaessa ohjelma pyytää käyttäjää painamaan välilyöntiä, jotta voi aloittaa pelaamisen.



Käyttöohje muuttui hieman suunnitelmastani, sillä päätin olla sisältämättä kenttäeditoria peliini. Lisäksi pelin voi lopettaa sulkemalla ikkunan. Jos pelaaja häviää tai läpäisee pelin, hän voi aloittaa pelin uudelleen painamalla välilyöntiä.

4. Ulkoiset kirjastot

Olen käyttänyt ohjelmassa seuraavia PyQt5:n kirjastoja lueteltuihin tarkoituksiin:

- PyQt5.QtWidgets, käytetty:
  - Peli-ikkunan luominen
  - Peli-ikkunaan eri toimintojen ja kuvien lisääminen
  - Peli-ikkunan asetusten asettaminen
  - Tasojen luominen

- PyQt5.QtCore, käytetty:
  - ”Skrollauksen” esto
  - Painettujen näppäimien tunnistaminen
  - Ajastin

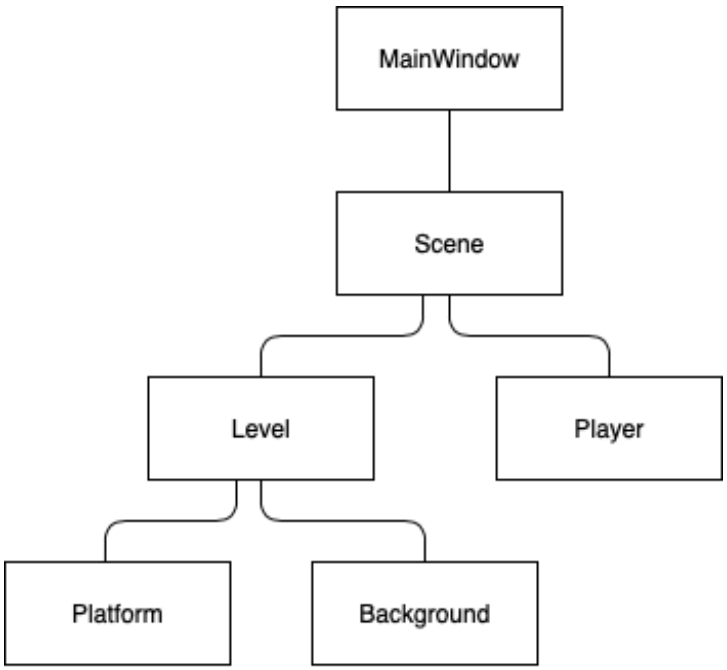
- PyQt5.QtGui, käytetty:
  - Taustakuvien ja pelihahmon kuvien lisääminen peliin
  - Painovoiman ja liikkuvuuden luominen vektoreiden avulla
  - Värien asettaminen

- PyQt5.Qt, käytetty:
  - Musiikin yhdistäminen eri käskyihin

- PyQt5.QtMultimedia, käytetty:
    - Musiikin asettaminen ja soittaminen pelissä

5. Ohjelman rakenne

Jaoin ohjelman viiteen luokkaan, sekä tein yhden erillisen moduulin pelin asetuksille. En lisännyt luokkakaavioon asetuksia, sillä ne eivät itsessään ole luokka, sekä niihin viitataan jokaisessa luokassa Platform ja Background luokkaa lukuun ottamatta. Vaikka useat pelin toiminnoista olisi voinut tehdä pelkästään metodeilla, koin luokkien tekemisen selkeyttävän ja helpottavan ohjelman käytettävyyttä.



Luokka MainWindow() sisältää sisällään kaikki peli-ikkunan tekniset asetukset, kuten skrollauksen eston, ikkunan koon määrittelyn, sekä itse pelin lisäämisen ikkunaan. Samassa moduulissa (main.py) on myös mukana itse pelin käynnistävä funktio. Päätin sisältää pelin käynnistävän moduulin tähän luokkaan, sillä sen alle rakentuvat kaikki muut luokat, eikä siihen viitata muissa luokissa. Itse peli toimii silloin, kun siihen luodaan näkymä Scene luokan avulla.

Luokka Scene pitää sisällään pelin toimivuudesta vastaavat ehdot ja luo sen graafisen näkymän. Luokkaan luodaan pelaaja Player luokan avulla ja uusia tasoja Level luokan avulla. Scene luokka vastaa pelin kulusta ja päivittää tapahtumia ajastimen avulla. Yksi keskeisistä metodeista on set_up_game(self), joka luo uuden pelin/kentän, kun peli käynnistetään, pelissä hävitään tai taso läpäistään. Käytin metodissa if-elif-else rakennetta, jonka mukaan metodi päättää, mikä taso/näyttö luodaan näkyville. Toinen keskeinen metodi game_update(self) päivittää peliä pelin sen hetkisten arvojen ja näppäinpainallusten perusteella. Metodi viittaa myös Player luokan update(self, keys_pressed) metodiin, jotta pelihahmo liikkuu näytöllä. Tätä metodia pyöritetään ajastimen avulla.

Luokka Player luo pelaajan, ja siinä tapahtuu myös pelaajan liikuttaminen ja painovoima. Kyseisessä luokassa on myös viitattu Scene luokkaan, jotta sen metodeja ja arvoja voidaan käyttää myös tässä luokassa. Luokan ehdottomasti keskeisen metodi on update(self, keys_pressed), jossa toteutetaan pelihahmon liikuttaminen painettujen näppäinten ja painovoiman mukaan. Metodissa tarkastetaan toisen metodin avulla eri objekteihin törmääminen, muutetaan painovoimaa ja nopeutta näppäinpainallusten ja törmäysten mukaan. Metodi viittaa myös jump-metodiin, jonka avulla pelihahmo voi pomppia. Metodin sisällä liikutetaan tasoja ja taustakuvia pelaajan vauhdissa sen liikkuessa eteenpäin. Lopuksi metodi myös tarkastaa, onko pelihahmo vielä hengissä tai onko pelikenttä läpäisty. Näiden perusteella se tekee tarvittavat muutokset pelin eri arvoihin, jotta seuraavalla Scene luokan game_update:lla asiat menevät oikein.

Luokka Level luo pelikenttiä annettujen arvojen perusteilla. Luokkiin Background ja Platform viitataan Level luokassa, kun pelikentälle luodaan oma taustakuva tai tasoja. Level luokan ainoat metodit luovat tasolle taustakuvan ja tasoja annettujen arvojen perusteella.

6. Algoritmit

Pelaajan liikkumista ja painovoimaa varten käytin seuraavaa algoritmia Player luokan update -metodissa:

- $velocity_t = velocity_{t-1} + acceleration$
- $position_t = position_{t-1} + (velocity + 0.5 * acceleration)$, jossa

jokainen tekijä (position, velocity & acceleration) on 2D vektori, joilla on arvot (x, y). velocity saa aluksi arvokseen (0, 0) ja position saa arvokseen sen kohdan, mihin pelihahmo laitetaan näytöllä. Jokaisen updaten mukana acceleration saa arvokseen aluksi (0, GRAVITY), jonka avulla pelihahmon painovoima toimii ja position vaihtuu, eli oletuksena pelaajan Y koordinaatti muuttuu GRAVITY:n verran. Pelaajan painaessa oikeaa tai vasenta nuolinäppäintä, accelerationin x arvo muuttuu, ja ylänuolta painaessa (ja hypyn onnistuessa) acceleration y muuttuu. Kaavassa t kuvaa aikaa, eli tässä tapauksessa se muuttuu joka updaten myötä. Näiden avulla pelihahmoa saadaan liikutettua todenmukaisesti nopeuden vaihtelujen mukaan.

Player luokassa tapahtuu myös törmäyksen tunnistus algoritmilla, jossa vertaillaan pelihahmon ja objektin koordinaatteja, leveyttä ja korkeutta. En osaa selittää kaavaa matemaattisesti auki (vain kynän ja paperin kanssa piirtämällä), mutta sen avulla tarkastetaan x ja y akseleilla, että osuvatko pelihahmon eri koordinaatit objektin koordinaatteihin. On tärkeää ottaa molemmat akselit huomioon, sillä muuten tulisi vääriä tulkintoja törmäyksestä (esim. voi olla täysin eri kohdalla y-akselilla, mutta samat x-arvot). Koordinaatteja tarkastellessa algoritmi ottaa huomioon pelaajan ja objektin leveydet ja korkeudet, jotta tarkastelu tapahtuu oikein.

7. Tietorakenteet

Olen suurimmaksi osaksi käyttänyt listoja tietorakenteiden varastoimiseen ja for in -looppeja niiden käsittelyyn. Useista pelin sisältämistä asioista tehdään olioita, mutta ne myös varastoidaan johonkin listaan, jotta niiden käsitteleminen olisi helpompaa. Olioiden tekeminen auttaa huomattavasti, sillä silloin listan arvoissa voi käyttää olin luokan metodeita, halutun asian löytämiseen.

8. Tiedostot

Ohjelma ei sinänsä saa mitään syötettä nuolinäppäimiä lukuun ottamatta, sillä kaikki syöte on jo annettu sille aikaisemmin peliä koodatessa. Suurin osa pelin syötteestä löytyy settings -moduulista, jossa ne kaikki ovat helposti muokattavissa. Suurimmat tiedostot ovat pelikenttien tasoja, jotka ovat listoja koordinaattien monikoista, joita voi helposti lisätä/muokata. Myös tasojen värit ovat helposti muokattavissa pelin asetuksissa. Pelin musiikki tulee mp3-tiedostoista, jotka on jo valmiiksi määritelty, mutta myös helposti vaihdettavissa.

9. Testaus

Testasin peliä useampaan otteeseen eri osa-alueilla, usein syöttämällä asetuksiin eri arvoja, ja kokeilemalla jos peli toimisi niiden puutteissa. Testaus ei sinänsä toteutunut samalla tavalla kuin kurssilla, sillä pelissä suurin osa virheistä tapahtuu pelihahmon liikuttamisen yhteydessä. Tämän takia päätin testata peliä yksinkertaisesti pelaamalla ja kokeilemalla kaikkia eri tyyppisiä skenaarioita, jota voisi tapahtua. Näin oli helppo paikantaa virheitä koodissa ja muokata sitä haluamaani päin. Esimerkkinä törmäyksen tunnistus: laitoin ohjelman tulostamaan tekstin joka kerta, kun törmään objektiin tahallisesti, jotta pystyin varmistamaan algoritmin toimimisen.

10. Ohjelman tunnetut puutteet ja viat

Itseäni eniten harmittava vika riippuu pelihahmon hyppimiseen ja törmäyksen tunnistamiseen. Tällä hetkellä törmäyksen tunnistus toimii oikein, mutta pelihahmon koordinaattien asettelussa tapahtuu joskus virheitä, kun se törmää eri tasoihin. Update funktion ideana on, että kun pelihahmo on tason päällä, sen pitäisi jäädä siihen, eikä tipahtaa alas. Tällä hetkellä se toimii lukuun ottamatta tapauksia, jolloin esimerkiksi pelkkä pelihahmon pää osuu tasoon. Tällöin funktio siirtää pelihahmon automaattisesti tason päälle, vaikka se ei oikeasti kuuluisi olla siellä. Sama ongelma toistuu, jos on hyppäämässä tasoa kohti, mutta esimerkiksi pelkkä pää osuu tasoon. Tällöin siis pelihahmo päätyy tason päälle, vaikka sen olisi kuulunut tippua siltä. Tämän virheen pystyy näkemään ensimmäisen tason alussa, kun koittaa hypätä lähimmän tason päälle ensimmäiseltä tasolta. Yritin korjata vikaa useamman kerran, mutta en päässyt haluamaani tulokseen.

Toisena puutteena on täydellinen törmäyksen estämisen puuttuminen pelistä. Halusin alun perin lisätä tasoja, joiden läpi olisi mahdotonta hypätä, mutta sen ohjelmointi tuotti minulle hankaluuksia. Tekemässäni pelin alkuperäisessä versiossa se toimi, mutta liikkuminen siinä oli paljon alkeellisempaa, sillä siinä ei ollut otettu huomioon fysiikan lakeja. Pidän tällä hetkellä pelissä siitä, että liikkuminen on luonnollista ja en tahtonut ottaa tätä ominaisuutta pois sillä hinnalla, että voisin lisätä täydellisen törmäyksen estämisen.

Pelin aloittaessa, kun Start screenin kuva tulee ruutuun ja painetaan välilyöntiä, Level 1 kuva ei ehdi tulla ruutuun, koska pelin päivitystahti on liian nopea. Tämä johtuu siitä, että välilyönti ei ehdi kadota keys_pressed:istä tarpeeksi ajoissa. Tämä ei ole niin kuin olisin asian halunnut menevän, mutta toisaalta sillä ei ole käytännön vaikutusta pelin kulkuun, eikä pelin pelaaja varmaan edes kiinnitä asiaan huomiota.

11. 3 parasta ja 3 heikointa kohtaa

Kolme parasta kohtaa:

1. Liikkuvuuden ja painovoiman todellisuudentuntu
2. Musiikit
3. Laajennettavuus

Kolme heikointa kohtaa:

1. Muutama vika törmätessä tasoihin
2. Koodissa on muutama toistuvuus
3. Pelissä ei ole erityisen suuria haasteita

12. Poikkeamat suunnitelmasta

Loppujen lopuksi poikkesin suhteellisen paljon suunnitelmastani. Koodaamiskokemukseni on vielä hyvin vähäistä, joten suunnitelmaa tehdessäni minulla ei ollut lähes mitään käsitystä siitä, miten peli tulisi koodata ja millainen edes sen rakenne tulisi olla. Tämän takia laadin pelin alkuperäisen suunnitelman erilaisten tutorialien pohjalta, mutta lopulta toteutukseni ei ole ollut samanlainen kuin ne.

Ajankäyttöarvioni oli ehdottomasti alakanttiin. Käytin projektiini huomattavasti enemmän aikaa, sillä se vaati todella paljon itseopiskelua. PyQt5:n sekä erilaisten algoritmien, kuten liikkuvuuden, toiminnan ymmärtäminen vaati itseltäni paljon aikaa.

Toteutusjärjestys ei sinänsä poikennut erityisesti, mutta useat suunnitelmaani lisäämästäni asioista toteutuivat hieman erilaisesti kuin olin etukäteen ajatellut. Huomasin eteneväni projektissa yksi asia kerrallaan, mutta kuitenkin parantelin samanaikaisesti muitakin projektin osa-alueita.

13. Toteutunut työjärjestys ja aikataulu

1.3-13.3

- Projektisuunnitelaman laatiminen
- Ohjaustapaaminen assistentin kanssa
- Eri toimintojen harjoittelu PyQt5:lla
- Erilaisten lähteiden ja tutorialinen lukemista

16.3-3.4

- Pelin ensimmäisen version tekeminen
- Pelihahmon liikkumisen ohjelmointi
- Erilaisten algoritmien vertailua ja kokeilemista (esim. törmäyksen tunnistus)
- PyQt5 kirjaston läpikäymistä
- Eri toimintojen harjoittelu PyQt5:lla

6.4-17.4

- Pelin toisen version aloittaminen (lähes koko koodi uusiksi)
- Algoritmien valitseminen toiseen versioon
- Peli-ikkunan muokkaamista
- Törmäyksen tunnistuksen ja painovoiman lisääminen
- Liikkuvuuden parantaminen
- Lisäominaisuuksien, kuten tasojen ja taustakuvien liikkuvuuden lisääminen

20.4-1.5

- Uusien kenttien tekeminen
- Erilaisten kuvien ja tekstien tekeminen peliin
- Koodin hienosääntöä ja sen jakaminen paremmin eri luokkiin
- Koodin jatkuva testaaminen
- Kommenttien lisääminen koodiin
- Musiikin lisääminen peliin
- Jatkuvaa pientä hienosäätöä ja viimeistelyä

2.5-8.5

- Projektin dokumentin laatiminen
- Viimeisiä korjauksia koodiin
- Pelin testaaminen

14. Arvio lopputuloksesta

Kokonaisuudessa tunnen onnistuneeni projektissa. Se toimii kuten pitääkin muutamaa pientä vikaa lukuun ottamatta. Täytin pelissä keskivaikean projektin vaatimukset, sekä lisäsin muutaman lisäominaisuutta peliin, kuten musiikin ja äänet.

Yritin pitkän aikaa korjata virhettä törmäyksen tunnistuksessa pelihahmon hyppiessä, mutta tuloksetta. Yritin myös lukuisia kertoja lisätä törmäyksen täydellisen estämisen peliin. Päädyin aina tilanteeseen, jossa koko törmäyksen estäminen meni täysin pieleen. Törmäyksen tunnistus toimii, mutta en osannut täysin muodostaa liikkuvuuden algoritmiani toimimaan niin, että se estäisi kaiken törmäämisen. Tämän takia päädyin pitämään koodini nykyisellään. Näiden kaltaisissa ongelmissa huomaan suhteellisen heikon fysiikan ja kohtalaisen matematiikan osaamiseni koituvan kohtalokseni.

Uskon, että valitsin käytettävät tietorakenteet ja luokkajaon hyvin. Se on tällä hetkellä selkeä ja koodin toistuvuutta on vain muutamassa kohtaa (esimerkiksi. set_up_game metodissa Scene luokassa). Uskon, että tämäkin toistuvuus on korjattavissa, esimerkiksi tekemällä listan eri pelikentistä settings:eihin ja käyttämällä sitä hyödyksi.

Tulevaisuudessa ohjelmasta voisi yrittää korjata sen viat ja puutteet, ja lisätä esimerkiksi vihollisia peliin. Näin peliin saataisiin lisää haastavuutta, koska tällä hetkellä se on suhteellisen helppo. Ohjelman rakenne soveltuu näiden muutosten tekemiseen, toki pieniä muutoksia siihen on luultavasti tehtävä.

Tällä hetkellä pelissä käydään for -looppeja muutamaan kertaan putkeen (esimerkiksi kun taustakuvia ja tasoja liikutetaan), ja uskon tämän olevan korjattavissa yksinkertaisempaan muotoon. Huomasin nämä ongelmat vasta suhteellisen myöhään, joten en päätynyt tekemään niihin muutoksia loppupalautuksia varten, jottei yllättäviä vikoja esiintyisi projektidemossa.

15. Viitteet

PyQt5 Documentation, https://doc.qt.io/qtforpython/

KidsCanCode, Game Development with Pygame, https://www.youtube.com/playlist?list=PLsk-HSGFjnaH5yghzu7PcOzm9NhsW0Urw

MDN web docs, 2D collision detection, https://developer.mozilla.org/en-US/docs/Games/Techniques/2D_collision_detection

Wikipedia, Platform game, https://en.wikipedia.org/wiki/Platform_game

Wikipedia, Collision detection, https://en.wikipedia.org/wiki/Collision_detection

Ohjelmoinnin Peruskurssi Y2 kurssimateriaali, https://plus.cs.aalto.fi/y2/2020/

Sofi Meronen, pelin musiikit

Pixabay, Level 1 taustakuva, https://pixabay.com/pl/illustrations/natura-g%C3%B3rskie-zach%C3%B3d-s%C5%82o%C5%84ca-2d-4466342/

Game Art 2d, Level 2 taustakuva, https://www.gameart2d.com/free-platformer-game-tileset.html

Pixabay, Level 3 taustakuva, https://pixabay.com/pt/illustrations/natureza-montanha-trabalho-art%C3%ADstico-4489569/

Pixabay, pelihahmon kuva, https://pixabay.com/pt/vectors/pinguim-mascote-tux-linux-pel%C3%BAcia-23331/

16. Liitteet
