Oskari Järvi
525747
AUT
30.4.2018

Yleiskuvaus

Projektin aiheena on tasohyppelypeli. Pelin tarkoituksena on ohjata hahmoa 2-uloitteisessa maailmassa hyppimällä erilaisten tasojen päälle. Ohjelmaan on toteutettu törmäyksen tunnistus, jotta hahmo ei läpäise tasoista läpi. Pelissä voittaa kun saavuttaa keltaisen maalin. Hahmo kuolee ja peli alkaa alusta jos osuu punaiseen ansaan. Peliin on suunniteltu yksi kenttä. Omasta mielestäni työ on toteutettu keskivaikean toteutuksen tasoisena.

Käyttöohje

Ohjelma käynnistetään ajamalla main.py -tiedosto. Ohjelman sisällä peli alkaa painamalla “Peli”-nappulaa.

Ulkoiset kirjastot

Ohjelman toteutuksessa on käytetty PyQt5:n QtWidgets, QtGui ja QtCore -kirjastoja grafiikan piirtämiseen, os-kirjastoa käyttöjärjestelmän käyttämiseen ja sys-kirjastoa ohjelman hallintaan.

Ohjelman rakenne

Main.py-tiedostossa alustetaan maailma ja käynnistetään graafinen käyttöliittymä ja muu ohjelma. GUI-luokassa käsitellään koko ohjelman piirtämistä, kuten esteiden, “main menun” ja hahmon piirtämistä. GUI-luokassa käynnistetään myös kello ohjelman tiettyjen metodien jaksollisen kutsumisen takia. GUI:ssa on mm. metodit mainmenu(), joka luo aloitusnäytön, game(), joka piirtää itse pelin ja hiscores(), joka piirtää parhaan 10 ajat hyödyntäen TopTenGraphics-luokkaa.

Sijainti-luokka toimii yksinkertaisena koordinaattiesityksenä, jossa on kokonaisluvut x- ja y-koordinaateille. Koordinaatit ovat pikseleinä alkaen pelin vasemmasta yläkulmasta oikealla ja alas. Sijainti-luokkaa hyödynnetään tasojen ja hahmon sijainnin käsittelyyn.

Maailman, eli ensimmäisen tason käsittely tapahtuu Maailma-luokassa. Se pitää sisällään kaiken tasoon liittyvän. GUI-luokka käytännössä piirtää kaiken mitä Maailma-luokka pitää sisällään. Maailma-luokassa oletusarvoisesti luodaan tasoon seinät, katon ja lattian.

Maailman esteet ja ansat luodaan Este-luokassa. Se sisältää esteiden vasemman yläkulman koordinaatit, sekä pituuden ja leveyden. Esteet piirretään GUI:ssa add_este_items()-metodissa. Maali-luokka sisältää käytännössä Este-luokan tiedot ja maalin piirtämisen samalla.

Hahmoon liittyvät asiat sisältyvät Hahmo-luokkaan, kuten mihin maailmaan se kuuluu ja sen sijainti maailmassa. Hahmon päivitys tapahtuu update()-metodissa. Siinä havaitaan mitä näppäintä on painettu ja liikutetaan hahmoa sen mukaan. Update()-metodista kutsutaan on_ground()-metodia, joka kertoo onko hahmo jonkin tason päällä, että hahmo voi hypätä. Siitä kutsutaan myös collision()-metodia, joka hoitaa törmäyksen tunnistuksen pelissä.

HahmoGraphicsItem-luokassa käsitellään hahmon grafiikat. Se piirtää hahmon Hahmo-luokan perusteella.

Kun pelaaja saavuttaa maalin, kutsutaan Voitto-luokkaa. Se tulostaa näytölle ilmoituksen pelin voittamisesta, sekä ajan joka siihen kului. Sen jälkeen Voitto-luokasta kutsutaan SaveStats-luokkaa, joka tallentaa tuloksen sav.txt tiedostoon. SaveStats-luokassa filehandler()-metodi luo sav.txt tiedoston, jos sitä ei ole olemassa ja tallentaa pelaajan nimen ja ajan, jos uusi aika on vanhaa aikaa parempi. GUI-luokassa hyödynnetään sav.txt tiedostoa parhaan 10 tulostamiseen.

Algoritmit

Ohjelmassa käytetään törmäyksen tunnistusta, jotta hahmo ei läpäise objekteja, mitä sen ei pidä. Algoritmi toimii siten, että jokaisen liikkeen jälkeen tarkistetaan ovatko hahmon ja minkään tason grafiikat päällekäin. Jos havaitaan päällekäisyyttä, niin vähennetään sijaintia siihen suuntaan mistä tultiin kunnes päällekäisyyttä kyseisen objektin kanssa ei tapahdu. Törmäyksen tunnistus käydään molemmille x- ja y-koordinaateille erikseen. Tämän kaltainen toteutus on valittu sen takia, että PyQt-kirjastossa on valmiina isCollidingWith()-metodi, jolla voidaan tarkistaa kahden graafisen objektin päällekäisyyttä.

Omassa toteutuksessani iteroidaan maailman jokaisen esteen läpi, koska tasot ovat melko pieniä ja esteitä ei ole tolkuttomasti. Suuremmassa ohjelmassa voisi iteroida lähimmät objektit läpi.

Ohjelmassa on myös tunnistus sille, että onko hahmo jonkin tason päällä, että voidaan hypätä. Tämän tunnistus toimii hieman samankaltaisesti kuin törmäyksen tunnistus. Hahmoa liikutetaan yhden pikselin verran alaspäin ja tarkistetaan onko päällekäisyyksiä. Jos päällekäisyyksiä on niin voidaan todeta hahmon olevan jonkin tason päällä.

Tietorakenteet

Ohjelmassa on käytetty pythonin valmiita tietorakenteita, kuten tuple, dictionary ja list.

Tiedostot

Ohjelma hyödyntää .txt-päätteistä tekstitiedostoa. Siihen tallennetaan parhaan 10 pelaajan nimi ja tulos muodossa (nimi)|(tulos).

Testaus

Ohjelman testaus oli jatkuvaa. Suurin osa testauksesta tapahtui visuaalisesti. Ajamalla tiedostoa näkee miten ohjelma tulostaa ja jos tuloste vastasi haluttua tyydyttiin toteutukseen. Esimerkiksi kun haluttiin tulostaa seiniä ja tasoja maailmaan voitiin ajaa ohjelma ja katsoa olivatko esteet halutulla tavalla.

Törmäyksen tunnistuksen testaus tapahtui käytännössä liikuttelemalla hahmoa ja katsomalla toimiiko se halutulla tavalla.

Ohjelman tunnetut puutteet ja viat

Kun pelaaja osuu punaiseen ansaan, niin pelaaja ei saa muuta indikaattoria kuolemasta kuin sen, että palaa takaisin aloitusruutuun.

Main menuun palaaminen on toteutettu mielestäni huonosti, sillä ohjelma käynnistyy kokonaan uudestaan. Lisäksi ohjelmassa olisi voinut välttää pop-up -ikkunoiden käyttämistä ja tulostaa kaiken yhteen ikkunaan.

Kun pelaaja aloittaa pelin ja kirjoittaa nimen, niin “Cancel”-nappula ei palaa takaisin main menuun, vaan pyytää nimeä kunnes se syötetään ja painetaan OK.

Pelistä ei pääse takaisin menuun.

Parhaat kohdat

- Ohjelma on muutakin kuin pelkkä peli – siinä on lisäksi myös valikko ja tulostaulukko
- Ohjelman reaaliaikainen toteutus

Heikot kohdat

- Palaaminen takaisin valikkoon on toteutettu omasta mielestäni huonosti

Poikkeamat suunnitelmasta

Ajat tallennettiin .txt-tiedostoon .csv-tiedoston sijaan, koska se olisi mielestäni tuonut liikaa monimutkaisuutta.

Törmäyksen tunnistusta ei tehty erilliseen luokkaan, sillä se osoittautui mielestäni tarpeeksi yksinkertaiseksi, jotta sen pystyi toteuttamaan metodina liikkeen apuna.

Suunnitelmassa oli ajateltu esteen olevan yläluokka seinä- ja taso-luokille, mutta se olisi ollut turhaa, sillä toteutin seinät ja tasot siten, että ne ovat käytännössä sama asia.

Peliin ei toteutettu mitään säätövalikkoa, jossa olisi voinut muokata esim. ohjausta.

Sijainti kuvattiin listan sijaan omana luokkanaan.

Toteutunut työjärjestys ja aikataulu

Aluksi luotiin Maailma-luokka, joka toimii pohjana kaikelle mitä peli sisältää. Sen jälkeen suunniteltiin GUI, jotta voidaan testata ohjelman toimintaa ja piirtää peliä.

Tämän jälkeen suunniteltiin maailmaan esteet ja niiden piirtäminen. Esteiden jälkeen luotiin hahmo-luokka ja sen piirtäminen. Tämän jälkeen luotiin hahmon liikkuminen sivusuunnassa, koska ei tarvitse ottaa “painovoimaa” huomioon, sekä suunniteltiin sivusuuntainen törmäyksen tunnistus. Tämän jälkeen luotiin y-suuntainen liikkuminen; “painovoima” ja hyppääminen. Kun liike oli saatu tehtyä lisättiin päävalikko ja maali, sekä voittoon liittyvät asiat. Päävalikon jälkeen luotiin aikojen tallennus ja niiden tulostaminen päävalikosta. Viimeisenä lisäsin tasoon ehdon “häviölle”, eli punaiset laatat, joihin osumalla peli alkaa alusta. Aikaa ohjelman toteutukseen meni n. suunnitellun verran.

Arvio lopputuloksesta

Omasta mielestäni ohjelma on pääpiirteissään onnistunut ja olen tyytyväinen tulokseeni. Tiettyjä asioita olisin tehnyt toisin, kuten sen että ohjelma pysyisi yhdessä ikkunassa jatkuvasti. Ohjelmaa on myös helppo laajentaa, sillä mikään luotu ominaisuus ei rajota uusia ominaisuuksia. Lisäksi siihen on helppo tehdä uusia tasoja, siten että maaliin tultua hahmo siirtyisi seuraavaan tasoon. Hahmojen ja esteiden grafiikkaa voisi parantaa. Tulosten tallennuksen voisi kryptata jollain, jotta käyttäjä ei pääsisi vapaasti muokkaamaan omia tuloksiaan.

Viitteet

https://doc.qt.io/qt-5/ - PyQt5 dokumentaatio

https://plus.cs.hut.fi/y2/2018/k05/pyqt/ - Kurssin oma PyQt5-ohje

Liitteet

https://version.aalto.fi/gitlab/jarvio1/tasohyppely – ohjelman git-repository