

General description

Yksinkertainen tasohyppelypeli, jossa Super Mario tyyliin liikutaan vasemmalta oikealle niin että vasemmalle voi palata vain sen matkan joka kyseisellä hetkellä on renderöity. Pelissä maa on laavaa, joten siihen koskeminen aiheuttaa välittömän kuoleman. Peli koostuu hahmosta, jota pelaaja liikuttaa, tasoista, joiden avulla pelaajan on tarkoitus liikkua, graafisesta taustasta, ja vihollisista, jotka yrittävät eri tavoin tuhota pelaajan. Alkuperäisestä suunnitelmasta peli eroaa siinä, että usean elämän sijaan on yksi elämä, mutta tasolle generoituu satunnaisesti paketteja, joista voi kerätä elämää (health). Maalin saavuttaa, kun etenee tarpeeksi pitkälle oikealle. Tasojen kokoja, sekä pelaajan ja vihollisen nopeuksia ja kokoja voi muuttaa 'obj_specs' tiedoston arvoja muuttamalla.

Instructions for the user

Ohjelma käynnistetään komentoriviltä menemällä projektin hakemistoon ja antamalla komento 'py main.py'. Kyseinen tiedosto sijaitsee "src-fol" kansiossa, joten hakemiston pitää osoittaa sinne.

Käynnistäminen avaa ikkunan, jossa on start -ja exit napit, joiden nimet selittävät jo itsessään funktionaalisuuden. Pelissä liikutaan vasemmalle ja oikealle a:sta ja d:stä, hypätään w:stä ja ammutaan välilyönnistä. Ilmassa voi hypätä kerran uudestaan painamalla w, eli tuplahypätä ts.

Elämää menettää vihollisen tai vihollisen ammuksen kosketuksesta ja satunnaisesti tasolle ilmestyvistä vihreistä laatikoista elämää voi kerätä. Pelin läpäisee kävelemällä maaliin, joka tulee vastaan tarpeeksi oikealle mennessä.

External libraries

PyQt : Graafisen käyttöliittymän toteutusta ja testausta varten

Random : Satunnaisten lukujen generoimiseen

"threading" ja "time" : Ohjelman suorituksen testausta varten. Time.sleep funktiolla koodin voi paussata n:ksi sekunniksi ja testatessa ohjelman suoritusta pitää QApplication olio käynnistää threadissa jotta testejä voidaan ajaa samanaikaisesti.

"unittest" yksikkötestausta varten

Structure of the program

GUI luokka sisältää graafisen käyttöliittymän

Player luokka sisältää pelaajaobjektin

Platform luokka sisältää taso-objektin, tasoa luodessa sille luodaan vihollinen tai ei

Enemy luokka sisältää vihollisobjektin, kun/jos taso-objektille luodaan vihollinen, asetetaan kyseinen taso-objekti luotavan vihollisobjektin "parent-tasoksi"

Lava-luokka luo laavaobjektin, jonka backgorund-luokka luo ja asettaa itselleen.

Background luo siis pelin taustan, eli laavaobjektin lisäksi skyobjectin (kuu/aurinko) ja taustan, joka on ruudun kokoinen nelikulmio.

GUI-luokka luo ikkunan (mainWindow), jonka näkymän (view) scene:ä vaihdetaan riippuen pelin tilasta. Kaikki edellä mainitut objektit luodaan ja lisätään sekä poistetaan näytettävään scene:en tarpeen mukaan. Kun peli aloitetaan, käynnistetään GUI-luokan ajastin joka kutsuu 10 millisekunnin välein grafiikka-objekteja päivittävää funktiota.

Kyseinen funktio kutsuu myös GUI:ssa määriteltä tasoja generoivaa funktiota.

GUI-olio luodaan main-luokassa, josta välitetään QApplication-objekti 'app' GUI-olion parametrina luokalle.

Algorithms

Vihollisobjekteilla on tieto niille kuuluvasta tasosta, jolloin vertailemalla omaa sijaintiaan tason reunojen sijainteihin estämme vihollista kävelemästä reunan yli.

Tasojen generointi hoituu lisäämällä ruudun 'ulkopuolelle' tietylle alueelle satunnaisesti koordinaatteihin tasoja yhtäaikaaisesti. Tasoja generoitaessa pidetään huoli, ettei yhdenkään lisättävän tason x tai y sijainti ole liian pieni suhteessa mihinkään toiseen lisättävään tasoon, jotteivat ne mene päällekkäin.

Pelaajan liikkuminen yksinkertaisesti toteutetaan muuttamalla pelaajaobjektin x ja y -koordinaatteja näppäinpainalluksiin yhdistettyjen funktioiden avulla. Mikäli pelaaja on puolessa välissä ruutua ja haluaa liikkua oikealle ei pelaajan sijaintia enää vaihdeta, vaan kaikkia tasoja liikutetaan vasemmalle pelaajan liikkumisnopeudella. Tämäkin tieto on välitettyä objekteille, jolloin ne osaavat liikkua tai olla liikkumatta oikein.

Putoaminen tapahtuu lisäämällä pelaajan putoamisnopeutta niin kauan kunnes se saavuttaa maximinsa, tai jos pelaaja putoaa tasolle tai laavaan. Hyppyfysiikka toimii samalla logiikalla paitsi, että hypätessä putoamisnopeus on alussa negatiivinen eli ”isompi”.

Data structures

GUI-luokassa kaikki lisätyt tasot ja viholliset ovat listassa. Koska viholliset ja tasot poistetaan niiden mentyä vasemmasta reunasta ulos, pysyvät listat lyhyinä ja olioiden käsittely niistä onnistuu nopeasti.

Vain kerran lisättävät oliot sidotaan GUI-luokassa muuttujiin ja käsitellään sellaisinaan.

Oliot sisältävät myös omia muuttujia, joita muuttamalla niiden tilaa ja käyttäytymistä voidaan muuttaa.

Eli kyseessä on siis mutable-oliot.

Files

Tiedostoja on vain yksi. obj_specs tekstitiedosto sisältää pelaajan ja vihollisen koot, ja nopeudet sekä

Tasojen leveyden ja korkeuden seuraavassa formaatissa:

```
Player : 20,2  
Enemy : 25,2  
Platform : 100,25
```

Eli 1. rivillä pelaajan koko ja nopeus

2. rivillä vihollisen koko ja nopeus

3. rivillä tason leveys ja korkeus

Testing

Testaus toteutettiin unittestillä sekä QTestillä. Unittesteillä testataan luokkia ja niiden metodeja ja QTestin sekä unittestin avulla ohjelman toiminnallisuus.

Jälkimmäisessä käytetään QBot-oliota joka käynnistää pelin ja ohjaa sitä. Testeissä kokeillaan menun toimivuus (pelin käynnistys sekä sulkeminen), pelaajan liikkuminen, vahingon teko viholliseen ja vahingonotto vihollisesta ja ammuksesta sekä maaliviivan ylittäminen.

Ohjelman testing.py luokka sisältää unit -ja QTestit

The known shortcomings and flaws in the program

Tasojen generointi ei vaikuta täysin satunnaiselta, eikä algoritmi ole muutenkaan täydellinen. Erityisesti jos tasojen kokoa ja pelaajan nopeutta säätää oikein txt-kansiossa ei seuraavalle tasolle hyppääminen joillain arvoilla ole edes mahdollista. Tämän korjaaminen vaatisi joko algoritmin muuttamista niin että se laskee tasojen etäisyyden myös pelaajan nopeuden funktiona tai että epäsopivat arvot txt-kansiossa korvattaisiin sopivilla default-arvoilla.

Obj_specs tiedoston ja sen toiminnallisuuden toteuttaminen jäi vähän viime tinkaankin, mistä johtuen vain tason kokoa tai pelaajan ja vihollisen kokoa ja nopeutta voi muuttaa tiedostossa. Formaatin laajentaminen edelleen aiheuttaa lisää ongelmia koodissa, jotka ovat täysin ratkaistavissa mutta vaativat muutoksia koodiin.

3 best and 3 worst areas

Ohjattavuus. Näppäinten tunnistus itsessään ei ollut iso homma mutta niiden samanaikaisen painamisen rekisteröiminen ja tunnistaminen oli. Painalluksen tunnistavan keyPressed:in ja vapautuksen tunnistavan keyReleaseEventin avulla välitettyjen totuusarvojen avulla ohjauksesta tuli sulava.

Objektien hallinnointi (optimointi). Poistamalla ruudun ulkopuoliset objektit saman tien ja generoimalla uusia vain muutaman kerrallaan on objektien käsitteleminen nopeata eikä viiveestä johtuvia virheitä aiheudu.

Taustan evoluutio, skyObject-olio vaihtaa väriä joka kierroksella päivän ja yön vaihtuessa.

Samalla taustan rgb-arvot vaihtuvat 'ajasta' riippuen oikeassa suhteessa alas tai ylös vaihtaen väriä tasaisesti vaaleansinisen ja mustan välillä.

Changes to the original plan

Alkuperäinen rakenne muuttui niin kuin oli oletettavissa.

Character luokan sijaan pelaajalla ja vihollisilla on omat luokat ja niiden grafiikkaobjektit sekä logiikka löytyy itse luokasta.

Huolimatta pienistä yksittäisistä eroavaisuuksista edellä mainitun ohella, on peli aika hyvin noudattanut suunnitelmaa.

Myös oikea aikahajautus oli aika hyvin linjoilla arvioidun kanssa, vaikka järjestys vähän erosikin.

Realized order and scheduled

1. Aloitusnäytön toteutus, eli pelin käynnistys avaa ikkunan kahdella napilla ja molempien nappien painallus rekisteröidään. 1-2h max?
2. Nappien toiminnallisuuden toteutus. 'Exit'-nappi lopettaa ohjelman ja 'Start'-nappi Aloittaa peli-loopin. 1-2h?
3. Pelaajan-hahmon toiminnallisuus. Luodaan grafiikkaobjekti, näppäinten painamiset muuttavat hahmon sijaintia haluamallamme tavalla. 2-8h?
4. Taso-objektien toteutus. Pelaaja voi liikkua tasojen päällä ja hyppiä niiltä toisille. 3-8h?
5. Vihollisten toteutus. (Liikkuvat edestakaisin tasolla kävelemättä yli) Pelaaja voi aiheuttaa niille vahinkoa ja toisinpäin. 2-5h?
6. Itse tason taustan toteutus. Maa laavaa joka aiheuttaa kuoleman, ja tausta muuten halutun näköinen. Elämien määrä näkyy, ja 'health' tilanne. 3-6h?
7. Scroll-efekti, eli uusien objektien luominen ja vanhojen siirtäminen vasemmalle luoden illuusion liikkumisesta maailmassa. 3-7h?
8. Maalin luominen ja pelin loppuminen se saavutettaessa. 1-4h?

Yllä alkuperäinen suunnitelma. 3. ja 4. vaiheita tuli toteutettua samanaikaisesti sillä testasin niiden vuorovaikutusta ja fysiikoita vielä kummankin luokan ollessa työn alla.

Heti näiden jälkeen aloin työstää jo scroll-efektiä sillä se oli oleellista pelaajan liikkumisen kannalta.

Seuraavaksi lisäsin viholliset, jota seurasi tausta.

Vahingonaiheuttaminen vihollisille tai pelaajalle tuli vasta loppupuolella sillä toteutus oli periaatteeltaan jo selvä. Vasta viimeiseksi kun kaiken muun totesin toimivan, lisäsin maalin ja maaliviivan.

Assessment of the final result

Toiminnallisuuden kannalta ohjelma suoriutuu hyvin, lagia ei esiinny ohjattavuus ja liikkuminen on sulavaa ja mekaniikat toimivat muutenkin niin kuin pitää. Kuitenkin ohjelman "suhteellisen" pieni koko antaa paljon anteeksi ja rakenteessa esiintyy varmasti puutteita. Erityisesti huomiota kiinnittää GUI:ssä esiintyvä `update_obj_pos()` joka myönnettäköön ei ole ehkä maailman kaunein funktio. Funktio suoriutuu hyvin mutta jos kyseessä olisi suurempi ohjelma, uskoisin refaktoroinnin olevan välttämätöntä. Suuri osa loopeista ja if-ehdoista saataisiin pois esimerkiksi lisäämällä objektien luokkiin muuttujia -ja/tai metodeja jolloin 'tarkistuksia' ei tarvitsisi tehdä niin sanotusti turhaan.

Ohjelman laajentaminen periaatteessa on aika suoraviivaista eikä vaadi suurta määrää työtä, vaikka graafista kenttäeditoria tai vastaavaa ei olekaan. Silti jollekin, joka ei koodia tunne voi mennä hetki, että tajuaa kokonaisuuden. Objektien muuttujien tai ominaisuuksien luku tekstitiedostosta jäi vähän viime tintaan mutta jos projektia jatkaisi, voisi tekstitiedoston formaattia laajentaa antaen isommat

vapaudet muuttaa pelin fysiikoita. Samalla varmasti nousisi esiin vähintään yksittäisiä ongelmia, joiden korjaaminen siistisi koodia

Olen yrittänyt ja suurelta osin noudattanutkin suurempaa kuvaa/suunnitelmaa mutta kuitenkin tekemisen vauhdissa on sinne tänne tullut tehtyä muutama osa väliaikaiseksi osa pysyväksi tarkoitettu mutta koodin sekaan unohtunut ratkaisu.

References

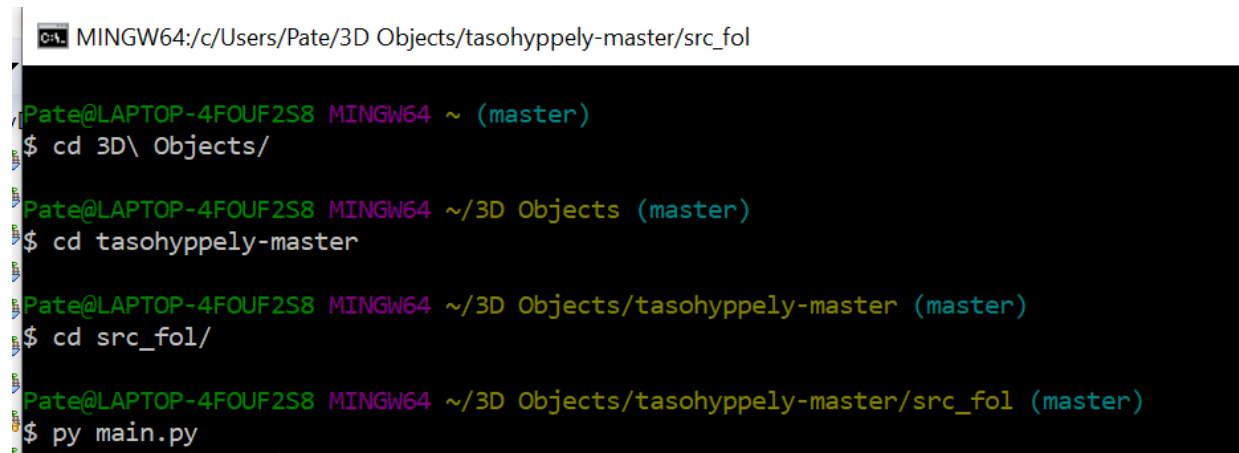
<https://doc.qt.io/>

<https://www.python.org/>

<https://pypi.org/project/PyQt5/>

<https://stackoverflow.com/>

Attachments



```
C:\Users\Pate\3D Objects\tasohyppely-master/src_fol

Pate@LAPTOP-4FOUF2S8 MINGW64 ~ (master)
$ cd 3D\ Objects/

Pate@LAPTOP-4FOUF2S8 MINGW64 ~/3D Objects (master)
$ cd tasohyppely-master

Pate@LAPTOP-4FOUF2S8 MINGW64 ~/3D Objects/tasohyppely-master (master)
$ cd src_fol/

Pate@LAPTOP-4FOUF2S8 MINGW64 ~/3D Objects/tasohyppely-master/src_fol (master)
$ py main.py
```

Käynnistys `cd → src_fol → py main.py`

