Y2 CS-A1121 - PROJEKTIN DOKUMENTTI

Nimi: Essi Tallgren

Opiskelijanumero: 710756

Koulutusohjelma: Bioinformaatioteknologia

Päivämäärä: 6.5.2020

Yleiskuvaus

Tein projektin aiheesta “tasohyppely”, johon tein muun muassa graafisen käyttöliittymän ja törmäyksen tunnistuksen. Tein kolme erilaista tasoa, joiden vaatimustaso nousee mitä pidemmälle menee (toisin sanoen taso 1 on helpoin ja taso 3 on vaikein). Pelin ehtona voittamiselle on päämäärään pääseminen, joka tässä tapauksessa on aarre aivan tason loppupäädyssä. Häviön ehtona on joko viholliseen tai ansaan osuminen tai “rotkoon tippuminen” eli kartalta ulos putoaminen. Tässä kohtaa olennaisia muutoksia suunnitelmaan nähden on se, että pelissä ei ole mahdollista kukistaa vihollisia.

Työ on suoritettu “vaativa” tasolla. Tämän voi perustella sillä, että projekti täyttää keskivaikean projektin vaatimukset ja siinä on niiden lisäksi muitakin lisäominaisuuksia. Näitä ovat esimerkiksi pelissä olevat liikkuvat viholliset, ansat, itse tehdyt grafiikat (kuvien suunnittelu ja piirtäminen omin käsin), useampi kuin yksi kenttä sekä taustamusiikki.

Käyttöohje

Navigoi komentorivillä projektikansion juureen ja kirjoita “python src/main.py” ja paina enter, jolloin ohjelma käynnistyy. Jos käytössäsi olevalla koneella on useampi python versio, kirjoita komentoriville “python3 src/main.py”.

Ohjelmassa voi siirtyä sivujen välillä joko eri tasoihin tai pelin ohjeikkunaan ikkunassa olevia nappeja painamalla. Pelissä ollessa hahmoa voi liikutella sivusuunnassa ‘A’ ja ‘D’ näppäimillä ja hyppiä voi painamalla välilyönti näppäintä. Takaisin menuun pääse pelissä painamalla ‘ESC’ näppäintä. Häviön tai voiton kohdalla tulee ikkuna jossa on jälleen nappeja joilla voi navigoida takaisin menuun, tason alkuun tai pois ohjelmasta.

Ulkoiset kirjastot

PyQt5 ja pygame. PyQt5 kirjasto käsittelee kaikki grafiikka- ja pelitoiminnot. Pygame kirjasto käsittelee tässä ohjelmassa ainoastaan musiikin soittamisen.

Ohjelman rakenne

Pelin luokat:

Menu - Menussa suoritetaan alkuikkunan luominen. Siinä näkyy painikkeet, joilla pääsee pelin eri tasoihin sekä pelin ohjeikkunaan. Siinä on myös painike “exit” jolla voidaan lopettaa ohjelman suoritus. Menussa myös suoritetaan musiikin soittamisfunktiot, jotka toimii muun ohjelman kanssa rinnakkaisesti, jolloin se ei vaikuta muun ohjelman toimintaan. Kun menusta siirrytään tasoihin tai infoikkunaan, menu piilotetaan hide() funktiolla. Menusta voidaan siis siirtyä World luokkaan ja sieltä takaisin.

World - Worldissa suoritetaan scenen ilmentäminen. World on QGraphicsSceneen pohjautuva luokka joka ilmentää grafiikat QGraphicsViewin avulla ja päivittää näissä ilmenevät muutokset update() funktion avulla. Menussa otetaan vastaan keyPressEventit ja viedään tieto näistä Fox luokkaan.

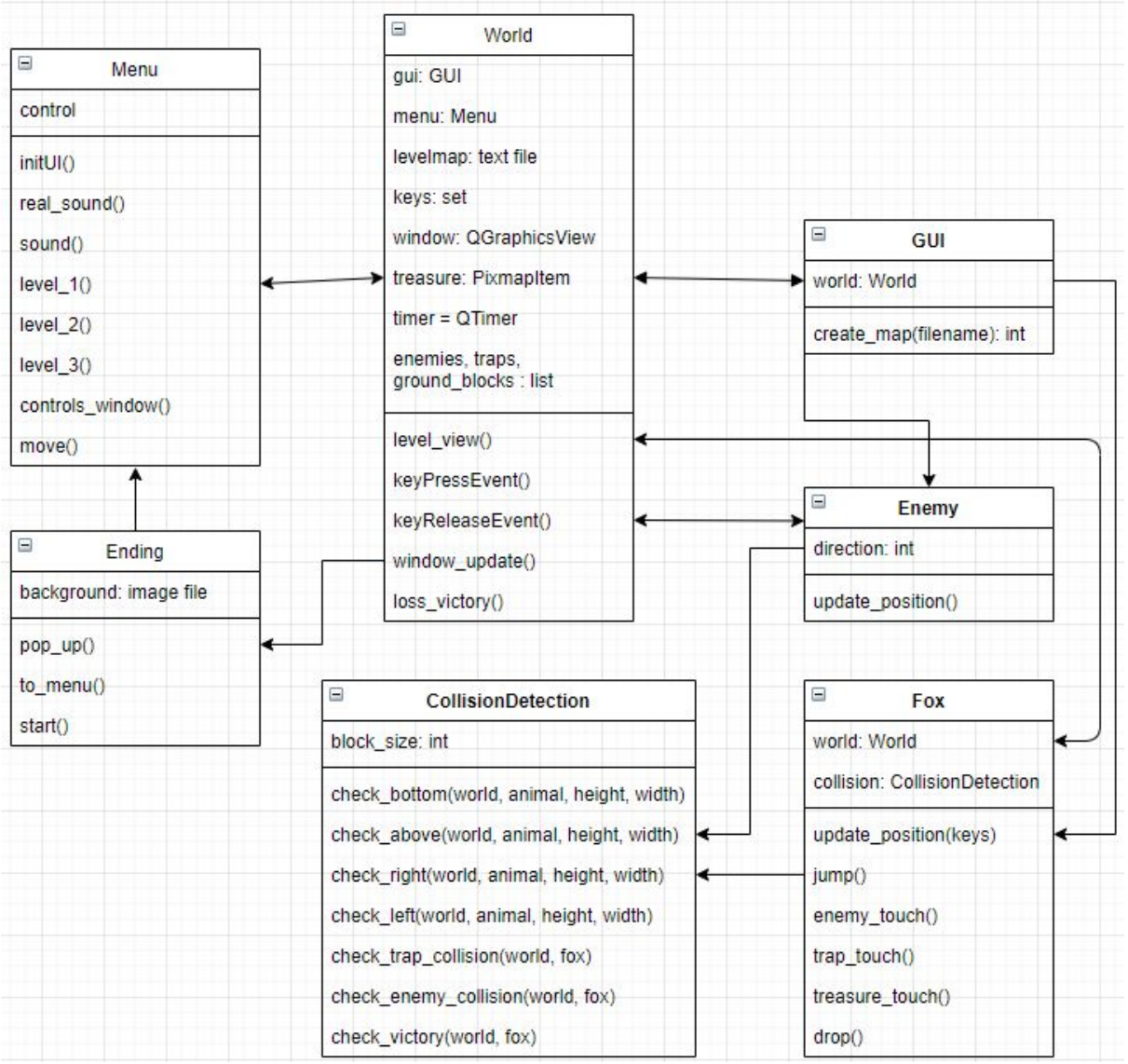
GUI - GUI luokka hoitaa karttatiedoston lukemisen ja sen pohjalta luo QPixmapItemit, jotka se lisää Worldiin.

Ending - Ending luokassa hoidetaan häviön tai voiton johdosta ilmestyvän ikkunan luominen QWidgetin avulla. Ikkunan nappeja painamalla voidaan joko aloittaa taso uudestaan jolloin scenen ajastin alkaa uudestaan ja pelihahmo liikkuu takaisin tason alkuun tai palata takaisin menuun josta voidaan navigoida haluamaan paikkaan. Ikkunasta voidaan myös sulkea ohjelma exit nappia painamalla.

Fox - Fox luokassa luodaan pelihahmo, joka on QGraphicsPixmapItem. Siinä on funktio, joka päivittää ketun sijainnin World luokasta tulleiden näppäinten painallusten ja CollisionDetection luokasta tulleiden sääntöjen mukaan. Fox luokassa myös käsitellään voitto ja häviö tapaukset, joiden havaitseminen tapahtuu CollisionDetection luokassa.

Enemy - Enemy luokassa luodaan viholliset, jotka ovat QGraphicsPixmapItemejä. Enemy luokassa tarkastellaan vihollisten liikkumista ja päivitetään eteneminen. Jos törmäyksen tunnistus palauttaa Enemy luokkaan tiedon että vihollinen osuu seinään, vihollinen kääntyy ja jatkaa matkaa seuraavaa seinää kohti.

CollisionDetection - CollisionDetection luokassa määritellään säännöt minkä mukaan hahmot voivat liikkua. CollisionDetection luokassa käydään läpi kartan esineet ja verrataan niiden koordinaatteja hahmon koordinaatteihin. Jos hahmo on liian lähellä tai kiinni esineessä esimerkiksi vasemmasta reunastaan, palauttaa funktio check_left() arvon True. Jos vasemmalla puolella ei ole esinettä, palautetaan arvo False.



Testaus luokat:

Test - Test luokassa käytetään yksikkötestausta testaamaan GUI luokan karttatiedoston lukemista.

CorruptedChessFileError - CorruptedChessFileError luokassa käsitellään Exceptionit jotka tulevat kun karttatiedostossa on virheitä.

Algoritmit

Ohjelmassa olevat algoritmit ovat muun muassa liikkuminen (jos sitä voi algoritmiksi kutsua), törmäyksen tunnistus ja hyppy. Liikkuminen tapahtuu 6 pikseliä per kuva nopeudella ja toimii siten, että joka päivityksen (eli “kuvan”) kohdalla hahmo siirtyy 6 pikseliä haluttuun suuntaan, eli vasemmalle tai oikealle.

Törmäyksen tunnistuksessa katsotaan vasen, oikea, ylä ja ala puoli jokainen erikseen ja jos millään puolella on esine liian lähellä, palauttaa sivusta vastaava funktio arvon True. Törmäyksen tunnistus vertaa hahmon ja kentällä olevien esineiden koordinaatteja toisiinsa ja sen perusteella ilmoittaa, osuuko ne toisiinsa.

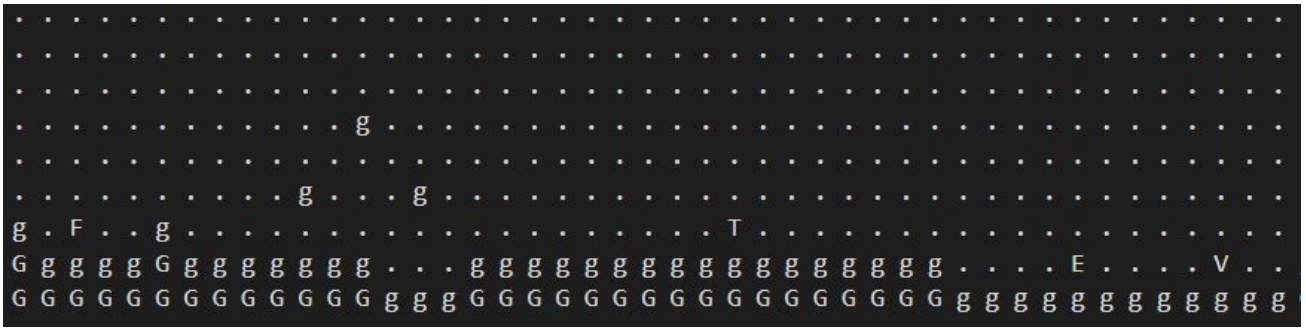
Hyppy toimii siten että painaessa välilyöntiä hahmo nousee ylöspäin nopeudella 10 pikseliä per kuva ja hidastuu joka päivityksen kohdalla painovoiman arvon verran. Tämä hidastuminen jatkuu kunnes nopeus on 4 pikseliä per päivitys ja kun hyppy ajastin on laskenut 38 hyppy loppuu ja hahmo tippuu maahan painovoiman vaikutuksesta nopeutuvalla vauhdilla, jonka nopeuden kasvu päättyy kun hahmo on tippunut 11 päivitystä. Nämä arvot ovat tulleet pääasiassa kokeilemalla. Eli katsomalla milloin hyppy on tarpeeksi korkea esineiden päälle pääsemiseksi ja milloin nopeus on tarpeeksi matala, että hahmo ei mene esineiden sisään.

Tietorakenteet

Tässä projektissa tietorakenteita ovat setit (set()) ja listat. Set() funktio on tehokas tietyn jäsenen löytämisessä, kun järjesyksellä ei ole väliä. Hahmon liikuttamisessa katsotaan löytyykö tietty kirjain painetuista näppäimistä ja siksi set() on näppäimien painalluksien tallentamiseen. Lista on hiukan nopeampi kuin setti jäsenten yli iteroimisessa, ja tämän vuoksi sitä ollaan käytetty muissa tiedon tallentamisissa. Esimerkiksi viholliset ovat tallennettuna listaan. Pelin pyöriessä, pitää koko ajan iteroida vihollislistan yli ja liikuttaa niitä scenessä, tämän vuoksi iteroimiseen on hyvä käyttää listaa.

Tiedostot

Projektissa käytetyt tiedostot ovat kuvatiedostot sekä tekstitiedostot. Tekstitiedostoissa on tallennettuna tasojen “kartat”, jonka pohjalta pelissä luodaan tasojen grafiikat. Esimerkkinä, tiedosto on suunnilleen muotoa:



Jossa eri merkit ovat erotettuina väleillä ja merkkien pohjalta tasoon laitetaan:
. = tyhjä kohta, johon ei laiteta oliota, ns. “taivas”.
g = 100 x 100 pikselin kokoinen neliö, joka kuvaa maata jossa on ruohoa päällä.
G = 100 x 100 pikselin kokoinen neliö, joka kuvaa maata ilman ruohoa.
F = 100 x 84 kokoinen kettu pelihahmo.
V = 100 x 74 kokoinen aarrearkku, joka on pelin määränpää.
E = 144 x 144 kokoinen vihollinen, poliisikoira.
T = 100 x 16 kokoinen ansa, karhunrauta.
R = 100 x 100 kokoinen kivi.

Kuvatiedostot ovat png muotoisia tiedostoja, joissa on kaikkien graafisten elementtien kuvat.

Testaus

Ohjelman testaus meni pääosin pelaamalla peliä useita kertoja ja kokeilemalla mahdollisia virheitä aiheuttavia toimintoja, kuten esineisiin törmäämistä eri kulmista ja nopeuksista. Itse hahmojen liikkumista ja tämän tapaisia toimintoja oli paras testata pelaamalla ja kokeilemalla useita kertoja ja näissä en käyttänyt esimerkiksi yksikkötestausta.

Testasin ohjelmaa sitä rakentaessa ajamalla ohjelmaa. Esimerkiksi törmäyksen tunnistusta testasin liikuttamalla hahmoa kohti tasossa olevia kappaleita eri kohdista ja kulmista ja tarkastelin menikö hahmo esineiden läpi vai ei ja pysähtyikö hahmo oikeassa kohdassa. Testauksen suunnittelussa ei itsessään ole niin sanotusti olennaisia aukkoja, mutta tällainen manuaalinen testaaminen voi aiheuttaa sen, että vaikka ajaisi ohjelman useaan kertaan, voi joku olennainen virhe jäädä huomaamatta. Ohjelma läpäisi kokeilulla tehdyt testit, mutta kuten sanoin manuaalisen testaamisen mahdollisesta ongelmasta, en voi olla täysin varma, että jotain virhettä ei varmasti tulisi.

Käytin yksikkötestausta GUI luokan create_map() funktion testaamiseen. Testasin tuleeko virheilmoitus jos karttatiedostossa on vääriä kirjaimia, puuttuu pelihahmo tai aarre tai jos tiedostoa ei voi lukea ollenkaan. Testauksen perusteella nämä ilmoituksen tulivat kun kuuluikin, eli läpäisee testit.

Ohjelman tunnetut puutteet ja viat

Ohjelmassa puutteita voisi olla yksikkötestauksen vähäisyys. Tein toki testausta tiedoston lukua varten, mutta muuten testaus on ollut vain manuaalista.

Ohjelmassa puutteena on myös pelihahmon liikkeen hienosäätö. Esimerkiksi törmäyksen tunnistus on tehty siten, että törmäysalue hahmolla on suorakulmion muotoinen, vaikka hahmo itse on ketun muotoinen. Eli esimerkisi hahmon hypätessä ylös kohti yläpuolella olevaa neliön muotoista esinettä siten, että hahmo ei ole täysin esineen alla, törmäyksen tunnistus pysäyttää liikkeen vaikka hahmo ei oikeasti koskisi esineeseen, sillä törmäysraja otetaan hahmon korkeimmasta kohdasta.

Puutteena on myös ehkä projektin ohjeen kohta “kiinnitä huomiota laajennettavuuteen”, jos tällä tarkoitetaan sitä että käyttäjä voisi itse tehdä tasoja tai jotain sen tapaista. Jos tämä kuitenkin tarkoitti sitä että uusia kenttiä on helppo koodata ohjelmaan, niin silloin tämä kriteeri täyttyy hyvin, sillä kartanlukua ei olla yksilöity tietylle tiedostolle vaan sillä voi lukea useita saman formaatin omaavia tiedostoja (formaatti näytetty ylempänä). Jos olisin tehnyt ohjelmaan toiminnon, että pelaaja voi itse suunnitella tason, tiedän kyllä kuitenkin miten tekisin sen (aika ei loppua kohden ikävä kyllä riittänyt tälle):

Olisin luonut menuun napin “Create level” jossa siirrytään menu luokan sisällä funktioon jossa ensin käyttäjä menee widgettiin jossa käyttäjä kirjoittaa tyhjään tiedostoon oman suunnitteleman kartan ohjeistusten mukaisesti (open(file, “w”) jolloin valittu tiedosto tyhjenee jos siellä on valmiiksi jotain, tämä mahdollistaa Create level toiminnon käyttämistä useaan kertaan). Kun käyttäjä on luonut haluamansa karttatiedoston ohjelma siirtyy samalla tavalla kuin muissa tasoissa World luokkaan jossa yhtenä muuttujana asetetaan luotu tiedosto jonka mukaan scene rakentuu. Tästä siis päästään itse luotuun tasoon ja peli toimii kuten normaalistikin. Tämä käyttäjän itse luoma taso on siis melko helppo toteuttaa, kunhan siihen olisi vaan tajunnut varata enemmän aikaa.

3 parasta ja 3 heikointa kohtaa

Kolme parasta kohtaa omasta näkökulmastani ovat ensimmäisenä grafiikat, vaikka se ei välttämättä ole tämän projektin tärkeimpiä osia. Toinen vahvuus on kartan toteuttaminen. Tiedostomuodon vuoksi kartan luominen on erittäin yksinkertaista ja pelkästään tiedostoa tutkailemalla voi päätellä miltä taso näyttää. Kolmantena vahvuutena sanoisin olevan törmäyksen tunnistuksen jaottelu kahteen osaan. Käyttämällä collidesWithItem toimintoa vihollisten, ansojen ja aarteen kohdalla, koodi on paljon paljon siistimpää kuin se voisi olla, ja säästytään turhalta “purkkakoodilta”. Toisena osana törmäyksen tunnistusta on vasen, oikea, ylä ja ala sivun tunnistus halutulle hahmolle.

Törmäyksen tunnistuksen toisesta osasta kun mainitaan, tullaan myös ensimmäiseen heikkouteen projektissani. Tunnistus on toimiva (ainakin omien kokeilujen perusteella), mutta jos hahmon nopeutta kasvattaa (pixels/frame) tunnistuksen “puskurialue” ei riitä ja hahmo menee esineiden sisään. Puskurialue kattaa 4 pikseliä esineen rajan molemmin puolin ja jos hahmo tulisi esinettä kohti esimerkiksi 10 pikselin nopeudella, se voisi mennä suoraan tämän puskurialueen yli ja mennä esineen sisään. Jos tältä haluttaisiin välttyä, ratkaisuna olisi esimerkiksi sellaisen mekanismin luominen, missä hahmo hidastaa nopeuttaan tullessaan tietylle etäisyydelle esineistä.

Myös hyppy on yksi heikkouksista sen “epäsiisteyden” vuoksi. Koodi olisi voinut olla yksinkertaisempi siltä osalta ja nopeudet olisi voinut laskea selvemmän kaavan kanssa.

Myös ylempänä mainittu hahmon liikkeen hienosäädön puute on ohjelman yksi heikkous.

Poikkeamat suunnitelmasta

Poikkeamia suunnitelmasta on esimerkiksi alussa mainittu vihollisten kukistamisen puuttuminen. Tämän johtuu siitä, että tein törmäyksen tunnistuksen collidesWithItem funktiolla. Jos olisin tehnyt vihollisen kukistamisen, olisi vihollisille täytynyt vielä erikseen tehdä törmäyksen tunnistus, joka tekee eri toimintoja osumissuunnasta riippuen.

Toteutunut työjärjestys ja aikataulu

Projektin tekeminen meni melko hyvin aikataulun mukaan. Odotettua enemmän meni tiedon keruuseen ja PyQt5 ymmärtämiseen. Työjärjestys oli myös melko lähellä suunnitelmaa, tosin alussa ennakoitu järjestys, että hahmo laitettaisiin tasoon

törmäyksen tunnistuksen luomisen jälkeen, oli oikeasti toiste päin. Järjestys oli siis kutakuinkin tällainen:

- Ohjelman pohjustus ja luokkien alustus
- Menu
- Musiikin soitto toiminnot
- Tasotiedostojen suunnittelu
- GUI:n tekeminen ja samanaikaisesti liikkumattomien olioiden luominen
- Worldin tekeminen, eli GUI:n sceneen laittamien olioiden visualisoiminen
- Hahmon luominen ja kartalle sijoittaminen
- Törmäyksen tunnistus ja hahmon liikuttaminen
- Vihollisten luominen ja kartalle sijoittaminen
- Voittaminen ja häviäminen
- Lisäominaisuuksia

Testaaminen ja grafiikoiden suunnittelu/piirtäminen oli koko projektin ajan jatkuvaa, eikä siis sijoittunut tiettyyn väliin työjärjestystä.

Arvio lopputuloksesta

Ohjelman laatu on mielestäni melko hyvä. Koodi on siistiä ja esimerkiksi tietorakenteiden valinta on tehty harkinnalla. Koodi on selvästi jäsennelty, eikä kaikki asiat ole samassa luokassa tai muuta sen tapaista. Hyviä puolia on grafiikat, törmäyksen tunnistuksen jaottelu sekä karttatiedosto ja sen pohjalta tasojen ilmentäminen. Olen myös tyytyväinen siihen, että sain jotain yksikkötestausta aikaiseksi, vaikka sen vähäisyys on silti samaan aikaan työn huonoissa puolissa.

Huonoja puolia on törmäyksen tunnistuksen pettäminen jos hahmon nopeus asetaan liian suureksi. Yksikkötestauksen vähäisyys. Hypyn “epäsiisteys” (koodi on sen osalta ehkä hiukan liian purkkaa ja siinä olisi voinut tehdä algoritmin paljon laskennallisemmin eikä niin paljon vaan kokeilemalla).

Viitteet

https://rpa-automation.blogspot.com/2018/05/python-pyqt5-set-background-image.html
https://plus.cs.aalto.fi/y2/2020/materials_k05/k05_pyqt_pyqt/
https://doc.qt.io/qt-5/qgraphicsview.html#centerOn-1
https://doc.qt.io/qt-5/qgraphicsitem.html#collidesWithItem
https://doc.qt.io/qt-5/qgraphicspixmapitem.html#details
https://doc.qt.io/qt-5/qgraphicsitem.html#paint
https://doc.qt.io/archives/qt-4.8/stylesheet-examples.html

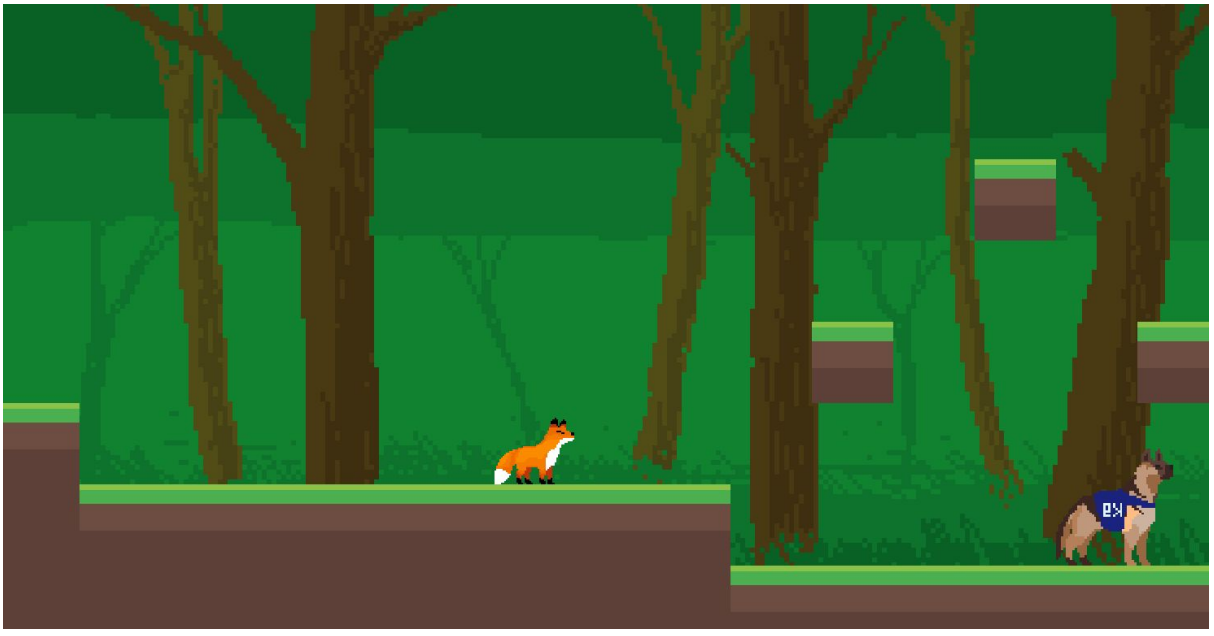
https://stackoverflow.com/questions/10283067/playing-music-with-pyglet-and-tkinter-in-python
RobotWorld tehtävä
Readin Chunked File tehtävä (testaus)
Googletukset
Muut Qt Documentation sivuston ohjeistukset

Liitteet

Menu:

Info ikkuna:

Peli ikkuna jossa näkyy vihollinen:



Aarre:

Ansa: