

Y2 Strategia Peli Dokumentaatio

Henry Gustafsson

787226

Elektroniikka- ja sähkötekniikka

08.05.2020

Yleiskuvaus projektista:

Strategiapeli, missä pelataan joko toista pelaajaa tai bottia vastaan ruudukko kentällä. Molemmilla puolilla on 3 erilaista pelattavaa hahmoa, Archer, Fighter ja Giant. Tehtävänä on eliminoida kaikki vastustajan hahmot voittaakseen. Jokaisella hahmolla on tietty määrä elämän pisteitä (hp), isku pisteitä (attack damage) ja etäisyys mihin voi lyödä (range). Pelin pelikenttä koostuu ruudukoista ja kentällä on kivi esteitä. Peli on toteutettu keskivaikea tasolla.

Käyttöohje:

Ohjelma käynnistetään juoksemalla main.py tiedosto, missä luodaan peli koodirivillä. Tämän jälkeen peli käynnistyy ja avaa ikkunan. Peliä pelataan klikkaamalla hiirellä hahmoja ja liikuttamalla niitä tai lyömällä niillä vastustajan hahmoja. Ikkunassa lukee päivitettyt tiedot hahmoista ja niiden elämän pisteistä. Jos peliä haluaa jatkaa myöhemmin, voi sen tallentaa painamalla savegame nappulaa ja antamalla siihen valinnaisen tallennustiedoston nimen.

Ulkoiset kirjastot:

Projektissa on käytetty seuraavia kirjastoja:

PyQt5, math, json, sys

Ohjelman rakenne:

Ohjelma toiminta jaetaan seuraaviin luokkiin:

Main, World, Square, Direction, Player, Character, Game, GUI, GameEngine, SquareGraphicsItem, CharacterGraphicsItem.

Main luokassa luodaan pelikenttä, pelaajat ja hahmot. Peliä voi pyörittää Main luokassa ja pelata printti muodossa jos graafinen käyttöliittymä (GUI) otetaan pois käytöstä. **World** luokassa luodaan pelikenttä, sijoitetaan pelihahmot ja kivet kentälle sekä päivitetään missä kukin hahmo sijaitsee. **Square** luokassa luodaan jokaisen ruudun tyyppi ja päivitetään missä ruudussa on mitäkin. **Character** luokassa luodaan eri hahmot ja päivitetään jokaisen hahmon elämän pisteiden määrä. **Direction** luokassa tarkistetaan mihin suuntaan hahmo on liikkumassa ja onko kyseinen ruutu vapaana tai olemassa. **Player** luokassa luodaan pelaajat, määritetään kuka lyö ja mitä hahmoa sekä päivitetään kenellä on mikäkin hahmo olemassa (elossa). **Game** luokassa luodaan pelin tallennus. Luokassa voidaan ladata tarvittavat tiedot uuteen tiedostoon tai ladata vanhasta tiedostosta pelin tiedot ja tilanne pelattavaksi.

GUI luokassa luodaan graafinen ohjelma ja juostaan peliä hiiri toiminnoilla.

SquareGraphicsItem luokassa luodaan grafiikka pelikentän ruutujen tyypeille ja annetaan näille klikkaus ominaisuus. **CharacterGraphicsItem** luokassa luodaan grafiikka pelin eri hahmoille ja päivitetään niiden olemassa olo sekä myös annetaan klikkaus ominaisuus. **GameEngine** luokassa suoritetaan pelin toiminnot grafiikka liittymän (GUI:n) kautta.

Viimeisellä sivulla on kuva UML-taulukosta.

Algoritmit:

Pelihahmojen hyökkäys etäisyys (attack range) on laskettu yksinkertaisesti pythagoraan lauseella ruutujen koordinaatteja hyväksikäyttäen. Hyökkäävän hahmon ja kohteen sijainti on ajantasainen ja näistä pystytään tarkistamaan onko etäisyys pienempi kuin hyökkäävän hahmon määritetty hyökkäys etäisyys.

Botin hahmoille on määritetty kohteensa ja näin se pyrkii liikuttamaan pelihahmoja vuorotellen kohteita päin. Botin hahmot Giant ja Archer jahtaavat ensin vastustajan Giant hahmoa. Botin Fighter jahtaa taas vertaistansa eli vastustajan Fighteria. Botti siis tarkistaa mihin suuntaan (north, east, south, west) kannattaa kutakin hahmoa liikuttaa laskemalla pythagoraan lauseella etäisyydet jokaisesta viereisestä ruudusta kohteen ruutuun ja valitsemalla näin lyhin etäisyys. Kun botti on tuhonnut kohteensa, valitsee se seuraavaksi uuden kohteen järjestyksessä Giant, Fighter ja Archer.

Tietorakenteet:

Pelin tiedostojen tallentamiseen käytetään json muotoa. Json muotoa on helppo ja yksinkertaista käyttää. Tallennuksen tiedoston voi avata helposti ja muokata sen tietoja jälkikäteen.

Tiedostot:

Tiedostot ovat tallennettuna json muodossa ja kaikki pitäisi toimia ongelmitta kun ensimmäinen tallennus on tehty ja tallennus avataan kirjoittamalla tiedoston nimi ilman '.json' päätettä.

Testaus:

Testaus tehtiin käsin suorittamalla haluttuja toimintoja ja printtaamalla tulokset. Kun peli on luotu, testit on tehty pyörittämällä peliä ja samalla etsien virheitä.

Ohjelman tunnetut puutteet ja viat:

1. Ohjelma avaa saveme.json tallennuksen jos kirjoitetaan virheellinen nimi tallennusta avatessa. While loop:lla ja try/except:llä voi mahdollisesti korjata virheen kunnes olemassa oleva tiedosto löytyy.
2. Hahmot jättävät silloin tällöin tuntemattomasta syystä jälkeensä grafiikka bugeja kuten liikkuesssa seuraavaan ruutuun, näkyy edelleen edellisessä ruudussa hahmon grafiikka, mutta hieman rikkiäisenä esim. kolmiosta puuttuu kulma tai pohja.
3. Tekstit ja grafiikka pelissä eivät ole kaikkienensa valmiita tai ihanteellisia.
4. Jos pelin tallentaa kaksinpelissä kun on pelaajan 2 vuoro ja peli ladataan uudestaan niin peli alkaa pelaajan 1 vuorosta eli pelaajan 2 vuoro ohitetaan. Tämä on helppo korjata tallentamalla tiedostoon kumman pelaajan vuoro on.
5. Botin liikkuminen ei ole kovin älykäs ja umpikujan tullessa botti saattaa jäädä pomppimaan edestakaisin kahden ruudun välillä eikä osaa tällöin etsiä lyhintä avointa reittiä.

3 parasta ja 3 heikointa kohtaa:

Parhaat kohdat:

1. Minusta Main luokan alku on hyvin suunniteltu funktioiden kanssa niin että toiminnot lähtevät ja päättyvät Main luokkaan.
2. Pelikentän koon, pelihahmojen ja kivien sijainnin määrittäminen on helposti ja selkeästi muokattavissa Main luokassa. Myös hahmojen ominaisuudet kuten health, attack damage ja range ovat helppo muokata Character luokasta.
3. GUI ja GameEngine luokkien funktiot ovat melko yksinkertaisia ja selkeitä ainakin omaan silmään.

Heikot kohdat:

1. Joidenkin muuttujien nimet ovat sekavat kuten chara ja dir ym. sekä aloin toistaa character muuttujalle eri merkityksiä.
2. Hypin joissakin toiminnoissa turhaan luokasta toiseen kuten hahmon liikuttamisessa tai toisen hahmon lyömisessä. Näin polkua on vaikea seurata mitä funktio oikeen tekee.
3. Koodissa on paljon pätkiä mitä voisi tehdä lyhyemmin käyttämällä funktioita avukseen. Myös siistimällä koodia se näyttäisi hienommalta esimerkiksi bot_actions() funktiota voisi paloitella vaikka Direction luokkaan ja yleisesti nimetä muuttujia selkeillä nimityksillä.

Poikkeamat suunnitelmasta:

Toteutus poikkesi sinänsä melko paljon suunnitelmasta. Aluksi aloin tekemään peliä heti GUI:n kanssa eikä siitä tullut mitään. Tämän jälkeen aloin tekemään peliä printti versiona jolloin pääsin hyvään tahtiin. Ajan käyttö oli vaihtelevaa sillä välillä jumin koodaamisessa tai en ehtinyt tehdä projektia eteenpäin. Toisinaan välillä sain hyvän tahdin siihen kun oivalsin miten lähdän seuraavaa steppiä toteuttamaan. Loppujen lopuksi kuitenkin peli jäi paljon yksinkertaisemmaksi mitä olin suunnitellut sillä pelin teko oli haastavampaa kuin olin ajatellut eikä aika riittänyt ihmeellisimpiin ominaisuuksiin. Onneksi kuitenkin sain pelin valmiiksi ja toimimaan suht hyvin.

Toteutunut työjärjestys ja aikataulu:

Aloitin projektin luomalla Main ja World luokat missä alustin pelikentän (kaksinkertaisen listan). Tämän jälkeen loin kivet pelikenttään Square luokassa. Testasin ohjelmaa printtaamalla pelikentän (grid) jolloin jokaiseen ruutuun printataan numero joka kuvastaa mitä ruutu sisältää. Tämän jälkeen loin pelaajat Player luokalla ja näille pelihahmot kentälle omilla ominaisuuksilla. Seuraavaksi tein funktiot millä sain Direction luokkaa hyväksikäyttäen hahmot liikkumaan kentässä ja lyömään vastustajan hahmoja. Ennen graafista käyttöliittymän luontia, tein Game luokan ja sain toimimaan tallennus menetelmän json kirjaston avulla. Kun peli toimi printti versiona, aloin tekemään GUI luokkaa sekä Square- ja CharacterGraphicsItem luokkia. GUI:n tekeminen oli minulle haastavaa, mutta lopuksi sain grafiikat piirrettyä ottamalla mallia robotworld tehtävästä. Seuraavaksi aloin kehittämään pelille GameEngine luokkaa missä toteutetaan samat funktiot kuin Main funktiossa, mutta graafisen käyttöliittymän GUI:n kautta.

Alkuperäinen suunnitelma oli tehdä heti grafiikat pelille ja testata peliä sen avulla. Totesin sen kuitenkin olevan hölmö ja vaikea tapa aloittaa pelin tekeminen niin aloin tekemään printti versiona pelin runkoa.

Arvio lopputuloksesta:

Opin paljon mistä ohjelman tekeminen kannattaa aloittaa ja miten ylipäättänsä luokat ja funktiot ym. toimivat. Koodista ehkä huomaakin kuinka osa menetelmistä ovat tehty hankalasti ja osa helpommin ja yksinkertaisemmin. Ainakin itse huomaan kehityksen ja ymmärtämisen sillä alku oli erittäin vaikeaa verrattuna loppua kohden. Luokat ovat minusta hyvin jaettu, mutta osa funktioista hyppii edestakaisin turhaan ja vaikealla tavalla. Muuttujia olisi voinut nimetä paremmin ja kaikkia muuttujia ei edes olisi tarvinnut jos koodia olisi vähän siistinyt. Projekti jäi vielä kesken enkä saanut kaikkia ominaisuuksia tehtyä kuten monimutkaisempaa maastoa ja hahmojen erikoisiskuja. Peliä on kuitenkin helppo kehittää tästä eteenpäin halutessaan. Olen tyytyväinen projektiin ja koen kurssin erittäin hyödylliseksi vaikkakin haastavaksi ja stressiä tuottavaksi.

Viitteet:

A+ materiaalit,

www.programiz.com,

www.stackoverflow.com,

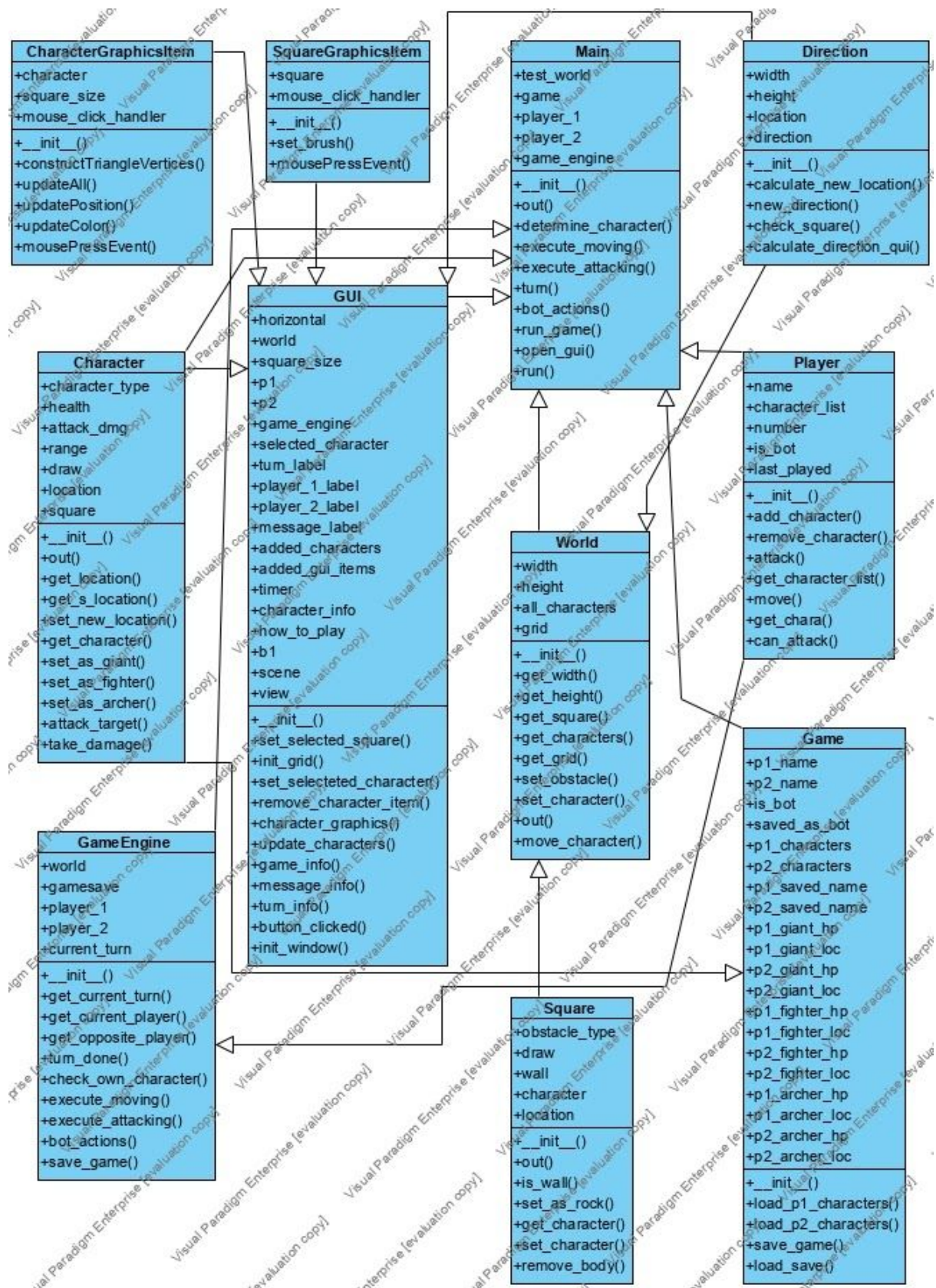
www.programtalk.com,

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

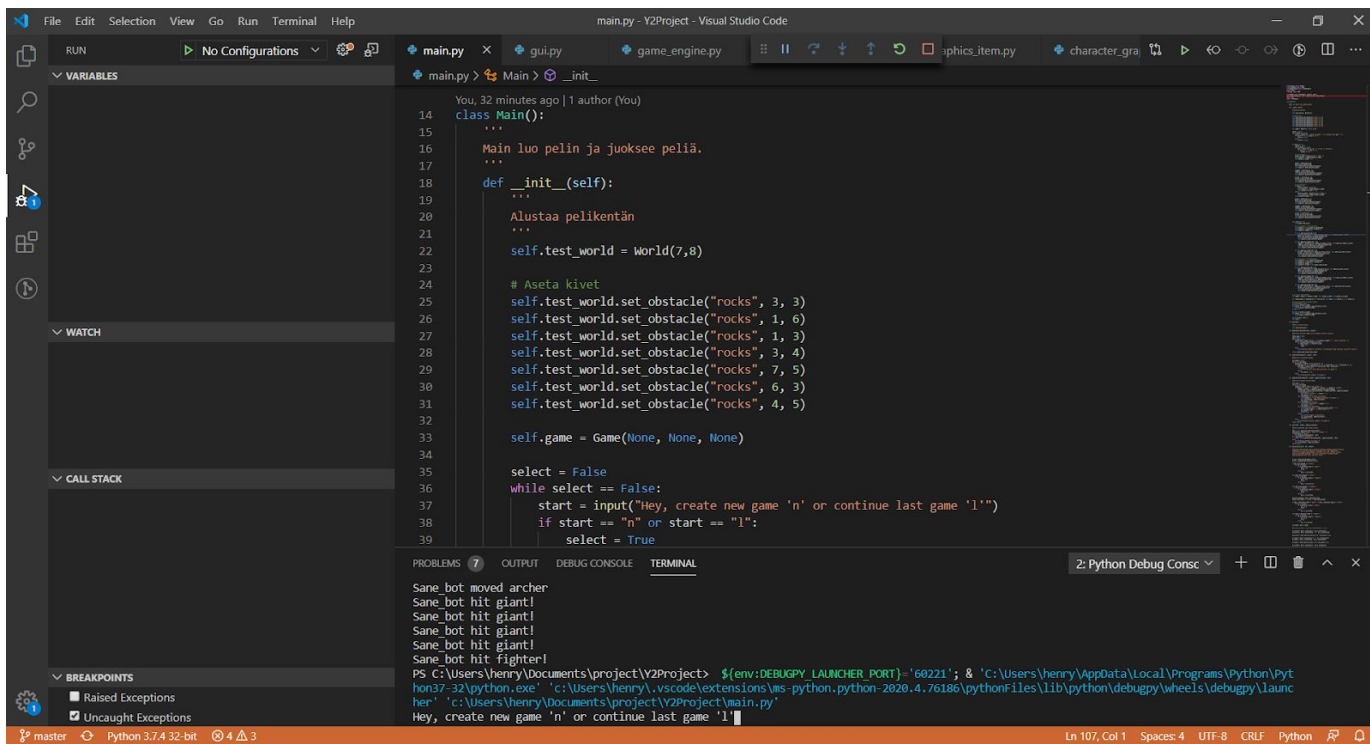
Liitteet:

Seuraavalla sivulla UML-taulukko luokista ja niiden funktioista.

Viimeisellä sivuilla on kuva ohjelman käynnistämisestä ja pelin ikkunan avautumisesta.



Ohjelma juostaan ja kirjoitetaan komentoriville ohjeiden mukaisesti.



```
main.py - Y2Project - Visual Studio Code
You, 32 minutes ago | 1 author (You)
class Main():
    """
    Main luo pelin ja juoksee peliä.
    """
    def __init__(self):
        """
        Alustaa pelikentän
        """
        self.test_world = World(7,8)

        # Aseta kivet
        self.test_world.set_obstacle("rocks", 3, 3)
        self.test_world.set_obstacle("rocks", 1, 6)
        self.test_world.set_obstacle("rocks", 1, 3)
        self.test_world.set_obstacle("rocks", 3, 4)
        self.test_world.set_obstacle("rocks", 7, 5)
        self.test_world.set_obstacle("rocks", 6, 3)
        self.test_world.set_obstacle("rocks", 4, 5)

        self.game = Game(None, None, None)

        select = False
        while select == False:
            start = input("Hey, create new game 'n' or continue last game 'l'")
            if start == "n" or start == "l":
                select = True

PROBLEMS 7 OUTPUT DEBUG CONSOLE TERMINAL
Sane_bot moved archer
Sane_bot hit giant!
Sane_bot hit giant!
Sane_bot hit giant!
Sane_bot hit giant!
Sane_bot hit giant!
Sane_bot hit fighter!
PS C:\Users\henry\Documents\project\Y2Project> $(env:DEBUGPY_LAUNCHER_PORT=-'60221'; & 'C:\Users\henry\AppData\Local\Programs\Python\Python37-32\python.exe' 'c:\Users\henry\.vscode\extensions\ms-python.python-2020.4.76186\pythonfiles\lib\python\debugpy\wheels\debugpy\launcher' 'c:\Users\henry\Documents\project\Y2Project\main.py')
Hey, create new game 'n' or continue last game 'l'
```

Tämän jälkeen peli-ikkuna aukeaa ja peliä voidaan pelata hiirellä klikkaamalla omaa hahmoa ja suuntaa mihin halutaan liikkua.

