Python Y2 Strategiapeli

Antti Matikainen, 353980, Automaatio- ja Systeemitekniikka, 2016 kevät

Osa 1: Yleinen kuvaus

Valitsin projektini aiheeksi pythonilla tehtävän strategiapelin. Inspiraationa, käytin elementtejä ns. tabletop strategiapeleistä, ja ruutupohjaisista strategiapeleistä. Käyttöliittymän toteutin käyttämällä Pygame:a. Strategiapeli sisältää graafisen käyttöliittymän, tekoälyn jota vastaan voi pelata, ja satunnaisesti luodut kartat. Pelissä on myös options menu, ja se tukee monta eri ikkunaresoluutiota.

Itse peli on toteutettu käyttämällä neljää eri looppia, ei mukaanlukien asetusvalikkoa. Ensimmäinen on itse päävalikko. Sitten on valikko, jossa valitaan pelin asetukset, esimerkiksi kartan koko ja monta pistettä pelaajilla on yksikköjen valintaa varten. Tämän jälkeen on valikko, jossa valitaan itse yksiköt. Viimeinen on itse pelilooppi, joka sisältää myös tekoälyn. Pelin hallinta perustuu puhtaasti hiiren käyttöön.

Tekoäly on toteutettu antamalla eri suunnille painoarvo, jonka avulla valitaan mihin suuntaan siirrytään. Painoarvo perustuu suuntaan missä pelaajan yksiköt on, ja kuinka paljon suojaa ruutu antaa.

Alkuperäisesti suunnitelmasta jouduttiin poistamaan joitain asioita, kuten kartta editori jossa pelaaja voi itse luoda kartan ja eri kartta tyypit. Muuten peli seuraa alkuperäistä suunnitelmaa.

Osa 2: Käyttöohje

Kun peli aloitetaan, avautuu päävalikko. Start Game painike siirtyy pelin setup menuun. Options painike antaa käyttäjän valita ikkunan koon, ja Quit Game painike sulkee pelin. Käyttäjä tekee valintansa painamalla nappia hiirellä.

Options valikon yläreunassa näkyy sen hetkiset asetukset. Käyttäjä voi muuttaa niitä painamalla esim. resoluutio nappia, tai FPS rajoitin nappia. Save Settings nappi tallentaa asetukset tiedostoon. Peli pitää käynnistää uudelleen tämän jälkeen. Main Menu nappi siirtyy takaisin päävalikkoon.

Start Game painike siirtää käyttäjän pelin setup osioon. Tässä vaiheessa valitaan kuinka monta pistettä kummallakin pelaajalla on yksiköiden valitsemiseen ja kuinka suuri pelikartta on. Ensiksi valitaan pistemäärä. Käyttäjä tekee tämän painamalla yhtä kuudesta painikkeesta. Tämän jälkeen pelaaja valitsee kartan koon. X = 10, Y = 8 on kartan minimikoko. Käyttäjä muuttaa kokoa käyttämällä +X / -X ja +Y / -Y painikkeita. Sen jälkeen kun pelaaja on valinnut kartan koon, hän lukitsee valinnan painamalla Confirm Map painiketta. Tämä aloittaa kartan satunnaisen luonnin. Kun tämä on tehty, ovat pelin asetukset valmiit, joten käyttäjä voi painaa Start Game nappia.

Tämän jälkeen pelaaja valitsee yksikkönsä. Tällä hetkellä valittavissa on kolme, mutta koska peli lukee yksikön tiedot .txt tiedostosta, on niiden lisääminen helppoa. Pelaaja painaa yhtä yksiköistä, joka avaa yksikön varustuksen valinnan. Kuten yksikköjä valittaessa, käyttäjä tekee valintansa painamalla yhtä menussa olevista painikkeista. Kumpaakin menua voi kelata käyttämällä ”Up” ja ”Down” nappeja. Sen jälkeen kun käyttäjä on valinnut varustukset, hän painaa ”Confirm Equipment” nappia. Yksiköllä on oltava yksi ase (ja niille ei voi antaa enempää), mutta muita varustuksia voi antaa niin monta kun tahtoo. Käytössä olevat pisteet näkyvät valikon yläreunassa. Sen jälkeen kun pelaaja on valinnut omat yksikkönsä, hän painaa ”Confirm Selections” painiketta. Sitten hän valitsee samalla tavalla yksiköt tekoälypelaajalle.

Kun pelin setup on valmista, siirtyy käyttäjä itse peli ruudulle. Karttaa voi siirtää käyttämällä ruudun ala-oikeassa reunassa olevia painikkeita. Aluksi, pelaaja asettaa yksiköt kartalle. Tämä

tehdään painamalla ruutua kartalla, ja sitten ruudun ala-laidassa olevaa yksikköä. Pelaaja tekee tämän jokaiselle yksikölleen, ja sitten painaa Next Player painiketta. Tämän jälkeen hän tekee saman tekoälyn yksiköille. Pelaaja voi asettaa yksikkönsä 4 ruutua kartan vasemmasta reunasta, ja tekoäly neljä ruutua kartan oikeasta reunasta. Sitten, kun kummankin pelaajan yksiköt ovat asetettu kartalle, alkaa itse peli.

Pelaaja voi valita yhden yksiköistään painamalla sitä kartalla, joka muuttaa sen kullanväriseksi että sen voi tunnistaa. Valitun yksikön statistiikat näkyvät ala-palkissa. Pelaaja voi siirtää yksikköä painamalla sen vieressä olevaa ruutua, ja sitten ”Move Selected Unit” painiketta. Hyökkääminen on mahdollista vaihtamalla hyökkäysmoodiin painamalla ”Attack Mode” painiketta. Painike pysyy vihreänä jos hyökkäysmoodi on päällä. Kun hyökkäysmoodin on päällä, hän voi painaa toista yksikköä, jolloin hän hyökkää sitä. Hyökkäysmoodin voi laittaa pois päältä painamalla samaa painiketta. Jokainen yksikkö voi hyökätä vain kerran per vuoro. Sen, onko yksikkö hyökännyt, näkyy sen info boksista kun se on valittu.

Sitten kun pelaaja on suorittanut vuoronsa, hän painaa ”End Turn” painiketta, jonka jälkeen tekoälypelaaja siirtää omat yksikkönsä. Jos toisella pelaajalla ei ole yhtään yksiköitä jäljellä, julistetaan toinen pelaaja voittajaksi.

Tietoa pelistä:

- Yksiköillä on seuraavat statistiikat:
- Nimi
- UnitID (Käytetään uusien yksikkö olioiden luontiin.)
- Cost (Hinta pisteissä.)
- UnitType
- HitPoints (Kuinka paljon elämää yksiköllä on vielä jäljellä.)
- Armor (Yksikön panssari.)
- MovementPoints (Yksikön maksimi movement point määrä. Movement pointteja käytetään liikkumiseen.)
- CurrentMovementPoints (Kuinka monta movement pointtia on jäljellä.)

Aseilla on seuraavat statistiikat:

- Nimi
- Cost (Hinta pisteissä.)
- OptimalRange (Kuinka pitkälle etäisyydelle ase tekee täyttä vahinkoa.)
- FalloffRange (Maksimi etäisyys jolle ase tekee vähennettyä vahinkoa.)
- Damage (Aseen tekemä vahinko.)
- ArmorPen (Kuinka paljon panssaria aseen tekemä vahinko läpäisee.)
-

Varustuksilla on seuraavat statistiikat:

- Nimi
- Cost (Hinta.)
- StatAffected (Mihin yksikön statistiikkaan varustus vaikuttaa.)
- Value (Kuinka paljon varustus laskee kyseistä statistiikkaa.)

Kartalla on neljänlaisia ruutuja:

- On keltaisia ruutuja, jotka ovat ”Open Terrain” ruutuja. Niihin liikkuminen vie yhden yksikön movement pointeista, ja se ei anna suojaa ruudussa olevalle yksikölle. (AccuracyModifier = 100)
- Vaalean vihreä ruutu on ”Light Terrain” ruutu. Siihen liikkuminen vie myös yhden movement pointin, ja se antaa pienen määrän suojaa yksikölle. (Accuracy Modifier = 75)
- Tumman vihreä ruutu on ”Heavy Terrain” ruutu. Siihen liikkuminen vie kaksi movement

pointtia, ja se antaa huomattavan määrän suojaa yksikölle. (Accuracy Modifier = 50)
- Tumman ruskea/harmahtava ruutu on ”Impassable Terrain” ruutu. Siihen ei voi liikkua, mutta sen yli voi ampua.

Hyökkäyksen tekemä vahinko saadaan käyttämällä seuraavaa kaavaa:

```
unmoddamage = abs(round(damage*((0.7 * accuracymodifier +
(random.random()*20))/100)))
damagedealt = max(0, unmoddamage - max(0, (TargetTile.UnitInSquare.Armor -
armorpen)))
```

Missä ”Accuracy Modifier” saadaan kohdeyksikön ruudun tyypistä, damage on hyökkääjän aseen vahinko attribuutti, armorpen on hyökkääjän aseen armor penetration attribuutti, ja Armor on kohdeyksikön panssari.

Osa 3: Ohjelman käyttämät luokat

Pelissä käytetään hyväksi monta eri oliota. Tässä on lista niistä ja niiden sisältämät attribuutit/funktiot:

- Game attribuutit:
  - Players (Sisältää listan pelissä olevista Player olioista)
  - HumanPlayerCount (Kuinka monta ihmispelaajaa pelissä on.)
  - AIPlayerCount (Kuinka monta tekoäly pelaajaa pelissä on.)
  - Map (Pelin kartta olio.)
  - Turncounter (Monesko vuoro on menossa.)
  - BaselineUnitArray (Array yksiköistä, jotka luetaan tiedostosta. Kun pelaaja valitsee yksikön armeijaansa, luodaan yksikkön näiden pohjalta. Näin tiedosto on luettava vain kerran.)
  - BaselineEquipmentArray (Sama kun BaselineUnitArray, mutta varustuksille ja aseille.)

- Game olion funktiot:
  - ReturnSpecificUnit(UnitID) (Palauttaa BaselineUnitArray:sta yksikön jolla on kyseinen UnitID. Käytetään kun sillon, kun pelaaja lisää yksikön armeijaansa.)
  - ReturnSpecificEquipment(EquipmentID) (Käytännössä sama, mutta aseille ja varustukselle.)
  - PopulateUnitArray() (Populoi BaselineUnitArray:n käyttämällä FileReader funktiota.)
  - PopulateEquipmentArray() (Käytännössä sama, mutta aseille ja varustuksille. Kutsuu myös FileReader funktiota eri muttujilla.)

- Player attribuutit:
  - Name (Pelaajan nimi.)
  - isAI (Onko kyseessä ihmispelaaja vai tekoäly.)
  - PlayerUnitList (Pelaajan yksiköt.)
  - Game (Game olio jonka osa pelaaja on.)
  - PointsAvailable (Kuinka monta pistettä pelaajalla on käytettävissä yksiköiden valitsemiseen.)
  - PlayerColor (Minkä värisiä pelaajan yksiköt ovat.)

- Player olion funktiot:
  - ReturnSpecificUnit(UnitID) (Sama kun Game oliossa, mutta käy läpi pelaajan PlayerUnitList arrayn.)

- Map attribuutit:
  - MapSize (Kartan koko.)
  - MapMatrix (Kahdesta arraysta muodostettu matriisi joka sisältää Tile olioita. MapSize määrittää koon.)
  - TerrainType (Kartan terrainin tyyppi. Ei käytetä tällä hetkellä mihinkään, koska en ehtinyt

implementoida eri tyyppejä. Ideana olisi ollut että pelaaja voi valita onko esim. kaupunki tai metsä.)
    - Game (Game olio)

- Tile attribuutit:
    - TileType (Ruudun tyyppi. Open/Light/Heavy/Impassable)
    - TerrainType (Feature jota en ehtinyt implementoida, samoin kun map funktiossa.)
    - Location (Missä ruutu on kartalla.)
    - UnitInSquare (Ruudussa oleva yksikkö.)

- Tile olion funktiot:
    - MoveUnit(TargetTile) (Siirtää ruudussa olevan yksikön TargetTile ruutuun.)
    - GetDistance(TargetTile) (Palauttaa etäisyyden kohderuutuun ruudusta jossa funktio kutsutaan.)

- Unit attribuutit:
    - OwningPlayer (Pelaaja joka omistaa yksikön.)
    - UniqueID (Yksikköä luodessa iteroitu numero jotta identtiset yksiköt voi erottaa toisistaan.)
    - UnitID (Ykskön UnitID. Käytetään siihen kun yksikköä luodaan BaselineUnitArray version perusteella.)
    - Name (Yksikön nimi.)
    - Cost (Yksikön hinta pisteissä.)
    - UnitType (Yksikön tyyppi (esim jalkaväki, ajoneuvo), pääasiassa vain flavoria, mekaaninen vaikutus ei implementoitu.)
    - HitPoints (Yksikön terveys.)
    - Armor (Yksikön panssari.)
    - MovementPoints (Yksikön maksimi movement points. Rajoittaa paljon yksikkö voi liikkua.)
    - CurrentMovementPoints (Yksikön sen-hetkinen movement point määrä.)
    - Equipment (Array joka sisältää yksikön aseet ja varustukset.)
    - UnitDeployed (Onko yksikkö asetettu pelikartalle.)
    - UnitColor (Minkä värinen yksikkö on pelikartalla.)
    - UnitCoordinates (Yksikön sijainti kartalla.)
    - HasAttacked (Onko yksikkö hyökännyt. Käytetään siihen että yksikkö ei voi hyökätä enempää kun kerran per vuoro.)

- Unit olion funktiot:
    - ReduceHitpoints (num) (Vähentää yksikön terveyttä.)
    - ReturnWeapon() (Palauttaa yksikön käyttämän aseen.)
    - EquipItem(Equip) (Jos on ase, lisää sen yksikön Equipment:iin. Jos on muunlainen varustus, , lisää sen, ja muuttaa attribuuttia varustuksen tyypin perusteella.)

- Unit olioon liittyvät funktiot, jotka eivät ole sidottu olioon:
    - CreateUnit(Player, UniqueID, UnitID) (Luo uuden yksikön BaselineUnitArray yksikön perusteella, jonka se sitten palauttaa.)
    - AttackUnit(OwnTile, TargetTile) (OwnTile ruudussa oleva yksikkö hyökkää TargetTile ruudussa olevaa yksikköä. Tehty vahinko perustuu kaavaan jonka selitin ”Tietoa pelistä” osiossa.)

Equipment olio koostuu kahdesta ala-oliosta, Weapon ja Gear:
- Weapon:
    - Name (Aseen nimi)
    - Cost (Aseen hinta pisteissä)

- EquipmentID (Käytetään weapon olio instanssin luomiseen BaselineEquipmentArray:n perusteella)
- Type (Onko kyseessä ase vai varustus. Käytetään pikku-väsäykseen.)
- OptimalRange (Etäisyys johon asti ase tekee täyttä vahinkoa.)
- FalloffRange (Etäisyys OptimalRange:n jälkeen jolla ase voi vielä tehdä vahinkoa, mutta tekee vähennetyn määrän.)
- Damage (Kuinka paljon vahinkoa ase tekee.)
- ArmorPen (Kuinka paljon kohdeyksikön panssaria ase läpäisee.)

-Gear:
- Name (Nimi)
- Cost (Hinta)
- EquipmentID (Sama kun Weapon oliossa.)
- Type (Sama kun Weapon oliossa.)
- StatAffected (Mihin yksikön attribuuteista varustus vaikuttaa.)
- Value (Paljon StatAffected muuttujan määräämää attribuuttia nostetaan.)

-Button/ButtonText. Button Text on olio, Button on funktio joka käyttää sitä. ButtonText ottaa sisään tekstiä, rect olion johon teksti sijoitetaan, värin, ja fontin koko. ButtonText luo tekstin, joka on asetettu kohde rectin keskelle. Tämän jälkeen Button käyttää kyseistä tekstiä ja piirtää napin ruudulle johon voi sitoa funktion. Voi myös käyttää siihen että piirretään boksi jossa on tekstiä. Muita käyttöliittymän elementtejä luodaan käyttämällä pygame:n Rect oliota.

Kumpikin Equipment olion ala-olioista luodaan käyttämällä CreateEquipment(Player, EquipmentID) funktiota, joka toimii samalla tavalla kun CreateUnit funktio.

Näiden lisäksi, pelissä käytetään seuraavaa funktioita joka ei liity mihinkään olioon.
- FileReader(filename, mode) (Lukee tiedoston, jonka nimi määritetään filename argumentissa. Mode argumentti kertoo mitä tiedostosta luetaan, ja missä formaatissa se palautetaan. Käyttää jsonreader:ia tämän mahdollistamiseksi.)

Osa 4: Ohjelman Rakenne:
Kun aloin tekemään projektia, ja harkitsemaan mitä luokkia tarvitsisin, aloitin pilkkomalla pelin sen komponentteihin. On itse peli. Pelissä on pelaajia. Pelissä on kartta. Kartta koostuu ruuduista. Ohjelman luokat on määritelty tätä philosophiaa käyttäen. Jokainen luokka on käytännössä kappale peliä. Luokan sisältämät attribuutit ovat määritelty ensin harkitsemalla mitä luokka tarvitsisi että se olisi kyseinen osa peliä. Tämän jälkeen joitain attribuutteja lisättiin tarpeen mukaan, kun paljastui että niitä tarvittaisiin.

Suhteet objektien välissä ovat hieman mielivaltaiset. Toisissa tapauksissa ne määriteltiin ohjelmoidessa kun paljastui että tarvitsisin jotain ohjelman toteuttamiseen. Halusin että pääsisin mistä vain käsiksi tarvittavaan tietoon. Tämän takia Player oliosta pääsee käsiksi Game olion, ja Unit olio sisältää linkin pelaajaan joka omistaa sen. Tein tämän sen takia, että ohjelmoinnin myöhemmissä vaiheissa ei ilmenisi ongelmaa, joka vaatisi huomattavia muutoksia luokkiin.

Jokaisen luokan tarkoitus on aika itsestäänselvä jo nimen perusteella. Yritin myös mahdollistaa sen, että tulevaisuudessa olisi mahdollista lisätä uusia toimintoja. Tämä onnistuu parhaiten jos ei ole liikaa asioita joita on muutettava.

Osa 5: Algoritmit
Pelin tekoäly algoritmi on hieman yksinkertainen. Se perustuu siihen, että ensin tarkistetaan tekoäly yksikön ympärillä olevien ruutujen tyypit. Näille annetaan painokerroin terrain tyypin perusteella. Ideana on että tekoäly haluaa asettua ruutuun joka antaa sille suojaa. Tämän jälkeen, tarkistetaan missä suunnassa pelaajan yksiköt ovat. Näiden perusteella yksikkö liikkuu johonkin suuntaan. Koska TileType on numero, painokerroin määrittyy 3-tile.Tiletype:n mukaan.

Tämän jälkeen, se hyökkää lähintä yksikköä, jotta kohteella on suurin todennäköisyys olla aseen optimaalisella rangella. Kyseessä on yksinkertainen toteutus, mutta se toimii. Siihen voisi implementoida osion, jossa se yrittää pysytellä tietyllä etäisyydellä pelaajan yksiköistä, mutta en ehtinyt implementoida sitä tavalla joka toimi.

Päädyin tähän toteutukseen ajattelemalla mitä tämänlaisessa vuoropohjaisessa strategiapelissä oleva tekoäly tekisi, ja pelkistin sitä hieman että voisin toteuttaa sen onnistuneesti. Tekoälyn perusta on toimiva, mutta yksinkertainen. Jos minulla olisi ollut enemmän aikaa, olisin lisännyt esimerkiksi sen, että tekoäly yrittää pysyä aseensa optimaalisella etäisyydella. Olisin myös muuttanut tekoälyä niin, että se katsoisi suunnassa olevien ruutujen tyyppiä 5 ruudun etäisyydelle. En kuitenkaan ehtinyt tehdä näin.

Osa 6: Tietorakenteet
Projektin suunnitteluvaiheessa, kun harkitsin mitä tietorakenteita käyttäisin, totesin että pyörän uudelleensuunnittelu ei ole kannattavaa. Tämän takia käytin pääasiassa olemassaolevia tietorakenteita.

Kun harkitsin eri vaihtoehtoja kartan toteuttamiseksi, ensimmäinen asia mikä tuli mieleen oli matriisi. Päädying tähän lopputulokseen koska matriisi on helppo tapa visualisoida kartta, Y kappaletta X levysiä rivejä. Se on myös hyvä tapa päästä käsiksi tiettyyn ruutuun jos tiedetään sen koordinaatit. Se on myös erittäin helppo toteuttaa, koska matriisi on vain kaksiulotteinen lista, joita python tukee erinomaisesti.

Kartassa käytetty matriisi on käytännössä monimutkaisin tietorakenne. Sen ulkopuolella käytin perus tupleja, stringejä, listoja ja niiden sekoituksia. Olisi ollut mahdollista käyttää eksoottisempaa tietorakennetta, mutta yksinkertaisuus on hyve.

Käytössä oli sekoitus muuttuvatilaisia ja muuttumattomia rakenteita. Joissain tapauksissa, esimerkiksi koordinaatti tupleissa, tämä ei ole ongelma. Jos koordinaatit muuttuu, korvataan tuple uudella. Se, onko tietorakenne muuttuvatilainen, ei käytännössä siis vaikuttanut valintoihin. Tietorakenteen valinta perustui siihen, mikä toimi hyvin ja oli yksinkertainen. Perusrakenteet toimivat hyvin tässä mielessä koska niiden käyttö on dokumentoitu erinomaisesti.

Osa 7: Tiedostot
Ohjelman FileReader funktio käsittelee json, eli javascript object notifier, formatoituja .txt tiedostoja. Kun suunnittelin ohjelmaa, harkitsin monta eri alternatiivista tapaa formatoida tiedosto. Sain kuitenkin vinkin tutuilta että json on erittäin hyvä vaihtoehto. Se on ihmisluettava, helppo käyttää, ja sen jäsentely on yksinkertaista. Tämä teki siitä ylivoimaisetsti käyännöllisimmän vaihtoehdon.

Näitä tiedostoja käytetään yksiköiden tietojen, varustusten tietojen ja pelin asetusten tallentamiseen. Vaikka yksiköt voisi iskeä suoraan python tiedostoon, tämä tekisi uusien yksikköjen lisäämisestä erittäin haastavaa. Koska yksiköt ovat tiedostossa, voi käyttäjä lisätä uusia erittäin helposti. Ihmisluotteavuus auttaa huomattavasti myös tässä mielessä.

Yksiköt käyttävät seuraavaa formaattia
{"Name": "Jager Light Inf", "ID": 1, "Cost" : 50, "Unit Type" : "Infantry", "Hit Points" : 20, "Armor" : 5, "Movement Points" : 3}

Varustukset käyttävät seuraavaa formaattia:
{"Name" : "Strider Movement Frames" , "Cost" : 20 , "ID" : 6, "Type" : "Gear" , "Stat Affected" : 3, "Value" : 2}
{"Name" : "Mark 3 Rifle", "Cost" : 0, "ID" : 1, "Type" : "Weapon", "Optimal Range" : 4 , "Falloff Range" : 8, "Damage" : 15 , "Armor Penetration" : 2}

Osioiden järjestys ei merkitse mitään, mutta on tärkeää että ID on uniikki, ja kaikki osiot ovat kuitenkin olemassa. Tällä hetkellä käytössä olevat tiedostot on lisätty koodin mukaan.

Osa 8: Testaus
Vaikka se on ehkä huono tapa, en käyttänyt diskreettejä yksikkötestaus settejä. Koska ohjelmassa käytettävät funktiot ovat yksinkertaisia, käytin prosessia jossa iskin testikoodia suoraan ohjelmaan. Kyseessä oli Test-driven developement periaatteen inspiroima prosessi. Käytännössä, kun implementoin funktiota tai toiminnallisuutta, ohjelmoin osan funktiota ja testikoodin joka testaa että ensimmäinen osio toimii. Sitten kun ensimmäinen osio toimii, jatkoin samaa prosessia kunnes koko funktio oli toteutettu. Ongelmana on kuitenkin se, että siistin testikoodin pois ennen kuin lisäsin koodin githubiin.

Esimerkiksi, kun implementoin ensimmäistä menua opetellessani pygame:n käyttöä, loin ensiksi laatikon. Tämän jälkeen lisäsin napit, sitten sain napit muuttamaan väriä. Tämän jälkeen lisäsin funktiot jotka printtasivat mitä nappi tekee. Lopulta sidoin funktion joka avaa pelin toisen osion loopin kyseiseen nappiin. Käytin samanlaista prosessia kun toteutin muita osioita projektista. Käytin myös TestiLaja.py:tä proof-of-concept koodi pätkien testaamiseen.

Ongelmana tässä kuitenkin on se, että testikoodi piti siistiä pois koska se teki koodin lukemisesta haastavampaa. Vaikka kaikki on testattu, ei testikoodista ole jäljellä tarpeeksi että testejä on mahdollista toistaa. Suunnitelmassa mainitsin käyttäväni perinteisiä yksikkötestaus tapauksia. Tajusin kuitenkin projektin aikana että itse funktiot eivät olleet monimutkaisia, vaan toiminnallisuus tuli siitä miten ne sitoutuivat yhteen. Tämän testaaminen perinteisillä testitapauksilla olisi ollut haastavaa.

Osa 9: Ohjelman puutteet ja viat
Suurin ongelma projektissa on tekoäly. En ennen projektin alkua ollut edes harkinnut miten tekoäly pitäisi toteuttaa. Tämä, ja ajan vähyys, johti siihen että tekoäly on erittäin yksinkertainen. Vaikka se toimii, en ehtinyt toteuttaa läheskään kaikka mitä olin suunnitellut sen suhteen. Esimerkiksi, optimaalisesti tekoäly yrittäisi hyökätä pelaajan yksikköjä jotka eivät ole suojassa, tai ottaisi huomioon ruudut jotka ovat sen liikkumisetäisyydessä, mutta eivät sen vieressä. Se voi myös joissain edge-case tapauksissa jäädä jumiin. Jos minulla olisi enemmän aikaa, tämä olisi ensimmäinen asia jota parantaisin.

Toinen ongelma jota en ehtinyt korjata on ohjelman käytettävyys. Vaikka kaikki toimii, on yksiköiden siirtäminen ja niillä hyökkääminen erittäin kompelöä. Ideaalisesti, jos pelaajalla olisi yksikkö valittuna, hän voisi vain painaa kartta ruutua oikealla painikkeella ja siirtyä siihen. En kuitenkaan keksinyt miten saisin tämän toteutettua järkevästi. Ohjelman käytettävyys on numero kaksi listalla asioita joita tahtoisin korjata.

Koodin luettavuus on myös ongelma. Tällä hetkellä jotkin toiminnallisuudet toteutettiin viime hetkellä, ja koodin parantelu saattaisi aiheuttaa ongelmia joita en ehdi korjata. Varsinkin teko-äly osio on parantelun tarpeessa. Myös se että kaikki perustuu funktiossa julistettuihin funktioihin on ongelma. Tämä tekee tiedoistoista tarpeettoman pitkiä. En kuitenkaan onnistunut keksimään tapaa toteuttaa toiminnallisuus tekemättä tätä. Jos minulla olisi ollut enemmän aikaa, olisin voinut parannella koodia.

Itse peliä voisi myös parannella. Tällä hetkellä, peli on erittäin yksinkertainen ja sen tasapainoa ei edes ole harkittu. Jos minulla olisi ollut enemmän aikaa, olisin voinut tasapainottaa yksiköitä ja varustuksia. Olisin myös tehnyt niin että yksiköiden tyyppi vaikuttaa siihen miten ne liikkuvat, ja implementoinut lentäviä yksiköitä.

On myös mahdollista että on yksittäisiä tapauksia jotka saattavat kaataa pelin jotka eivät ilmenneet testaus prosessissa. Olen yrittänyt korjata niitä sitä tahtia mitä assistentti on niitä löytänyt, mutta epäilen että on joitain joita en ole vielä huomannut.

Osa 9: 3 parasta ja 3 heikointa kohtaa
Jos kysytään ohjelman vahvuuksista, olen ylpeä siitä miten hyvin sain kartan toimimaan. Se tukee eri kokoja, ja scrollaus toimii erinomaisesti. Olen myös sitä mieltä että eri ruudunkokojen tuki on hyvin onnistunut. Ainoa miinus on että liian pienet ruudut voivat aiheuttaa ongelmia. Olen myös ylpeä siitä että onnistuin toteuttamaan projektin sellaisella tavalla että lisätoiminnallisuus on helposti toteutettavissa.

Heikoimmat kohdat ovat selitetty hyvin edellisessä osioissa.

Osa 10: Poikkeamat sunnitelmasta ja aikataulu
Seurasin projektin suunnitelmaa aika läheisesti. Jo suunnitteluvaiheessa totesin että ei kannata iskeä mitään kiveen. On vanha lausahdus joka kuvaa tilannetta erinomaisesti. ”No plan survives contact with the enemy.” Olin varma että jos sunnittelen tekeväni jotain tietyllä tavalla joutuisin muttamaan sitä. Tämän takia harkitsin suunnitellessani eri vaihtoehtoja. Jouduin muuttamaan joitain yksittäisiä funktioita ja en ollut ottanut huomioon pygame:a, mutta muuten seurasin suunnitelmaa.

Myös aikataulu oli tarkoituksella epämääräinen. Suunnittelin vain missä järjestyksessä toteuttaisin eri osiot. Ainoa ongelma jonka tämä aiheutti oli se, että koska pygame osio vei huomattavasti enemmän aikaa kun suuunnittelin. Tämän takia tekoälyn toteuttamiseen jäi vähemmän aikaa kun olin suunnitellut. Se on yksi syistä miksi se on yksi pelin heikoimpia kohtia.

Osa 11: Arvio lopputuloksesta
Kun ottaa huomioon kaiken mitä olen tämän raportin aikana maininnut, tulin lopputulokseen että projekti oli yleisesti keskiverto. Siinä on hyvät puolensa, ja huonot puolensa. Jotkin osat on toteutettu hyvin, toiset eivät. Kuten mainitsin osassa 8, varsinkin tekoäly on puutteellinen. Peli ei myöskään ole erityisen käyttäjäystävällinen. Olen kuitenkin iloinen siitä miten sain kartan, sen skrollauksen ja sen skaalattavuuden toimimaan oikein. Vaikka lopputuloksesa on selvästi viat, voin ylpeästi sanoa että se on minun.

Osa 12: Lähteet

1. Python 3.5 dokumentaatio
2. Sekalaisia Stackoverflow kysymyksiä. Esimerkiksi miten etsiä listasta tuple jolla on pienin arvo, ja muita yksittäisiä pikku juttuja joista en ollut varma.
3. Pygame dokumentaatio. http://www.pygame.org/docs/
4. Pygame peli tutoriaali: http://programarcadegames.com/
5. Pygame nappi tutoriaali: https://pythonprogramming.net/pygame-buttons-part-1-button-rectangle/
6. TextCenterer koodi perustuu tähän stackoverflow postiin: http://stackoverflow.com/questions/32673965/pygame-blitting-center