CS-A1121 Projektidokumentaatio, Otto Huttunen, 1019113, Energia- ja konetekniikka, 10.5.2023

1. Yleiskuvaus

Projektissa on luotu vuoropohjainen strategiapeli, jossa vastustajana on älykäs tietokonevastustaja. Inspiraation lähteenä projektissa on käytetty Civilization-pelisarjan pelejä ja sen ominaisuuksia. Tavoitteenani oli luoda maailma, joka perustuu neliönmuotoisiin palasiin (tile) ja luoda maailma siten, että siihen on helppo lisätä eri mahdollisuuksia.

Loin useampia erilaisia käytettäviä hahmoja (unit) ja lisäsin näille eri ominaisuuksiksi liikkua, hyökätä ja tietylle hahmolle mahdollisuuden rakentaa kaupunkeja. Tässä kohtaa projektia on eriydytty suunnitelmasta, koska aikarasitteiden takia en ehtinyt luoda enempää ominaisuuksia hahmoille, tosin hahmojen pohjat on luotu siten, että uusien ominaisuuksien lisääminen on helppoa. Loin pelikartan siten, että ne muodostuvat satunnaisesti. Tavoitteena oli myös lisätä peliin karttaeditori, eli pelaaja voi itse luoda oman pelikartan, mutta tähän tavoitteeseen ei päästy, vaan kartan muokkaaminen onnistuu vain tallennustiedostoista.

Projekti on toteutettu alkuperäistä suunnitelmasta eroten keskivaikeana muiden kouluun liittyvien aikarasitteiden takia.

2. Käyttöohje

Ohjelma käynnistetään kansioiden ulkopuolella olevalla main2.py tiedostolla. PyCharmissa ohjelma avautuu myös game-kansiossa olevalla main.py tiedostolla, mutta se ei toiminut PyCharmin ulkopuolella.

Aluksi valitset, aloitatko pelin vai jatkatko vanhaa. Tämän jälkeen annat ohjelmalle nimen ja kartan koon. Ohjelma luo sinulle aloitushahmon vasempaan yläkulmaan.

Tavoitteesi on tuhota vastustajan kaikki kaupungit. Tässä apuna on hahmot, jota voit luoda kaupunkeja klikkaamalla sekä juuri hahmot, joilla voit hyökätä ja jotka omaavat eri ominaisuuksia. Nämä ominaisuudet ovat nähtävissä ja muokattavissa configuring.txt tiedostossa.

Kartassa on eri laattoja, joista veden ja vuorten laatalle ei voi liikkua. Laatat antavat bonuskertoimen hahmojen puolustukselle ja nämäkin ovat nähtävissä konfigurointitiedostossa.

Voit liikkua ja hyökätä kullakin hahmolla kerran vuorossa. Kun olet tehnyt haluamasi asiat, klikkaa ’next turn’ – painiketta, jonka jälkeen voit taas liikkua. Oikealla tapahtumaikkunassa näet pelissä tapahtuvat asiat.

3. Ulkoiset kirjastot

Ainoa käytetty ulkoinen kirjasto on PyQt6.

4. Ohjelman rakenne

Ohjelma koostuu seuraavista rakenteista: peliä hallinnoivat tiedostot, hahmot, laatat ja hahmojen graafiset tiedostot. Hallinnoivat tiedostot ja graafiset tiedostot ovat samassa kansiossa alussa tehdyn virheen takia, enkä lähtenyt korjaamaan tätä virhettä, koska projektia tehdessä sain huonoja kokemuksia tiedostojen siirtämisestä kansioista toisiin.

Peliä hallinnoiviin tiedostoihin kuuluvat kaikki game-alkavat .py tiedostot, map.py, player.py, ai_player.py, ai.py sekä gui.py ja startwindow.py. Kaksi viimeistä hallinnoivat grafiikkaa, mutta tekevät lisäksi paljon muuta ohjelman eteen.

Game.py tiedostossa oleva olio Game hallinnoi koko peliä. Siihen kytketään kaikki osatekijät eli karttaolio Map, pelaajaolio Player, grafiikkaolio Gui sekä AI pelaaja Aiplayer sekä AI:n aivot eli AI. Game-olio hallinnoi pelimaailman luomista ja pitää kontrollissa kaikkia peliin liittyvää. Game oliota viittaamalla pääsee käsiksi oikeastaan pelin koodissa, minne tahansa, mikä on elintärkeää koodin toimivuuden kannalta. Game-olio vastaa myös pelin hahmojen luonnista.

Map olio vastaa nimensä mukaisesti pelin kartan hallinnoimisesta sekä luomisesta. Sen vastuualueelle kuuluu myös tietää, onko joku hahmo liikkumassa tai hyökkäämässä, joten se hallinnoi mahdollisia hahmojen toimintoja.

Game_configuring.py tiedostossa on Configure-olio, joka vastaa pelin konfigurointitiedoston luvusta ja tallentaa itseensä kaikki tiedot. Tämä olio tallennetaan Game-olioon, jota viittaamalla ohjelman kaikki osat löytävät omat konfigurointitietonsa.

Lopusta liitteistä löytyy näennäinen UML-malli.

5. Algoritmit

Ohjelma käyttää laskentakaavaa hahmojen tekemän vahingon laskemiseen, oman ja vihollisen etäisyyden laskemiseen sekä AI:n liikkumissuunnan määrittämiseen.

Viholliseen tehty vahinko lasketaan kaavalla:

```
damage = int(30 * math.exp(((combat_strength1 - (combat_strength2 *
defence_bonus)) / 25)  * (random.randint(75, 125) / 100)))
```

, jossa damage on vihollisen kokema vahinko, combat_strength1 on hyökkääjän voimakkuus, combat_strength2 on puolustajan voimakkuus sekä defence_bonus on puolustajan laatan antama kerroin puolustajan voimaan. Lisäksi vahinkoon liittyy satunnaiskerroin, jonka suuruus on 0.75–1.25. Tämä algoritmi on otettu Civilization 6 pelistä (lähde löytyy viitteistä).

Oma ja vihollisen välinen etäisyys lasketaan kaavalla:

```
distance = abs(self.coordinates[0] - city.coordinates[0]) +
abs(self.coordinates[1] - city.coordinates[1])
```

, jossa lasketaan siis oman ja vihollisen x- ja y-suuntaisten koordinaattien erotuksen itseisarvot ja summataan ne yhteen. Tämä aiheuttaa sen, että esimerkiksi hahmo, jonka kantama on 1, niin voi hyökätä vain suoraan oman itsensä viereen tai yläpuolelle, mutta ei diagonaaliin. Algoritmi on valittu siksi, että se tekee ohjelmasta simppelin.

Lisäksi ohjelma käyttää algoritmia AI:n ohjaaman hahmon suunnan valintaan. AI:n ohjaama hahmo liikkuu siten, että se testaa ensin ylempänä annetulla algoritmilla, mikä on lähin vastustajan hahmo tai kaupunki. Tämän jälkeen algoritmilla lasketaan, onko x- vai y-suunnassa pidempi matka vastustajan luo ja hahmo liikkuu sen akselin suuntaisesti, millä on pidempi matka. AI liikkuu niin pitkään vastustajaa kohti, kunnes on hahmon kantaman sisällä, jonka jälkeen tämä hyökkää. Samaa algoritmia on käytetty myös ns. pakenemissuunnan määrittämiseen, jossa ohjelma toimii vain päinvastaisesti. Tätä käytetään AI:n Settler-hahmojen menemistä määritellyn matkan päähän AI:n kaupungista.

Esteen kohdatessa AI kokeilee ensin, 90 asteen käännökset molempiin suuntiin ja tämän jälkeen liikkuu taaksepäin. Varmasti netistä olisi löytynyt parempi polunlöytöalgoritmi, mutta halusin tehdä algoritmin itse. Kuitenkin osuin umpikujaan tässä kohdassa, koska AI jäi liikkumaan edestakaisin näissä tilanteissa. Ratkaisuksi tein sen, että Settlerien kohdalla noin joka viides liike on satunnaisgeneroitu, jolloin ne eivät jumitu ns. algoritmin mahdollistamaan vankilaan.

6. Tietorakenteet

Ohjelmassa tieto on tallennettu eri olioiden ominaisuuksiin. Yleensä ottaen nämä ovat pelkkiä muuttujia, mutta joukossa on myös muutama lista ja matriisi. Kaikki ohjelmassa tallennettu tieto on melkeinpä muuttuvatilaisia, koska useimpia tietorakenteita pitää muokata ohjelman suorituksen aikana.

7. Tiedostot

Ohjelma käyttää pelin tallentamiseen ja konfigurointiin samanlaista tietorakennetta, joka on muokattu versio tehtäväkierrosten Human Writeable File-tehtävästä. Valinta osui tähän formaattiin siksi, että sitä on lukijan helppo ymmärtää ja etenkin konfiguroinnissa tärkeää. Valitsin formaatin myös siksi, koska osaan mielestäni hyvin ohjelmoida formaatin kirjoittamisen sekä lukemisen.

Formaatti käyttää siis erikoismerkkiä (ohjelman tapauksessa #) ja sen jälkeistä tekstiä datatyypin tunnistamiseen (esim. #Game Info). Ohjelma lukee tämän otsikon, jonka jälkeen ohjelmalle on määritetty, mitä dataa tämän ns. otsikon jälkeen pitää tulla. Luku on ohjelmoitu joko siten, että se lopettaa otsikon alaisen datan luvun, kun tarvittavat tiedot on saatu tai silloin, kun seuraava #-alkuinen otsikko esiintyy tiedostossa. Tallennustiedostot ovat helposti käsiteltävissä ja niitä voi muokata vapaasti testiskenaarioiden luomiseen.

8. Testaus

Testasin ohjelman alussa pelin taistelusysteemiä, jotta se oikeasti tappaisi hahmot ja nämä katoaisivat pelistä. Ohjelman loppuvaiheessa nämä testit eivät toimi enää, koska ohjelma tarvitsee paljon lähtöarvoja toimiakseen, joten niiden testaaminen yksikkötestien avulla olisi erittäin hankalaa. Suurinta osaa ohjelmaa olen itse käsin testannut antamalla sille eri lähtöarvoja ja koittanut kaataa peliä. Näissä kohdin, kun ongelma on tullut vastaan, niin olen korjannut sen.

Ohjelmasta löytyy yksikkötestit tallennustiedostojen virheilmoituksille sekä konfigurointitiedoston virheilmoituksille. Konfigurointitiedoston testaukset ovat omissa kansioissaan, koska ajava tiedosto ottaa

configure.txt aina siitä kansiosta, missä ajava tiedosto sijaitsee, jolloin jouduin tekemään useita eri kansioita eri testien tekemistä varten.

9. Ohjelman tunnetut puutteet ja viat

Testautin peliä parilla kaverillani ja he kommentoivat, että pelissä on hankala havaita, minkä hahmon olet valinnut sekä kuinka suuri kantama hahmolla on. Tämän korjaisin enemmällä ajalla, että toisin graafisesti näkyvästi, minkä hahmon olen valinnut sekä kuinka pitkälle hahmo voi liikkua tai hyökätä. Graafiseen käyttöliittymään lisäisin helpottavia ominaisuuksia, jottei esimerkiksi tallennustiedoston nimeä tarvitsisi joka kerta erikseen kirjoittaa.

Ohjelmasta puuttuu varsinaiset tehdyt kartat, vaan omasta päätöksestäni tein ohjelman siten, että joka peli olisi erilainen ja kartta muodostettaisiin satunnaisgeneraatiolla ja eri laattojen painotuksia voi muokata konfiguraatiotiedostossa. Karttaa pystyy itse kyllä muokkaamaan tallennustiedostossa, mutta se on hankalaa. Tähän olisin luultavasti siirtynyt seuraavaksi, jos minulla olisi ollut enemmän aikaa projektin tekoon.

Ohjelma ei siis ole niinkään puutteellinen tai viallinen, mutta se on aika raakile sekä yksinkertainen. Isommalla aikamäärällä olisin saanut ohjelmaan enemmän sisältöä ja pelikokemusta miellyttävämmäksi. Lisäksi peli ei ole ns. tasapainotettu eli en ole pelannut peliä sen verran, että olisin löytänyt hyvää balanssia hahmojen voimasuhteisiin, mutta tätä pystyy jälkeenpäin muokkaamaan konfigurointitiedostossa.

Näiden lisäksi ohjelman generoimat settlerit muodostuvat ala- ja yläkulmiin, jolloin ne voivat syntyä vahingossa veteen tai vuorelle, mihin ne eivät muuten pääsisi.

10.3 parasta ja 3 heikointa kohtaa

Mielestäni AI:n liikkuminen (ai.py, rivit 118-248) on samaan aikaan ohjelmani yksi parhaista koodinpätkistä, mutta myös samalla yksi heikkouksista. Parasta siksi, koska keksin itse kyseisen menetelmän ja sain sen kirjoitettua ohjelmaan vaivattomasti. Heikkous siksi, että olen varma, että asian olisi voinut lukea netistä, jolloin lopputulos olisi luultavasti ollut nopeampi, parempi ja lyhyempi.

Hyvää koodia mielestäni ovat myös game_init.py ja game_saver.py sekä game_configuring.py. Kaikki nämä käsittelevät mielestäni tiedostoja hyvin, koodinpätkät ovat simppeleitä ja ne toimivat heti sekä skaalautuvat hyvin. Esimerkiksi oma läppärini jaksaa pyörittää 1000x1000 karttaa sekä tallentamaan, että käynnistämään kyseisen kartan eli ohjelma pystyy yhtä hyvin tallentamaan 25 laattaa kuin miljoona laattaa (tosin latausaika on hieman eri).

Lisäksi yleisesti koko gui.py on mielestäni hyvin toteutettua koodia. Se hallinnoi selkeästi grafiikkatavaroita ja koodia oli ohjelmaa tehdessä helppo muokata jälkeenpäin.

Huonoja kohtia ohjelmassa on turhat tiedostot ja yleinen järjestys. Kaikki laatat olisi voitu laittaa tile.py tiedostoon, koska esim. water.py sisältää huimat yhdeksän riviä koodia. Havahduin asiaan kuitenkin sen verran myöhään, että en enää uskaltanut ohjelman toimivuuden ja mahdollisen lisätyömäärän takia alkaa siirtämään luokkia yhteen tiedostoon. Sama koskee myös kaikkia hahmojen tiedostoja. Kaikki hahmojen grafiikat pitäisi olla omassa kansiossa, mutta tein niitä yksitellen game-kansioon, jolloin havahduin tähänkin liian myöhään ja edellä mainituista syistä en lähtenyt muokkaamaan.

Järjestyksessä epäonnistuin silloin, kun aloin vain tekemään enkä ollut suunnitellut järjestystä enempää. Suuremmalla aikamäärällä puuttuisin tähän suuremmin.

11.Poikkeamat suunnitelmasta

Alkuperäisestä suunnitelmasta poikettiin poistamalla hahmoista Builder sekä Galley. Galley (vene) tippui hahmoista siksi, että kartat toteutettiin satunnaisgeneraatiolla, jolloin suuria vesialueita ei tullut, jolloin Galley muuttui turhaksi. Sen lisääminen olisi tullut ajankohtaiseksi, kun karttaeditori olisi tuotu ohjelmaan. Builderin otin pois siksi, että tässä pelin vaiheessa, jossa kaupungit tuottavat rahaa ja muut ns. ominaisuudet esim. liikkuminen on ohjelmoitu pelaajan ja laatan välisen etäisyyden kautta esimerkiksi teiden luominen olisi ollut turhaa. Builderin lisääminen tulee ajankohtaiseksi, kun peliin lisättäisiin lisää algoritmeja.

Ajankäytöstä poikkeamista käsittelen seuraavassa palautuksen kohdassa enemmän.

12.Toteutunut työjärjestys ja aikataulu

Pvm	Käytetty aika (h)	Mitä toteutettu?
22.2	3	Projektisuunnitelma, Tile- ja Unit-luokkien raamit
6.3	2	Kartta-olion teko, GUI-olion teko, pelin kartta satunnaisgeneraatiolla näkyviin
20.3	6	Ensimmäinen UnitGraphicsItem, hahmojen liikkuminen kartalla mahdolliseksi
3.4	2	Lisää GraphicsItemejä hahmoille, kaupunki ja sille mahdollisuus luoda hahmoja
4.4	2	GUI:hin lisää hienosäätöä
5.4	1	Archerin GraphicsItemin luonti
13-14.4	2	READ.ME päivitys sekä testien päivitys
26.4	3	Kaikki GraphicsItemit tehty, AI:lle luotu pohja
27.4	2	Bugien korjailua / hahmot eivät kadonneet kuollessa
2.5	3	Bugien korjailua / hahmot eivät vieläkään katoa, hahmojen rakennuksen bugeja
3.5	6	Bugien korjailua, tehty aloitusvalikko, pelin pystyy nyt tallentamaan ja aloittamaan tallennuksesta
4.5	1	Bugien korjailu, pelille voittoikkuna
5.5	5	AI:n kokonaisvaltainen luominen
8.5	3	Konfigurointitiedosto sekä poikkeusten luonti
9.5	3	Konfigurointiominaisuuksien ajo tiedostoihin, testit konfiguroinnille ja tallennuksille
10.5	4	Bugien korjausta ja yleistä siistimistä
11.5	4	Dokumentaatio ja palautus
Yht.	52	

Todellinen ajankäyttö on noin puolet pienempi kuin projektisuunnitelman ja se johtuu kahdesta asiasta. Ensinnäkin ammuin projektisuunnitelmassa aikataulutuksen erittäin yläkanttiin ja aliarvioin omat kykyni. Ammuin yläkanttiin, koska minulle kerrottiin, että yhteensä projekti on erittäin haastava ja aikaa vievä (mitä se kyllä oli). Toiseksi olen joutunut karsimaan hieman suunnitelman ominaisuuksia, koska minulla on ollut

valtava työmäärä muihin kouluhommiin projektin ulkopuolella ja tämä on hellittänyt vasta 5. periodissa, jonka huomaa siitä, että yli puolet projektiin käytetyistä päivistä on 4. periodin tenttiviikolla ja sen jälkeen.

Projektisuunnitelmassa olin luonut eräänlaisen työjärjestyksen ja tekemäni työ jollain tavalla noudatteli tätä. Alussa tein ohjelmaa ilman grafiikkoja, mutta tämän jälkeen päätin, että alan luomaan ohjelmaa siinä järjestyksessä, missä alkaisin pelata sitä. Eli ensin pelin grafiikat ja kartta avautuisi, jonka jälkeen tekisin kaupungin jne. Näin ohjelmoin pelin loppuun asti lisäten ominaisuuksia yksi kerrallaan, jonka jälkeen pelistä tulisi enemmän pelattava.

13.Arvio lopputuloksesta

Ohjelma on mielestäni mukavan simppeli ja oikeilla (eli pelikokemuksen kautta testatuilla) asetuksilla sitä on varmasti mukava pelata. Sain mielestäni hyvin toteutettua alkuperäistä visiotani, vaikka jäikin siitä hieman puutteelliseksi ajan puutteen takia. Ohjelman koodi on mielestäni hyvää ja sen laajentaminen ei ole hankalaa. Loin kaiken ohjelmaan liittyvän logiikan ja koodin itse ja mielestäni onnistuin hyvin siinä. Onnistuin myös hyvin tämänhetkisten asetusten lisäämisen konfigurointitiedostoon ja sen laajentaminen on aika helppoa, joten laajennukset ja niiden konfiguroinnin lisäys kävisi käden käänteessä.

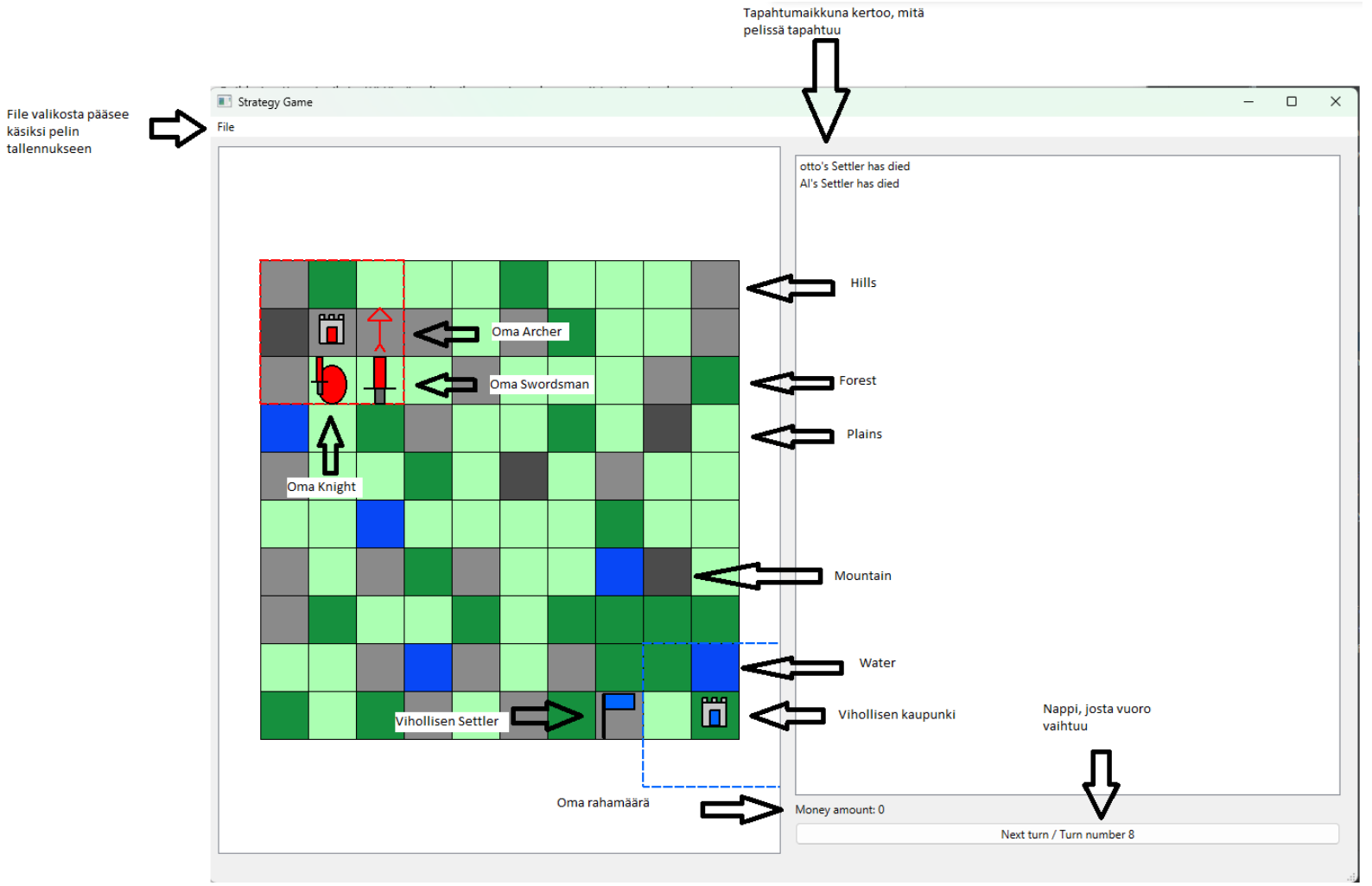
Graafinen käyttöliittymä on ihan ok, mutta se tarvitsisi silti paljon parannuksia, joita käsittelin ohjelman tunnetuissa vioissa ja puutteissa.

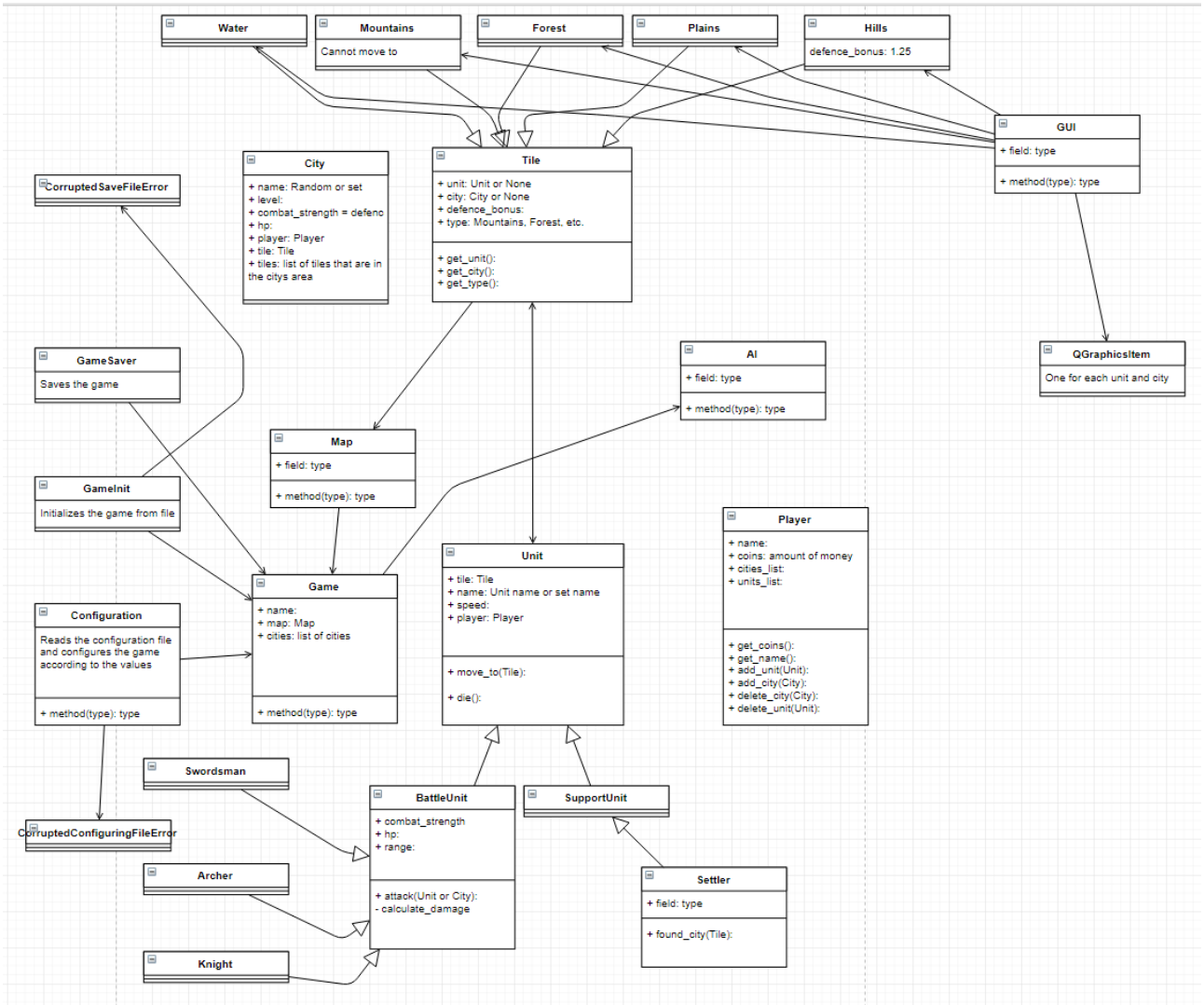
Käsittelin 3 parasta ja 3 heikkoutta kohdassa kansioiden epäselkeyttä ja se jäi minua vaivaamaan, mutta koitin kerran hieman muokata järjestystä ja siitä tuli pelkkää sotkua, joten kyseiset huonot kokemukset siinä vaikuttivat ohjelman repositoryn muotoon. Muutenkin jäi hieman harmittamaan, etten pystynyt luomaan kaikkea mitä halusin, mutta lopputulos on mielestäni erittäin onnistunut, etenkin, kun tein suurimman osan ohjelmasta 4. periodin jälkeen. Tästä kurssista voisi olla hauska jatkokurssi, että tällä kurssilla tehtyjä ohjelmia laajennettaisiin ja niistä tehtäisiin tehokkaampia.

14.Viitteet

Civilization 6 Vahinkosysteemi:
https://www.reddit.com/r/civ/comments/amighb/civilization_vi_damage_formula_and_graph/

15.Liitteet


Tapahtumaikkuna kertoo, mitä pelissä tapahtuu
File valikosta pääsee käsiksi pelin tallennukseen
Strategy Game
File
otto's Settler has died
Al's Settler has died
Oma Archer
Oma Swordsman
Oma Knight
Hills
Forest
Plains
Mountain
Water
Vihollisen Settler
Vihollisen kaupunki
Nappi, josta vuoro vaihtuu
Oma rahamäärä
Money amount: 0
Next turn / Turn number 8


Water
Mountains
Cannot move to
Forest
Plains
Hills
defence_bonus: 1.25
GUI
+ field: type
+ method(type): type
City
+ name: Random or set
+ level:
+ combat_strength = defenc
+ hp:
+ player: Player
+ tile: Tile
+ tiles: list of tiles that are in the citys area
Tile
+ unit: Unit or None
+ city: City or None
+ defence_bonus:
+ type: Mountains, Forest, etc.
+ get_unit():
+ get_city():
+ get_type():
CorruptedSaveFileError
GameSaver
Saves the game
AI
+ field: type
+ method(type): type
QGraphicsItem
One for each unit and city
Map
+ field: type
+ method(type): type
GameInit
Initializes the game from file
Player
+ name:
+ coins: amount of money
+ cities_list:
+ units_list:
+ get_coins():
+ get_name():
+ add_unit(Unit):
+ add_city(City):
+ delete_city(City):
+ delete_unit(Unit):
Unit
+ tile: Tile
+ name: Unit name or set name
+ speed:
+ player: Player
+ move_to(Tile):
+ die():
Game
+ name:
+ map: Map
+ cities: list of cities
+ method(type): type
Configuration
Reads the configuration file and configures the game according to the values
+ method(type): type
Swordsman
BattleUnit
+ combat_strength
+ hp:
+ range:
+ attack(Unit or City):
- calculate_damage
SupportUnit
CorruptedConfiguringFileError
Archer
Settler
+ field: type
+ found_city(Tile):
Knight