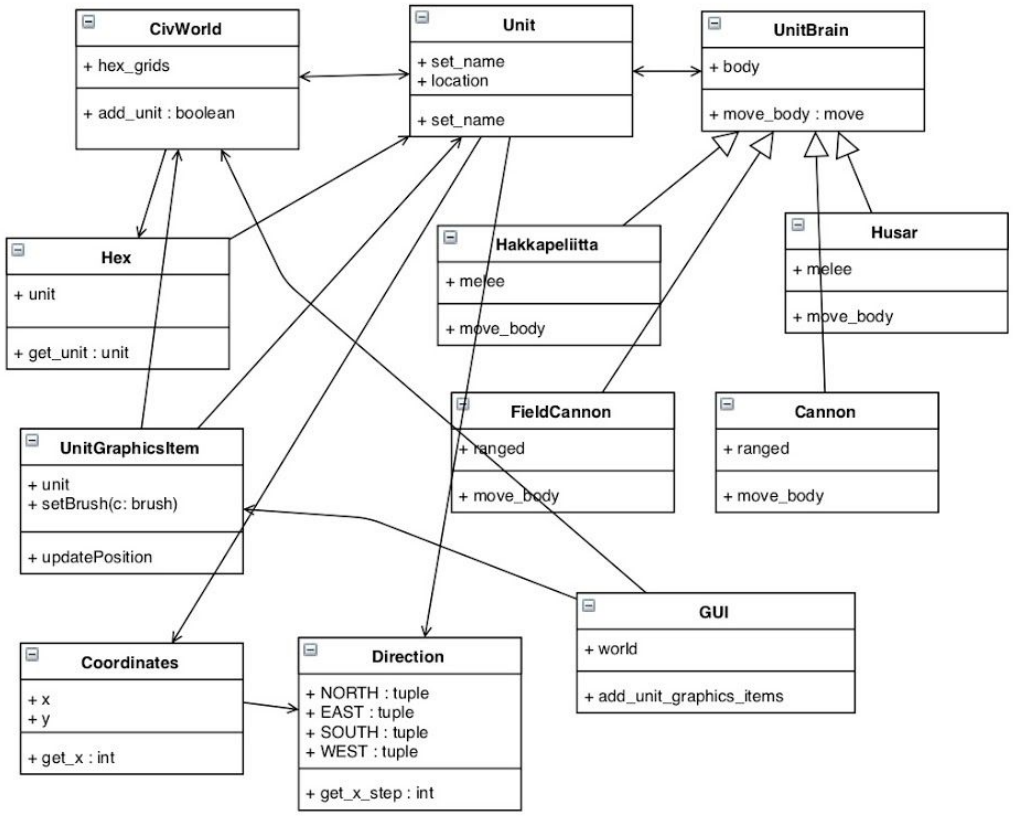
Tekninen suunnitelma
“Sweden at the Gates of Vienna”
Y2, kevät 2020

Karri Liikkanen
790 158
ENY
1. vuosikurssi
6.3.2020

1 Ohjelman rakennesuunnitelma

Sovellan ohjelmassa rakenteena apuna kurssilla ollutta robottimaailmaa, sillä ohjelmani on hyvin samankaltainen. Uskoisin, että luokkia kannattaa olla melko runsaasti. Yksiköille tehdään oma pääluokka ja oma “aivo”luokka, jonka itse yksikköluokat perivät. Kts. Kuva 1.



Kuva 1: UML-kaavio

2 Käyttötapauskuvaus

Käyttäjä käynnistää ohjelman. Hän kohtaa aloitusruudun, jossa kerrotaan tekstillä pelin taustatarina. “On vuosi 1632. Ruotsi on Lützeniin hyökkäämisen sijaan marssinut suoraan pedon sydämeen, Wieniin, jumalattomien Habsburgien siveettömään katoliseen pääkaupunkiin. Vain Ferdinand II:n paikallinen varuskunta husaareja on puolustamassa koko maailman merkittävintä kaupunkia Ruotsin suurvallan mahdilta… jne.” Käyttäjä päätyy sen jälkeen valikkoon, josta hän voi valita puolensa: Ruotsi tai Habsburgit. Ohjelma asettaa tässä vaiheessa pelaajalle arvoksi tämän valinnan ja vastaavasti AI:lle arvoksi toisen vaihtoehdon. Valinnan jälkeen käyttäjä päätyy itse peliin.

GUI latautuu ja näyttää pelaajalle pelikentän (kts. Projektisuunnitelma, Kuva 4). Pelaaja valitsee seuraavaksi jonkin yksiköistä, jota hän käyttää. Hän valitsee tämän jälkeen yksikölle joko liikkumisen (move) tai hyökkäämisen (attack). Taustalla ohjelma kutsuu oikeita metodeja, esim. hakkapeliitta.lyoMiekalla ja sen jälkeen damagea antavia metodeita ja tallentaa uudet arvot olioihin, jotka olivat kyseessä. Pelaajan jälkeen on AI:n vuoro. Tekoälyä voisin ajatella pystyväni tekemään ainakin sen verran, että jos yksikkö saavuttaa tietyn damage-tason (esim. alle 20% healthia jäljellä), niin se ei enää hyökkää vaan linnoittautuu ja alkaa healata. Myös toiset yksiköt voisivat pyrkiä tällöin “auttamaan” haavoittunutta. Nämä toimisivat koodissa jonkinlaisten if, else -rakenteiden takana.

Peli loppuu joko siihen, että voittoehto (melee-yksikkö vastustajan kuninkaan viereen) täyttyy tai siihen, että pelaaja lopettaa pelin. Pelaajan voittaessa hänet palkitaan fanfaarilla ja tekstillä, joka kertoo joko, että Ruotsi on pelastanut maailman katolisuuden ikeestä tai että Habsburgit ovat voittoisia ja saavat jatkaa irstasta toimintaansa. Tämän jälkeen peli lopettaa toimintansa.

3 Algoritmit

Onnekseni voin käyttää apuna myös Y1-kurssilla tekemääni CivilizationUnit -ohjelmaa, jossa harjoiteltiin juurikin hyökkäys- ja heal-algoritmien tekemistä. Vaikein osuus on varmasti graafisen käyttöliittymän rakentaminen. GUIn rakentemiseen löytyy toki runsaasti ohjeita netistä (esim. https://youtu.be/3li1SdZ1Ru8 – HUOM! Careless Whisper alkutunnarina!)

4 Tietorakenteet

Yksikköjen tiedot tallennetaan luokkamuuttujiin, joten en tiedä kuinka laajoja tietorakenteita varsinaisesti tarvitaan.

5 Kirjastot

Käytetään PuQt-kirjastoa, joka kurssilla mainittiin sallituksi projektissa. Tarvitaan nimenomaan graafisen käyttöliittymän toteuttamisessa.

6 Aikataulu

Tästä päivästä on aika tarkalleen 2 kuukautta aikaa tehdä projektia. Uskoisin, että GUI on vaikein osuus, joten sen hiominen jätetään loppuun hyväksyen se riski, että ulkoasu ei välttämättä tule olemaan kovin hieno. Ihan ensimmäiseksi haluan tehdä yksikköluokat, koska niiden toiminta on minulla uskoakseni tutuinta ja siten saan koodin aloitettua ja pääsen

testaamaan sitä. Ne on lisäksi hyvä olla valmiina kun rupeaa rakentamaan CivWorld-luokkaa, jossa käytetään yksiköitä.

Viikko 1: Suunnittelua ja ensimmäisten koodien kirjoittamista, yksikköluokkien luonti
Viikko 2: Testien kirjoittamista ja tärkeimpien metodien tekemistä, CivWorld-luokan luonti
Viikko 3: Uusien luokkien lisäämistä, Hex-luokka, jolloin pääsee tekemään GUIta
Viikko 4: Luokkien laajentamista, testaamista
Viikko 5: Ensimmäinen yritys saada ohjelma pyörimään
Viikko 6: Ohjelman pitäisi pyöriä karvalakkimallina
Viikko 7: Hiomista, bugien korjaamista, GUIn hiomista
Viikko 8: Palautus, loppupaniikki

7 Yksikkötestaussuunnitelma

Keskityn erityisesti alussa yksikköjen metodeihin (lyoMiekalla, ammuTykilla), jotta saan yksiköt pelaamaan keskenään yhteen. Tästä edetään testaamaan itse Unit-luoka metodeita, esim. set_name. Testaamista helpottaa se, jos pyrkii koko ajan menemään testaaminen edellä (TDD).

8 Kirjallisuusviitteet ja linkit

Kuten edellä todettiin, erityisesti GUIn tekemiseen löytyy todella runsaasti hyvän oloista materiaalia. En ehtinyt hirveästi vielä tutustua potentiaaliseen kirjallisuuteen, mutta eiköhän sitä löydy!