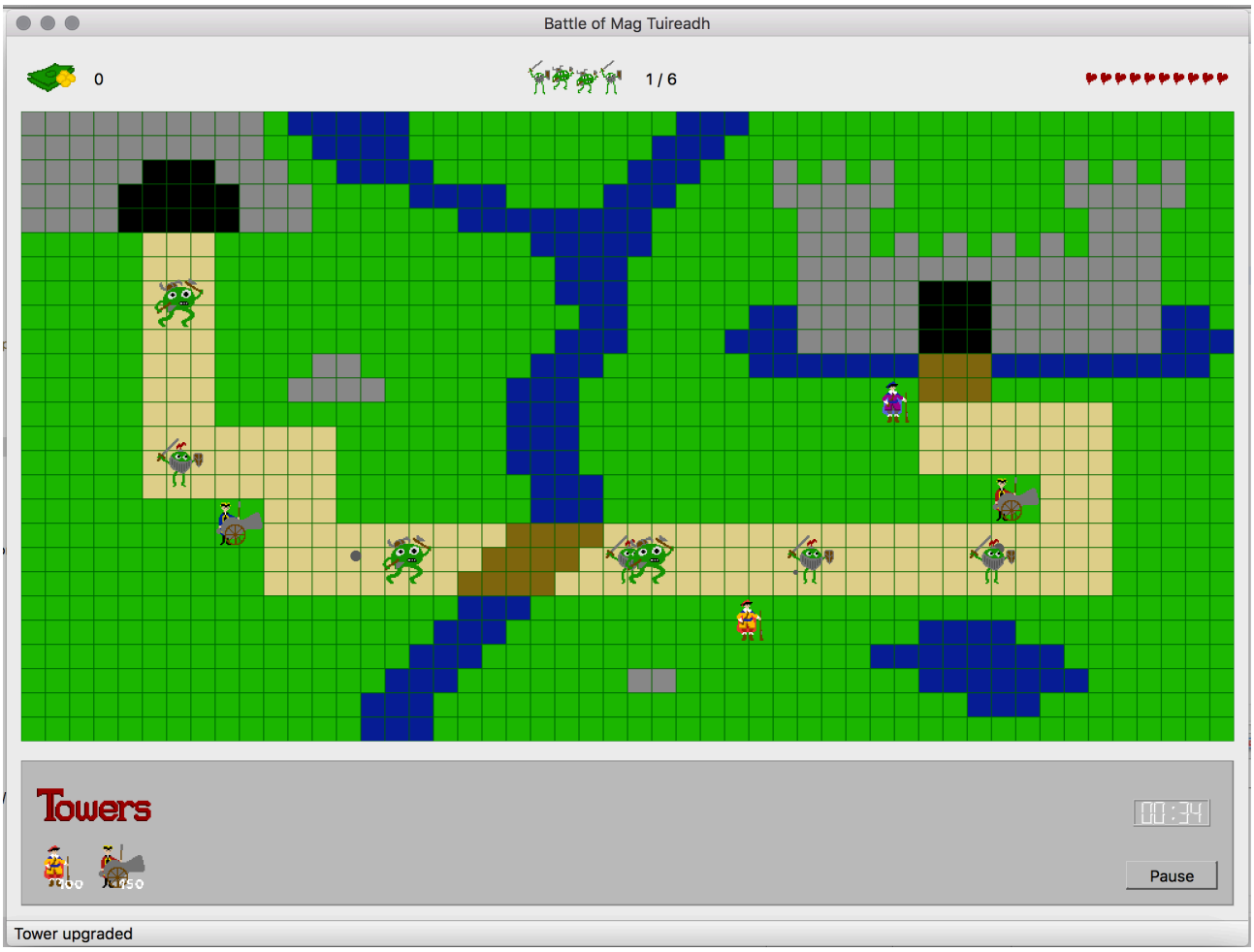
Tornipuolustuspeli

CSE-A1121 Ohjelmoinnin peruskurssi Y2

Riia Luhtala
461801
Information and Service Management (MSc)
Aalto BIZ, 2014
26.4.2016

1. Yleiskuvaus

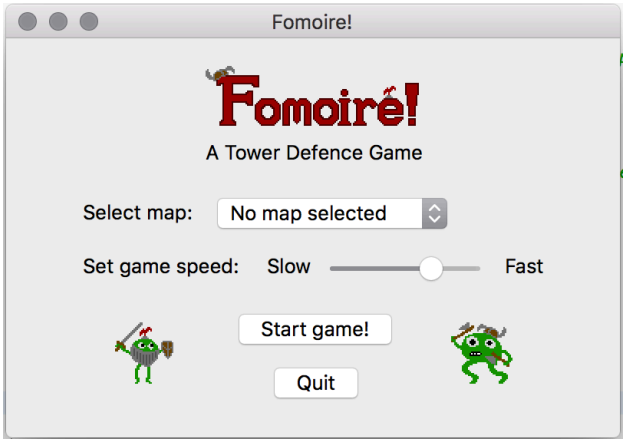
Projektin aiheena oli tornipuolustuspeli ja tavoitteena oli tehdä melko perinteinen ja simppeli versio. Mutta niin että sitä olisi helppo lähteä laajentamaan ainakin karttojen ja uusien torni- ja vihollistyyppien osalta. Pelissä viholliset kulkevat ennalta määrättyä reittiä kohti määränpäätään ja pelaajan tavoitteena on estää heitä pääsemästä sinne rakentamalla vihollisia ampuvia torneja reitin varrelle. Pelaaja voi rakentaa torneja vain kartan vihreälle alueelle. Torneja on kahdenlaisia ja ne voidaan kertaalleen päivittää, jolloin tornin attribuutit paranevat ja sen ulkoasu muuttuu. Tykkitorni ampuu hitaammalla tahdilla, mutta pidemmälle tehden vahinkoa kaikkiin ruudussa oleviin vihollisiin. Muskettitorni taas ampuu nopeammalla tahdilla, mutta lyhyemmälle ja satuttaen vain yhtä vihollista kerrallaan. Vihollisia on myös kahdenlaisia ne eroavat toisistaan nopeuden ja kestävyyden osalta. Kuvassa 1 on nähtävissä yleinen pelinäkymä. Arvioisin että oma toteutukseni on jossain keskivaikean ja vaikean välillä. Tiedän että parannettavaa olisi vielä rutkasti, mutta en koe myöskään menneeni sieltä missä aita on matalin.



Kuva 1 Pelinäkymä (Kartta 4)

2. Käyttöohje

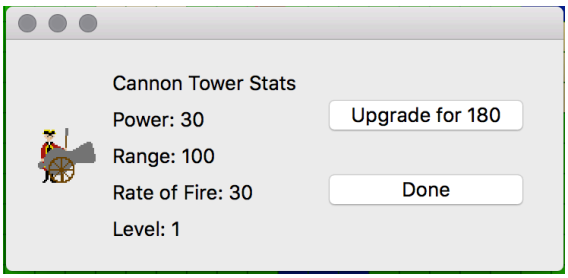
Ohjelma käynnistetään launcher.py-tiedostosta. Tällöin avautuu aloitusvalikko (Kuva 2). Pelaaja voi valita pudotusvalikosta haluamansa kartan. Valikosta löytyvät kaikki Maps-kansiossa olevat tiedostot. Tämän lisäksi hän voi säätää pelinopeuden haluamakseen.



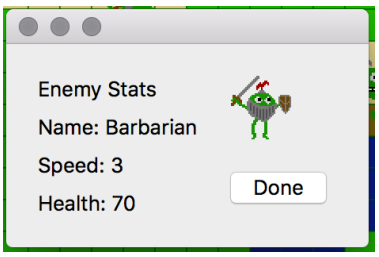
Kuva 2 Aloitusvalikko

Torneja rakennettaan klikkaamalla pelinäkymän vasemmassa alareunassa olevia nappeja. Rakentaminen onnistuu vain kun peli ei ole pysäytetty tai loppu ja pelaajalla on tarpeeksi rahaa. Tämän lisäksi torni täytyy rakentaa kartan vihreälle alueelle. Mikäli pelaaja yrittää rakentaa tornia esimerkiksi tien päälle, ohjelma ilmoittaa hänelle että tämä ei ole sallittua. Pelaaja myös näkee valitun tornin ääriviivat ja ampumissäteen kartalla (Kuva 6). Klikkaamalla hiiren oikeaa nappia tornivalinta peruuntuu. Pelaaja voi myös vaihtaa valintaa klikkaamalla toista torninappia. Yläpalkissa näkyy vasemmalla pelaajan käytettävissä olevat rahat, keskellä vihollisaallon numero ja aaltojen kokonaismäärä ja oikealla jäljellä olevat elämät. Pelin voi hetkellisesti keskeyttää painamalla Pause-nappia. Kulunut aika näkyy sen yläpuolella olevassa laskurista.

Tornien päivittäminen tapahtuu klikkaamalla niitä. Tällöin aukeaa ponnahdusikkuna, jossa näkyy tornin tämän hetkiset attribuuttien arvot ja upgrade-painike (Kuva 4). Jos muutat mieltäsi, paina Done-nappia (jos ikkunan sulkee vasemmasta ylänurkasta ohjelma kaatuu). Mikäli torni on jo maksimitasolla sitä ei voi enää päivittää. Myös vihollisten tietoja pääsee tarkastelemaan klikkaamalla niitä (Kuva 5).



Kuva 4 Tornin tiedot



Kuva 5 Vihollisen tiedot

Kuva 6 Tornien rakentaminen

3. Ohjelman rakenne

UML-kaavio ohjelmasta löytyy liitteistä (Liite 1). Siitä on jätetty pois kaikki setterit ja getterit, mutta voitte olettaa että sellaiset on asetettu melkeinpä jokaiselle ohjelmasta löytyvälle attribuutille.

Launcher-luokka avaa ohjelman ja sitä kautta määritellään UserInterface-luokan tarvitsemat attribuutit: selectedMap ja gameSpeed. StartGame-metodi luo uuden UserInterface-olion, jonka alla itse pelinäkymä luodaan. Ensimmäiseksi Userinterface luo uuden GameBoard-olion ja kutsuu tälle metodia readMapData. Tällä saadaan määriteltyä pelin aloitusasetukset. Tämän jälkeen UserInterface luo GameStats- (pelinäkymän yläpalkki), MapView-(keskellä oleva karttanäkymä) ja BottomButtons-oliot (pelinäkymän alapalkki). GameStats-näyttää pelin kautta oleelliset tiedot: käytettävissä olevat rahat, vihollisaallot ja jäljellä olevat elämät. BottomButtons taas käy läpi GameBoardin towersAvailable-listan ja luo käytettävissä oleville torneille ostonapit (BuyButton-luokka). Itse peli pyörii aika pitkälti MapView:n kautta. Se muun muassa kutsuu vihollisia, ampuu torneilla ja liikuttaa vihollisia ja ammuksia. Voisi ajatella että GameBoard alustaa pelin tallentaen sen tiedot ja attribuutit, mutta MapView muokkaa ja käyttää näitä attribuutteja ja piirtää pelinäkymän pelaajalle. Iso osa MapView:n metodeista voisi hyvin olla myös GameBoardin alla.

Tower-luokka kuvaa torneja. Sillä on tornityyppien mukaiset alaluokat. Tornin fyysistä ilmentymistä kartalla kuvaa ClickableTower-luokka. Se on ”painike,” jonka pelaaja asettaa kentälle. Jokaiseen Tower-luokan olioon liittyy periaatteessa yksi ClickableTower-olio. Tapa jolla tornien rakennus on toteutettu aiheutti sen että tornit jakautuvat tällä lailla kahteen luokkaan. Todennäköisesti ostometodia hiomalla yksi torniluokka olisi riittänyt, mutta päätin antaa asian olla ja keskittyä muiden ominaisuuksien kehittämiseen, koska nykyinenkin tapa toimi. Enemy-luokkaa ei tarvinnut sammallailla jakaa kahtia. Se pohjautuu aika pitkälti ClickableTower-luokkaan. Enemyn alta löytyy myös alaluokat eri vihollistyypeille. Projektiileille on myös oma luokkansa Projectile, joka jakautuu alaluokkiin ammustyyppien mukaan.

4. Algoritmit

Yritän tässä käydä läpi luokkien kannalta olennaisimmat algoritmit. Mitään hirvittävän monimutkaisia en ohjelmassani ole käyttänyt.

4.1. GameBoard

ReadMapData-metodi käy karttatiedoston rivi riviltä läpi ja asettaa GameBoardin attribuutit. Tämä on muunnelma kierrokselle 4 tekemästäni shakkipelinlukumetodistani. SortEnemyPath-algoritmi lajittelee luetut vihollispolun koordinaatit oikeaan järjestykseen asettaen annetut aloituskoordinaatit listan ensimmäiseksi olioksi ja järjestäen sen perusteella loput.

4.2. MapView

MapView:n alta löytyvät periaatteessa kaikki peliä pyörittävät metodit. UserInterface-luokan timerEvent kutsuu näitä, niin että ensin luodaan mahdolliset uudet viholliset, sen jälkeen kaikkia vihollisia liikutetaan, sitten tornit ampuvat (eli luodaan uudet projektiili-objektit) ja lopuksi ammuksia liikutetaan. Hieman yksinkertaistettuna MapView-luokan metodit siis vain käyvät objektilistoja (luodut tornit, viholliset ja projektiilit) läpi ja kutsuvat halutuille objekteille niiden omia metodeja.

Koska karttanäkymä on ruudukko, on calculateClosestCorner-algoritmi tornien asettelun kannalta elintärkeä. Se laskee jakojäännösten avulla hiiren osoitinta lähinnä olevan ruudukon kulman. Alla olennaisin osuus metodin koodista (Python on itsessään jo melkein pseudokoodia)(Kuva 7).

```
if mouse_x % blockSize < blockSize / 2:
    closest_corner_x = mouse_x - (mouse_x % blockSize)
else:
    closest_corner_x = mouse_x + (blockSize - mouse_x % blockSize)

if mouse_y % blockSize < blockSize / 2:
    closest_corner_y = mouse_y - (mouse_y % blockSize)
else:
    closest_corner_y = mouse_y + (blockSize - mouse_y % blockSize)
```

Kuva 7 calculateClosestCorner-algoritmi

4.3. Enemy

Vihollisten reitti määräytyy enemyPath-koordinaattilistan perusteella. MoveEnemy-metodi pitää huolen vihollisten liikuttamisesta. Vihollispolun seuraavan ruudun sijainti suhteessa nykyiseen kertoo mihin vihollinen on seuraavaksi menossa. Mutta suunta vaihtuu, mikäli sen tarvitsee vaihtua, vasta sen jälkeen kun vihollinen on päässyt nykyisen ruudun keskipisteeseen.

4.4. Tower

InRange-metodi laskee Pythagoraan lausetta hyväksikäyttäen (verraten tornin ja vihollisen x- ja y-koordinaatteja) onko vihollinen tornin hyökkäyssäteen sisällä.

4.5. Projectile

Projektiilit seuraavat niiden kohdevihollisia. Vihollinen liikkuu ensin ja sen jälkeen projektiili siirtyy kohti vihollisen nykyistä sijaintia kunnes se jossain vaiheessa saavuttaa tämän (ammusten nopeus on aina vihollisten nopeutta suurempi). Suunta lasketaan siis jokaisella liikkumiskerralla uudestaan. Tässäkin käytetään Pythagoraan lausetta.

5. Tietorakenteet

Ainoa tietorakenne mitä ohjelmani käyttää on lista. Niin erilaisten karttaelementtien koordinaatit, sallitut tornityypit, vihollisaaltojen tiedot kuin luodut tornit, viholliset ja projektiilit tallennetaan listoihin. En tiedä ovatko ne tilankäytön tai tehokkuuden kannalta kaikkein optimaalisimpia, mutta tunsin listoihin liittyvät metodit hyvin ja koin että listat olivat helppo ja toimiva ratkaisu. Minulla ei oikeastaan edes ole kokemusta vaihtoehtoisista toteutustavoista.

6. Tiedostot

Karttatiedostot on tallennettu tavallisina tekstitiedostoina (.txt), koska kenellä tahansa pitäisi olla osaamista ja ohjelma niiden luomiseen. Lisäksi kurssilla oli opeteltu tekstitiedostojen lukemista, joten se tuntui myös siinä mielessä järkevältä vaihtoehdolta.

Pelissä käytetyt kuvat ovat kaikki Portable Network Graphics -tiedostoja (.png), koska ne mahdollistavat kuvan osittaisen läpinäkyvyyden (näin esimerkiksi viholliskuvan tausta ei peitä pelikenttää).

6.1. Karttatiedostojen luominen

Tiedosto on jaettu kolmeen eri osaan, jotka erotetaan otsikoilla #mapinfo, #waves ja #map. Ensin annetaan kartan perustiedot (Kuva 7). Tiedot kannattaa kirjoittaa mahdollisimman samassa muodossa kuin esimerkissä, vaikka karttatietojen lukumetodi ei kaikissa tapauksissa välitäkään siitä, jos jossain on iso kirjan tai ylimääräinen väli. Kokoon pitää ilmoittaa kartan leveys ja korkeus merkkeinä (tässä järjestyksessä) x:llä erotettuna. Kartan täytyy olla vähintään 25 merkkiä leveä ja 7 merkkiä korkeaa ja enintään 70 merkkiä leveä ja 28 merkkiä korkea. Sen jälkeen annetaan aloituselämät (väliltä 1-20) ja –rahat (vähintään 100). Sitten

listataan käytettävissä olevat tornit (t1 = musketti, t2 = kanuuna) ja polun aloituskoordinaatit x, y –muodossa. Jos polku alkaa kartan vasemmasta ylänurkasta koordinaatit olisivat 1, 1. Eli y:n arvot kasvavat alaspäin. Olisi ehkä parempi, jos tämä toimisi intuitiivisemmin ja y:n arvot nousisivat alhaalta ylöspäin.

```
#mapinfo
name: Battle of Mag Itha
size: 50x26
lives: 20
money: 1000
towers: t1, t2
path start: 6, 4
```

Kuva 7 Kartan perustiedot

Tämän jälkeen määritellään vihollisaaltojen tiedot (Kuva 8). Ensin annetaan aaltojen määrä. Sen jälkeen niistä kukin listataan omalle rivilleen. Ensimmäinen luku kertoo vihollisten ilmestymisintervallin aallon sisällä. Sen jälkeen on listattu viholliset (e1 = barbaari, e2 = berserkeri). Tässä pitää käyttää aina pilkkua välimerkkinä, välilyönti ei ole pakollinen.

```
#waves
number of waves: 6
50, e1, e1, e1, e1, e1, e1
30, e1, e1, e2, e1, e1, e2
20, e1, e2, e2, e1, e1, e1, e2, e2
10, e2, e2, e1, e1, e1, e2, e2, e1, e1, e1, e2
10, e1, e1, e1, e1, e2, e1, e2, e1, e1, e2, e1, e2
10, e1, e1, e1, e2, e1, e2, e2, e2, e2, e2, e2, e2, e2
```

Kuva 8 Vihollisaaltojen määrittely

Lopuksi ”piirretään” itse kartta (Kuva 9). Alta löytyvät käytettävissä olevat symbolit ja niiden merkitykset. Vihollispolun täytyy olla yhtenäinen ja haaraton ja polun seuraavan pisteen täytyy olla aina joko pysty- tai vaakasuunnassa polun aikaisemman pisteen vieressä. Esimerkki karttatiedostosta löytyy kokonaisuudessaan liitteistä (Liite 2).

Karttasymbolit ja niiden merkitykset

P = vihollispolku tiellä (keltainen)
0 (nolla) = tie/hiekka (keltainen)

D = vihollispolku vedessä (sininen)
R = vesi (sininen)

B = vihollispolku sillalla (ruskea)
W = puu (ruskea)

Q = vihollispolku luolassa (musta)
C = luola/sisäänkäynti (musta)

M = vuori (harmaa)
X = vihollispolku vuorella (harmaa)

\+ = ruohikko (vihreä)

```
#map
RRRRRRRRRRRRRRRRRRR000000000RRRRRRRRRRRRRRRRR+++++
RRRRRRRRRRRRRRRRR0000+++000000RRRRRRRRRRRMRRMM++++
RRRRRRRRRRRRRRR++++++++++++000000000000+M++MM++++
RRRRRRRRRRRRRR+++M++++++++++++++000++++MM+MM++M+
+RRRRRRRRRRR+++++MMMMM+M+++++++++++++++++MMMMM+M+
++RRRDRRRRR00+++++MMWCMMM+++++M++++++++++MMMMMMM+
++000P000000++++++MMWWMMMM++MMM++++++++++MMMMMM++
+++00P0000++++++0000000000000000++++++++++MMMMMM++
++++0P0+++++++++0PPPPPPPPPPPPPPP0+++++++++MMMMMM++
++++0P0+++++++++0P000000000000P0+++++++++MCCMMMMM+
++++0P00000000000P0++++++++++0P0+++++++++MCCMMMMM+
++++0PPPPPPPPPPPPP0++++++++++0P0++0000000MCCMMMMM+
++++000000000000000++++0000000P0++0PPPPPPPPQMMMMM+
++++++++++++++++++++++0PPPPPPPP0++0P00000000MMMMM+
RRR+++++++++++++++++++0P0000000++0P0+++++++MMMMM+
RRRRR+++++++++++++++++0P0+++++++0P0+++++++MMM++
+RRRRRRRRRR++++++++++0P0+++++++0P0++++++++++++
+++RRRRRRRRR+++++++++WWB0W+++++++WBW++++++++++++
++++++++RRRRRRRRRRRRRRWRDRRRRRRRRRRWBWRRRR+++++++++
++++++++++++RRRRRRRRRRRRBRRRRRRRRRWBWRRRRRR+++++++
++++++++++++++++++++++WW+0PW+++++++WBW+++RRRRR+++++
++++++++++++++++++++++0P0+++++++0P0++++RRRRRRRRR
++++++++++++++++++++++0P0000000000P0+++++RRRRRRRR
++++++++++++++++++++++0PPPPPPPPPPPP0++++++++++++++
++++++++++++++++++++++0000000000000++++++++++++++
++++++++++++++++++++++++++++++++++++++++++++++++++
++++++++++++++++++++++++++++++++++++++++++++++++++
```

Kuva 9 Kartta merkein piirrettynä

7. Testaus

Testasin ohjelmaa pääasiassa manuaalisesti. Graafinen käyttöliittymä oli mukana alusta asti ja metodeja ja luokkia koodatessani mietin heti myös sen miten ne tulevat näkymään pelaajalle. Tällöin minun oli helppo testata koodini toimivuus ohjelmaa pyörittämällä ja nähdä mahdolliset virheet nopeasti.

Testaus tapahtui aika pitkälti niin kuin olin suunnitelmassani esittänyt. Karttadatan lukua varten tein metodin printMapInfo, jonka tulosteesta pystyin tarkistamaan toimiiko readMapData niin kuin pitää. Tein myös muutaman testimetodin, mutta itselle ne tuntuvat hitaammalta tavalta testata ohjelman toimivuutta. Pyöritin ohjelmaa myös virheellisillä karttatiedostoilla ja tarkistin että ne aina aiheuttivat oikeanlaiset virheilmoitukset, jotta käyttäjän on helppo korjata virheet luomassaan karttatiedostossa.

Aina kun sain jonkun osa-alueen tehtyä, esimerkiksi tornien rakentamisen, laitoin ohjelman pyörimään ja yritin käydä läpi kaikki mahdolliset variaatiot siitä, mitä käyttäjä edes teorian tasolla saattaisi yrittää tehdä. Näin bongasin monia pieniä virheitä, jotka jonkun ajatusvirheen takia olivat koodausvaiheessa jääneet huomaamatta. Lisäksi testaaminen graafisen käyttöliittymän kanssa helpotti ymmärtämään esimerkiksi mikä ensimmäisissä versioissani calculateClosestCorner- ja moveEnemy-metodeistani oli pielessä. Lisäksi pelitietojen päivittymiseen (tai yleensä päivittymättömyyteen) liittyvät ongelmat näkyivät nopeasti.

8. Ohjelman tunnetut puutteet ja viat

Ainoa isompi bugi, jonka pelistäni olen löytänyt on jo aikaisemmin mainittu kaatuminen silloin kun popup-ikkunan sulkee sen ylälaidasta. Tätä ongelmaa ei ollut ennen kuin lisäsin ohjelmaan aloitusvalikon. Ongelma katoaa, jos en piilota aloitusvalikkoa pelin ajaksi, mutta silloin sulkiessani ikkunan väärällä tavalla aloitusvalikko pomppa pelinäkymän päälle. En oikein käsitä mikä yhteys näillä kahdella asialla on, PyQt5-ymmärrykseni on vielä sen verran rajallista.

En myöskään ole panostanut hirvittävän paljoa aikaa pelin tai karttojen vaikeusasteen hiomiseen. Esimerkiksi tornien ja vihollisten attribuutteja (ampumistahti ja –säde, kestävyys ja nopeus jne.) tai vihollisten määrää ja ilmestymisintervalleja kartoissa ei ole optimoitu. Olen keskittynyt lähinnä varmistamaan että peli toimii teknisesti ja koodissani ei ole isoja virheitä tai heikkouksia. Uskon kuitenkin että jo nykyisillä säätömahdollisuuksilla olisi attribuuttien arvoja ja karttatiedostoja hiomalla mahdollista saada aikaan toimiva ja sopivan haastava pelikokemus.

Karttadatan lukumetodi olisi todennäköisesti myös kannattanut jakaa pienempiin paloihin, jolloin sen eri osien testaaminen testimetodien avulla olisi käynyt kätevämmin. Lisäksi se olisi ehkä tällöin myös hiukan helpommin ymmärrettävissä ja paranneltavissa.

Projektiili-luokassa on pieni virhe. Ammukset nimittäin voivat osua maaliin päässeeseen viholliseen (mikäli vihollinen ehtii sinne ennen kuin sitä päin ammuttu ammus on osunut siihen). Tällöin pelaajaa saattaa saada rahaa mikäli ammus tappaa kyseisen, jo näkymättömäksi muuttuneen, vihollisen.

Vihollisten kutsuminen olisi myös tarvinnut pienen korjauksen. Nyt seuraava vihollinen aallon ensimmäisen jälkeen saattaa tulla pienemmällä intervallilla kuin mitä aallon tiedoissa on määritelty, koska metodi vertaa intervallia kuluneeseen aikaan eikä siihen koska edellinen vihollinen on lähetetty. Tämän voisi korjata lastEnemySent-attribuutilla.

9. Parhaat ja heikoimmat kohdat

Olen tyytyväinen ohjelman visuaaliseen puoleen, käyttöliittymään ja grafiikoihin. Ilme on mielestäni yhtenäinen ja toimiva ja käyttöliittymä selkeä ja intuitiivinen. On tietysti vielä paljon ominaisuuksia, jotka olisin halunnut peliin lisätä, esimerkiksi Restart- ja Main Menu –napit. Mutta jossain vaiheessa oli pakko päättää, että hienosäätö saa riittää.

Toinen asia mihin tähtäsin alusta asti ja missä koen onnistuneeni on pelin muokattavuus. Erilaisten vihollisten ja tornien luomisen pitäisi olla suhteellisen helppoa ja uusien karttojen tekeminen ja lisääminen peliin ei myöskään vaadi hirveästi työtä. Ja pelin asetuksien ja ulkonäön säätäminen onnistuu myös kätevästi globals-moduulissa olevien attribuuttien kautta. Ja koska käyttöliittymän eri osaset on toteutettu omina luokkinaan, voi sitäkin melko ongelmitta lähteä muokkaamaan ja kehittämään.

Heikkona puolena voisin mainita ainakin sen miten tornit valitsevat kohteensa ja ampuvat. Tällä hetkellä tornit ampuvat pisimmälle polulla matkannutta hyökkäyssäteen sisällä olevaa vihollista riippumatta siitä onko vihollinen hyökkäyssäteen sisällä enää silloin kun ammus osuu siihen. Ammukset yksinkertaisesti seuraavat vihollisia kunnes osuvat. Tämä aiheuttaa sen että ammusten nopeuden täytyy aina olla vihollisten nopeutta suurempi. Parempi vaihtoehto olisi laskea vihollisen tuleva sijainti hyökkäyssäteen sisällä ja laskea ammuksen reitti niin että ammus ja vihollinen kohtaavat tässä pisteessä. Tämä vaikutti kuitenkin huomattavasti monimutkaisemmalta tavalta ja vaatisi sitä että tekisin metodin (joka aika pitkälti tosin pohjautuisi vihollisten nykyiseen liikkumismetodiin), joka laskee vihollisen sijainnin x:n aikayksikön päästä. Sitten pitäisi laskea mihin asti ammus samassa ajassa ehtisi ja yrittää löytää lähin aika ja sijainti missä nämä kaksi risteävät (tässä voi olla pieni marginaali). Päätin ensin toteuttaa helpommin koodattavan seurausalgoritmin ja kokeilla tätä mielestäni parempaa tapaa, jos aikaa jää. Todennäköisesti hion peliäni vielä kurssin jälkeen ja mietin tämän metodin loppuun.

Toinen kehitettävä asia voisi olla vihollisten liikkuminen, joka nyt on melko suoraviivaista ja jäykkää (kaikki viholliset kulkevat tasan samaa reittiä ja ne voivat mennä päällekkäin). Joku vapaammin määritelty ruuhkasimulaatio tyyppinen liikkumismetodi voisi olla pelaamisen kannalta mielenkiintoisempi. En valitettavasti ehtinyt sen tarkemmin perehtyä muihin vaihtoehtoihin, joten en aivan osaa sanoa miten sellaisen voisi tässä pelissä toteuttaa.

10. Poikkeamat suunnitelmasta

Peli vastaa aika pitkälti teknisen suunnitelmani käyttötapakuvausta. Minulta on jäänyt pois vain pelaajan nimen kysyminen (nykyisessä muodossa peli ei tee tiedolla mitään) ja nappi ensimmäisen aallon kutsumista varten (aalto lähtee liikkeelle automaattisesti pienen viiveen jälkeen). Lisäominaisuutena aloitusvalikossa on mahdollisuus säätää pelinopeutta.

Lopullinen luokkajakoni ei kuitenkaan ihan vastaa alkuperäistä suunnitelmaani. Jätin Wave- ja Player-luokat pois. Jälkimmäinen oli ”story modea” varten, jota en toteuttanut ja oma toteutukseni aalloista oli sen verran yksinkertainen etten vielä kokenut tarpeelliseksi luoda niille omaa luokkaa. Lisäksi Game-, Map- ja GameInterface-luokat muuttuivat GameBoard-, MapView- ja UserInterface-luokiksi ja osa metodeista ja muuttujista vaihtoi paikkaa niiden välillä. Lisäksi UserInterfacesta irtaantui vielä BottomButtons- ja GameStats-luokat.

Koodasin pelin lähes täysin siinä järjestyksessä missä olin ajatellutkin ja uskon että aika-arvionikin osuivat melko oikeaan siltä osin kun olin tarkempia tuntimääriä miettinyt.

11. Toteutunut työjärjestys ja aikataulu

Kuten olin suunnitellutkin lähdin liikkeelle karttadatan lukumetodista ja alustavaan versioon tuhraantui arvioimani yksi päivä (5-6 tuntia). Lisäsin ensimmäiset tiedostot Niksulaan 7.3. Seuraavana päivänä tein kartanlukumetodiin vielä joitain päivityksiä, loin käyttöliittymässä käyttämäni neljä luokkaa (UserInterface, MapView, GameStats ja BottomButtons) sekä kehitin kartanpiirtometodin. 10.3. Sain torninostonapit lisättyä ja 17.3. mennessä pelaaja pystyi rakentamaan torneja kartalle. 18. – 24.3. parantelin tornien rakentamista (lisäsin varjon, joka näyttää mihin tornia ollaan laittamassa ja mikä sen hyökkäyssäde on) ja koodasin popup-ikkunan, joka aukeaa kun tornia klikkaa sekä kirjoitin tornien päivitykseen tarvittavat metodit. Ennen checkpointia 9.4. koin olevani suurin piirtein puolessa välissä projektini kanssa (käyttöliittymä ja tornien rakentaminen olivat valmiita, mutta vihollisten liikkuminen ja ampuminen vielä tekemättä). Sen jälkeen aloin työstämään vihollisluokkaa. 12.4. ensimmäinen versio vihollistenliikkumismetodista oli valmis. Tämän jälkeen siirryin projektiili-luokkaan ja tein paljon pieniä korjauksia ohjelmaan, jotka lisäsin Niksulaan 16. – 19.4. Seuraavana päivänä 20.4. sain vihollisten liikkumisen toimimaan haluamallani tavalla. 23. – 24.4. kehitin tornien ampumiseen ja pelin loppumiseen tarvittavat metodit. 25.4. siistin koodia, kirjoitin kommentteja ja parantelin karttoja. 26.4. lisäsin aloitusvalikon. Tämän jälkeen (ja oikeastaan läpi koko prosessin) olen tehnyt peliin pieniä parannuksia ja kosmeettisia muutoksia. Aina kun olin jumissa jonkun haastavamman metodin kanssa piirtelin meditatiivisena harjoituksena pelin grafiikoita.

12. Arvio lopputuloksesta

Olen melko tyytyväinen siihen mitä sain aikaan, vaikka kyllähän ohjelmasta löytyy vielä vaikka kuinka paljon parannettavaa. Kuten yllä jo mainitsinkin on pelin visuaalinen ilme mielestäni verrattain onnistunut. Tiedän että se ei ollut projektin kannalta mitenkään merkittävä osa-alue, mutta itselle sitäkin tärkeämpi. Pyrin mahdollisimman yksinkertaiseen toteutukseen, josta löytyy kuitenkin kaikki pelaamisen kannalta välttämättömimmät ominaisuudet. Alla on listattu joitain harkitsemiani lisäyksiä:

- Pelaajan kannalta olisi hyvä, jos vihollisten päällä olisi palkki joka näyttää kuinka paljon sillä on vielä kestoa jäljellä (tämän voisi varmaan toteuttaa esim. QProgressBar-widgetillä).
- Olisi myös kiva, jos ammuksen osuminen näkyisi pelaajalle jollain tavalla (esim. välähtävänä kuvana).

- Pelissä pitäisi olla mahdollisuus myydä torneja.
- Joillain vihollisilla voisi olla vastustuskykyä tietynlaista vahinkoa vastaan.
- Mistä päästään siihen että pelissä voisi olla myös ainakin taikavahinkoa tekevä tornityyppi.
- Ampuvien tornien lisäksi pelaajan arsenaalissa voisi olla esimerkiksi myös tielle asetettavia piikkejä, jotka satuttavat ja hetkellisesti hidastavat niiden yli käveleviä vihollisia.
- Peliin voisi lisätä mahdollisuuden toiseen vihollispolkuun, jolloin vihollisia voi kartalla tulla samaan aikaan kahdesta eri suunnasta.
- Puhuin jo aikaisemmin siitä että sekä tornien ampumista, että vihollisten liikkumista voisi parantaa.
- BottomButtons-luokkaan voisi lisätä Main menu- ja Restart-napit, sekä mahdollisuuden säätää pelinopeutta kesken pelin.
- Haluaisin myös että tornien ja vihollisten attribuuttien arvot näkyisivät niitä klikattaessa tai tornia ostettaessa nykyisessä ikkunassa, sen sijaan että ne tulevat omaan popup-ikkunaansa. Säädin tämän kanssa jonkun aikaa projektin alkuvaiheessa kunnes luovutin, koska toteutus ei näyttänyt hyvältä ja tietojen päivittymisen kanssa oli vähän ongelmia. Uskon että osaisin ehkä nyt tehdä tämän paremmin.
- Haluaisin peliin tarinamoodin.
- Karttoja sekä tornien ja vihollisten attribuuttien arvoja pitäisi hioa, jotta pelin ja karttojen vaikeusasteet saisi kohdalleen.

Toinen hyvä piirre ohjelmassa on sen laajennettavuus. Puhuin jo aikaisemmin siitä kuinka peliin on helppo tehdä uusia vihollis- ja tornityyppejä ja karttojen lisääminen on myös yksinkertaista. Globals-attribuuteilla voi kätevästi muuttaa pelin värimaailmaa, mutta esim. blockSizea ei voi ihan vapaasti mennä säätämään, lähinnä siksi että viholliset ja tornit ovat kuvatiedostojensa takia määrätyn kokoisia. Mutta pienillä muutoksilla niidenkin koon voisi varmaan sitoa blockSizeen. Uskon myös että kaikki yllämainituista ominaisuuksista pitäisi olla suhteellisen helposti lisättävissä. Vain tarinamoodi ja tornien/vihollisten tietojen näyttäminen vaatisi hiukan enemmän miettimistä.

Ohjelman luokkajako on mielestäni ihan hyvä, mutta ehkä metodeja voisi vielä jonkun verran siirrellä luokkien välillä. Esimerkiksi peliä pyörittävät metodit olisi voinut periaatteessa siirtää MapView:sta Gameboard-luokkaan, jolloin attribuutteihin viittaaminen olisi vaatinut vähemmän kirjoittamista (nyt kaiken edessä lukee ”self.parent.gameboard.”).

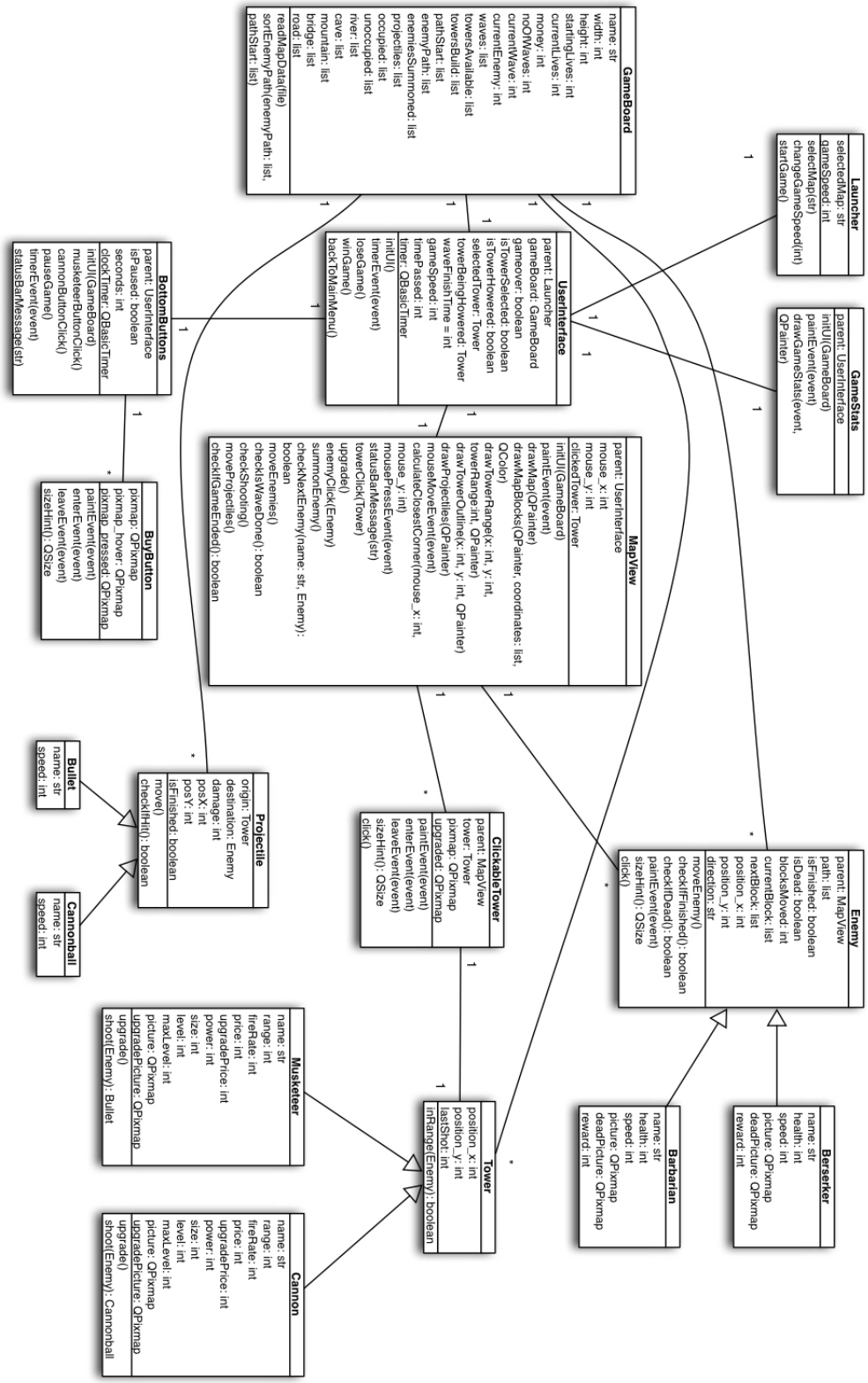
Minun on vaikea antaa sen selkeämpää arviota lopputuloksesta, koska en tiedä millaisiin urotekoihin opiskelijat tällä kurssilla yleensä yltävät. Sen tiedän että näin paljon vaivaa projektin eteen ja opin älyttömästi uutta Pythonista, graafisista käyttöliittymistä ja ohjelmoinnista ylipäätään. Kun kävin valmista koodiani tätä dokumenttia varten läpi, huomasin että toteuttaisin monet alussa koodaamani asiat nyt erilailla. Kun alla on tämän jälkeen yksi astetta monimutkaisempi koodausprojekti osaisin todennäköisesti tehdä myös projektisuunnitelman tulevaisuudessa paremmin. Eli ei tämä mikään täydellinen ja virheetön suoritus ollut, mutta lopputulos on kuitenkin (pientä vältettävissä olevaa kaatumisongelmaa lukuun ottamatta) toimiva ohjelma.

13. Viitteet

PyQt5 tutorial. Osoite: http://zetcode.com/gui/pyqt5/
StackOverFlow. Osoite: http://stackoverflow.com
Python Standard Library. Osoite: https://docs.python.org/3/library/index.html
Wander-tower-def. Osoite: https://github.com/mlisbit/wander-tower-def
Python PyGame Tower Defence. Osoite: http://pygame.org/project-Python+PyGame+Tower+Defence-1296-4410.html
A+ -materiaalit. Osoite: https://plus.cs.hut.fi/y2/2016/

14. Liitteet

Liite 1 UML-kaavio. En ole kaikista noista yhdysviivoista varma GameBoardin ja MapView:n outo suhde tekee tästä vähän vaikeaa.



Liite 2 Esimerkki karttatiedostosta

```
#mapinfo
name: Battle of Mag Tuireadh
size: 51x26
lives: 10
money: 800
towers: t1, t2
path start: 10, 6

#waves
number of waves: 6
50, e1, e1, e1, e2, e2, e1, e2, e1, e2, e1, e2, e1
80, e2, e1, e2, e1, e1, e2
60, e1, e2, e2, e1, e1, e1, e2, e2
70, e2, e2, e1, e1, e1, e2, e2, e1, e1, e1, e2
20, e1, e1, e1, e1, e2, e1, e2, e1, e1, e2, e1, e2, e2, e1
40, e1, e1, e1, e2, e1, e2, e2, e2, e2, e1, e2, e2, e1


#map
MMMMMMMMM+RRRRR++++++++++RRR++++++++++++++++++++
MMMMMMMMM++RRRR++++++++++RRR++++++++++++++++++++
MMMMMCCCMMM++RRRR++++++++RRR+++M+M+M++++++++M+M+M+++
MMMMCCCCCMMM++++RRRR++++RRR++++MMMMM++++++++MMMMM+++
MMMMCCQCCMMM++++++RRRRRRRR++++++MMM+++++++++MMM++++
++++++0P0++++++++++++++RRRRR++++++MMM+M+M+M+M+M+MMM++++
++++++0P0+++++++++++++++RRR++++++++MMMMMMMMMMMMMMM++++
++++++0P0+++++++++++++++RRR++++++++MMMMMCCCMMMMMMM++++
++++++0P0+++++++++++++++++RR++++++RRMMMMMCQCMMMMMMMRR++
++++++0P0+++++++++++++++++RRR++++RRRMMMMMCQCMMMMMMMRRR+
++++++0P0++++MM++++++++RRR++++++RRRRRRRWBWRRRRRRRRR++
++++++0P0+++MMMM+++++RRR+++++++++++++++WBW++++++++++
++++++0P0++++++++++++RRR+++++++++++++++0P000000++++++
++++++0P000000++++++++RRR+++++++++++++++0PPPPPP0++++++
++++++0PPPPPP0++++++++RRR+++++++++++++++000000P0++++++
++++++000000P0+++++++++RRR+++++++++++++++++++0P0++++++
++++++++++0P0+++++++++RRR+++++++++++++++++++0P0++++++
++++++++++0P000000000WWWW00000000000000000000P0++++++
++++++++++0PPPPPPPPPBBBPPPPPPPPPPPPPPPPPPPP0++++++
++++++++++00000000WWWW0000000000000000000000++++++
++++++++++++++++++RRR++++++++++++++++++++++++++++++
++++++++++++++++++RRR++++++++++++++++++RRRR++++++++++
++++++++++++++++++RRR++++++++++++++++++RRRRRRRR++++++++
++++++++++++++++++RRR++++++++MM++++++++++RRRRRRR++++++++
++++++++++++++++++RRR++++++++++++++++++++++RRR++++++++++
++++++++++++++++++RRR++++++++++++++++++++++++++++++++++
```