Yleissuunnitelma

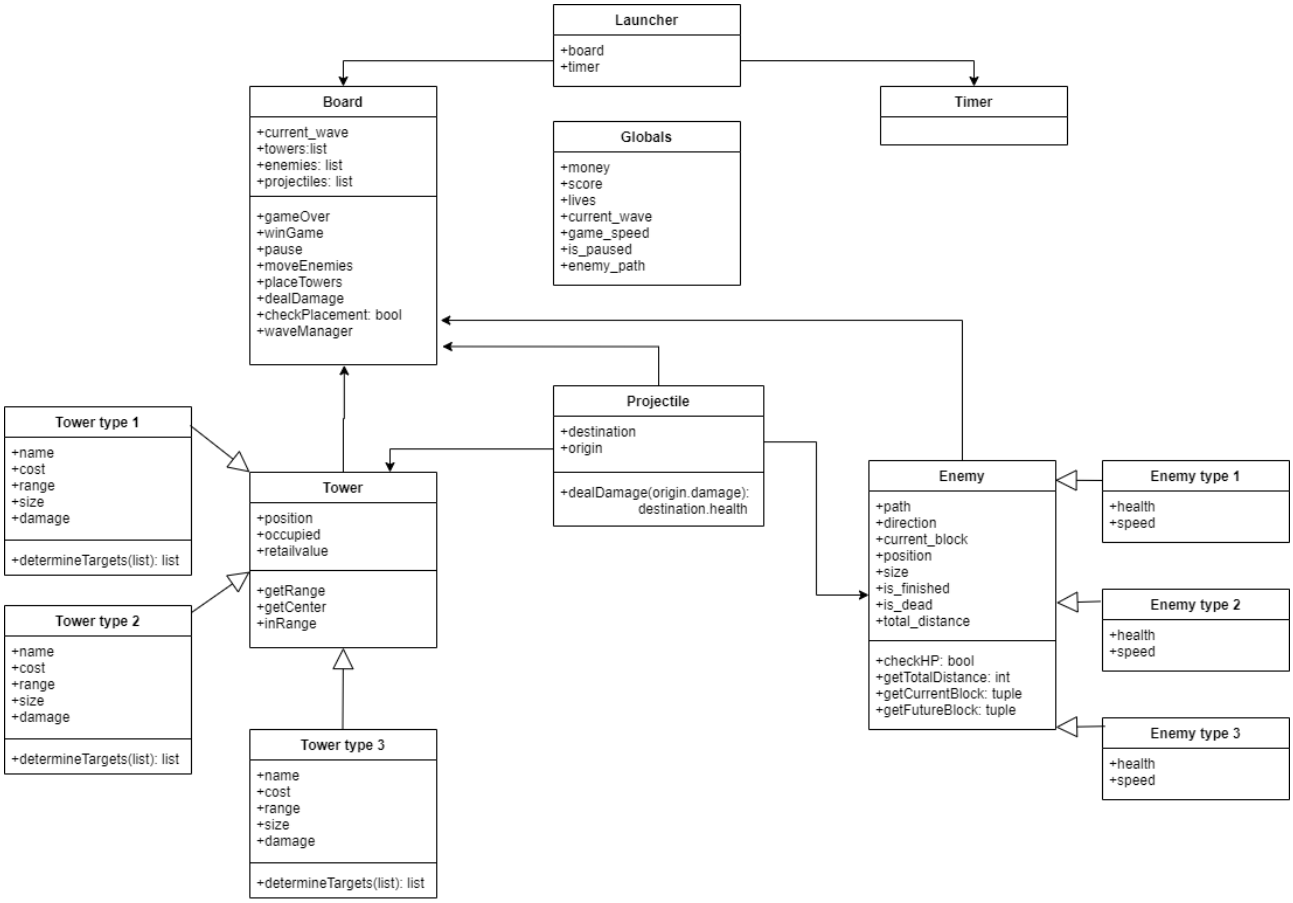
Veeti Kuivalainen

717005

EST 2. vuosi

Ohjelmoinnin peruskurssi Y2

Ohjelman rakennesuunnitelma



UML-kaavio kertoo itse pelin toiminnan kannalta tärkeistä luokista. PyQT-grafiikat ja vähemmän tärkeät luokat ja metodit, kuten tiedostonluvun, olen jättänyt kaaviosta pois. Suurin osa ohjelman toiminnoista tapahtuu Board-luokassa eli pelilaudalla. Pelilaudalla näkyviä olioita ovat tornit, viholliset ja projektiilit. Tornit ja viholliset vuorovaikuttavat keskenään projektiilien ja koordinaattijärjestelmän välityksellä. Sekä torneja että vihollisia on monenlaisia ja alaluokat perivät yläluokan. Graafista käyttöliittymää alan tekemään RobotWorld-tehtävän mallin mukaan.

Board-luokassa ovat pelin tärkeimmät metodit. Se päättää, milloin peli on voitettu tai hävitty. Sinne lisätään tornit ja viholliset ja liikutellaan niitä. Enemy-oliot tietävät oman polkunsa ja elämänsä. Siksi niillä on metodit, joilla voi kysyä niiden sijaintia, tulevaa sijaintia ja elämien määrää. Torneilla on yksi metodi, joka määrittää kantaman sisällä olevat viholliset ja päivittää siellä olevien vihollisten listaa koko ajan. Projektiilit taas ovat vahingon välittäjiä ja niiden ainoa metodi kertoo viholliselle, että siihen on osuttu.

Käyttötapauskuvaus

Pelin lähes kaikki tapahtumat tapahtuvat pelilaudalla, joka ei ole pelaajalle näkyvissä. Pelaaja näkee pelin PyQT-grafiikkojen kautta ja antaa komentoja klikkailemalla PyQT-widgettejä. Taustalla pyörii koko ajan ajastimella tahdistettu ohjelma. Joka kerta kun pelin tilanne päivittyy, päivittyvät myös grafiikkaoliot. Grafiikkaolioita klikkailemalla pelaajalle näytetään tietoa oikean taustaolion ominaisuuksista.

Algoritmit

Peli perustuu ruutuihin, joilla on tieto sisällöstään: onko ruudussa vihollinen, torni vai projektiili. Torni voi käyttää algoritmia ammuttavan vihollisen löytämiseen. Se voi priorisoida vaikkapa ensimmäisen vihollisen, joka sen kantamassa on. Silloin täytyy koordinaattisysteemin avulla kerätä lista kantaman sisällä olevista vihollisista ja ampua sitä, jolla kuljettu matka on pisin.

Tietorakenteet

Projektissa ei tarvita kovinkaan paljon tietorakenteita. Listoja tarvitaan osoittamaan pelilaudan ruudut, tornit ja viholliset. Tarvitaan myös lista jokaiselle tornille, jossa ovat kantaman sisällä olevat viholliset. Taulukkona voidaan esittää vihollisten kulkema polku.

Kirjastot

Tarvittavia kirjastoja ovat PyQT5, jolla tehdään grafiikat ja käyttöliittymä. Todennäköisesti myös math-kirjasto, josta saadaan käteviä funktioita ja json-kirjasto jos vihollisaallot tallettaa json-tiedostona.

Aikataulu

9.-15.3.	Käyttöliittymän teko, pelaaja voi painella tärkeimpiä nappeja ja esimerkiksi sulkea pelin tai pysäyttää ajastimen.
16.-22.3.	Kenttien lataaminen, vihollisten liike kentällä ja tornien asettelu.
23.-29.3.	Tornit voivat ampua vihollisia, jotka ovat kantaman sisällä ja valitsevat niistä ensimmäisen.
30.3.-5.4.	Viholliset tulevat aalloittain, rahankäyttö, pisteet, elämät.
6.-12.4.	Käyttöliittymän viimeistelyä ja grafiikkojen parantelua, projektiilien animointi
13.4.-->	Testailua ja laajentamista

Yksikkötestaussuunnitelma

Ensimmäisenä tulee vastaan kentänlataamistiedoston eheyden testaaminen. Testataan, mahtuuko vihollisen polku kentälle ja piirrettävälle ruudulle. Polun pitää olla myös ehjä eli tiedostossa peräkkäisten ruutujen täytyy olla vierekkäin kentällä. Testit tapahtuvat syöttämällä virheellinen tiedosto pelille ja katsomalla mitä tapahtuu.

Rahankäyttöjärjestelmää tulee testata, että oikea määrä rahaa lähtee torni asetettaessa ja että rahat eivät voi mennä miinukselle. Sama elämien kanssa, pitää kokeilla lähteekö elämiä aina vihollisen päästessä maaliin ja ovatko vihollisen elämät positiivisia kokonaislukuja. Tornien toiminnallisuudesta voidaan testata determineTargets-metodia, jonka pitäisi päivittää listaa kantaman sisällä olevista vihollisista. Se ei saisi sisältää esimerkiksi kuolleita vihollisia tai samaa vihollista kahdesti.

Käyttöliittymän testaaminen tapahtuu käytännössä peliä ajettaessa. Sen huomaa jos jokin widget ei toimi halutulla tavalla kun sitä klikkaa. Silti pitää varautua esimerkiksi tuplaklikkauksiin ja testata etteivät ne tuota silmämääräisesti huomaamatonta häiriötä.

Kirjallisuusviitteet ja linkit

Aloitan projektin teon ottamalla mallia aiemmasta RobotWorld-tehtävästä ja googlaan erilaisia PyQt:lla tehtyjä pelejä. Github-linkkejä näyttäisi löytyvän ihan mukavasti ja Youtubessa on joitakin tutoriaaleja pelejen luomiseen. Lisäksi käytössä ovat tietysti Pythonin ja PyQt:n dokumentaatiot.