Y2 Projektityön dokumentti

1. **Henkilötiedot**

Tower Defence
Ville Strengell
594590
ELEC/AUT
21.4.2019

2. **Yleiskuvaus**

Tavallinen tornipuolustus, jossa grafiikoina on vain palloja ja neliöitä. Torneja voi ostella sivupalkista, ja niitä voi asettaa kentälle estääkseen vihollisaaltojen pääsyn loppuun asti.

3. **Käyttöohje**

Alkuvalikosta voi valita “Settings”, josta saa valittua kartan painamalla ensin jompaakumpaa vaihtoehdoista, ja sitten “back”.

Torneja ostetaan sivupalkista, josta klikkaamalla luodaan epäaktiivinen torni kentän laidalle. Torni kuuluu sitten raahata hiirellä kentälle, ja asettaa johonkin vihreistä blockeista. Tällöin torni aktivoituu. Painamalla sivupalkista “next wave” vihollisia alkaa rymyämään kentälle ja tornit toivottavasti päivittävät ne ennen maaliin pääsyä. Kun pelaajan HP:t tippuvat alle 0, peli loppuu ja palataan main menuun.

4. **Ulkoiset kirjastot**

En oikeastaan käyttänyt mitään muita ulkoisia kirjastoja kuin QT.

5. **Ohjelman rakenne**

Itse peliä pyöritetään “Peli” luokassa, joka perii QMainWindow:luokan. Peliin on importattu kaikki muut osat, eli Mapper, Enemy, Tower, Player ja WaveManager. Näistä luodaan sitten olioita pelin edetessä. Pelin looppina toimii QTimerin avulla toteutettu update-funktio, jossa päivitetään vihollisten ja luotien liikkeet sekä tornien pyöriminen yms.

Päätin käyttää Pelin luomiseen QMainWindowia, sillä sen avulla sain kaiken toimimaan alussa hyvin. En kokenut, että esimerkiksi QDialogin tai QViewin käyttämisestä saisi tarpeeksi etua, että vaihtaisin niihin tutusta QMainWindowista. Harkitsin jonkinaikaa QViewin käyttöä, ja näkymän vaihtuessa olisin vaihtanut vain QGraphicsSceneä, mutta päätin jäädä alkuperäiseen pohjaan, joka minulla oli jo valmiina.

Kun ohjelma ajetaan, siinä aukeaa QGraphicsScene, joka on kovakoodattu main menu-näkymä. Kun painetaan “play” nappia, se scene poistetaan ja samaan ikkunaan vanhan

scenen tilalle luodaan uusi QGraphicsScene pelille. Se lukee Mapper-olion avulla tiedostosta millainen kartta on luotavissa.

Mapper-luokka huolehtii kartan lukemisesta tiedostosta ja sen piirtämisestä. Siihen on myös tallennettu jokainen ruutu kaksiulotteiseen listaan. Listan avulla voidaan myöhemmin määrittää, mihin ruutuihin torni voidaan asettaa ja mihin ei, mistä ruudusta viholliset aloittavat matkansa ja miten ne pääsevät loppuun saakka.

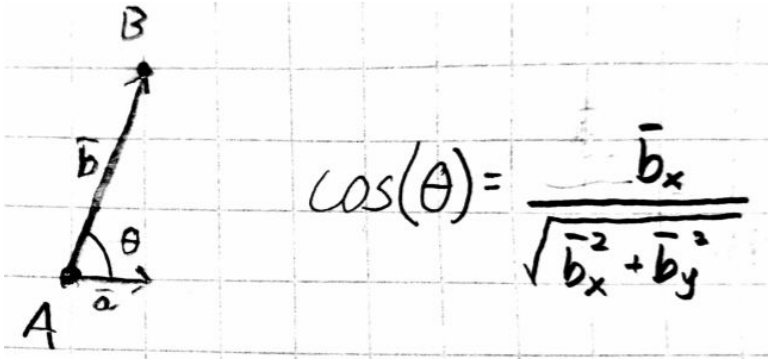
Tornien ja vihollisten luokat perivät QGraphicsItemin, tornit neliön ja viholliset soikion. Halusin, että tornit ovat neliön muotoisia jolloin ne asettuisivat hyvin kartalle. Vihollisiksi pallot olivat melko yksiselitteinen valinta. Kolmioiden kääntely liikesuunnan mukaan olisi vain aiheuttanut enemmän vaikeuksia.

Projectiles luokka importataan torneille, ja kun torni ampuu, se lisää pelissä olevaan projectiles-listaan projectilen, jolla on tietty suunta ja nopeus. Kun projectile osuu kohdeviholliseensa, se katoaa ja pudottaa kohteen hp:ta tornin damagen verran.

6. **Algoritmit**

Ehkä olennaisimmat laskut tapahtuvat tornien sisällä, kun piippua käännetään kohti kohdeviholista. Tätä varten tarvitaan tornin ja vihollisen etäisyyden x- ja y-komponentit.

Määritin yksikkövektorin a, joka on vaakatasossa, ja sen avulla laskin vektorien välisen kulman. Laskusta tuli mielestäni todella yksinkertainen, enkä keksinyt helpompaa tapaa muuttaa koordinaatit kulmaksi.



Koodissa:

```
x_dist = enemy.ScenePos().x() - self.scenePos().x()
y_dist = enemy.ScenePos().y() - self.scenePos().y()
angle = degrees(acos(x_dist / hypot(x_dist, y_dist)))
```

Tässä laskussa saatua kulmaa voi käyttää myös luotien suunnan määrittämisessä. X-koordinaatille täytyy ottaa cosini kulmasta, ja y-koordinaatille sin. Tällöin luoti kulkee samalla nopeudella vaakatasossa kuin poikittain.

7. **Tietorakenteet**

Käytin pelkkiä listoja asioiden tallentamiseen. En kokenut tarpeelliseksi käyttää mitään sen monimutkaisempaa tietorakennetta.

8. **Tiedostot**

Kartat on tallennettu tekstitiedostoon, jossa jokaiselle kartan ruudulle on määritetty tietty merkki, ja niiden avulla kartan saa piirrettyä todella helposti ja nopeasti. Katsomalla esimerkkiä jo luoduista kartoista uuden luomisen pitäisi onnistua keneltä tahansa.

Päätin kokeilla ConfigParserin käyttöä joillekin arvoille, ja siitä tulikin ihan kätevä tapa hakea esimerkiksi ruutujen ja ikkunan koko. Asetustiedostosta “config.ini” on helppo käydä vaihtamassa näitä, ja tällöin ne tarvitsee käydä vaihtamassa vain yhdestä paikasta.

9. **Testaus**

Testasin kaikkia osioita vain kokeilemalla. Piirtyykö haluttu asia halutulla tavalla, liikkuuko kaikki niinkuin pitääkin yms. Ongelmien ilmetessä debuggasin printtien avulla. En tehnyt mitään yksikkötestejä, sillä koin niiden olevan turhaa ajanhukkaa tämänkokoiselle projektille.

10. **Ohjelman tunnetut puutteet ja viat**

Yksi ongelma on se, että jos ostaa tornin ja yrittää asettaa sitä kartan ulkopuolelle, ohjelma kaatuu. Tämä johtuu siitä, että se alue on blocks-listan ulkopuolella, ja ohjelma antaa IndexErrorin kun torni tarkistaa, onko se radalla vai hyväksytyllä ruudulla.

Toisena mieleen tulee se, että torneja voi laittaa päällekkäin. Tätä en kerennyt korjaamaan, mutta korjaisin sen suurinpiirtein seuraavasti:
Antaisin jokaiselle blockille(ruudulle) boolean arvon “has_tower”. Sitten torneja luodessa antaisin niille “blocks”-listan konstruktorissa, ja tarkistaisin “mouseReleaseEvent”:issä onko kyseinen torni toisen tornin päällä.

Välillä silloin, kun maali on jonkin tornin rangen sisällä, ja vihollinen pääsee elossa perille, ohjelma kaatuu antaen IndexErrorin viitaten enemies-listaan. En ole varma mistä se johtuu, ja se tapahtuu todella harvoin, joten en jaksanut lähteä ratkaisemaan ongelmaa.

11. **3 parasta ja 3 heikointa kohtaa**

Parasta:

Omasta mielestäni kartan piirto onnistuu todella hyvin. Tällä tavalla toteutettuna voi luoda loputtomasti karttoja hyvin intuitiivisesti. Vihollisten alkupistekin muodostuu automaattisesti kun karttaan tehdään aloituspiste.

Ohjelma on hyvin skaalautuva. Ruutujen koon muuttaminen ei vaikuta pelin kulkuun tai vihollisten/tornien piirtymiseen tai toimintaan.

Huonoa:

Pelin kulku itsessään ei ole hyvin optimoitu. Kierrosten vaikeutuminen myöhemmille aalloille on todella tönkkö, kasvattaen vain vihollisten määrää ja niiden health pointseja.

Tornit myös välillä ampuvat ohi vihollisista.

12. **Poikkeamat suunnitelmasta**

Päätin olla tekemättä yhtä pohjaluokkaa “object”, sillä en uskonut sen olevan yhtään helpompi toteutus kuin se, että päivitän erikseen tornit, viholliset ja luodit. Olin myös tottunut C++/C koodaamiseen, ja luulin esimerkiksi get-/set-metodien luomisen hyödylliseksi. Päädyin kuitenkin hakemaan arvot yksinkertaisesti enemy.hp, tower.dmg yms. Päätin myös tehdä tornit siten, että ne eivät ennakoi vihollisten liikettä. Muuten toteutin pelin aikalailla suunnitelmani mukaan.

Ajankäytön arvio meni aivan metsään, sillä käytin reilusti enemmän aikaa kuin odotin. Päätin vain tehdä tarkkaa työtä jokaisessa vaiheessa. Toteutusjärjestys pysyi lähes suunniteltuna.

13. **Toteutunut työjärjestys ja aikataulu** Kerro tässä yleisellä tasolla missä järjestyksessä projekti lopulta toteutettiin (mielellään myös päivämäärät). Missä poikettiin suunnitelmasta?

20.3. - 25.3. Main menun luominen ja pelin avaaminen QPushButtonilla.
25.3. - 28.3. Vihollisten luominen ja niiden liikkuminen rataa pitkin.
29.3. - 3.4. Tornien piirtyminen.
4.4. Tornien tähtääminen vihollisiin.
7.4. - 8.4. Projectile luokka, ja tornien ampuminen ja vihollisten elämien menetys/kuoleminen
Tämän jälkeen unohdin committailla hetkeen, mutta parantelin paljon sitä, mitä olin aiemmin tehnyt. Esimerkiksi tornien asettuminen ruutuihin ja aaltojen manageröinti

14. **Arvio lopputuloksesta**

Olen mielestäni tehnyt pelin todella hyvin. Uusien tornien, vihollisten ja karttojen luominen on helppoa, joten pelistä puuttuu enää pelkkä pelattavuuden suunnittelu. Joitain osia laajennettavuudesta olisi voinut tehdä paremmin, mutta aikarajoitteiden takia jouduin tyytymään tällaisiin ratkaisuihin. Esimerkiksi uuden tornin luomisessa täytyy muokata muutamaa eri tiedostoa ja niissä useaa funktiota. Myös useat vakiot voisivat olla siirrettynä config.ini tiedoston puolelle sen sijaan, että niille annetaan arvot keskellä koodia. Teinkin tuon ConfigParserin kokeilun ja oppimisen vuoksi.

Varmaan suurin asia mitä muuttaisin jos aikaa riittäisi, on se, miten tornit ampuvat luoteja. Tällä hetkellä onnistuu helposti vain yhden luodin luominen, vaikka yleinen tornityyppi tämänkaltaisissa peleissä on sellainen, että se ampuu usean luodin kerrallaan moneen suuntaan.

Toinen iso lisäys olisi se, että torneja pystyisi päivittämään. En saanut tornien päivitystä toimimaan hyvin, ja päätin olla lisäämättä sitä ominaisuutta.

Luokkajako mielestäni on aika optimaalinen, en tiedä, kuinka sitä voisi parantaa.

“Traceback (most recent call last):

File "C:\Users\ville\PycharmProjects\tower-defence\src\Peli.py", line 354, in update

tow.aim_at(self.enemies[tow.furthestTarget[1]].pos().x(),
self.enemies[tow.furthestTarget[1]].pos().y())

IndexError: list index out of range”

Tällainen virhe tulee joskus, kun maali on tornin kantaman sisällä ja vihollinen pääsee elossa maaliin. En ole keksinyt, mistä se johtuu, sillä vihollisten kadotessa se poistetaan enemies-listasta samantien.

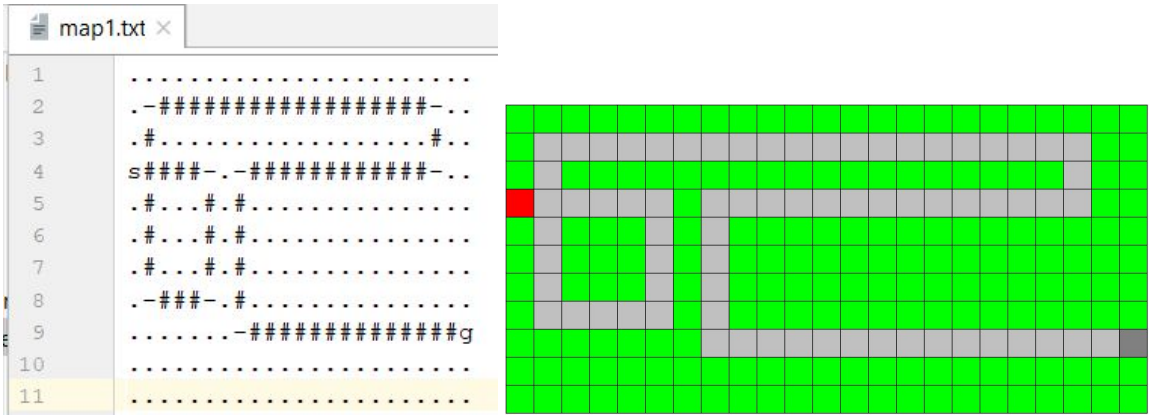
15. **Viitteet**

Aluksi katselin youtubesta opetusvideoita PyQt5 kirjastoon liittyen, ehkä tärkeimpänä “Mark Winifield” -kanavan “PyQt5 Lessons” sarja ja “thenewboston”-kanavan python tutoriaalit.

QT dokumentaatio
Pythonspot.com
Kurssin oppikirja ja kierrokset

16. **Liitteet**

Esimerkki kartan piirtymisestä tekstitiedostosta:



Esimerkki pelin kulusta kierroksella 4.