Clones TD Battles

Sampo Lenkola

1018130

Tietotekniikka 2022

17.04.2023

2. Yleiskuvaus

Projektini ideana on tornipuolustuspeli, jossa pelaaja pyrkii estämään vihollisten pääsemisen esteradan lävitse. Pelaaja pystyy sijoittamaan esteradan ympärille erilaisia puolustajia, jotka pyrkivät tuhoamaan viholliset. Peli päättyy, kun tietty määrä vihollisia pääsee puolustuksen läpi, eli peli jatkuu loputtomasti, jos pelaaja on tarpeeksi taitava.

Peliä pelataan asettamalla eri värisiä neliöitä radan ympärille, jotka ampuvat vain oman värisiä palloja (vihollisia). Tornit ampuvat vain tietyllä etäisyydellä, joka havainnollistetaan tornia asettaessa. Yhden tornin asettaminen maksaa 50$ ja pelaajalla on pelin aluksi käytettävissä 500$. Lisäksi pelaaja pystyy parantamaan kaikkia jo asettamiaan torneja erillisestä nappulasta. Tornien parantaminen maksaa aluksi 800$ ja jokaisen parantamisen jälkeen summa kasvaa 100$. Jotta pelaaja pystyy puolustamaan yhä suurempia määriä palloja (vihollisia), on hänen jatkuvasti asetettava uusia puolustajia kartalle sekä parannettava niitä. Jotta tämä olisi mahdollista, saa pelaaja jokaisen kierroksen lopuksi 300$ lisää rahaa.

Pelin loputtua pelaaja voi tallentaa tuloksensa ja pelata peliä uudestaan. Pelaajalla on myös mahdollisuus tehdä omia karttoja menu-valikon kautta sekä käydä katsomassa parhaita suorituksiaan valikosta.

Olen toteuttanut pelin vaativalla tasolla.

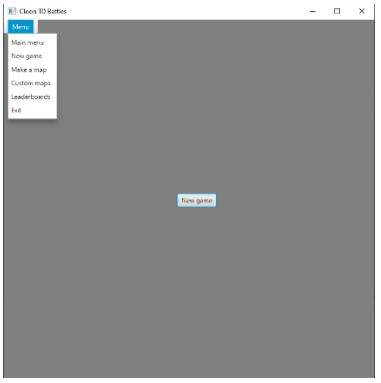
3. Käyttöohje

Näkymä 1.

Ohjelma käynnistetään tiedoston MainFX kautta, jolloin ruudulle avautuu aloitusruutu: *Näkymä 1.*

Aloitusruudussa on lyhyt teksti sekä painike: Start, jota painamalla pääsee seuraavaan ruutuun: *Näkymään 2.*

Näkymä 2. on väliruutu, jonka sisältää nappulan, jonka avulla voi nopeasti aloittaa uuden pelin, sekä Menu valikon. Menu valikko on näkyvissä kaikissa muissa näkymissä, paitsi *Näkymässä 1*. **Menu valikko sisältää 6 kohtaa:**

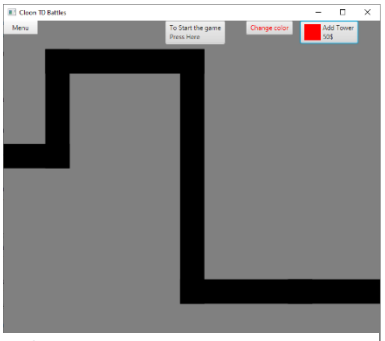


Näkymä 2.

1) **Main menu**
   Main menu kohdan avulla pääsee takaisin *Näkymään 1.*

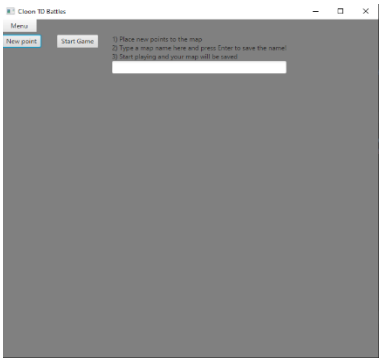
2) **New game**
   New game kohta ohjaa käyttäjän uuteen peliin *(Näkymä 3)*. Se aloittaa aina uuden pelin normaalilla kartalla, joten jos pelaaja painaa kyseistä kohtaa esimerkiksi pelatessaan itse tekemäänsä karttaa, häviää pelaajan oma kartta kokonaan.
   Tämä painike sekä *Näkymän 2.* keskellä oleva painike suorittavat täsmälleen samat komennot, joten sillä ei ole väliä, kumpaa käyttäjä käyttää.

3) **Make a Map**
   Make a Map kohta ohjaa käyttäjän *Näkymään 4.* Tässä näkymässä on kaksi nappulaa: New point sekä Start Game. Lisäksi ruudulta löytyy tekstikenttä.



Näkymä 3.

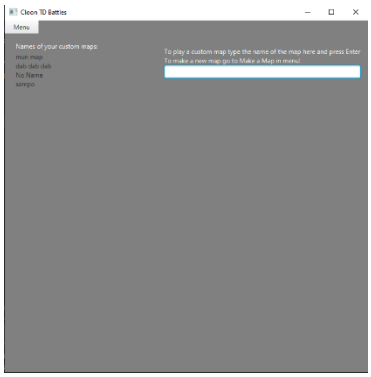
   Näkymän tarkoituksena on se, että pelaaja voi luoda oman tasonsa. Taso määrittyy pisteiden kautta, joita käyttäjä voi lisätä painikkeella New point. Kun käyttäjä painaa New point painiketta, tulee ruudulle näkyviin haalea ympyrä, joka seuraa hiiren liikettä. Kun käyttäjä painaa missä tahansa kohtaa hiirtä, asettuu pallo tähän kohtaan. Jokainen pallo merkkaa kääntymiskohtaa polulla, jossa viholliset kulkevat. Polku alkaa aina ruudun vasemmasta reunasta, siltä korkeudelta, johon ensimmäinen piste on asetettu. Polku loppuu aina Ruudun oikeaan reunaan, sillä korkeudella, johon käyttäjä on viimeisen pisteen asettanut. Pisteitä voi asettaa mielivaltaisen määrä, eikä niillä ole mitään muita sääntöjä, kuin että niitä pitää olla vähintään yksi.
   Kun käyttäjä on mielestään saanut tason valmiiksi, voi hän nimetä tasonsa, jolloin se tallentuu erilliseen tiedostoon. Pelin voi aloittaa painamalla painiketta Start Game. Jos uuden pelin aloittaa ilman, että uusi taso on nimetty, tulee tason nimeksi No Name.



Näkymä 4.

4) Custom maps

Custom map ohjaa käyttäjän *Näkymään 5.* Kyseisessä näkymässä pelaaja voi katsella aikaisemmin tekemiään tasoja sekä pelata niitä. Tasojen nimet ovat vasemalla ja niitä pelataan kirjoittamalla tason nimi oikealla puolella olevaan tekstikenttään. Kun jokin aikaisempien tasojen nimistä on kirjoitettu oikein, voi pelaaja painaa enteriä ja aloittaa pelaamisen.



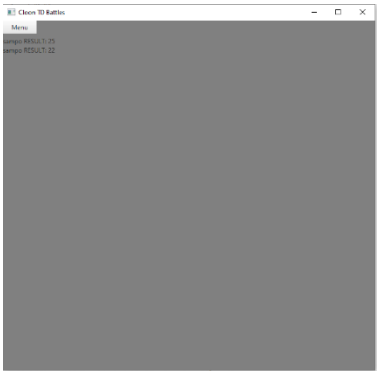
Näkymä 5.

5) Leaderboards

Leaderboards kohta ohjaa käyttäjän *Näkymään 6.* Tässä näkymässä käyttäjä voi katsella edellisiä tuloksiaan. Tulokset ovat järjestetty parhaimmasta huonoimpaan ja käyttäjän kirjoittamasta nimestä näkyy aina maksimissaan 6 ensimmäistä merkkiä. Tulokset tallennetaan vain alkuperäisestä kartasta, joten käyttäjän itse tekemien karttojen tuloksia ei tallenneta. Tulokset säilyvät, vaikka käyttäjä uudelleen käynnistäisi ohjelman.

6) Exit

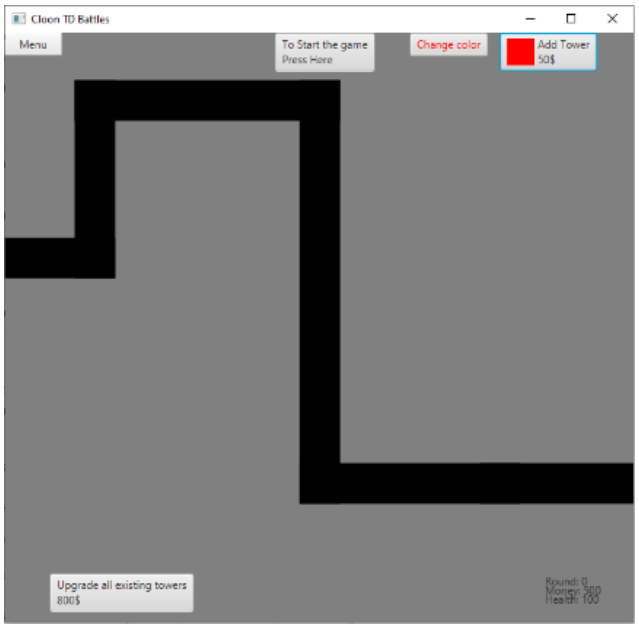
Exit painike sulkee ohjelman.



Näkymä 6.

Pelin pelaaminen

Itse pelin pelaaminen tapahtuu *Näkymässä 3.* Uuden pelin voi aloittaa kaikista näkymistä Menu valikon kautta, lukuun ottamatta *näkymää 1.*



Näkymä 3.

Näkymässä 3. on Menu valikko sekä **4 painiketta:**

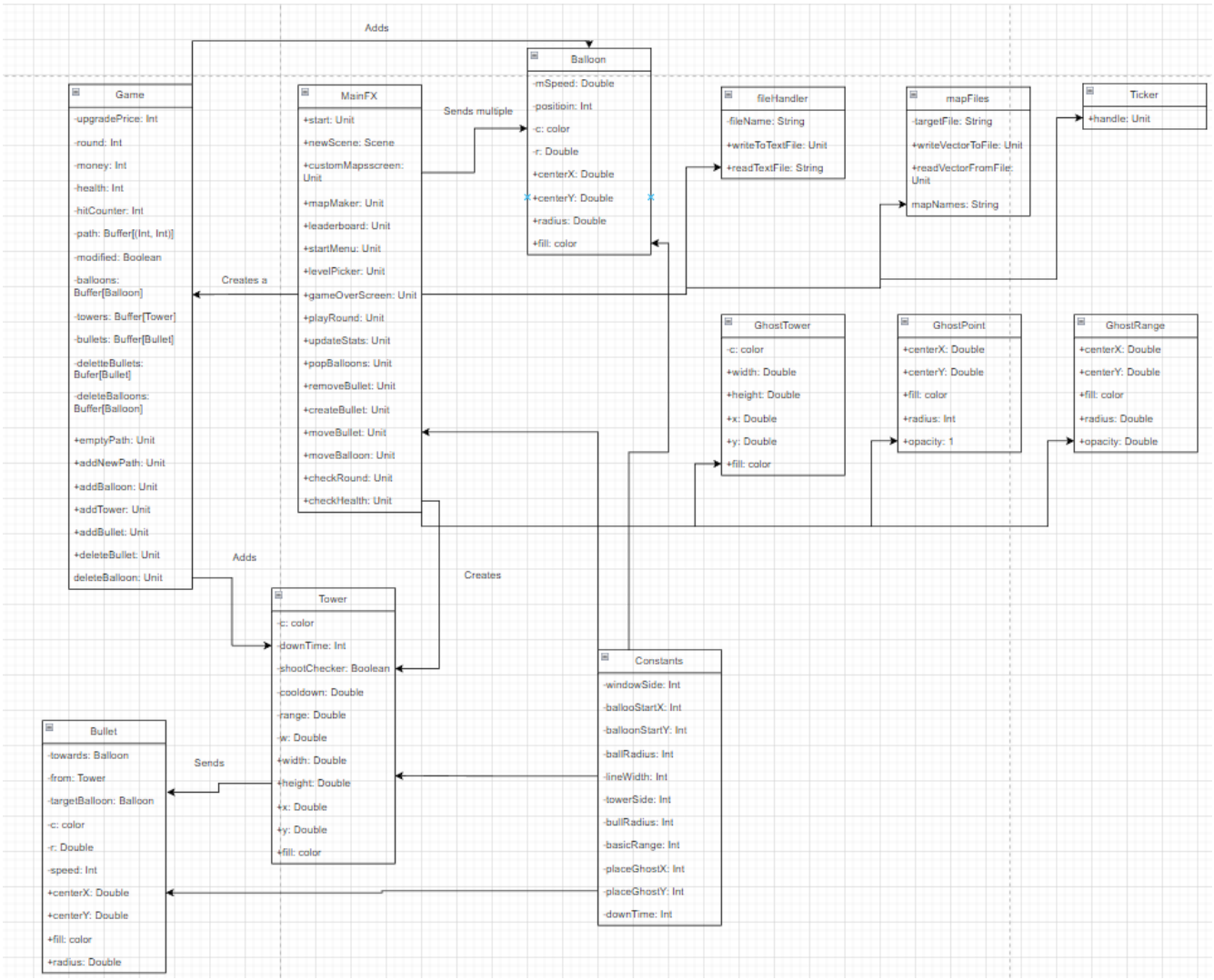
1) **To Start the game Press Here**
   Tämän painikkeen avulla käyttäjä aloittaa pelin eli lähettää ensimmäiset viholliset polkua pitkin. Ennen kuin käyttäjä on painanut kyseistä painiketta, voi hän asettaa torneja polun ympärille. Tämä on suositeltavaa, sillä tornien asettamisessa saattaa tulla kiire pelin edetessä.

2) **Change color**
   Tämän painikkeen avulla käyttäjä voi vaihtaa uuden tornin (ei vielä asetettu) väriä. Painiketta painamalla tornin väri vaihtuu painikkeessa **Add Tower.** Tämän painikkeen painaminen ei kuitenkaan vielä valitse uutta tornia, joten sitä voi käyttää huoletta.

3) **Add Tower 50$**
   Tämän painikkeen avulla käyttäjä voi asettaa uuden tornin, jos hänellä on tarpeeksi rahaa käyttävänä. painamalla kyseistä painiketta (olettaen, että rahaa on tarpeeksi), syntyy kursorin päälle haalea ympyrä sekä torni. Haalea ympyrä havainnollistaa uuden tornin ampumisetäisyyden, jotta käyttäjä voi valita parhaan mahdollisen paikan tornilleen. Tornin voi asettaa mihin tahansa kartalla. Itse tornin asettaminen tapahtuu painamalla kursoria haluamallaan paikalla. Kun torni on asetettu, vähentyy pelaajalta 50$ hänen rahoistaan. HUOM! kun Add Tower painiketta on painettu ei tornin asettamista voi enää perua.

4) **Upgrade all existing towers 800 $**
   Tämän painikkeen avulla käyttäjä voi parantaa kaikki jo asettamiaan torneja. Parantaminen maksaa 800$. Kun käyttäjä painaa painiketta, kaikkien jo asetettujen tornien ampumisetäisyys sekä ampumisnopeus kasvavat huomattavasti. Aina, kun käyttäjä painaa kyseistä painiketta, nousee hinta 100$. Tämän vuoksi käyttäjän on mietittävä sitä, milloin on järkevintä käyttää kyseistä toimintoa. HUOM! kyseisen painikkeen käyttäminen ei vaikuta mitenkään torneihin, jotka asetetaan käyttämisen jälkeen. Näin ollen Painikkeen **Add Tower** kautta lisätyt tornit ovat aina saman tasoisia, riippumatta siitä, monesti **Upgrade all existing towers** painiketta on käytetty.

4. Ohjelma rakenne



Kuva 1: UML-mallinnus

Ohjelman rakenteen esittely:

1) **Balloon**
   Tämä luokka perii ScalaFX:n Circle luokan ominaisuudet sekä lisää siihen muutaman lisäominaisuuden. Kyseessä on UI-komponentti, jolla mallinnetaan pelissä kulkevia palloja.

2) **Bullet**
   Tämä luokka perii ScalaFX:n Circle luokan ominaisuudet sekä lisää siihen muutaman lisäominaisuuden**.** Kyseessä on UI-komponentti, jolla mallinnetaan tornien ampumia panoksia.
3) **GhostPoint**
   Tämä luokka perii ScalaFX:n Circle luokan ominaisuudet sekä lisää siihen muutaman lisäominaisuuden**.** Kyseessä on UI-komponentti, jota hyödynnetään, kun käyttäjä haluaa rakentaa oman tasonsa.

4) **GhostRange**

Tämä luokka perii ScalaFX:n Circle luokan ominaisuudet sekä lisää siihen muutaman lisäominaisuuden**.** Kyseessä on UI-komponentti, jota hyödynnetään uusien tornien asettamista. Ympyrä, joka muodostuu ruudulle näyttää uuden tornin ampumaetäisyyden.

5) **GhostTower**
   Tämä luokka perii ScalaFX:n Rectangle luokan ominaisuudet sekä lisää siihen muutaman lisäominaisuuden**.** Kyseessä on UI-komponentti, jota käytetään, kun pelaaja haluaa asettaa uuden tornin ruudulle. GhostTower on väliaikainen torni, joka liikkuu hiiren mukana, kunnes pelaaja asettaa tornin ja GhostTower häviää. Tällöin GhostTowerin sijalle näytöllä tulee Tower

6) **Tower**
   Tämä luokka perii ScalaFX:n Rectangle luokan ominaisuudet sekä lisää siihen muutaman lisäominaisuuden**.** Kyseessä on UI-komponentti, jolla mallinnetaan torneja ruudulla. Lisäksi luokassa on pelilogiikkaan kuuluvia muuttujia.
   - downTime kertoo kuinka kauan kestää, että uusi luoti voidaan lähettää.
   - cooldown on ajastin downTimelle
   - range kertoo tornin ampumisalueen
7) **Constants**
   Constants objekti sisältää vain var- sekä val-muuttujia ja sen ideana on helpottaa pelilogiikan kehittämistä. Sen avulla voidaan muun muassa helposti vaihtaa UI- komponenttien kokoja.
8) **fileHandler**
   FileHandler objekti sisältää tiedostonkäsittelyyn kuuluvia funktioita. Tämän objektin avulla käyttäjän saamat tulokset tallennetaan tekstitiedostoon. Objektin avulla voidaan myös lukea samaa tekstitiedostoa, ja näin tehdään esimerkiksi, kun pelaaja siirtyy käyttöliittymässä *näkymään 6.*
9) **Game**
   Game luokan avulla voidaan luoda uusi game-olio, jota tarvitaan aina, kun aloitetaan uusi peli. Tämän luokan avulla pidetään kirjaa kaikista yksittäisen pelin tapahtumista ja se sisältää muun muassa kaikki pallot, luodit sekä tornit. Lisäksi luokka sisältää polun, jota pitkin pallot kulkevat.
10) **MapFiles**
    MapFiles objekti sisältää tiedostonkäsittelyyn kuuluvia funktioita. Objektin avulla tallennetaan sekä haetaan tietoa käyttäjän itse tekemistä tasoista. Se sisältää kolme functiota, joista yksi kirjoittaa tekstitiedostoon ja kaksi muuta lukevat siitä.
11) **Ticker**
    Ticker luokan avulla mallinnetaan ajankulkua pelissä.
12) **MainFX**
    MainFX objecti on koko ohjelman tärkein tiedosto. Se toimii ohjelman ns. päätiedostona ja kuten UML-kaaviostakin näkyy, siihen liittyy lähes jokainen muu luokka jollakin tapaa. MainFX:ssä yhdistyy käyttöliittymä sekä pelilogiikka. Yksi tärkeimpiä funktioita, joita se sisältää on newScene, jonka avulla tehdään uusi näkymä käyttöliittymässä. Kyseistä funktiota käytetään hyödyksi jokaisessa muussa funktiossa, jossa tehdään uusi ruutu, kuten: customMapsscreen, mapMaker ja leaderboard. Pelilogiikkaa käsittelevät funktiot löytyvät luokasta muuttujan ticker alta, noin puolestavälistä tiedostoa. Nämä funktiot käsittelevät samalla pelissä tapahtuvia asioita ja päivittävät samalla käyttöliittymää. Ne tekevät yhteistyötä myös runsaasti Game-olion kanssa. Niistä tärkein on playRound funktio, joka kutsuu jokaista muuta pelilogiikkafunktiota pelin edetessä. Aivan tiedoston lopusta löytyvät kaikki käyttöliittymän nappuloiden toiminnot.

Vaihtoehtoiset ratkaisumallit/ muuta

Yksi suurimmista muutoksista minkä ohjelman rakenteeseen olisi voinut tehdä on MainFX:n hajoittaminen useampaan tiedostoon tai joidenkin sen sisältämien toimintojen delegoiminen esimerkiksi Game-luokkaan ja pelin komponenttien: Tower, Balloon… luokkiin. Tämän myötä luokkajaosta olisi voinut tulla selkeämpi ja MainFX tiedosto olisi sisältänyt lähes pelkästään käyttöliittymään liittyviä toimintoja. Myös ohjelman tulkitseminen ja jatkokehitys olisi ollut huomattavasti helpompaa, jos yhdessä tiedostossa ei olisi ollut noin 500 riviä koodia, kun taas toisissa alle 100. Koska projekti toteutettiin yksin, ei tämä tuottanut haasteita ohjelman rakentamisessa, mutta jos ohjelmaa olisi tehnyt useampi henkilö tai ohjelmaa lähdettäisiin jatkokehittämään jonkun muun toimesta, olisi se suhteellisen vaikealukuista.

Alkuperäiseen suunnitelmaan verrattuna luokkia on tullut huomattavasti lisää. Tämä johtuu siitä, että ohjelmaa suunniteltaessa ei ollut otettu kaikkea vielä huomioon ja esimerkiksi Ghost- alkuiset luokat osoittautuivat erittäin hyödyllisiksi käyttöliittymää tehdessä. Alkuperäisessä luokkajaossa oli myös eriytetty luokat gameState, window: game sekä Arena, mutta lopullisessa ohjelmassa toteutus tehtiin Game-olion sekä MainFX-tiedoston kautta. Arena luokan poisjättäminen oli sinänsä fiksua, sillä alkuperäinen toimintatapa tuntui ohjelmaa tehdessä kömpelöltä, sillä Arenan ja gameStaten sovittaminen yhteen ei tuntunut sopivalta.

5. Algoritmit

Ohjelmani sisältää useita algoritmeja ja monet niistä käsittelevät itse pelissä tapahtuvia muutoksia. Tärkeimpiä niistä ovat: **polun määrittäminen, pallojen liike sekä tornien ampuminen.**

Polun määrittäminen

Polun määrittäminen ohjelmassa tapahtuu listan avulla, joka sisältää koordinaatteja käyttöikkunasta. Koordinaattien avulla ohjelma muodostaa polun ns. käännöspisteet, joita käytetään apuna muun muassa käyttöliittymää rakentaessa sekä pallojen liikkeen mallintamisessa. Polun määrittämistä mallinnetaan mm. MainFX tiedostossa funktiossa newScene.

Koska polun muodostaminen listasta on automatisoitu, eikä se sisällä mitään muita sääntöjä, kuin sen että se alkaa vasemmalta ja loppuu oikealle, voi pisteitä olla mielivaltainen määrä, mielivaltaisissa paikoissa. Tämä mahdollistaa esimerkiksi sen, että käyttäjä voi vapaasti tehdä omanlaisia karttoja, välittämättä mistään säännöistä.

Pallojen liike

Pallot lähetetään kartalle yksinkertaisen algoritmin avulla, jossa otetaan huomioon monesko, kierros on käynnissä. Algoritmi hyödyntää Scala.util.Randomia, jotta pallojen lähettämisessä tulisi eroavaisuuksia jokaisella pelikierroksella. Pallojen liikettä mallinnetaan MainFX tiedostossa funktiossa moveBalloon

Pallot lähtevät liikkeelle ikkunan vasemmasta reunasta ja lähtöpiste määräytyy polun ensimmäisen pisteen y-koordinaatin mukaan. Tämän jälkeen pallot liikkuvat aina seuraavaa polun käännöspistettä kohti. Liikkumisessa on hyödynnetty Pythagoran lausetta, jotta pallot liikkuvat tasaisesti kohti seuraavaa käännöspistettä. Kun viimeinen käännöspiste on saavutettu liikkuvat pallot kohti oikeaa reunaa, kunnes ne poistuvat kokonaan näytöltä. Kun pallo häviää ruudulta, poistuu se myös koko pelistä.

Tornien ampuminen

Tornien ampuminen käsittelee montaa eri tapahtumaa samanaikaisesti. Aina, kun aikaa kuluu, eli jokaisella ”tikillä” ohjelma tarkastaa jokaisen tornin osalta, onko sen ampumaetäisyydellä palloja, jotka ovat saman värisiä kuin torni. Jos on, niin ohjelma luo Bullet-luokasta uuden ilmentymän tornin keskipisteeseen. Tämän jälkeen torni ja uusi luoti eivät riipu mitenkään toisistaan.

Kun uusi luoti on luotu, annetaan sille kohde, eli yksi pallo. Luoti kulkee kohti palloa ja sen liikettä on mallinnettu pallojen liikkeen mukaan pythagoran lauseen avulla.

Kun torni on luonut uuden luodin, käynnistyy ohjelmassa uusi ajastin, jonka aikana torni ei pysty luomaan uusia luoteja, vaikka sen ampumaetäisyydellä olisi sopivia palloja. Uusi luoti voidaan muodostaa vasta sitten, kun ajastin on päättynyt.

6. Tietorakenteet

Ohjelmassa käytetään eniten Buffereita sekä Vectoreita. Bufferia kätetään muun muassa polun määrittämisessä ja se on muuttuvatilainen. Bufferi olisi yhtä hyvin voinut olla muuttumaton, sillä polku ei muutu sen jälkeen, kun se on kerran muodostettu. Bufferia käytetään myös pallojen, tornien sekä luotien kirjassa pidossa. Näissä tapauksissa muuttuvatilainen Bufferi on oiva valinta, sillä nämä kokoelmat muuttuvat pelin edetessä.

Vectoreita käytetään tiedostojen hallinnassa, kuten mapFiles tiedostossa. Tässä käytetään muuttumattomia kokoelmatyyppejä, sillä tiedostoon tallentaessa ei kokoelman itse sisältö muutu.

7. Tiedostot ja verkossa oleva tieto

Ohjelma käsittelee tekstitiedostoja, joita käytetään hyödyksi tulosten tallentamiseen ja uusien tasojen tekemiseen sekä pelaamiseen. Koska tiedostoihin kirjoittaminen on tarkoitettu tapahtuvan itse ohjelman kautta (ei käsin), rajataan käyttäjän tekemiä virheitä.

Käyttäjän saamat tulokset kirjataan seuraavassa muodossa tiedostoon Results.txt:

<Käyttäjän syöttämä nimi> <välilyönti + ”ROUND:” + välilyönti> <käyttäjän tulos>

esimerkki tapaus:

Sampo ROUND: 3

Käyttäjän itse tehdyt kartat tallennetaan seuraavassa muodossa tiedostoon SaveMap.txt:

<”###”><Käyttäjän syöttämä nimi>

<koordinaattipiste> <”, ”> <koordinaattipiste>

<koordinaattipiste> <”, ”> <koordinaattipiste>

<koordinaattipiste> <”, ”> <koordinaattipiste>

…

<----->

esimerkkitapaus:

###Testi

242, 280

232, 12

321, 654

33, 22

---

8. Testaus

Kuten projektisuunnitelmassa kirjoitin, testaus tapahtui lähinnä käyttöliittymän kautta. Käyttöliittymän kautta testasin pelilogiikkaa, sillä aloitin projektini luomalla käyttöliittymän. Tämä nopeutti projektini edistymistä, sillä näin aina tarkasti, miten eri komponentit käyttäytyivät ruudulla ja niiden hienosäätäminen oli nopeaa ja helppoa. Toinen testaustapa oli komentoriviin printtaaminen. Tämä oli käytännöllistä silloin, kuin asiat olivat menneet pahasti pieleen ja esimerkiksi käyttöliittymässä ei näkynyt ollenkaan sinne asettamaani komponenttia.

Vaikka testiohjelmat voivat olla hyödyllisiä monissa tapauksissa, en ole kokenut tarvetta käyttää erillisiä testiohjelmia ohjelmani testaamisessa.

Olen testannut ohjelmaani käyttöliittymän ja komentorivin kautta, mikä on osoittautunut tehokkaaksi tavaksi varmistaa ohjelman toimivuus. Tällä tavalla olen voinut simuloida käyttäjän käyttäytymistä ja tarkastella ohjelman käyttäytymistä reaaliajassa. Lisäksi olen tehnyt manuaalisia testejä ohjelman eri ominaisuuksille ja toiminnallisuuksille, kuten tornien asettamiselle, pallojen liikkumiselle ja torjumiselle.

Vaikka testiohjelmat voivat olla erittäin hyödyllisiä monimutkaisempien ohjelmien testaamisessa, olen kokenut, että ohjelmani yksinkertaisempi luonne ei vaadi niiden käyttöä. Olen kuitenkin tietoinen siitä, että hyvät testikäytännöt ovat tärkeitä ohjelman toimivuuden ja laadun kannalta, ja olisin itsekkin voinut hyödyntää testiohjelmia joissakin tapauksissa.

Yhteenvetona voidaan todeta, että en ole käyttänyt testiohjelmia, koska olen kokenut manuaalisten testien riittävän varmistamaan ohjelman toimivuuden. Kuitenkin, jos ohjelman monimutkaisuus kasvaa olisi testiohjelmien käyttö hyvä lisä.

9. Ohjelman tunnetut puutteet ja viat

Puutteita suunnitelmaan verrattuna:

Yleissuunnitelmaani verrattuna toteutus muuttui. Olin suunnitellut tekeväni erilaisia puolustajia, joilla on erilaisia ominaisuuksia. Päädyin kuitenkin supistamaan puolustajien toiminnallisuuksia, sillä niiden lisääminen olisi kasvattanut työmäärää huomattavasti. Lisäksi olin suunnitellut ohjelman grafiikoiden olevan paljon parempia, kuin lopputuloksessa. Projektin edetessä mahdollisten kehityskohteitten määrä kuitenkin kasvoi vähintään yhtä nopeasti, kuin projekti eteni, joten päädyin keskittymään muuhun, kuin ohjelman ulkonäköön.

Muita ongelmia:

- Jos Custom map tekstiruutuun kirjoittaa jonkun muun nimen, kuin olemassa olevan tason nimen, heittää ohjelma exceptionin. Ohjelman käyttö voi kuitenkin jatkaa normaalisti
- Jos käyttäjä aloittaa uuden pelin, vanhan ollessa vielä käynnissä, niin uuden pelin pallot lähtevät liikkeelle ennen kuin käyttäjä painaa start painiketta
- Jos käyttäjä on jo pelannut yhden pelin ja käyttänyt Change color painiketta, käyttäjän aloittaessa uuden pelin, näyttää Add Tower painike väärää väriä
- Make a map ruudussa käyttäjän on mahdollista peittää kaikki nappulat, jolloin käyttäjä voi jäädä jumiin kyseiseen ruutuun. Käyttäjän on myös mahdollista peittää ohjeet, siten että niitä ei voi enää lukea.
- Jos käyttäjä luo useita saman nimisiä karttoja, voi hän pelata niistä vain yhtä
- Jos käyttäjä unohtaa painaa enteriä kirjoittaessaan omaa karttaa, tulee sen nimeksi aina No Name. Koska No Name nimisiä karttoja voi syntyä useita, voi vain yhtä niistä pelata
- Jos tekstitiedostoihin kirjoittaa käsin, voi ohjelmassa syntyä virhetilanteita. Näin ei kuitenkaan tulisi tapahtua, sillä tekstitiedostoihin kirjoittaminen tapahtuu käyttöliittymän kautta.
- Tiedostoihin kirjoittamisessa voi tapahtua odottamattomia virheitä, jos käyttäjä käyttää erikoismerkkejä
- Käyttöliittymän ikkunan kokoa pystyy muuttamaan itse. Jos näin tekee pelatessa peliä, niin kartta jatkuu pidemmälle, kuin alkuperäisessä näytössä. Silti pallot häviävät alkuperäisen ruudun päätöskohdassa. Jos ruutua kerran suurentaa tai pienentää, on vaikeaa löytää alkuperäinen kohta, johon kartta loppuu.

10. 3 parasta ja 3 heikointa kohtaa

3 parasta:

1. Omien karttojen tallentaminen ja pelaaminen
   On hauskaa, että ohjelmassani käyttäjä voi toteuttaa omanlaisia karttoja. Myös uusien karttojen tekeminen sujuu helposti käyttöliittymän kautta, eikä käyttäjän tarvitse itse kirjoittaa tekstitiedostoon mitään. Tällöin käyttäjä säästyy syntaksin opettelulta ja ongelmatilanteiden arvuuttelulta.
2. Pelin haastavuus
   Peli on suhteellisen vaikea ja nopea tempoinen, jolloin käyttäjä pääsee kokeilemaan eri taktiikoita ja joutuu miettimään esimerkiksi, milloin käyttää Upgrade painiketta.
3. Navigoiminen käyttöliittymässä
   Ohjelman käyttöliittymä on suhteellisen suoraviivainen, eikä monessa kohdassa tarvitse arvuutella miten pääsee eteenpäin. Varsinkin Menu- valikko helpottaa navigoimista.

3 heikointa:

1. Tiedoston käsittelyn virhetilanteet
   Tiedoston käsittelyssä löytyy joitakin virhetilanteiden hallitsemistapoja, mutta ne ovat suhteellisen alkeellisia, eivätkä hallitse kaikkia virhetilanteita asiallisesti.
2. Käyttöliittymän ulkonäkö
   Käyttöliittymän tyyli on suhteellisen alkeellinen, eikä pelin grafiikat ole mitenkään loisteliaat. käyttöliittymästä ja varsinkin pelistä olisi voinut tehdä huomattavasti hauskemman, jos siihen olisi lisännyt kuvia, värejä, eri fontteja ja ehkä joitakin animaatioita.
3. Ohjelman rakenne

Ohjelman rakenne on muuten suhteellisen hyvä, mutta MainFX tiedostoa tulisi parsia hieman. Kuten aiemmin mainitsin tämä helpottaisi ohjelman jatkokehittämistä ja sen luettavuutta.

11. Poikkeamat suunnitelmasta, toteutunut työjärjestys ja aikataulu

Toteutin projektin suurin piirtein samassa järjestyksessä, kuin olin suunnitellutkin. Aloitin käyttöjärjestelmän miniversiosta, jonka jälkeen siirryin peli komponenttien tekemiseen yksikerrallaan. Tämän jälkeen aloin yhdistelemään pelikomponentteja sekä niiden toiminnallisuuksia toisiinsa. Lopulta aloin hiomaan käyttöliittymää sekä tiedostonkäsittelyä. Lopuksi vielä päädyin lisäämään omien karttojen tekemisen sekä niiden tallentamisen ohjelmaani.

Projektini ajankäyttö täsmäsi yleisellä tasolla odotuksiani. Kuitenkin huomasin, että varsinkin projektin alkuvaiheeseen eli ohjelmointi ympäristön rakentamiseen meni odotettua pidempään, enkä saanut kaikkia palasia toimimaan haluamallani tavalla. Esimerkiksi Scala Swingin käyttäminen osoittautui huonoksi vaihtoehdoksi verrattuna ScalaFX:ään, joten päädyin vaihtamaan noin kymmenen tunnin jälkeen Swingin FX:ään. Lisäksi päädyin ohjelmoimaan pääasiassa Scala 2 ympäristössä, sillä Scala 3 käytössä osoittautui joitakin ongelmia.

Toinen kohta, joka kesti odotuksiani pidempään oli projektin viimeistely. Muutama viikko ennen asettamaani deadlinea luulin olevani käytännössä valmis, mutta valmistautuessani palauttamaan projektin, huomasinkin sen sisältävän monia pieniä bugeja sekä hiomiskohtia. Tässä kohtaa aikatauluni suunnitelmassa ei osunut kohdilleen, vaan viimeistelyyn olisi pitänyt jättää enemmän aikaa. Asetin deadlineni kuitenkin lisäpistepalautukseen, joten pystyin jatkamaan projektin hiomista vielä viikon extraa.

12. Kokonaisarvio lopputuloksesta

Kokonaisuudessaan olen tyytyväinen projektiini. Kyseessä oli ensimmäinen suurempi kokonaisuus, jonka olen luonut alusta asti itse ja olen oppinut paljon tehdessäni sitä. Erityisen tärkeäksi huomasin projektin suunnitteluvaiheen, joka selkiytti paljon mitä teen seuraavaksi.

Eniten pidän siitä, kuinka hyvin sain toimimaan itse pelin ja kuinka sitä voi pelata lähes mutkattomasti. Kehityskohteena omasta mielestäni on sen grafiikat, sillä mielestäni hyvä pelikokemus pitää aina sisällään myös ulkonäön. Olen myös ylpeä käyttöliittymän kokonaisuudesta ja sen monista eri toiminnoista. Lisäksi myös siitä, kuinka helppokäyttöinen käyttöliittymä kokonaisuudessaan on.

Jatkokehityksen kannalta projektin koodia kannattaisi refaktoroida MainFX tiedoston kannalta, mutta muuten muutosten tekeminen ei aiheuttaisi ongelmia. Koodiin on lisätty runsaasti kommentteja sekä tärkeimpiä muuttujia on helppoa vaihtaa Game-luokasta sekä Constants objektista.

Jos jatkokehittäisin itse projektia, olisi aivan ensimmäisenä mielessä käyttöliittymän grafiikoiden päivittäminen ja tyylittely. Tämän jälkeen lähtisin kehittämään peliä ja varsinkin torneja sekä niiden ominaisuuksia. Peliin voisi lisätä esimerkiksi erilaisia torneja, joista mainitsin alkuperäisessä suunnitelmassani.

Jos lähtisin tekemään projektia uudelleen, muuttaisin luokkajakoa siten, että käyttöliittymä olisi erillään pelilogiikasta. Lisäksi tekisin enemmän apufunktioita esimerkiksi käyttöliittymän uusien ikkunoiden tekemiseen, jolloin kehittäminen sujuisi huomattavasti nopeammin ja uusien toimintojen tekeminen olisi mukavampaa. Lisäksi pyrkisin panostamaan siihen, että tekisin projektia pikkuhiljaa, enkä viikon välein suurempina rupeamina.

13. Viitteet

-ScalaFX tutoriaaleja: https://youtube.com/playlist?list=PLKUBmwYhjbwSxOjw5tBt8Lfit39tP1jyd

-ScalaFX toimintoja: https://javadoc.io/doc/org.scalafx/scalafx_2.12/latest/index.html

-Yleisiä Scalan toimintoja: https://plus.cs.aalto.fi/o1/2022/wNN/scala/

14. Liitteet

Dokumentti sisältää:

Näkymät 1-6 sekä *Kuvan 1.*

Liitteet ovat dokumentin seassa kohdissa 3 ja 4.

Ohjelmakoodi erillisessä tiedostossa.