Tornipuolustuspeli

Jere Stigell
868747
Tietotekniikka
1. Vuosikurssi
15.4.2022

2. Yleiskuvaus

Tornipuolustuspeli jossa pelaaja asettaa kartalle torneja jotka puolustavat polun päässä olevaa kohdetta. Peli täyttää kaikki 3 vaatimusta jotka ovat lueteltu kurssipohjalla: graafinen käyttöliittymä jossa grafiikkana käytetään geometrisia muotoja ja peli on käyttäjän laajennettavissa omilla kartoilla.

3. Käyttöohje

Peli käynnistetään Game.scala tiedostosta. Peli aukeaa default karttaan (kts. liitteen kuvat) josta voi suoraan aloittaa pelaamisen. Pelaaja voi myös ladata oman kartan/tallennetun pelin painamalla nappia “Load Game” ja hakemalla hakemistosta sav-päätteinen tiedosto. Pelin voi tallentaa painamalla “Save Game” nappia jolloin valitaan polku ja nimi sav-tiedostolle. Enemmän tietoa tallennuksesta löytyy SaveData kansion ReadMe.txt tiedostosta.

Torneja voi ostaa kahta erilaista tornin tyyppiä vastaavaa nappia painamalla. Uusi aalto käynnistetään “Start new wave!” nappia painamalla. Nappien vieressä on pelaajan statsit josta selviää nykyinen aalto, paljonko pelaajalla on rahaa ja health pointteja. “Quit” napista voit sulkea koko pelin. Kun kartan kaikki aallot on voitettu voi pelaaja siirtyä seuraavalle kartalle. Mikäli pelaaja kuolee, voi kartan aloittaa uudestaan tai lopettaa pelin kokonaan. Kun kaikki kartat on voitettu peli loppuu ja sen voi vain sulkea.

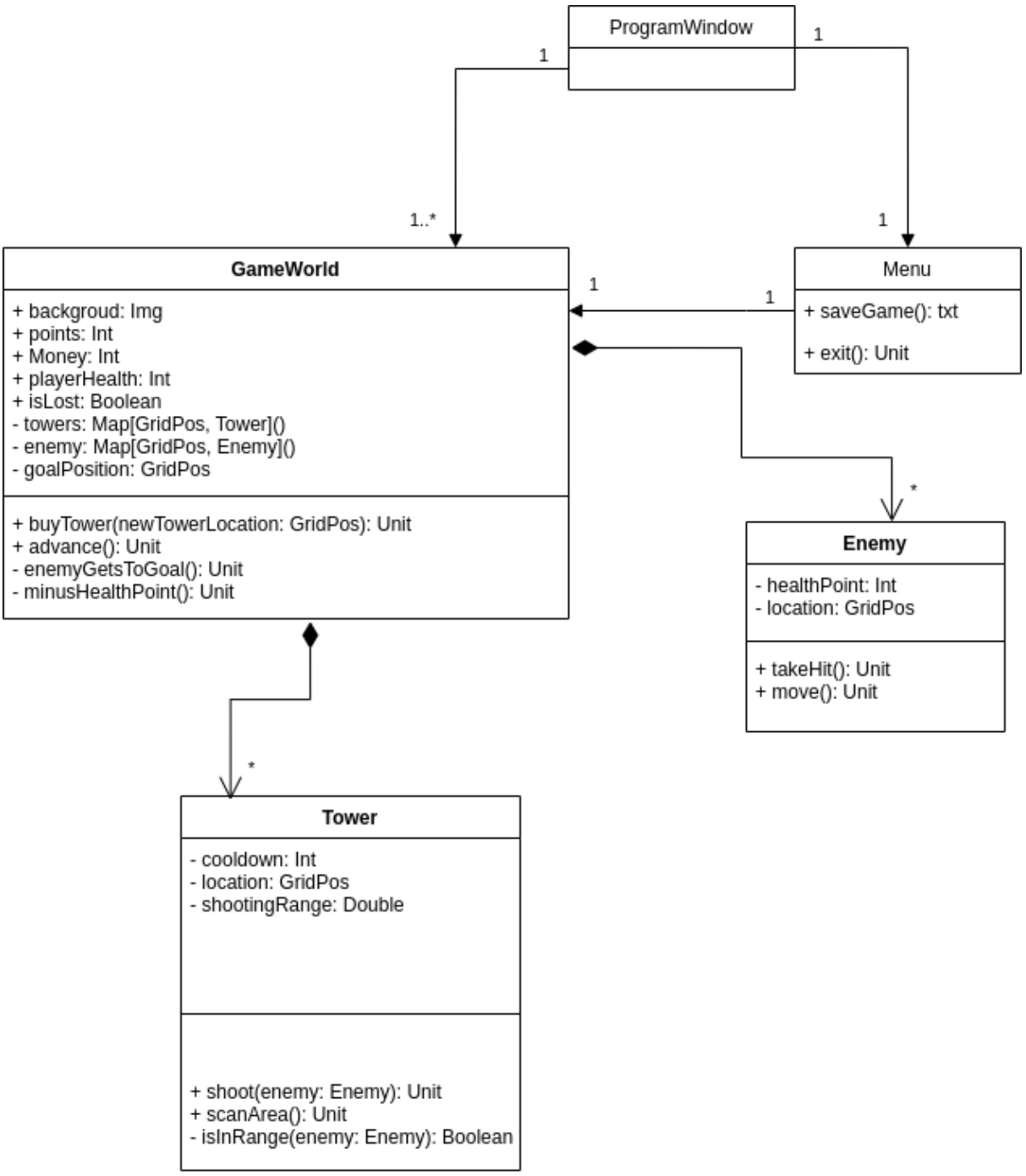
4. Ohjelman rakenne

Peli koostuu torni- ja vihollisolioista jotka täyttävät pelikentän (**Area** objekti). Pelin ikkuna ja kenttä nappuloiden piirretään **Game** objektissa. Torneja on kahta eri tyyppiä pieni (**SmallTower**) ja iso (**BigTower**) jotka jatkuvat abstraktia luokkaa Tower. Vihollisia on myös kahta eri tyyppiä, pieniä (**SmallEnemy**) ja isoja (**BigEnemy**) jotka jatkuvat abstraktia luokkaa Enemy.

Update objekti päivittää pelin eri toimintoja kun sitä kutsutaan. Sitä kutsutaan monesta eri paikasta kun halutaan päivittää pelin sisäistä tilaa. Esimerkiksi kun torni on ostettu, vihollinen on tuhottu, vihollinen on saavuttanut reitin päätepisteen, health pointsit ovat loppuneet, jne. Update hoitaa tilojen päivityksen ja tarkastaa onko peli vielä kesken, voitettu vai hävitty.

Pelin I/O:ssa voidaa peliä kirjoittaa/tallentaa tiedostoon ja lukea tiedostosta. **Loader** ja **Reader** hoitavat tallennetun tiedoston lukemisen ja uuden kartan käynnistämisen. **Saver** hoitaa pelin tallentamisen erilliseen tiedostoon. Saver hakee Area:n vihollis- ja torniolioiden tiedot tallennusta varten. Ladatessa peliä Loader asettaa viholliset ja tornit kentälle, sekä asettaa uuden kartan ja päivittää vihollis reitit yms. Update objektin avulla. Virheellisen tallennustiedon tilanteessa asianmukaiset varoitukset annetaan pelaajalle ja asetetaan peli tietyiltä osin oletusarvoille.

WaveController pitää huolen, että vihollis aallot käynnistetään oikein ja, että sitä ei käynnistetä jos peli on voitettu. **Player** objektissa pidetään kirjaa raha ja health point määrästä. Player kommunikoi suoraan tornien ja vihollisten kanssa sekä Area:n välityksellä.



Alkuperäinen UML-kaavio, joka ei vastaa enää ohjelman todellista rakennetta.

5. Algoritmit

Kartta generointi:

Koska pelaaja voi ladata omia karttoja ilman, että hänen tarvitsee määritellä polkua (aikaa vievää, ei hauskaa..) piti minun keksiä algoritmi joka laskee Array(Array(Int)) tyyppisestä muuttujasta oikean polut aloituspisteesta (kartalla numero 2) polkua

pitkin (Kartalla numero 1) päätepisteeseen (kartalla numero 3) jossa sijaitsee kohde jota pelaajan tulee puolustaa.

Funktio *enemyInitialLocation* etsii Array kartalta sen pisteen jossa on aloituspiste eli numero 2. Koska numero 2 esiintyy kartalla vain kerran, ei kartta tarvitse kuin kerran loopata läpi ja etsiä aloituspisteen koordinaatit.

Funktio *findInitialDirection* etsii ensimmäisen suunnan johon vihollisen on tarkoitus liikkua kun se tulee kartalle. Toisin sanoen etsitään aloituspisteen ympäriltä (vaaka- ja pystysuunnassa) numero 1. Kartat eivät saa sisältää kahta numero 1:stä numero 2 lähellä. On siis olemassa tietyt säännöt kartan muodostamiselle.
Funktio *findBannedAreas* kartoittaa polun koordinaatti välin. Näin voidaan varmistaa, että pelaaja ei voi asettaa tornia polun keskelle. Polulle lasketaan aloituspiste (x1, y1) ja lopetuspiste (x2, y2). Yhdessä nämä muodostavat kartalle suorakulmion jonka sisälle tornia ei saa laittaa.

Funktio *enemyPath* etsii koko kartan läpi vihollisen reitin ja suuntavektorit. Reitti on kokoelma käännöspisteitä kartalla. Reitti kertoo missä pisteessä käännytään ja suuntavektori uuden suunnan johon mennään käännöksestä. Reitti lasketaan etsimällä lähtöpisteestä aina seuraava 1 vaaka- tai pystysuunnassa. Käännöksissä otetaan reittipiste ja vektori muistiin. Toistetaan niin kauan, että saavutetaan kartan loppupiste joka on merkitty karttaan numerolla 3.

PathFinder.scala tiedoston algoritmit eivät ehkä ole kaikkein resurssi tehokkaimpia mutta en nähnyt syytä lähteä optimoimaan niitä tämän enempää koska kartan polut ja suuntavektorit muodostetaan vain kerran kartan ensimmäisen latauksen aikana. Ei siis ole kovin paljon väliä vaikka kartan datan muodostamisessa menisikin muutama sata millisekuntia.

Tornit:
Torni skannaa omaa lähialuettaan perustuen omaan “range” arvoon. Mikäli tämän rangen sisällä sijaitsee vihollisia, torni hyökkää havaituista vihollisista ensimmäisen kimppuun. Torni käy läpi kaikki viholliset ja vertailee omaa sijaintiaan vihollisen sijaintiin huomioiden rangen. Torneja ei voi myöskään laittaa toistensa päälle.

6. Tietorakenteet

Ohjelmassa käytetään pääasiassa mutable.Buffer kokoelmatyyppiä johon säilötään kartan tornit ja viholliset, sekä näiden reitit.

7. Tiedostot ja verkossa oleva tieto

Pelin eteneminen voidaan tallentaa tiedostoksi joka voidaan ladata myöhemmiin uutta pelikertaa varten. Pelaajan on myös mahdollista tehdä omia karttoja joissa on pelaajan määräämä määrä vihollisia ja aaltoja. Alla kuvakaappaus tallennustiedoston rakenteesta:

```
# The game map is formed under the #MAP.
# Each row represents row on a grid style map that has dimensions m x n.
# Minimum recommended size is 5x5
# Symbols are:
# 0 = Area where towers are placed
# 1 = Enemy path
# 2 = Enemy start point, must be on the side of the map
# 3 = End of the enemy path, player must protect this area

# Enemies are added one enemy type per row under the #ENEMY.
# 0 = small enemy
# 1 = big enemy
# "enemy type" [0 or 1] / "amount of enemies" [0 - 100]

# Towers are added one enemy type per row under the #TOWER.
# 0 = small tower
# 1 = big tower
# "tower type" [0 or 1] / "amount of towers" [0 - 100]

# Towers locations are added under #TOWERLOCATION. Each tower has one
# location coordinate x and y which are separated from others with semicolon ";".
# Amount of locations must match with the amount of specific towers and
# location coordinates should not go outside the map perimeter.
# First row of locations is for small towers.
# Second row of locations is for big towers.

# Set the healthpoints under the #HEALTH.
# Must be positive integer.
# "Current health" / "Maximum health"

# Set the money in the bank under the #MONEY.
# Must be positive integer.

# Set the number of waves under the #WAVES.
# Must be positive integer.
# "Current wave" / "Maximum waves"

### EXAMPLE SAV-FILE ###

#MAP
0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
3,1,1,1,0,1,1,1,0,0,0,0,1,1,1,2
0,0,0,1,0,1,0,1,0,0,0,0,1,0,0,0
0,1,1,1,0,1,0,1,1,1,0,0,1,0,0,0
0,1,0,0,0,1,0,0,0,1,0,0,1,0,0,0
0,1,0,0,0,1,0,0,0,1,0,0,1,0,0,0
0,1,0,0,0,1,0,0,0,1,1,0,1,1,1,0
0,1,1,1,1,1,1,0,0,0,0,1,0,0,0,1,0
0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,1,0
0,1,1,1,1,1,1,1,1,1,1,1,0,0,0,1,0
0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,1,0
0,1,0,1,1,1,0,1,1,1,0,1,1,1,1,1,0
0,1,1,1,0,1,1,1,0,1,1,1,0,0,0,0
0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0

#ENEMY
0/8
1/8

#TOWER
0/2
1/2

#TOWERLOCATION
907.0,748.0;542.0,747.0;
350.0,395.0;175.0,401.0;

#HEALTH
90/100

#MONEY
310

#WAVES
1/2
```

Pelit siis tallennetaan sav-päätteiseen tiedostoon johon on kirjattu kartta, vihollistyypit ja niiden määrä, sekä nykyinen aalto ja aaltojen kokonaislukumäärä. # merkitys rivit ovat kommentteja jotka helpottavat kartan luomista. Useampi esimerkki tallennetusta tiedostosta löytyy kansiosta SaveData/SaveFiles.

8. Testaus

Lähinnä PathFinder.scala:lle kirjoitettiin yksikkötestejä joilla varmistettiin sen toimivuus mahdollisimman monen erilaisen kartan kanssa. Tämä myös helpottaa muutosten tekemistä algoritmeihin kun pystyin samantien kokeilla vieläkö se toimii halutulla tavalla. Ohjelma läpäisee kaikki yksikkötestit.

PathFinder:n yksikkötestejä lukuunottamatta muita automatisoituja testejä ei ollut, loppu testaus tehtiin pelaamalla. Viallisten tallennus tiedostojen käsittelyä voi testata lataamalla niitä kansiosta SaveData/InvalidSaveFiles. Eri tavalla vialliset tallennustiedostot ovat nimetty vian mukaan, esimerkiksi InvalidHealth.sav, siinä health pointit ovat väärin kirjattu omalle rivilleen. Toimivat tallennetut pelit löytyvät kansiota SaveData/SaveFiles.

9. Ohjelman tunnetut puutteet ja viat

Tunnettuja puutteita ja vikoja tällä hetkellä:

- PathFinderin polku heittää välillä hieman eli vihollinen ei ole polun keskellä. Tämä vika johtuu suurimmaksi osaksi siitä, että käyttäjä voi rakentaa niin ison tai pienen kartan kuin haluaa. Koska kaikki skaalautuu ohjelmaikkunan leveyden ja korkeuden mukaan tulee sinne pientä heittoa välillä. Paras korjaus olisi se, että kartan koko rajoitettaisiin tietylle välille jolloin liian suuria tai pieniä karttoja ei voisi ladata. Toisaalta nykyinen joustavuus tekee pelistä hauskan kokeilla miten suuria karttoja voidaan pyörittää.
- Polulle ei voi asettaa torneja mutta polun tunnistus heittää sen verran, että ulkokurviin voi tornin laittaa hieman polun päälle mutta sisäkurviin ei, pitäisi korjaantua jos kävisi polku algoritmia läpi ja lisäisi pieniä “korjauskertoimia”
- Range ei näy kartalla kun tornia asetellaan. Tämä olisi tärkeä kun pelaaja haluaa laittaa tornin strategisesti parhaalle paikalle, silloin on tärkeää nähdä

omin silmin sen range. Range voitaisiin esittää hieman läpinäkyvänä ympyränä tornin ympärillä.

- GUI layout ja menu. Nyt kaikki napit on ängetty yhteen paikkaan vaikka ne voisi olla myös siistimmin ja loogisemmin aseteltu. Myös päävalikko puuttuu, peli alkaa suoraan ensimmäisestä kartasta.

10. 3 parasta ja 3 heikointa kohtaa

3 heikointa:

- Tornilla ei ole kääntyvää piippua tai ampumisanimaatiota joilla se voisi indikoida pelaajalle, että se toimii ja mitä vihollista se juuri tähtää. Animaation lisääminen ei pitäisi olla suuri työ. Kääntyvä piippu vaatisi skannauksen aikana vihollisen sijainnin varmistamisen ja piipun kääntämisen siihen suuntaan.
- Ohjelman koodin rakenne. Koodia pitäisi refaktoroida kovalla kädellä ja luokka rakenteita uudistaa. Nyt on liian paljon kutsuja eri objektien välillä tai ne muokkaavat toistensa tietoja. Osa viittauksista saattaa mennä ristiin jolloin yhteen paikkaan arvo päivittäminen ei välttämättä päivity oikein kaikkialla. Yhden kohdan muuttaminen voi hajoittaa jonkun muun kohdan koodissa. Tämä on kriittinen kohta jos ohjelmasta halutaan tehdä helpommin laajennettava tulevaisuudessa.
- Useamman säikeen puute piirrossa ja tornien/vihollisten päivittämisessä. Suorituskyky kyykkää pahasti kun ruudulla on paljon vihollisia ja torneja. Kaikki toiminta tapahtuu yhden säikeen varassa eli ensin kaikki tornit skannaa ympäristöään vihollisten varalta, ampuvat niitä tarvittaessa ja sitten kaikki viholliset liikkuvat eteenpäin, jos ovat elossa. Suorituskykyä voisi parantaa myös muuttamalla tornien alueen skannausta niin, että vain ne tornit skannaavat aluettaan joiden lähellä tiedetään olevan vihollisia, on ihan turha skannata kartan toisessa päässä olevia torneja jos siellä ei ole vihollisen vihollista.

3 parasta:

- PathFinder. Ihan sama minkälaisen kartan pelaajan lataa, sille pystytään muodostamaan polku jolla voi pelata. Vaatii kuitenkin sen, että kartan tekijä

noudattaa muutamaa ehtoa kuten vain yksi aloitus- ja lopetuspiste, sekä polut eivät saa olla kylki kyljessä tai mennä toistensa ylitse.
- Käyttöliittymän joustavuus, GUI joustaa vastaamaan pelaajan luomaa karttaa. Tietty todella pienet kartat näyttävät kauheilta, mutta pelaaja voi tehdä yllättävänkin suuria karttoja.
- “cooldown”-indikaattori, kun torni ampuu. Vaikka tornilla ei olekaan ampumis- tai kääntymisanimaatoita. Jäähtymis-indikaattori viestii pelaajalle, että nyt torni on hyökännyt vihollista kohti ja odottaa kunnes voi ampua uudestaan.

11. Poikkeamat suunnitelmasta, toteutunut työjärjestys ja aikataulu

Alussa olin hieman jäljessä aikataulusta koska aloitin hieman myöhässä projektin tekemisen, palautin ensimmäisen progress raportin viikon myöhässä koska meni viikot sekaisin. Sain kuitenkin otettua aikataulun kiinni maaliskuun lopulla. Muuten etenin täysin aikataulutuksen mukaisesti ominaisuuksien kanssa. Alkuperäisestä suunnitelmasta poikettiin lähinnä päävalikon ja musiikin osalta. Päävalikkoa ei tehty vaan peli aukeaa suoraan ekalle kartalle. Musiikkia tai ääniefektejä ei lisätty ollenkaan.

Opin aikataulutuksesta sen, että on todella vaikea arvioida kuinka paljon mikäkin osa tarvitsee työtä. Mitä paremmin projektisuunnitelma on pyritty tekemään sitä vähemmän syntyy painetta projektin koodaus vaiheessa kun ominaisuuksia voi kirjoittaa yksi kohta kerrallaa. On hyvä jakaa projekti mahdollisimman pieniin osiin jotta sitä on helpompi tehdä.

12. Kokonaisarvio lopputuloksesta

Ohjelma on toimiva tornipuolustuspeli. Pelaaja voi tehdä omia karttoja ja tallentaa etenemistään tiedostoon. Olen tyytyväinen, että sain kasaan näinkin toimivan pelin jota pelaaja pystyy jopa hieman kustomoimaan.

Ongelmia on varsinkin koodin laadussa ja pelin suorituskyvyssä. Koodia kirjoitettiin vuoronperään hieman sieltä sun täältä ja aina ei ollut tarkkaa suunnitelmaa miten toteuttaisin jonkin asian, etsin vain nopeimman ratkaisun aikataulussa pysymisen

takia. Ei hyvä ratkaisu pelin laajentamisen kannalta. Suorituskyky on heikko jos pelissä on paljon torneja ja vihollisia, moni säie tukea ei ole joten kun kaikki tehdään yhdellä säikeellä peli hidastuu silminnähtävästi tietyn torni ja vihollis määrän jälkeen.

Myös pelin vihollisten nopeus, niiden tekemä vahinko pelaajalle, tornien hinnat ja niiden tekemä vahinko vihollisille pitäisi optimoida paremmin jos haluaa parantaa pelikokemusta. Nyt suurin osa em. arvoista on vain hatusta heitettyjä ja peli tuntuu tietyiltä osin liian helpolta tai todella vaikealta.

Jos aloittaisin projektin nyt alusta, tekisin alkupään UML kaavion paljon tarkemmin ja ylipäätänsä suunnittelisin asiat tarkemmin ennen kuin alan kirjoittamaan koodia. Tämän projektin aikana selkeni todella hyvin, että hyvin suunniteltu on puoliksi tehty. Hyvin suunniteltuna ei olisi tulisi tehtyä niin paljon huonoja koodaus päätöksiä joita on jälkeenpäin todella vaikea korjata rikkomatta koko peliä.

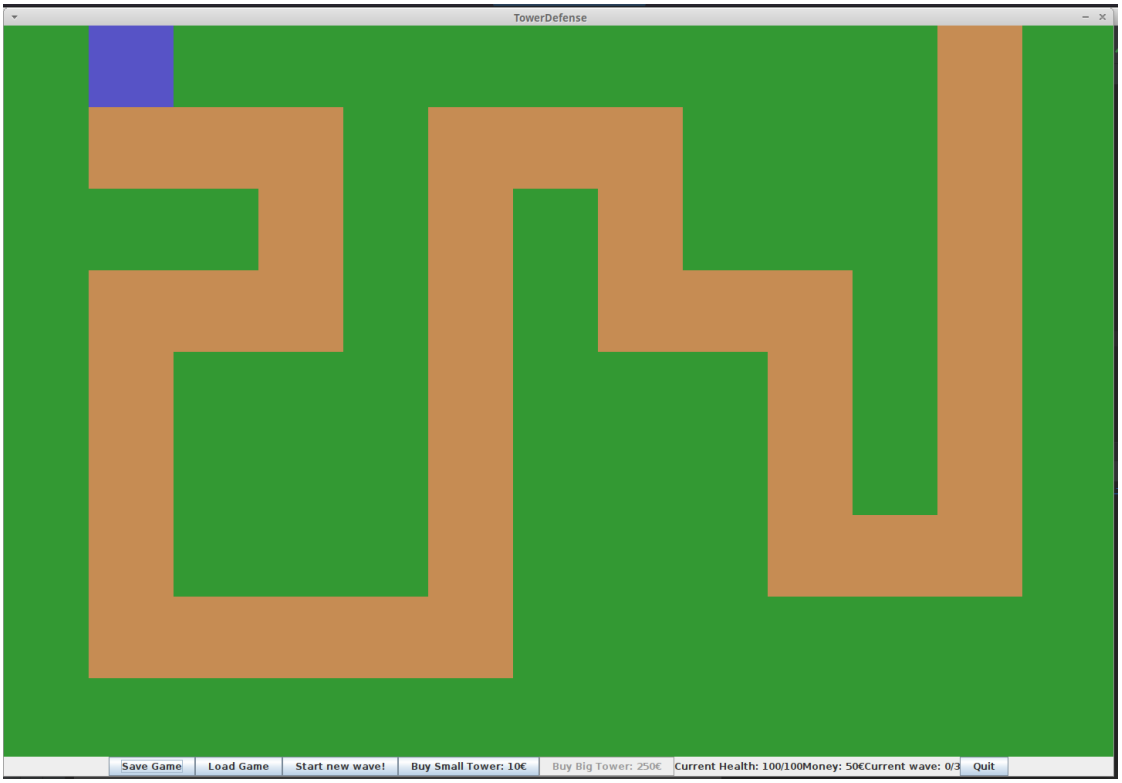
13. Viitteet

https://www.gamedeveloper.com/design/tower-defense-game-rules-part-1-
https://www.raywenderlich.com/2709-how-to-make-a-tower-defense-game-tutorial

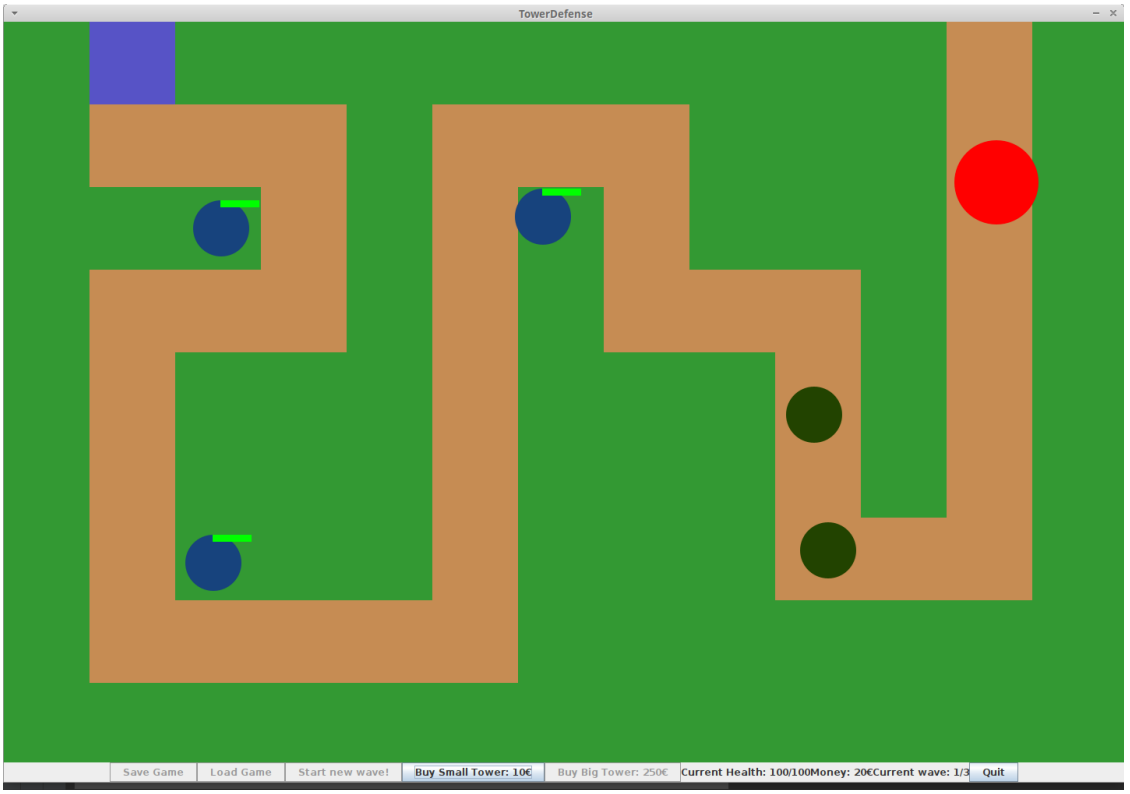
14. Liitteet

Projektin lähdekoodi.

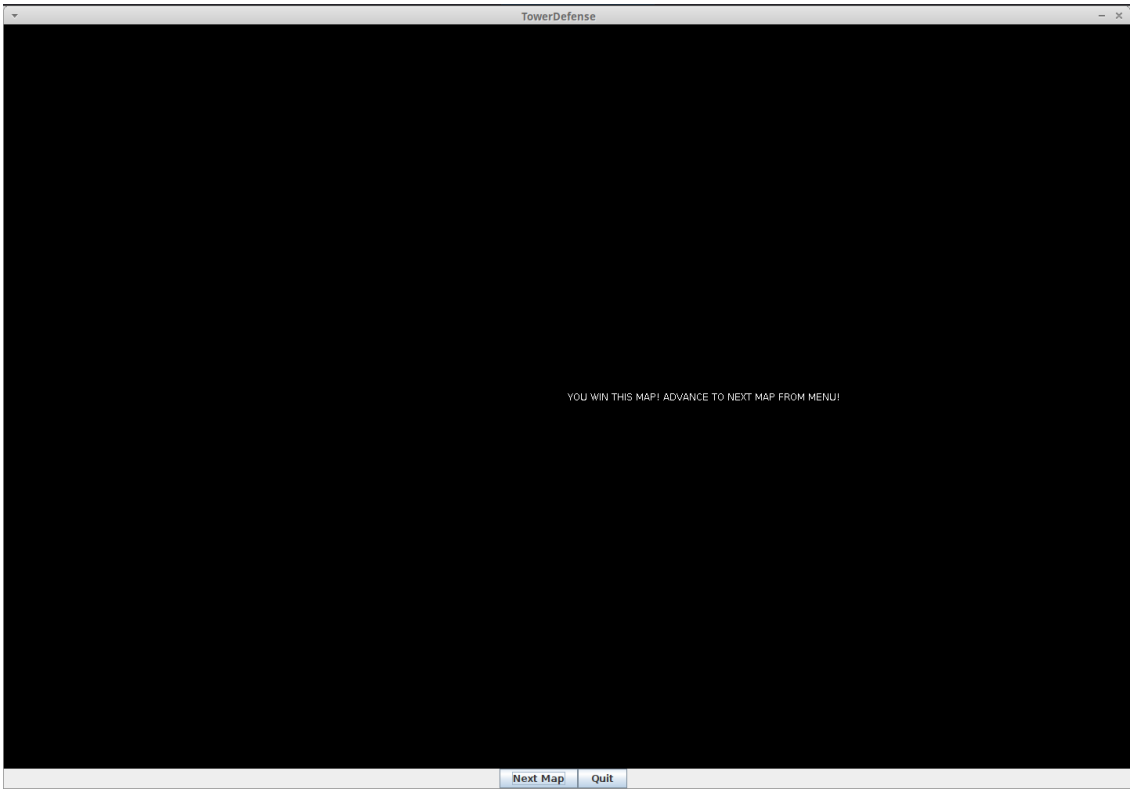
Kuvankaappaukset pelistä:



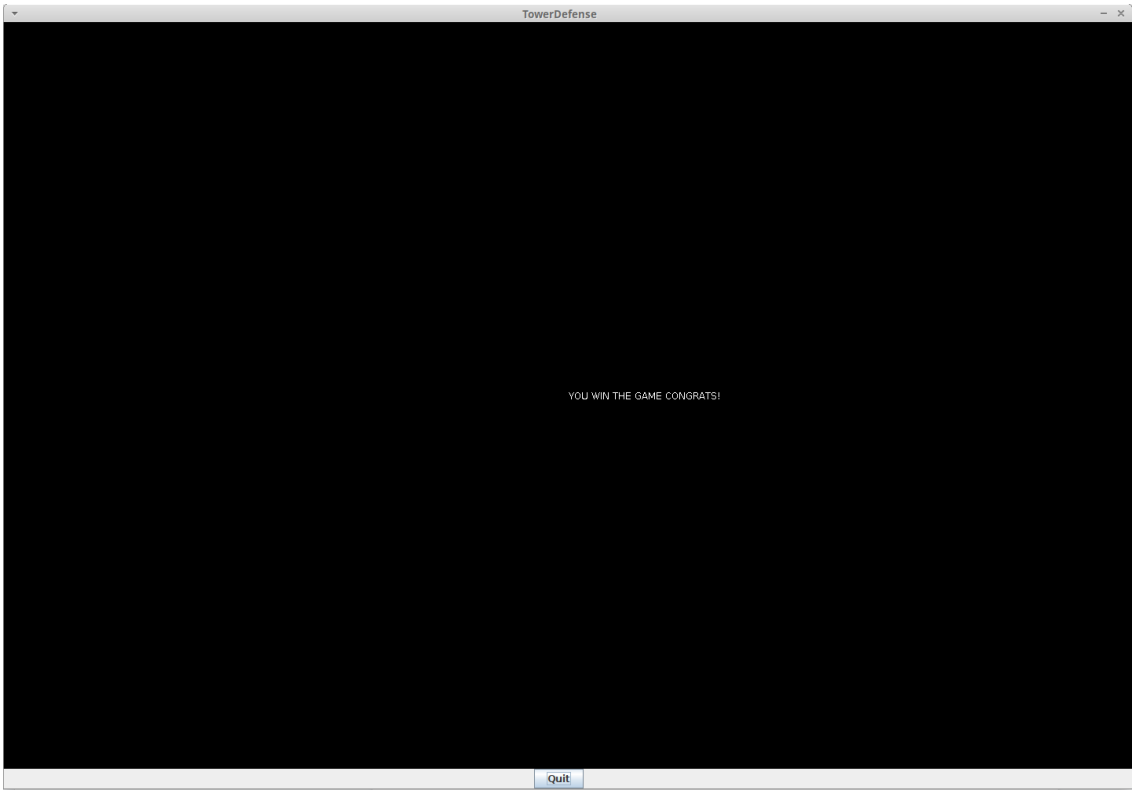
Aloitusnäkymä kun peli käynnistetään. Torneja voi ostaa alavalikosta. Myös pelin voi ladata ja tallentaa sekä pelistä poistua alavalikon kautta.



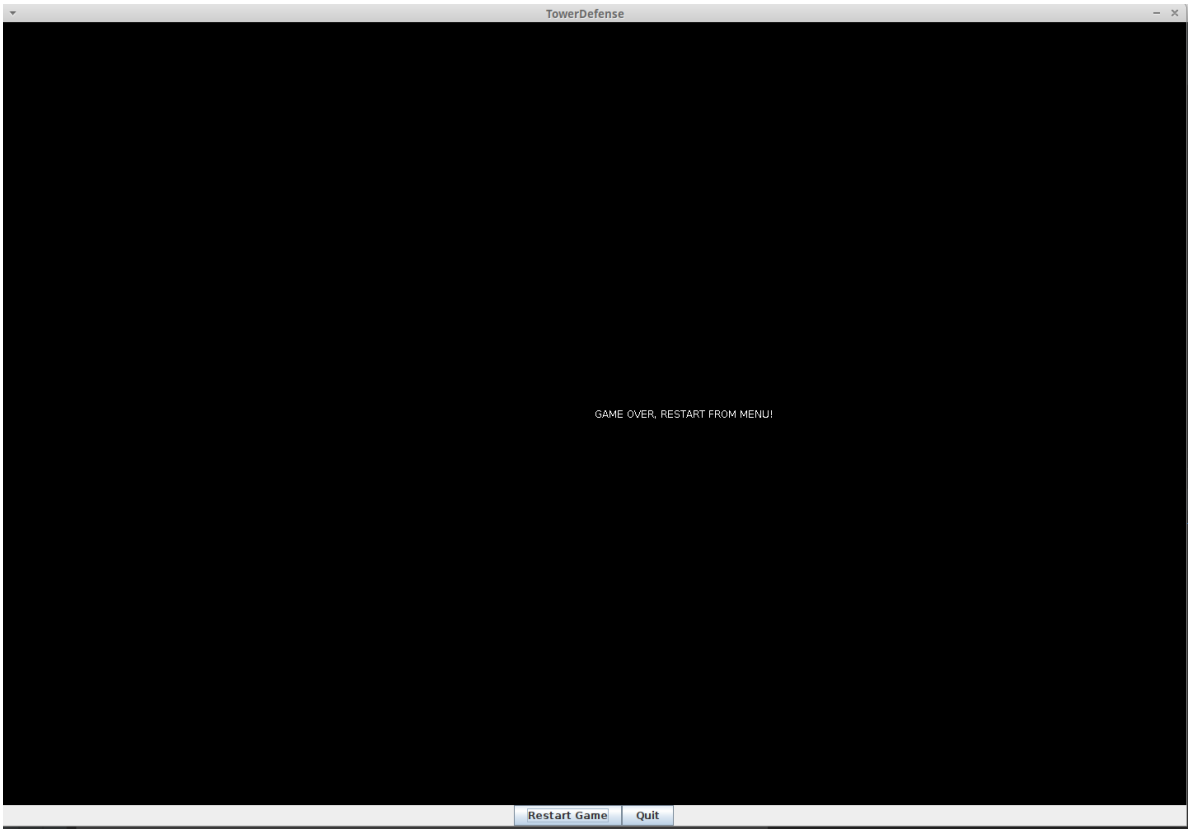
3 pientä tornia (siniset pallot) asetettuna pelikentälle ja kolme vihollista polulla (2 vihreää ja 1 punainen pallo).



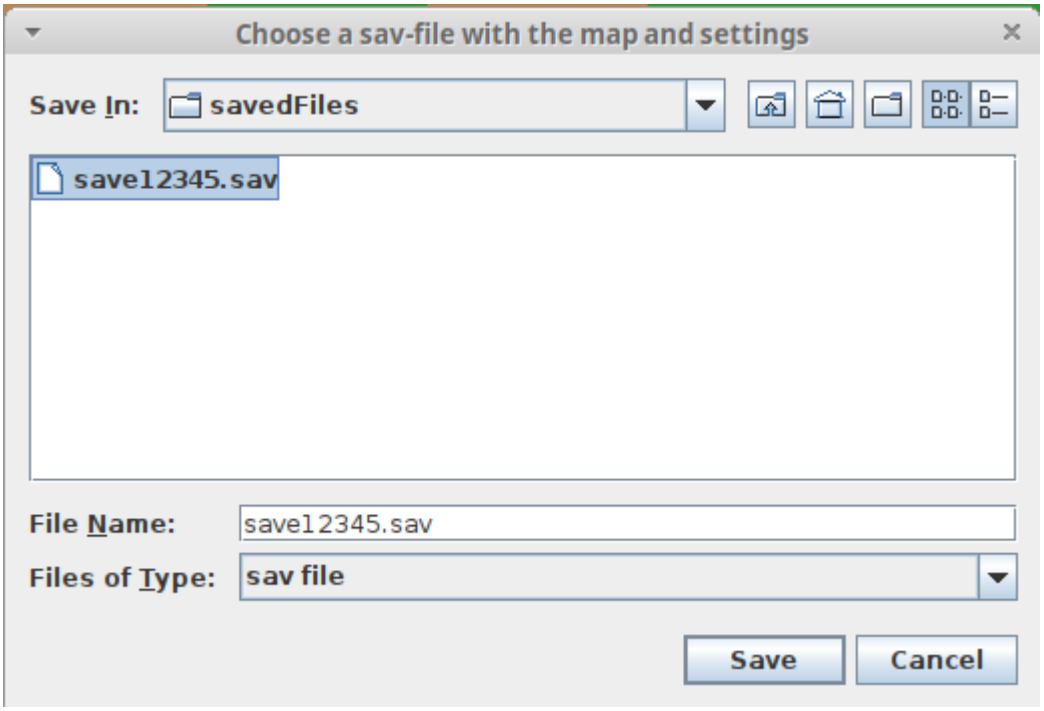
Kun pelaaja voittaa kartan seuraava ruutu tulee esille, “Next map” napilla jatketaan seuraavalle kartalle.



Kun pelaaja voittaa kaikki kartat tulee peli voitettu ilmoitus. Tällöin pelistä voi vain poistua.



Jos pelaajan health pointsit loppuvat peli loppuu.



Pelin voi ladata tiedostosta ja sen voi tallentaa tiedostoon alavalikosta. Valitse oikea tiedosto tai tallennettaessa anna tiedosto nimi.