OS2 Projektidokumentaatio: Tower defense game

Akseli Aho

608884

Tietotekniikka / 1. vuosi

08.05.2019

Yleiskuvaus

Projektini aihe on torninpuolustuspeli. Pelin ideana on puolustaa ennalta määrättyä polkua pitkin kulkevia vihollisia pääsemästä kartan poikki asettamalla erilaisia torneja matkan varrelle. Pelaajalla on tietty määrä elämiä ja rahaa, jonka puitteissa hänen pitää toimia ja onnistua tehtävässään. Pelin voittaa, kun on mennyt 10 kierrosta, mutta peliä voi jatkaa ikuisuuksiin niin kauan kuin elämiä riittää. Jossain vaiheessa kuitenkin tasot ovat niin haastavia, että pelin jatkaminen ikuisuuksiin on käytännössä mahdotonta. Torninpuolustus projekti oli mahdollista suorittaa joko keskivaikeana tai vaikeana. Mielestäni lopullinen toteutukseni oli keskivaikea. Projektiaihe oli sinänsä mielenkiintoinen, koska torninpuolustuspeli on aihe, joka on lähes loputtomiin kehitettävissä. Esimerkiksi mietin usein, kuinka monta erilaista tykkiä, vihollista ja ammustyyppiä pitäisi luoda.

Käyttöohje

Ohjelman käyttö on melko suoraviivaista. Ohjelma käynnistetään IDE:stä (Eclipse), jonka jälkeen pelaaja voi asettaa torneja kartalle. Tornin asettaakseen pelaaja klikkaa haluamaansa tornia sovelluksen oikeasta reunasta, jonka jälkeen hän painaa pelikentällä haluamaansa kohtaa. Torni asettuu tämän jälkeen paikalleen ja ampuu automaattisesti vihollisia näitä havaittuaan. Pelin edetessä kannattaa torneja asettaa lisää kentällä, koska aaltojen vihollismäärä kasvaa pelin edetessä. Pelissä on painike ”Start Round”, jolla saa aloitettua uuden kierroksen vanhan päätyttyä. Pelin voi sulkea sulkemalla ikkunan, jossa peli on käynnissä.

Ohjelman rakenne

Ohjelman rakenteessa on kolme pääosa-aluetta:

1. data
2. gui
3. helpers

Pakettiin data sisältyy kaikki pelin logiikkaan liittyvä koodi. Data paketin yhteen vetävä luokka on Game, josta luodut oliot kuvaavat kukin yhtä pelisessiota. Pelissä täytyy tietenkin olla pelaaja, joten tätä varten meillä on Player luokka. Player luokka vastaa pelaajalle kuuluvista tehtävistä. Pelaaja voi asettaa tykkejä, pelaajalla on elämät ja pelaaja on tietty määrä rahaa. Tykit, joita pelaaja voi asettaa ovat kukin oma luokkansa, jotka periytyvät Traitista Tower. Trait on hyvä ratkaisu mielestäni tässä tilanteessa, koska kun pelaajalla on useita eri tykkejä voi hän silti suorittaa niille tietyt metodit niiden luokista riippumatta, jos kyseiset metodit ovat määritelty yliluokkaan Tower. Tykit eivät itsessään tee mitään, joten täytyy olla myös luokka Projectile. Tämä luokka kuvaa ammusta jonka tykki ampuu. Yksittäisen ammuksen tehtävä on tietää mitä kohti se lentää, onko se osunut kohteeseen ja sen on myös voitava vahingoittaa kohdettaan siihen osuessa. Kohteet ovat kuvattu luokalla Enemy. Kaikilla vihollisilla on samat perusominaisuudet ja metodit. Viholliset osaavat itse määritellä polkunsa minkä tahansa kartan läpi. Tämä oli itselleni tärkeä tavoite saada toteutettua jatkokehitystä varten. Vaikka pelissäni ei ole vielä level editoria, on sellainen helppo kehittää Playerin setTile metodin ja vihollisten automaattisen reitinluonnin takia. Vihollisilla on myös enemytype parametri niitä luotaessa. Nämä enemytypet ovat objekteja, jotka periytyvät piirreluokasta EnemyType. Jokaisella näistä objekteista on joukko erilaisia juuri tälle kyseiselle vihollistyypille ominaisia ominaisuuksia. Viholliset eivät suinkaan liiku kartalla yksin vaan kierrokset koostuvat aalloista eli niin sanotuista Wave:sta. Jokainen aalto koostuu tietystä määrästä vihollisia ja Wave luokassa metodit päivittävät myös siihen kuuluvien vihollisten tilaa. Koska tasoja on monia, niin on myös aaltoja. Luokka WaveManager hallitsee kyseisen Game objektin koko pelin kaikkia aaltoja. WaveManager tietää millon pitää luoda uusi aalto ja milloin vanha aalto on päättynyt. Pelin kenttä koostu erilaisista Tile:stä, jotka ovat nimensä mukaisesti laattoja pelikentällä. Jokaisella tilellä on oma tyyppi esim. Grass tai Dirt ja näitä tyyppejä kuvaa piirreluokka TileType. Piirreluokan ansiosta erilaisia laattatyyppejä on jatkossa helppo lisätä peliin. Yksittäisistä tile:stä ei ole

vielä mitään iloa, joten tilejen kokonaisuutta kuvaa luokka TileGrid. TileGrid sisältää omassa konfiguraatiossani 20 x 15 tileä. Yksinkertaisuudessaan tilegrid on siis iso array joka sisältää pienempiä arrayta, jotka edelleen sisältävät tilejä.

Kaikilla ylläolevista luokista on metodit update ja draw. Nämä metodit ovat pelin kannalta kaikista tärkeimpiä, koska ne päivittävät pelin tilaa ja laittavat asiat liikkeelle pelissä ajan kulkiessa. Pelin saatetaan käyttäjän näytölle näkyviin ScalaFX:llä tehdyn gui:n avulla. Luokassa Drawing on pelin grafiikan luomiseen liittyvä koodi. Drawing luokassa voi muokata pelikarttaa yksinkertaisesti muuttamalla kartalta muistuttavan 2 ulotteisen array:n polkua joka koostuu numeroista yksi. Drawing luokassa luodaan myös Game olio, joka on lopulta itse peli, joka pyörii grafiikan taustalla. Drawing luokassa ei ole kuitenkaan kaikkia grafiikan piirtämiseen liittyviä metodeja. Suurin osa näistä metodeista on sijoitettu apuluokkaan Helpers. Tämä luokka toimii ikään kuin välikätenä pelin logiikan ja gui:n välillä. Loin luokan siksi, että halusin pitää itse Drawing luokan mahdollisimman yksinkertaisena ja varsinkin animation timerin koko olisi voinut kasvaa hyvinkin suureksi ilman Helpers apuluokkaa.

Algoritmit

Ohjelman tärkeimmät algoritmit ovat:

1. Vihollisten automaattinen reitinvalinta
2. Ammusten kohdentaminen viholliseen
3. Kartan luominen arraysta

Vihollisten automaattinen reitinvalinta nousi esille kun mietin miten saan viholliset liikkumaan tiettyä polkua pitkin. Helpoin ratkaisu olisi ollut merkitä em. polun kulmat, joissa vihollisen tulee kääntyä ja asettaa ohjelmakoodiin käännökset juuri näihin kohtiin. Ongelma tässä toteutuksessa on se, että jos ohjelmaa halutaan laajentaa ja toteuttaa level editori, joka muuttaa karttaa niin reitti pitäisi aina manuaalisesti luoda uudestaan. Automaattisessa reitinvalinnassa, vihollinen skannaa polkuaan niin kauan eteenpäin kunnes vastaan tulee jokin muu kuin halutun tyyppinen laatta. Tämän jälkeen se tarkastaa ympäriltään kaikki suunnat ja etsii suotuisan laatan. Tämän jälkeen vihollinen taas skannaa eteenpäin jne.

Vihollinen siis tietää reittinsä heti sen synnyttyä kartalle, koska skannaaminen tapahtuu vihollista luodessa.

Ammusten kohdentamiseen viholliseen liittyy kaksi eri luokkaa. Player luokassa on metodi, joka laskee mikä vihollinen on kaikista lähinnä tykkiä. Tämä laskutoimitus tapahtuu pythagoraan lauseella. Kun kohde on valittu niin Projectile luokan calculateDirection metodi ottaa hallinnan. Tämä metodi laskee x suuntaisen etäisyyden suhteessa y suuntaiseen etäisyyteen kohteesta ja muodostaa x ja y suuntasille nopeuksille kertoimet lasketun suhteen perusteella.

Kartan luominen arraysta oli ensimmäinen suurempi algoritmi, jonka toteutin projektissani. Alkuperäinen idea oli lukea tekstitiedostosta kartta, joka koostuisi nimenomaan ykkösistä ja nollista, mutta hylkäsin idean ajanpuutteen ja sen tuoman vähäisen lisäarvon takia. Ohjelma osaa siis lukea arraysta ykköset ja nollat ja muodostaa niistä Gridin. Gridi sen jälkeen luo Tile:stä koostuvan arrayn, joka on tämän jälkeen helppo piirtää näkyviin näytölle tile:jen draw metodilla.

6. Tietorakenteet

Ohjelmassa käytetään scalan valmiita kokoelmatyyppejä. Näistä tyypeistä Buffer ja Array ovat käytössä ohjelmassa. Arrayta käytetään silloin kun tiedetään, että pelin päivittäminen ei muuta tai sen ei tarvitse muuttaa kyseisen kokoelman rakennetta, mutta jos tiedetään, että kokoelma rakenne voi muuttua niin käytetään buferia. Esimerkiksi pelaajan asettamat tykit ovat laitettu bufferiin juuri sen takia, että ennalta ei voida sanoa kuinka monta tykkiä pelaaja tulee asettamaan session aikana.

7. Tiedostot ja verkossa oleva tieto

Ohjelmassa ei ole paljon itse ohjelmakoodin ulkopuolelta haettavaa tai saapuvaa tietoa. Projektikansiossa on kuitenkin res tiedostopolku, joka sisältää peliin tarvittavia

kuvatiedostoja. ScalaFX:n toimintaa varten on projektikansion Referenced Libraries osiossa ScalaFX:n kirjastotiedosto eli tiedosto, jonka avulla saa scalaFX:n ominaisuudet käyttöön.

8. Testaus

Torninpuolustuspeli perustuu täysin graafiseen käyttöliittymään ja tämän tiedon nojalla kaikki testaus tehtiin GUI:ta apuna käyttäen. Kun lisäsin uuden toiminnallisuuden peliin testasin heti gui:n avulla toimiiko kyseinen metodi. Esimerkkinä tästä voisi olla tornien asettamismetodi: metodin luotuani avasin käyttöliittymän ja kokeilin lisäytyykö torni pelikentälle sitä klikattuani. Ajoittain käytin myös konsolia testaukseen erilaisilla println komennoilla. Esim jos halusin varmuuden testauksen ajaksi siitä mitkä ovat milloinkin hiireni koordinaatit, saatoin tulostaa ne konsoliin aina pelin päivittyessä.

9. Ohjelman puutteet ja viat

Ohjelmassa on suorityskyvyllisiä ongelmia jossain määrin. Esim kun vihollisaallot alkavat kasvamaan isommiksi, niin gui alkaa hidastelemaan. En saanut selvitettyä mistä hidastuminen johtuu. Hidastumisen seurauksena huomasin myös, että viholliset saattavat poiketa em. reitiltään, koska peli ilmeisesti ei enään päivitä itseään niin nopealla tahdille.

10. 3 parasta ja 3 heikointa kohtaa

Kolme parasta kohtaa projektissani ovat mielestäni

1. Projektin luokkarakenne yleisesti
2. Automaattinen reitinvalinta
3. Graafisen käyttöliittymän toteutus

Sain mielestäni yleisesti hyvin organisoitua peliin liittyvät aihealueet luokkiin. Peliä on helppo laajentaa ja luokat eivät poikkea toistensa kanssa suuresti. Luokat kuvaavat mielestäni myös onnistuneesti vain sille tarkoitettua aihetta.

Automaattinen reitinvalinta oli myös mielestäni yksi projektini parhaimmista paloista. Syitä olen jo perustellut ylempänä, mutta nimenomaan pelin jatkokehiteltävyys on paljon korkeammalla tasolla tämän ominaisuuden takia.

Graafinen käyttöliittymä onnistui mielestäni myös hyvin. Sain tehtyä siitä selkeän ja toimivan.

Kolme heikointa kohtaa projektissani olivat mielestäni

1. Tykkien vähyys
2. Vihollisten vähyys
3. Suorituskyky

Tykkejä olisi voinut kehittää peliin enemmän ja se olisi nostattanut pelaamiskokemuksen uudelle tasolle. Tällä hetkellä tykkejä on vain kaksi erilaista ja peli voi olla ehkä hieman tylsä sen takia.

Sama ongelma toistui vihollisten kohdalla. Vihollistyyppejä on vain yksi. Näillä vihollisilla on aina samat ominaisuudet, eikä vihollisilla ole esimerkiksi mitään erikoistaitoja kuten vauhdin kiihtyminen tms.

Kolmas heikko kohta oli suorituskyky. Niin kuin aikaisemmin jo mainitsin, pelissä on suorituskyvyllisiä ongelmia vihollisten ja tykkien määrän kasvaessa. Ongelmaa en saanut korjattua, vaikka yritin optimoida metodit aina mahdollisimman kevyeksi tietokoneelle.

11. Poikkeamat suunnitelmasta, toteutunut työjärjestys ja aikataulu

Projekti toteutettiin jotakuinkin järjestyksessä:

1. Tile, tilegrid ja tiletype
2. Helpers
3. Drawing
4. Enemy, enemytype ja checkpoint
5. Wave ja WaveManager
6. Tower, CannonTower ja FlameTower
7. Player

8. Projectile
9. Game

Projektissa pysyttiin hyvin pitkälti suunnitelmassa. Muutamia apuluokkia ja piirreluokkia tuli suunniteltujen luokkien lisäksi. Ajankäytön kannalta suunnitelma meni mönkään. Kurssin alettua laitoin projektin heti aluilleen, mutta projektin teki, unohtui sen jälkeen miltei kuukaudeksi, joka harmittaa jälkeenpäin, koska kun lopuksi aloitin taas työstämään projektia säännöllisesti niin aloin nauttimaan sen tekemisestä. Ajankäytöllisesti projektin tekemiseen meni karkeasti arvioiden noin 100h.

12. Kokonaisarvio lopputuloksesta

Projektin tekeminen oli hyvin opettava kokemus. Vaikka opin paljon uusia asioita scalalla ja scalaFX:llä koodaamisesta, niin mielestäni tärkein oppitunti, jonka opin projektin tekemisen yhteydessä oli ajankäytön hallinta. Huomasin, että en osannut suunnitella ja arvioida tarpeeksi tarkasti omaa ajankäyttöäni ja projektin tekeminen jäi liian viimetinkaan. Projektini toimii, niin kuin perinteisen torninpuolustuspelin kuuluukin, mutta se on vähän tylsä. Tähän syynä on pelin vähäiset vaihtoehdot aseissa, kartoissa ja vihollistyypeissä. Ohjelmaa voisi parantaa optimoimalla sen suorituskyvyn korkeammalle tasolle sekä lisäämällä siihen uusia tykkejä sekä level editorin.

13. Viitteet

-SCALAFX API

-MARK LEWIS YOUTUBE

-INDIE PROGRAMMER YOUTUBE

-STACKOVERFLOW JAVAFX

-STACKOVERFLOW SCALAFX

-OS2 KURSSIMATERIAALI

-O1 KURSSIMATERIAALi