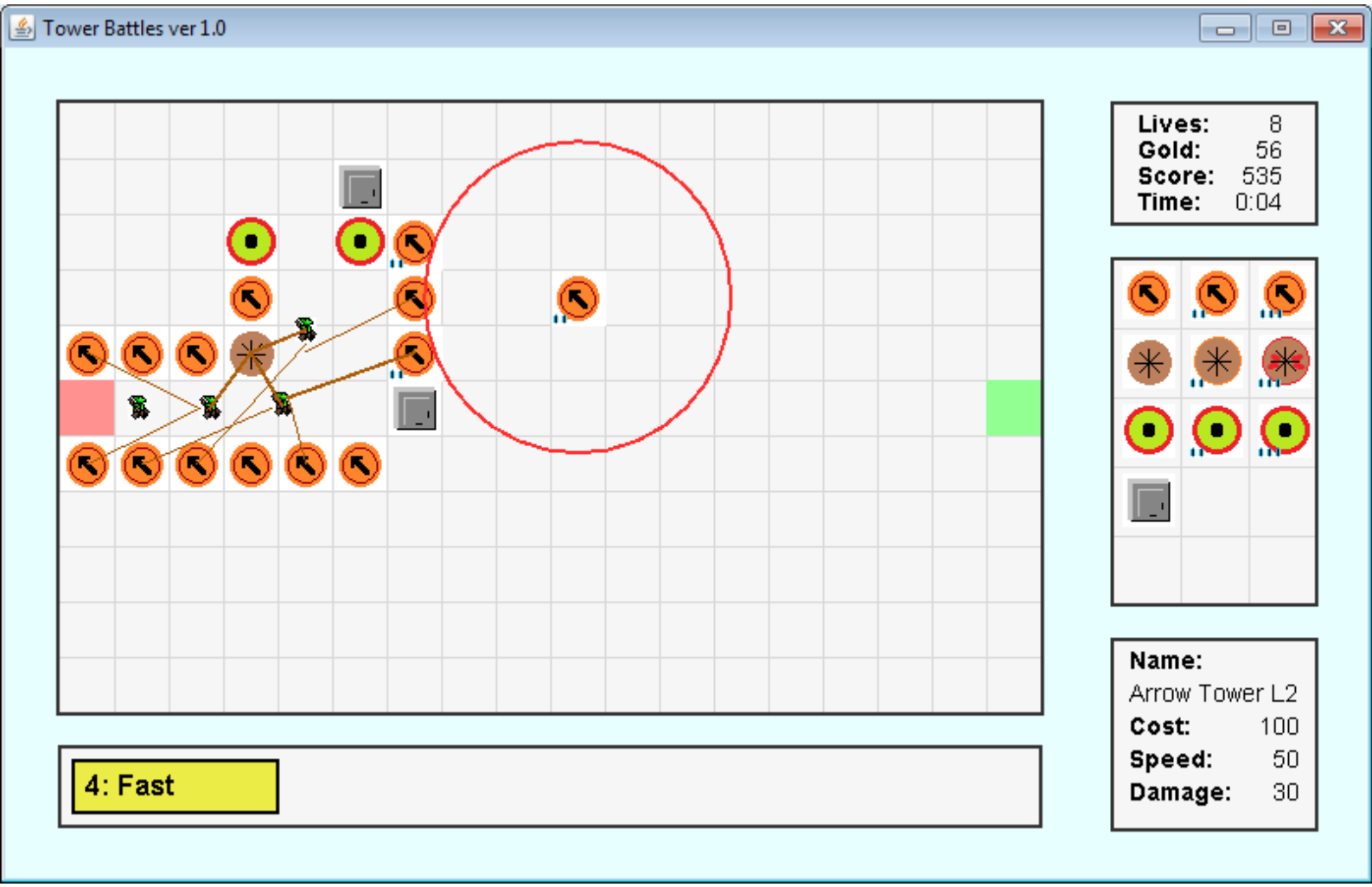
CSE-2120: Ohjelmointistudio 2 – projekti
Projektityön dokumentti
Aihe 370, Tower Defence
11.5.2015

Esa Koskinen 429788 esa.t.koskinen@aalto.fi
Tietotekniikan koulutusohjelma, 2. vuosikurssi



Tower Battles -pelin ensimmäinen julkaisuversio

Yleiskuvaus

Lähdin toteuttamaan perinteistä tornipuolustuspeliä, hieman Desktop Tower Defencen tyyliin. Ideana on kuitenkin, että mahdollisimman moni osa pelistä olisi kustomoitavissa, jonka johdosta niin uusien kenttien kuin vihollisten tai tornityyppienkin luonti on käyttäjälle varsin vaivatonta. Monessa kohtaa toteutusta on asetettu tehokkuus ja mielenkiinto pelaajan näkökulmasta toteutuksen helppouden ylitse, ja huomattavasti helpompia ratkaisuja tämän tyylisen pelin tekemiseen on lukuisia. Tämän vuoksi ei liene epäilystäkään, etteikö työ kattaisi kurssilla esitettyjen vaativan työn vaatimuksia.

Graafinen käyttöliittymä on varsin johdonmukainen ja toivon mukaan edes joskus aiemmin genren pelejä pelanneelle erittäin intuitiivinen. Alkuvalikko on toistaiseksi erittäin pelkistetty: pelaaja valitsee kentän ja peli alkaa. Peli koostuu kahdesta vuorottelevasta vaiheesta – jokaisessa vihollisaallossa pelaajalla on ennalta määrätty aikaraja, jonka aikana torneja voi asetaa kentälle. Tämän jälkeen punaisesta neliöstä ilmestyy vihollisia, jotka pyrkivät vihreää neliötä kohti ja vahingoittavat pelaajaa siihen osuessaan. Kaikki viholliset voitettuaan, alkaa seuraavan aallon valmisteluvaihe. Mikäli aikaa jää yli, voi pelaaja halutessaan painaa ”Skip” -nappia aloittaakseen vihollisaallon etuajassa; tämä myös antaa pelaajalle jonkin verran pisteitä.

Antamalla ohjelmalle käynnistyksen yhteydessä -debug -komennon, konsolille tulostuu pelin aikana sen toimintaan liittyviä viestejä.

Rakenne

Tower Battles on varsin perinteiseen tapaan jaettu kahteen eri osaan – käyttöliittymään ja pelilogiikkaan. Pelin ytimenä on Launcher-luokalla käynnistettävä Manager-olio, joka hoitaa pelin varsinaisen alustuksen, käynnistää käyttöliittymän ja määrittelee olennaisimmat vakiot. Tämän lisäksi lähtökohtana oli, että Manager hoitaisi Java-tyylisesti kaikkea kanssakäymistä käyttöliittymän sekä muiden ulkoisten luokkien kanssa erinäisten apumetodien välityksellä. Tästä huolimatta aikarajoitusten vuoksi osassa koodia on suoria viittauksia ”syvemmällä” sijaitseviin tietorakenteisiin ym.

Pelin logiikka mallintaa aika tarkasti sitä, miten yksi pelisessio on jaettu: Stage -luokka kuvaa yksittäistä pelikertaa (”kenttää”), joka sisältää tiedon saatavilla olevista torneista ja vihollisaalloista. Vihollisaallot (Wave) koostuvat vihollisista (Enemy), jotka puolestaan tallentavat tiedon sijainnistaan, elämäpisteistään, hakevat reitin maalia kohti, ym. Jokainen pelaajan asettama torni on Tower -luokan ilmentymä, joka sisältää tiedon edellisestä ampumishetkestään, sijainnista, ym.

Käyttöliittymä puolestaan koostuu kahdesta luokasta – MainFrame -luokan perivästä GUI-luokasta muodostetaan ikkuna ja toteutus vaadittaville graafisen käyttöliittymän metodeille. Lähdin toteuttamaan graafista käyttöliittymää ensin Scagella ja sitten Slickillä, mutta nämä epäonnistuivat; tarkempi selitys GUI-luokan kommenteissa. Päädyin sitten käyttämään AWT:n Graphics2D -luokkaa piirtämiseen: GameScreen (perii luokan scala.swing.Panel) hoitaa kaiken siltä osin hyödyntämällä Managerissa ilmoitettua pelin tilaa.

Olennaista pelin struktuurille on 60 kertaa sekunnissa suoritettava update() -metodi, joka on parallelisoitu säännöllisesti toistuva tapahtuma Managerissa. Managerin päivitys aktivoi niin käyttöliittymän kuin kentän päivitysmetodit, jotka puolestaan välittävät viestin eteenpäin, aina yksittäisten vihollisten ja tornien tasolle. Nämä hoitavat pelin kulkuun liittyvät asiat ja päivittävät omaa tilaansa sen avulla.

Algoritmit

Pelin kannalta keskeisimmät algoritmit lienevät reitinhaku ja tiedostonluku. Reitinhaku on toteutettu A* -algoritmilla sen varsin hyvän tehokkuuden vuoksi, käyttäen heuristiikkana absoluuttista etäisyyttä maalista. Kenties olennaisin ero standarditoteutuksiin on naapureiden hakemisessa: graafin (=pelialueen) reunoilla ei voi vain yksinkertaisesti hakea taulukosta edellistä tai seuraavaa ruutua mikäli se ylittää taulukon rajat. Tämän lisäksi viistoon kulkeminen on nopeampaa, mutta viistoon voi siirtyä jos ja vain jos jompikumpi sen ja lähtöpisteen vierekkäisistä ruuduista on vapaa.

Jotta peli näyttäisi graafisesti järkevältä tornien ollessa ruudukossa, viholliset siirtyvät aina yksittäisen ruudun puoliväliä kohti, eivätkä voi koskaan poistua ruudun kulmasta.

Tiedoston lukemisessa käytetään hyväksi rivin ensimmäistä merkkiä – luetaan rivejä kunnes vastaan tulee informaatio-lohkoa merkitsevä huutomerkki, jonka jälkeen luetaan rivejä seuraavaan huutomerkki-alkuiseen lohkoon (tai tiedoston loppumiseen) asti. Lohkon nimestä riippuen lohkon sisällä olevat tiedot käsitellään eri tavoin, kuitenkin usein jakamalla ne yhtäsuuruusmerkin kohdalta kahtia ja sijoittamalla se avain-arvo-parina tekstimuotossa Map-tyypin kokoelmaan. Lohkon päättyessä näistä tiedoista muodostetaan haluttu asia, tai todetaan ettei tietoja annettu tarpeeksi.

Eräs tästä johtuva haittapuoli on, ettei yhtäsuuruusmerkkejä voi käyttää asioiden nimissä (eikä peli osaa niihin kunnolla varautua), mutta toisaalta yhtäsuuruusmerkin käyttö nimessä on varsin harvinaista, ja niin tiedostojen kirjoittaminen kuin lukeminenkin on varsin vaivatonta. Koodi on kuitenkin kohtalaisen rumaa lukuisten merkkijono-operaatioiden vuoksi.

Tiedostoformaatti

Tower Battles käsittelee paljon asioita ulkoisten tiedostojen kautta; olennainen ja näille yhteinen piirre on lohkotteisuus. Mielestäni lohkotyyppinen formaatti on erittäin toimiva, sillä monia asioida käsitellään pelissä yksittäisinä olioina ja käyttäjälle uusien asioiden luonti onnistuu helposti olemassaolevia kopioimalla. Lohkot alkavat aina huutomerkillä (!), ja tiedostoja voi kommentoida ristikkomerkillä (#) alkavilla riveillä.

Kenttätiedostot haetaan stages-kansiosta sekä sen alikansioista (mutta ei alikansioita syvemmältä), kun taas mahdolliset kuvatiedostot sijoitetaan images-kansioon. ImageIO-kirjastolla BufferedImage -tyyppiseksi muunnettavissa olevat kuvaformaatit ovat tuettuja. Muut tiedostot (enemies.txt, macros.txt, profile.txt, readme.txt sekä towers.txt) ovat pelin ylimmässä hakemistossa. Suosittelen tutustumaan esimerkkitiedostoihin ennen spesifikaation lukemista.

Kenttätiedojen lohkoissa tulee olla vähintään versiomäärite, sallitut tornit sekä yksi tai useampi vihollisaalto. Kaikki versiossa 1.0 tuetut kenttätiedostojen lohkot:

!Tower Battles 1.0
Ilmaisee version; tarkka formaatti ei tiukasti spesifioitu, kunhan teksti ”1.0” esiintyy siinä.

!availabletowers
Sisältää yhdellä tai useammalla rivillä listan kentässä sallituista torneista. Tornien tulee olla määritelty towers.txt -tiedostossa.

!wave
Jokainen wave-lohko määrittää yhden vihollisaallon. Mahdollisia muuttujia ovat
goldbonus = xxx (kokonaisluku) – Antaa pelaajalle xxx määrän rahaa aallon alkaessa.
lifebonus = xxx (kokonaisluku) – Antaa pelaajalle xxx elämää aallon alkaessa.
buildphase = xxx (kokonaisluku) – Aallossa on xxx sekuntia aikaa rakentaa torneja ennen vihollisia.
type = xxx (teksti) – Tyyppi määrittelee aallon tyypin; vaikutus on puhtaasti visuaalinen, käyttöliittymän aalto-indikaattorin hyödynnettävissä. Mikäli arvo ei ole tunnettu, asetetaan arvoksi ”normal”. Mahdollisia arvoja ovat:
normal, slow, fast, mixed, flying, spawning, boss

Tämän jälkeen seuraa vapaavalintainen määrä vihollisia, muodossa
enemyid, amount, delay, initialdelay (String, Int, Int, Int; mitattuna 1/60 sekunnin hetkinä)
Suunnitelmasta poiketen ei aritmetiikan käyttö ole vielä mahdollista. Tekstimuotoisten laskujen prosessointi sijaitsee Manager.scala -tiedoston parseCalc-oliossa, mutta muistin sen olemassaolon vasta jälkikäteen enkä ehtinyt hyödyntää toiminnallisuutta. Nimeämiskonventiosta on poikettu funktionomaisen käytettävyyden saamiseksi.

!highscores
Ei vielä toteutettu; laskee ja tallettaa parhaat tulokset, jotka käyttöliittymä voi halutessaan esittää.

towers.txt määrittelee kenttien käyttöön erinäisiä torneja, aina yhden per lohko. Myös versiolohkon tulee olla olemassa.

!definetower
id = xxx (teksti) – Tornin tunnus.
name = xxx (teksti) – Tornin nimi.
image = xxx (teksti) – Tornin kuvatiedoston nimi, esimerkiksi mytower.png.
cost = xxx (kokonaisluku) – Tornin hinta.
speed = xxx (kokonaisluku) – Tornin latausaika; pienempi speed ampuu nopeammin. Ampumisen estämiseksi voi käyttää arvoa -1.
damage = xxx (kokonaisluku) – Tornin tekemä vahinko.
range = xxx (kokonaisluku) – Tornin hyökkäisetäisyyden säde pikseleinä; jotta torni ampuisi ruutunsa ulkopuolelle, tulee rangen olla vähintään 16.
upgrade = xxx (teksti) – Sen tornin tunnus, johon tämän tornin voi kehittää. Ei vielä toteutettu.
special = xxx (teksti) – Tornin erikoisuus; tällä hetkellä vasta ”wall”, ”aoe”, ”slowXXX” ovat toteutettuja – wall estää tornia ampumasta, aoe antaa tornin ampua kaikkiin hyökkäysalueensa vihollisiin, ja slowXXX (jossa XXX korvataan halutulla kokonaisluvulla, esim slow60) hidastaa vihollisia.

enemies.txt määrittelee viholliset kenttien käyttöön; jälleen, yhden per lohko. Versio tulee mainita.

!defineunit
id = xxx (teksti) – Vihollisen tunnus.
image = xxx (teksti) – Kuvatiedoston nimi.
hp = xxx (kokonaisluku) – Vihollisen elämäpisteet.
damage = xxx (kokonaisluku) – Vihollisen tekemä vahinko vihreään ruutuun päästessään.
speed = xxx (kokonaisluku) – Vihollisen etenemisnopeus, pikselien kymmenesosissa per frame.
goldgain = xxx (kokonaisluku) – Vihollisen tuhoamisesta saatava rahamäärä.
scoregain = xxx (kokonaisluku) – Vihollisen tuhoamisesta saatavat pisteet.

macros.txt ei ole toteutettu; sen tarkempi formaatti kuvattu projektin suunnitelmassa.

profile.txt on ennätystuloksia varten, mutta tällä hetkellä tarpeellinen vain pelin tiedostonhallinta-oikeuksien varmistamiseen.

Testaus

Ajan puutteesta johtuen ohjelman toteutus eteni pitkälti yksi placeholder kerrallaan; kirjoita koodia, testaa se, siirry seuraavaan funktioon. Yksikkötestejä en hyödyntänyt lainkaan, mutta väliaikaisia debug-viestejä oli erittäin runsaasti algoritmien toimintaa tarkistettaessa.

Puutteet

Projektissa on lukuisia asioita, jotka vielä haluaisin toteuttaa, tässä niistä listattuna olennaisimmat. Ensinnäkin moni toteutus ja ratkaisu jäi ajanpuutteen vuoksi hieman purkka-liima-korjatuksi, ja koodissa viittausperusteet ovat hieman vaihtelevat (koodin laatu ja spesifikaatiossa pysyminen on aivan selvästi kääntäen verrannollista jäljellä olevaan aikaan). Moni asia on kohtalaisen virhealtis. Käyttöliittymässä lukuisia pieniä puutteita, mutta erittäin keskeistä on dialogien piirtämättömyys – en vain keksi kivaa ratkaisua sille. Kenttävalikko on aivan selkeästi vaiheessa vaikka toimii hyvin, ja pelin päättyminen yhden kentän jälkeen ei ainakaan lisää pisteitä.

Makrot, parhaiden tuloksien käsittely, aritmetiikka tiedostonlukemisessa, ym ym jäivät (odotuksien mukaan) kokonaan tekemättä. Huomasin myös, että reitinhaun painotukset vinosti haetuille ruuduille eivät toimi tietyissä tilanteissa kunnolla, mutta reitti on aina lyhyempi kuin pelkällä vierekkäisten ruutujen hakemisella. Torneja tulisi voida asettaa myös vihollisten ollessa kentällä.

Parhaat palat

Ohjelma toimii, ja se toimii hyvin. Käyttöliittymä on erittäinkin sulava ja intuitiivinen, ja jotenkin onnistuin kasaamaan kaiken tarvittavan pelilogiikan puolelta alle kymmenessä päivässä. Olen yllättynyt, että ulkoisia tiedostoja onnistuneesti hyödyntävä tornipuolustuspeli oli mahdollista kasata Scalalla näin lyhyessä ajassa tälläistä tyyliä ja ohjelmarakennetta käyttäen, mutta lopputulos on itselleni enemmän kuin tyydyttävä.

Aikataulu

Olen jo muutamaan otteeseen maininnut ajan puutteen; totta tosiaan, aloittaessani kuvittelin (syystä tai toisesta) deadlinen olevan vasta 17. päivä, ja olin suunnitellut asiat sen mukaan. Goblinista selvisikin, ettei näin ollut, ja olen nyt reilun viikon aikana työstänyt projektia ehkä noin 65-85 tuntia. Ensimmäiset ~15 tuntia menivät Scagen ja Slickin toimintaa ihmetellessä, ja heitettyäni ne roskiin alkoi olla jo valtava kiire; pyysin siis lisäaikaa. Loppujen lopuksi myöhästyn siitäkin deadlinestä, mutta uskallan vielä tässä vaiheessa väittää tuloksen olevan varsin kohtuullinen.

Toisen päivän illalla jatkoin siis Graphics2Dllä, ja tein alustavan luokkaraketeen kasaan. Kirjoittelin käyttöliittymää, ja lopputulos piirsi kaikkien UI-elementtien taustat sekä reunat.

Seuraavat kolme päivää menivät vihollisten toimintaa sekä tiedostojen lukemista ihmetellessä, ja kuudentena päivänä (6.5.) sain vihdoin jotain liikkumaan ruudulla. Seuraavat pari päivää menivät tornien toteuttamisen parissa, sitten pari päivää ohjelmointi 2- ja tietokannat-kurssien parissa, ja lopulta kirjoittelin käyttöliittymän koodia ja korjasin loppuja virheitä nukkumatta lainkaan 11.5.-13.5. Varsin tiukka tahti, siis! Kuitenkin käytetty aika on erittäin lähellä suunnitelman arviota, ja oli mielenkiintoista nähdä ohjelman kasvavan huikeasti päivä päivältä kohti toimivaa lopputulosta.

Laajennettavuus

Ohjelman oli tarkoitus olla erittäin laajennettava koodipuolella, mutta loppujen lopuksi tämä ei toteutunut. Huolellisemmalla suunnitelmalla, pidemmällä aikahaarukalla ja spesifikaatiossa pysymisellä näin kenties olisi voinut olla, mutta kokonaisuuden osaset vaikuttavat olevan erittäin tiukasti kiinni toisissaan. Luokkien sisäiset arvot ja metodit sekä niiden välinen kanssakäyminen muodostavat erittäin monimutkaisen verkoston, jonka massiivinen laajentaminen ei liene helppoa ilman kattavaa refaktorointia. Kuitenkin perusidea on varsin hyvä – yksittäisillää, piilotetuilla toiminnallisuuksilla voi saada paljonkin aikaan vaikuttamatta muiden luokkien toimintaan. Torneille ja vihollisille onkin erittäin helppo lisätä uusia erikoiskykyjä.

Näiden lisäksi laajennettavuuden pääpaino – eli uusien geneeristen vihollisten, kenttien ja tornien luonti – on erittäin kattavasti tuettu ja kohtuullisen toimivaa. Varmastikin on mahdollista hyödyntää *Tower Battlesia* monenlaisten pelikokemuksien tarjoamiseen, ja siinä suhteessa sanoisin projektin onnistuneen erinomaisesti.

Viitteet

Ticker-olion toimintaperiaatteen suunnitellut Otfried Cheong, kuvattu sivulla http://otfried.org/scala/timers.html ja haettu 5.5.2015.

greengoblin.png ja skeleton.png ovat osa Steve Riddettin ”MR Fangame Kit” -projektia, joka on Creative Commons 2.0 -lisenssin alainen ja vapaasti saatavilla sivustolta http://themetalbox.com.

A* -toteutuksen suunnittelun yhteydessä käytetty sivuja
http://wiki.gamegardens.com/Path_Finding_Tutorial ja
http://en.wikipedia.org/wiki/A*_search_algorithm.

Tietorakenteisiin ja käyttöliittymän toteutuksiin liittyvien metodien kuvaukset haettu Java 7 APIsta sekä Scala 2.11.4 -dokumentaatiosta:
http://docs.oracle.com/javase/7/docs/api/
http://scala-lang.org/api/2.11.4/