Juho Sillanpää
589903
Bioinformaatiotekniikka
11.5.17

Tornipuollustus-projekti

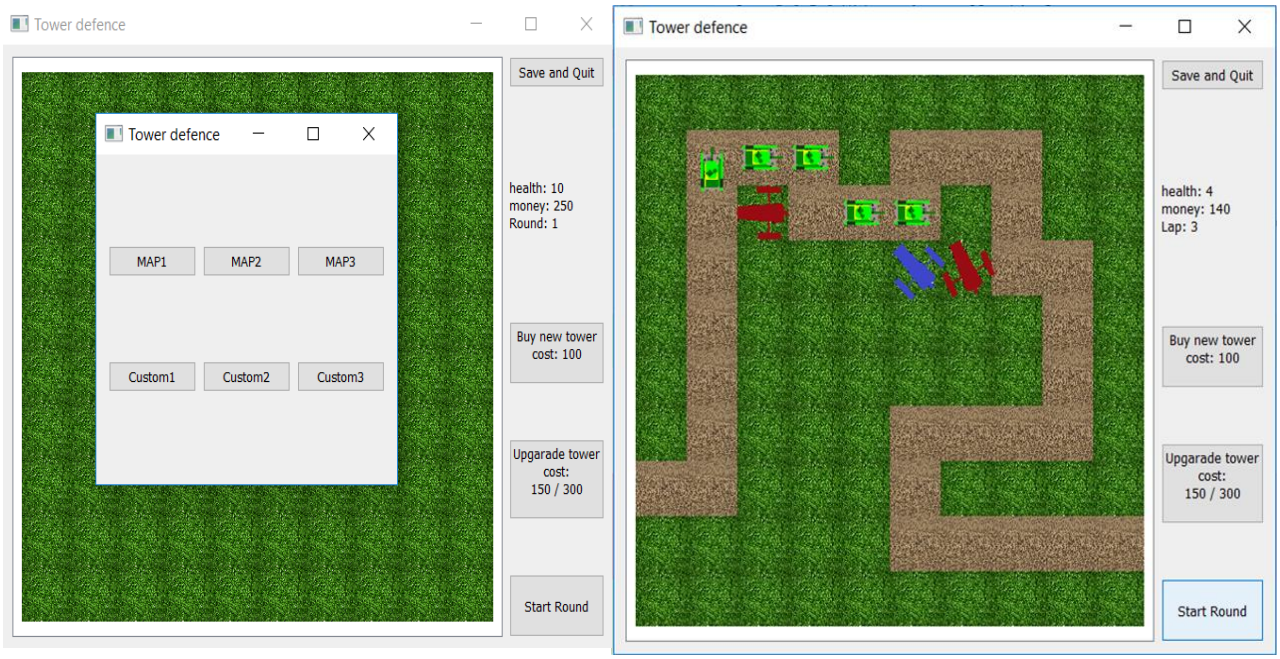
Yleiskuvaus

Tehtävänä on tehty graafinen tornipuollustus peli. Pelissä on kartalle piirretty reitti, jota pitkin erilaiset viholliset pyrkivät pääsemään kartan toiseen päähän tukikohtaan. Pelaajan on tarkoitus asettaa reitin varrelle torneja, jotka ampuvat vihollisia ja pyrkivät estämään niiden pääsyn tukikohtaan. Pelaajalla on valittavana erilaisia karttoja, joita hän voi myös itse luoda. Viholliset tulevst reitille kierroksittain ja kierrosten vihollistyypit ja määrät vaihtelevat. Perusideana on, että niitä tulee koko ajan yhä enemmän tai ne ovat yhä kestävämpiä. Luodessaan uuden kartan pelaajan täytyy myös luoda eri kierrosten vihollismäärät sekä tyypit. Peli päättyy mikäli tietty määrä vihollisia pääsee tukikohtaan asti.

Käyttöohje

Ohjelma käynnistetään ajamalla main.py tiedosto terminaalissa tai esimerkiksi Eclipsessä. Pelin muiden tiedostojen tulee olla samassa hakemistossa.
Pelaaja saa aluksi valita aloittaako uuden pelin vai jatkaako tallennettua peliä. Tämän jälkeen valitaan pelattava kartta. Tässä on hyvä huomioida, että “Custom”-kartat ovat alustamattomia tyhjiä karttoja, jotka pelaajan täytyy ensin itse luoda. Tälläisen kartan valitseminen ennen sen luomista ja tallentamista, johtaa todennäköisesti ohjelman kaatumiseen.
Kartan luominen tapahtuu kirjoittamalla gamefile.txt tiedostoon “#Custom1” (tai 2/3) rivin perään 10 riviä joissa on jokaisessa tasan kymmenen merkkiä. Merkki “x” tarkoittaa että siihen tulee maata, ja merkki “-” tarkoittaa tietä. Virheellinen määrä rivejä tai merkkejä riveillä voi johtaa ohjelman odottamattomaan toimintaan. Reitti tulee myös rakentaa vasemmasta reunasta oikeaan reunaan. Muussa tapauksessa ohjelma ei osaa sijoittaa vihollisia kartalle oikein, kääntää niitä oikein eikä myöskään laskea läpi päässeitä vihollisia.
Tämän jälkeen “Rounds for Custom1/2/3” rivin ja “&” rivien väliin voi kirjoittaa tiedot niin monesta kierroksesta kuin haluaa. Jokaisen rivin tulee sisältää tasan kolme kaksoispisteellä erotettua elementtiä. Ensimmäinen on vihollisten lukumäärä kierroksella, toinen on niiden tyyppi ja kolmas on niiden elinvoima. Sisäänrakennettuja tyyppejä ovat “car” ja “tank”, mutta halutessaan näitäkin voi luoda lisää. Tallentamalla samaan hakemistoon muiden tiedostojen kanssa haluamansa png-kuva, ja kirjoittamalla tyyppi, kohtaan kuvan nimen (ilman päätettä .png) pelaaja saa käyttöönsä halamansa näköisen vihollisen. Tässä kannattaa huomioida, että kuva kannattaa kääntää niin, että sen rintamasuunta on oikealle päin. Näin kuva kääntyy käännöksissä oikeaan suuntaan.
Kartan valinnan jälkeen avautuu itse pelinäkymä. Uusi torni tapahtuu klikkaamalla “Buy new tower”- nappulaa, jonka jälkeen torni asetetaan oikeaan paikkaan näppäimien “asdw” avulla. Näiden toiminta on yleisen logiikan mukainen. Torni asetetaan tiettyyn kohtaan painamalla “e”-näppäintä ja oston voi perua painamalla “q”-näppäintä. Tornin kehittäminen tapahtuu painamalla “Upgarade Tower”-nappia ja sen jälkeen tornia,jota haluaa kehittää. Mitä kehitetympi torni on, sitä suurempi iskuvoima sillä on vihollisiin.
Tornin ostaminen maksaa 100 rahaa, ja sen koroittaminen maksaa ensimmäisellä kerralla 150, seuraavalla 300 jne. Kun tornia kehittää, niin se muuttaa väriään. Periaatteessa tornia voi kehittää loputtomasti, mutta väri ei vaihdu enää toisen kehittämiskerran jälkeen. Rahaa pelaaja saa jokaisesta tuhomastaan vihollisesta 10 ja läpi päästystä kierroksesta 100.
Pelin missä vain vaiheessa pelaaja voi tallentaa pelin painamalla oikean yläkulman nappia “Save and quit”. Mikäli liian moni vihollinen (10kpl) pääsee reitin läpi, niin kartta sulkeutuu ja näytööle

ilmestyy “Game over”-teksti.



Ohjelman rakenne

Main-tiedostossa ohjelma käytännössä luo gui-tiedoston GUI-luokan olion. Gui vastaa pelilaudan ikkunoista, grafiikoiden piirtämisestä ja erilaisten komentojen käsittelystä ja pelin pyörittämisestä. GUI-luokka sisältää myös erilaiset ajastimet jotka huolehtivat mm. Vihollisten siirtämisestä, tornien ampumisesta, sekä tornien ammuksien piirtämisestä ja poistamisesta. GUI-luokka avaa myös start_window-tiedoston Start_window luokan toiminnan aktivoinnista. Start Window huolehtii pelin alustukseen käytettävien grafiikoiden esittämisestä ja kontrolloi filereader-tiedoston metodeja. Erilaisten ikkunoiden yhteydessä hyödynnetään paljon muun muassa QpushButton:ia sekä QgridLayout:ia. Ohjelman karttojen ja kierrosten tiedot sekä tallennetut pelit ovat tallennettuina tiedostoihin, joita filereader-tiedoston metodit lukevat.
Käynnissä olevan pelin tiedot ovat tallessa Game-luokassa. Luokassa luodaan pelilaudan matemaattinen pohja luomalla 10x10 kokoinen lista, jonka jokainen jäsen on Square-luokan ilmentymä. Ne tietävät oman sijaintinsa ja sen onko niissä torni sekä kuuluvatko tiehen. Kun filereader-metodien avulla alustetaan tietty pelikartta, niin se muuttaa tarvittavat Game-olion kartan osat tieksi.
Lukiessaan tietyn kartan kierrosten tietoja filereader:in metodi luo jokaista kierrosta kuvaamaan Round-olion, joka tietää kierroksensa vihollisten tyypin, niidet elinvoiman, ja lukumäärän. Kun Round-olio on luotu niin se lisätään Game-olion rounds listaan, jolloin niihin päästään käsiksi Game-luokan kautta.
Uusi kierros käynnistetään GUI-luokan kautta, ja se hakee Game-oliosta seuraavan kierroksen ja lähettää kyseiseen Round-olion koodatut vihollisyksiköt eteenpäin tähän erikoistuneen ajastimen avulla. Viholliset ovat Enemy-luokan ilmentymiä, ja tietävät oman sijaintinsa, oman elinvoimansa ja tyyppinsä. Graafisesti jokaista vihollista kuvataan EnemyGraphicsItem-luokan oliolla. Luokka on QgraphicsPixmapItem-tyyppinen luokka ja se piirtää kartalle vihollisen tyyppiä vastaavan kuvan, joka on tallennettuna .png-muodossa pelin kanssa samaan hakemistoon. Kuva skaalataan ja sen paikka haetaan sitä vastaavasta Enemy-oliosta ja asetetaan oikealle paikalleen. Kun vihollista siirretään GUI-olion toimesta niin olion uusi paikka ja suunta muutetaan ensin Enemy-luokan sisällä ja sen jälkeen sen sen graafinen olio päivittää sijaintinsa ja suuntansa.
Tornien suhteen pätee melko samanlainen luokka-malli. Tower-luokka lähinnä kuvaa tornia pelin tiedoissa ja TowerGraphicsItem, joka perii myös QGraphicsPixmapItemin, kuvaa tornia graafisesti, ja muun muassa kääntää sen osoittamaan seuraavaksi ammuttavaa vihollista kohti. Ammuttavan

vihollisen valinta taphtuu GUI-luokan sisällä, ja se laskee ensin vihollisten etäisyydet torneihin, ja mikäli etäisyys on tarpeeksi pieni, niin torni ampuu sitä ja luokka luo viivan kuvastamaan ammusta. Tornit ampuvat samaa vihollista kunnes se on kantaman ulkopuolella tai kuollut. Kuolleet viholliset saavat tiedon kuolemastaan, jolloin niitä ei voi ampua enään ja ne poistetaan näytöltä ja scenestä. Pelilaudan ruutuja kuvaavat grafiikat luodaan BackgroundGraphicsItem-luokassa joka myös perii QgraphicsPixmapItemin. Ruutujen, tornien sekä vihollisten kuvaamiseen on valittu kaikkiin Qpixmap-tyyppinen kuva. Valinta helpottaa uusien vihollistyyppien luomista, joka on nyt käyttäjälle hyvin helppoa ja selkeyden vuoksi kaikki muutkin kuvat on luotu samalla menetelmällä. Valinta mahdollistaa myös helpohkosti graafisesti uudenlaisten karttojen tekemisen, muokkaamalla BackgroundGraphicsItem:stä sellaisen, että se saa myös tiedon millainen kuva kuvaa tietä ja millainen maata, samaan tapaan kuin iten vihollisten graafinen kuva luodaan. .

Algoritmit

Tornin ja vihollisen väinen etäisyys lasketaan yksnkertaisella pythagoraan lauseella:
r = sqrt((x1-x2)*2 + (y1-y2)*2)
Tornin kääntöminen tapahtuu math.atan2-metodin avulla, joka laskee kulman x-akselin ja pisteiden läpi piirretyn suoran väliltä.. Metodi toimii tarkoitukseen loistavasti, sillä se tarkistaa itse onko piste 2 pisteen 1 alapuolella ja jos on niin palauttaa suorien välisen tylpän kulman.

Vihollisoliot löytävät reitin tukikohtaan pyrkimällä kulkemaan kokoajan oikealle päin.

Tietorakenteet

Ohjelman tallennetut tietorakenteet olivat kirjoitettuna tiedostoihin, mikä oli mielestäni simppeli ja toimiva ratkaisu. Käynnissä olevan ohjelman tiedot olivat tallennettuna suurimmaksi osaksi listoihin. Listat valittiin sillä ne olivat simppeli ja toimiva tapa kuvata tarvittavia tietorakenteita.

Tiedostot

Ohjelma käsittelee “gamefile.txt” ja “saved_games.txt” tekstitiedostoja sekä erilaisia .png-muodossa olevia kuvatiedostoja. Tekstitiedostoissa tiedot ovat esitettyina pääosin luettavissa olevassa muodossa, erityisesti “gamefile.txt”-tiedostossa, sillä pelaaja voi halutessaan muokata sitä. Ohjeet muokkaukseen löytyvät kyseisen tiedoston alusta sekä käyttöohje-kohdasta. Halutessaan pelaaja voi muokata myös “saved_games.txt”-tiedostoa ja ns.tallentaa olemattomia pelejä, mutta tiedosto on hiukan vaikealukuisempi, koska näin ei ole tarkoitus tehdä.
Kuvatiedostot taas ovat png-tyyppisiä cropattuja kuvia, ja niitä käyttäjä voi halutessaan tallentaa samaan hakemistoon lisää ja sisällyttää peliin.

Testaus

Pelin graafisten osien toiminta on testattu ajamalla peliä, ja tutkimalla liikkuvatko viholliset oikealla tavalla kaikissa tilanteissa, ampuvatko tornit vihollisia oikein ja kuolevatko viholliset kun niiden on tarkoitus kuolla. Visuaalisten havaintojen lisäksi on tutkittu tarkasti print-käskyjen avulla, että erilaiset sijainnit ovat oikein, elinvoima vähenee odotetun määrän, sekä että tornit maksavat oikean summan. Näitä testauksia on tehty aina kun luodaan pienikin uusi osa, jolloin suurelle kokonaisuudelle ei ole tarvinnut alkaa jälkeenpäin keksimään testausmenetelmää. Testauksessa ei ole hyödynnetty yksikkötestejä, koska ne katsottiin tarpeettomiksi.
Testauksen ainut aukko on tallennusformaatin lukijan käyttäytyminen, kun tallennetaan useita samannimisiä pelejä. Tätä ei testattu, sillä tallennus-metodi kirjoittaa vanhan tiedoston päälle, ja mikäli tallennettuna on jo samanniminen peli, niin se poistaa vanhan pelin tiedot.

Testauksessa edettiin pitkälti suunnitelman mukaan. Ainoana erona oli, että kantamaa ei testattu värjäämällä alue punaiseksi, vaan kun ammus näytettiin viivana, niin tämäkin asia tuli testattua samalla.

Ohjelman tunnetut puutteet ja viat

Ohjelmakoodia tehdessä opin vasta puolessa välissä, että pääsen käsiksi eri luokan olioiden sisällä oleviin muuttujiin ilman erillisiä funktioita, jotka palauttavat ne. Tämän vuoksi moni luokka sisältää paljon get_jotakin-tyyppisiä funktioita. En korjannt näitä kaikkia pois, sillä ohjelma toimi näin hyvin ja niiden muuttamisessa olisi saattanut tulla virheitä, joiden korjaaminen olisi vienyt paljon aikaa.
Lisäksi eräs puute on se, ettei yhdellä kierroksella voi olla kuin tietyn tyyppisiä vihollisia, mikäli jatkaisin projektia niin päivittäisin tähän uudenlaisen tallnnusformaatin, joka mahdollistaisi useammanlaisen vihollisen yhdellä kierroksella. Graafisen liittymän puutteita ovat myös tornin asettamisen kömpelyys. Siihen kaikken paras tapa olisi ollut mielestäni drag and drop-tyyppinen ratkaisu, mutta tein kyseiset metodit projektin alkupuolella, ja taidot eivät tähän riittäneet vielä silloin. Nyt uskoisin, että tälläinen voisi onnistua.
Isoin ohjelman puute on kuitenkin se, että jos vihollisten reitti haaraantuu, niin viholliset vaitsevat kaikki saman reitin, eli mikäli jonkinlaista valinnanvapautta on, niin koodi ei huomaa sitä. Tämä olisi ensimmäisiä päivityskohteita koodissa, mikäli jatkaisin projektia.
Graafiseen liittymään liittyen pelin loppumisen yhteyteen olisi voinut lisätä vaihtoehdon aloittaa uusi peli. Ja näiden valikoiden ulkonäköä ja funktionaalisuutta olisi voinut parannella esimerkiksi mahdollistamalla takaisinmenemisen.
Viimeinen tunnettu puute on se ettei ohjelma käsittele syntyviä erroreita lainkaan. Nämä on kuitenkin jätetty pois, sillä ohjelman testauksen yhteydessä erroreita ei syntynyt, eli ainoa keino saada niitä syntymään on muuttaa tiedostojen tietoja väärin. Eräs kehitystarve olisikin lisätä näihin rakenteita jotka aktivoituvat erroreiden syntyessä ja opastavat käyttäjää korjaamaan syntyneet virheet.

3 parasta ja kolme heikointa kohtaa

Heikoin kohta on mielestäni viholli-olioiden reitin valitseminen, sillä se on melko yksinkertainen, vaikkakin toimiva, ellei kartta sisällä jonkilainsta labyrinttiä. Toinen heikko kohta on vihollisten liikkeen kankeus. Kolmas hieman huono ratkaisu oli, että piste johon vihollisoliot ilmestyvät etsitään Round-luokassa, sillä tällöin se tehdään niin monta kertaa kuin kierroksia on. Veisi vähemmän resursseja tehdä tämä esimerkiksi Game-luokassa, jolloin se tarvitsisi tehdä vain kerran per kartta. Käytännössä tästäkään ei ole kuitenkaan huomattavissa määrin haittaa tietokoneelle.
Paras kohta on mielestäni vihollisen luomiseen liittyvä mahdollisuus antaa käyttäjälle helppo tapa luoda omia vihollistyyppejään. Koodi ei itsessään ole haastava tehdä, vaan erittäin simppeli, mutta sekin on mielestäni hyvä asia, sillä virheiden mahdollisuus pienenee. Tämän ainoa heikko kohta on, ettei ohjelma osaa reagoida siihen, jos vihollisen tyypin (+".png") nimistä tiedostoa ei löydykkään hakemistosta.
Toinen erittäin hyvä puoli ohjelmassani on sen laajennettavuus helposti erilaisten asetustiedostojen kautta. Torneja kuvaavalle luokalle voidaan tehdä sama laajennus kuin vihollista kuvaavalle luokalle, joka mahdollistaa erinäköisten tornien ottamisen mukaan peliin ja niiden erilaiset tulivoimat. Pienellä vaivalla osa torneista saataisiin myös ampumaan nopeammin/hitaammin.

Poikkeamat suunnitelmasta

Graafinen käyttöliittymä poikkeaa hieman suunnitelluta, kute myös toteutunut luokkajako. Käyttöliittymä oli mielestäni toimiva myös tälläisellä ulkonäöllä ja luokkajaossa taas oli hiukan puutteita ja muutama mielestäni loppujenlopuksi turha luokka.

Tarkoituksena oli myös, että tornit laskisivat tuhoamiensa vihollisten määrät ja, että sen ampumisnopeutta olisi voinut kehittää, mutta tälläistä ei ehditty toteuttaa. Sama päti myös eri tyyppisten vihollisten erilaisiin nopeuksiin sekä nappulaan joka mahdollistaisi pelin pyörittämisen nopeammin.
Ajankäytön suunnitelmat eivät toteutuneet ollenkaan, sillä asioiden opetteleminen vei paljon enemmän aikaa kuin olin odottanut, ja muut kiireet veivät aikaa projektin tekemiseltä. Suunniteltu toteutusjärjestys ei ollut toimiva, joten siitä luovuttiin. Käytännössä ohjelman osa-alueita toteutettiin siinä järjestyksessä kun niitä tarvittiin pelin aloituksen ja pelaamisen kannalta.

Toteutunut työjärjestys ja aikataulu

Projekti toteutettiin luomalla ensin Game- ja Square-luokat, osa filereader-metodeista sekä GUI:sta osa joka vastaa pelilaudan piirtämisestä. Myöhemmin kartan piirtämistä vielä muutettiin, kun siinäkin otettiin käyttöön QPixmap. Tämän jälkeen luotiin Tower- ja TowerGraphicsItem-luokat. Sen jälkeen tehtiin filereader-metodi, joka lukee kierrosten tiedot ja tehtiin Round-luokka, sekä Enemy- ja EnemyGraphicsItem-luokat. GUI-luokkaa toki täydennettiin joka välissä. Kun kartalle oli saatu asetettua tornit ja sen jälkeen saatu viholliset liikkumaan kartalla, niin aloin työstämään vihollisten ampumista ja lopuksi tehtiin Start_window luokka.
3.5. Projekti aloitettiin alusta, sillä aikaisemmin tekemäni alustavat toiminnot, joita en ollut tallentanut gittiin, olivat tehty väärällätavalla.
4.5. Tornit saatu tulostettua oikeille paikoilleen sen mukaan minkä tasoisia ne ovat (kolmioina)
5.5. ajoneuvo saatu ruudulle, mutta ei liiku.
9.5.ajoneuvo liikkuu, ja tornit ovat myös Pixmap-tyyppisi, eivätkä kolmioita. Ohjelma toimii.
12.5. Lopulliset hienosäädöt tehty.

Arvio lopputuloksesta

Vaikka projekti on tehty melko lyhyessä ajassa niin siihen on käytetty noin 80 tuntia aikaa, sillä olen tehnyt niin pitkää päivää. Mielestäni projektini lopputulos on todella hyvä, sillä ohjelma täyttää vaaditut kriteerit ja siinä on lisäominaisuuksia myös. Tälläisiä ovat esimerkiksi sen helppo laajennettavuus ja mahdollisuus käyttäjälle luoda omia reittejä ja omia vihollisia. Tehtävänannossa mainittiin, että grafiikat voivat olla palloja ja neliöitäkin, joten plussaa ovat myös pelin huomattavasti sitä paremmat grafiikat, sillä se kääntää tornia sen mukaan minne torni ampuu ja piirtää sen lisäksi ammuksen. Toki paranneltavaa on paljon, mutta ne ovat lähinnä graafista hienosäätöä ja lisäominaisuuksien lisäilyä, eli ne eivät ole oleellisia puutteita.

Luokkajaon yhteydessä Enemy ja EnemyGraphicsItem luokat olisi voinut yhdistää yhdeksi luokaksi selkeyden vuoksi. Myös Tower- luokalle olisi voinut tehdä saman,mutta se olisi vaatinut hieman korvaavia toimenpiteitä esimerkiksi tallennetun pelin lataamiseen liittyen.
Muuten luokkajako oli mielestäni hyvä ja toimiva. GUI-luokasta tuli huomattavasti suurempi kuin muista luokista, mutta ainakaan itseä se ei haitannut.

Viiteet

http://zetcode.com/gui/pyqt5/
https://commons.wikimedia.org/wiki/File:T-34_top_view_vector.svg
https://pixabay.com/en/cannon-artillery-heavy-weapon-gun-145684/
https://pixabay.com/p-303765/?no_redirect
http://wallpaper-gallery.net/single/grass-texture/grass-texture-20.html
http://www.publicdomainpictures.net/view-image.php?image=187658&picture=pisek-textury
http://www.cs.hut.fi/~ttsirkia/Python.pdf

https://grader.cs.hut.fi/static/y1/
https://doc.qt.io/qt-5/qgraphicspixmapitem.html
https://pythonspot.com/en/pyqt5/
https://wiki.python.org/moin/PyQt