Tampereen yliopisto

Miikka Mensio ja Markus Mensio

BUSDESTROYER 2000 PELI

Ohjelmointi 3 -kurssiprojekti

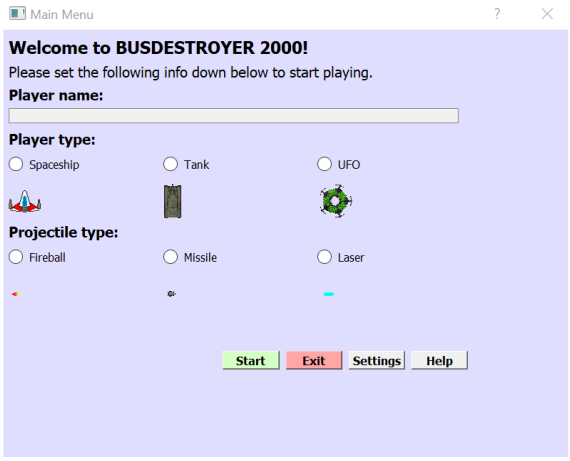
TIE-02402 Ohjelmointi 3: Tekniikat
Syksy 2020

SISÄLLYSLUETTELO

1.1 Pelin säännöt ja peliohjeet 3
1.2 Luokkien vastuunjako 4
1.2.1 Luokkakaavio 4
1.2.2 Spesifisemmät luokkakuvaukset: 4
1.3 Lisäosat 7
1.4 Ryhmän työnjako 9
1.5 Lähteet 10

1.1 Pelin säännöt ja peliohjeet

BusDestroyer 2000 on yksinpeli, jossa tarkoituksena on ampua ja sitä kautta tuhota Tampereen keskustan kartalla reaaliaikaisesti liikkuvia Nysse-busseja (siniset nelikulmiot) ja matkustajia (violetit hahmot) ja samanaikaisesti kerätä punaisia bonustimantteja kartalta. Pelissä oikeita linjoja ajavat Nysset liikkuvat pelikartalla oikeilla reiteillä todenmukaisen aikataulun mukaan. Kartalla autonomisesti liikkuvat matkustajat voivat nousta pysäkeiltä (kuvattu pelissämme punaisilla lipuilla kartalla) Nysseihin sekä voivat jäädä toisilla pysäkeillä bussien kyydistä pois. Pelimme pelaaja hallitsee kolmesta eri vaihtoehdosta (avaruusalus, tankki tai ufo) valitsemaansa pelihahmoa, joka voi olla vuorovaikutuksessa kartalla liikkuvien Nyssejen, matkustajien ja bonustimanttien kanssa. Pelin alkaessa pelaajalle avautuu MainMenuDialog (kuva 1), jossa pelaajan täytyy asettaa itselleen nimimerkki pistekirjanpitoa varten, sekä valita pelaaja- ja ammustyyppi.



Kuva 1: MainMenuDialog eli pelin ns. aloituskonfiguraatiodialogi.

Valittavia pelaajatyyppejä on kolme: 1) avaruusalus (kuva 2), 2) tankki (kuva 3) ja ufo (kuva 4). Ammustyyppejä on myös kolme vaihtoehtoa, jotka ovat tulipallo (kuva 5), ohjus (kuva 6) ja laser (kuva 7). Pelaaja-ammus-kombinaatioita ei ole mitenkään rajoitettu, vaan pelaaja voi valita itselleen mieluisimman yhdistelmän. Pelaajan on mahdollista avata helpDialog painamalla aloitusmenun Help-nappia, jolloin pelaajalle avautuu lisätietoa mm. pelistä, sen tavoitteista ja säännöistä.

Kuva 2

Kuva 3

Kuva 4

Kuva 5

Kuva 6

Kuva 7

BusDestroyer 2000:n tavoitteena on aikaa vastaan kerätä mahdollisimman monta pistettä tuhoamalla Tampereen keskustan karttaa kuvaavalla pelialueella liikkuvia Nyssejä ja matkustajia, sekä kerätä kartalle satunnaisesti ilmestyviä bonustimantteja. Peli-ikkunassa (gameWindow) bussipysäkit on merkattu punaisilla lipuilla (kuva 8), matkustajat violeteilla hahmoilla (kuva 9), Nysse-bussit sinisillä nelikulmioilla (kuva 10) ja bonustimantit punaisilla timanteilla (kuva 11).

Kuva 8

Kuva 9

Kuva 10

Kuva 11

Pelin alkaessa varsinaisen peli-ikkunan yläpaneelissa esitetään pelaajalle reaaliajassa kerätyt pisteet ja jäljellä oleva aika sekunnin tarkkuudella QLCDNumber -widgeteissä. Pelatessa kartalle ilmestyy punaisia bonusobjekteja satunnaisiin x-koordinaatteihin 4 sekunnin välein, joita kuvataan kartalla pienillä punaisilla timanteilla. Ilmestyessään kartalle em. timantit lähtevät liikkumaan kartalla alaspäin QGraphicsViewin positiivisen y-akselin suuntaan, ja jos pelaaja onnistuu keräämän timantin itselleen joko liikkumalla sen lokaatioon tai ampumalla, niin hänen pistesaldonsa kasvaa

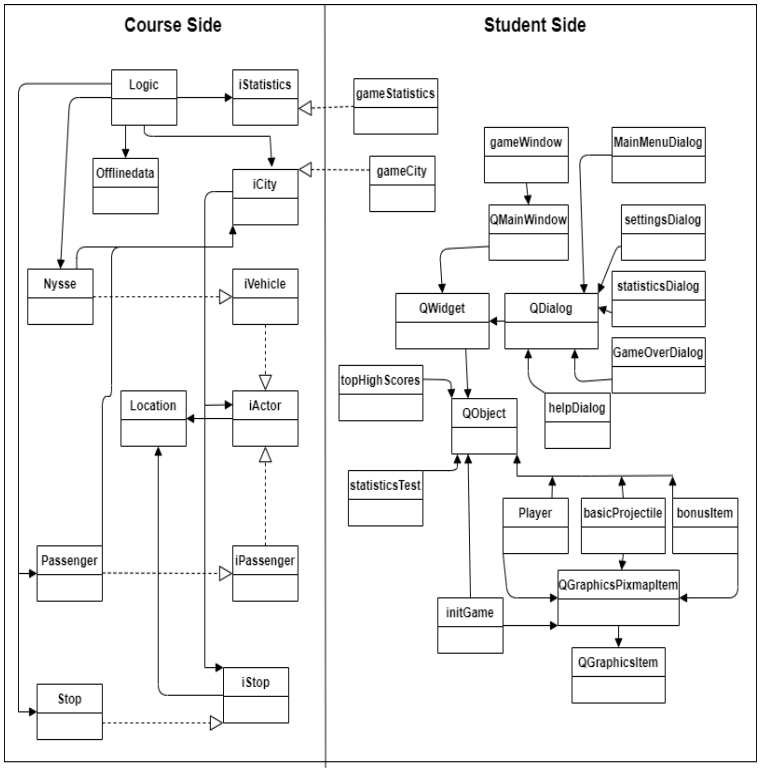
10 pisteellä. Lisäksi jokaisesta tuhotusta matkustajasta tai Nyssestä pelaaja saa 10 pistettä lisää pistesaaliseensa.

Peli loppuu, kun peliajastimen aika menee nollaan, jonka jälkeen pelaajalle näytetään erillinen GameOverDialogi, johon on myös integroitu huippupistesysteemi, jossa näytetään 10:n parhaan pelaajan nimimerkki sekä pisteet. GameOverDialogista on mahdollista myös Statistics-QPushButtonia painamalla navigoida statisticsDialogiin, jossa pelaajalle näytetään pelin aikana kerätyt pisteet, kerättyjen timanttien, tuhottujen Nyssejen ja matkustajien lukumäärä. Kyseinen statisticsDialog käyttää hyväkseen luomaamme iStatisticsin aliluokkaa gameStatiscticsia, joka siis implementoi rajapinnan iStatistics virtuaalifunktiot. GameOverDialogista on mahdollista myös käynnistää uusi peli, tai mahdollisesti sulkea koko ohjelma. Huippupistesysteemin datasäiliöksi olemme luoneet erillisen top10highscores.txt -tiedoston, johon pelaajan nimimerkki ja skaalatut pisteet tallennetaan.

1.2 Luokkien vastuunjako

1.2.1 Luokkakaavio

Alla olevassa kuvassa 12 esitetään BusDestroyer 2000 -pelimme luokkien vastuujakoa kuvaava luokkakaavio, jossa on eritelty kurssin puolen (Course Side) tarjoama koodi omasta koodistamme (Student Side).



Kuva 12: Pelin luokkakaavio (muokattu kurssin puolen tarjoamasta luokkakaaviosta)

1.2.2 Spesifisemmät luokkakuvaukset:

MainMenudialog -luokka (ks. kuva 1):
Pelin ns. konfiguraatioikkuna, joka kysyy pelaajalta tarvittavia tietoja ennen varsinaisen pelin pääikkunan käynnistämistä. Tarjoaa myös navigointimahdollisuuden muihin näkymiin: settingsDialog ja helpDialog. MainMenuDialogin pääasiallisena tehtävänä on tarjota BusDestroyer 2000:n pelaajalle pääkonfiguraatiovalikko, jossa pelaaja voi asettaa itselleen nimimerkin, sekä valita pelaaja- ja ammustyypin. Pelin käynnistämisen lisäksi pelaajalla on mahdollista muuttaa pelin asetuksia settingsDialogissa tai avata erillinen helpDialog, joka kertoo pelaajalle oleellista tietoa mm. pelistä, sen tavoitteista, säännöistä ja pelissä käytettävistä näppäimistä.

gameWindow -luokka (itse toteutettu kartanpiirto):
Varsinainen *itse toteutettu* QMainWindow peli-ikkuna, joka sisältää mm. Score ja Time-Left QLCDNumber -widgetit, jotka kertovat pelaajan pisteet ja jäljellä olevan peliajan sekä valitun ikkunan koon mukaan skaalautuvan Tampereen keskustaa kuvaavan pelikartan.

Pelin pääikkunan toteuttava gameWindow on koko pelin selkäranka, ja sen vastuu on suuri, sillä se mahdollistaa varsinaisen gameplayn ja mm. näyttää pelikartalla käyttäjän valitseman pelihahmon, Nysset, matkustajat ja uniikit bonustimantit. gameWindow alustaa QGraphicsScenen ja QGraphicsViewin sekä määrittelee niiden asetukset (mm. peliruudun koko aluksi 800x600 pikseliä, peli-ikkunan koon skaalautuvuus, kartan asettaminen taustakuvaksi).

Peli-ikkuna emittoi neljän sekunnin välein bonustimanttigeneraattorille signaaleja, jolloin uusi timantti ilmestyy kartalle. Lisäksi gameWindowin vastuuna on toteuttaa _gameTimer peliajastin, joka lopettaa pelin, kun ennaltamäärätty aika kuluu loppuun. gameWindowin konstruktorissa emittoidaan sekunnin välein timeout -signaali metodille, joka vähentää peliaikaa. Ruudunpäivityssignaali emittoidaan 20 millisekunnin intervalleissa graphicsViewin viewportille, jotta saavutettaisiin sulavampi kuvataajuus.

Luokan gameWindow konstruktorissa alustetaan myös Logic-olio, jolle kutsutaan initGame -luokan metodia initLogic, joka käytännössä alustaa ja konstruoi CourseSiden back-endin, jolloin kurssin puolen integraatio omaan ohjelmaan mahdollistuu. Myös Player -olio eli kartalla liikkuva pelihahmo luodaan gameWindowin konstruktorissa.

Luokassa implementoidaan myös resizeEvent -metodi, joka mahdollistaa pelikartan koon skaalautumisen, kun käyttäjä suurentaa tai pienentää peli-ikkunaa. Peli-ikkunan kokoa muutettaessa uusi ikkunan leveys ja korkeus päivitetään automaattisesti sekä gameWindowin että muiden relevanttien luokkien attribuutteihin, mikä mahdollistaa mm. sen, että pelaaja ei voi liikkua ulos kartalta missään tilanteessa.

initGame -luokka:
Luokka initGame on varsinaisen CourseSide -integraation toteuttava luokka eli sen vastuuna on piirtää gameCity -luokassa datasäiliöön tallennetut iActorit (Nysset ja matkustajat) ja iStopit (bussipysäkit) pelikartalle. Luokassa initGame päädyimme ratkaisuun, jossa tallensimme sanakirjaan (std::map) avaimeksi älykkään osoittiminen joko iActoriin tai iStopiin ja arvoksi QGraphicsPixmapItemin, joka mahdollistaa pysäkkien, Nyssejen ja matkustajien lisäämisen näkyviin sceneen. Luokassa lisätään myös omat Qt:n resurssitiedostosta löytyvät png-kuvat jokaista kartalta löytyvää pysäkkiä, Nysseä ja matkustajaa kohti.

Kartan tilannekuvan päivittämisestä pitää huolen konstruktorissa 200 millisekunnin välein emittoitu signaali, joka yhdistetään moveSceneActors -slottiin, joka siis mahdollistaa kartalla olevien Nyssejen ja matkustajien realistisen liikkumisen kartalla. Jos pysäkin, Nyssen tai matkustajan koordinaatti ei ole peli-ikkunan dimensioiden määrittelemissä rajoissa, niin aktoreita ei piirretä kartalle, joka on muistinhallinnallisesti tehokkaampaa.

Luokassa initGame toteutetaan lisäksi CourseSiden tarjoaman **CreateGame**:n metodi **std::shared_ptr<ICity> createGame()**, jossa siis luodaan uusi osoitin gameCityyn, joka lopuksi palautetaan. Luokan tarjoama jäsenfunktio **initLogic** on myös oleellinen, sillä se alustaa CourseSiden tarjoaman Logic -olion ja lopuksi kutsuu Logicin jäsenfunktiota **finalizeGameStart**. Metodia *initLogic* kutsutaan pelin gameWindow pääikkunassa, jolloin kurssin puolen back-end käynnistyy taustalla.

gameCity -luokka:
Luokka gameCity periytyy CourseSiden tarjoamasta rajapinnasta iCity ja implementoi iCity:n virtuaalifunktiot. Projektiryhmämme päätyi gameCity -luokassa ratkaisuun, jossa kaikkiin (iStop) pysäkkeihin osoittavat älykkäät osoittimet ja kaikkiin aktoreihin (iActor) osoittavat älykkäät osoittimet tallennettiin omiin julkisiin attribuuttivektoreihinsa (*allStops* ja *allActors* -vektorit).

gameStatisctics -luokka:
Luokka gameStatistics periytyy CourseSiden tarjoamasta rajapinnasta iStatistics. Luokka toteuttaa iStatisticsin virtuaalifunktiot, jonka lisäksi se mm. tarjoaa funktion kerättyjen bonustimanttien lukumäärän kirjanpitoon.

Luokan gameStatistics pääasiallinen tehtävä on manageroida pelin sisäisten ja ulkoisten tapahtumien kirjanpitoa, eli ts. se tallentaa pelaajaa mahdollisesti kiinnostavaa statistiikkatietoa. Luokan gameStatistics tallentamaa dataa käytetään mm. pääpeli-ikkunan piste-widgetissä, sekä statisticsDialogin statistiikkaesityksessä.

settingsDialog -luokka:
Käyttöliittymä-Dialogi, joka perii QDialogin. Asetusvaihtoehtoja on kaksi: peliajan pituus yhdestä kolmeen minuuttiin (QComboBox) (oletusarvo 2 min) ja musiikkiefektit päälle/pois (QCheckBox) (oletusarvo musiikit pois päältä).

AsetusDialogissa valinnat tehtyään pelaajan täytyy erikseen painaa Save-nappia, jotta halutut asetukset tallentuisivat. Asetusten tallennuksen jälkeen pelaaja voi palata Back-nappia painamalla takaisin MainMenuDialogiin. Ellei Save-nappia erikseen paineta, niin tarkoituksenmukaisesti käytetään oletusarvoisia asetuksia

helpDialog -luokka:
Käyttöliittymä-Dialogi, joka perii QDialogin. QDialog, jossa pelaajalle avautuu lisätietoa pelistä, sen tavoitteista ja säännöistä QTextBrowseriin. Käyttäjä voi poistua helpDialogista takaisin MainMenuDialogiin painamalla Dialogin oikeasta yläkulmasta löytyvää punaista raksia (QPushButton). Tämä luokka on pitkälti implementoitu Qt Designerissa.

GameOverDialog -luokka:
Käyttöliittymä-Dialogi, joka perii QDialogin. GameOverDialogissa näytetään 10 parhaan pelaajan nimimerkki ja peliajan mukaan skaalatut pisteet, jolloin pisteet ovat vertailukelpoisia, vaikka pelaajat olisivatkin pelanneet eri mittaisia sessioita. Huippupistesysteemi perustuu topHighScores -luokan toiminnallisuuteen, jossa pelaajan pisteet haetaan gameStatistics -oliosta. Pisteiden skaalaus tehtiin kaavalla

$$int\ skaalatutPisteet = \frac{pisteet}{peliaika\ (\mathrm{min})} \qquad \text{(kaava 1)}$$

GameOverDialogin tehtävä on ilmoittaa pelaajalle pelin loppumisesta ja se tarjoaa pelaajalle mahdollisuuden katsoa tilastotietoja pelisessiostaan painamalla Settings-nappia, jolloin statisticsDialog näytetään käyttäjälle.

GameOverDialogista on mahdollista myös käynnistää uusi peli painamalla Play Again -nappia, tai vaihtoehtoisesti sulkea koko ohjelma painamalla Close-nappia. Huippupistesysteemin datasäiliönä olemme luoneet erillisen top10highscores.txt -tekstitiedoston Game -kansioon, johon pelaajan nimimerkki ja peliajan mukaan skaalatut pisteet tallennetaan muodossa: *nimimerkki: skaalatatutPisteet*.

statisticsDialog -luokka:
Käyttöliittymä-Dialogi, joka perii QDialogin. Luokan tarkoituksena on näyttää pelaajalle hänen pelissä keräämänsä pisteet sekä kerättyjen timanttien, tuhottujen Nyssejen ja matkustajien lukumäärä. Luokka statisticsDialog käyttää hyväkseen luomaamme iStatisticsin aliluokkaa gameStatisctics, joka siis implementoi rajapinnan iStatistics virtuaalifunktiot.

Luokan tarjoama data haetaan gameStatistics -luokasta, joka huolehtii pelinaikaisesta statistiikkakirjanpidosta ja tallentaa muutokset julkisiin (public) attribuutteihinsa, joita muut luokat voivat hyödyntää.

topHighScores -luokka:
Luokan vastuuna on lukea ja kirjoittaa top10highscores.txt -tekstitiedostoon pelaajan nimimerkki ja pelaajan saamat *skaalatut* pisteet muodossa *nimimerkki: skaalatutPisteet,* jossa *skaalatutPisteet* lasketaan ohjelmassamme edellä *kappaleessa 1.3.2* esitellyllä kaavalla 1.

Luokka pitää huolen siitä, että kaikkien pelisessioiden nimimerkki-pistesaldo -yhdistelmät tallentuvat tekstitiedostoon kronologisessa järjestyksessä, mutta GameOverDialogin Top 10 High Scores -QTextBrowserissa näytetään ainoastaan 10 kaikkien aikojen parasta pelaajaa skaalattujen pisteiden perusteella. Luokalla on erillinen sortAndDisplay -funktio, joka järjestää tekstitiedostosta löytyvät pelaajat suurimmasta pistesaldosta pienimpään 10 pelaajan osajoukkoon ja tallentaa em. sortatun top 10 -merkkijonon julkiseen *strScores* -attribuuttiinsa, joka sitten esitetään pelaajalle GameOverDialogin QTextBrowserissa.

Player -luokka:
Pelin päätoimijaluokan toteuttava ja sen toiminnallisuuden tarjoava luokka, joka periytyy QObjectista ja QGraphicsPixmapItemista. Player luokka luo pelaajan valitseman pelaajatyypin mukaisen päätoimijan peli-ikkunan kartalle ja mahdollistaa pelaajan *tasaisen* liikkumisen kartalla näppäimistön nuolinäppäimistä. Ampuminen tapahtuu välilyöntinäppäintä painamalla, jolloin Player -luokka generoi uuden basicProjectile -olion. Pelaaja pystyy liikkua kartalla yksittäisin nuolinäppäinpainalluksin, mutta liikkumismekaniikka tukee myös näppäimen pohjaanpainamista, jolloin pelaaja liikkuu yhdenmukaisesti kartalla. Luokan Player konstruktori emittoi 50 millisekunnin välein timeout-signaaleja Player-olion liikkumisesta vastaavalle funktiolle.

Pelaajaluokka myös tarkistaa, onko pelaaja liikkunut bonustimantin positioon, ja jos on, niin se kasvattaa pelaajan pisteitä 10:llä, poistaa kartalta ja ohjelman muistista bonustimantin ja tallentaa em. törmäyksestä tiedon tilastokirjanpitoon gameStatistics-olioon. Lisäksi pelaajaluokka mm. huolehtii ääniefektikonfiguraatioista, pelattavan hahmon leveyden ja pituuden tarkistamisesta sekä siitä, että pelaaja ei voi liikkua yli kartan rajojen oli kyseessä sitten mikä tahansa peli-ikkunan näytönkoko.

basicProjectile -luokka:
Pelaajan ampuman ammuksen toteuttava luokka, joka periytyy QObjectista ja QGraphicsPixmapItemista. Luokka basicProjectile pääasiallinen tehtävä on luoda ammusolio pelaajan ampuessa peli-ikkunassa välilyöntinäppäintä painamalla. Muita luokan tehtäviä on lisätä pelaajan MainMenuDialogissa valitsema ammustyyppikuva ammuksen ikoniksi sekä tallentaa ammuksen dimensiot (leveys ja korkeus) sen attribuutteihin. Luokan basicProjectile konstruktori emittoi 50 millisekunnin välein timeout-signaaleja ammuksen liikkumisesta vastaavalle funktiolle.

Ammusoliolla on myös mahdollisuus liikkua ylöspäin vakionopeudella ja jos se kohtaa matkallaan bonustimantin, matkustajan tai Nyssen, niin se lisää pelaajan pistesaldoa kymmenellä ja poistaa kartalta sekä itsensä että kohdanneen objektin ja deletoi molemmat törmäyksen osapuolet ohjelman muistista. Lisäksi ammuksen ja Nyssen, matkustajan tai bonustimantin välisestä törmäyksestä tallentuu tieto tilastokirjanpitoon gameStatistics-olioon.

bonusItem -luokka:
Playerin lisäksi pelin toisen uniikin toimijan toteuttava luokka, joka periytyy QObjectista ja QGraphicsPixmapItemista. Luokan tehtävänä on generoida QGraphicsPixmapItem -bonustimantti satunnaiseen x-koordinaattiin senhetkisen peli-ikkunan koon sallimissa rajoissa. Luokan vastuuna on myös liikuttaa timanttia kartalla vakionopeudella alaspäin QGraphicsViewin positiivisen y-akselin suuntaan ja poistaa timantti kartalta, jos pelaaja ei ennätä sitä keräämään, ennen kuin timantti liikkuu ulos kartalta. Luokka asettaa myös QPixmap -perustaisen png-kuvan olion ikoniksi, jotta se erottuisi kartalta.

statisticsTest -luokka:
Qt:n testiluokka, joka sisältää yksikkötestit iStatisticsin aliluokan gameStatistics metodeille. Testeissä testataan mm. pelinaikaisen statistiikkakirjanpidon toimivuutta: pisteidenlaskenta, tuhottujen Nyssejen ja matkustajien kirjanpito ja pelaajan keräämien timanttien kirjanpito.

1.3 Lisäosat

Tasainen ruudunpäivitys:
QGraphicsViewin viewportin update()-funktiota kutsutaan Main Windowissa (gameWindow -luokka) tasaisin intervallein QTimerin timeout -signaalin avulla. Näin saavutetaan tilanne, jossa pelialueelle päivitetään tasaisin väliajoin kertyneet muutokset. Lisäksi *initGame* luokka käyttää QTimerin timeout-signaalia hyödyksi kartalta löytyvien Nyssejen, matkustajien ja bonustimanttien kirjanpidossa, ja tuhotut objektit poistetaan kartalta, eli siis ts. ainoastaan tietyllä ajanhetkellä olemassa olevat aktorit piirretään kartalle.

QGraphicsViewin ruudunpäivitys on lisäksi asetettu NoViewportUpdate -tilaan, jotta näytöllä olevat objektit liikkuisivat mahdollisimman tasaisesti ja että latenssia eli viivettä olisi gameplayssa mahdollisimman vähän. Nimensä mukaisesti NoViewPortUpdate ei päivitä ruutua autonomisesti ilman, että ViewPortin update() -metodia kutsutaan ohjelmassa. Lisäominaisuus on implementoitu gameWindow -luokassa.

Minimaalinen ruudunpäivitys:
Ruudunpäivityksen ajan hetkellä QGraphicsScenestä päivitetään ainaostaan QGraphicsViewin ViewPorttiin tulleet muutokset. Lisäksi pelaajan pisteet ja peliaikalaskuri päivittyy tarpeen mukaan eli silloin, jos pelaajan pisteet kasvavat tai kun sekuntikellon aika pienenee yhdellä sekunnilla. Koko karttaa ei siis piirretä aina uudelleen yksittäisten toimijoiden tilan muuttuessa. Lisäominaisuus on implementoitu lähinnä gameWindow -luokassa.

Pelihahmon tasainen liike:
Pelissä pelaajan valitsemalle hahmolle ja ammukselle on ohjelmassamme toteutettu tasainen liike nuolinäppäimin ja ampumismahdollisuus välilyöntinäppäintä painamalla. Pelaaja voi liikkua joko yksittäisin näppäinpainalluksin, mutta ohjelmamme tukee myös nuolinäppäinten pohjaan painamista, jolloin alus liikkuu yhtäjaksoisesti haluttuun suuntaan. Pelaaja voi liikkua neljään eri ilmansuuntaan (länsi, itä, pohjoinen ja etelä).

Lisäksi pelaajan nopeutta voi kasvattaa (painamalla ”+”-näppäintä) tai pienentää (painamalla ”-” näppäintä) järkevissä määrin pelin aikana. Lisäominaisuus on implementoitu Player -luokassa.

Pelin näytön automaattinen koon mukainen skaalautuminen:
Pääpeli-ikkunaamme (gameWindow -luokka) voi halutessaan raahata eri kokoiksi, ja suurentaa koko näytön leveydelle, jolloin kaikki pelikartan elementit skaalautuvat automaattisesti peli-ikkunan kokoon. Jos pelaaja suurentaa näytön täyteen mahdolliseen kokoonsa, niin kaikki QMainWindowissa olevat elementit ja hahmot skaalautuvat täyteen ruutuun mukaan lukien Tampereen keskustaa kuvaava pelikartta. Lisäominaisuus on implementoitu gameWindow -luokassa.

Top10-lista / Huippupistesysteemi:
Kaikkien aikojen 10 parasta pelaajaa skaalattujen pisteiden mukaan esitetään pelaajalle pelin päättyessä GameOverDialogissa. Kaikkien pelisessioiden nimimerkki-skaalattuPistesaldo -yhdistelmät tallentuvat tekstitiedostoon kronologisessa järjestyksessä, mutta GameOverDialogin Top 10 High Scores -QTextBrowserissa näytetään ainoastaan 10 kaikkien aikojen parasta pelaajaa skaalattujen pisteiden perusteella.

GameOverDialogin QTextBrowserissa top10highscores.txt -tekstitiedostosta löytyvät pelaajat järjestetään suurimmasta pistesaldosta pienimpään 10 pelaajan osajoukkoon. Tekstitiedoston sisältämä data säilyy lisäksi pelin sulkemisen ja uudelleenkäynnistyksen yli.

Huippupistesysteemi on toteutettu siten, että se tallentaa ja kirjoittaa pelaajan piste- ja nimimerkkitiedot sijaintiin, jossa Qt:n Build-Directory sijaitsee, ja automaattisesti luo uuden top10highscores.txt -tekstitiedoston, ellei se sitä ennestään löydä. Tämä on toteutettu sen takia, että voitaisiin varmistaa huippupistesysteemin toiminta oli pelaajan Build Directory -polku mikä tahansa. Lisäominaisuus on implementoitu lähinnä topHighScores – ja GameOverDialog -luokissa.

Laaja liikenneväline- ja ammusvalikoima (3 + 3 vaihtoehtoa):
Valittavia pelaajatyyppejä on kolme kappaletta: 1) avaruusalus (kuva 2), 2) tankki (kuva 3) ja ufo (kuva 4). Ammustyyppejä on myös kolme vaihtoehtoa, jotka ovat tulipallo (kuva 5), ohjus (kuva 6) ja laser (kuva 7). Pelaaja-ammus-kombinaatioita ei ole mitenkään rajoitettu, vaan pelaaja voi valita itselleen mieluisimman yhdistelmän.

Pelaajalla on siis monipuolinen liikenneväline ja -ammusrepertuaari käytettävissään liikkuessaan pelin kaupungilla. Lisäominaisuus on implementoitu Player -luokkaan (pelaajatyypit) ja basicProjectile -luokkaan (ammustyypit).

Ylimääräinen uniikki toimija: bonusItem -luokka (bonustimantti):
Peliin on ohjelmoitu bonustimanttigeneraattori, joka generoi QGraphicsPixmapItem -bonustimantteja satunnaisiin x-koordinaatteihin senhetkisen peli-ikkunan koon sallimissa rajoissa.

Kartalle syntyneet timantit liikkuvat kartalla vakionopeudella alaspäin QGraphicsViewin positiivisen y-akselin suuntaan ja timantti poistetaan kartalta ja ohjelman muistista, jos pelaaja ei ennätä sitä keräämään, ennen kuin se liikkuu ulos peli-ikkunasta.

Bonustimantille asetetaan QPixmap -perustainen png-kuva olion ikoniksi, jotta se erottuisi kartalta paremmin. Lisäominaisuus on implementoitu lähinnä bonusItem -luokassa, mutta gameWindow -luokassa tapahtuu timanttien generointi ja lisääminen QGraphicsSceneen.

settingsDialog ja asetusvaihtoehdot (musiikki ja peliaika):
settingsDialogiin navigoidaan kuvassa 1 näkyvästä ”Settings”-napista. Asetusvaihtoehtoja on kaksi: Peliajan pituus yhdestä kolmeen minuuttiin (QComboBox) (oletusarvo 2 min) ja Musiikkiefektit päälle/pois (QCheckBox) (oletusarvo musiikit pois päältä).

Jokaiselle ammustyypille on oma ominaisääniefektinsä: tulipallolla (magic cast -ääniefekti, ohjuksella räjähdystyyppinen ääni ja laserilla laser-blaster ääni)

AsetusDialogissa valinnat tehtyään pelaajan täytyy erikseen painaa Save-nappia, jotta halutut asetukset tallentuisivat. Ellei Save-nappia erikseen paineta, niin tarkoituksenmukaisesti käytetään oletusarvoisia asetuksia. Asetusten tallennuksen jälkeen pelaaja voi palata Back-nappia painamalla takaisin MainMenuDialogiin.

Lisäominaisuus on implementoitu lähinnä settingsDialog -luokassa, mutta asetusten arvoja käytetään hyväksi myös Player- ja basicProjectile -luokissa.

Pelaajan statistiikkaesitys statisticsDialog:
statisticsDialogiin navigoidaan GameOverDialogin ”Statistics” -napista. Tämä QDialogi tarjoaa pelaajalle mahdollisuuden tarkastella pelinaikaista statistiikkaansa eli tilastotietoa: pelaajalle näytetään hänen pelissä keräämänsä raakapisteet eli ei-skaalatut pisteet sekä mm. kerättyjen timanttien, tuhottujen Nyssejen ja matkustajien lukumäärä. Luokka statisticsDialog käyttää hyväkseen luomaamme iStatisticsin aliluokkaa gameStatisctics, joka siis implementoi rajapinnan iStatistics virtuaalifunktiot.

Dialogin pelaajalle esittämät data haetaan gameStatistics -luokasta, joka huolehtii pelinaikaisesta kirjanpidosta ja tallentaa muutokset julkisiin (public) attribuutteihinsa, joita muut luokat voivat hyödyntää. Lisäominaisuus on implementoitu statisticsDialog -luokassa ja tilastodatan tarjoaa iStatisticsin aliluokka gameStatistics.

Pelaajan peli-/apumanuaali helpDialog:
helpDialogiin navigoidaan kuvassa 1 näkyvästä ”Help”-napista. QDialog, jossa pelaajalle avautuu lisätietoa mm. pelistä, sen tavoitteista ja säännöistä QTextBrowseriin. Käyttäjä voi poistua helpDialogista takaisin MainMenuDialogiin painamalla Dialogin oikeasta yläkulmasta löytyvää punaista raksia (QPushButton). Lisäominaisuus on implementoitu helpDialog -luokassa.

1.4 Ryhmän työnjako

Projektin aikainen yhteistyö sujui molempien projektiparin osapuolten mielestä hyvin ja sujuvasti eikä kumpikaan kokenut vääryyttä esim. henkilökohtaisessa työtaakassa tai vastuunjaossa. Pyrimme projektissa siihen, että jaamme vastuita henkilökohtaisten vahvuuksien ja mielenkiinnon mukaan ja mielestämme onnistuimme tässä hyvin.

Alla olevassa taulukossa 1 on esitetty karkea työnjako ryhmän työnjaosta, mutta esim. committien lukumäärä ei kohdallamme ole paras mittari, sillä käytimme melko paljon mielestämme toimivaa Extreme Programming ohjelmistotyöskentelymetodia eli ns. pariohjelmointia, jossa toinen konkreettisesti koodaa yhdellä tietokoneella, ja toinen valvoo toisen työskentelyä, jolloin bugien ym. virheiden lukumäärä olisi pienempi ja koodin laatu parempi. Loppuen lopuksi siis vaikka molemmilta ryhmän jäseniltä ei välttämättä tullut yhtä paljon committeja GitLabin repositorioon, niin todellisuudessa molempien työpanos projektiin oli noin 50 prosenttia.

Taulukko 1: Ohjelmointi 3-kurssiprojektin karkea työnjako.

Tehtävä	Vetovastaava
settingsDialog	Markus
helpDialog	Miikka
MainMenuDialog	Molemmat
helpDialog	Miikka
GameOverDialog	Markus
Pääikkuna (gameWindow)	Molemmat
basicProjectile	Miikka
bonusItem	Miikka
Player	Molemmat
initGame	Miikka
gameCity	Molemmat
gameStatistics	Miikka

topHighScores	Markus
Yksikkötestit (statisticsTest)	Miikka
Kaupunkiin liittyvä toiminallisuus (kurssin puolen koodin integrointi omaan ohjelmaan)	Molemmat
Sopimussuunnittelu luokille	Molemmat
Course Siden koodin perehtyminen	Molemmat
Peliin sopivien kuvien ja ääniefektien etsiminen netistä	Markus
Projektin aikataulutus	Markus
Dokumentaation teko	Molemmat

1.5 Lähteet

Kaikki projektissa käytetyt kuvat pl. laser-ammuksen ja Nyssen (kuvat 6 ja 9 on itse piirretty) on saatu rojaltivapaasta kuvapankista opengameart.org (https://opengameart.org/). Äänitiedostot on saatu rojaltivapaasta ääniefektipankista freesound.org (https://freesound.org/).

Disclaimer:
Luokkakaaviossa (kuva 12) tehdään rajanveto kurssin puolen tarjoaman ja itse tuottamamme koodin välille.