Tampereen Teknillinen Yliopisto

Ohjelmistotekniikka

Ohjelmistojen suunnittelu TIE-20200

Harjoitustyö: syksy 2015

Git-palautus:

https://gitlab.rd.tut.fi/ohjsuun_2015-2016/tomato3.git

tomato3

Anssi Varjonen, 234063, anssi.varjonen@student.tut.fi

Konsta Boman, 239928, konsta.boman@student.tut.fi

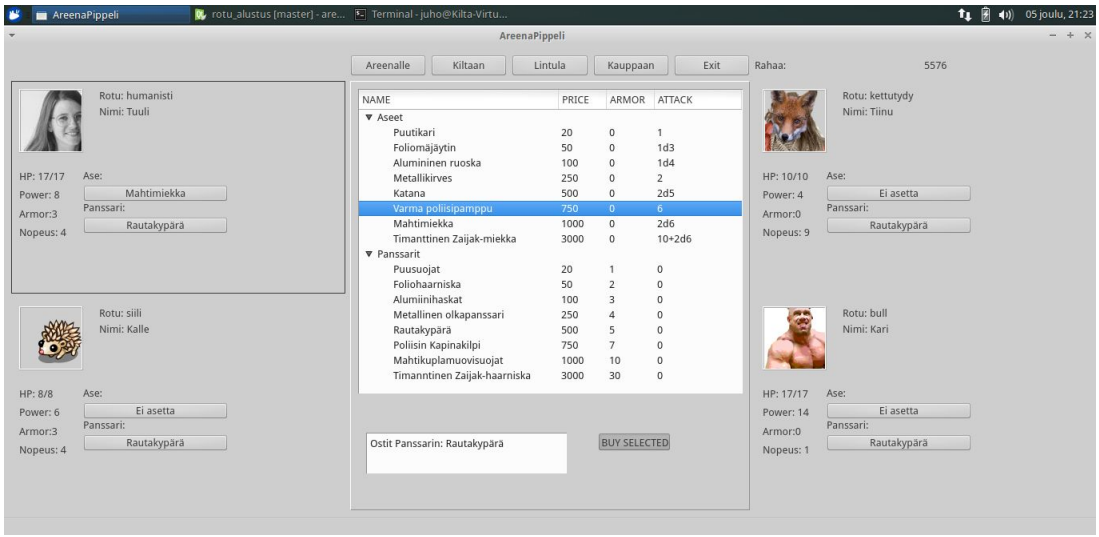
Juho Vähätalo, 240549, juho.vahatalo@student.tut.fi

1. Ohjelman esittely

Harjoitustyömme on Areenapeli, jossa joukko oman joukkueen taistelijoita taistelee vihollisen joukkuetta vastaan vuoropohjaisesti Areena 4:n mukaisesti. Omaan joukkueeseen valitaan pelaajat, joille voidaan ostaa erilaisia voimia ja tavaroita, helpottaakseen taistelun voittamisessa. Pelin ohjelmointi on toteutettu Qt Creatorilla.

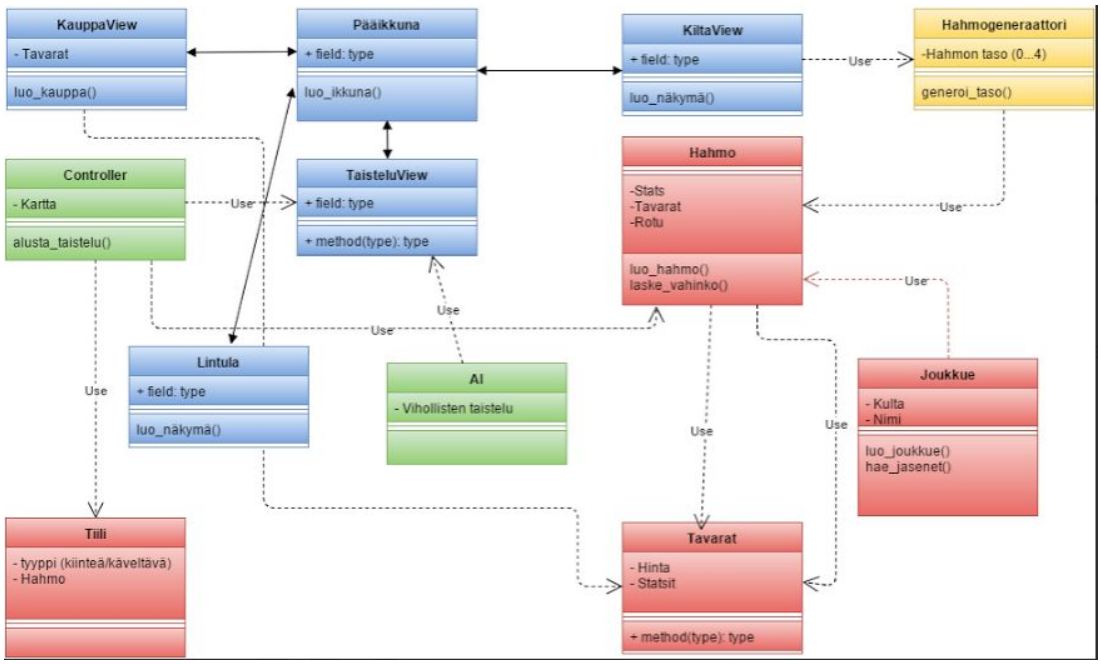
Pelissä pelaaja aloittaa ostamalla joukkueeseensa kiltahuoneelta taistelijoita (vähintään yksi kappale) joilla on eri rotuja ja ominaisuuksia. Tämän jälkeen pelaaja voi ostaa taistelijalleen lisää erilaisia voimia treenaamalla Lintulassa tai erilaisia tavaroita Kaupasta. Pelaaja voi myös tarkastella pelikautensa otteluohjelmaa. Kun pelaaja on mielestään valmis hän siirtyy joukkueensa kanssa Areenalle taistelemaan.

Areenalla pelaaja liikuttaa ja hyökkää tietokoneen ohjaamaa joukkuetta vastaan. Taistelun voittaa se pelaaja, jonka joukkueen taistelija(t) on viimeisenä hengissä. Taistelun jälkeen pelaaja pystyy saamillaan rahoillaan ostamaan parempia tuotteita ja lisää erilaisia voimia.
Pelin lopettaminen tapahtuu painamalla yläkulmassa rastia tai Exit.



Kuva 1: Kuvakaappaus kauppanäkymästä.

Ohjelman toiminnassa perusideana on että Mainwindow luo näkymän, jonka reunoille lisätään hahmo-widgettejä, joille myös voidaan luoda erilaisia toimintoja widgettien avulla. Fightingscenessä luodaan QML-näkymä, jossa taistelu tapahtuu. Näiden lisäksi luodaan myös pop-up ikkunoita. Koko ohjelmaa pyörittää taistelunäkymässä controlleri ja sitten
Rakenne korkealla tasolla, toisin sanoen mikä osa vastaa mistäkin, ja sillaviisiin. Kerro mielenkiintoisimmat osat ohjelmasta, miten ovat tärkeimmät käyttämäsi suunnitteluratkaisut jne.



Kuva 2: ohjelman korkean tason rakenne

Alkuperäiseen suunnitelmaan verrattuna periyttämisestä luovuttiin. Näiden lisäksi päädyttiin toteuttamaan erilaisia pieniä luokkia jotta piirtäminen helpottuisi. Näitä ei ole huomioitu kuvassa 2. Lisäksi toteutettiin AI:lle oma luokka (tästä lisää seuraavissa kappaleissa). Alkuperäiseen suunnitelmaan nähden tahdottiin lisätä yksi uusi näkymä (Lintula).

2. Eri osien rakennetta

Tässä osiossa käydään läpi ohjelman eri luokkia aakkosjärjestyksessä.

2.1. ArenaMember

ArenaMemberissä luodaan pelaajan taisteluhahmo, sekä erilaisia ominaisuuksia sille. Näitä ominaisuuksia ovat rotu, nimi, health, power, armor, liike, nopeus, hinta, armor ja ase. Osa näistä on alustettu int:ksi ja osa QStringeiksi. Ominaisuuksia on jäsennelty struct statsin avulla. Lisäksi taisteluhahmolle on jäsenfunktioita lisaa_maxhp, lisaa_power, lisaa_nopeus, osta_ase, osta_armor. Jäsenfunktioissa käsitellään myös taisteluun liittyviä kuten osuman laskeminen ja niistä ilmoittaminen.

2.2. ArenaTeam

ArenaTeamissa alustetaan joukkue. Joukkueeseen voidaan ostaa ArenaMembereitä neljä kappaletta ja siihen kuuluu ominaisuudet voitto, nimi, rahamäärä ja onko valittu. Lisäksi ArenaTeamissa luodaan pelin AI joukkueet ja niille tasojen mukaiset aseet ja suojat. Rahan liikehdintää seurataan myös tässä jäsenfunktioiden avulla.

2.3. Controller

Controllerissa tarkastellaan oman joukkueen ja vastustajan joukkueen välisen taistelua. Tässä on siis toteutettuna taistelun toteuttamiseen vaadittavia jäsenfunktioita, kuten

taistelun käynnistämisen, voiko hahmoa liikuttaa-tarkastelun, hahmon liikuttaminen, vuoronpäättäminen, onko kaikki hahmot liikkuneet-tarkastelut, kuoleman tarkistaminen, voiton tarkistus. Controller hyödyntää ArenaMembereiden tietoja sekä käyttää ArenaTeamia omien ja vihollisten joukkueiden tietojen vertailuun.

2.4. FightingScene

FightingScenessä käsitellään itse taistelua ja luodaan QML:lää varten tausta jossa hyödynnetään myös Qt:n ominaisuuksia, kuten esim. taistelunäkymän loki ja taistelijan taistelukuvake. Tässä myös alustetaan tarvittavia tietoja. Luokka luo QML widgetin ja jota ylläpitää Map ja Controller.

2.5. KiltaHuone

Kiltahuoneesa luodaan pelaajan joukkueen taistelijoiden ostonäkymä. Ostonäkymä on sijoitettu MainWindowiin widgettinä. Tässä näkymässä pelaaja siis valitsee hahmon olemassa olevasta listasta. Luokka hyödyntää ArenaTeamia muun muassa rahamäärän saamiseksi sekä ArenaMemberiä yksittäisten hahmojen ominaisuuksia varten. Luokka myös käyttää sitä varten luotua Kiltamodelia. Kiltahuoneessa myös joukkueen taistelijoiden määrä on rajoitettu neljän taistelijan ostamiseen.

2.6. KiltaModel

KiltaModelissa ohjaillaan Kiltahuone-näkymää ja sen tapahtumia lisäämällä generoituja taistelijoita ja poistamalla ostetun taistelijan rivin luettelosta. KiltaModel palauttaa Kiltahuoneen hakemien toimintojen mukaan erinäisiä tietoja.

2.7. LintulaWindow

Tässä luokassa käsitellään pelaajan taistelijoiden voimien kehittämistä. Lintulassa pelaaja voi valitulle hahmolleen suorittaa seuravat toiminnot: Treenaa kestävyyttä, Treenaa voimaa ja Treenaa nopeutta. Näillä toiminnoilla voidaan lisätä taistelijan ominaisuuksia joita lisäämisessä taistelijan tietoihin hyödynnetään ArenaMemberin ja AreenaTeamin tietoja. Voiman ostamisen jälkeen ominaisuuksien arvot päivittyvät.

2.8. Main

Main luokassa luodaan käynnistetään pelin pääikkuna MainWindow

2.9. MainWindow

MainWindow on pelin pääikkuna, josta pelaaja voi valita pelin toimintoja.

- Pelin alustaminen ja sammuttaminen
- Otteluohjelman arpominen
- Kiltahuoneelle, Lintulaan, Kauppaan, Sarjataulukon ja Taisteluun siirtyminen
- Joukkueiden tietojen muokkaaminen (voittojen lisääminen)
- Pelaajan taistelijoiden ikkunoiden kutsuminen

MainWindowissa on alusnäkymä, jonka päälle luodaan muut näkymät (taistelua ja sarjataulukkoa lukuunottamatta).

2.10.Map

Mapissa ohjataan taistelunäkymän kentällä liikkeitä, alustetaan kenttä kontrollille sekä tarkastellaan erilaisia toimintoja. Mapissa toteutetaan seuraavanlaiset jäsenfunktiot:

- startFight (tappelun käynnistämisessä tehtävä kentän osien alustaminen ja taistelun alustaminen)
- liikuJohonkin (Taistelijan siirtäminen pelikentällä)
- endTurn (Controllerille pelialueen ilmoittaminen)
- playerchangedindex (QML vaihtaa käyttäjän toimesta indexin-vaihdon)
- getMindex (Palauttaa hahmon indexin)
- rowCount, data (QAbstractListModelin pakolliset toteutukset)

2.11. MyQFrame

MyQFrame on luotu jotta QFrame pystyy ottamaan clicked() signaalin vastaan.

2.12. Plebarpoja

Plebarpojassa luodaan structi *statsit* ja alustetaan niille vihollisten joukkueiden arvot sekä pelaajan joukkueen arvot. Näitä arvoja luodessa Plebarpoja hyödyntää ennalta annettujen tietojen pohjalta valmiita lukuarvoja, joiden perusteella arvotaan pelaajien ominaisuuksia. Sekä pelaajan että vihollisen taistelijoiden ominaisuuksia arvotaan. Näitä tietoja hyödynnetään luokasta ArenaMember.

2.13. Plebruutu

Plebruutu on MainWindowissa nähtävät neljä ruutunäkymää, joihin sijoitetaan pelaajan taistelijoiden tiedot ja kuva. Plebiruudussa on toteutettuna myös pelaajan tavaroiden myyminen kauppanäkymää varten. Plebruutu hyödyntää tiedoissaan ja tietojen hakemisessa ArenaMemberiä ja ArenaTeamia.

Plebruutua hyödynnetään myös FightiningScenessä aktiivisen jäsenen tiedot.

2.14. SarjataulukkoScene

SarjataulukkoScenessä luodaan pelin otteluohjelma ja sen perusteella sarjataulukko. Otteluohjelma on luotu siten että pelaajan joukkueen lisäksi on olemassa viisi muuta joukkuetta, joita vastaan pelataan. Niiden pohjalta luodaan sarjataulukko. SarjataulukkoScene hyödyntää MainWindowissa luotua otteluohjelmaa sekä ArenaTeam ja ArenaMemberin tietoja.

2.15 Shop

Shop on pelin tavaroiden luokka. Se lukee tiedostosta kahdenlaisia tavaroita, jotka jaotellaan aseisiin ja panssareihin. Tavarat on sijoitettuna *Shobject* nimiseen structiin. Tavaroita (ja niiden tehoja) siirretään pelaajan (sekä vastustajan) taistelijoille. Tavaroita hyödynnetään myös ShopScene näkymässä.

2.16. ShopScene

ShopScene on kaupan näkymä, joka yksi keskiwidgeteistä, joita luodaan MainWindowin päälle. Shopscenessä on QTreeviewillä esillä Shop-luokan tavarat sekä niiden tiedot (Nimi,

hinta, puolustusteho ja hyökkäysteho. Näitä tavaroita voidaan ostaa taistelijoille. Myös tavaroiden myymisessä hyödynnetään

2.17. Tile

Tile on yksittäinen pelikentän osanen, jolle sijoitetaan Mapin ohjeiden mukaan tietoja. Se sisältää tiedot ruudussa olevista tavaroista.

2.18. AI (käyttämätön)

AI on vastustaja liikkeiden arpoja, joka sijoitetaan QML:n tilalle ilmoittamaan liikkumiset. Funktio oli toteutettu rekursiivisesti, mutta ajanpuutteen vuoksi oli pieniä virheitä jonka takia se ei toteutunut. Toteutus kuitenkin löytyy koodissa, mutta sitä ei kutsuta. Käyttöönotto ei vaadi montaa muutosta.

3. Asentelu/ajeluohjeistus

Ohjelman pystyy kääntämään Qt Creatorin 5. versiossa. Ohjelma hyödyntää siis valmiiksi asennettuja Qt Creatorin kirjastoja ja QML:ää.

4. Muut helposti muunneltavissa olevat asiat

4.1 Tavaroiden alustaminen

Tavaroiden alustaminen toteutuu *./areenapeli/assets/tavaroiden_alustus.txt* tekstitiedostossa. Tavara on tiedostossa muodossa *tavaran_nimi;tavaran_hinta;puolustus_teho;hyökkäys_teho;*. Tavaroita on kahta erilaista aseita ja panssareita ja ne erotellaan *puolustus_tehon* mukaan. Jos Tavaralla on *puolustus_teho* nolla niin tavara luokitellaan aseeksi. Aseet lisäävät hyökkäysvoimaa arvotulla todennäköisyydellä lyöntihetkellä. Esim. Timanttinen Zaijak-miekka;3000;0;3d6 voi saada *hyökkäys_tehon* 3 kertaa kuusitahkoisen nopan heiton silmälukujen summan.

4.2 Pelaaja-, in game- ja kenttätextuurikuvien lisääminen

Kaikki kuvat lisätään ./areenapeli/assets/ kansioon muodossa .png. Kuvat ovat suhteellisen pieniä jotta niiden lukeminen ei olisi raskasta. Itse toteutuksessa toteutetaan kuvien croppaaminen oikeaan kokoon. Kuvien nimeämisessä käytettään seuraavaa periaatetta.

Kuva	**Nimeämistyyli**
Kenttä textuuri	malli_texture.png
Hahmon in game kuva	malli_ingame.png
Hahmot ja muut objectit	malli.png

5. Muutoksia suunnitteluvaiheeseen verrattuna

Ohjelman suunnitteluvaiheeseen nähden pysyttiin suhteellisen hyvin alkuperäisessä suunnitelmassa. Muutama ylimääräinen luokka lisättiin, kun huomasimme niiden toteuttamisen helpottavan itse työtä.

Ohjelmaan oli suunniteltuna AI joka ajanpuutteen takia jäi toteuttamatta loppuun. Siitä lisää seuraavassa kohdassa.

6. Analyysi, retrospekti, viisastelut

Harjoitustyö oli suhteellisen onnistunut, paitsi AI jäi keskeneräiseksi. Jatkoajatuksena tulimme siihen tulokseen että olisi jälleen viisasta aloittaa harjoitustöiden tekeminen ajoissa. Myös käytettävyys asiat jäivät haittaamaan, mutta toimivuus on pääasialla.

AI:n tekeminen oli ongelmallista. Ohjelmointitunneissa siihen käytettiin 8h, mutta siltikään sitä ei saatu toteutettua. Lopulta päädyttiin siihen että AI:n tekeminen jätettiin kesken ja sitä ei kutsuta, jotta palautukseen saataisiin toimiva ohjelmakoodi. Todennäköisenä syynä oli jokin pieni virhe jonka löytämiseen ei ollut järkevää käyttää resursseja. Myös hienosäätöä olisi voitu parantaa.

Jos tämä työ oltaisiin viimeistelty loppuun asti niin oltaisiin toteutettu uudelleen käytettäviä luokkia enemmän. Esimerkiksi QFramella oltaisiin tiilin textuuri näytetty ja onko se sellainen objekti johon voi liikkua yms.