Ohjelmistojen suunnittelu TIE-20200

Train Crash - Loppuraportti

9.12.2017

Pauli Jaakkola 218681

Henri Laakso 240062

Inkeri Välkki 246800

Ohjeet ohjelman kääntämiseen

Ohjelma on toteutettu C++:lla ja QML:llä. Helpoiten ohjelma kääntyy QtCreatorilla, avaamalla TrainCrash.pro projektitiedosto. Peli toimii Qt:n versiolla 5.9 ja vaatii Qt Positioning kirjaston, joka ilmeisestipuuttuu ainakin Linux-desktopilta, mutta tulee Qt:n oletusasennuksen mukana.

Muutoksia suunnitteluvaiheesta

Mitaleiden ansaitsemista ei ole toteutettu.

StationsModel ja TracksModel luodaan joka pelin alussa kuten muutkin mallit. Lisättiin kauppaa varten ostettavien junien katalogi. Lisättiin myös pelaajamalli pelaajan valitsemista ja pelaajien (ja mitalien, jos/kun ne toteutetaan) tallentamista ja lataamista varten. Myös nämä uudet mallit periytyvät QAbstractListModelista.

Tilakone luotiinkin QML:ssä, jotta se pääsisi helpommin käsittelemään käyttöliittymää tilasiirtymien yhteydessä. Osa kontrollerien metodeista toteutettiinkin signaaleina ja kytkettiin sitten tilakoneen määrittelyssä.

Huoltojunan liikuttelu saatiin välinäyttöön vain puolivalmiiksi eli huoltojuna teleporttaili raiteilla kulkemisen sijaan. (Nyt se toimii oikein). Huoltojunia voi olla pelissä useita yhtä aikaa alun perin suunnitellun yhden sijaan.

Luokkien attribuutit ja rajapinnat eivät enimmäkseen vastaa alkuperäistä luokkadiagrammia. Suunnitteluvaiheessa emme oikein osanneet ennustaa, mitä

niiden pitäisi olla.

Suunniteltuja muutoksia ja havaittuja ongelmia

Mitaleiden ansaitseminen pitäisi toteuttaa.

StationsModelin ja TracksModelin voisi luoda alkuperäisen suunnitelman mukaisesti jo ohjelman käynnistyessä. Tämä säästäisi hieman resursseja. Vielä tehokkaampaa olisi ladata myös päivän aikataulut vain kerran eikä joka pelin alussa. Sitten pitäisi tietenkin huolehtia siitä, että aikataulut ladataan jos päivä ehtii vaihtua pelien välillä.

Peliin pitäisi ehkä lisätä latausnäyttö, koska hitaammilla internet-yhteyksillä aikataulujen lataus saattaa kestää pitkäänkin ja tuona aikana pelaaja voi joko ihmetellä missä matkustajajunat ovat tai siivoilla roskia pois kaikessa rauhassa.

Pelin sopiva haasteellisuus ja pelattavuus vaativat vielä paljon säätöä. Myös käyttöliittymän ulkonäköä olisi syytä parannella ja brändäystä miettiä. Käyttäjätestausta ei ole tehty (eikä kannattaisikaan tällä pelattavuudella).

Myös automaattiset testit puuttuvat, yksikkötesteistä lähtien.

Refaktoroinnille olisi monin paikoin kysyntää. Esimerkiksi QML-tiedostojen sisältö ja `TracksModel::getShortestPath` ovat kovin monoliittisia ja useissa paikoissa käytetään `std::dynamic_pointer_cast`:ia, mikä on aika huonoa olio-ohjelmointia.

Liitteet:

Alkuperäinen suunnitelma

Ohjelmistojen suunnittelu TIE-20200

Train Crash

Suunnitelma

6.10.2017

Pauli Jaakkola 218681
Henri Laakso 240062
Inkeri Välkki 246800

Toteutusteknologia

Ohjelmisto toteutetaan Qt-sovelluskehyksellä. Käyttöliittymä toteutetaan käyttäen QML-kieltä. Pelin tila ja logiikka taas toteutetaan C++:lla ja kytketään QML-ympäristön käytettäväksi.

Ohjelma käyttäjän näkökulmasta

Liitteenä olevassa kaaviossa (stateDiagram) on esitettynä pelin toiminta karkealla tasolla.

Peli alkaa aloitusnäkymästä, jossa valitaan pelaaja ja aloitetaan peli. Varsinaisessa pelitilassa on kaksi pääasiallista näkymää: kartta, jossa itse pelaaminen junilla tapahtuu, ja kauppa, jossa junia voi ostaa ja korjata. Kummastakin pelitilanäkymästä peli voidaan pysäyttää. Kun peli päättyy, siirrytään tulosnäkymään, josta voi halutessaan siirtyä takaisin aloitusnäkymään aloittaakseen uuden pelin. Ohjelma voidaan sulkea kaikista näkymistä.

Tärkeimmät komponentit ja rajapinnat

Ohjelman luokkajakoa ja luokkien rajapintoja on kuvattu liitekaaviossa classDiagram ja olioiden välisiä suhteita liitekaaviossa objectDiagram.

Mallit ja näkymien päivittyminen

Ohjelmassa käytetään push MVC -mallia, jonka toteuttaminen Qt:lla on suoraviivaista. Mallit toteutetaan C++:lla ja ne ovat joko QObjectin tai säiliömäisten mallien (asemat, matkustajajunat, roskat ynnä muut) tapauksessa QAbstractListModelin aliluokkia. Qt:n property-järjestelmä vie mallien muutokset näkymiin automaattisesti kunhan malliluokkien toteutukset muistavat lähettää asiaankuuluvat signaalit.

Asemien ja ratojen tiedot luetaan ohjelman käynnistyessä StationsModel- ja TracksModel-luokkien ilmentymiin. Tämän jälkeen niitä ei ole enää tarvetta eikä mahdollista muuttaa. Junien ja esteiden määrä ja sijainti sen sijaan muuttuvat, joten ServiceTrainsModel, PassengerTrainsModel- ja ObstaclesModel-luokista täytyy luoda uudet ilmentymät joka peliä varten ja päivittää niitä pelin kuluessa. Matkustajajunien liikkeet perustuvat HTTP-API:sta noudettaviin tietoihin todellisten junien liikkeistä. Riippuen matkustajajunadatan määrästä ja formaatista ladataan joko koko data kerralla sisäiseen muistiin tai haetaan vähissä erin pelin aikana.

Pelin tilan kuten edellisessä kappaleessa mainitut mallit, asiakastyytyväisyyden, pelaajan avataran ja hänen käytettävissään olevat kupongit omistaa suoraan tai välillisesti pelimalli, joka on GameState-luokan ilmentymä.

Kontrollerit

Pelin logiikka löytyy enimmäkseen kontrolleriolioista, joiden vastuulla on muun muassa näkymien vaihtuminen, pelaajan siirtojen toteutuminen sekä matkustajajunien ja esteiden luominen ja liikkeet. Myös malliolioissa on logiikkaa, mutta sen tarkoitus on vain mallin oman tilan muuttaminen johdonmukaisesti kontrollerien sitä pyytäessä.

Useimmat kontrollerioliot liittyvät johonkin näkymään (StartController, MapController, StoreController ja ResultsController). Lisäksi löytyy GameController, joka huolehtii kartta- ja kauppanäkymille yhteisistä toiminnoista sekä screenState, joka on näkymien vaihtumisesta huolehtiva tilakone (luokan QStateMachine ilmentymä). Kontrollerit pyritään pitämään tilattomina ja toteuttamaan ainokaisina (‘Singleton’).

Kontrollerien metodeja kutsutaan joko käyttäjän syötteiden tai ajan kulumisen perusteella. Ajan kulumista kuvaa QTimerin ilmentymä frameTimer, jonka tehtävänä on kutsua säännöllisin väliajoin GameControllerin metodeja, jotta matkustajajunat pysyvät liikkeessä ja esteitä ilmestyy radoille.

Aikataulusuunnitelma

Ennen välinäyttöä

- Liikkuminen pelin eri näyttöjen välillä
- Kartta ratoineen ja asemineen
- Huoltojunan liikuttelu
- Roskien ilmestyminen ja kerääminen

Välinäytön jälkeen

- Matkustajajunat
- Junien törmäykset, vahingoittuminen ja korjaus
- Kauppa
- Mitalit