

Ohjelmistojen suunnittelu TIE-20200

# Train Crash

Suunnitelma

6.10.2017

Pauli Jaakkola	218681
Henri Laakso	240062
Inkeri Vålkk	246800

<b>Toteutusteknologia</b>	<b>3</b>
<b>Ohjelma käyttäjän näkökulmasta</b>	<b>3</b>
<b>Tärkeimmät komponentit ja rajapinnat</b>	<b>3</b>
Mallit ja näkymien päivittyminen	3
Kontrollerit	4
<b>Aikataulusuunnitelma</b>	<b>4</b>
Ennen välinäyttöä	4
Välinäytön jälkeen	4

# Toteutusteknologia

Ohjelmisto toteutetaan Qt-sovelluskehysellä. 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 esitetty pelin toiminta karkealla tasolla.

Peli alkaa aloitusnäkökymästä, 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 pelitilannäkymästä peli voidaan pysäyttää. Kun peli päättyy, siirrytään tulostilannäkymään, josta voi halutessaan siirtyä takaisin aloitusnäkökymään aloittaakseen uuden pelin. Ohjelma voidaan sulkea kaikista näkökymistä.

## 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 säiliömaisten 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, asiakastytyvyyden, pelaajan avatarin 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 kaupanäkymille yhteisistä toiminnoista sekä screenSize, 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 GameControllern 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