Teekkarin Sekoilu Seikkailut (TSS) 🔥 🍕

Teekkarin sekoilu seikkailut on peli, jossa päähenkilö toimittaa ruokatilauksia nälkäisille Teekkareille. Tavoitteena on viedä tilaukset perille mahdollisimman nopeasti, jotta ruoka pysyy lämpimänä ja asiakkaat tyytyväisinä. Pelin voittaa, kun on kerännyt tarpeeksi rahaa maksaakseen opintolainan takaisin.

Sisällysluettelo

Tarkemmat säännöt

- Pelaaja voi kantaa kolmea ruokaa kerrallaan*
- Pelaaja toimittaa aina viimeisimmän tilaamansa ruuan (aka FIFO)
- Pysäkille, jossa ei ole teekkaria ruuan toimittamisesta menettää ruuan
- Kaikki ruuat ovat samanarvoisia
- Ruoka maksaa 10€*
- Ruualla on kolme tilaa: kuuma, lämmin ja kylmä
- Ruuan tilat vaihtuvat 20 sekunnin välein ostohetkestä*
- Kuumasta saa 40€, lämpimästä 30€ ja kylmästä 20€**
- Opintolaina on 1000€*

Muuta:

- Bussien määrä per bussilinja*
- Bussin vauhti on random välillä x ja x+1, missä x=BUS_DEFAULT_SPEED (yksikkö scenen koordinaatistossa per frame)*
- Bussin aloitussuunta bussilinjalla on random
- Bussin odotusaika pysäkillä on random välillä 1-3 sekuntia
- Teekkareiden määrä aloittaessa*
- Teekkareiden maksimi määrä kartalla*
- Teekkareiden spawnaamistahti*

*configuroitavissa initScenen header filestä

**configuroitavissa initScenen header filestä, "FOOD_PRICE_FACTOR_MODIFIER" kaava:

```
price_ * (state_ + FOOD_PRICE_FACTOR_MODIFIER);
```

, missä state = 2,1,0 (Kuuma, lämmin, kylmä) ja price_ ruuan ostohinta.

Lisäominaisuudet

- Tasainen ruudunpäivitys. Gameloop käyttää Qt:n advance metodia, joka vastaavasti kutsuu jokaiselle bussille advance metodia. Tällä hetkellä ruudunpäivitys on 60fps.
- Minimaalinen ruudunpäivitys. Qt:n graphicsview viewport update mode on defaultisti minimal viewport update, joka dokumentaation mukaan

  ```
  "QGraphicsView will determine the minimal viewport region that requires a
  redraw,
  minimizing the time spent drawing by avoiding a redraw of areas that have
  not changed."
  ```

- Pelihahmon tasainen liike. Pelihahmo liikuu tasaisesti bussien mukana.
- Pelin tilan seuranta. Graafinen palkki esittää reaaliajassa pelaajan rahatilanteen ja edistymisen kohti tavoitetta.
- Oma lisäominaisuus. Ruuan kuljettamien kioskeista teekkareille ja raha. Nämä ovat selkeästi vaatineet koodaustyötä ruuankuljettamispelin toteuttamiseksi.

Työskentely

Työskentelyssä harjoiteltiin itsenäisesti ***ketterää ohjelmistokehitystä***. Meillä oli 1 viikon mittainen sprintti/**iteraatio** ja joka viikon alussa käytiin läpi mitä oltiin saatu aikaseksi viime viikolla(**retro**) ja mitä tavoitellaan seuraavalta viikolta(**planning**). **Backlogi** oli myös käytössä, johon määriteltiin ominaisuuksia pelin muodostuessa.

Trelloa käytettiin hyväksi ***työskentelyn suunnitteluun ja työskentelyn seuraamiseen***. Tauluina olivat jokaiselle iteraatiolle TODO, DOING, DONE ja RETRO. Tämän lisäksi backlog oli omana taulunansa. Puolivälissä projektia tehtiin myös Possible Features taulu, johon listattiin ideoita mitä saatettiin ottaa backlogiin.

Linkki trelloon, josta löytyy kollaasi jokaisen iteraation DONE ja RETRO tauluista ja tietysti myös tyhjä backlog. Bonuksena myös Possible Features taulu.

Branchejä käytettiin hyväksi versionhallinnassa. Aluksi oli tapana tehdä oma bränchinsä jokaiselle uudelle ominaisuudelle, mutta ihan projektin viimeisellä viikolla ryhdyin käyttämään devT bränchiä, jossa tein ominaisuudet, jotka sitten mergettiin masteriin.

Aluksi tehtiin ***prototyyppi***, jossa vaatimuksina oli pelaajan liikkuminen pysäkiltä toisille bussien avulla. Ekat pari viikkoa tuli käytännössä luettua dokumentaatiota Graphics View Framework käytöstä.

Prototyypin jälkeen saatiin idea tehdä ***ruuankuljetuspeli***, jota tehtiin viikko kerrallaan. Tavoitteena oli saada peruspeli valmiiksi.

Suurimmaksi haasteeksi osoittautui työnjako. Syinä olivat osaaminen ja ajankäyttö. Projektin alussa myös ilmeisesti tuli aliarvioitua projektin vaativuus suhteessa tavoitteeseen (arvosana 4). Omiin oppimistavoitteisiini tämä ei kuitenkaan vaikuttanut, eli arkkitehtuurin harjoitteluun laajemmassa projektissa. Asiasta keskusteltiin projektin aikana, mutta ratkaisua ei keksitty.

Työnjako prosentteina on noin 90% suhde 10%. Tunteja ei mitattu, mutta lopputuloksen perusteella työnjako on realistinen.

Tekninen dokumentointi

Yleiskatsaus

Projektin rakenne on hyvin yksinkertainen. TeekkarinSekoiluSeikkailut kansiosta löytyy varsinainen peli ja Tests kansiosta testit luokille, joille on koettu olevan tarpeellista toteuttaa testejä. Tarpeellisella tarkoitetaan paljon logiiikkaa sisältäviä luokkia, kuten pelaaja ja bussilinja.

Pelin configuraatio löytyy juuresta initScenen header tiedostosta.

Kaikki on tehty käsin, eli nollasta on aloitettu **ilman kurssin tarjoamaa kirjastoa**. Syitä tähän oli muutamakin. Ensinnäkin kurssikirjaston dokumentaatio oli projektia aloittaessa (ihan ensimmäisellä viikolla kun se julkaistiin) todella heikko. Sitä ei oikeastaan ollut laisinkaan, luokkajako lisättiin myöhemmin. Toinen syy oli, että selaillessa kurssin kirjastoa emme oikeen nopeasti katsottuna päässeet perille miten sitä tulisi käyttää, koska ei ollut mitään yleiskatsausta. Luottamus oli siis heikko, joten näimme paremmaksi vaihtoehdoksi toteuttaa asiat itse. Emme myöskään halunneet olla riippuvaisia kurssikirjaston mahdollisista puutteista/toiminnallisista ratkaisuista. Halusimme myös oppia tekemään ratkaisut itse ja mitä asioita pitää ottaa huomioon isompaa projektia tehdessä.

Projektin arkkitehtuuri syntyi tehdessä, eli kokeillen eri jaottelua, pyrkien noudattamaan hyviä tapoja ja paljon refaktorointia välissä. Mottona on ollut "Keep things simple & small".

Teknistä velkaa käsiteltiin siten, että projektin alkuvaiheessa pyrittiin välttämään sitä ja loppuvaiheessa sitä otettiin, koska projektin loputtua ei sitä tarvitse enää maksaa. Tosin jatkokehitysmahdollisuuksia pidettiin mielessä eri ominaisuuksia tehdessä, joten aivan fiasko ei lopputulos pitäisi olla. Ja tottakai arvostelua toteutuksen laadun suhteen pidettiin myös mielessä alusta loppuun.

Tarkempaa arvioita kuinka hyvä projektin arkkitehtuuri on tai **jatkokehitysmahdollisuudet** ovat, on kuitenkin vaikeaa arvioida etukäteen kokemattomuuden ja ohjelmistojen jatkotarpeiden arvaamattomuuden takia.

Kansiorakenne

Pelikansion juuresta löytyy kaksi kansiota, UI ja gameObjects, sekä peliluokka "Game" ja kolme läheisesti peliluokkaan liittyviä tiedostoja: GameObjects, initScene ja MainMenu. GameObjects luokka tallentaa vain tietyt peliobjektit muistiin, joita ei joko lisätä sceneen, tai on muuten kätevä tallentaa myöhempää refereointia varten (Pelaaja). initScene on vain aputiedosto, josta löytyy pelin alustamiseen liittyviä asioita ja header filestä pelin configuraatio vakiot. MainMenu on main menu dialogi, joka käynnistetään pelin alussa.

gameObjects kansio koostuu kahdesta kansiosta, graphical ja handlers, sekä parista tiedostosta. Idea on, että graphical kansio sisältää pelin sceneen lisättävät graafiset objektit, jotka siis näkyvät pelissä ja assets kansion, mistä löytyy kaikki grafiikat svg muodossa ja Qt:n resurssitiedosto. Svg siitä syystä, että peliruudun skaalautuessa kuvat pysyvät tarkkoina, koska vektorigrafiikka. Handlers kansiosta löytyy luokat, jotka

käsittelevät näitä pelissä olevia objekteja. Esimerkiksi TeekkariHandler käsittelee teekkareita, kuten niiden lisäämistä ja poistamista skenestä. Loput pari tiedostoa ovat logiisia luokkia, jotka eivät siis esiinny skenessä visuaalisesti. Wallet on pelaajan lompakko, joka käsittelee rahan ja BusLine on bussilinja, mikä käsittelee pysäkkejä ja tarjoaa busslinjan bussille.

UI kansiosta löytyy myös handlers kansio. Samalla periaatteella "Lautaset" käsittelee yksittäistä "Lautanen" komponenttia. Huolehtii mm. niiden luomisesta, poistamisesta ja liikuttamisesta UI:ssa. ProgressBar on modattu Qt:n progress bar, johon on vain lisätty voittamisehto.

Luokkajako

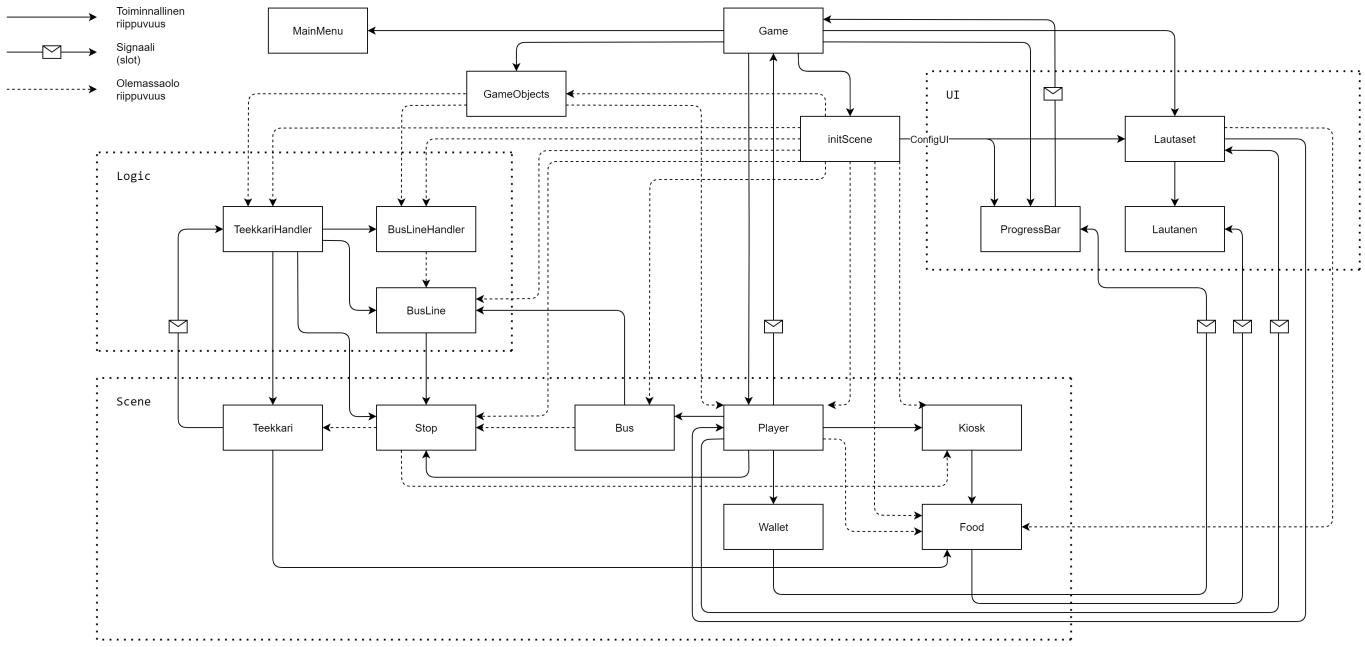
Luokkajako on esitetty kahdessa kuvassa. Toinen edustaa olemassaolo sekä toiminnallisia riippuvuuksia ja toinen pelkästään toiminnallisia riippuvuuksia. Olemassaolo riippuvuuksissa luokan toiminallisuus voi muuttua miten vain, vaikuttamatta luokan toimintaan millään tavalla. Tämä riippuvuus ei ole siis ohjelmoijan näkökulmasta kriittinen kun mietitään muutosten vaikutusta ohjelman käyttäytymiseen. Tämä siis motiivina jaottelun suhteen.

Toiminalliset riippuvuudet = Kutsuu luokan metodeja (1 tai enemmän) ja näin ollen on riippuvainen luokan toiminallisiista riippuvuksista.

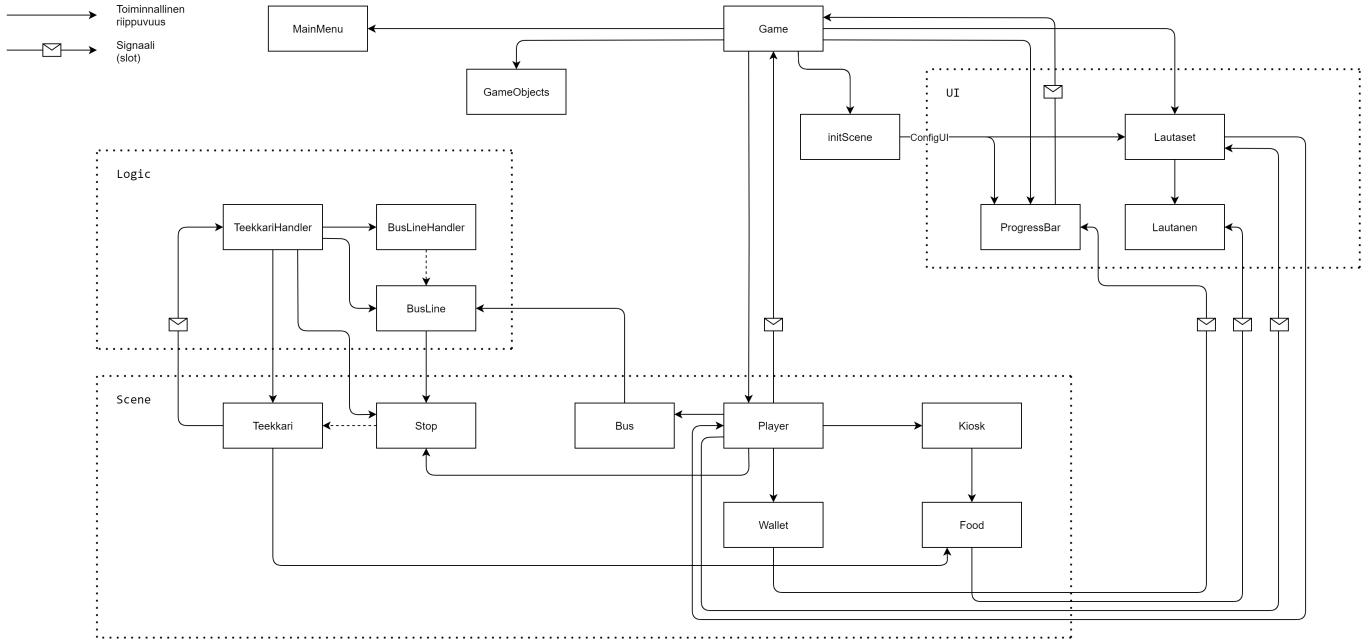
Riippuvuus olemasssaolosta = Ei käytä luokan toiminnallisuutta mitenkään hyväksi (metodikutsuja nolla).

Signaaali = Luokan signaali ja mihin luokkaan se on kytketty (slot)

Olemassaolo sekä toiminnalliset riippuvuudet



Toiminnalliset riippuvuudet



Testaaminen

***Pääasiassa* testit ovat luokkien yksikkötestejä**, mutta esimerkiksi pelaajalta löytyy "Scenario" testejä, jotka vastaavat **integraatiotestausta**.

"Scenario_Skenaario_OdotettavaTulos" tyyliset ***scenaario testit*** tarkoittavat yksinkertaisesti jotain tilannetta, missä testataan toimiiko peli niin kuin pitäisi.

Testit pyrittiin toteuttamaan microsoftin unit testaamisen best practises tyylillä. Osittain tätä rikottiin sijoittamalla integraatiotestejä unit testien sekaan, mutta esimerkiksi nimeämisessä ja toteutuksessa tehtiin pitkälti niinkuin neuvottiin.

"Code coveragea" ei ole mitattu, eikä se ole varmaankaan kovinkaan korkea. Testaaminen ei ole missään nimessä täysin kattavaa ja painotuksena onkin ollut monimutkaisemman logiikan testaaminen virheiden välttämiseksi. Monimutkaisen määrittely on ollut koodarin omatunnon vastuulla ja paikoittain on tullut varmastikin yliarvoitua omia taitojansa.

TDD merkitys on hieman eri aiempien poikkeuksien lisäksi. Yksikkötestit eivät kohdennu luokan sisäiseen toiminallisuuteen, vaan ulkoiseen. Eli testataan ulkoista rajapintaa ja varmistutaan, että luokka **käyttäytyy** oikein. Asia paremmin selitettynä ja syvemmin puheessa Ian Cooper - TDD, Where Did It All Go Wrong.

Kritiikki

Kansiorakenne kritiikki

BusLine on välimaaston tapaus. Busslinja on tavallaan jo (pysäkkien) käsittelijä, mutta se luokiteltiin loogiseksi luokaksi eikä sijoitettu handlers kansioon sekavuuden välttämiseksi. Tasoja oli siis kolme (pysäkki "Stop", busslinja "BusLine", bussilinjat "BusLineHandler"), eli ehkä toisenlainen luokittelu olisi ollut sopivampi jatkoakin ajatellen.

Food ei oo graafinen objekti, vaan looginen. Alkuperäisen ajatuksen ja laiskuuden takia jäänyt väärän paikkaan.

Luokkajako kritiikki

TeekkariHandler selvästi ylittää rajoja ja on riippuvainen osa-alueista, jotka eivät sille kuulu.

Riippuvuuksien karsimiseksi, pitäsi BusLineHandleri riittää tekemään tarvittavat toimenpiteet teekkareille. Esim. findRandomStopWithoutTeekkarissa pysäkkeihin ja busslinjoihin liittyvät asiat pitäsi siirtää kokonaan BusLineHandlerin vastuulle, samoin destroyTeekkarissa ja spawnTeekkarissa. Eli noudattaa ns. "Tell Don't Ask" periaatetta.

Näin ollen riippuvuuksia olisi enää luonnolliset Teekkari ja BusLineHandler. Abstraktio taso olisi myös samalla tasolla ja operoisi järkeväämin myös jatkokehityksen kannalta.