Ohjelmointityön itsearviointi

Ville Kuokkanen 9.7.2019

Tiivistelmä ohjelmointityöstä

TCP-Shakkipeli on Javalla ja Javafx 12:ta toteutettu shakkipeli. Peli on pelattavissa verkon kautta kahdestaan, joista toinen pelaajista toimii palvelimena ja toinen käyttäjänä. Pelissä ei ole mahdollisuutta pelata yksin tietokonetta vastaan. Shakki-tilannetta ja ohilyöntiä lukuunottamatta peli noudattaa shakin sääntöjä: jokaisen pelin alussa pelaajille arvotaan puoli, jonka jälkeen valkoinen pelaaja aloittaa.

Tausta ja tavoitteet

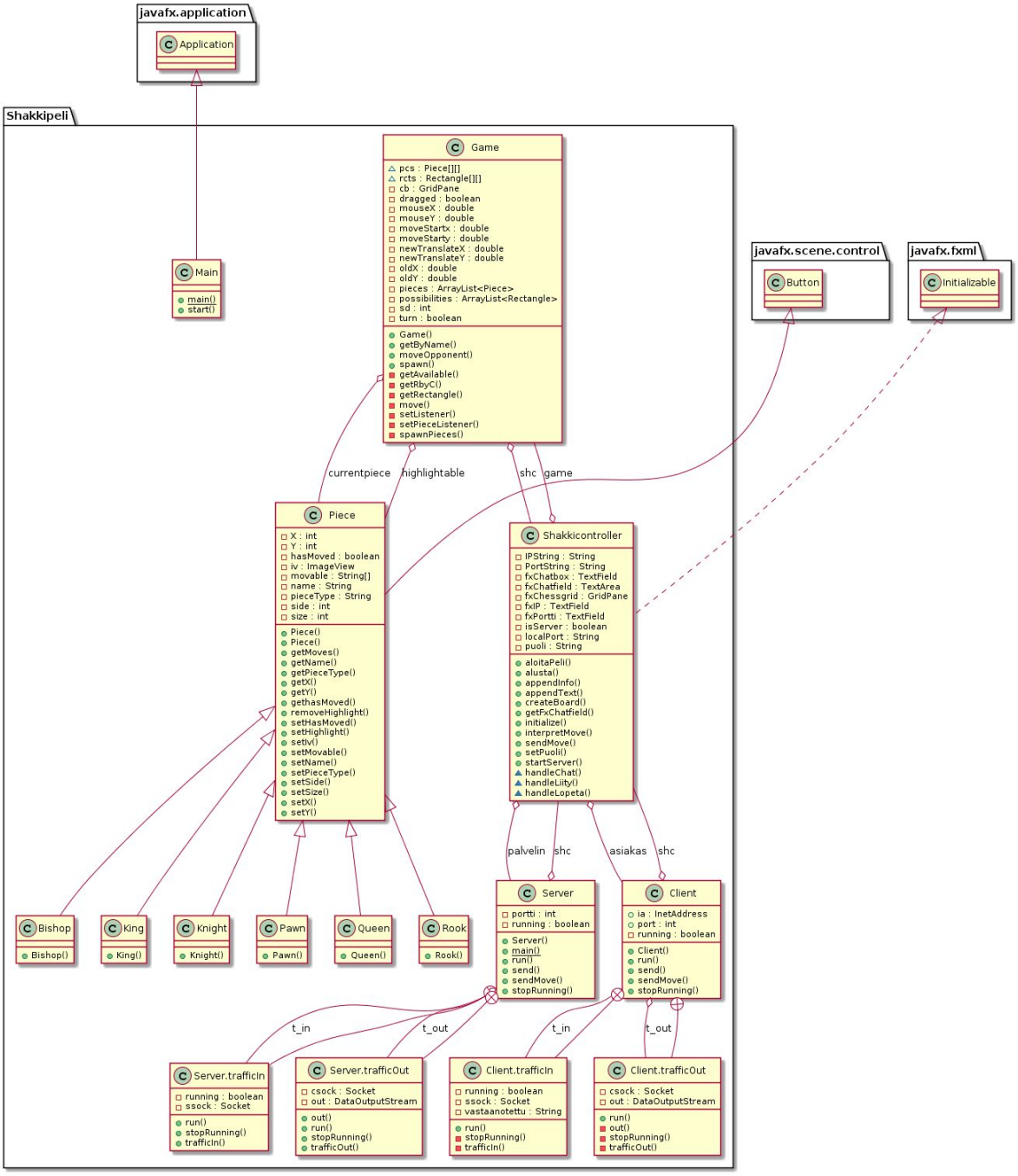
Tarkoituksena oli tehdä shakkipeli jollain ohjelmointikielellä, josta pohdiskelun jälkeen päädyin javaan, koska java ja javafx ovat jo ennestään tuttuja. Shakkipeli on mielestäni hyvä ohjelmointityön aihe, koska siinä tarvitaan osaamista sekä käyttöliittymästä että taustalla toimivista shakin säännöistä ja kahden pelaajan välisestä tcp-kommunikaatiosta. Tavoitteena oli alunperin tehdä peli vähemmillä säännöillä, mutta lopulta pelistä tulikin melko lailla oikean shakin kaltainen.

Käytännön toteutus

Lopullisen pelin rakenne on seuraavanlainen:
Main.java:ssa ladataan mainview.fxml, joka taas avaa oman kontrollerinsa Shakkicontroller.java:n. Lähes kaikki käyttöliittymän tapahtumat, kuten toisen peliin liittyminen (IP-osoitteen ja portin syöttäminen) ja toiselle viestin kirjoittaminen tapahtuu tässä. Poikkeuksena tästä on shakkinappuloiden liikuttelu, jonka käsittely tapahtuu Game.java:ssa.

Peliä avatessa käyttäjältä kysytään portti, jota kuunnellaan. Tähän porttiin avataan palvelin, joka käynnistetään heti peliä avatessa. Palvelinta toiminta käsitellään Server.java:ssa. Uusi palvelin käynnistetään Shakkicontroller.java:sta käsin portilla ja shakkicontrollerilla. Liityttäessä toisen peliin IP:n ja portin syöttämisen jälkeen liittyvästä osapuolesta tulee asiakas (Client), jonka toimintaa käsitellään Client.java:ssa. Asiakkaan ja palvelimen välinen viestintä ja liikkeiden välittäminen tapahtuu UTF-8 merkistökoodauksella. Tehty siirto on toiselle lähetettäessä muotoa:”tunniste, x-koordinaatti (int), y-koordinaatti (int), nappula (Piece)”. Server ja Client jakavat vielä edelleen verkko-kommunikaation kahteen uuteen säikeeseen: trafficIn ja trafficOut. Säikeistys on tehty siksi, jotta viestien ja peliliikkeiden lähettäminen olisi mahdollisimman vaivatonta, eikä pelissä tulisi ongelmia “timeout”:ien ym. kanssa, jos säikeistystä ei olisi käytetty.

Yhteyden muodostamisen jälkeen luodaan uusi peli Game.java:n mukaisesti gridpane:lla, puolella ja shakkicontroller:lla. Game.java:ssa luodaan nappulat pelikentälle. Kaikki nappulat perivät Piece.java:n ja jokaisella nappulatyypillä esim. tornilla on oma puolesta riippuva kuva ja tieto mahdollisista liikuttavista suunnista.
Ohjelman tietorakenteet:
Shakkilaudan ruudut: int[][]
Shakkilaudan nappulat: int[][]
Hiiren koordinaatit pelilaudalla: double X, double Y
Valittu- ja korostettu nappula: Piece
Nappulan mahdolliset liikesuunnat: String[]
Viestien lähettäminen ja vastaanottaminen: DataOutputStream ja BufferedReader
Ruudut, joihin siirto voidaan tehdä: ArrayList<Rectangle>



IntelliJ:n sketch it! pluginilla tehty luokkakaavio.

Oman työn arviointi

Valmiista ohjelmasta tuli kaikin puolin alkuperäiset tavoitteet ylittävä. Esim. peliin toteutin tornituksen sekä liikkeiden validoinnin, josta tuli melko elegantti sillä pelissä näkyy mahdolliset siirrettävät ruudut vihreällä reunuksella, eikä näin ollen tieto laillisista siirroista ole pelkästään pelaajalla. Viikkojen 4 ja 5 kohdalla ajattelin lisäksi tekeväni peliin “shakin”, mutta tämä osoittautui liian aikaa vieväksi, sillä ohjelman sisäisiin rakenteisiin olisi pitänyt tehdä merkittäviä muutoksia.

Ylivoimaisesti vaikeinta peliä tehdessä oli saada nappulat liikkumaan kentällä siten, että nappulaa voidaan vetää tavalliseen tapaan kentällä painamalla hiiren näppäin pohjaan ja liikuttamalla nappulaa ruudusta toiseen. Vaikeata, mutta jokseenkin odotettua oli käsitellä pelin eri rajatapauksia, kuten milloin ratsu saa siirtyä mihinkin ruutuun ja että nappulat pysyvät kentän sisällä. Jostain syystä ongelmia oli myös merkistökoodauksien kanssa, jotka on kuitenkin suurimmilta osin korjattu (peliä avatessa porttia kysyttäessä ä-kirjain saattaa näkyä väärin ja chatissa jotkin erikoisemmat unicode-merkit eivät välttämättä toimi oikein).

Koko ohjelmaa en ole saanut kaadettua, mutta jos toinen pelaaja lopettaa pelin yllättäen tai yhteydessä tulee jokin muu virhe, saattaa tapahtua niin, että peliin tai pelistä ei voi liittyä. Tämä yleisimmin ratkeaa jos painaa “Lopeta peli” ja sen jälkeen liittyy uudestaan. JavaFX:n käyttö tässä pelissä ei ollut täysin ongelmatonta. Esim. nappulan vetämiseen scenessä ei ollut suoraviivaista toteutusta tai valmiita metodeja joilla olisi päässyt samaan lopputulokseen, kuin sillä miten siirtelyn lopulta tein. Siirtämiselle on nappuloissa hiiren painalluksille ja vetämisille ym. omat kuuntelijat, joissa translaatio käsitellään scenen x ja y koordinaattien perusteella. Javan kuuluisi myös olla alustariippumaton, mutta toisin kuin windowsilla, linuxilla testatessa ohjelma ei aina toimi oikein. Mac -ympäristössä peliä ei ole testattu. Viimeksi Javaa ja JavaFX:ää käyttäessäni javafx tuli java-jakelun mukana. Nykyisin näin ei ole, vaan JavaFX täytyy lisätä erikseen, joten ohjelman kääntäminen ja käyttäminen ei ole läheskään yhtä suoraviivaista.

Aikataulu piti alussa todella hyvin, olin neljännen viikon kohdalla etuajassa. Yllättäen tulleet kesätyöt pitkittivät ohjelman tekoa parilla viikolla. Nappuloiden siirtojen toteuttaminen oli ylivoimaisesti aikaa vievin, mutta toisaalta todella keskeinen osa peliä. Nappuloiden siirtämisen toteuttamiseen olisi pitänyt varata paljon enemmän aikaa, mutta toisaalta, koska en ollut aikaisemmin tehnyt JavaFX:ssä vastaavanlaista, olisi tämä ollut mahdotonta tietää.

Pelin ohjelmointi prosessina vastasi juurikin sitä, jota mm. kursseilla ohjelmointi 2 ja oliosuuntautunut suunnittelu -kursseilla on käyty läpi. Ohjelman rakenne tuli melko nopeasti selville ja täydentyi itse ohjelmaa tehdessä, nyt toisella kerralla ollessa tekemisissä JavaFX:n kanssa ohj. 2 kurssin jälkeen paljon nopeammin ja selkeämmin. Pidin myös mielessä

ohjelmaa tehdessä, sen että eri oliolla on omat vastuut ja tiedot, eikä tietyn alueen olioilla ole pääsyvaltaa muualle. Esim. kaikki käyttöliittymän kanssa tekemisissä olevat tapahtumat viedään aina Shakkicontrollerin kautta.

Työssä käytetyt lähteet

JavaFX:n kääntö- ja ajamisohjeet:
https://openjfx.io/openjfx-docs/#install-javafx

JavaFX dokumentaatio:
https://openjfx.io/javadoc/11/

Useisiin Javan ja JavaFX:n ongelmakohtiin, jos suoraa vastausta oli hankala löytää dokumentaatiosta:
https://stackoverflow.com/

Ohjeita säikeistykseen Javassa:
https://www.tutorialspoint.com/java/java_multithreading

Pelin eri elementtien värit:
https://material.io/design/color/#tools-for-picking-colors

Shakkinappulat:
https://commons.wikimedia.org/wiki/Category:SVG_chess_pieces

Siirtojen mahdollisuuksien miettimiseen:
Fyysinen shakkilauta

Ohjeet mahdolliselle ylläpitäjälle ja jatkokehitykseen

Shakkipeli ei varsinaisesti tarvitse suurta määrää ylläpitoa, koska peli ei käytä mitään ulkoisia rajapintoja. Javan ja JavaFX:n uusien päivitysten myötä ohjelma saattaa tietenkin hajota, joten uusien versioiden testaus ja mahdollisten muutosten tekeminen peliin voi mahdollisesti tulla olemaan tarpeellista. Myös merkistökoodauksen muuttuessa ohjelma saattaa toimia tarkoitettua poikkeavalla tavalla, mutta tätä on ohjelmakoodista melko helppo muuttaa esim.

muuttamalla lähetettävät viestit suoraan biteiksi ja tekemällä merkistökoodauksen tulkinnan vastaanottajan päässä.

Sovelluksen jatkokehitystä voi jatkaa useampaan suuntaan. Peliin voisi tehdä “shakin”, jota varten jokaisessa siirrossa tulisi tarkistaa kuninkaan turvallisuus ja itse shakki-tilanteessa selvittää kaikista omista nappuloista, voidaanko jotain nappulaa siirtää siten, että voidaan välttää pelin loppuminen. Shakin tekeminen mitä luultavimmin toteutustavasta riippuen vaatii jokseenkin perinpohjaisia muutoksia nappuloiden käyttäytymiseen ja siirtojen validointiin. Peliin voisi myös jatkokehittää tietokonevastustajan. Tämän voisi toteuttaa esim. siten, että käyttää joko jotain valmista shakin tekoälyä, tai tekee vastaavan itse. Aloittaessa peli tietokonetta vastaan ainut asia, jota pitäisi muuttaa toista pelaajaa vastaan pelatessa on se, että shakkilaudan nappuloiden tilannetta ei käännetä puolta kierrosta, vaan omat ja tietokoneen tekemät siirrot asettuvat suoraan laudalle samoin päin.

Yhteenveto

TCP-Shakkipeli on Javalla ja JavaFX:llä toteutettu kahdestaan verkon kautta pelattava peli. Alkuperäisenä tarkoituksena oli kehittää pelistä melko yksinkertainen versio shakista, mutta lopputuloksesta tuli hieman alkuperäistä suunnitelmaa parempi. Peli koostuu useista eri luokista, joilla on ohjelmassa omat selkeät vastuualueet. Pelissä on yksi päänäkymä, josta käsin liitytään toisen peliin, pelataan peliä ja lähetetään viestejä toiselle pelaajalle. Sovelluksen kehitys tapahtui aluksi etuajassa, jonka jälkeen muista syistä kehitys pitkittyi. Haastavinta pelissä oli saada nappulat siirtymään kentällä vetämällä niitä. Peliä voi jatkokehittää esim. tekemällä siihen tietokonevastustajan ja shakki -tilanteen käsittelyn.