1. Henkilötiedot

Sisustussuunnittelu

Nimi: Hilla Paasio

Opiskelijanumero: 101493325

Koulutusohjelma: Tietotekniikka

Vuosikurssi: 2023

Päiväys: 18.4.2024

2. Yleiskuvaus

Sovellus on apuväline sisustussuunnitteluun ja objektien sovitteluun huoneissa. Huonekaluja ja pohjapiirroksia kuvataan 2D:nä ylhäältä käsin. Sovelluksessa voi lisätä valmiiseen pohjapiirrokseen huonekaluja tai luoda oman pohjapiirroksen. Objektien väriä ja kokoa ja suuntaa voi muuttaa. Pohjapiirroksia voi tallentaa tiedostoon. Sovellus auttaa huonekalujen sijoittamisessa tarkistaen, että tietyt asiat eivät mene päällekkäin. Sovelluksessa oli tarkoitus olla ominaisuus, jolla suunnitelman voi tallentaa uudelleen muokattavaan tiedostoon, mutta vain kuvatiedostoja käytettiin.

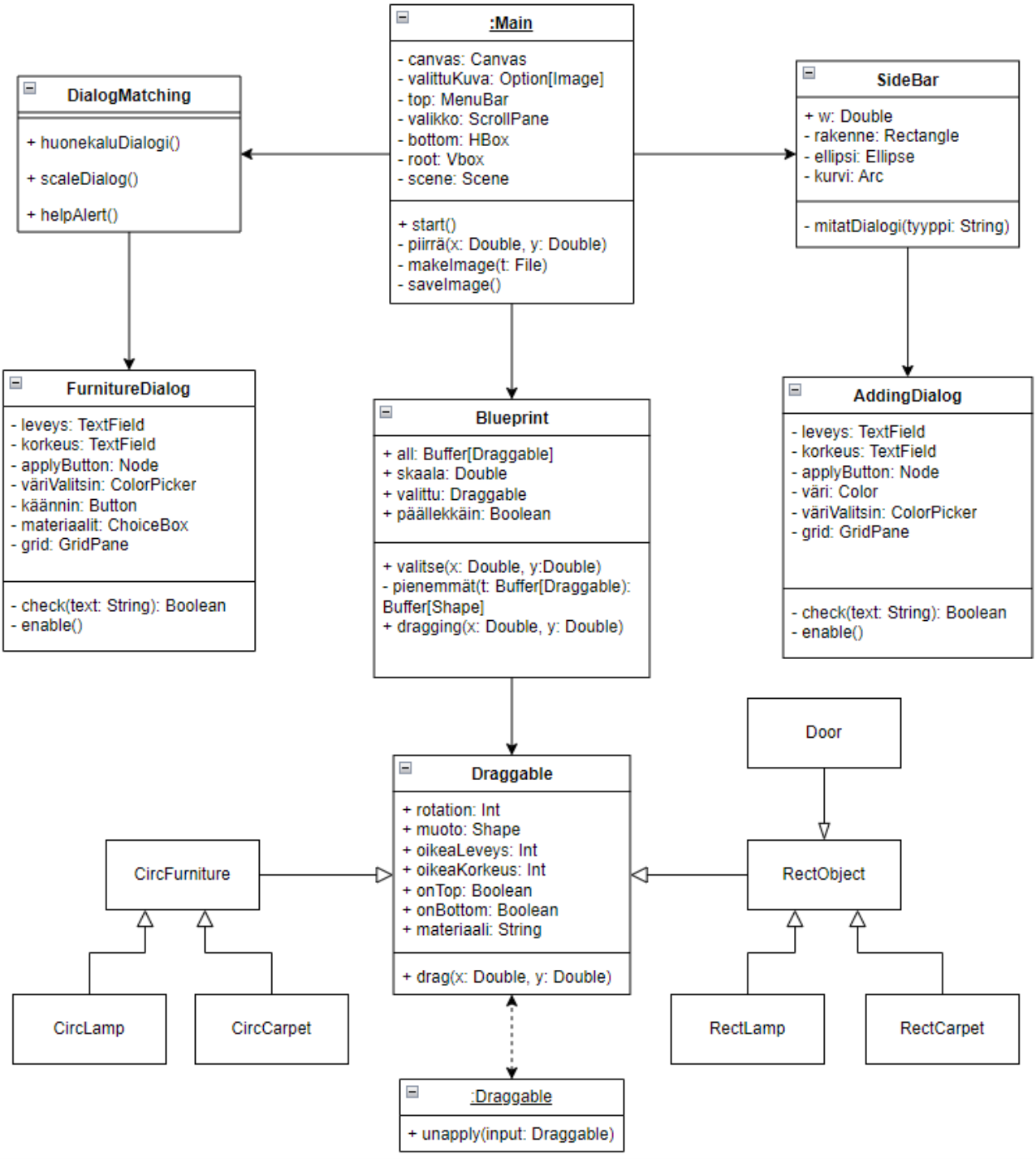
Työ on toteutettu vaativalla vaikeusasteella.

3. Käyttöohje

Ohjelma käynnistetään ajamalla se IDE:ssä. Ohjelmalla päänäkymään voi lisätä objekteja sivupalkista klikkaamalla haluttua objektia. Tällöin ilmestyy ponnahdusikkuna, jossa kysytään objektin mittoja ja värejä. Sovellus ohjaa käyttäjää syöttämään oikeanlaisen syötteen. Kun objekti on lisätty, sitä voi raahata painamalla hiiren vasemman näppäimen pohjaan sen kohdalla. Kun objektia klikkaa oikealla näppäimellä, tulee esiin toinen ponnahdusikkuna, jossa kerrotaan objektin mitat, väri ja mahdollinen materiaali. Objektia voi kääntää tästä menusta, tai sen voi poistaa. Ohjelman ylävalikossa on help-painike, josta saa tarvittaessa apua. Yläpalkissa on myös skaalaus vaihtoehto, jota painamalla tulee ponnahdusikkuna, jossa kysytään prosentteina mihin skaalaan suunnitelman objektit muokataan. Ensimmäisenä ylämenussa on tiedostonkäsittely. Menusta voi valita kuvan avaamis –vaihtoehdon, jolloin tietokoneen resurssienhallinta tulee näkyviin. Käyttäjä voi

valita haluamansa kuvatiedoston. Kuva piirtyy päänäkymän keskelle ja sen päälle voidaan lisätä objekteja. Valmis sisustussuunnitelma voidaan tallentaa tiedostoihin samasta menusta. Menussa on myös valikko näkymän tyhjentämiselle.

4. Ohjelman rakenne



Ohjelma on eroteltu pääpiirteittäin kahteen osaan, GUI:ta käsittelevään app-packageen ja huonekalujen logiikkaa käsittelevään logic-packageen. App-packagessa on pääluokka main, josta start() metodin kautta ohjelma pyöritetään. Mainissa luodaan siis muiden luokkien olioita käytettäväksi. Mainissa suoritetaan ohjelman kokoaminen kasaan. Siellä suoritetaan myös kuvatiedostojen käsittely metodeilla makeImage ja saveImage ja ohjelman päänäkymän graafinen päivittäminen metodilla piirrä. Mainista on siirretty muita GUI:in liittyviä ominaisuuksia omiin luokkiinsa, jotta eri tehtäviä voidaan jaotella eri tiedostoihin.

Luokka SideBar hoitaa ohjelman huonekalumenun. Luokassa on jokaista huonekalutyyppiä varten oma shape olionsa, jota klikkaamalla menussa aktivoi metodin mitatDialogi, joka hoitaa esille ilmestyneen ponnahdusikkunan syötteen käsittelyn. Inputin perusteella pohjapiirrokseen lisätään oikeanlaisia objekteja.

Itse ponnahdusikkuna luodaan luokassa AddingDialog. AddingDialog itse on Scalafx luokan Dialog ilmentymä. Dialogissa on kaksi tekstikenttää ja värivaihtoehto kenttä. Luokassa tapahtumankuuntelijat ja metodit check ja enable huolehtivat, että käyttäjän syöte on oikeanlainen.

Luokka DialogMatching hoitaa ohjelmassa skaalausdialogin ja help-dialogin sekä käsittelee FurnitureDialog luokan syötettä. Luokassa metodi scaleDialog siis luo dialogin ja käsittelee syötteen vaihtaen suunnitelman skaalaa. Metodi helpAlert luo alertin, jossa on käyttäjää opastavaa tekstiä. Metodi huonekaluDialog käsittelee FurnitureDialogin syötettä suorittaen tarvittavat toimenpiteet, eli huonekalun koon, värin tai rotaation muuttamisen, tai huonekalun poistamisen.

Itse dialogi luodaan taas luokassa FurnitureDialog, joka on samankaltainen kuin AddingDialog. Dialogissa on siis taas kaksi tekstikenttää, joiden syötettä aktiivisesti tutkitaan, ja värivaihtoehto kenttä. Lisäksi dialogissa on nappula huonekalun pyörittämiselle sekä poistamiselle, ja mahdollinen materiaalivalikko.

Luokassa Blueprint säilytetään kaikkia käyttäjän lisäämiä huonekaluja puskurissa. Lisäksi luokassa on muita raahaamiseen liittyviä metodeita. Metodi valitse valitsee hiiren

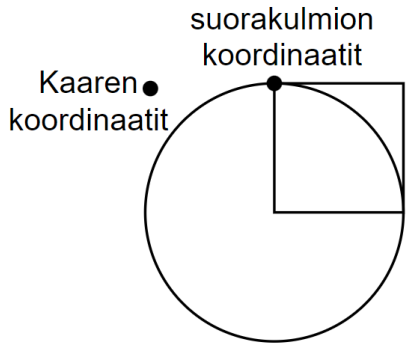
raahauksen alkuvaiheessa raahattavan objektin, ja huolehtii siis siitä, että vain yksi objekti on valittu. Sen lisäksi ohjelma ottaa talteen hiiren alkukoordinaatit raahauksen alettua metodilla muutaAlku. Blueprint luokka hoitaa myös itse raahauksen tapahtumisen metodilla dragging. Metodi tarkistaa, ettei asiat mene päällekkäin väärillä tavoilla. Metodista kutsutaan Draggable luokan metodia drag.

Draggable luokka kuvastaa kaikkia objekteja ja on kaikkien objektien yliluokka. Luokka oli ennen piirreluokka, mutta osoittautui hyödylliseksi muuttaa se oikeaksi luokaksi, sillä silloin siitä pystyi luomaan sopivan placeholder olion dialogien syötteen käsittelyyn. Jokaisella Draggable luokan oliolla ja alioliolla on muoto ja metodi drag, jolla huonekalun sijaintia oikeasti muutetaan. Lisäksi skaalauksen takia olion oikeat mitat voi olla erit, kuin piirretyn kuvion mitat, joten näistä pidetään kirjaa. Olioilla on sen lisäksi arvo siitä, onko se muiden huonekalujen päällä tai alla. Tätä tietoa käytetään hyväksi esimerkiksi päällekkäisyyksien tutkimisessa ja kuvioiden piirtämisessä.

5. Algoritmit

Sisustussuunnittelu aiheena on erittäin käyttöliittymä pohjainen, joten kompleksisia algoritmeja ei tarvita. Alun perin suunnitelmana oli toteuttaa algoritmi sille, että huonekalut eivät mene päällekkäin. Jos ohjelma olisi toteutettu tällaisen oman algoritmin pohjalta, lopputulos saattaisi olla parempi. Kuitenkin scalafx:stä löytyi suoraan työkalut päällekkäisyyksien tutkimiseen: Shape luokan intersects metodi. Ainut ongelma oli se, että ellipseilläkin muodon hitbox on suorakulmion muotoinen.

Tämän takia oven muodossa päätettiin valmiiksi käyttää suorakulmion mallista muotoa ja toteuttaa muodon ulkonäkö vain piirto metodissa. Tämän takia ovi oli helpompi lisätä, koska koodia piti muokata vähemmän, sillä suorakulmio oli jo olemassa. Oven piirtämiseen tarvittiin kuitenkin laskentaa, sillä scalafx:n Arc luo ympyrän kaaren, mutta sen koordinaatit ovat samankaltaiset kuin koko ympyrällä.



Jokaiselle oven rotaatiolle laskettiin siis omat koordinaatit. Riippuen oven rotaatiosta, piirretään kaari, jonka koordinaateista mahdollisesti poistetaan ympyrän säde, eli suorakulmion leveys/korkeus. Kaaren leveyden tulee taas olla yhteensä kahden suorakulmion leveyden suuruinen. Kaaren aloitus ja lopetuskulma riippuvat rotaatiosta.

Saman tyylistä matematiikkaa käytetään ympyrän sijoittamisessa ruudulle, sillä ympyrällä koordinaatit on ilmoitettu keskipisteen koordinaatteina.

Matematiikkaa käytetään myös, kun ikkuna luodaan ja sen kokoa muutetaan. Lähes kaikki säiliöiden koot ovat sidoksissa koko ikkunan kokoon.

6. Tietorakenteet

Huonekalujen/rakenteiden säilönä käytetään muuttuvatilaista puskuria, sillä huonekaluja tulee pystyä lisäämään ja poistamaan. Kun kaikki canvasille piirrettävät objektit ovat samassa puskurissa, ne on helppo käydä läpi for loopilla. Lisäksi puskurissa olevia olioita pystytään muokkaamaan kätevästi, kun valitaan tietty objekti. Alun perin tarkoituksena oli pystyä tallentamaan puskurin sisältö esimerkiksi json tiedostoon, jolloin oli hyvä, että kaikki tieto oli yhdessä säilössä. Tätä toiminnallisuutta ei kuitenkaan toteutettu. Lisäksi scalafx:n luokissa tietoa eri Nodejen lapsista säilytetään usein Arrayssä.

7. Tiedostot ja verkossa oleva tieto

Tiedosto käsittelee tarvittaessa kuvatiedostoja, jotka ovat binaaritiedostoja. Ohjelmaan voi ladata kuvan koneeltaan ja koneelle pystytään tallentamaan pohjapiirrustuksen sisältö. Tiedostoja ei kuitenkaan ole pakko käyttää.

8. Testaus

Ohjelmassa oli aluksi toimiva testitiedosto, johon olisi voinut luoda yksikkötestejä tarvittaessa. Kuitenkin kaikki sisältö oli helposti nähtävissä ja kokeiltavissa käyttöliittymän kautta ja yksikkötestit osoittautuivat melko turhiksi. Ohjelmaa testattiin siis pääosin käyttöliittymän kautta, mutta debuggauksessa käytettiin myös paljon println komentoa. Käyttöliittymän kautta pystyi testaamaan lähes kaikkea toiminnallisuutta. Esimerkiksi eri menujen toimintaa ja niiden käyttämiä metodeja sekä huonekalujen siirtelemistä, kääntelemistä ja muokkaamista. Ohjelman toiminnallisuutta pystyi testaamaan sitä mukaa, mitä se valmistui. Myös mahdollisia virhetilanteita pystyi testaamaan käyttöliittymän kautta. Testattiin esimerkiksi sitä, miten sovellus reagoi, jos vääränlainen tiedosto avataan tai syöte on vääränlainen. Käyttöliittymän kautta testaamisella vältettiin myös se mahdollisuus, että testit olisivat viallisia tai huonoja. Suunnitelmassa yksikkötestauksella oli suurempi painoarvo kuin todellisuudessa. Suunniteltuna oli esimerkiksi testata metodia, jolla tarkistetaan, onko kaksi objektia päällekkäin. Tätä metodia ei kuitenkaan itse luotu, vaan se saatiin suoraan scalafx:stä, joten testausta ei tarvittu.

9. Ohjelman tunnetut puutteet ja viat

Ohjelmassa ehkä keskeisin mahdollinen vika on, että päällekkäisyydet havaitaan aina suorakulmion malliselta alueelta, joten vaikka asiat eivät visuaalisesti menisikään päällekkäin, ohjelman mielestä ne menevät. Vian voisi korjata muuttamalla tapaa, millä päällekkäisyyksiä tutkitaan. Päällekkäisyyksien tutkimiseen voisi esimerkiksi luoda omia metodeita eri muotojen välillä käyttämällä matemaattisia laskelmia. Myös scalafx:stä saattaisi löytyä sopivia työkaluja tämän toteuttamiseen.

Suunnitelmissa oli myös mahdollistaa tekstitiedostoon tallentaminen tavalla, millä kyseinen tiedosto voidaan avata uudestaan, ja kaikki tieto huonekaluista säilyy. Tätä ei toteutettu, sillä oli epäselvää, tarvittiinko sitä vai ei. Tämän puutteen voisi korjata lisäämällä sopiva tiedostomuoto ja tallentamistapa, sekä tiedoston avaamistapa. Toisaalta se voisi olla vaikeaa, sillä huonekalut ovat omia muuttuvatilaisia olioitaan.

Kun huonekalun info ikkunan avaa, niin kun materiaalia muuttaa, se muuttuu objektissa, vaikka ei painaisikaan apply nappia. Kaikki muut ominaisuudet muuttuvat vasta, kun apply nappia on painettu.

Sovelluksen skaalaus toimii siten, että huonekalujen koko yksinkertaisesti muuttuu, mutta niiden sijainti ei muutu, joten huonekalujen sijainti suhteessa toisiinsa muuttuu. Tämän takia skaalaus kannattaa tehdä heti alussa, ennen useamman huonekalun lisäämistä. Puutteen voisi mahdollisesti korjata muuttamalla myös objektien sijaintia suhteellisesti.

Kun ohjelman skaalaa muuttaa, silloin kun huonekaluja on jo asetettu, huonekalut saattavat mennä päällekkäin, ilman että ohjelma tajuaa sitä. Päällekkäisyys havaitaan vain sille huonekalulle, jota on viimeisimpänä liikutettu. Tämän voisi korjata niin, että aina skaalatessa käydään läpi kaikki huonekalut ja katsotaan, onko niillä päällekkäisyyksiä muiden kanssa.

Huonekaluja pystyy siirtämään osittain näkymän ulkopuolelle, mutta kokonaan vain ala- ja oikean reunan ulkopuolelle. Olisi mahdollista muuttaa tätä niin, että huonekaluja pystyy siirtämään kaikkialle tai pelkästään rajojen sisällä. Koska työssä on käytetty placeholder olioita esimerkiksi valittu muuttujassa, drag metodia ei olisi niin simppeliä muuttaa. Jotta esineitä voisi liikuttaa vain rajojen sisällä, Draggable oliolle tulisi välittää ikkunan koko jotenkin. Toisaalta se, että tavaroita pystyy liikuttamaan rajojen ulkopuolella, mahdollisesti lisää käyttäjäystävällisyyttä.

Projektin kriteereissä mainittiin, että huonekaluista näkyisi esimerkiksi kuva info ikkunassa. En kuitenkaan lisännyt mitään kuvaa. Ikkunaan voisi lisätä melko helposti vain tietyn muodon, joka adaptoituu syötettyihin tietoihin, tai oikean esittelykuvan esimerkiksi oikean maailman ovesta. Toisaalta sovellukseni on sellainen, että siinä ei ole erikseen eroteltu esimerkiksi pöytää ja sänkyä, vaan on vain yksi tietty suorakulmion mallinen huonekalu. Ongelman voisi korjata lisäämällä lisää vaihtoehtoja ja esimerkiksi luokkaan Draggable uuden muuttujan kuvalle, joka näytetään infoikkunassa.

Puutteellista on myös se, että huonekalua ei voi raahata valikosta päänäkymään, vaan sitä klikataan ja se ilmestyy vasemmalle ylös. Jos lisäys tehtäisiin raahaamalla, päällekkäisyyttä voisi olla vaikeampi tutkia.

10. 3 parasta ja 3 heikointa kohtaa

Hyvää ohjelmassani on se, että ikkunan kokoa voi muokata, jotta sovellus varmasti mahtuu käyttäjän näytölle. Usein minulle käy niin, että muiden koneilla jotkin ohjelmat toimivat

hyvin, mutta omalla koneellani ikkuna ei mahdu näytölle ja sovellus muuttuu käyttökelvottomaksi. On myös hyvä, että ikkunaa suurentamalla saa enemmän tilaa canvasille.

Toinen hyvä asia ohjelmassani on, että ohjelma ilmoittaa, jos käyttäjä on tehnyt virheen, ja asia ei jää käyttäjälle epäselväksi.

Kolmas hyvä asia on, että huonekaluluokat perivät toisia huonekaluluokkia, joka merkittävästi vähentää toisteisuutta.

Huonoa ohjelmassani on se, että objekteja liikuteltaessa ne teleporttaavat siten, että niiden keskikohta on hiiren kohdalla. Tämä tekee ohjelman käyttämisestä rasittavampaa, kuin jos hiiri pysyisi samassa kohdassa suhteessa objektiin. Tämän ratkaisemiseen voisi todennäköisesti käyttää matikkaa ja muita laskutoimituksia.

Toinen huono asia ohjelmassani on se, että soikioiden ja ovien päällekkäisyys määritetään suorakulmiosta, jonka sisään ne mahtuvat, eikä itse muodosta. Ongelmaan olisi voinut itse luoda metodin, millä päällekkäisyys määritetään. Tähän liittyen huonekaluja voi pyörittää vain 90 asteen kulmissa. Huonekalujen pyörittämiseen olisi voinut käyttää Shape luokan rotate metodia, mutta toteutuksen olisi pitänyt olla hieman erilainen. Silloin päällekkäisyydet olisivat voineet haitata entistä enemmän.

Kolmas huono asia on skaalauksen toimivuus. Kun valmista pohjapiirrosta skaalaa, asioiden suhteelliset sijainnit muuttuvat. Kun skaalaa suurentaa, asiat voivat mennä päällekkäin, ja ohjelma ei ymmärrä sitä.

11. Poikkeamat suunnitelmasta, toteutunut työjärjestys ja aikataulu

Ensimmäisessä sprintissä ehdin oikeastaan vain tutustumaan scalafx:ään ja luomaan projektiin package-rakenteen.

Toisessa sprintissä loin guin ja siihen päänäkymän, sivupalkin ja ylämenun. Loin myös pääpiirteittäisen luokkarakenteen objekteille. Sain piirrettyä päänäkymään suorakulmioita, joita pystyi alkeellisesti siirtelemään.

Kolmannessa sprintissä toteutin melkein kaiken lopun. Loin toimivan sivumenun ja ylämenun. Loin huonekalujen päällekkäisyyksien tutkimisen. Loin dialogeja, joilla

huonekaluja pystyi lisäämään ja käsittelemään. Loin erilaisia huonekalutyyppejä ja muotoja. Tein ikkunan koosta muokattavan. Lisäsin tiedoston käsittelyä eli kuvan avaamisen ja tallentamisen.

Neljännessä sprintissä siistin koodia ja jaoin koodia eri tiedostoihin luokkarakenteen avulla.

Suunnitelmasta poikettiin tiedostomuodon kohdalla niin, että suunnitelmaa ei voi enää sulkemisen jälkeen avata muokattavaan tilaan, vaan pelkästään kuvatiedostona. Yksikkötestausta ei juuri suoritettu, sillä käyttöliittymä oli tehokkaampi tapa testaamiseen.

Noudatin suunnitelmani aikataulua melko hyvin. En kuitenkaan aloittanut ihan alussa heti tekemään projektia. Asiat vähän jäivät loppupäähän, mutta niin olin suunnitellutkin. En ollut suunnitellut käyttäväni niin paljon aikaa eri muotojen käsittelyyn tai esimerkiksi loppuraportin kirjoittamiseen. Opin löytämään tietoa netistä ja muokkaamaan esimerkiksi java koodiesimerkkejä toimivaksi scalaan.

12. Kokonaisarvio lopputuloksesta

Ohjelma on hyvin simppeli, mutta olen mielestäni onnistunut välttämään liikaa toisteisuutta koodissa. Kaikki tarvittavat kriteerit on täytetty. Työssä ei ole oleellisia puutteita, mutta asiat olisi mahdollisesti kannattanut toteuttaa eri tavalla. Olisin etukäteen tutustunut enemmän eri scalafx luokkiin ja todennäköisesti valinnut Canvasin tilalle Panen, sillä Pane ilmeisesti myös tietää, mitä sen päällä on. Tämä olisi voinut muuttaa toteutusta huomattavasti. Luokkarakenne oli mielestäni melko toimiva.

Ohjelmaa voisi tulevaisuudessa parantaa tekemällä paremman päällekkäisyysalgoritmin ja lisäämällä uusia huonekaluja tai tietoja huonekaluista, esimerkiksi kuvan. Muuttaisin objektien kääntämistä siten, että ne voivat kääntyä sulavammin, eikä vain 90 asteen kulmissa. Lisäisin myös suunnittelemani toisen tiedostomuodon, mihin suunnitelman voi tallentaa.

Ohjelman rakenne soveltuu melko hyvin laajennusten tekemiseen. Esimerkiksi huonekaluja on erittäin helppo lisätä lisää. Eri muotojen lisääminen voi osoittautua vaikeammaksi.

Jos aloittaisin projektin alusta, yrittäisin löytää paremman tavan tutkia päällekkäisyyksiä. Tutkisin myös etukäteen, miten olioita voisi tallentaa tekstitiedostoon. Loppujen lopuksi projekti tuli valmiiksi ja siinä ei ole merkittäviä vikoja, joten se on ihan hyvä.

13. Viitteet ja muualta otettu koodi

En suoraan kopioinut koodia netistä, vaan muokkasin koodista sopivaa omaan tarkoitukseeni. En käyttänyt AI työkaluja. Käytin työssäni scalafx ja javafx kirjastoja. Sain inspiraatiota toisilta opiskelijoilta muun muassa siitä, että koodin voi jakaa pienempiin luokkiin eri tiedostoihin. Olen käyttänyt esimerkkinä muun muassa seuraavia lähteitä:

- Javafx dokumentaatio:
  https://javadoc.io/doc/org.scalafx/scalafx_2.13/latest/index.html

  https://docs.oracle.com/javase/8/javafx/api/javafx/scene/control/ColorPicker.html
- Scalafx dokumentaatio:

  https://javadoc.io/static/org.scalafx/scalafx_2.13/14-R19/scalafx

  https://www.scalafx.org/docs/dialogs_and_alerts/
- Stack Overflow:

Pääsivu: https://stackoverflow.com/

Esimerkiksi seuraavat kysymykset:

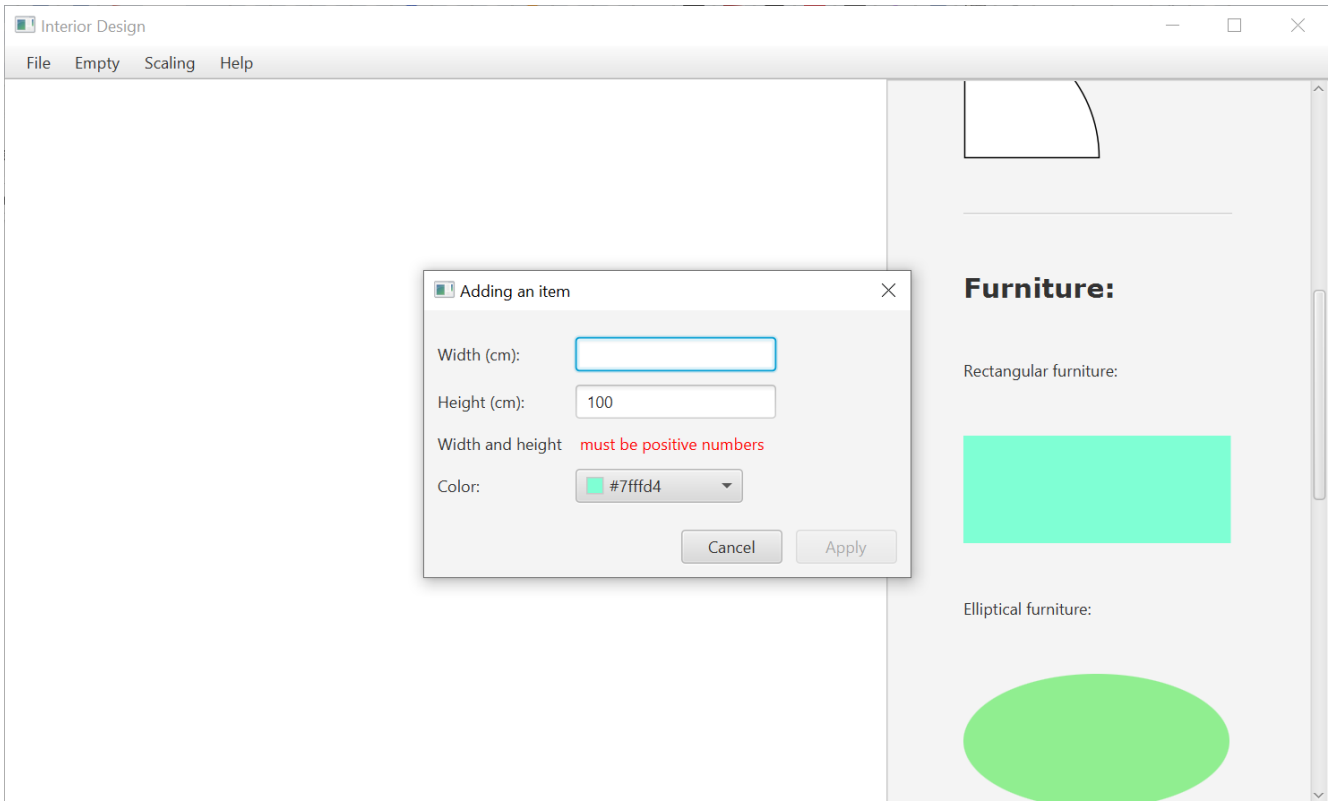
https://stackoverflow.com/questions/38216268/how-to-listen-resize-event- of-stage-in-javafx

https://stackoverflow.com/questions/69321301/javafx-save-canvas-to-png-file

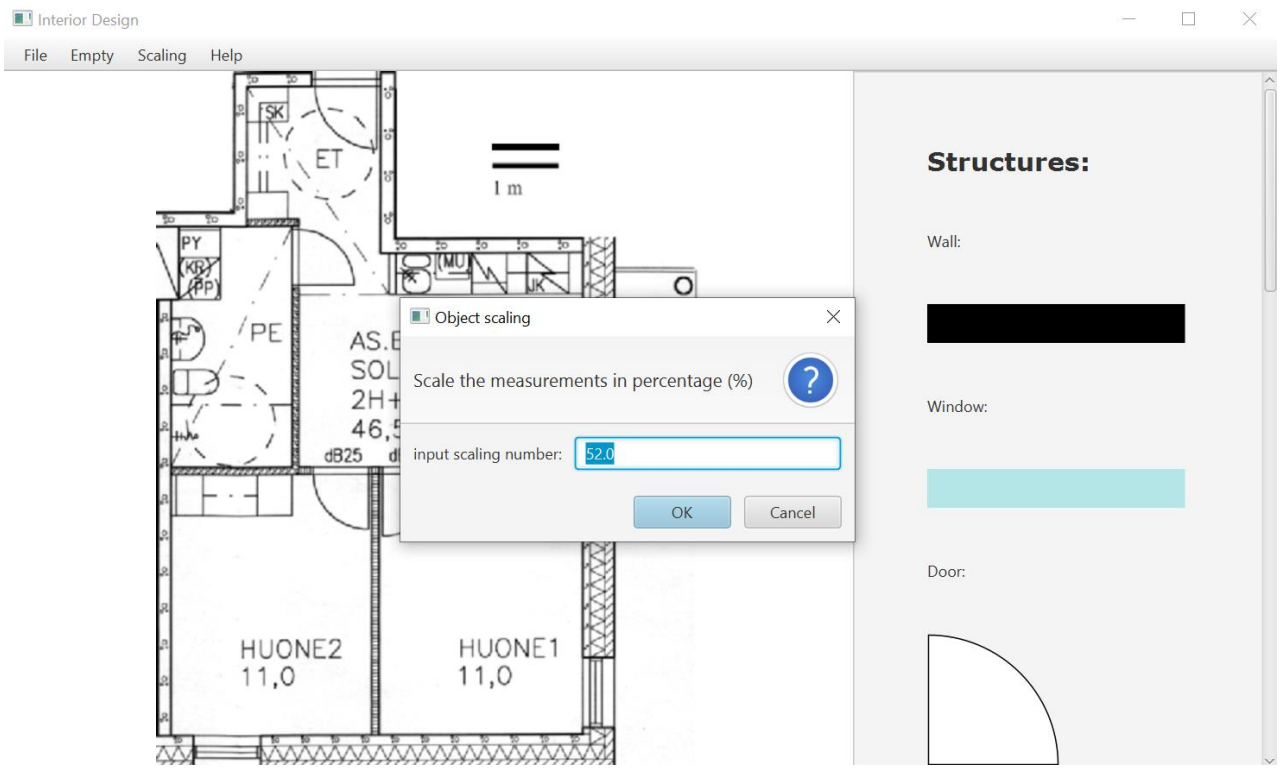
https://stackoverflow.com/questions/30160899/value-change-listener-for-javafxs-textfield

- Medium:
- https://medium.com/@abilngeorge/scala-apply-unapply-unapplyseq-methods-ab8ad22356d0

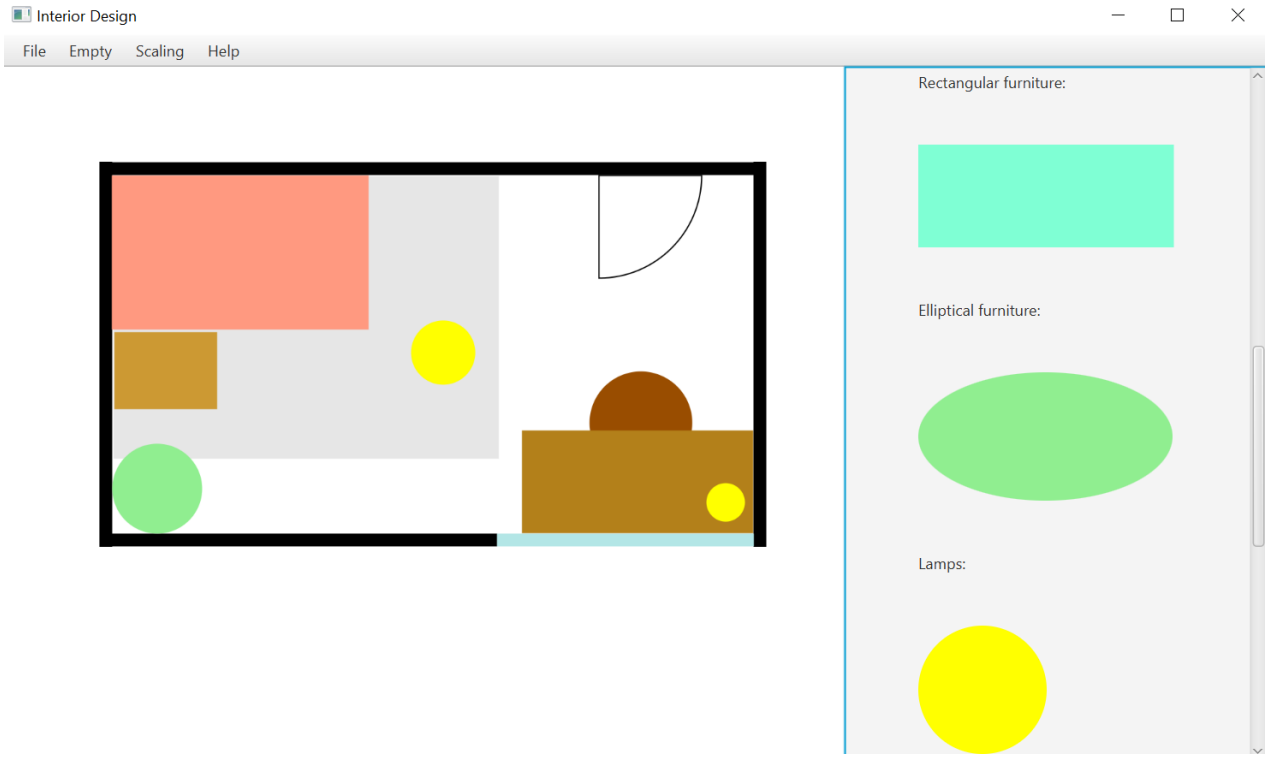
Kun klikataan oikealla olevassa menussa lisättävää huonekalua:



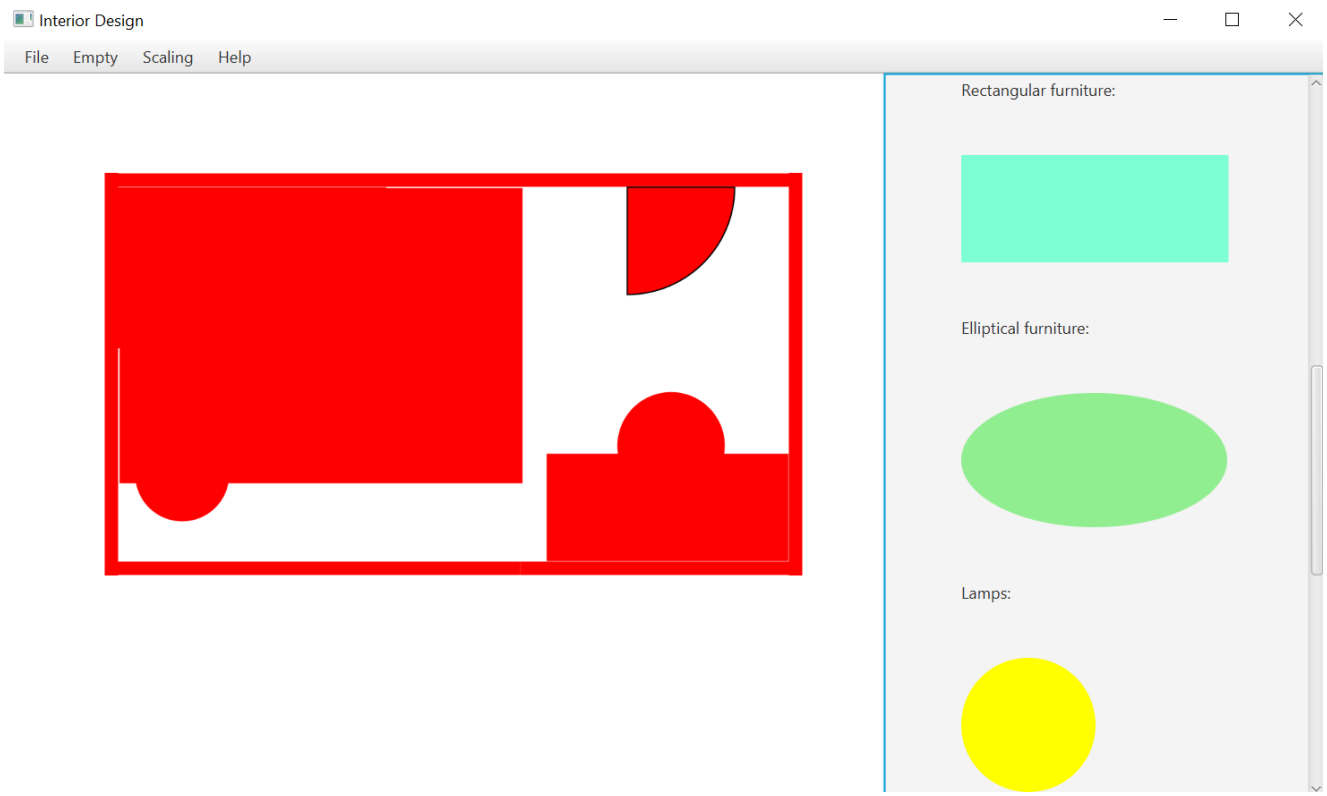
Kun on ladattu pohjapiirros ja löydetty sopiva skaala, jolloin 100 cm pitkä seinä on saman pituinen kuin pohjapiirroksessa metrin skaalausviiva:



Sovelluksella luotu pohjapiirros, jonka voi tallentaa tiedostoon:



Kuva, kun siirrettävä esine menee toisen esineen päälle:


Interior Design
File
Empty
Scaling
Help
Rectangular furniture:
Elliptical furniture:
Lamps: