1. **Henkilötiedot**

Magic of interior design

Nimi: Vilma Judin

Opiskelijanumero: 101718453

Koulutusohjelma: Tietotekniikan kandidaattiohjelma

Vuosikurssi: 2023

Päiväys: 24.4.2024

2. **Yleiskuvaus**

Projekti on toteutettu vaativalla vaikeustasolla. Käyttäjä voi lisätä ohjelmaan oman pohjapiirustuksensa graafisessa muodossa tai luomaan oman pohjapiirustuksensa erillisessä siihen tarkoitetussa näkymässä. Käyttäjä voi tallentaa luomansa pohjapiirustuksen kuvatiedostoon tai käyttää sitä suoraan huoneiston pohjapiirustuksena. Yleisessä suunnitelmassa ajattelin toteuttavani ohjelman siten, että käyttäjän on pakko valita toinen näistä kahdesta vaihtoehdosta, ennen kuin hän pääsee lisäämään huonekaluja suunnitelmaan. Projektia toteuttaessani päätin kuitenkin, että ohjelma avautuu suunnittelunäkymään, jossa esitetään esimerkki pohjapiirustus. Käyttäjä pääsee tästä näkymästä nappia painamalla lisäämään tai luomaan oman pohjapiirustuksensa.

Pohjapiirustus esitetään keskellä käyttöliittymän näkymää. Sen oikealla puolella on huonekaluja sisältävä valikko. Huonekalun saa lisättyä pohjapiirustukseen painamalla nappia, jolloin esiin tulee ikkuna, jossa käyttäjä voi muuttaa huonekalun muutettavia ominaisuuksia, kuten sen mittoja. Näitä ominaisuuksia voi muuttaa myös myöhemmin. Huonekalut saa vedettyä paikoilleen, kuitenkin niin, että ne eivät voi olla pohjapiirustuksen ulkopuolella eivätkä ne voi olla toistensa päällä. Tietyt huonekalut, kuten lamput tai matot, voivat olla toisten huonekalujen päällä tai niiden alla. Kun huonekalut lisätään pohjapiirustukseen, niiden mitat skaalautuvat oikeaan skaalaan. Käyttäjä voi tallentaa pohjapiirustuksen kuvatiedostona. Yleisessä suunnitelmassa

halusin lisätä ohjelmaan ominaisuuden, jolla käyttäjä voi lisätä omia huonekalujaan ohjelmaan, mutta en toteuttanut kyseistä ominaisuutta.

3. Käyttöohje

Käyttäjä voi luoda oman pohjapiirustuksensa siihen tarkoitetussa ikkunassa. Pohjapiirustuksen pituuden ja leveyden voi muuttaa käyttäjän haluamiin mittoihin painamalla Change Scale - näppäintä. Näkymään kasataan yhdellä kertaa koko pohjapiirustus. Käyttäjä voi kasata pohjapiirustuksen käyttämällä valikossa näkyviä elementtejä, kuten ovia, seiniä, ikkunoita ja kaappeja. Kaikkien näiden elementtien mittoja voi muuttaa ja ne voidaan kääntää haluttuun asentoon. Elementit raahataan paikoilleen eivätkä ne voi mennä päällekkäin tai näkymän ulkopuolelle. Seinät voivat mennä toistensa päälle, mutta eivät muiden elementtien päälle. Käyttäjä voi lisätä haluamansa määrän elementtejä ja myöskin poistaa jo olemassa olevia elementtejä. Mikäli käyttäjä haluaa aloittaa pohjapiirustuksen laatimisen alusta, hän voi painaa Restart - näppäintä, joka poistaa kaikki alustalle laitetut osat. Pohjapiirustuksen voi tallentaa kuvatiedostoon tai sitä voi käyttää suoraan suunnitelman pohjana. Tällöin laaditun pohjapiirustuksen elementtien liikuttaminen ei enää onnistu.

Suunnitteluikkuna aukeaa näkymään, jossa pohjapiirustuksen kohdasta löytyy ainoastaan yksi huone. Käyttäjä voi joko lisätä oman pohjapiirustuksensa kuvatiedostona painamalla add floorplan - näppäintä tai mikäli hän on laatinut oman pohjapiirustuksensa ja painanut sen ikkunassa use this floorplan - näppäintä. Kummassakin tapauksessa käyttöliittymän keskelle ilmestyy käyttäjän oma pohjapiirustus. Pohjapiirustuksen kiinteiden esineiden, kuten ovien, paikat voidaan lukea ainoastaan jälkimmäisessä vaihtoehdossa, sillä en onnistunut implementoimaan niiden lukemista kuvan pikselien värien perusteella. Eli siis mikäli käyttäjä on laatinut oman pohjapiirustuksensa, niin huonekalut eivät voi mennä pohjapiirustuksen seinien päälle. Pelkästään kuvatiedostoa käytettäessä käyttäjän on itse tarkistettava, että hän ei sijoita huonekaluja seinien tai ovien päälle.

Käyttäjä voi lisätä näkymään erilaisia huonekaluja painamalla graafisen käyttöliittymän oikeassa laidassa olevasta huonekalupaneelista valitsemansa

huonekalun add new *Huonekalun Nimi* - näppäintä. Tällöin ohjelma avaa ikkunan, jossa käyttäjä voi muuttaa huonekalun mittoja, värejä ja kääntää sen haluamaansa muotoon. Värin muutos tapahtuu colorPickerin avulla. Mittoja puolestaan muutetaan spinnerin tai useamman spinnerin avulla. Huonekalun mittoja muutettaessa käyttäjä kertoo ohjelmalle huonekalun toivotun leveyden ja korkeuden. Esimerkiksi ympyrän muotoisten objektien kohdalla on ympyrän halkaisija käyttäjän antaman mitan suuruinen. Huonekalun kääntäminen tapahtuu painamalla Rotate – nappia, jolloin huonekalu kääntyy 45 astetta. Värin muutoksen ja huonekalun asennon näkee ikkunassa olevasta esimerkkihuonekalusta. Muutokset asetetaan näkymässä näkyvään huonekaluun käyttäjän painaessa Submit – nappia. Tässä samassa ikkunassa sijaitsee myös Delete – nappi, jota painamalla käyttäjä voi poistaa tietyn huonekalun näkymästä. Tämän ikkunan saa aukeamaan painamalla huonekalua hiiren toisella näppäimellä. Kun huonekalua painetaan hiiren päänäppäimellä, voidaan sitä liikuttaa näkymässä. Suurin osa huonekaluista ei saa mennä toisten huonekalujen päälle/alle. Mikäli jokin huonekalu asetetaan toisen huonekalun päälle, palaa liikutettu huonekalu lähtökoordinaatteihinsa. Sama tapahtuu, jos mikään osa huonekalusta liikutetaan käyttöliittymän näkymän ulkopuolelle. Lamput ja matot ovat erikoistapauksia, jotka voidaan sijoittaa muiden huonekalujen päälle/alle, mutta niitäkään ei voida sijoittaa käyttöliittymän ulkopuolelle.

4. Ohjelman rakenne

Luokka Furniture mallintaa lopullisessa toteutuksessani kaikkia mahdollisia graafiseen käyttöliittymään lisättäviä objekteja, myös niitä, joita käytetään pohjapiirustuksen laatimiseen. Kaikkien huonekalujen kokoa ja kulmaa voi muuttaa. Suurimman osan niistä väriä voi myös muuttaa. Tämä poikkeus suunnitelmastani johtuu siitä, että huonekalujen käsitteleminen osoittautui helpommaksi vaihtoehdoksi. Esimerkiksi kun laadin metodia, joka tarkistaa huonekalujen ja seinien päällekkäisyyden oli helpompaa.

Luokka Wall on luokan Furniture alaluokka. Se on erillinen alaluokka, sillä seinät voivat mennä toistensa päällä, mutta muut objektit eivät voi mennä seinien päälle eivätkä

seinät voi mennä muiden objektien päälle. Sen lisäksi kaikki seinät ovat suorakaiteen muotoisia ja niiden väriä ei voi muuttaa.

Luokka Door mallintaa pohjapiirustuksen laatimisessa tarvittavia ovia. Ovilla on tietty muoto, ne ovat muotoa Arc. Ovien väriä ei voi vaihtaa, ne ovat keskeltä valkoisia ja niiden reunat ovat mustat, muistuttaen oikeiden pohjapiirustusten ovien merkintätapaa.

Luokka Window on luokan Wall kaltainen, mutta ikkunat eivät voi mennä toistensa päälle. Kaikki ikkunat ovat suorakulmioita eikä niiden väriä voi muuttaa.

Luokka Counter kuvaa esimerkiksi keittiön kaappeja tai hellaa tai vaikka tiskiallasta. Tämän luokan olioiden väriä ei voi muuttaa, ne ovat keskeltä valkoisia ja niillä on mustat reunat.

Luokka Rug on luokan Furniture erillinen alaluokka, koska halusin, että matot voidaan sijoittaa pohjapiirustuksessa kaikkien muiden huonekalujen alle, riippumatta siitä, lisättiinkö matto ennen toista huonekalua vai vasta toisen huonekalun jälkeen. Koska mattoja voi olla erimuotoisia, oli kaikista kätevintä tehdä uusi luokka Rug. Mikäli huonekalu kuuluu alaluokkaan Rug, se sijoitetaan muiden huonekalujen alapuolelle käyttöliittymässä.

Luokka Lamp on erillinen luokka samankaltaisesta syystä kuin luokka Rug. Halusin, että lamput ovat kaikkien muiden huonekalujen yläpuolella, vaikka ne olisi lisätty ennen muita huonekaluja. Lamp-alaluokan avulla tämä oli mahdollista toteuttaa.

Luokka FurniturePanel mahdollistaa graafisen käyttöliittymän huonekaluvalikon helpon luomisen. Jokaisen luodun huonekalun pohjalta voidaan luoda uusi FurniturePanel olio, joka sitten voidaan lisätä graafisessa käyttöliittymässä olevaan huonekaluvalikkoon. FurniturePanel-luokka mahdollistaa myös huonekalujen lisäämisen graafiseen käyttöliittymään, sillä siellä voidaan kopioida jo olemassa oleva huonekalu huonekalun copy-metodilla ja sitten lisätä kyseiselle huonekalulle toiminnallisuuksia käyttämällä DraggableMaker ja PopUpMaker luokkien olioita ja niiden metodeja.

DraggableMaker-luokka useita ohjelman toiminan kannalta tärkeitä metodeja. Luokan metodi intersectionCheck tarkistaa, ovatko huonekalut toistensa päällä.

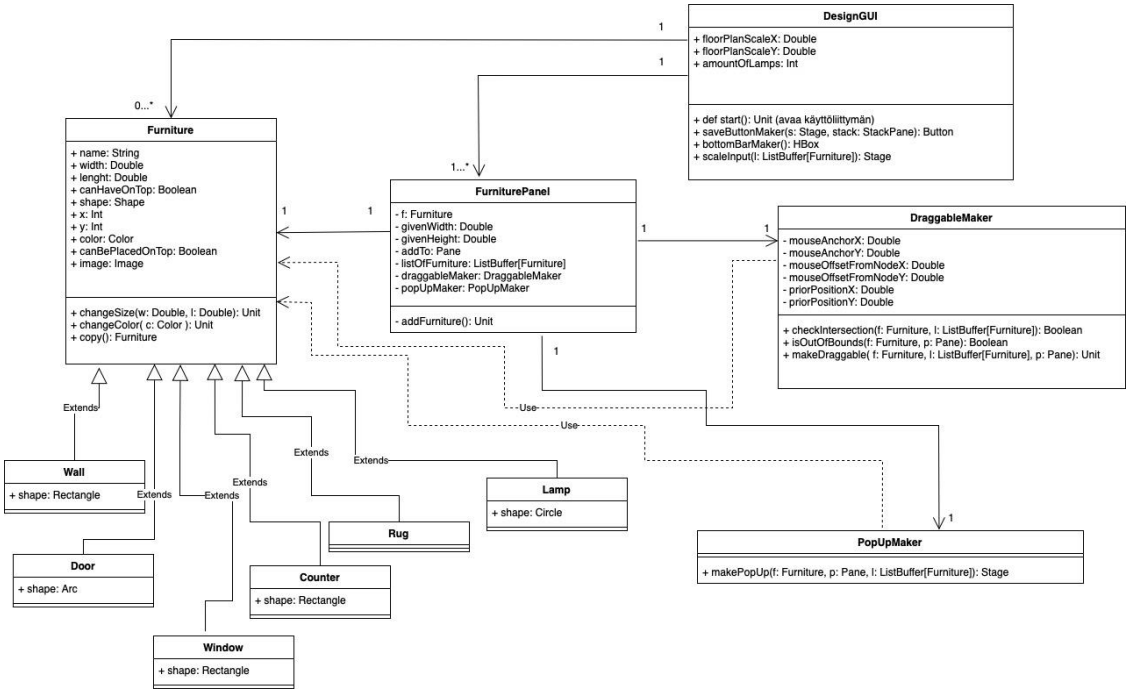
IsOutOfBounds-metodi puolestaan tarkistaa, että huonekalu sijaitsee käyttöliittymän näkymän sisäpuolella. Näitä metodeja kutsutaan makeDraggable-metodissa, joka tekee jostakin tietystä huonekalusta siirrettävän jollakin tietyllä alustalla.

Luokan PopUpMaker oliot luovat uuden ikkunan, jossa voidaan muuttaa huonekalujen ominaisuuksia, kun kutsutaan makePopUp-metodia. Tätä metodia kutsutaan FurniturePanel olion koodissa, tarkemmin silloin, kun jonkin huonekalun muotoa painetaan hiiren toisella näppäimellä.

DesignGUI objekti mahdollistaa ohjelman ajamisen ja käyttöliittymän avaamisen. Metodi start avaa käyttöliittymän. Metodi saveButtonMaker vähentää koodin toisteisuutta ja luo napin, jonka avulla voidaan tallentaa jonkin tietyn stackPanen näkymä. BottomBarMaker-metodi on samankaltainen kuin saveButtonMaker. ScaleInput-metodi luo ikkunan, jossa käyttäjä voi antaa ohjelmalle oman pohjapiirustuksensa pituuden ja korkeuden. Tämä metodi myös ottaa käyttöön käyttäjän antamat mitat ohjelman mittakaavaan ja muuttaa jo olemassa olevien huonekalujen mitat oikeaan mittakaavaan. DesignGUI:ssa on myös koodia, mikä mahdollistaa käyttäjän oman pohjapiirustuksen lisäämisen käyttöliittymään.

Toisin kuin alkuperäisessä suunnitelmassani, toteutukseni ei sisällä luokkia Apartment ja Room. Projektia työstäessäni tulin siihen lopputulokseen, että parempi ratkaisu on antaa käyttäjän laatia suunnitelmansa koko pohjapiirustukseen yhdellä kertaa. Mikäli ohjelman näkymän maksimi pituus ja leveys (20m, 20m) eivät ole tarpeeksi suuret, hän voi laatia huoneet yksi kerrallaan ja tallentaa ne eri tiedostoihin.

Luokkien DraggableMaker ja PopUpMaker koodin olisi voinut sijoittaa suoraan DesignGUI:hin, mutta mielestäni on siistimpää, että nämä monimutkaiset ja pitkät metodit on toteutettu omina luokkinaan.



5. Algoritmit

Huonekalujen skaalan määrittämiseen käytetään ohjelmassa jakolaskua, jossa jaetaan sen alueen leveys, johon huonekalut voidaan sijoittaa, käyttäjän antamilla mitoilla. Käyttöliittymän pituus ja leveys skaalataan käyttäjän antamien mittojen mukaan. Sen jälkeen huonekalut skaalataan kyseiseen skaalaan kertomalla niiden muodon pituus ja leveys kyseisillä arvoilla (tosin tähän käytän Scalan valmista metodia). Esimerkiksi huonekaluikkunan leveys / (käyttäjän antama leveys (m) * 100).

Tiettyjen huonekalujen pyörittämisen yhteydessä on laskettava erikseen niille keskikohta, jotta ne pyörivät oikean akselin ympäri, eivätkä esimerkiksi suorakaiteet vasemman alakulmansa ympäri. Tämä saadaan laskettua helposti suorakaiteille, sillä keskikohdan piste sijaitsee tasan leveyden ja pituuden puolivälissä, jaetaan siis molemmat pituuden kahdella oikean pisteen saamiseksi: (pituus / 2, leveys / 2).

Arc muoto pyörii ympyrän keskikohdan ympäri, mutta käyttöliittymän ulkonäön ja käytännöllisyyden takia on parempi, jos se pyörii oman keskikohtansa ympäri. Koska ympyrän keskikohta on arc muodolle sen vasemmassa yläkulmassa, on sen keskikohdan y-pisteen koordinaatti negatiivinen: (pituus / 2, - leveys / 2).

Huonekalujen liikuttamisen yhteydessä on laskettava sen kohdan, jota hiirellä klikataan, ero huonekalun omiin x- ja y-koordinaatteihin, sillä muuten huonekalu

hyppää vääriin koordinaatteihin sen liikuttamisen päätteeksi. Tämä voidaan laskea erikseen x- ja y-koordinaateille vähennyslaskulla, esimerkiksi x-koordinaateille: huonekalu.x - hiiri.x. Huonekalun liikuttamisen aikana on myös vähennettävä hiiren klikkauskohdan x-koordinaatti tapahtuman x-koordinaatista, jotta huonekalu liikkuu oikein hiiren mukana. Kun huonekalusta päästetään irti, täytyy hiiren koordinaatteihin puolestaan lisätä alussa laskettu erotus, jotta huonekalu pysyy juuri siinä paikassa, johon se liikutettiin, eikä se hyppää hiiren varsinaisiin koordinaatteihin.

6. Tietorakenteet

Päädyin käyttämään projektissani mutable tyyppisiä ListBuffereita, sillä ne ovat erittäin hyödyllisiä tietyllä pohjapiirustuksella sijaitsevien huonekalujen seuraamiseen. Itse huonekaluolioita on ohjelmani kannalta tärkeää säilyttää, sillä huonekaluolioilla on tieto esimerkiksi huonekalun koordinaateista ja väristä. Sen lisäksi huonekaluolioilla on hyödyllisiä metodeja, kuten changeSize ja chageColor. Näitä huonekalulistoja ohjelma hyödyntää lähinnä tarkastaessaan huonekalujen päällekkäisyyksiä, sillä näin tarkistus voidaan tehdä yksittäin jokaisen listassa olevan huonekalun kohdalla. Tämä mahdollistaa esimerkiksi sen, että vaikka huonekalun päällä olisi lamppu, ei sitä voi laittaa toisen huonekalun päälle. Tähän voisi käyttää myös esimerkiksi buffereita ja ratkaisu olisi aivan yhtä toimiva. Valitsin listBufferin, sillä se voi sisältää myös duplikaatteja.

7. Tiedostot ja verkossa oleva tieto

Ohjelma käsittelee käyttäjän sille antamia kuvia, jotka ovat binaaritiedostoja. Ohjelman ei tarvitse käsitellä muun tyyppisiä tiedostoja, sillä en päätynyt tekemään huonekaluille erillisiä pitkiä kuvauksia. Testausta varten projektin test kansiosta löytyy muutamia pohjapiirustuksia.

8. Testaus

Päädyin suorittamaan lähes kaiken testauksen graafisen käyttöliittymän kautta, toisin kuin olin kirjoittanut suunnitelmassani. Tämä johtui siitä, että alustavan graafisen käyttöliittymän luominen ei ollut yhtä hankalaa, kuin aluksi luulin. Tulin nopeasti siihen lopputulokseen, että projektin osien testaaminen yksittäin johtaisi suuriin ongelmiin myöhemmin. Tämä johtopäätös johtui siitä, että ohjelman graafinen käyttöliittymä on suuri ja monimutkainen. Sen kaikkien osien on pystyttävä toimimaan hyvin yhdessä. Huomasin, että pieni virhe jossakin ohjelman osassa saattoi tehdä koko ohjelmasta käyttökelvottoman.

Käyttäjä ei voi antaa ohjelmalle väärän muotoisia tiedostoja, sillä fileChooser estää muiden kuin kuvatiedostojen käytön. Sen lisäksi ohjelmassa ei ole tekstikenttiä, vaan esimerkiksi mittojen muuttamiseen käytetään spinnereitä ja värin muuttamisen colorPickeriä, jolloin niiden arvot ovat aina oikeaa tyyppiä.

9. Ohjelman tunnetut puutteet ja viat

Kun käyttäjä kertoo ohjelmalle pohjapiirustuksensa mitata, mikäli ne eivät ole suhteessa 3/2 (leveys suurempi mitta), vääristyvät kaikkien huonekalujen muodot. Tämä johtuu siitä, että huonekaluja sisältävän alueen mitat ovat kyseisessä suhteessa. En kuitenkaan osannut päättää, mitään muuta järkevää tapaa, sillä käyttäjä saattaa haluta esimerkiksi neliön muotoisen huoneen, jota varten olisi muutettava ohjelman ikkunan kokoa. Ongelman voisi siis korjata esimerkiksi muuttamalla käyttöliittymän ikkunan muotoa siten, että sen mitat sopivat annetun pohjapiirustuksen mittojen suhteeseen, jolloin huonekalujen muodot eivät vääristyisi. Minulla ei ollut aikaa implementoida kyseistä metodia kunnolla, yritin sellaisen laatimista hieman, mutta yleensä se johti ohjelman toiminnan häiriintymiseen.

Kun käyttäjä kääntää huonekaluja, esimerkiksi niiden kulma saattaa mennä toisen huonekalun päälle tai käyttöliittymän ikkunan ulkopuolelle. Tämän voisi korjata käyttämällä DraggableMaker-luokan intersectionCheck metodia sekä saman luokan outOfBounds-metodia. Käyttäjälle pitäisi ilmoittaa, että kyseiset muutokset eivät ole mahdollisia, sillä huonekalu menee käyttöliittymän ulkopuolelle tai toisen huonekalun

tai seinän päälle. Tämä kannattaisi toteuttaa jonkin sortin error viestin avulla, kun käyttäjä koittaa painaa submit-nappia.

Eräs toinen ongelma huonekalujen kääntämisen suhteen, kun huonekaluja kääntää eri asentoon, outOfBounds-metodi ei aina toimi kunnolla. Huonekaluja ei voi edelleenkään siirtää kokonaan käyttöliittymän ulkopuolelle, mutta yleensä ongelma esiintyy niiden alakulman kohdalla ja sen voi siirtää hieman käyttöliittymän alareunan alapuolelle. Bugi tapahtuu yleensä silloin, jos huonekalun kulma on mennyt kääntämisen aikana käyttöikkunan alareunan alapuolelle ja huonekalu on 45 asteen kulmassa. Tosin tämä bugi voi tapahtua muutenkin, mikäli huonekalu on 45 asteen kulmassa.

Kun huonekalujen kokoa tai kulmaa muuttaa ja sitten klikkaa seuraavan kerran huonekalun muutosikkunaan, muutosikkunassa sijaitseva esimerkkihuonekalu saattaa olla osittain tekstin tai colorPickerin alla. En ole aivan varma mistä tämä ongelma johtuu, mutta uskon, että osaisin korjata kyseisen ongelman, mikäli minulla olisi enemmän aikaa jäljellä.

10. 3 parasta ja 3 heikointa kohtaa

Se, miten huonekalut lisätään pohjapiirustukseen, on osa, josta olen ylpeä, sillä matot menevät kaikkien muiden huonekalujen alle ja lamput kaikkien muiden huonekalujen päälle. Ei siis tarvitse lisätä ensin mattoja, sitten huonekaluja ja viimeisenä lamppuja, vaan niitä voi lisätä missä vain kohtaa ja ne ovat oikeassa järjestyksessä.

Toinen osa, joka toimii todella hyvin, on sen tarkistaminen, että huonekalut ovat kokonaan piirtoikkunassa. Huonekalun kulmalla tai koolla ei ole väliä, sille mikään osa siitä ei voi mennä suunnittelualueen ulkopuolelle. Tässä osassa on valitettavasti bugi, josta olen selittänyt enemmän kohdassa yhdeksän. Bugi ei ole kovin suuri, joten olen silti todella ylpeä tästä metodista.

Kolmas osa, josta olen todella ylpeä, on se, kuinka ohjelma tarkistaa sen menevätkö huonekalujen päällekkäin. Esimerkiksi samassa kohdassa voi olla vaikka sohva, lamppu ja matto, mutta sohvan päälle ei silti saa siirrettyä esimerkiksi pöytää. Toisaalta vaikkapa maton kulman päälle pöydän saa siirrettyä, kunhan se ei ole sohvan päällä.

Skaalaus on yksi projektini heikommista osa-alueista. Kuten selitin jo puutteiden kohdalla, huonekalujen muodot saattavat vääristyvät käyttäjän antaessa mitat ohjelmalle. Tämän vuoksi huonekalujen muodot ovat sitä epäselvemmät mitä kauempana mittakaava on 3/2 mittakaavasta, esimerkiksi ympyrät alkavat muistuttamaan ellipsejä. Halusin kuitenkin laatia ohjelmasta sen verran käyttäjäystävällisen, että käyttäjä saa itse päättää pohjapiirustuksensa mitat.

Toinen heikko kohta ohjelmassani liittyy huonekalujen muutosikkunaan. Huonekaluja kuvaavien muotojen asettelu kyseisessä ikkunassa on välillä hieman huono, jolloin ne menevät osittain esimerkiksi tekstin tai colorPickerin alle. Olen yrittänyt useampaa eri toteutusta tälle, mutta mikään niistä ei ole toiminut tämänhetkistä toteutustani paremmin.

11. Poikkeamat suunnitelmasta, toteutunut työjärjestys ja aikataulu

En osannut toteuttaa seinien lukemista kuvatiedostosta, vaikka suunnittelin tekeväni niin. Yritin useampaa eri toteutusta, mutta kaikki niistä aiheuttivat suuria ongelmia huonekalujen liikuttamisen suhteen. Koska huonekalujen liikuttamisen onnistuminen hyvin on tärkeämpää, päätin poistaa seinien lukemisesta vastaavan metodin kokonaan.

Sen lisäksi en päätynyt toteuttamaan sitä, että käyttäjä voi lisätä omia huonekaluja ohjelmaan. Tämä johtui pitkälti siitä, että minulla ei riittänyt aikaa kyseisen metodin toteuttamista varten.

Ensimmäisen sprintin aikana laadin alustavan luokkarakenteen ja alkukantaisen graafisen käyttöliittymän, joka ei vielä tehnyt mitään. Samalla tutustuin graafisen käyttöliittymän mahdollisiin rakenne-elementteihin. Tämä vastaa pitkälti sitä, mitä olin kirjoittanut suunnitelmaani. Aika-arvioni tälle sprintille oli puolestaan aivan liian suuri, käytin todennäköisesti noin kolme tuntia projektiin tämän sprintin aikana.

Toisen sprintin aikana toteutin pohjapiirustuksen lisäämisen käyttöliittymän keskelle. Aloin myös laatimaan sitä osaa ohjelmasta, joka mahdollistaa huonekalujen

lisäämisen käyttöliittymään. Tämä jälleen vastaa pitkälti sitä, mitä olin suunnitellut toteuttavani. Aika-arvioni tällekin sprintille oli yläkanttiin, vietin ehkä noin viisi tuntia projektin parissa tällä sprintillä. Virheellinen aika-arvioni johtui pitkälti siitä, että uskoin kuvatiedostojen lukemisen olevan paljon monimutkaisempaa, kuin se oikeasti on.

Kolmannen sprintin aikana sain huonekalut liikkumaan, implementoin niiden värin vaihtamisen sekä tarkistuksen siitä, että huonekalut eivät mene toistensa päälle. Toteutin myös ohjelman näkymän tallentamisen kuvatiedostoon tämän sprintin aikana. Toteutin myös suunnitelmassani kuvatut luokat Furniture ja Part. Tämän sprintin aikana vietin projektin parissa todella paljon aikaa, todennäköisesti noin 30–40 tuntia. Virhe aika-arviossani johtuu todennäköisesti siitä, että en ollut edellisten sprinttien aikana työstänyt projektia kovin paljon, joten tämä aika olisi jakautunut tasaisemmin kolmen ensimmäisen sprintin ajalle, mikäli olisin käyttänyt aikaisemoien viikkojen aikana enemmän aikaa projektin parissa.

Neljännen sprintin aikana toteutin huonekalujen skaalauksen ja pohjapiirustuksen mittojen muuttamisen. Sen lisäksi huonekalujen kokoa voi nyt muuttaa. Aloitin sellaisen metodin laatimisen, joka tarkistaa, että huonekalut eivät mene käyttöikkunan ulkopuolelle. Käytin tämän sprintin aikana aikaa myös aikaa siihen, että yritin saada ohjelman lukemaan käyttäjän sille antaman pohjapiirustuksen pikseleiden värit ja sen kautta tarkistaa meneekö huonekalu pohjapiirustuksen seinien päälle. Tämä ei kuitenkaan onnistunut. Jälleen seuraan suunnitelmaa aika hyvin, tosin työstän myös metodeja, joita en ollut suunnitelmaa tehdessäni edes ajatellut. Käytin tämän sprintin aikana aikaa projektiin noin 15 tuntia.

Viidennen eli viimeisen sprintin aikana olen lähinnä kommentoinut koodia, laatinut tätä dokumenttia ja korjannut rikkinäisiä osia ohjelmastani. Sen lisäksi lisäsin huonekalujen tietoikkunaan mallikuvat. Tälle sprintille en ollut suunnitelmassani asettanut suuria tavoitteita. Olen kuitenkin käyttänyt tämän sprintin aikana projektiin noin 30 tuntia, sillä dokumentin kirjoittaminen, koodin siistiminen ja metodien refaktorointi ovat vieneet enemmän aikaa kuin suunnitelmaa tehdessäni uskoin niiden vievän.

Suurimmat erot olivat ajankäytössä. Alussa minun oli vaikea aloittaa projektia, joten työstänyt sitä paljon. Sen lisäksi ensimmäisten sprinttien aikana muut koulutyöt pitivät minut kiireisenä. Pääsiäisloman ja tenttiviikkojen aikana käytin paljon aikaa projektin parissa ja huomasin nauttivani sen tekemisestä. Kokonaisuudessaan ajankäyttöarvioni osui kuitenkin oikeaan. Toteutusjärjestyksen olin suunnitellut melko toimivasti, joten siihen en tehnyt suuria muutoksia.

Opin prosessin aikana, että asiat kannattaa aloittaa ajoissa. Sen lisäksi opin, että aikatauluttaminen on tärkeää. Tajusin myös, että välillä dokumentaation, kurssimateriaalin tai tutoriaalien lukemisesta on enemmän hyötyä kuin siitä, että vietän useita tunteja yrittäen saada samaa metodia toimimaan yksinäni. Tämä tapahtui esimerkiksi huonekalujen liikuttamisen ja tiedostojen tallentamisen kohdalla.

12. Kokonaisarvio lopputuloksesta

Kokonaisuudessaan olen todella tyytyväinen siihen, miltä projekti näyttää ja omaan työpanokseeni projektin suhteen. Projektin tekeminen oli ajoittain todella palkitsevaa ja välillä taas todella turhauttavaa. Valitsemani työjärjestys oli todella toimiva ja uusien metodien testaaminen graafisessa käyttöliittymässä oli sen vuoksi todella helppoa. Uskon, että tämä helpotti koko projektin laatimista kokonaisuudessaan, sillä graafinen käyttöliittymä on tämän projektin kaikista tärkein osa.

Mielestäni ohjelman laatu on melko hyvä. Se toteuttaa kaiken, mitä tehtävänanto vaatii sen toteuttavan ja sillä on vielä lisäksi muita ominaisuuksia. Esimerkiksi se, että huonekalut eivät saa mennä käyttöliittymän näkymän ulkopuolelle on mielestäni todella hyvä lisä ohjelmaan, vaikka se ei aina toimi aivan täysin.

Yksi ohjelmani huonoista puolista on se, että huonekalut täytyy lisätä manuaalisesti itse koodiin, eikä niitä voi lisätä esimerkiksi käyttöliittymässä. Tämän metodin toteuttaminen olisi vaatinut huomattavan määrän aikaa, mutta se on toteutettavissa ohjelman nykyisen toteutuksen päälle.

Työni oleellisin puute on skaalauksen tämänhetkisen implementaation aiheuttama huonekalujen muotojen vääristyminen. Tämän olisi voinut korjata esimerkiksi siten, että käyttäjä voi muuttaa ainoastaan yhtä pohjapiirustuksen mitoista, jolloin mittojen

suhde pysyisi aina samana. Olen kuitenkin sitä mieltä, että tämä olisi laiska implementaatio, samoin kuin se, että käyttäjä antaa ohjelmalle ainoastaan tietyn skaalan. Kuten kuvasin aikaisemmin, tämänhetkisen implementaationi ongelman saisi korjattua esimerkiksi siten, että käyttöliittymän mittoja muutetaan mittakaavasta riippuen. Tällöin huonekalujen mitat eivät vääristyisi.

Tulevaisuudessa ohjelmaa voisi parantaa esimerkiksi implementoimalla metodin, joka tarkistaa, että huonekalut eivät mene seinien tai muiden huonekalujen päälle, jos niiden kokoa tai kulmaa muutetaan. Sen lisäksi ohjelmaa voisi parantaa siten, että mahdollistetaan käyttäjän omien huonekalujen lisääminen. Näistä molemmat ovat toteutettavissa nykyisen projektini päälle.

Uskon, että ohjelmani tämänhetkinen rakenne soveltuu hyvin muutosten tai laajennusten tekemiseen suurimmaksi osaksi. Luokkarakenne on mielestäni toimiva eikä sitä tarvitse muuttaa, mikäli haluaa laajentaa ohjelmaa. Esimerkiksi mikäli halutaan implementoida graafisen käyttöliittymän koon muuttuminen riippuen käyttäjän syöttämistä mitoista, täytyisi graafisessa käyttöliittymässä käyttää GridPanea HBoxien ja VBoxien sijaan. Tiedän tämän, sillä aloitin kyseisen toteutuksen laatimista, mutta minulta loppui aika enkä kerennyt tekemään metodia kunnolla loppuun, joten poistin sen.

Mikäli aloittaisin projektin nyt alusta, olisin pyrkinyt työstämään projektia enemmän ja laatimaan selkeitä tavoitteita itselleni projektin suhteen alusta alkaen.

13. Viitteet ja muualta otettu koodi

Toteuttaessani huonekalujen liikuttamista aloitin yrittämällä toteuttaa sen aivan itse. Koska tämä on ensimmäinen kerta, kun työskentelen kunnolla graafisten elementtien parissa, epäonnistuin tässä. Seuraavalla yrityksellläni sain huonekalut liikkumaan, mutta niiden liikuttaminen oli todella bugista. Päädyin etsimään tietoa netistä, ja löysin tämän E. Eden-Rumpin (2020) tutoriaalin siitä, kuinka liikuttaa muotoja javafx:ssä. Päädyin tekemään huonekalujen liikuttamisen tämän tutoriaalin pohjalta ja lisäsin omat metodini DraggableMaker luokkaan. Sen lisäksi lisäsin tiedon huonekalun aikaisemmasta sijainnista, jotta se voidaan palauttaa aikaisempiin

koordinaatteihinsa, mikäli se on toisen huonekalun päällä tai graafisen käyttöliittymän näkymän ulkopuolella.

Osasin laatia filechooserin näkymän itse, mutta stackPanen muuttaminen tallennettavaksi tiedostoksi tuotti minulle hieman vaikeuksia, enkä ollut varma, kuinka edetä. Löysin tietoa tästä StackOverflowsta.

Löysin tietoa JavaFX:n metodeista, luokista ja niiden toiminnasta Oracle-nettisivulta. ScalaFX:n vastaavista löysin tietoa ScalaFX-nettisivulta.

Lähteet:

Eden-Rump, E. (2020) *The best way to drag shapes in Javafx.* EdenCoding. Available: https://edencoding.com/drag-shapes-javafx/#a-the-simplest-solution [24.8.2024]

Stack overflow (2017) *JavaFX: Save view of a pane to image [duplicate].* Available: https://stackoverflow.com/questions/38028825/javafx-save-view-of-pane-to-image

Referenced: [24.8.2024]

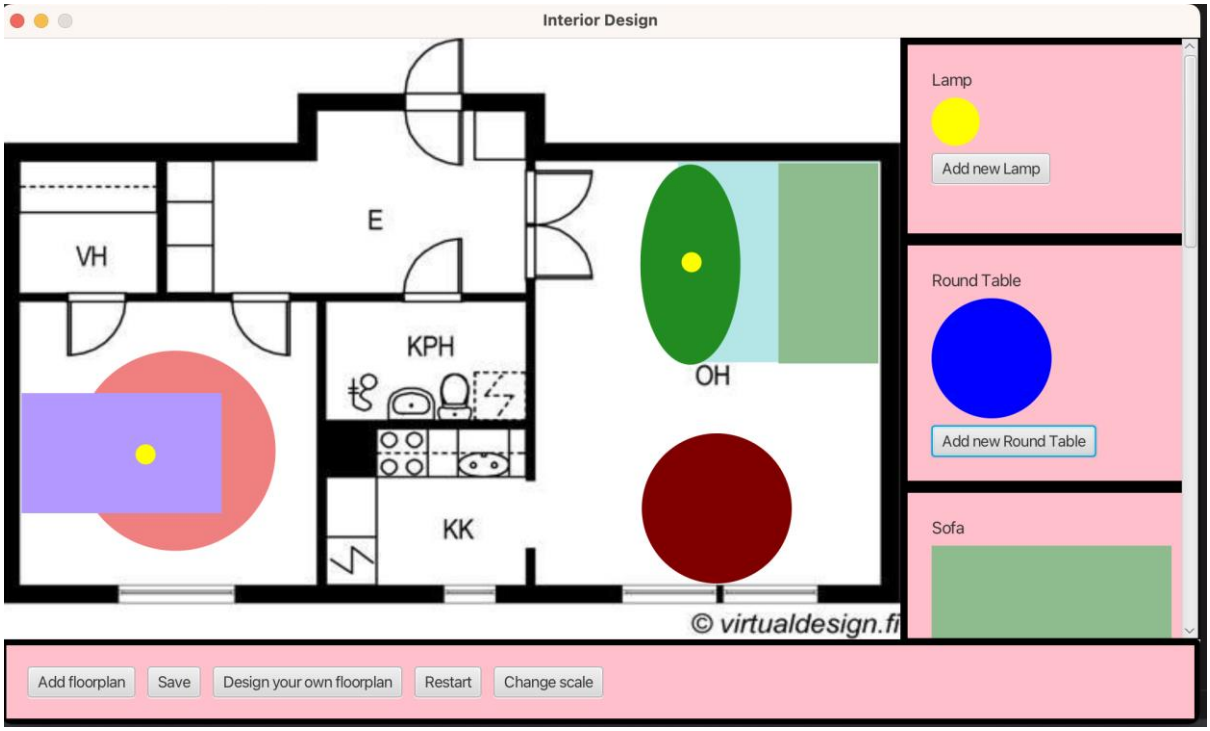
Oracle (n. d.) Available: https://docs.oracle.com/javase/8/javafx/api/javafx/scene/SnapshotResult.html [24.4.2024]

ScalaFX (n. d.) *What is ScalaFX.* Available: https://www.scalafx.org/docs/home/

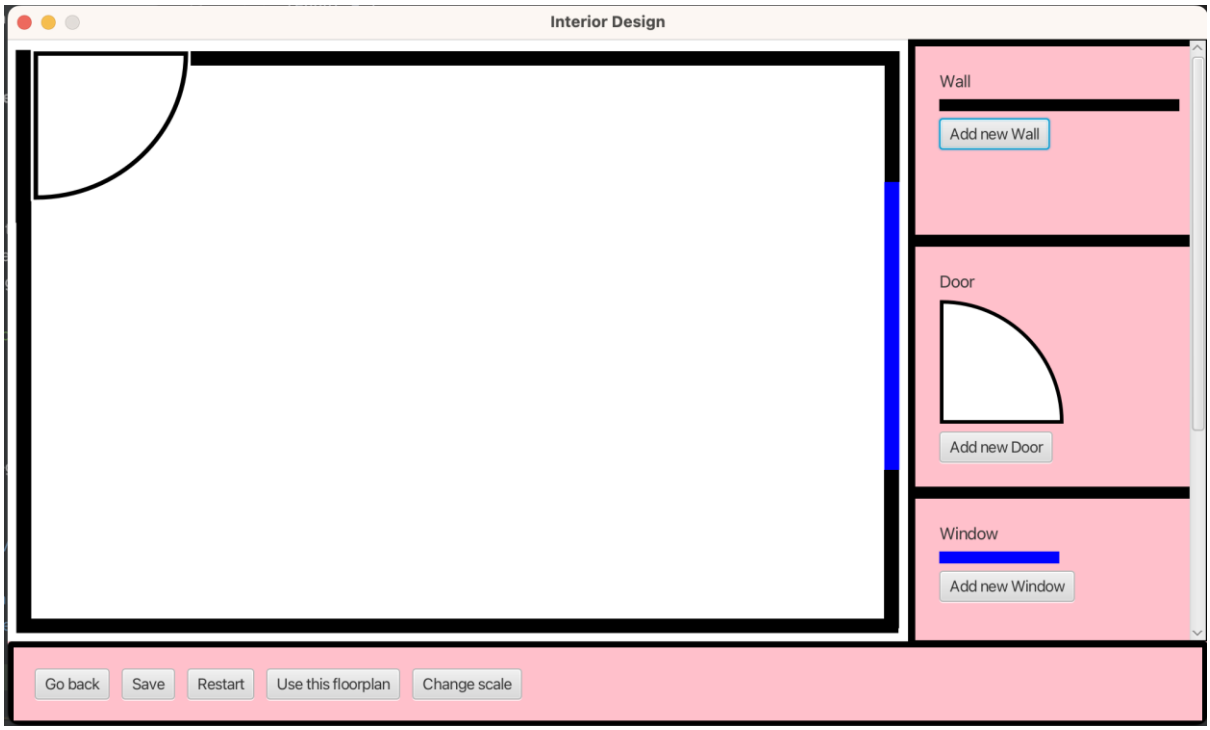
Referenced: [24.8.2024]

14. Liitteet

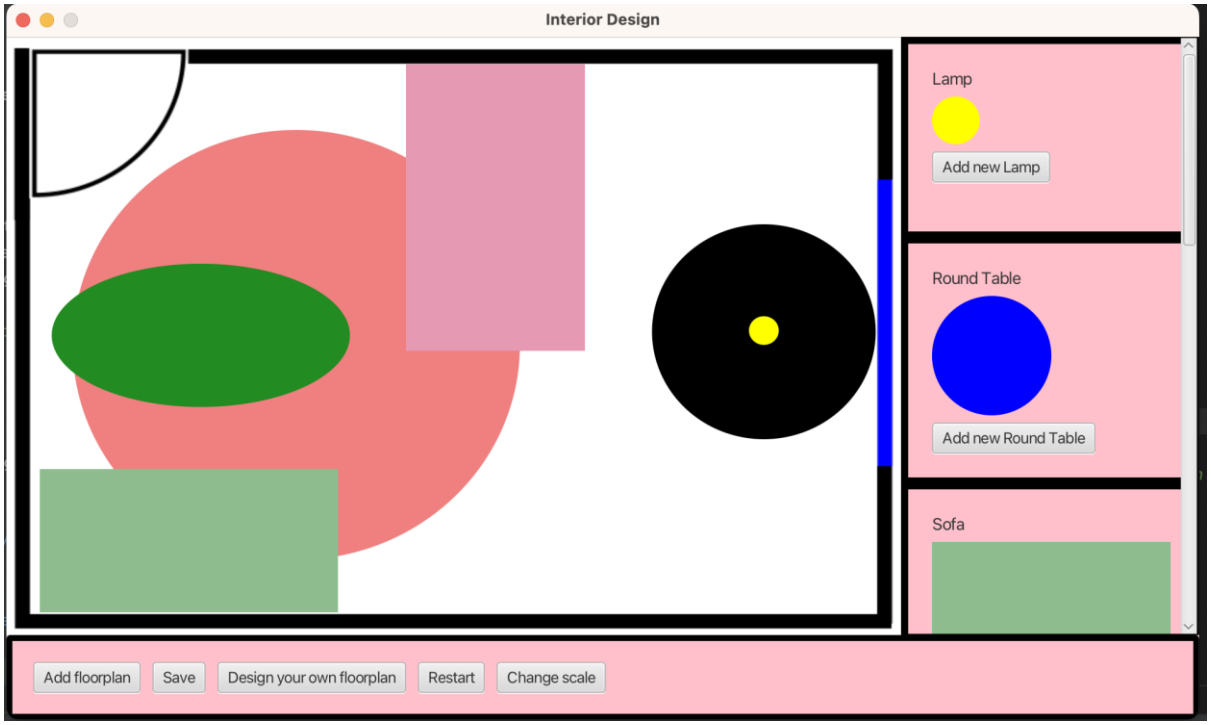
Käyttäjä on lisännyt pohjapiirustukseen sohvan, pöydän, sängyn ja muita huonekaluja.



Käyttäjä luo oman huoneensa pohjapiirustuksen, joka on kokona 6 m x 4 m



Käyttäjä päättää käyttää kyseisen huoneen pohjapiirustusta ja suunnittelee, mitä laittaa huoneeseen:


Interior Design
Lamp
Add new Lamp
Round Table
Add new Round Table
Sofa
Add floorplan
Save
Design your own floorplan
Restart
Change scale