

Peppi Tarkka

101510143

Tietotekniikan kandidaatti – 2023

Dream Space – dokumentaatio

Yleiskuvaus

Dream Space on tietokoneohjelma, jonka perusideana on suunnitella unelmien sisutus käyttäjän tulevaan asuntoon. Käyttäjä pystyy lataaman kuvan pohjapiirroksesta omista tiedostoistaan, jonka jälkeen valittu kuva piirtyy ruudulle. Ohjelmassa voi valita vasemmalta valikosta huonekaluja, joita halutaan lisätä pohjapiirroksesi päälle. Näin sovelluksen käyttäjä voi siis hahmotella unelmiensa sisutuksen oman asuntonsa pohjapiirroksen päälle. Ohjelma on toteutettu keskivaikean vaikeustason mukaisesti, mutta vaativan vaikeustason piirre (huonekalujen värit) on myös sisällytetty ohjelman toteutukseen.

Asunnon pohjapiirros luetaan PNG- tai JPG-muodossa annetusta tiedostosta ja se esitetään ohjelmassa keskellä ruutua. Ruudun vasemmassa laidassa on valikko, jossa esitetään erilaisia huonekaluja. Huonekalut ovat kuvattu ylhäältä katsottuna, joten ne ovat geometrisiä muotoja. Käyttäjän halutessa tietyn huonekalun asuntoon, hän valitsee sen valikosta ja siirtää huonekalun pohjapiirroksen päälle. Samalla käyttäjä valitsee huonekalun muodon, värin ja antaa mitat, jotta suunnittelu säilyy realistisena ja suunnitelmaa voi hyödyntää paremmin oikeassa elämässä. Valitun huonekalun voi sijoittaa pohjapiirrokseseen haluamalleen paikalle ja sen voi kääntää haluttuun asentoon huoneessa.

Voit luoda esimerkiksi olohuoneen, jossa on sohva, matto, tv-taso sekä televisio, jonka voit asettaa esimerkiksi seinälle tai tv-pöydän päälle. Valitessasi huonekalujen värit näet selkeästi, jos joidenkin huonekalujen värimaailmat ovat ristiriidassa toistensa kanssa. Tällöin unelmakodin suunnittelija välttää isoimmat sisustusvirheet, kun hän suunnittelee omaa kotiaan. Lisäksi mitoitus huonekaluissa auttavat hahmottamaan, sopiiko haluttu uusi huonekalu sen suunnitellulle paikalle huoneessa.

Samoja huonekaluja voi olla monta huoneessa, ja ohjelma pitääkin huolta siitä, etteivät huonekalut mene toistensa päälle. Poikkeuksena ovat tietysti ne huonekalut, jotka voivat asettaa oikeastikin toisen huonekalun päälle (esimerkiksi lamput, jotka voivat olla katossa tai pöydällä).

Suunnitelman voi lopuksi tallentaa tiedostoon, ja käyttäjä voi tutkia sitä halutessaan sieltä.

Käyttöohje

Ohjelma käynnistetään koodin luokan "Main" objektista "Main", jossa on pieni vihreä nuoli nimen vieressä. ^[1] Käyttäjälle avautuu ikkuna, jossa vasemmalta löytyy erilaisia nappeja, mitä käyttäjä voi painaa. ^[2] Ylimpänä esiintyy "File" niminen nappi ^[3]. Painaessa tekstiä "File", nimen alle muodostuu valikko kahdesta napista: "New" ja "Save". Kun käyttäjä haluaa ladata tiedostoistaan oman asuntonsa pohjapiirroksen kuvan, täytyy hänen painaa nappia "New". Tällöin tietokone avaa tiedostonäkymän, josta käyttäjä voi valita halutun kuvatiedoston, joka on muotoa PNG tai JPG. Käyttäjän valitessa tiedoston, kuva piirtyy keskelle ohjelman ruutua. ^[4]

Tämän jälkeen käyttäjä voi valita huonekalun vasemman sivun valikosta, joissa jokainen huonekalu on nimetty. Valitessaan huonekalun ruudulle pompahtaa ikkuna, jossa käyttäjältä kysytään huonekalulle muotoa ^[5], jos vain muoto on edes mahdollista antaa kyseiselle huonekalulle. Huonekaluille tuoli, sänky, lappu ja TV muoto on jo valmiiksi valittu. Muodon valittua pompahtaa uusi ikkuna ^[6], jossa huonekalun mittoja kysytään; suorakulmaisella kappaleella kysytään leveyttä sekä pituutta, ja ympyrän ja ellipsin muotoisilta kappaleilta pyydetään käyttäjän asettamaan kappaleiden halkaisijat. Mitat asetetaan huonekalulle senttimetreinä. Jos mittaan halutaan lisätä desimaalinumeroita, on tämä mahdollista, mutta desimaalit on erotettava pistesymbolilla (.). Jos käyttäjä käyttää pillkku-symbolia (,), ei ohjelma osaa lukea annettua mitta. Tällöin ikkuna, jossa kysytään mittoja ei poistu, mutta huonekalukaan ei piirry näytölle

Samassa ikkunassa ^[6] missä kysytään huonekalulle mitat, pyydetään käyttäjää valitsemaan huonekalulle myös väri. Tämän voi valita painamalla värin valinta nappia, jolloin käyttäjälle ilmestyy ruudukko erilaisista väreistä. Lisäksi ruudukon alareunasta voi käyttäjä painaa nappia "Custom Color" jolloin erilaisilla säädöillä voi ohjelman käyttäjä luoda itse halutun värin huonekalulleen.

Kun mitat ja värit ovat lisätty, piirtyy huonekalu pohjapiirustuksen päälle. Käyttäjä voi siirtää huonekalun haluttuun paikkaan liikuttamalla hiirtä samalla, kun hiiren vasen näppäin on painettuna pohjassa. Jos käyttäjä haluaa kääntää huonekalua, voi hän painaa hiiren oikeaa näppäintä. Tällöin kuvaan ilmestyy nappi "Rotate".^[7] Tätä painamalla huonekalu kääntyy 45 astetta myötäpäivään.

Käyttäjä voi lisätä huonekaluja kuvaan niin monta kuin itse haluaa. Joitakin huonekaluja ei kuitenkaan voi laittaa päällekkäin. Esimerkiksi et voi asettaa sohvan päälle pöytää. Logiikka on sama kuin reaali maailmassa. Jos kuitenkin käyttäjä sattuu asettamaan huonekalun toisen huonekalun päälle, mihin se ei kuulu, ohjelma syöttää näytölle ilmoituksen.^[8] Ilmoituksessa lukee tieto siitä, ettei kyseistä huonekalua voi laittaa tämän toisen huonekalun päälle. Kun ilmoituksen painaa pois, huonekalu palaa samalle paikalle, mistä se lähtikin.

Ohjelmassa on kuitenkin pieni viallisuus, joka pitää ottaa huomioon suunnitellessa sisustusta. Huonekalut voivat kasautua toistensa päälle vain siinä järjestyksessä kuin käyttäjä on ne

lisännyt näytölle. Esimerkiksi, jos käyttäjä haluaa asettaa pöydän maton päälle, on hänen ensin valittava valikosta matto ja tämän jälkeen vasta pöytä. Lisäksi on otettava huomioon, että vaikka käyttäjä olisi valinnut huonekalut oikeassa järjestyksessä näytölle, kun siirret pöydän maton päälle, täytyy nimenomaan siirtää pöytä maton päälle eikä matto pöydän alle. Jos käyttäjä siirtää mattoa, joka asettuu pöydän alle, ohjelma luulee, että matto on pöydän päällä vaikka näin ei olisi. Tällöin syntyy vain virheilmoitus, eikä matto asetu oikein pöydän alle.

Kun käyttäjän sisustussuunnitelma on valmis, voi hän tallentaa sen vasemman yläkulman "File" -valikon napista "Save". Suunnitelma tallentuu kuvana tietokoneelle käyttäjän valitsemaan tiedostoon, mistä sitä voi tutkia myöhemmin halutessaan.

Ohjelman rakenne

Ohjelma koostuu neljästä luokasta: Main.scala, Controller.scala, Furniture.scala sekä sizeSelect.scala.

Luokka Main.scala pitää sisällään ohjelman graafisen käyttöliittymän ja sen sisältämät elementit, kuten napit ja niiden toiminnot. Tämä käyttöliittymän luokka käyttää jokaista muuta luokkaa toiminnoissaan.

Furniture.scala -niminen luokka kuvaa huonekaluja, joita on mahdollista asettaa pohjapiirustuksen päälle. Jokaisella Furniture-luokan objektilla on nimi, joka kertoo, mikä huonekalu on kyseessä. Esimerkiksi sohvalla lukee nimen kohdalla "Sofa", matolla "Carpet" ja niin edespäin. Jokaisella huonekaluoliolla on lisäksi muoto (shape), mikä kertoo nimensä mukaisesti huonekalun muodon. Luokka on perinyt Scalafx:n luokan Shape. Furniture-luokalla on metodi overlapMistake, joka kertoo käyttäjälle, jos siirretty huonekalu on liikutettu paikkaan, missä sen ei ole mahdollista olla. Metodia käytetään, kun ohjelma tarkistaa voiko tietyn huonekalun päälle siirtää käyttäjän liikuttamaa huonekalua.

Luokka Controller.scala pitää sisällään huonekalujen siirtelyyn ja kääntämiseen tarvittavan metodin. Luokan keskeisenä tehtävänä on siis mahdollistaa huonekalujen siirtely, kääntely sekä pitää huolta, että huonekalun väärinasetteluun sattuessa näytölle pompahtaa ilmoitus "Overlap mistake!". Kaikki luokan Controller.scala metodien tapahtumat tapahtuvat Main-luokan näyttämöllä (stage). Lisäksi Controller-luokka käyttää toiminnoissaan luokkaa Furniture.scala, sillä kyseisessä luokassa on metodi "overlapMistake", jota tarvitaan huonekalujen liikuttamisen toiminnassa. DragController.scala:n metodi "createHandlers" sisältää toiminnallisuuden, joka tarvitaan, jotta huonekalu voidaan siirtää paikasta toiseen liikuttamalla hiirtä, kun tämän vasen nappi on painettuna pohjassa. Lisäksi nappi "Rotate" ja sen toiminnot ovat sisällytetty tähän metodiin.

Annettujen mittojen ja värin avulla huonekalujen piirrosta huolehtii luokka SizeSelect.scala. Jokaiselle muodolle on muodostettu omanlainen ilmoitus, missä kysytään mittoja ja värejä.

Poikkeuksena on metodi "sizeSelectForTV". Tässä metodissa erilaista on se, että huonekalulta "TV" ei kysytä väriä, vaan väri on automaattisesti musta. Jokainen eri "sizeSelect" -metodi ottaa "Alert" -kyselybokseista käyttäjän syöttämät mitta-arvot ja mahdollisen värin. Näillä tiedoilla metodit muodostavat huonekaluista annettujen tietojen mukaiset kuvat näytölle. Luokan metodit käyttävät luokkaa "Furniture". Jokainen luokan metodi palauttaa huonekalun, jonka käyttöliittymä "Main" lukee ja lisää sen ohjelman ruudulle.

Toteutunut projektin luokkarakenne eroaa huomattavasti suunnitelman rakenteesta. Ensinnäkin, jokaiselle huonekalulle ei tehty omaa luokaa, vaan eri huonekalut ovat kaikki saman luokan "Furniture" sisällä. Tähän päätökseen päädyttiin, kun huomasin että eri huonekalut voidaan erottaa toisistaan pelkän nimen avulla. Erilaiset geometriset muodot, jotka huonekalulla on mahdollista olla, erotellaan jo "Alerteissa", mitkä ponnahtavat näytölle, kun käyttäjä valitsee huonekalun. Tämän vuoksi ei tarvita erillistä luokkaa, missä mahdolliset muodot esitetään. Lisäksi metodissa, missä käsitellään huonekalujen päällekkäin asettelujen mahdollisuuksia voi huonekalut ja niiden asettelumahdollisuudet erotella pelkän nimen avulla. Kun loin koodia, huomasin, että matto on ainoa huonekalu, jonka päälle voi laittaa muita huonekaluja ja lamppu on ainoa huonekalu, joka voi mennä minkä tahansa muun huonekalun päälle. Tämän huomauksen ansiosta metodi, missä käsitellään päällekkäisyydet ja niiden virheet oli varsin helppo toteuttaa. Lopulta siis usean eri huonekalun luokan korvasi pelkkä luokka "Furniture.scala".

Alkuperäisessä suunnitelmassa luokka "Furniture" oli periytynyt luokasta "Shape", joka sisälsi erilaisia muotoja, joilla oli omat pinta-alat. Lopullisessa toteutuksessa tämä "Shape" luokka jäi kuitenkin toteuttamatta samoin kuin kaikki tämän luokan sisältämät muoto-objektit. Päädyin tähän päätökseen, kun huomasin, että scalalla on oma Shape luokka, joka kattoi kaikki tarvittavat muodot ja niiden ominaisuudet ohjelmaa varten. Ainoa muoto, mikä jäi toteuttamatta, oli puoliympyrä. En kokenut muodon olemassaoloa niin tarpeelliseksi suunnittelu ohjelmassa ja lisäksi päätökseen vaikutti se, ettei kyseistä muotoa löytynyt Scalan "Shape" -luokan sisältämistä muodoista.

Luokka "App" alkuperäisessä suunnitelmassa vastaa suurimmaksi osaksi lopullisessa toteutuksessa luokkaa "Main". Hiiren avulla tehtävät toiminnot erotettiin kuitenkin omaan luokkaan. Tämä johtui täysin siitä, että luokan "Main" koodi alkoi käymään niin pitkäksi, että sitä oli raskasta lukea. Tämän vuoksi "Controller" on oma luokkansa. Tämä muutos paransi huomattavasti koodin luettavuutta ja selkeyttä.

Luokkia, niiden sisältämiä metodeita ja suhteita on kuvattu UML-kaaviossa ^[9], joka löytyy kohdasta "Liitteet".

Algoritmit

Ohjelma käyttää tiettyä algoritmia, kun huonekaluja mitoitetaan kuvan mittasuhteiden mukaiseksi. Tämä on toteutettu kertomalla annetut mitat kertoimella 0,636. Tällöin huonekalut skaalautuvat realistisesti kuvaan.

Esimerkiksi neliskulmaisen huonekalun mittoihin algoritmia on käytetty seuraavasti:

$sideX = \text{annettu mitta koordinaatiston } x \text{ -suuntaiselle sivulle.}$

$sideY = \text{annettu mitta koordinaatiston } y \text{ -suuntaiselle sivulle.}$

$$width = sideX \cdot 0,636 \quad height = sideY \cdot 0,636$$

Näillä lasketuilla tiedoilla on muodostettu "Rectangle(width: Double, height: Double, fill: Paint)" -objekti.

Kun tarkastellaan huonekalujen mahdollisia päällekkäisyyksiä, käytetään algoritmia, joka tarkistaa ensin mikä huonekalu on kyseessä ja tämän jälkeen katsoo, onko huonekalu jonkun tai joidenkin huonekalujen päällä, missä sen ei kuuluisi pystyä olemaan. Lamppu pystyy olemaan kaikkien huonekalujen päällä, maton päällä voi olla mikä tahansa huonekalu, ja TV voi olla maton yläpuolella tai pöydän päällä. Loput huonekalut eivät voi mennä päällekkäin.

Tietorakenteet

Ohjelmassa tiedon varastointiin käytetään Option-olioita sekä Array-kokoelmia. Main.scala -luokan Option-oloihin "possibleObject" sekä "possibleFurniture" säilötään tietoja, mitä ohjelma saa eri vaihtoehtojen nappien painalluksista. Muodostettavan huonekalun muodon pitää sisällään "possibleObject" ja valitun huonekalun nimen tiedon säilyttää "possibleFurniture". Kerätyillä tiedoilla muodostetaan uusi huonekalu.

Valitsin välitiedon keräykseen Option-oliot niiden helppokäyttöisyyden vuoksi. Option-oliot voivat pitää sisällään vain yhtä tietoa kerrallaan, joten eri huonekalujen tiedot eivät mene sekaisin. Huonekalut muodostuvat siis täysin riippuen käyttäjän valinnoista.

Ohjelman tietokokoelmat säilytetään Array-kokoelmissa. Valitsin Arrayt, sillä ne ovat muuttuvatilaisia (mutable) ja helppokäyttöisiä. Ohjelmassa kaikki muodostuvat huonekalut kootaan kokoelmaan "furnitures". Tätä kokoelmaa käytetään, kun tutkitaan, mitkä huonekalut ovat päällekkäin ja mitkä ei. Kokoelma päivittyy aina, kun yksi huonekalu lisätään kuvaan.

Tiedostot ja verkossa oleva tieto

Ohjelma käsittelee kuvatiedostoja, jotka ova JPG tai PNG muodossa. Kuvatiedosto on kuva asunnon pohjapiirroksista. Ohjelmassa käytetyn testikuvan on mitat ovat 1280 x 1240 ja koko on 39 kt. Ohjelma pystyy myös tallentamaan kuvatiedoston PNG-muodossa käyttäjän omiin tiedostoihin.

Testaus

Toteutuksen aikana ohjelmaa testattiin paljon graafisen käyttöliittymän sekä komentorivin kautta. Yleisin tapa, miten ohjelmaa testattiin, oli komentoriville eri välitietojen tulostaminen. Näin tarkistin, että huonekalut, joita ohjelman käytön aikana luodaan muodostuvat oikein. Kun yritin saada huonekaluja liikkumaan oikein käytn testaukseen graafista käyttöliittymää. Lisäksi etsiessäni virheitä koodissa, tulostelin huonekalujen muuttuvia koordinaatteja ja etsin sitä kautta virheitä koodissani. Graafisen käyttöliittymän kautta tarkistettiin myös kaikkien pienien ponnahtusikkunoiden toiminta. Katsottiin, että nämä niin kutsutut ”Alertit” ponnahtavat vain niille kuuluvilla ajoilla.

Koodin testauksessa käytettiin korvausolioita. Kun luotiin luokkaa ”Furniture” luokka oli vielä keskeneräinen, kun sitä käytettiin käyttöliittymässä. Tällöin siis käytettiin ”Furniture”-luokan Stub-olioita.

Ohjelma läpäisi kaikki tehdyt testit lopulta eikä olennaisia aukkoja jäänyt. Projektisuunnitelmassa esitetystä versiosta toteutunut testaus erosi hieman. Graafista käyttöliittymää testattiin ennen kuin huonekalujen luokka oli valmis. Graafisen käyttöliittymän ja komentorivin avulla toteutettu testaus oli myös suuremmassa osassa kuin suunnitelmassa mainittu Stub- ja Dummy-testaus.

Ohjelman tunnetut puutteet ja viat

Ohjelman suurin vika liittyy huonekalujen päällekkäisyyksiin. Voit asettaa huonekaluja kuvaan päällekkäin vain siinä järjestyksessä, kun asetat niitä kuvaan. Esimerkiksi, jos lisäät kuvaan ensin pöydän ja sen jälkeen maton, viimeisenä lisätty huonekalu tulee aina piirtymään sitä edeltävän huonekalun päälle. Tässä tapauksessa, vaikka siirtäisit pöytää maton päälle, ohjelma piirtää tapahtuman niin, että pöytä olisi asettumassa maton alle.

Tähän ohjelman vikaan liittyy myös toinen mahdollisesti käyttäjää hämmentävä virhe. Kuvitellaan tilanne, missä tällä kertaa käyttäjä olisikin lisännyt ensin maton kuvaan ja tämän jälkeen pöydän. Nyt käyttäjän siirtäessä mattoa pöydän kanssa samaan alueeseen, matto piirtyy kuvassa pöydän alle. Tämä näyttää käyttäjän silmiin oikealta, mutta ohjelma kuitenkin tulkitsee tapauksen niin, että mattoa olisi siirretty nyt pöydän päälle.

Huonekalut piirretään ohjelmassa alueelle nimeltä "floorPlanBox". Kaikki käytön aikana luodut huonekalut on lisätty alueen "lapsiksi", eli kokoelmaan, jossa on kirjattuna kaikki alueen sisältämät elementit. Päällekkäisyys alueessa määräytyy sen mukaan, missä kohtaa kokoelmassa elementti sijaitsee. Viimeisin kokoelman elementti on päällimmäisenä ja ensimmäinen elementti alimmaisena. Todennäköisesti huonekalujen piirtoon kuuluvan päällekkäisyys vian olisi siis ollut mahdollista korjata, muokkaamalla tämän kokoelman elementtien paikkaa. Kun käyttäjä klikkaa huonekalua, tämä huonekalu olisi voinut mennä viimeiseksi elementiksi kokoelmassa, jolloin se olisi piirtynyt liikuttaessa kaikkien muiden elementtien päälle.

Toinen ohjelman puutteellisuus liittyy valkoisten huonekalujen piirtoon. Kun ohjelmaan piirretään valkoinen huonekalu, tämä sulautuu valkoiseen taustaan niin hyvin, ettei käyttäjä välttämättä löydä sitä. Valkoisille huonekaluille ei siis ole piirretty ääri viivoja, joiden avulla ne erottaisivat valkoisesta taustasta. Kyseiset huonekalut kuitenkin näkyvät hyvin vasten värillisiä huonekaluja sekä tummia seiniä. Tämän virheen korjaaminen jäi toteuttamatta, ajan puutteen sekä vian epäolennaisuuden vuoksi.

Kun käyttäjä lisää huonekalun, ei sitä voi kuvasta poistaa. Tämä toiminto puuttuu ohjelmasta, jonka vuoksi elementtien poistaminen kuvasta on mahdotonta. Toiminnon olisi ollut mahdollista sisältää esimerkiksi hiiren toimintona, jossa hiiren oikeaa painamalla voit kääntämisen lisäksi valita poistaa huonekalun. Tällöin huonekalu olisi poistunut alueen "floorPlanBox" "children"-kokoelmasta, jolloin samalla huonekalu poistuisi kuvan näkymästä.

Käyttäjän lisäämä kuva pohjapiirustuksesta skaalautuu näytön keskelle niin kuin sen kuuluukin. Kuitenkin, jos käyttäjä haluaa ohjelman olevan koko tietokoneen koko näytön kokoisena, kuva ei skaalaudu keskelle ruutua, vaan jää paikoilleen sen alkuperäiselle paikalle. Tällöin kuva ei siis ole keskellä ruutua, jolloin se ei täytä tehtävässä annettua kriteeriä.

Huonekalujen realistinen skaalautuvuus kuvaan on muodostettu kertomalla mitat kertoimella 0,636. Tämä kerroin on valittu testaamalla siten, että huonekalun mittaa on verrattu testikuvassa annetun seinän pituuteen. Valittu kerroin sopii tämän kuvan mittasuhteisiin, mutta jos käyttäjän valitsema pohjapiirustus sisältäisi eri mittasuhteet, mitä testikuvassa on, skaalautuvuus ei välttämättä olisi täysin realistinen.

3 parasta ja 3 heikointa kohtaa

Projektin parhaat kohdat ovat mielestäni tiedostojen luku, käyttöliittymän helppo ja selkeä käytettävyys sekä huonekaluille annettavien värimahdollisuuksien suuruus.

Menusta "File", voi käyttäjä ladata oman kuvan sekä tallentaa tehdyn suunnitelman. Tämä on mielestäni hyvin toimiva sekä hyvä ominaisuus. Napit selkeästi kertovat, mitä ne tekevät ja käyttäjä on helppo käyttää toimintoja.

Käyttöliittymässä kerrotaan selkeästi, mikä nappi tekee mitäkin. Huonekalut ovat selkeästi merkitty, ja käyttäjän valitessaan huonekalun ohjelma kertoo tarkasti, mitä tehdä seuraavaksi. Käyttäjä ei voi tehdä virheitä mittoja antaessa, sillä ohjelma ei anna mennä eteenpäin, jos kaikkia mittoja ei ole annettu tai ne ovat annettu väärin. Huonekalujen päällekkäisyydestä tulee selkeä viesti, jonka jälkeen käyttäjä voi jatkaa suunnittelutyötä normaalisti eteenpäin. Kun käyttäjä painaa hiiren oikeaa näppäintä huonekalun päällä, tulee esiin nappi, joka kysyy, haluaako käyttäjä kääntää huonekalua. Huonekalu ei siis automaattisesti käänny mihinkään ilman, että käyttäjältä asiaa on kysytty.

Erilaisten sisutussuunnitelmien määrä on loputon eri väri vaihtoehtojen ansiosta. Voit itse valita värin laajasta väripaletista ja tämän lisäksi, jos paletin värit eivät jostain syystä miellytä, voi käyttäjä luoda itse oman värin.

Vaikka ohjelman vahvuudet ovatkin ohjelman selkeydessä, myös ohjelman suurimmat heikkoudet löytyvät käyttöliittymän toiminnasta.

Suurimmat heikkoudet ovat huonekalujen pysyvyys sekä huonekalujen lisäämisen järjestyksen merkitys. Huonekaluja ei voi poistaa, joka tarkoittaa sitä, että suunnittelijalla ei ole mahdollisuutta tehdä virheitä, kun hän lisää huonekaluja kuvaan. Tämä on ongelmallista, sillä käyttäjä voi vahingossa antaa väärät mitat tai väärän värin. Lisäksi poisto mahdollisuus olisi hyvä olla, sillä huonekalujen lisäämisjärjestyksellä on suuri merkitys suunnitelman piirron kannalta. Jos käyttäjä haluaa maton jonkun huonekalun alle, on se lisättävä ennen huonekalua, jonka käyttäjä lopulta siirtää maton päälle. Tämä on epäkäytännöllistä ohjelman käytettävyyden näkökulmasta ja lisäksi kyseinen viallisuus lisää suunnittelussa mahdollisesti tapahtuvien virheiden määrää.

Huonekalut piirretään ohjelmassa alueelle nimeltä "floorPlanBox". Kaikki käytön aikana luodut huonekalut on lisätty alueen "lapsiksi", eli kokoelmaan, jossa on kirjattuna kaikki alueen sisältämät elementit. Päällekkäisyys alueessa määräytyy sen mukaan, missä kohtaa kokoelmassa elementti sijaitsee. Viimeisin kokoelman elementti on päällimmäisenä ja ensimmäinen elementti alimmaisena. Todennäköisesti huonekalujen piirtoon kuuluvan päällekkäisyys vian olisi siis ollut mahdollista korjata, muokkaamalla tämän kokoelman elementtien paikkaa. Kun käyttäjä klikkaa huonekalua, tämä huonekalu olisi voinut mennä viimeiseksi elementiksi kokoelmassa, jolloin se olisi piirtynyt liikuttaessa kaikkien muiden elementtien päälle.

Huonekalujen poiston olisi ollut mahdollista sisältää esimerkiksi hiiren toimintona, jossa hiiren oikeaa painamalla voit kääntämisen lisäksi valita poistaa huonekalun. Tällöin huonekalu olisi poistunut alueen "floorPlanBox" "children"-kokoelmasta, jolloin samalla huonekalu poistuisi kuvan näkymästä.

Lisäksi yksi mainittava heikkous ohjelmassa on huonekalujen skaalautuvuus. Huonekalut ovat ohjelmoitu skaalautumaan mallikuvan mukaisiin mittasuhteisiin, mutta jos käyttäjä haluaisi valita oman kuva pohjapiirustuksesta, ei skaalautuvuus välttämättä pysyisi yhtä realistisena.

Poikkeamat suunnitelmasta, toteutunut työjärjestys ja aikataulu

Ensimmäisellä kahdella viikolla projektin toteutus aloitettiin hahmottamalla luokkia sekä niiden metodeita. Tämän jälkeen siirryttiin muodostamaan käyttöliittymää niin, että sen pohja muodostettiin ja siihen lisättiin huonekaluille kuuluvia nappeja. Napit saatiin toimimaan niin, että ne muodostivat haluttuja ”Alert”-kysymys laatikoita.

Kolmannelle viikotapaamiselle valmiina oli kuvatiedoston luku ja sen sijoittaminen käyttöliittymään. Lisäksi kuvan päälle saatiin muodostumaan eri huonekalut.

Neljännelle viikotapaamiselle huonekalujen luokka ”Furniture” oli korjattu sen sisältämistä virheistä ja huonekalujen liikuttamista varten oleva luokka ”DragController” oli hahmoteltu. Luokka ei tässä vaiheessa kuitenkaan toiminut halutulla tavalla, sillä huonekalujen liikuttaminen osoittautuikin vaikeammaksi tehtäväksi, mitä luulin.

Viimeisellä tapaamisella huonekalut liikkuvat halutulla tavalla ja päällekkäisyyksien virheet tulivat näkyviin. Suunnitelma saatiin tallentumaan myös kuvana tiedostoihin. Lisäksi koodia oli siistitty ja uusi luokka ”sizeSelect” oli muodostettu. Ylimääräisiksi huomautetut luokat oli myös poistettu. Ohjelma oli siis lähes valmis viimeisellä viikolla lukuun ottamatta skaalauksia sekä huonekalujen kääntämistä. Nämä toiminnot kuitenkin toteutettiin viimeisen viikon aikana.

Ensimmäiset kaksi viikkoa olivat projektin toteutussuunnitelman mukaiset, mutta tämän jälkeen toteutuksen aikataulu ja toimintojen toteutusjärjestys poikkeaa huomattavasti suunnitelmasta. Suunnitelmassa oli ajateltu, että luokat ”Shape” ja ”Furniture” olisivat muodostettu ennen käyttöliittymän toteutusta. Näin ei kuitenkaan tehty, sillä luokan ”Furniture” ominaisuuksia pystyttiin kunnolla vasta testaamaan, kun käyttöliittymä oli siihen valmis.

Suunnitelmasta poiketen keskityttiin ennen muita luokkia ohjelman käyttöliittymään ja sen toimintoihin. Opin projektin aikana, että on hyvä aloittaa ensin käyttöliittymästä ja vasta sen jälkeen kunnolla miettiä luokkien toteutuksia. Näin luokkia voi testata hyvin käyttöliittymän avulla.

Projektin suunnitelmassa oli ajateltu, että projekti olisi debuggausta vaille valmis viikkojen 25.3.-8.4. jälkeen. Näin ei kuitenkaan ollut, sillä huonekalujen oikeanlainen liikuttaminen vei projektin teossa suunnattomasti aikaa. Projektiin ei myöskään ehditty suunnitella lisäyksiä vaikeustasolta ”vaikea”, sillä keskivaikean tehtävän kriteerit olivat vielä toteuttamatta, silloin kun suunnitelman mukaan lisäyksiä olisi voinut miettiä.

Suunnitellut työtunnit olivat suurin piirtein oikeaa suuruusluokkaa, mutta usein työmäärä ylitti aika-arvion. Suunnitelmasta myöhästyminen johtui suurimmaksi osaksi siitä, etten tiennyt tiettyjen toimintojen vaativuudesta. Kamppailin pitkään huonekalujen lisäämisen ja niiden liikuttamisen kanssa, ja lisäksi tiedoston tallentamisen kanssa esiintyi ongelmia.

Opin kuitenkin prosessin aikana, että vaikka jossain projektin osien toteutuksessa voi kestää oma aikansa ei tämä tarkoita, että lopullinen toteutuksen valmistuminen poikkeaa aikataulusta. Osa toteutuksista vei enemmän aikaa, mitä aluksi oli ajateltu, mutta osa projektin toteutuksista taas vei paljon vähemmän aikaa, mitä olisin luullut. Loppua kohden koodin toiminta ja ongelmien ratkominen olikin jo niin tuttua, että jumiin ei juurikaan jääty uusia toimintoja luodessa. Seuraavaa projektia varten tiedän nyt ainakin, että hiiren toimintojen luominen voi viedä aikaa.

Kokonaisarvio lopputuloksesta

Olen hyvin tyytyväinen projektin lopputulokseen ja ylpeä itsestäni, että jaksoin yrittää loppuun asti. Välillä tiettyjen bugien kohdalla tuntui, että projekti ei ehkä koskaan tule toimimaan halutulla tavalla palautuspäivään mennessä. En kuitenkaan luovuttanut, vaan yritin keksiä jatkuvasti uusia toteutusratkaisuja ja lopulta aina tulosta syntyi.

Hyvää ohjelmassa on sen helppo käytettävyys. Käyttöliittymässä kerrotaan selkeästi, mikä nappi tekee mitäkin. Huonekalut ovat selkeästi merkitty, ja käyttäjän valitessaan huonekalun ohjelma kertoo tarkasti, mitä tehdä seuraavaksi. Käyttäjä ei voi tehdä virheitä mittoja antaessa, sillä ohjelma ei anna mennä eteenpäin, jos kaikkia mittoja ei ole annettu tai ne ovat annettu väärin. Huonekalujen päällekkäisyydestä tulee selkeä viesti, jonka jälkeen käyttäjä voi jatkaa suunnittelutyötä normaalisti eteenpäin. Kun käyttäjä painaa hiiren oikeaa näppäintä huonekalun päällä, tulee esiin nappi, joka kysyy, haluaako käyttäjä kääntää huonekalua. Huonekalu ei siis automaattisesti käännä mihinkään ilman, että käyttäjältä asiaa on kysytty.

Lisäksi mahdollisia sisustussuunnittelu mahdollisuuksia on äärettömän paljon. Huonekaluja voi siirtää ja kääntää niin paljon kuin haluaa, ja ne saa asetettua kuvaan siten kuin käyttäjä itse haluaa. Lisäksi väri vaihtoja on mahdollista paljon. Käyttöliittymän ulkomuoto on myös kaunis ja yksinkertainen.

Ohjelman laatu on hyvä, mutta virheitä myös löytyy. Suurin viallisuus liittyy huonekalujen päällekkäisyyksiin. Voit asettaa huonekaluja kuvaan päällekkäin vain siinä järjestyksessä, kun asetat niitä kuvaan. Esimerkiksi, jos lisäät kuvaan ensin pöydän ja sen jälkeen maton, viimeisenä lisätty huonekalu tulee aina piirtymään sitä edeltävän huonekalun päälle. Tässä tapauksessa, vaikka siirtäisit pöytää maton päälle, ohjelma piirtää tapahtuman niin, että pöytä olisi asettumassa maton alle.

Huonekalut piirretään ohjelmassa alueelle nimeltä "floorPlanBox". Kaikki käytön aikana luodut huonekalut on lisätty alueen "lapsiksi", eli kokoelmaan, jossa on kirjattuna kaikki alueen sisältämät elementit. Päällekkäisyys alueessa määräytyy sen mukaan, missä kohtaa kokoelmassa elementti sijaitsee. Viimeisin kokoelman elementti on päällimmäisenä ja ensimmäinen elementti alimmaisena. Todennäköisesti huonekalujen piirtoon kuuluvan päällekkäisyys vian olisi tulevaisuudessa mahdollista korjata, muokkaamalla tämän kokoelman elementtien paikkaa. Kun käyttäjä klikkaa huonekalua, tämä huonekalu voisi

mennä viimeiseksi elementiksi kokoelmassa, jolloin se piirtyy liikutettaessa kaikkien muiden elementtien päälle. Tämä jäi kuitenkin toteuttamatta ajan loppumisen vuoksi.

Toinen puute ohjelmassa on huonekalujen poistamisen mahdollisuuden puute. Huonekaluja ei voi poistaa, joka tarkoittaa sitä, että suunnittelijalla ei ole mahdollisuutta tehdä virheitä, kun hän lisää huonekaluja kuvaan. Tämä on ongelmallista, sillä käyttäjä voi vahingossa antaa väärät mitat tai väärän värin. Lisäksi poisto mahdollisuus olisi hyvä olla, sillä huonekalujen lisäämisjärjestyksellä on suuri merkitys suunnitelman piirron kannalta. Jos käyttäjä haluaa maton jonkun huonekalun alle, on se lisättävä ennen huonekalua, jonka käyttäjä lopulta siirtää maton päälle. Tämä on epäkäytännöllistä ohjelman käytettävyyden näkökulmasta ja lisäksi kyseinen viallisuus lisää suunnittelussa mahdollisesti tapahtuvien virheiden määrää.

Huonekalujen poiston olisi ollut mahdollista sisältää esimerkiksi hiiren toimintona, jossa hiiren oikeaa painamalla voit kääntämisen lisäksi valita poistaa huonekalun. Tällöin huonekalu olisi poistunut alueen "floorPlanBox" "children"-kokoelmasta, jolloin samalla huonekalu poistuisi kuvan näkymästä. Kyseisen napin olisi voinut lisätä MenuItem:inä Control-luokan metodiin "createHandlers". Tämä toiminto kuitenkin jäi toteuttamatta, sillä sitä ei vaadittu tehtävän annossa. Lisäksi aika ei olisi riittänyt toteutukseen.

Ohjelman rakenne on hyvä pohja parannuksille sekä muutoksien tekemiseen. Luokkajakoa olisi voinut kuitenkin laajentaa paremmin. Tällä hetkellä käyttöliittymän luokassa "Main" on sisällytettyä paljon koodia, jota olisi voinut jakaa luokiksi tai objekteiksi. Jos esimerkiksi käyttöliittymän alueet olisi jaettu omiksi objekteiksi, olisi helpompi toteuttaa huonekalujen liikuttamiseen sekä poistoon liittyvät metodit. Tällä hetkellä "floorPlanBox" – objektin sisältöön ei pääse käsiksi, sillä tämä olio on luotu metodin "start()" sisälle. Muuttamalla tätä olisi ehkä mahdollista tehdä tarvittavat muutokset.

Jos aloittaisin projektin nyt alusta en sisällyttäisi niin paljon koodia luokkaan "Main". Tekisin enemmän luokkia metodeille sekä käyttöliittymän sisällöille, jotta koodi olisi siistimmän näköinen. Samalla käyttöliittymän elementteihin pääsisi ehkä helpommin käsiksi. En myöskään suunnittelisi kaikille huonekaluille omaa case Classia, vaan sisällyttäisin suoraan kaikki samaan luokkaan "Furniture" niin kuin lopullisessa toteutuksessa on nyt tehty. En myöskään aloittaisi tekemään luokkaa "Shape" sillä scalaafx:llä on jo saman tyyppinen valmis luokka kirjastossaan. Lisäksi tekisin enemmän töitä alussa. Vaikka ensimmäisillä viikoilla käyttöliittymän teko olikin helppoa, lopussa syntyi vaikeuksia tiettyjen toimintojen sisällyttämisen kanssa. Näihin toteutuksiin olisi voinut varata enemmän aikaa, jolloin ohjelmaan olisi ehtinyt miettiä paremmin bugien korjauksia sekä laajennuksia.

Viitteet ja muualta otettu koodi

Koodia tehdessä isona apuna on ollut scalaFX:n ja javaFX:n kirjastot. Sieltä erilaisten luokkien kuten "Shape" ja eri "Pane"-luokkien sisältämiä metodeita on voinut tarkistaa. Näin olen voinut selvittää, jos luokalla on sopivia metodeita tiettyyn toimintoon, mitä olen luomassa. Nettisivut, joita olen käyttänyt apuna ovat:

javadoc.io sekä stackoverflow.com

Kuitenkaan kummastakaan nettisivusta koodia ei ole kopioitu, vaan mahdollisia metodeita ja luokkia on vain haettu. Lisäksi alertteihin ja dialogeihin esimerkkejä sai sivustolta:

https://www.scalafx.org/docs/dialogs_and_alerts/

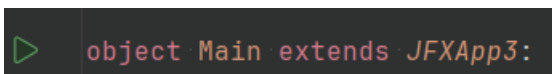
Huonekalujen liikuttamiseen sain vinkkiä opiskelijalta Teea Ahola. Sain Teealta inspiraation muuttujiin previousX ja previousY. Näin Teealta, että hänen koodinsa sisälsi kyseiset muuttujat, ja sain siitä inspiraatiota lisätä samanlaiset omaan koodiini. Kuitenkin se, missä ja miten käytän kyseisiä muuttujia, on täysin itse mietittyä. Teea lähetti lisäksi linkin blogi postaukseen, jossa oli tutoriaali toimintoon, millä voi liikuttaa elementtejä hiirellä. Tämä tutoriaali oli kuitenkin kirjoitettu javalla eikä scalalla. Tästä blogipostauksesta sain apua koodin pohjan tekemiseen ja siihen, mitä elementtejä koodin olisi hyvä sisältää. Metodin lopullinen toiminta on kuitenkin itse mietittyä, sillä saaduilla vinkeillä ei kuitenkaan oikein toimivaa toimintoa saatu aikaan.

Linkki Blogiin: <https://edencoding.com/drag-shapes-javafx/>

Debuggauksessa olen satunnaisesti käyttänyt ChatGPT:tä apuna. Tekoälyä käyttäessä olen kopioinut oman koodini ChatGPT:lle, ja kysynyt missä virhe olisi tai miten tekoäly muuttaisi koodia. Usein tästä keinosta ei ole ollut hyötyä, mutta joskus tekoälystä on ollut apua kysymyksiini. Isoin hyöty tekoälystä oli, kun käänsin javalla tehtyä koodia scalaksi. Tällöinkään suoraan vastausta ei tekoälyltä saanut, mutta tiettyjä piirteitä oppi javan koodauskielestä. Opin tekoälyn avulla, miten javalla merkitään eri metodit ja muuttujat. Tämä auttoi huomattavasti hahmottamaan tutoriaalia, jossa käytettiin javaa, kun tein hiiren toimintoja. Kuitenkaan missään vaiheessa koodia ei suoraan ole kopioitu ChatGPT:ltä.

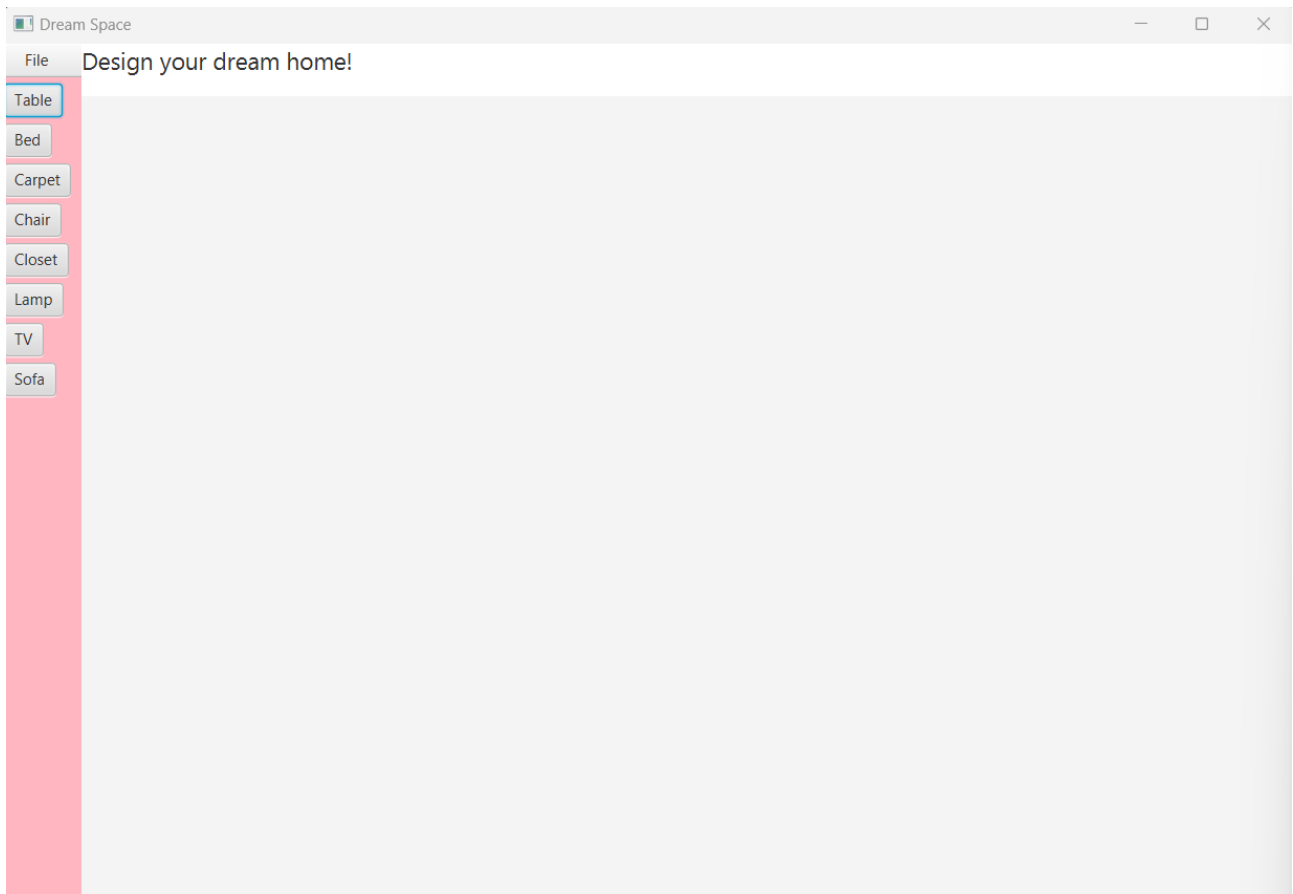
Liitteet:

[1]:

A screenshot of a code editor showing the line `object Main extends JFXApp3:` in a dark-themed interface. A green play button icon is visible on the left side of the line.

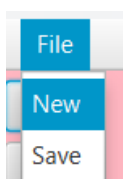
Kuva 1: Ohjelman käynnistys

[2]:



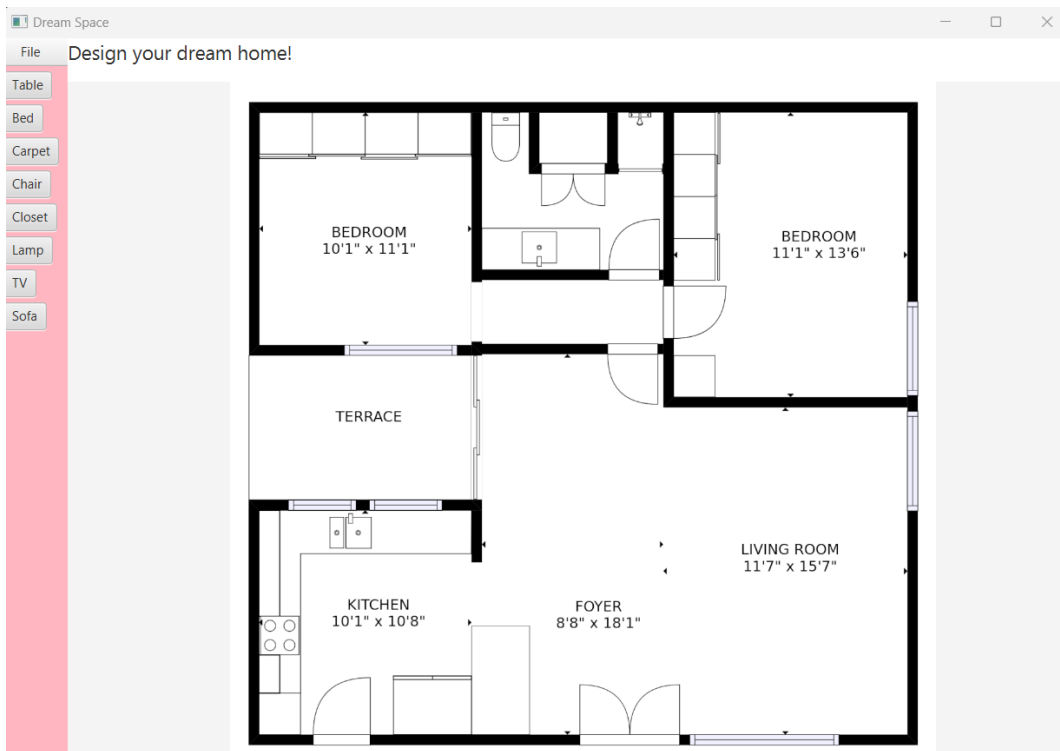
Kuva 2: Käyttöliittymän alkunäkymä

[3]:



Kuva 3: File-menu. Kun käyttäjä haluaa lisätä kuvan, pitää hänen painaa File-menun nappia "New". Jos käyttäjä haluaa tallentaa tehdyn suunnitelman kuvana, pitää hänen painaa File-menun nappia "Save".

[4]:



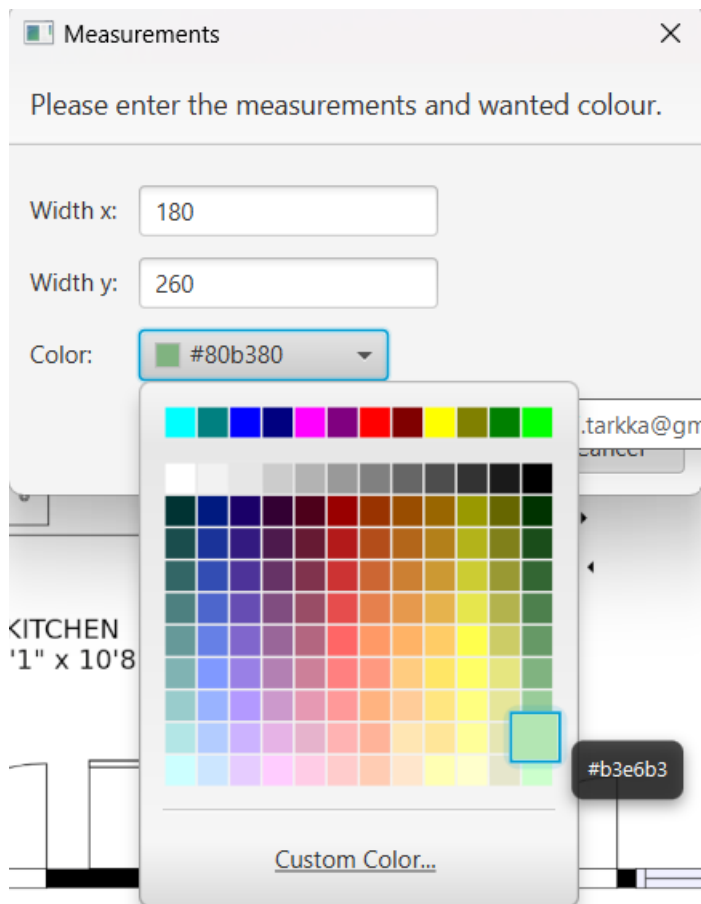
Kuva 4: Käyttöliittymä

[5]:



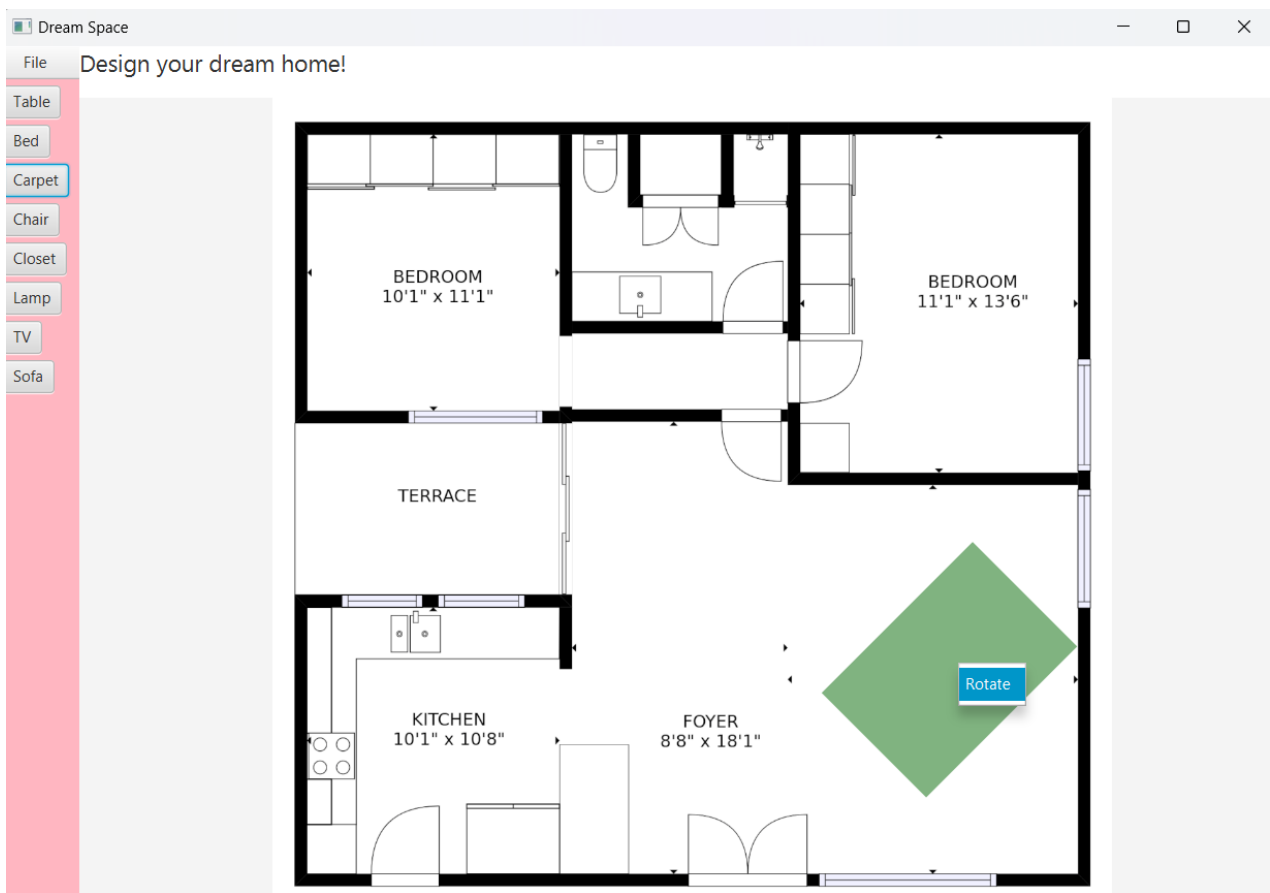
Kuva 5: Ikkuna, joka kysyy käyttäjältä hänen valitsemalleen huonekalulle muotoa.

[6]:



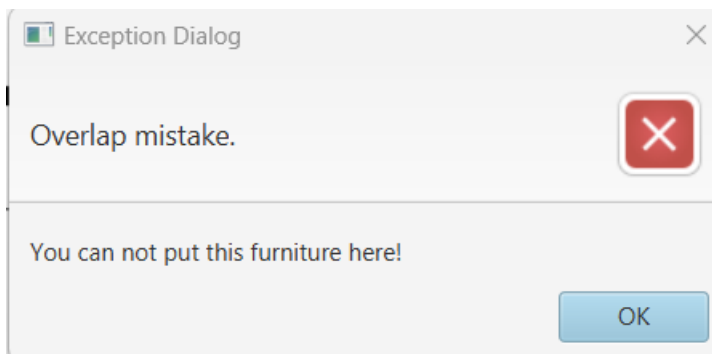
Kuva 6: Ikkuna, joka kysyy käyttäjältä hänen valitsemalleen huonekalulle mitat ja värin.

[7]:



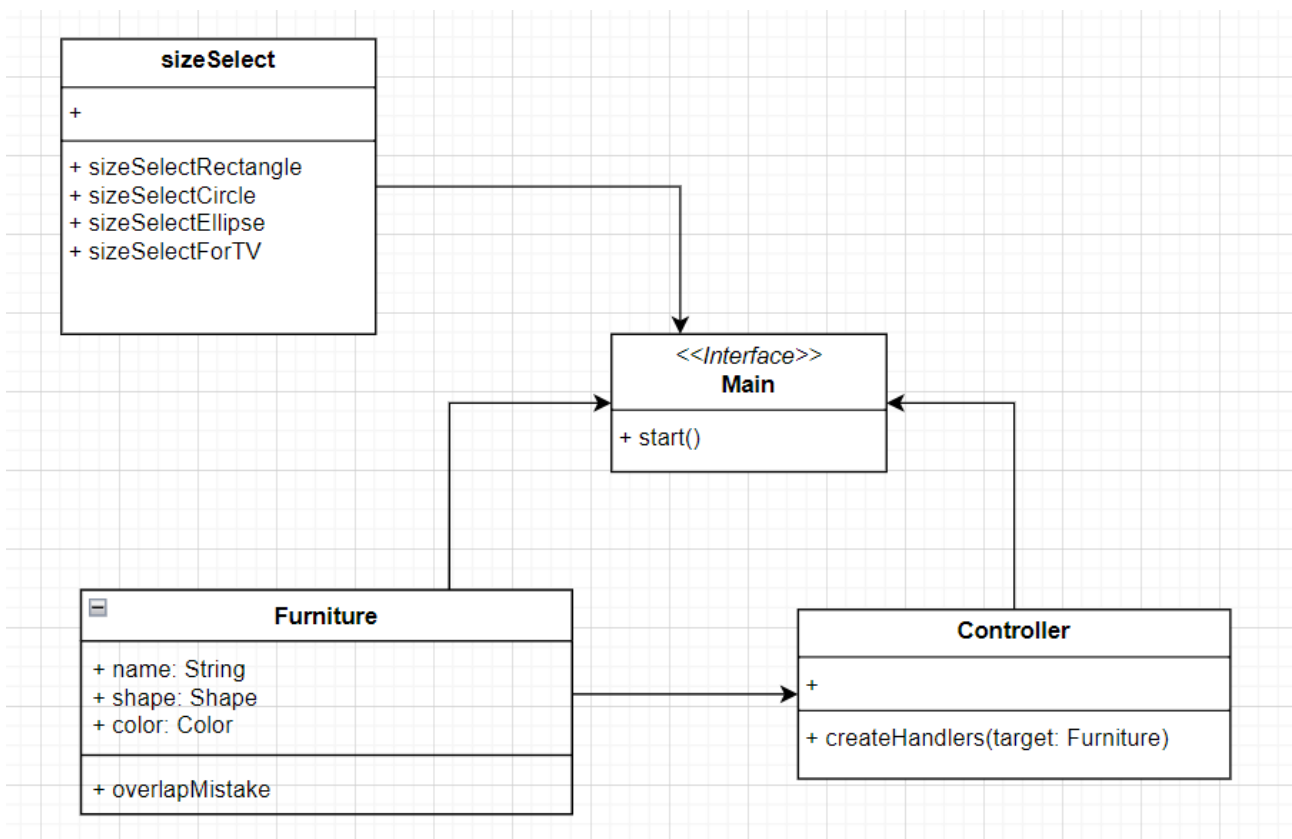
Kuva 7: Kun käyttäjä painaa huonekalun päällä hiiren vasenta nappia tulee esille nappi "Rotate". Tätä nappia painamalla huonekalu kääntyy 45 astetta myötäpäivään.

[8]:



Kuva 8: Jos käyttäjä siirtää huonekalun toisen huonekalun päälle, mihin sitä ei olisi mahdollista siirtää todellisessa elämässä, syntyy näytölle ilmoitus ”Overlap mistake!”.

[9]:



Kuva 9: UML-kaavio

