OULUN AMMATTIKORKEAKOULU

Akseli Lahtinen

GroupBuilder Oy

Päiväkirjamuotoinen opinnäytetyö

GroupBuilder Oy

Päiväkirjamuotoinen opinnäytetyö

Akseli Lahtinen
Opinnäytetyö
Kevät 2019
Tradenomi
Oulun ammattikorkeakoulu

TIIVISTELMÄ

Oulun ammattikorkeakoulu
Tradenomi, Web-ohjelmointi

Tekijä(t): Akseli Lahtinen
Opinnäytetyön nimi: GroupBuilder Oy, päiväkirjamuotoinen opinnäytetyö
Työn ohjaaja: Pekka Ojala
Työn valmistumislukukausi ja-vuosi: Syksy 2019 Sivumäärä: 53

Päiväkirjamuotoinen opinnäytetyöni käsittelee Unity-ohjelmoijan työtehtäviä GroupBuilder Oy:ssä. GroupBuilder Oy on huoneistojen sekä talojen esittelyohjelmistoon keskittyvä ohjelmointiyritys. Esittelyohjelmistoa käytetään 3D-mallien esittämiseen.

Työni tavoitteena oli korjata ohjelmistovirheitä, sekä kehittää uusia ominaisuuksia ohjelmistoon, kuten uusi käyttöliittymä. Olen vastuussa käyttöliittymän kehittämisestä, mutta en suunnittelusta. Käytän työssäni C#-ohjelmointikieltä, sekä Unity-pelimoottorin editoria.

Opinnäytetyön tietoperustana on käytetty laajalti Unity-pelimoottorin omaa dokumentaatiota, Unity Answers -palvelua, sekä Unity-keskustelupalstaa. Opiskellessani C#-ohjelmointikieltä, käytän tietoperustanani myös Microsoftin C#-dokumentaatiota.

Opinnäytetyön aikana sain valmisteltua rungon uuden käyttöliittymän toiminnallisuudelle, jonka työstämistä jatkoimme päiväkirjamerkintöjen jälkeen. Viimeisen viikon aikana käyn myös läpi, kuinka lisäämme API-päätepisteitä, joiden avulla ohjelmisto voi keskustella tietokantojen kanssa. Toiminnallisuuden valmistelun jälkeen, olimme valmiita suunnittelemaan käyttöliittymälle parempaa ulkonäköä.

Asiasanat: ohjelmointivirheet, ohjelmointikielet, Unity, käyttöliittymät

ABSTRACT

Oulu University of Applied Sciences
BBA, web-programming

Author(s): Akseli Lahtinen
Title of thesis: GroupBuilder Oy, diary thesis
Supervisor(s): Pekka Ojala
Term and year when the thesis was submitted: Fall 2019 Number of pages: 53

My thesis handles the work of Unity-programmer at GroupBuilder Oy. GroupBuilder Oy is a software company that works on software used for showcasing 3D-models of apartments and buildings.

The goal of my work was to fix software errors, and develop new features to the software, such as a new user interface. I am responsible of developing the user interface, but not designing it. I use C#-programming language and Unity-engine's editor.

For the knowledge base of this thesis, I use Unity's own documentation, Unity Answers -service and Unity Forums. When studying C#-programming language, I use Microsoft's C#-documentation.

During this thesis, I finished the basic framework of the new user interface, and we kept working on it after the thesis. During last entries, I also go through how we create new API-endpoints, which the software uses to communicate with databases. After finishing the framework, we were ready to design better look for the user interface.

Keywords: software errors, programming languages, Unity, user interfaces

SISÄLLYS

1 JOHDANTO

Päiväkirjamuotoinen opinnäytetyö alkoi 4.2.2019 ja päättyy 26.4.2019. Tämän opinnäytetyön raportointi tapahtuu päivittäisillä työkuvauksilla sekä viikoittaisella analyysilla.

Groupbuilder Oy on ohjelmistoyritys, joka suunnittelee ja toteuttaa ohjelmistoja, joiden avulla voidaan käsitellä ja esitellä asuntojen sekä rakennusten 3D-malleja. Näitä ohjelmistoja käyttävät monet eri rakennusyritykset myydessään asuntoja. Ohjelmiston avulla voidaan muuttaa minkä tahansa rakennuksen pohjapiirustus 3D-malliksi. Yrityksen toimisto sijaitsee Oulun keskustassa, mutta yrityksellä on myös työntekijöitä ympäri maailmaa.

Työtehtävinäni on suunnitella ja toteuttaa asunnon esittelyohjelmistoon eri ominaisuuksia, kuten käyttöliittymä, Virtual Reality -tuki sekä virheenkorjauksia. Nämä työtehtävät vaativat Unity-pelimoottorin osaamista, C#-ohjelmointikielen ymmärrystä, sekä ohjelmistosuunnittelun ja toteutuksen perustaitoja.

Lista keskeisistä ammattikäsitteistä:

- Asset Bundle: Datapaketti, joita käytetään yleensä datan helppoon jakamiseen netin yli. Sisältää mm. 3D-malleja, materiaaleja ja tekstuureja.
- C#, C-sharp: Ohjelmointikieli, jota käytetään Unity-pelimoottorissa.
- Dot product: Laskentakaava, jonka avulla saadaan selville, onko tietty piste peliobjektin edessä tai takana.
- Frustum: Katkaistu kartio.
- Kääntäminen, rakentaminen: Ohjelmiston tai Asset Bundlen kääntäminen muotoon, jonka voi jakaa.
- Peliobjekti: Unityssä käytettävä mikä tahansa objekti/asia, joka koostuu eri Unity-komponenteista.
- Unity: pelimoottori, editori, jolla sovellusta tehdään.
- Unity-komponentti: Komponentti, jonka avulla voidaan esim. toistaa ääntä, lisätä 3D-malli, näyttää kuvaa, lisätä painikkeita yms. ominaisuuksia.
- Virtual Reality: Virtuaalitodellisuus, lyhennettynä VR.

2 NYKYTILANTEEN KUVAUS

Työskentelen GBuilder 3D-sovelluksen parissa, jonka avulla käyttäjälle voidaan esitellä erilaisia asuntoja. Sovellusta käytetään joko GBuilder-internetsivuston kautta (WebGL), mobiililaitteilla (Android tai iOS), tai VR-laitteella (Oculus GO, GearVR).

Sovelluksessa näytetään valittu asunto 3D-muodossa, jota käyttäjä voi tarkastella kääntelemällä kameraa 3D-maailmassa. Käyttäjä voi myös aktivoida kävelytilan, jonka avulla käyttäjä voi kävellä asunnon sisällä. Tämän lisäksi käyttäjä voi käyttää sovellusta asunnon sisustamiseen, joko vaihtamalla materiaaleja, kuten seinien värejä, lattian ja pöytien pintoja jne. sekä lisäämällä asuntoon huonekaluja.

Työtehtäviäni ovat muun muassa käyttöliittymän suunnittelu ja toteutus, ohjelmiston testaus, ohjelmistovirheiden etsiminen ja korjaus, Virtual Reality -version kehitys ja Asset Bundle -datapakettien rakentaminen. Kaikkiin työtehtäviin vaaditaan tietoa C#-kielestä, sekä Unity-pelimoottorin käytöstä. Käyttöliittymän suunnittelussa ja toteutuksessa pyrin tekemään käyttöliittymästä mahdollisimman helppokäyttöisen peruskäyttäjälle. Tämä vaatii tietotaitoa Unityn UI-elementtien käytöstä, jota olen hankkinut tehtävieni aikana.

Ohjelmistovirheiden etsimisessä ja korjauksessa tutkin ohjelmiston koodia ja etsin mahdollisia ongelmatilanteita. Testaan myös ohjelmistoa yrittämällä “rikkoa” sitä: Käytän painikkeita ns. väärässä järjestyksessä, yritän tehdä jotain mitä ohjelmistoa ei ole suunniteltu tekemään jne. Jos virhetapauksia ilmenee, yritän paikantaa mistä ne ovat saaneet alkunsa ja pyrin korjaamaan ne. Jos en tiedä, onko kyseessä virhetapaus, kysyn työtovereiltani mitä mieltä he ovat asiasta.

Ohjelmiston Virtual Reality -version kehityksessä suunnittelen ja toteutan ohjelmistoa toimimaan Oculus Go -virtuaalilaseilla. VR-kehitys vaatii laitteiston tuntemusta, sekä on osattava ajatella perinteisen käyttöliittymän kääntämistä VR-laitteistolle sopivaksi. Olen oppinut VR-laitteiston käyttöä työtehtävieni aikana.

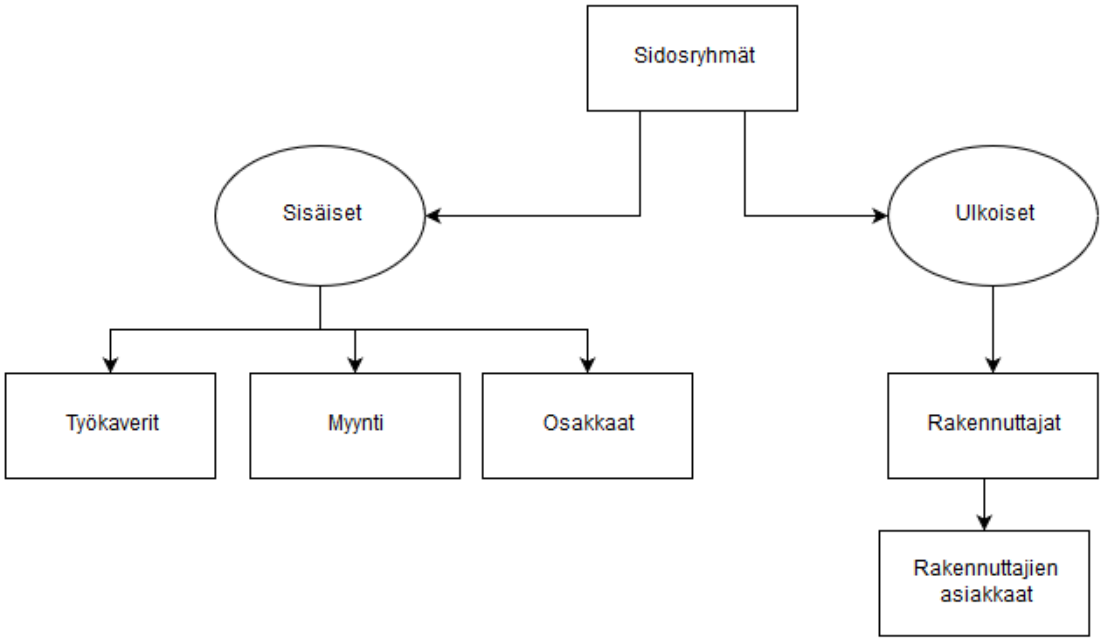
Asset Bundle -kääntäminen tarkoittaa, että ajan läpi muutamia eri skriptejä, joiden avulla luodaan Asset Bundleja. Näitä Asset Bundleja käytetään ohjelmistossa, kun dataa siirretään internetin yli. Tässä tarvitsin aluksi työtoverini apua ja ohjeita.

Olen vielä aloitteleva toimija, joten minun pitää kysyä usein neuvoja asioiden tekemisessä. Minun pitää myös kysyä neuvoja oikeanlaisen ohjelmointitavan seuraamisessa, jotta kirjoitan koodia oikeaan paikkaan oikealla tavalla: En voi esimerkiksi kirjoittaa kameraan liittyvää koodia käyttöliittymään liittyvän koodin sekaan.

Minun pitää kehittää sosiaalisia taitojani, sillä en ole kovin hyvä keskustelemaan kenellekään. Olen usein hiljaa asioista ja en välttämättä uskalla kysyä neuvoa, vaikka tarvitsen sitä. Minun tulee myös oppia enemmän siitä, millainen ohjelmistomme kokonaisrakenne on, sillä tiedän tällä hetkellä vain, miten ohjelmiston VR-puoli toimii. Tämän lisäksi minun tulee oppia oikeanlaisia tapoja kirjoittaa koodia, jotta teen siitä mahdollisimman selkeästi luettavaa sekä kevyttä ja toimivaa. Tähän auttaisi esimerkiksi erilaisten suunnittelumallien oppiminen.

2.1 SIDOSRYHMÄT TYÖPAIKALLA

Oheisessa kaaviossa (Kuvio 1) käydään läpi kaikki sidosryhmät, joihin työni vaikuttaa. Sidosryhmiä on sekä sisäisiä, että ulkoisia. Sisäisiin sidosryhmiin kuuluvat kaikki, joihin työni vaikuttaa työpaikalla, kun taas ulkoisiin sidosryhmiin kuuluvat kaikki työpaikan ulkopuolella olevat ryhmät.



KUVIO 1. Sidosryhmäkaavio

Sisäisiin sidosryhmiin kuuluu:

- Työkaverini, joiden kanssa toteutan sovellusta. Tekemäni työ vaikuttaa heidän työhönsä suoraan, sillä työskentelemme osittain samojen skriptien parissa.
- Myynnin työntekijät, jotka myyvät sovellusta eteenpäin. Koska 3D-osuus on sovelluksen näkyvin osa, työni tulee vaikuttamaan myynnin toimintaan.
- Osakkaat, eli yrityksen omistaja, sekä muut osakkaat, joiden osakkeiden arvo on riippuvainen tuotteesta.

Ulkoisiin sidosryhmiin kuuluu:

- Rakennuttajat, jotka käyttävät sovellusta asuntojen esittelyyn sekä myymiseen. Sovelluksen tulee siis näyttää mahdollisimman hyvältä.
- Rakennuttajien asiakkaat, jotka ostavat asuntoja rakennuttajilta. Asiakkaat voivat käyttää sovellusta sisustukseen, jotta he näkevät miltä asunto näyttäisi sisustettuna, sekä he voivat vaihtaa esim. seinien tapetteja ja näkevät suoraan, paljonko se tulisi maksamaan.

2.3 VUOROVAIKUTUSTAIDOT TYÖPAIKALLA

Tilanteita, joissa tarvitaan vuorovaikutustaitoja, ei ole paljoa. Teen töitäni toimistossa toisen henkilön kanssa, joka myös tekee samaa asiaa. Keskustelen hänen kanssaan välillä päivän tehtävistäni, sekä mitä ohjelmistoon tulee lisätä. Silloin tällöin meillä on palavereja, joissa keskustelemme muiden kanssa esim. ohjelmiston käyttöliittymän ulkoasusta. Keskustelemme näissä palavereissa siitä, mitä olen saanut aikaan sekä mitä minun pitää tehdä seuraavaksi. En ole toistaiseksi keskustellut kenenkään asiakkaan parissa, sillä se ei kuulu vastuualueeseeni.

En ole kovin sosiaalinen persoona, joten minulle vuorovaikutustaidot ovat olleet usein se heikoin osa. Pyrin kuitenkin poistumaan mukavuusalueeltani ja keskustelemaan asioista, jos tarve vaatii. Joskus minulla on vaikeuksia selittää vastaan tulleita ongelmiani, joten joudun selittämään asian monta kertaa.

3 PÄIVÄKIRJARAPORTOINTI

4.2.2019, Maanantai

Olin saanut tehtäväkseni käyttöliittymän skaalauksen mobiililaitteita varten, joten tavoitteenani oli saada se tehtyä. Käyttöliittymä oli liian pieni puhelimissa, mutta sopivan kokoinen tablettilaitteissa. Skaalausta varten etsin mobiililaitteen DPI:n (Dots Per Inch), ja laskin sen avulla kuinka iso mobiililaitteen näyttö on tuumissa. Jos näytön resoluutio oli tarpeeksi pieni, käytin DPI:tä avuksi käyttöliittymäobjektien skaalausta varten.

Kävin myös palaverissa, jossa keskustelimme vanhan käyttöliittymän päivittämisestä paremmaksi. Tarkoituksena on suunnitella lähes kokonaan uusi käyttöliittymä yhdessä asiakkaiden ja työntekijöiden kanssa, ja käyttöliittymän toteutus jäisi minulle.

Saavutin päivän tavoitteeni, sillä päivän päätteeksi sain skaalauksen toimivaksi. Käyttöliittymä skaalautui oikein puhelimessa, mutta tablettilaitteessa se pysyi normaalikokoisena.

5.2.2019, Tiistai

Päivä oli melko rauhallinen ja minulla ei ollut mitään erityistä tehtävää, joten etsin virheitä ohjelmistosta ja kokeilin eri ominaisuuksia. Löysin virheen, jonka vuoksi ohjelmistossa esitettävien huoneiden tekstuureihin tuli erilaisia mustia viivoja, joita ei pitäisi näkyä. Tämä johtui siitä, että tekstuurit tarvitsevat anisotrooppinen suodatus -ominaisuuden. Asetus meni pois päältä, koska ohjelmiston FPS (Frames Per Second) tippui liian alas. Kun FPS tippuu liian alas, ohjelmiston 3D-laatua heikennetään. Heikoimmalla tasolla anisotrooppinen suodatus meni kokonaan pois päältä.

Meillä oli toinen palaveri, jossa suunnittelimme yhdessä, millaista uutta käyttöliittymää ryhdymme tekemään. Palaverissa kävimme läpi monia eri ideoita, mutta emme saaneet kunnollista suunnitelmaa valmiiksi.

Päivän päätteeksi korjasin vielä yhden ohjelmistovirheen mobiililaitteen skaalauksessa: Aktivoimattomat elementit skaalautuivat vahingossa, mikä johti siihen, että ne näyttivät liian suurilta, kun ne aktivoitiin uudelleen.

6.2.2019, Keskiviikko

Tutkin aamun mobiililaitteen skaalausta, sillä kävi ilmi, ettei se toiminutkaan yhtä hyvin kaikissa puhelimissa. En kuitenkaan löytänyt sille mitään järkevää syytä, joten etsin muita virheitä ohjelmistosta, mutta en löytänyt mitään tällä kertaa. Päivän tavoitteenani minulla oli päästä suunnittelemaan uutta käyttöliittymää, mutta en voinut tehdä mitään siihen liittyvää ennen palaveria.

Kävimme keskipäivällä kolmannen palaverin uudesta käyttöliittymästä ja saimme yhdessä mietittyä millainen sen tulisi olla. Päädyimme ratkaisuun, jossa nappia painamalla avautuisi isompi valikko, jossa on nykyiset painikkeet ja niiden lisäksi teksti, joka kertoo mitä kyseinen painike tekee. Tällä hetkellä kaikki painikkeet ovat koko ajan käyttäjän näkyvissä ilman tekstejä, mikä voi hämmentää käyttäjää.

Loppupäivän vietin uuden käyttöliittymän suunnittelussa käyttäen Unityä. Otin vanhan käyttöliittymän ja asettelin kaikki painikkeet uudelleen omaan valikkoon, joka aukeaisi yhdestä painikkeesta. Näkyvien painikkeiden lukumäärä tippui seitsemästä painikkeesta kahteen.

Opin Unityn UI-elementtien sijoittelukomponenteista ja kuinka niitä käytetään, sillä käytin Vertical Layout Group -komponenttia painikkeiden asettelemiseen. Täten painikkeet ovat aina samassa kohdassa, oli paneelin koko mikä tahansa.

7.2.2019, Torstai

Päivän tavoitteena oli tehdä ohjelmistoon erilainen kameratila. Tällä hetkellä vakiokameratilana toimii perspektiivikamera, eli se näyttää 3D-mallin normaalissa 3D-muodossa. Tarkoituksena oli korvata tämä kamera ortograafisella kameralla, joka näyttää 3D-mallin suoraan ylhäältäpäin kaksiulotteisena pohjapiirustuksena.

Kameratilalle löytyi jo valmis pohja, jota käytettiin toisessa kamerassa, jolla luodaan huoneistosta kartta. Otin tämän pohjan takaisin käyttöön, sekä varmistin, että se toimisi vakiokamerana ohjelmistoa käynnistäessä.

Tämän jälkeen siirryin tekemään kameran keskittämistä tiettyyn huoneeseen. Tällä hetkellä kamera liikkui siihen kohtaan huoneistossa, mihin käyttäjä klikkasi hiirellään. Sen sijaan halusimme, että mihin tahansa käyttäjä klikkaisi huoneessa, kamera siirtyisi huoneen keskelle, jotta koko huoneisto näkyisi kuvassa. Sain tämän toiminnon aloitettua, mutta en valmiiksi asti, sillä keskittäminen ei toiminut toivotulla tavalla.

Minulle opetettiin C#-kielen LINQ-kyselyistä, joiden avulla pystyisi helposti hakemaan huoneen, johon on klikattu. Opin myös C#-kielen lambda-operaattorista, joka helpottaa tietojen hakemista.

8.2.2019, Perjantai

Jatkoin kameran keskittämisominaisuuden parissa ja sain sen toimimaan vaihtamalla yhden parametrin toiseksi. Koodi oli kuitenkin rakenteellisesti väärässä paikassa, joten minua pyydettiin tekemään sille oma luokka. Tein staattisen luokan, johon lisäsin uuden metodin keskittämistä varten. Tätä metodia voidaan käyttää nyt mistä tahansa, kunhan vain syöttää sille sen vaatiman tiedon, eli tässä tapauksessa klikatun peliobjektin.

Jouduin etsimään metodille sopivan paikan, sillä se oli myös alkuperäisessä suunnitelmassa rakenteellisesti väärässä paikassa. Sain luvan tehdä oman kameratila-luokan, jossa käytin kyseistä metodia. Tämä myös auttaa siinä, että en vahingossa riko kartan kameraa samalla. Opin paljon koodin rakenteesta, sillä jouduin tutkimaan sitä paljon tehdessäni uutta kameratilaa.

Viikko 1, 4.2.2019- 8.2.2019

Kehityin viikon aikana aika nopeasti ja olen tyytyväinen oppimaani määrään. Uskalsin kysyä neuvoa, jos en ollut varma jostain. Minulle myös kerrottiin, että olin tehnyt vaikutuksen palavereissa, sillä olin tiennyt oikeita termejä ja osannut kertoa ideoistani ymmärrettävästi.

Jouduin selvittämään LINQ-kyselyt sekä Unityn UI-elementtien komponentit ja miten ne toimivat. Opin myös lambda-operaattorista, sekä ohjelmiston koodin rakenteesta. Selvitin näitä joko internetin avulla tai kysymällä muilta.

Viikon aikana minulla ilmeni eniten ongelmia ohjelmiston käyttöliittymän skaalauksessa puhelimen näyttöä varten. En saanut tätä ongelmaa yksin ratkaistua, mutta aloitin ongelman ratkaisemisen tavalla, jota käytetään valmiissakin ratkaisussa: Näytön koon laskeminen puhelimen DPI:n avulla.

Käyttöliittymän skaalaamiseen on kaksi erilaista tapaa Unityn UI design -dokumenttien mukaan: ankkureita käyttämällä tai resoluutiota käyttämällä (Unity, 2019a). Skaalasimme käyttöliittymän resoluution avulla, sillä se toimi paremmin meidän käyttöliittymäasettelumme kanssa. Vaikka käytimmekin resoluutioskaalausta, tekstit ja ikonit olivat liian pieniä laitteilla, joissa on korkea DPI. Tätä varten jouduimme manuaalisesti pienentämään referenssiresoluutiota, jota resoluutioskaalaus käyttää, jotta saisimme käyttöliittymän näyttämään isommalta. Tämän vuoksi emme käyttäneet ankkureita, sillä ne eivät käytä referenssiresoluutiota tai muuta vastaavaa, joiden arvoon olisimme voineet vaikuttaa.

Käyttöliittymän skaalaamisen saimme toimimaan siten, että ensiksi haimme näytön koon vasemmasta yläkulmasta oikeaan alakulmaan tuumissa DPI:n avulla:

$$\sqrt{(Näytön\ leveys\ pikseleinä/DPI)^2 + (Näytön\ korkeus\ pikseleinä/DPI)^2}$$

Jos näytön koko on alle kuusi tuumaa, etsimme resoluutioskaalaus-komponentin, ja laskemme komponentin referenssiresoluutiota niin paljon, että käyttöliittymän koko kasvaa tarpeeksi suureksi. Emme siis voineet toteuttaa skaalausta suoraan ohjeiden mukaisesti, sillä se ei toiminut halutulla tavalla, vaan jouduimme kehittämään ratkaisun itse.

11.2.2019, maanantai

Päivän tavoitteena oli saada jonkinlainen pohja huoneen valitsemisefektiä varten. Kun käyttäjä haluaa valita tietyn huoneen asunnosta, ohjelmiston tulee näyttää jonkinlainen efekti, jotta käyttäjä saa jonkinlaisen palautteen, että huone on valittavissa.

Käytin huoneen valitsemiseen Line Renderer -komponenttia, jonka avulla voi piirtää viivoja 3D-maailmaan mihin vain haluamieni verteksipisteiden kohdalle. Tein tätä varten metodin, jonka avulla voidaan etsiä klikatun huoneen kattopeliobjekti (tai lattiapeliobjekti jos kattoa ei ole), peliobjektin verteksipisteet, jotka sitten syötetään Line Render -komponentille, joka piirtää viivan huoneen ympärille verteksipisteiden avulla. Verteksipisteistä tulee poistaa mahdolliset duplikaatit, joiden avulla vältetään liiallinen piirtäminen, joka säästää suorituskykyä, sekä ehkäisee viivojen piirtämistä vääriin paikkoihin.

Opin aika paljon siitä, miten Line Render-komponenttia voidaan käyttää Unityssä. Esimerkiksi sitä voi käyttää eri peliobjektien korostamiseen.

12.2.2019, tiistai

Asetin tavoitteekseni saada huoneen valitsemisefektin toimimaan siten, että kun käyttäjä liikuttaa hiiren osoittimen jonkun huoneen päälle, efekti menee päälle. Kun käyttäjä siirtää hiiren osoittimen toisen huoneen päälle, efekti sammuu edellisestä huoneesta ja menee päälle uudessa huoneessa. Kun käyttäjä klikkaa huonetta, efekti jää huoneen kohdalle päälle, eikä sammu ennen kuin käyttäjä klikkaa toista huonetta.

Efektin tekeminen onnistui toivotulla tavalla, mutta minun täytyi selventää koodiani ja yksinkertaistaa sitä. Päivän lopuksi keskityin Asset Bundle -datapakettien generointiin, joka on enimmäkseen valmiiden skriptien käyttöä ja odottamista.

13.2.2019, keskiviikko

Tavoitteenani oli saada asunnon 3D-malli mahtumaan kokonaan kameran kuvaan, oli asunto minkä kokoinen tahansa. Tähän oli jo valmis metodi, mutta se ei toiminut mielestäni halutulla tavalla.

Ohjelmiston ladattua asunnon, kamera noudattaa ortograafista projektointia. Täten kaikki 3D-elementit näyttävät ikäänkuin valokuvalta, kun elementtejä katsotaan suoraan ylhäältäpäin. Jotkut asunnot saattavat kuitenkin olla liian isoja, joten ne eivät mahdu kameran kuvaan. Löysin tälle kuitenkin metodin, joka suurentaa tai pienentää ortografisen projektoinnin kokoa asunnon rajojen mukaan, jotta asunto mahtuu kuvaan. Tämä metodi kuitenkin suurensi projektointia liikaa, joten asunto oli joko juuri sopiva tai liian kaukana.

Sain muokattua metodia sopivaksi siten, että rajoitan kuinka suureksi projektoinnin kokoa voi kasvattaa tai pienentää, sekä laskin sen eri tavalla. Jos asunto on liian suuri kuvaan korkeussuunnassa, kasvatamme projektoinnin kokoa hieman asunnosta lasketun koon lisäksi. Jos asunto on sopivan kokoinen, pienennämme projektoinnin kokoa hieman, jotta asunto ei vaikuta olevan liian kaukana. Metodin tekeminen onnistui hyvin, mutta se tulee testata läpikotaisin erikokoisilla näytöillä sekä kuvasuhteilla.

14.2.2019, torstai

Aloitin suunnittelemalla ja tekemällä materiaalikameraan uutta ominaisuutta, jonka avulla käyttäjä viedään hänen valitsemaansa huoneeseen. Kun käyttäjä klikkaa huonetta, se valitaan. Kun käyttäjä klikkaa huonetta uudelleen, kameran projektointi muuttuu ortografisesta tavalliseksi perspektiivi-projektoinniksi ja kamera liikutetaan valitun huoneen keskelle. Käyttäjälle myös annetaan mahdollisuus kääntää kameraa. Lisäsin tähän vielä painikkeen, jolla pääsee takaisin edelliseen näkymään, jossa näkyy koko asunto.

Työkaverini oli tehnyt metodin, jonka avulla pystyn hakemaan asunnon lattian reunapisteet, joten voin tehdä huonetta ympäröivästä valintaviivasta paremman näköisen, sillä se ei valitse enää turhia verteksipisteitä. Metodi toimii hyvin, mutta joudun laskemaan viivojen paikan uudelleen, sillä joskus ne menevät väärään paikkaan.

15.2.2019, perjantai

Aamulla minua pyydettiin, ettei huoneen valintaviiva käyttäisi Unityn tag-ominaisuutta, sillä se heikentää suorituskykyä. Muokkasin sen jälkeen valintaviivan koodia siten, että tagien sijasta se vertaisi vain kahta eri peliobjektia keskenään: LineRenderer ja ClickedLineRenderer. Muokkaamisessa ei mennyt kovin kauaa ja sain ominaisuuden toimimaan halutulla tavalla.

Tämän jälkeen siirryin korjaamaan koodia, jotta saisin muutkin kameratilat toimimaan. Kun käyttäjä on materiaalivaihtotilassa ja yrittää vaihtaa kameraa, kamerat jäävät jumiin eivätkä toimi halutulla tavalla. Tämän korjaamiseen meni aikaa, mutta kävi ilmi, että koodissa oli vain muutama ominaisuus pois päältä, joten laitoin ne takaisin päälle. Kameratilat toimivat normaalisti, mutta ohjelmisto vaatii silti syvempää testausta.

Viikko 2, 11.2.2019- 15.2.2019

Kehittymiseni on ollut viikon aikana hyvin tasaista, mutta en oppinut juurikaan mitään uutta, paitsi verteksipisteiden hakemisen mistä tahansa peliobjektin mesh-komponentista. Jouduin selvittämään tämän, jotta saisin Line Renderer -komponentin piirtämään viivoja verteksipisteiden kohdalle. Opin myös, miten tehdään uusia syötemetodeja ohjelmistoon, sillä minun piti tehdä Mouse Hover -metodi, jonka avulla tarkistan, minkä huoneen päällä hiiri on, jotta oikea huone voidaan ympäröidä Line Renderer -komponentilla.

Tämän kanssa minulla tuli kuitenkin ongelmia, sillä joskus Line Renderer -komponentti piirsi viivoja peliobjektien yli, eikä peliobjektin reunoja pitkin. Tämän sain ratkaistua työkaverini avulla: hän teki minulle metodin, joka hakee peliobjektin reunimmaiset pisteet, mikä toimi hyvin. Line Renderer -komponentti piirsi viivat oikeaan muotoon, mutta silti väärään paikkaan. Tämä on kuitenkin ongelma, jota minä en voi ratkaista, sillä ongelma tapahtuu mesh-komponenttien generointivaiheessa.

Selvittääkseni minkä tahansa mesh-komponentin verteksipisteet, Unityn dokumentaation mukaan minun tulee hakea tämän peliobjektin MeshFilter-komponentti, sekä poimia komponentista mesh-komponentin tiedot. Tämän jälkeen minun tulee tehdä lista kaikista verteksipisteistä. (Unity, 2019b). Tämä toimi oikein hyvin, mutta jos mesh-komponenti oli eri muotoinen kuin pelkkä neliö, Line Renderer -komponentti piirsi viivoja poikittain peliobjektin yli, mitä ei haluttu tapahtuvan. Koitin ratkaista tätä Unityn keskustelupalstalta löytyvien ohjeiden avulla, eli poistamalla listasta verteksejä, jotka olivat päällekäin (Unity Answers, 2019a). Tämä auttoi asiaa, mutta silti jotkin peliobjektit piirsivät viivoja vääriin paikkoihin. Yleisin ongelma oli, että Line Renderer -komponentti piirsi viivoja ikäänkuin spiraalina: Se aloitti peliobjektin ensimmäisestä pisteestä, yleensä peliobjektin pohjasta ja jatkoi viivojen piirtämistä ylöspäin viimeiseen verteksipisteeseen, joka oli usein peliobjektin päätypuolella. Tämän vuoksi viivoja saattoi joskus tulla peliobjektin pohjasta läpi.

Unityn dokumenteissa oleva toimintamalli oli juuri se mitä halusinkin, mutta se ei ikävä kyllä toiminut mesh-komponenttiemme kanssa, sillä jotkut niistä ovat todella monimutkaisia ja niissä on monia verteksipisteitä. Käytin Unity Answers -sivustolta ratkaisua, jonka joku muu oli keksinyt, mutta se ei ollut tarpeeksi hyvä tätä ongelmaa varten. Jos mesh-komponentit olisivat yksinkertaisempia,

käyttäisin tätä ratkaisua, mutta työnantajamme haluaa, että kaikki peliobjektit ovat mahdollisimman realistisen näköisiä, minkä vuoksi objekteissa on enemmän verteksejä kuin tarvitsisi.

18.2.2019, maanantai

Minua pyydettiin korjaamaan koodistani virheitä, jotka aiheuttivat NullRef-virheitä. Käytin aamun asian tutkimiseen ja löysin kohtia, joissa olin unohtanut tarkistaa, onko syöttämäni tieto “null” vai ei. Jos tieto oli null ja se pääsi jatkamaan eteenpäin, jokin metodeista saattoi kaatua.

Kun olin saanut virheet korjattua, tein loppupäivän LineRenderer-skriptiä, jonka avulla voidaan korostaa haluttuja objekteja, esim. huonekaluja. Tätä varten jouduin laskemaan peliobjektin bounds-arvoista jokaisen kulman verteksipisteen, jotta voisin vetää viivoja pisteiden välille oikein. Sain kulmien pisteet helposti käyttämällä “bounds.min” ja “bounds.max” ominaisuuksia (Unity Answers, 2019b). Yritin vetää pisteiden välille LineRender-komponentilla viivoja suoraan, mutta ne menivät ristiin päällekäin eivätkä toimineet halutulla tavalla. Tämän korjasin siten, että etsin tietyn järjestyksen missä viivat vedetään. Esim. LineRenderer käy ensin läpi kuution yläosan, sitten vetää viivan alimpaan pisteeseen, piirtää kuution alaosan, jonka jälkeen se piirtää puuttuvat viivat. Tämä onnistui oikein hyvin, ja käyttämällä GUI/Text-materiaalia sain viivat näkymään jopa seinien läpi, jotta oikean peliobjektin valinta onnistuu kunnolla.

19.2.2019, tiistai

Tavoitteena oli saada materiaalitilan kamera toimimaan siten, että sitä voisi pyörittää huoneen ympäri yläkulmasta katsoen. Huone pysyy siis paikoillaan ja huoneen keskelle luodaan ankkuripiste, jonka ympäri kameraa pyöritetään. Pystyin käyttämään 3D-kameratilan koodia tämän luomiseen, mutta karsin paljon käyttämättömiä asioita koodista pois.

Tämän jälkeen aloin suunnittelemaan, miten saisin korostettua LineRender-komponentilla tiettyyn pakettiin kuuluvat materiaalit (ns. linkitetyt materiaalit), yhden materiaalin sijasta. Minulle tehtiin työkalumetodi, jonka avulla saan helposti haettua mitkä peliobjektit kuuluvat mihin pakettiin. En saanut ongelmaa selvitettyä kokonaan, sillä joskus viivat menivät pois päältä silloin kun niiden ei olisi pitänyt.

20.2.2019, keskiviikko

Jatkoin siitä mihin edellisenä päivänä jäin, eli linkitettyjen materiaalien korostamisesta. Jouduin tekemään suurimman osan tähän liittyvästä koodistani uusiksi, sillä olin lähtenyt ratkaisemaan ongelmaa väärästä näkökulmasta. Yritin ääriviivata linkitettyjä materiaaleja samalla tavalla kuin huoneita, joten se ei toiminut, sillä huoneiden ääriviivaamiseen ei tarvitse tarkistaa kuin vain yhden peliobjektin tila.

Ratkaisin ongelman siten, että tallensin linkitetyt materiaalit eri tauluihin: tavallinen highlight-taulu, sekä clicked highlight -taulu. Jos materiaalia ei klikattu hiirellä, vaan hiirellä liikutettiin materiaalin päältä siten, että peliobjektiin ilmestyy ääriviivat, se tallennettiin highlight-tauluun. Kun materiaalia klikataan, ääriviivat jäävät materiaaliin kiinni ja kaikki linkitetyt materiaalit siirretään clicked highlight -tauluun. Unityn foorumeiden mukaan, voin vertailla näitä tauluja helposti SequenceEqual-metodin avulla (Unity Answers, 2019c) ja sain selville mitkä materiaalit minun tulee jättää käsittelemättä, kun hiirtä liikutetaan materiaalin ylitse. Opin siis tänään paljon taulujen käsittelystä, sekä materiaalien ID-numeroista.

21.2.2019, torstai

Tavoitteena oli saada piilotettua kaikki muut huoneet, kun käyttäjä oli valinnut yhden huoneen. Ratkaisuni oli seuraavanlainen: Kun käyttäjä valitsee huoneen, tehdään lista kaikista huoneista, paitsi siitä, jonka käyttäjä on valinnut. Sitten listassa olevat huoneet pistetään pois päältä. Tämä toimi hyvin, mutta seinät eivät noudata Unityn hierarkiaa, joten jouduin ratkaisemaan seinien piilottamisen eri tavalla. Seinien koot eroavat huoneen seinien päällysteiden koosta, sillä ne generoidaan eri tavalla: Seinä saattaa olla koko talon pituinen, mutta huoneen seinän päällystä on huoneen seinän kokoinen. Tämän takia en pystynyt jättämään seiniä paikoilleen, vaan minun piti säilyttää vain pelkät huoneen seinien päällysteet. Etsin kaikki seinät ja siirsin ne “HiddenLayer”-tasolle, ja Unity Answers -sivuston mukaan voin käyttää tähän bitwise-operaattoria, jonka avulla piilotin tämän tason kameralta (Unity Answers, 2019d). Täten kaikki seinät ”HiddenLayer”-tasolla jätetään renderöimättä.

Sen jälkeen etsin kaikki päällysteet, jotka ovat muissa huoneissa kuin käyttäjän valitsemassa, sekä laitan ne pois päältä. Viimeiseksi, etsin ikkunat ja ovet, jotka laitetaan pois päältä käyttäen niiden bounds-arvoja: Jos ikkunan tai oven bounds-arvot leikkaavat seinän päällysteen bounds-arvojen kanssa, se jätetään päälle, mutta kaikki muut laitetaan pois päältä. Tähän käytin avukseni Unityn

dokumentaation mukaisesti Bounds.Intersects-ominaisuutta (Unity, 2019c), jonka avulla voidaan tarkistaa, leikkaavatko bounds-arvot toisiaan.

22.2.2019, perjantai

Päivän tavoitteenani oli saada seinät käyttäytymään siten, että kaikki seinät, jotka olivat kameran ja huoneen keskipisteen välissä menisivät pois päältä, jotta käyttäjä näkisi huoneen sisälle ilman että seinät ovat tiellä. Tein metodin, joka käy läpi valitun huoneen kaikki päällystepeliobjektit ja laskee ensin käyttäen Unityn dokumentaation mukaisesti Vector3.Dot-ominaisuutta (Unity, 2019d), osoittaako kyseisen päällysteen forward-akseli kohti huoneen keskiosaa, eli onko päällyste huoneeseen päin vai ei. Jos ei, päällysteen vektori kerrotaan luvulla -1, jotta saadaan käänteinen vektori. Tämä tarvitaan siksi, että saadaan aina vektori, joka osoittaa keskelle huonetta sillä muuten seinät menevät pois päältä väärissä tilanteissa.

Tämän jälkeen käytetään taas Vector3.Dot-ominaisuutta, jotta saadaan tieto siitä, onko kamera päällysteen edessä vai takana. Jos se on päällysteen takana, päällyste laitetaan pois päältä.

Sain tämän ominaisuuden toteutettua melko nopeasti, mutta huoneen keskipiste on huono vertailukohta, sillä huoneet, jotka ovat esim. L-muotoisia, saattavat luoda huoneen keskipisteen huoneen ulkopuolelle. Näissä tilanteissa päällysteiden pois päältä laittaminen ei luultavasti toimi.

Viikko 3, 18.2.- 22.2.2019

Opin tällä viikolla miten eri peliobjektit on järjestetty ohjelmiston hierarkiassa, sekä kuinka voin etsiä esim. huonekalun tietystä huoneesta. Isoin asia minkä selvitin viikolla oli miten piirretään peliobjektille ääriviivat LineRenderer-komponentilla. Jouduin myös selvittämään, miten poimin kaikki peliobjektit, jotka kuuluvat samaan ryhmään (linkitetyt materiaalit), sekä miten piilottaa muut huoneet paitsi se, jonka käyttäjä on valinnut.

Suurin ongelma minulla oli linkitettyjen materiaalien korostamisen kanssa, sillä yritin poimia peliobjektien bounds-arvoja Unityn työkalujen avulla. Tämä palautti aina väärän tiedon, koska Unityn hierarkia toimii eri tavalla kuin objekteja varten tehty oma hierarkia. Käytin myös samaa tekniikkaa muiden huoneiden piilottamiseen, sillä muuten en saanut oikeata bounds-dataa, jonka avulla testaan, leikkaako ikkuna tai ovi seinän kanssa.

Tehdessäni LineRenderer-komponentilla ääriviivoja mihin tahansa peliobjektiin, minua kehotettiin käyttämään peliobjektin bounds-arvoja. Jokaisella peliobjektilla on bounds-pisteet, jotka ovat yleensä niin, että ne muodostavat kuution. Löysin pisteet helposti, mutta yrittäessäni vetää LineRender-komponentilla viivoja niiden välille, viivat ilmestyivät väärin ja leikkasivat ristiin. Tämä johtuu siitä, että LineRenderer-komponentti tekee vain yhden viivan, eikä katkaise viivaa toisen verteksipisteen jälkeen.

Tästä syystä jouduin etsimään, miten luoda kaikki verteksipisteparit joiden läpi viiva vedetään, sekä missä järjestyksessä verteksit tulee käydä läpi. Selvitin tämän kokeilemalla ensin LineRender-komponentin verteksipisteiden muokkausta Unityn editorissa. Siirtämällä pisteitä editorissa tajusin, että minun pitää ensin piirtää kuution alaosa ja sen jälkeen yksi jalka, joka menee kuution yläosaan. Yläosan käytyä läpi, jalat piirretään niin että osa viivoista menee kuution alaosassa jo olevien viivojen päälle.

Syy, miksi jouduin tekemään tämän näin vaikeasti, on siis se, etten voi katkaista viivaa keskeltä ja piirtää toista viivaa samalla LineRenderer-komponentilla, vaan tarvitsisin yhteensä kahdeksan LineRenderer-komponenttia jokaista peliobjektia kohti, koska viivoja tulee piirtää yhteensä kuudentoista verteksipisteen välille. Haluan pitää LineRenderer-komponenttien määrän niin vähäisenä kuin vain voin suorituskyvyn vuoksi.

Seurasin tätä varten toimintamallia, jossa kehotettiin käyttämään Debug.DrawLine-ominaisuutta viivojen piirtämiseen, tai Unity Answers -foorumien mukaan shaderia joka luo peliobjektiin ääriviivat (Unity Answers, 2019e). Debug.DrawLine-ominaisuus piirtää kyllä viivoja, mutta ne näkyvät vain Unityn Editorissa, eikä valmiissa tuotteessa. Shader oli toinen vaihtoehto, mutta uusien materiaalien lisääminen objekteihin ajon aikana voi olla raskaampaa kuin LineRenderer-komponentilla piirtäminen, tai se ei toimisi oikein, sillä kaikki peliobjektit poimivat tietonsa yhdestä ja samasta materiaalista: Shaderin piirtäessä ääriviivoja yhtä peliobjektia varten, on mahdollista, että se piirtäisi ääriviivat kaikkiin objekteihin, jotka käyttävät samaa materiaalia. Näiden syiden vuoksi jouduin keksimään oman ratkaisuni ja käyttäisin sitä jatkossakin. Jos voisin lisätä materiaaleja huoletta siitä, että se rikkoisi jotain, käyttäisin ennemmin ääriviivoja piirtävää shaderia.

25.2.2019, maanantai

Tavoitteenani oli tehdä materiaalikameralle siirtymä, kun kamera liikkuu pohjapiirrustusnäkymästä huonenäkymään. Koska kamera on ortograafinen pohjapiirrustusnäkymässä, minun piti kehittää sulava siirtyminen ortograafisesta projektoinnista perspektiivi-projektointiin.

Käytin tähän apuna kameran projektointimatriisia, johon voin syöttää haluamiani tietoja: Tein metodin käyttäen apunani viestiä Unityn keskustelupalstalla, joka käyttää Mathf.Lerp-ominaisuutta ja muokkaa matriisin nykyisestä projektointimatriisista haluttuun, esim. ortograafiseen projektointimatriisiin sulavasti (Unity Forums, 2019). Tein kameralle myös rotaatio- ja liikkumismetodit, joiden avulla kamera voidaan liikuttaa oikeaan paikkaan ja kääntää oikeaan asentoon. Opin siis, miten kameran projektointimatriisit toimivat, sekä miten niitä voi käyttää hyödyksi, kun kameran projektointia vaihdetaan ortograafisen ja perspektiivi-projektoinnin välillä.

26.2.2019, tiistai

Aloitin uusien valikkojen ja painikkeiden tekemisen. Kun käyttäjä valitsee tietyn huoneen, avataan lista huoneen vaihdettavista materiaaleista. Kun materiaalia klikkaa listasta, materiaali korostetaan samalla LineRenderer-komponenttia käyttävällä metodilla, jota olen aiemmin työstänyt.

Lisäksi huomasimme, että jos huone on hieman vinossa, LineRenderer-komponentti piirtää peliobjektien korostusviivat väärässä kulmassa. Korjasimme tämän käyttämällä peliobjektin sharedMesh-komponentia, joten korostusviivat piirretään aina peliobjektin paikallisessa 3D-avaruudessa, eikä globaalissa 3D-avaruudessa.

27.2.2019, keskiviikko

Jatkoin valikkojen työstämisen parissa. Tällä kertaa työstin materiaalivalikkoa, josta käyttäjä voi valita esim. erivärisen seinän. Materiaalivalikko oli alun perin tehty siten, että se luodaan vaakasuoraan näytön alalaitaan. Minun piti käydä muokkaamassa materiaalivalikon koodia siten, että se luotaisiin pystysuoraan. Tämän lisäksi lisäsin sille ominaisuuden liukua ruudun vasemmasta laidasta esiin, sekä takaisin piiloon, kuten muutkin ohjelmiston valikot. Valikkoon ladattavat tiedot piti myös järjestää vaakasuorasta pystysuoraan. Eniten aikaa minulla meni löytää, miten saada valikko liukumaan piiloon tai esiin oikeissa tilanteissa. Tämä vaati koodin tutkimista.

28.2.2019, torstai

Jatkoin siitä mihin eilen jäin, sillä valikot eivät aina avautuneet oikeissa tilanteissa. Yritin selvittää, miten alkuperäinen valikon liikuttaminen oli tehty ja huomasin, että se ei tule toimimaan minun tapauksessani, sillä metodit olivat tehty käyttämään Toggle-komponenttia, jota ei käytetä kaikissa painikkeissa.

Tein metodille tuen myös Button-komponentille, jotta saisin sen toimimaan useammissa tilanteissa. Tämä ei ikävä kyllä riittänyt, vaan jouduin tekemään materiaalipeliobjekti-listalle oman metodin, joka sulkee tai avaa sen kun toinen valikko avataan, joko nappia painamalla tai jostain muualta.

1.3.2019, perjantai

Sairaana.

Viikko 4, 25.2.- 1.3.2019

Opin viikon aikana miten valmis koodipohja käsittelee käyttöliittymää ja mitä minun tulee muokata, jotta saan sen toimimaan halutulla tavalla. Opin myös, miten kameran projektoinnin voi vaihtaa sulavasti ortograafisesta projektoinnista perspektiiviprojektointiin, sekä päinvastoin.

Minulla tuli eniten ongelmia käyttöliittymän siirtymien kanssa, sillä käyttöliittymän koodi oli kirjoitettu tietyllä tavalla: käyttöliittymässä käytetään apuna Toggle-komponenttia, jota ei voi lisätä kaikkiin paikkoihin, sillä esim. painike voi olla vain joko Toggle- tai normaali Button-komponentti. Jouduin täten tekemään koodiin tuen Button-komponentillekin, mutta en ole varma, kuinka hyvin se toimii. Tämä ei siltikään auttanut valikon siirtymien kanssa, vaan avasi välillä vanhan valikon uuden päälle. Sen vuoksi tein oman metodin, jonka avulla uusi valikko voidaan avata. Se myös tarkistaa onko vanhoja valikkoja auki ja sulkee ne. Luultavasti joudun tekemään tämän uusiksi, sillä kyseinen ratkaisu ei ole ihanteellinen, koska sitä ei voi käyttää kaikkien valikkojen kanssa.

Tehdessäni siirtymää kameralle, halusin tehdä projektointimuotojen välille sulavan siirtymän. Tähän onnekseni löytyi toimintamalli heti internetistä, joten seurasin löytämiäni ohjeita. Unityn keskustelupalstojen ohjeiden mukaan, kamerat käyttävät Matrix4x4-luokkaa, eli projektointimatriisia, johon tallennetaan kaikki kameran tarvitsema tieto, kuten esim. korkeus, leveys, kuvasuhde ja näkökenttä (Unity Forums, 2019).

Käyttämällä for-silmukkaa, jossa käytetään Mathf.lerp-ominaisuutta, saadaan sulavasti siirrettyä tieto vanhasta projektointimatriisista uuteen projektointimatriisiin. Siirtymä toimi täydellisesti, mutta kameran tila oli vielä väärässä, joten kaikki hiiren syötteet menivät vääriin koordinaatteihin. Tämän vuoksi kameran tila pitää vielä vaihtaa oikeaksi, jotta kaikki syötteet, jotka luottavat kameraan toimivat.

En löytänyt tälle toimintamallille eri vaihtoehtoja, sillä kamera käyttää Matrix4x4-luokkaa molemmissa projektointimuodoissaan. Tämän vuoksi päädyin käyttämään löytämääni toimintamallia ja koska se toimi niin hyvin, tulen käyttämään sitä jatkossakin.

4.3.2019, maanantai

Tavoitteenani oli saada materiaalikamera toimimaan siten, ettei se voisi mennä lattian alapuolelle. Tein siis kameran pyöritysmetodiin tarkistuksen: Jos kamera menee lattian alapuolelle, se siirretään takaisin lattian päälle, ja kameran paikallissijainti resetoidaan. Tämän avulla sain aikaan efektin, jossa kamera, jota yritetään liikuttaa lattian alapuolelle, tavallaan liu'utetaan lähemmäs ankkuriaan.

Korjasin ongelman, joka keskeytti luomani siirtymisefektin: Kun ohjelmistoa käytetään WebGL-alustalla, ohjelmisto laskee mitä grafiikka-asetuksia sen tulisi käyttää, jotta toiminnallisuus pysyisi mahdollisimman sulavana. Kun grafiikka-asetuksia vaihdetaan, kameran projektointimatriisi resetoituu, joka täten keskeyttää siirtymisefektin. Korjasin ongelman niin, että grafiikka-asetuksien muutokset pysäytetään ennen siirtymisefektiä ja jatketaan taas sen jälkeen.

5.3.2019, tiistai

Käytin tämän päivän ohjelmistovirheiden etsimiseen ja korjaamiseen. Etsin kaikesta mitä olin tehnyt erinäisiä ongelmia ja yritin selvittää niitä. Yksi isoimmista ongelmista oli, että peliobjektien ääriviivat eivät aina lähteneet pois päältä, syystä tai toisesta. Tämä johtui siitä, että eräässä metodissa vertailtiin peliobjektien nimiä ja jos objekteilla oli eri nimet, vaikka peliobjektit olivat samassa ryhmässä, ääriviivat eivät lähteneet pois päältä. Vaihdoin vertailuksi nimien sijasta peliobjektien BundleID:n ja metodi alkoi toimimaan oikein.

6.3.2019, keskiviikko

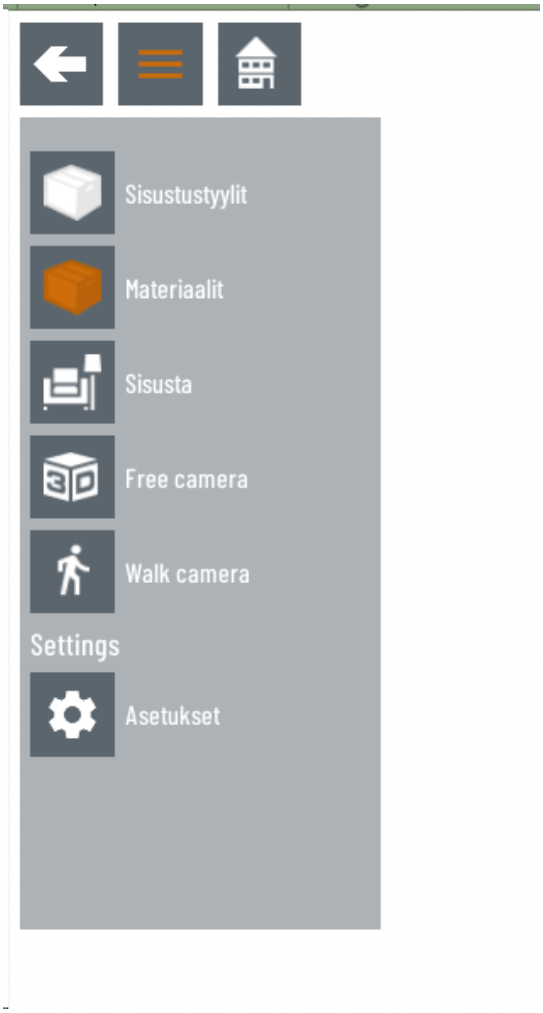
Suunnittelin ja toteutin päivän aikana materiaalitilan kameralle kontrollipainikkeet, joiden avulla kameraa voi pyörittää sekä kuvaa voi zoomata. Tämä oli haluttu ominaisuus, sillä hiirellä tai sormella kameran kääntäminen voi olla joillekin vaikeaa. Toteutuksessa ei mennyt kauaa, tosin jouduin tekemään monta eri versiota, sillä minulta pyydettiin uusia ominaisuuksia aina kun olin saanut yhden tehtyä. Uutena asiana opin EventTrigger-komponentista: EventTrigger-komponentilla voi helposti lisätä esim. OnPointerDown- ja OnPointerUp-tapahtumia, joiden avulla voin tarkistaa, onko painike pohjassa vai ei. Jos painike on pohjassa, kameraa liikutetaan niin kauan, kunnes painikkeesta päästetään irti. Loppupäivän vietin uuden valikkojärjestelmän suunnittelussa, sillä vanha menetelmä ei toimi halutulla tavalla, vaan saattaa avata vääriä valikkoja kesken kaiken.

7.3.2019, torstai

Jatkoin uuden valikkojärjestelmän suunnittelua ja toteutusta, ja sain sen osittain toimimaan. Sen sijaan että käytetään Toggle-komponentteja, katsotaan vain mikä on edellinen valikko ja suljetaan se, sekä avataan uusi tilalle. Tämä menetelmä toimii normaaleissa valikoissa, mutta materiaalilista ei avaudu oikein, kun esim. päävalikko avataan. Tämä luultavasti tarvitsee jonkinlaisen apumetodin, joka avaa materiaalilistan oikeassa tilanteessa.

8.3.2019, perjantai

Uuden valikon testiversio (kuvio 2), jossa suurin osa vanhassa käyttöliittymässä olevista painikkeista on siirretty päävalikkoon. Päävalikon ulkopuolella on vain takaisinpäin-painike, päävalikon oma painike sekä huonelistauspainike.



KUVIO 2. Uusi valikko

Jatkoin uuden valikkojärjestelmän toteutusta ja sain sen toimimaan suurimman osan ajasta. Tämä järjestelmä toimii paremmin kuin vanha järjestelmä uusien valikkojen kanssa. Ongelmana on kuitenkin, että joskus, kun jonkun valikon pitäisi aueta uudelleen, se ei aukea. Tämän lisäksi kehitin myös takaisinpäin-painikkeen, jonka avulla voi liikkua materiaalinvalintatilasta taaksepäin karttanäkymään. Minun pitää myös selventää ja suoraviivaistaa koodiani, sillä tällä hetkellä se ei ole optimaalista.

Viikko 5, 4.3.- 8.3.2019

En oppinut juurikaan kovin monia uusia asioita tällä viikolla, paitsi EventTrigger-komponentin käytön, jonka toimintaa jouduin selvittämään kamerakontrolleja varten. Jouduin myös selvittämään viikon aikana muutamia ongelmia koodissani, mutta suurin osa niistä oli joko huolimattomuusvirheitä, tai vanhan ja uuden koodin yhteensopimattomuutta.

Kaksi suurinta ongelmaa olivat kameran siirtymäefektin keskeytys, kun laatuasetuksia muutetaan, sekä peliobjektien ääriviivojen ilmestyminen vääriin paikkoihin. En löytänyt vastauksia siihen miksi laatuasetuksien vaihtaminen keskeyttää siirtymäefektin, mutta päättelin sen johtuvan siitä, että vaihtaessa laatuasetuksia, Unity resetoi kameran projektointimatriisin. Peliobjektien ääriviivojen oikeat paikat löysin käyttämällä peliobjektien bounds-arvoja, josta löysin kaikkien verteksipisteiden oikeat paikat.

Tehdessäni kameran painikkeita, huomasin että Unityn Button-komponentti sisältää vain OnClick-metodin, mikä tekee tarvittavan asian vain, kun painiketta klikataan, eli se ei rekisteröi painikkeen pohjassa pitämistä. Tähän löysin ohjeet Unity Answers -sivulstolta, jossa kehotettiin käyttämään EventTrigger-komponenttia, johon voisi syöttää OnPointerDown- ja OnPointerUp-tapahtumia (Unity Answers, 2019f).

Jouduin tekemään kameran pyöritykselle metodin Update-metodin sisälle, jota päivitetään joka kerta kun kuva päivitetään, eli esim. 60 kertaa sekunnissa. Painikkeella on oma muuttujansa, joka ilmoittaa mihin suuntaan kameraa pyöritetään ja kuinka nopeasti. Esim. -1 pyörittää hitaasti vastapäivään, kun taas 1 pyörittää hitaasti myötäpäivään. Tein myös samanlaisen metodin kameran ylös- sekä alaspäin kääntämiselle, sekä zoomaukselle. Zoomauksessa minun piti syöttää

zoomausmetodille, mihin kohti ruutua zoomataan, eli tässä tapauksessa keskelle, johon käytin Screen.height sekä Screen.width ominaisuuksia. Zoomausmetodille pitää myös syöttää, kuinka paljon halutaan zoomata.

En löytänyt tarkkaa toimintamallia kyseisille painikkeille, joten sovelsin löytämääni tietoa parhaani mukaan ja sain täten toimivat painikkeet. EventTrigger-komponentti on kuitenkin mielestäni hyvä tapa lisätä enemmän toiminnallisuutta Unityn UI-painikkeisiin ja käytän sitä varmasti jatkossakin.

11.3.2019, maanantai

Korjasin uuden valikkojärjestelmän toteutuksessa tulleita virheitä ja sain selville, että logiikka, jota olin seurannut oli toimiva, mutta esim. coroutine joka sulkee ja avaa paneeleja, saattoi jäädä jumiin, kun se yritti sulkea ja avata samaa valikkopaneeli yhtaikaa. Ehkäisin tämän tarkistamalla, ettei samaa paneelia yritetä avata kahdesti. Tein myös metodin, jonka avulla voidaan helposti sulkea kaikki valikkopaneelit, joita käytän takaisinpäinpainikkeessa.

Koodissani on vielä paljon testattavaa ja siivottavaa. Koodaustaitoni ovat kuitenkin parantuneet huomattavasti.

12.3.2019, tiistai

Aloitin decorate-tilan lisäämisen uuteen valikkojärjestelmään. Decorate-tilassa voi koristella asuntoa eri objekteilla ja tilan aktivoituessa se avaa uuden valikkopaneelin, josta voi valita objekteja. Sain tämän valikkopaneelin toimimaan hyvin uuden valikkojärjestelmän kanssa. Suurin ongelma oli kameratilan vaihtamisen kanssa, sillä se saattoi sulkea paneeleja, joita ei ollut tarkoitus sulkea. Sain tämän ongelman kuitenkin korjattua tarkistamalla, mikä tila oli käytössä.

13.3.2019, keskiviikko

Sain tänään palautetta työstäni: Minua pyydettiin tekemään käyttöliittymästä sellainen, että käyttäjä tietää, mitä tapahtuu, kun klikkaa painiketta. Toinen ongelma on, että osa painikkeista menee päälle yhtä aikaa, sekä joskus kamera menee mallin ulkopuolelle.

Aloitin painikkeiden korjaamisesta, ja kävi ilmi, että minun piti käyttää samaa ToggleGroup-komponenttia kaikissa toggle-painikkeissa. Tämä tekee sen, että vain yksi toggle-painike voi olla päällä, eli muut toggle-painikkeet menevät pois päältä. Tämä korjasi itsessään monta ongelmaa, mutta jouduin tekemään koodiini pieniä muutoksia, jotta valikot toimivat oikein.

14.3.2019, torstai

Pyysin lisää palautetta työstäni ja minua pyydettiin tekemään lista huoneista pohjapiirrosnäkymää varten. Kun käyttäjä liikuttaa hiirtä huoneen nimen kohdalle listassa, haluttu huone korostetaan, ja kun käyttäjä klikkaa huonetta listasta, huoneeseen siirrytään.
Tein huonelistaan tarkistuksen, joka katsoo ensin, missä tilassa ohjelmisto on. Jos ohjelmisto on pohjapiirrostilassa, korostetaan huone. Muussa tapauksessa listaa käytetään vain kameran liikuttamiseen.

Huoneen korostamisen sain tehtyä EventTrigger-komponentin avulla, joka luodaan samalla kun huoneistolista luodaan. Unity Answers -sivuston mukaan, tähän komponentiin pitää lisätä toinen EventTrigger-ominaisuus, joka määrittää mitä komponentti tekee ja missä tilanteessa (Unity Answers, 2019g). Tässä tapauksessa halusin, että kun hiiri on huonelistassa olevan painikkeen päällä, painikkeessa ilmoitettu huone korostetaan.

15.3.2019, perjantai

Jatkoin siitä mihin jäin eilen, eli huoneiden (Kuvio 3) ja peliobjektien (Kuvio 4) korostamisesta, kun hiiren osoittimen liikuttaa listassa huoneen tai peliobjektin painikkeen päälle. Tein aamulla peliobjektien korostamisen, joka toimii samalla tavalla kuin huoneiden korostus.

KUVIO 3. Huoneen korostus kartassa ja listassa

KUVIO 4. Materiaalipeliobjektien korostus huoneessa ja listassa

Sen lisäksi tein myös ominaisuuden, joka ympäröi painikkeen, kun huone korostetaan. Tätä varten jouduin lisäämään jokaisen LineRenderer-komponentin lisäksi myös UIElement-komponentin, johon tallennan, mitä painiketta tämä LineRenderer-komponentti vastaa. Esimerkiksi olohuoneen korostava LineRenderer vastaa olohuonepainikkeeseen listassa.

Tämä toimi muuten ihan mallikkaasti, mutta jouduin tekemään seiniä varten oman painikkeen, sillä muuten seinäpainikkeita olisi ollut useampia jokaista seinän osaa varten. Seinällä on siis oma painike, joka sisältää kaikki huoneen seinien osat. Tämän avulla sain painikkeen korostuksen toimimaan oikein, ilman ylimääräisiä painikkeita.

11.3.- 15.3.2019, viikko 6

Ymmärrän hieman paremmin, miten Unityn UI-elementit toimivat, sekä osasin kehittää toimivan käyttöliittymäjärjestelmän, jonka avulla voidaan avata ja sulkea paneeleja. Tästä teki vaikeampaa se, että minun piti kirjoittaa kyseinen järjestelmä vanhan päälle ilman, että rikkoisin mitään. En ole ihan varma, rikoinko jotain, mutta se selvinnee jossain vaiheessa.

Jouduin selvittämään vanhaa koodia, josta jouduin soveltamaan metodeja tai korvaamaan niitä uutta käyttölittymäjärjestelmää varten. Minulla tuli lukuisia ongelmia, mutta yksi vaikeimmista oli paneelien avautuminen väärään aikaan. Korjasin tämän siten, että tein lukuisia tarkistuksia, joiden avulla voidaan katsoa, mikä käyttöliittymäelementti on päällä ja mikä ei.

En löytänyt tähän tilanteeseen minkäänlaisia toimintamalleja, sillä kyseessä oli itse suunniteltu käyttöliittymäjärjestelmä. En ole seurannut mitään esimerkkejä, vaan keksin järjestelmän ihan itse, soveltaen vanhaa järjestelmää pohjana.

Järjestelmä toimii siten, että jokaisella valikkopaneelielementillä on totuusarvo (boolean), joka laitetaan päälle tai pois sen mukaan, onko paneeli päällä vai ei. Järjestelmä myös tallentaa edellisen käytetyn paneelin muuttujaan, jonka avulla suljemme aina vanhan paneelin uuden tieltä. Tämän lisäksi, minun piti kehittää takaisinpäin-metodi, jonka avulla pääsemme aina nykytilanteesta edelliseen tilanteeseen; Tilanteilla tarkoitan tilaa, jossa käyttöliittymä on, esim. materiaalinvalinta-tila, josta siirrymme takaisinpäin pohjakartta-tilaan. Takaisinpäin-metodin kehittäminen ei ollut niin vaikea, sillä minun pitää vain tarkistaa, missä tilassa ohjelmisto on ja sen perusteella tehdä

muutoksia käyttöliittymään. Käyttöliittymäjärjestelmä vaatii vielä kovasti testausta ja korjauksia, mutta toistaiseksi se vaikuttaa toimivan ihan hyvin.

18.3.2019, maanantai

Päivän tavoitteenani oli tehdä käyttöliittymästä vieläkin yksinkertaisempi. Sain palautetta, että takaisinpäin-painike oli joskus hämmentävä, sillä se teki välillä eri asioita. Tein niin, että painike piilotetaan, kun sillä ei ole tarvetta (Kuvio 5).

KUVIO 5. takaisinpäin painikkeen piilotus

Lisäksi myös yhdistin kameranäppäimet yhteen valikkoon, sen sijaan että kameratilat olisivat molemmat omat valikkonsa. Tämä mielestäni selkeytti käyttölittymää huomattavasti, sekä helpottaa käyttöä, sillä käyttäjän ei tarvitse vaihdella tilaa valikosta, vaan voi vaihtaa sen suoraan näytön yläreunasta. Tämän jälkeen aloin suunnittelemaan, miten saisin näytöllä näkyvät kameran painikkeet toimimaan kaikissa kameratiloissa.

19.3.2019, tiistai

Suunnittelin ja toteutin metodia, jonka avulla voidaan siirtyä suoraan valitsemaan materiaaleja huoneessa, jossa käyttäjä jo on: Kun käytetään vapaa kamera -tilaa, käyttäjä voi kävellä huoneistossa. Jos käyttäjä valitsee materiaalitilan, tällä hetkellä pohjakartta avautuu kuten normaalisti. Metodin avulla voidaan ohittaa pohjakartta kokonaan, ja siirtyä suoraan huoneeseen, jossa käyttäjä käveli aiemmin.

Opin tänään Dictionary-muuttujasta, jonka avulla pystyin listaamaan huoneiden bound-datan sekä huoneiden SpaceViewModel-datan samaan listaan (Microsoft, 2019a). Stack Overflow -sivuston artikkelin mukaan, voin käyttää tähän avuksi KeyValuePair-muuttujaa, jonka avulla pystyin helposti etsimään haluamani tiedot Dictionary-muuttujasta (Stack Overflow, 2019).

20.3.2019, keskiviikko

Jatkoin eilisestä, eli metodista, jonka avulla voidaan siirtyä valitsemaan materiaaleja, kun käyttäjä on jo huoneessa. Metodin kanssa ilmeni ongelmia, sillä kamera ei aktivoitunut oikein, kun kameratilaa vaihdettiin. Ratkaisin kameran aktivoinnin siten, että sen sijaan että käytämme normaalia aktivointitilaa, tarkistamme ensin, onko kamera huoneen sisällä ja hyppäämme sitten normaalin kamerakäyttäytymisen yli, eli kamera ei edes yritä avata pohjakarttaa, vaan siirtyy suoraan materiaalinvalinta-kameratilaan. Tämän lisäksi korjasin myös ohjelmistovirheitä, joita ilmeni aiemmin tekemäni GeometryTools-luokan kanssa, sekä aloitin suunnittelemaan, kuinka saisin asunnon mallin pysymään kamerassa, eikä se menisi kameran ulkopuolelle missään vaiheessa.

21.3.2019, torstai

Jatkoin suunnittelua asunnon mallin pitämisestä koko ajan kameran näkökentässä ja löysin keinon, jonka avulla pystyin toteuttamaan tämän. Käytin CalculateFrustumPlanes-metodia, minkä avulla sain kameran frustum-pinnat. Tämän jälkeen vertasin Unityn dokumentaatiossa esitellyn TestPlanesAABB-metodin avulla, onko asunnon bounds frustum-pintojen sisällä (Unity, 2019f). Jos bounds-arvot ovat frustum-pintojen sisällä, asunto on kameran näkökentässä. Tein nämä testit omaan metodiinsa, joka palauttaa true-arvon, jos kamera näkee asunnon, mutta ohitan testit

kokonaan, jos kamera on talon bounds-arvojen sisäpuolella. Tämän avulla pystyn pysäyttämään kameran liikuttamisen, jos asunto menee kameran ulkopuolelle. Metodi kuitenkin vaatii hienosäätöä, sillä kamera saattaa pysähtyä liian lähellä asuntoa.

22.3.2019, perjantai

Siirryin kameran pyörittämisen suunnitteluun, sillä toivottiin, ettei kameraa voi kääntää asunnon lattian alapuolelle. Tätä varten joudun kehittämään tarkistuksen, ettei kamera voi mennä lattian koordinaattien alapuolelle, mutta toistaiseksi en ole saanut sitä toimimaan oikein. Jatkoin myös UI-elementtien toteutusta, sillä yritän saada ne mahtumaan ruudulle, oli käyttäjän resoluutio mikä tahansa.

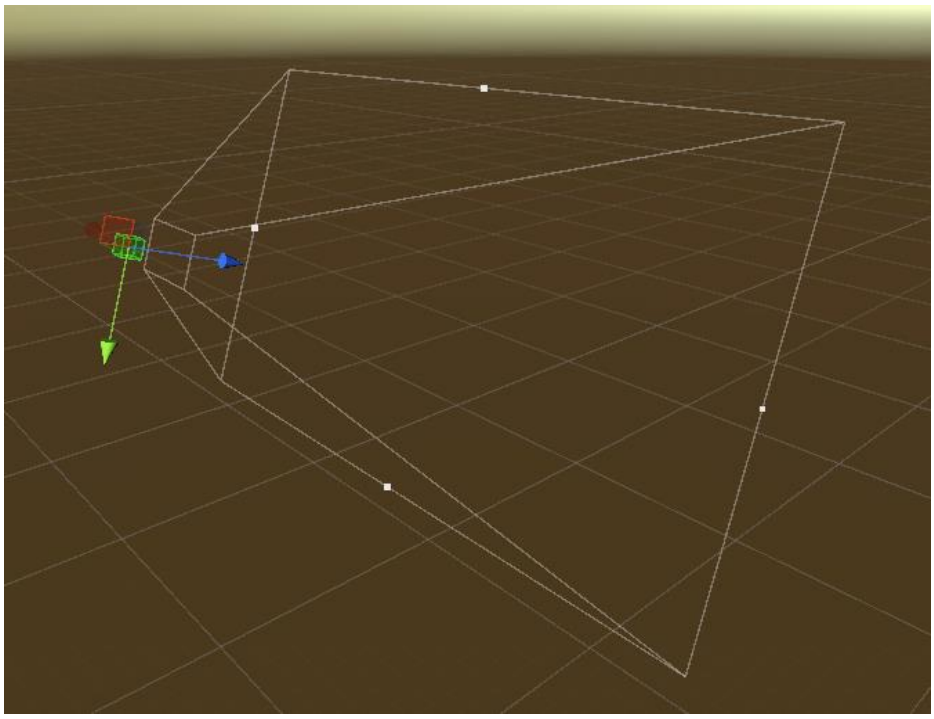
18.3.- 22.3.2019, Viikko 7

Opin tällä viikolla muutamia eri asioita, kuten Dictionary ja KeyValuePair-muuttujat, sekä frustum-pinnat. Jouduin selvittämään näitä asioita eri ominaisuuksia varten, jotta saisin ne toimimaan mahdollisimman hyvin. Eniten minulla tuli ongelmia käyttöliittymän skaalaamisessa, sillä Unityn UI-elementit voivat olla hyvin hämmentäviä, koska mielestäni osa niiden komponenteista sekä arvoista on nimetty oudosti.

Viikon mielenkiintoisin asia, josta opin, oli kameran frustum-pinnat (Kuvio 6) sekä miten niitä voi käyttää peliobjektin pitämiseen kameran kuvassa. Tässä tapauksessa frustum on perspektiivikameran näkökenttä, joka koostuu Unity dokumentaation mukaan kuudesta eri pinnasta (Unity, 2019e):

1. Near frustum, joka on kameran lähin pinta. Kaikki mitä jää kamerapeliobjektin ja near frustum pinnan väliin, jätetään renderöimättä.
2. Far frustum, joka on kaukaisin pinta. Kaikki mihin far frustum ei osu, jätetään renderöimättä.
3. Top, left, right, bottom frustum pinnat. Nämä ovat pinnat, jotka tulevat near- ja far- pintojen väliin.

Microsoftin ohjeiden mukaan, pinnat muodostavat yhdessä pyramidin ja kaikki mitä on näiden pintojen sisällä, kamera näkee ja täten renderöidään näytölle (Microsoft, 2012).



KUVIO 6. kameran frustum-pinnat

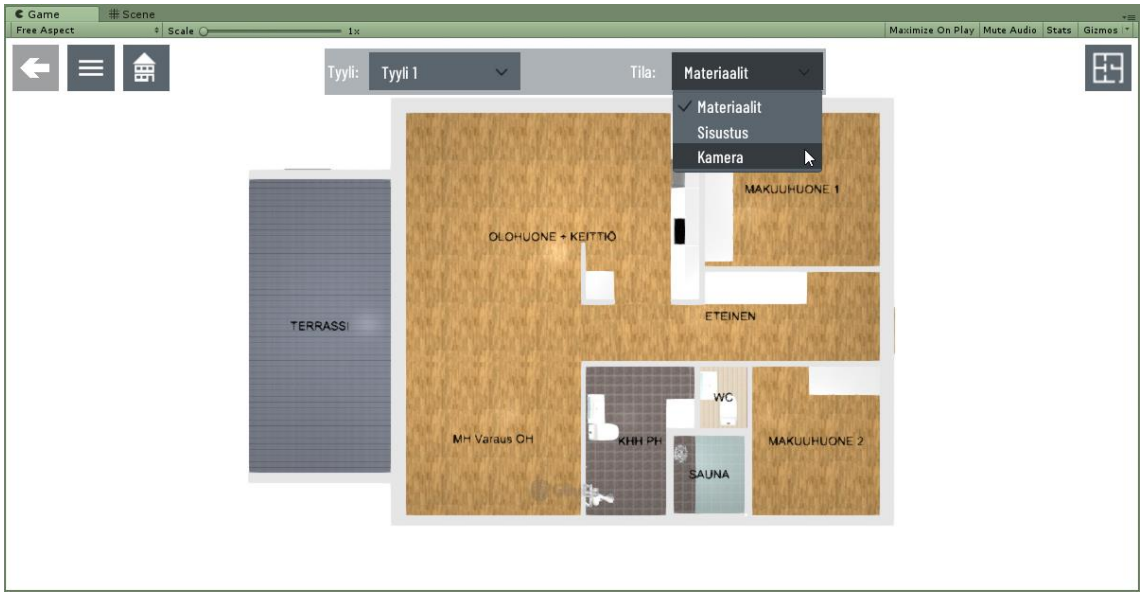
Unityn virallisten ohjeiden mukaan, voin syöttää nämä pinnat Unityn TestPlanesAABB-metodiin, joka etsii, onko tietyn peliobjektin, tässä tapauksessa asunnon, bounds-arvot frustum-pintojen sisällä (Unity, 2019f).

Jotta saan asunnon pysymään aina kamerassa, testaan ensin näillä metodeilla, onko asunto kameran kuvassa. Jos on, en tee mitään, mutta jos ei, siirrän kameran positioon, jossa se viimeksi näki asunnon. Tämä toimii ihan hyvin, mutta jouduin pienentämään asunnon bounds-arvoja, sillä muuten kamera näki asunnon, vaikka näkyvissä oli vain yksi pieni kulma. Pienentämällä bounds-arvoja asunto on “pois” kameran kuvasta, kun asunto on puoleksi ulkona kuvasta. Täten asunto ei katoa ikinä kuvasta.

Olisin vaihtoehtoisesti voinut myös verrata, onko asunnon keskipiste kamerassa käyttäen WorldViewToViewPort-metodia, mutta en käyttänyt sitä sillä Unity Answersin mukaan, se ei ole yhtä tarkka (Unity Answers, 2019h). Siksi päädyin testaamaan, onko asunto frustum-pintojen sisällä ja se toimii hyvin tähän tarkoitukseen.

25.3.2019, maanantai

Kehitin uutta statuspalkkia (Kuvio 7), jonka avulla voidaan vaihtaa tilojen välillä nopeasti. Statuspalkki myös näyttää, missä tilassa käyttäjä on. Tämän kanssa minulla tuli hieman ongelmia, sillä en voinut suoraan laittaa painikekomponentteja vaihtamaan Dropdown-komponentin tilaa, koska muuten sain Stack Overflow-virheen. Virhe ilmaantuu koska Dropdown-komponentti vaihtaa painikkeen tilaa, joten painikkeet sekä Dropdown-komponentti viestittelevät keskenään loputtomiin. Tein tästä syystä painikkeille oman metodin Drodpown-komponentin tilan vaihtamista varten, joka laittaa Dropdown-komponentin epäaktiiviseksi tilan vaihtamisen ajaksi.



KUVIO 7. statuspalkki

Korjasin myös ongelman kamerassa: Joissain tapauksissa, kun käyttäjä on huoneessa ja valitsee materiaalitilan, kamera saattoi alkaa pyörimään ympäri todella nopeasti. Korjasin tämän asettamalla kameran katsomaan huoneen keskipistettä ennen kuin kameraa käännetään.

26.3.2019, tiistai

Korjasin ohjelmistossa ilmeneviä virheitä: Deco-peliobjektit jäivät paikoilleen, kun käyttäjä valitsee materiaaleja huoneeseen. Joissain tapauksissa ne jäivät siis leijumaan ilmaan. Ratkaisin tämän piilottamalla deco-peliobjektit, kun käyttäjä on materiaalivalintatilassa. Korjasin myös ongelman takaisinpäin-painikkeen kanssa, sillä jos materiaalilistaan ei tullut mitään materiaaleja, painike meni taaksepäin liikaa. Ratkaisin tämän tarkistamalla oikein, onko paneeli päällä vai ei, sillä olin tarkistanut sen väärällä tavalla ennen.

28.3.- 3.4.2019

Olin sairaslomalla.

4.4.2019, torstai

Aloitin päivän tarkastelemalla, mitä olin tehnyt ennen sairaslomaani, sekä mitä tehtäviä olin kirjoittanut ylös. Tämän jälkeen minua pyydettiin palaveriin, jossa sain palautetta UI:n tilanteesta. Palaverin jälkeen aloitin tekemään minulle palaverissa annettua tehtävää: Kun jokin materiaalipeliobjekti valitaan, käyttäjä haluaa kameran kohdistavan kohti valittua kohdetta. Koska kamera kiertää ankkuripistettä materiaalinvalintamoodissa, toteutin tämän niin että ankkuripiste siirretään peliobjektin bounds-arvojen keskipisteeseen. Tämä toimi ihan hyvin, joten tein metodille tuen bundle-objekteja varten. Metodi siis osaa nyt myös hakea keskipisteen esim. kaappirivistä, eikä vain tarkenna yksittäiseen kaappiin.

5.4.2019, perjantai

Siirryin korjaamaan virhettä seinienpiilotusmetodissani: Seinät tarkistavat mikä on seinän etu- ja takaosa laskemalla sen kameran ankkurin kohdasta. Tilanteissa, joissa seinä saattoi olla ankkurin vieressä, seinät saattoivat kadota, vaikka niiden ei pitäisi.

Korjasin ongelman kehittämällä metodin, joka ensin tarkistaa, mikä on seinän etuosa, käyttämällä raycastia. Seinän bounds-arvoista lasketaan seinän keskiosa, sekä vasen tai oikea seinän kulma. Tähän kulmaan luodaan raycast, jolla kokeillaan, onko seinän vieressä lattia. Jos lattia löytyy,

luodaan seinän keskelle etupuolelle toinen vektori, jonka avulla voidaan laskea seinien piilotuksen. Jos lattiaa ei löydy, vektori vain siirretään toiselle puolelle seinää, ja sitä käytetään testaamiseen.

Viikko 8, 25.3.- 26.3. ja 4.4.- 4.5.

Olin viikon sairaslomalla, joten minulla ei tällä viikolla ollut kamalasti erikoista tekemistä. Osaamiseni ei ole siis kehittynyt suuntaan taikka toiseen.

Isoin asia, jonka jouduin selvittämään, oli löytää covering-peliobjektit, eli seinien etu- ja takaosa. Sain tämän kuitenkin selvitettyä melko nopeasti, sillä se oli vain enimmäkseen vektorien laskemista.

Statuspalkin tekeminen normaalin valikon tilalle oli melko vaivatonta, sillä sen sijaan että olisin tehnyt kaiken uudestaan statuspalkkia varten, se on vain erillinen valikko, joka käsittelee piilossa pysyvää päävalikkoa. Minulta pyydettiin mahdollisimman yksinkertaista käyttöliittymäsuunnittelua, joten statuspalkissa näkyy tällä hetkellä vain kaksi asiaa, tyylit/teemat, joille ei ole vielä toiminnallisuutta sekä eri kameratilojen valinnat. Koska statuspalkki käyttää eri dropdown-valikoita tilan ja tyylin vaihtamiseen, aktivoitu tyyli sekä tila on koko ajan näkyvissä. Tämä helpottaa käyttäjää, sillä hän tietää heti missä tilassa ohjelmisto on, eikä hänen tarvitse availla valikoita.

Verrattuna vanhaan käyttöliittymäsuunnitteluun, jossa käytettiin päävalikkoa, tämä on paljon yksinkertaisempi sekä helpompi käyttää muiden mielestä. Vanha päävalikko toimii myös ihan hyvin, mutta sen mukana tuli liikaa ylimääräisiä klikkauksia.

8.4.2019, maanantai

Minua pyydettiin viimeviikolla tekemään materiaalipaneelille muutoksia, ettei se aukea esinelistan päälle, vaan aukeaisi esinelistan sisälle valitun esineen alle. Aloin toteuttamaan tätä, mutta huomasin, että minun pitää tehdä suurempia muutoksia, jotta esinelista tuntuisi toimivammalta. Kaksi listaa sisäkkäin on mielestäni huono asia, varsinkin kun molemmilla on omat vierityspalkkinsa. Minun tulee luultavasti muuttaa materiaalipaneeli sellaiseksi, että se avautuu dropdown-listana. En kuitenkaan aloittanut suunnittelemaan tätä vielä, sillä huomenna on tarkoitus pitää palaveri, jossa päätetään, millainen lopullinen käyttöliittymä on. Sen sijaan siirsin

materiaalipaneelin aukeamaan esinelistan viereen, sekä korjailin löytämiäni pieniä virheitä ohjelmistossa.

9.4.2019, tiistai

Jatkoin käyttöliittymän parissa työskentelyä ja siirsin statuspalkin paikan eri kohtaan (Kuvio 8). Tällä tavoin valikot näyttävät entistä yhteneväisemmiltä, sekä käyttäjän fokus pysyy koko ajan vasemmassa laidassa, kun hän tekee muutoksia ja valintoja.

KUVIO 8. Statuspalkki valikon yläpuolella

Korjasin myös ongelman kamerassa: Valittaessa jonkin peliobjektin, kamera saattoi pyörähtää 360 astetta ympäri, vaikka sen ei tarvitsisi. Vanhassa koodissani käytin EulerAngles-dataa kameran kääntämiseen, mutta vaihdoin sen Quaternioneihin, joka toimi tässä tilanteessa paremmin, sillä Unityn ohjeiden mukaan, Quaternion.Lerp-ominaisuus hakee aina lyhimmän mahdollisen reitin, kun vektoria käännetään paikasta toiseen (Unity, 2019g).

10.4.2019, keskiviikko

Tavoitteenani oli etsiä ja korjata ohjelmistovirheitä, sillä haluan varmistaa ohjelmistolle hyvän pohjan ennen kuin minulta pyydetään enemmän toiminnallisuuksia ohjelmistoon. Korjasin myös metodin, jolla käännetään kamera kohti valittua peliobjektia, sillä se saattoi välillä katsoa väärään suuntaan. Tämän lisäksi lisäsin zoomausmetodin, joka pyrkii sommittelemaan peliobjektin mahtumaan näytölle siirtämällä kameraa taaksepäin: Metodi laskee valitun peliobjektin bounding-pisteiden suuruuden ja siirtää sen avulla kameraa kauemmaksi peliobjektista.

11.4.2019, torstai

Korjasin ongelman, jonka vuoksi valittu materiaalipeliobjekti ei korostanut materiaalipeliobjektilistasta omaa nimeään. Ongelma ilmaantui materiaalia vaihtaessa, koska kyseinen materiaalipeliobjekti resetoidaan ja siinä samalla se menettää kaikki lapsipeliobjektit, mukaan lukien korostus-lapsipeliobjektin. Sen sijaan että ääriviivaus tehdään omaan peliobjektinsa, lisäsin sen suoraan materiaalipeliobjektiin. Tämä teki ääriviivauksesta paljon vakaamman.

Tämän lisäksi tein kameran kohdistusmetodiin tarkistuksen, onko valittu materiaalipeliobjekti lattia, seinä, vai katto ja kameraa täten käännetään oikeassa kulmassa kohti peliobjektia. Tein myös pienen metodin, jonka avulla huoneiden ja peliobjektien ääriviivat häivytetään päälle.

12.4.2019, perjantai

Keskityin tänään enemmän pienempiin tehtäviin, kuten koodin siivoamiseen ja pienien ongelmien korjaamiseen. Minulta pyydettiin parempaa kameran kuvakulmaa, kun käyttäjä valitsee lattian tai katon, joten koetin hienosäätää niitä paremmaksi. Kamera laskee pienen offsetin käännettäessä kamera kohti valittua peliobjektia, jonka avulla kamera pyrkii siirtämään kameraa hieman lähemmäksi tai kauemmaksi valitusta peliobjektista, mutta sitä pitää testata vielä eri asunnoissa.

8.4.- 12.4.2019, Viikko 8

Pääsin takaisin normaaliin työtahtiini tällä viikolla, sekä mielestäni osaamiseni on kehittynyt hieman, sillä huomaan mahdolliset ongelmat paljon nopeammin. Täten voin korjata osan ongelmista ennen kuin niitä edes oikeasti tulee.

Isoimmat ongelmat ilmenivät kameran kääntämisessä oikeaan suuntaan, sillä en oikein tiennyt, mitä minun tulisi käyttää kameran kääntämiseen, vektoreita vai quaternioneja. Ensiksi kamera siirretään pisteeseen, joka lasketaan peliobjektin bounding-boxin keskipisteestä, mutta siirretään pois peliobjektin sisältä liikuttamalla sitä peliobjektin forward-akselia pitkin. Sen jälkeen käännetään kamera kohti peliobjektia.

Kokeilin aluksi kameran kääntämistä varten muuttamalla Unityn dokumentaatiossa esiteltyä EulerAngles-vektoria, jota käytetään peliobjektin kääntämiseen oman akselinsa (x, y tai z) ympäri (Unity, 2019h). Yritin kääntää kameraa käyttämällä sen y-akselia ja muuttamalla sen käänteiseksi peliobjektin y-akseliksi (y*-1). Tämä toimi hyvin, jos peliobjekti oli sivuttain, eli 90 tai -90 asteessa, mutta kaikissa muissa tapauksissa kamera saattoi kääntyä väärin. Totesin ettei tämä ole luotettava ratkaisu ja aloin tutkimaan quaternioneja.

Käyttämällä quaternioneja saan aina lyhimmän mahdollisen kääntämisreitin kohti peliobjektia, sillä sen sijaan että käännämme peliobjektia oman akselinsa ympäri, peliobjekti käännetään neljän eri akselin avulla (x, y, z, w). Vältämme myös Unityn dokumentaatiossa mainitun Gimbal Lock-ongelman, eli käännettäessä peliobjektia ympäri se ei vahingossa mene uudelleen samaan suuntaan kuin se oli aiemmin (Unity, 2019i).

Kääntäessäni kameran kohti peliobjektia, käytän Quaternion.LookRotation-ominaisuutta, jonka avulla kameran forward-akseli käännetään kohti mitä tahansa tahansa vektoria (Unity, 2019j). Jotta saan kameran kääntymään kohti valittua peliobjektia, syötän tälle metodille peliobjektin forward-vektoriin ja käännän sen päinvastoin. Vaihtoehtoisesti voin myös käyttää peliobjektin back-vektoria. Täten kamera käännetään aina peliobjektin takaosaa varten, joten kamera katsoo aina suoraan peliobjektiin.

Koska quaternionien avulla peliobjektien kääntäminen on paljon sulavampaa, tulen käyttämään niitä jatkossakin. Jos tarvitsen vain nopean peliobjektin käännöksen tiettyyn suuntaan, tietyn asteiden verran, tietyn akselin ympäri, esim. oikealle 90 astetta, käytän luultavasti siihen EulerAngles-vektoria.

15.4.2019, maanantai

Yritin selkeyttää ja puhdistaa koodistani turhaa toistoa, jotta sen lukeminen olisi helpompaa. Sen jälkeen muokkasin esinelistan painikkeita niin, että painikkeen voi laittaa päälle ja pois. Aiemmin kun painiketta klikkasi uudelleen, lista eri materiaaleista latautui uusiksi. Halusin tähän tilalle ratkaisun, jonka avulla listan pystyy myös sulkemaan painamalla samaa painiketta uudelleen, kun lista on auki. En käyttänyt Button-komponentin sijasta Toggle-komponenttia, sillä olisin joutunut muokkaamaan paljon koodia sen vuoksi. Ongelma siis ratkaistiin muutamalla pienellä muutoksella: Joka painikkeen UIElement-komponentille tallennetaan, onko sillä avattu paneeli vai ei ja painiketta

painaessa se tarkistetaan. Jos paneeli on jo avattu samalla painikkeella, lista eri materiaaleista suljetaan.

16.4.2019, tiistai

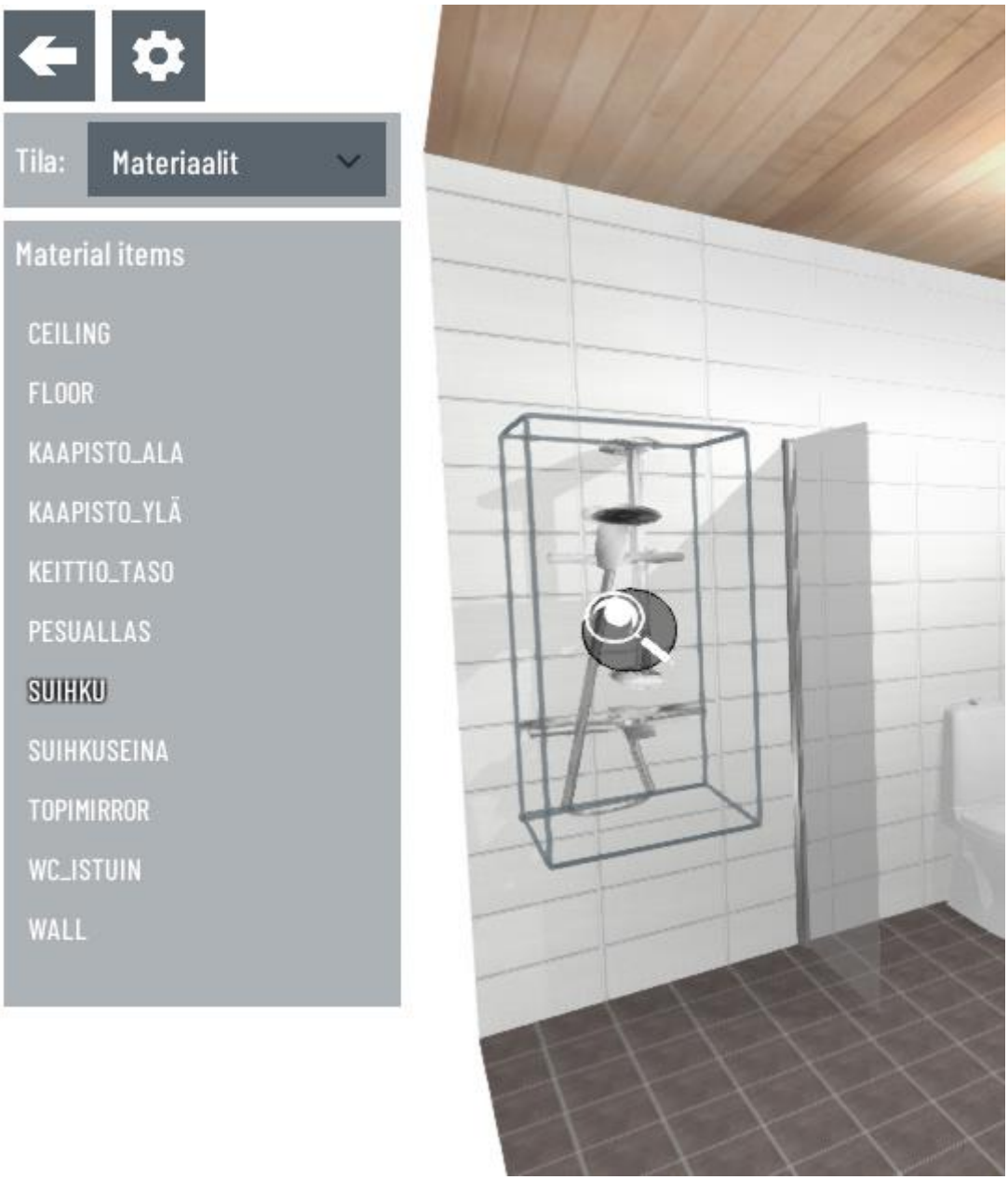
Aloitin materiaalipaneelin siirtämisen esinelistan sisälle, jotta listaa olisi helpompi käyttää. Onnistuin muokkaamaan nykyistä materiaalipaneelia niin, että se aukeaisi esinelistan sisälle (Kuvio 9).

KUVIO 9. Materiaalivalikko esinelistan sisällä

Tämän kanssa tuli kuitenkin ongelmia, sillä koodin pitää laskea oikea koko esinelistalle, jottei esineiden nimet sekä materiaalien kuvat mene päällekkäin. Tämän lisäksi materiaalipaneeli pitää sulkea oikealla tavalla, jottei mikään jää näkymään väärin.
En kuitenkaan tehnyt tätä menetelmää kokonaan valmiiksi, sillä minulle on tulossa toteutusohjeet, kuinka materiaalipaneeli halutaan tehdä valmiiseen tuotteeseen. Tallensin siis kyseisen koodin talteen, jotta saan sen nopeasti takaisin käyttööni, jos sitä tarvitaan myöhemmin. Minun tulee luultavasti kuitenkin tehdä materiaalipaneeli kokonaan uusiksi, sillä vanha materiaalipaneeli vaatii enemmän töitä, jotta sen saa taipumaan tähän käyttöön.

17.4.2019, keskiviikko

Kehitin ohjelmistolle ominaisuuden, jonka avulla voidaan lisätä kuvake (Kuvio 10) valittuun huoneeseen tai esineeseen. Kuvake ilmaisee, että käyttäjän tulee klikata esinettä tai huonetta uudelleen, jotta kamera siirretään kohteeseen.



KUVIO 10. Suurennuslasi-ikoni

Tietokoneella käytettävässä versiossa tätä ei periaatteessa tarvita, sillä käyttäjä voi liikuttaa hiiren kohteen päälle ja esine korostetaan, mutta mobiililaitteilla tämän tyylinen varmistus auttaa välttämään vahinkopainalluksia.

18.4.2019, torstai

Kehitin ohjelmiston materiaalivalinnan esinelistaan ratkaisun, jonka avulla voidaan hakea huoneessa näkyvät ovet. Koska seinät ja ovet ovat saman peliobjektin aliobjekteja, pystyin tätä hyödyntämällä hakemaan seinästä ovet. Tämä ei ole kuitenkaan hyvä, lopullinen ratkaisu, mutta

kehitin sen sitä varten, että voin säätää kameran toimivaksi ovien kanssa. Lisäksi korjasin myös virheen ohjelmistossa, jonka vuoksi esinelistan eri esineiden päälle sekä pois painallus ei toiminut kunnolla.

Viikko 9, 15.4.- 18.4.2019

Tämä viikko oli aika rauhallinen, joten käytin sen enimmäkseen vanhan koodin tutkimiseen sekä vanhojen virheiden ratkaisuun. Osaamiseni on mielestäni pysynyt ennallaan.

Mielenkiintoisin ongelma tällä viikolla oli ikonin lisääminen valitun esineen tai huoneen keskelle. Ratkaisin ongelman seuraavanlaisesti: Kun käyttäjä klikkaa mitä tahansa huonetta tai esinettä, se määrätään valituksi ja käyttäjälle näytetään jokin ikoni, tässä tapauksessa suurennuslasi. Kun käyttäjä klikkaa huonetta tai esinettä uudelleen, kamera siirretään huoneeseen tai esineen vierelle.

Ikoni lisätään huoneeseen etsimällä huoneen lattian keskipiste, johon luodaan uusi peliobjekti, eli ikoni. Tein ikonin luomista varten oman skriptin, joka lisätään uuteen peliobjektiin sen luomisvaiheessa. Skripti lisää peliobjektiin SpriteRenderer-komponentin, jonka avulla voidaan renderöidä mikä tahansa 2D-kuva 3D-maailmassa. Peliobjekti käännetään kohtisuoraan kohti kameraa, sekä sille lisätään materiaali, joka käyttää samanlaista shaderia kuten LineRendererit, eli ikoni näkyy missä tahansa tilanteessa muiden peliobjektien läpi.

Kun käyttäjä valitsee huoneen sijasta esineitä, lisätään ikoni esineen boundingboxin keskipisteeseen. Ikoni myös kääntyy kohti käyttäjän kameraa, jotta ikoni näkyy aina oikealla tavalla. Tämä toimi hyvin, mutta ikoni oli liian iso, jos kameraa liikutti lähemmäksi. Lisäsin siis ikoniin metodin, joka laskee etäisyyden kameran ja ikonin välillä, jakaa sen kymmenellä ja lisää sen sitten ikonin uudeksi kooksi. Tämä toimi todella hyvin, mutta jouduin myös käyttämään Mathf.Clamp-ominaisuutta, jotta ikoni ei mene liian pieneksi tai suureksi.

En löytänyt tähän ongelmaan valmiita toimintamalleja, joten keksin ratkaisun itse. Ratkaisu on mielestäni toimiva.

23.4.2019, tiistai

Minua pyydettiin tekemään VR-ohjelmistoon ohjelmiston sulkemispainike, joka avaa ensin ikkunan, jossa käyttäjältä kysytään, haluaako hän varmasti sulkea sovelluksen. Painikkeen tekeminen oli yksinkertaista, sillä lisäsin vain uuden painikekomponentin käyttöliittymään. Ikkunan avaaminen oli hieman monimutkaisempi, koska se oli tehty 2D-käyttöliittymää varten, joten jouduin tekemään siihen muutoksia, jotta saan sen VR-käyttöliittymää varten toimivaksi. 2D-käyttöliittymän elementit eivät toimi VR-käyttöliittymässä, sillä Unityn ohjeiden mukaan, VR näyttää vain 3D-avaruudessa olevia elementtejä, eikä piirrä ruudun päälle 2D-elementtejä (Unity, 2019k).

24.4.2019, keskiviikko

Käytin aamun VR-käyttöliittymän viimeistelyyn, jossa korjasin virheen, jonka vuoksi käyttäjä pystyi klikkaamaan tiettyjen käyttöliittymäelementtien läpi. Virhe johtui ylimääräisestä GraphicRaycaster-komponentista, joten elementti luki käyttäjän syötteen kahdesti. Iltapäivällä siirryin aivan eri asiaan, API-päätepisteisiin (endpoint). Tämä on minulle kokonaan uutta, ja minua pyydettiin tekemään API-endpoint, jonka avulla pystyn hakemaan huoneen kaikki esineet, myös ovet ja ikkunat. Tämän API-endpointin avulla voin siis lisätä materiaalipeliobjekti-listaan kaikki huoneen ovet ja ikkunat. Sain API-endpointin valmiiksi, tosin työkaverini teki koodista hieman nopeamman. Hän myös neuvoi minulle, miten VSCode-ohjelmistoa voi käyttää Git-versiohallinnan kanssa.

25.4.2019, torstai

Koetin soveltaa tekemääni API-endpointtia sovelluksessa ja koettaa etsiä sen avulla valitun huoneen ovet. API-endpoint kyllä toimii, mutta kävi ilmi, että käyttäjän täytyy aina kirjautua sisään ennen kuin API-kutsuja voi tehdä. Testausvaiheessa minun pitää käyttää siis jotain kovakoodattua käyttäjätunnusta, että voin kokeilla käyttää endpointia.

Sain myös sähköpostin, jossa oli muutamia käyttöliittymään tehtäviä parannusehdotuksia. Parannusehdotukset olivat pieniä, kuten esim. “piilota ja lukitse hiiri kameraa kääntäessä” joten siirryin niiden pariin. Opin kuitenkin hiiren lukitsemisesta, että selaimet eivät anna lukita hiirtä, ellei sovellus ole koko näytön kokoisena, tai kun käyttäjä itse laukaisee OnButtonDown-metodin (Unity, 2019l).

26.4.2019, perjantai

Jatkoin seuraavasta parannusehdotuksesta, eli aloin tekemään indikaattoria, jonka avulla käyttäjä näkee, mihin kohti huoneistoa kamera siirtyy. Kun käyttäjä klikkaa lattiaa tai seinää, piirretään pieni ympyrä (Kuvio 11) siihen kohtaan mihin kamera seuraavaksi liikkuu.

KUVIO 11. Liikkumisympyrä

Ympyrä pysyy paikoillaan kameran liikkuessa. Tein tämän käyttämällä apunani aiemmin tekemääni MouseHover-metodia, joka laskee mihin käyttäjä osoittaa hiirellään. Indikaattorin tekeminen oli hyvin yksinkertainen tehtävä loppujen lopuksi. Sen jälkeen tein kovalla kiireellä VR-ohjelmistoon pieniä UI-muutoksia, kuten painikkeiden siirtämisen eri paikkaan, sekä ominaisuuden ohjaimen Back-painikkeelle, jonka avulla käyttäjä voi kirjautua ulos.

23.4.- 26.4.2019, viikko 10

Opin tämän viikon aikana API-endpointien kehityksen perusteet, joka oli minulle melko uusi asia. Käytimme niiden kehittämiseen Javascript-kieltä, sekä Postman-ohjelmistoa. Postman on ohjelmisto, jonka avulla voi helposti tarkistaa, toimiiko joku tietty API-lauseke tai osoite. Sen käytön oppiminen ei vienyt kauaa ja käytin sitä apuna testatessani omaa endpointiani.

Endpointin ohjelmoinnissa haemme tietokannasta tarvittavat asiat, minun tapauksessani asunnon huoneet sekä huoneiden huonekalut. Endpoint palauttaa kaiken tiedon JSON-tiedostona. Saadakseni tiedon selkeästi luettavaksi, endpoint palauttaa ensin huoneen ja luo huoneelle oman arrayn, jonka sisälle huoneen huonekalut yms. listataan. Tämä tehdään asunnon jokaiselle huoneelle.

Nyt voin käyttää tätä endpointia ohjelmistossa, kun haluan esimerkiksi hakea kaikki huoneen ovet tai ikkunat, joita ei saa Unityn hierarkian kautta, sillä Unityssä ne ovat seinien lapsiobjekteja. Tämä johtuu siitä, ettei seiniä generoida huonekohtaisesti.

Opin myös käyttämään VSCode-editoria Gitin kanssa, sillä ohjelmisto kytkee itsensä Gitiin automaattisesti asennuksen aikana. Microsoftin ohjeiden mukaan VSCodessa on komentotulkki, jonka saa auki painamalla Ctrl + Shift + P (Microsoft, 4.4.2019b). Tähän komentotulkkiin kirjoittamalla esim. “Git Push” VScode käyttää Gitin push-komentoa ja lähettää tekemäni muutokset repositioon.

Minulla ei tullut erityisiä suurempia ongelmia tällä viikolla, sekä minulla ei ole verrattavia toimintamalleja API-endpointien tekemiseen tai VSCoden käyttämiseen, sillä seurasin työkavereideni neuvoja asiassa.

4 POHDINTA

Olen mielestäni kehittynyt paljon työssäni. Ymmärrän ja osaan enemmän ohjelmointia kuin työt aloittaessani, sekä olen pyrkinyt kehittämään vuorovaikutustaitojani. Osaan kirjoittaa selkeämpää koodia, sekä etsiä ja kehittää ratkaisuja ongelmiin. Uskallan myös kysyä enemmän kysymyksiä, enkä pelkää sosiaalisia tilanteita.

Uusia menetelmiä minulle ovat esim. tapahtumat, sekä C#-ohjelmointimallit, joiden käytön olen oppinut hiljattain. Koska työhöni ei löydy usein suoraan erilaisia ratkaisumalleja, olen oppinut kehittämään niitä itse tarvittaviin tilanteisiin. Hyvänä esimerkkinä toimii minkä tahansa peliobjektin pitäminen kamerassa, jonka laskemiseen käytän kameran frustum-pintoja, sekä seinien piilottaminen käyttämällä apunani dot-productia, joka lasketaan kameran ja seinissä olevien kiintopisteiden koordinaateista.

Mielenkiintoisin uusi asia minkä opin, olivat tapahtumat, joiden avulla voin kutsua mitä tahansa metodia mistä tahansa luokasta, sekä kuinka tapahtumille voi syöttää useampia metodeja, jotka voidaan kaikki laukaista yhtäaikaisesti. Tämä on jäänyt mieleeni hyvin ja tulen varmasti käyttämään sitä jatkossakin, sillä sen avulla saan pidettyä koodiani puhtaampana. Sen sijaan, että viittaan jokaiseen haluamaani luokkaan tai etsin niitä skenestä, voin tehdä tapahtuman, joka laukaisee metodin mistä tahansa luokasta tarvittaessa.

Olen myös pitänyt mielenkiintoisena sitä, kuinka paljon työtä voi kulua pelkän käyttöliittymän parissa. Saadakseni kaikki uuden käyttöliittymän elementit toimimaan yhdessä, minun oli tehtävä paljon muutoksia vanhaan koodiin. Olen hyödyntänyt työn analysointia oppimisen tukena, sillä uudet opitut asiat jäävät mieleeni paljon paremmin, kun kerron niistä.

LÄHTEET

Microsoft 2012. What is a view frustum? Saatavissa: https://docs.microsoft.com/en-us/previous-versions/windows/xna/ff634570(v%3dxnagamestudio.42). Hakupäivä 22.3.2019.

Microsoft 2019a. Dictionary<TKey,TValue> Class. Saatavissa: https://docs.microsoft.com/en-us/dotnet/api/system.collections.generic.dictionary-2?view=netframework-4.7.2. Hakupäivä 19.3.2019.

Microsoft 2019b. User Interface. Saatavissa: https://code.visualstudio.com/docs/getstarted/userinterface#_command-palette. Hakupäivä 26.4.2019.

Stack Overflow 2019. What is the best way to iterate over dictionary? Saatavissa: https://stackoverflow.com/questions/141088/what-is-the-best-way-to-iterate-over-a-dictionary. Hakupäivä 19.3.2019.

Unity 2019a. Designing UI for Multiple Resolutions. Saatavissa: https://docs.unity3d.com/Manual/HOWTO-UIMultiResolution.html. Hakupäivä 8.2.2019.

Unity 2019b. Scripting API: Mesh.vertices. Saatavissa: https://docs.unity3d.com/ScriptReference/Mesh-vertices.html. Hakupäivä 11.2.2019.

Unity 2019c. Bounds.Intersects. Saatavissa: https://docs.unity3d.com/ScriptReference/Bounds.Intersects.html. Hakupäivä 21.2.2019.

Unity 2019d. Vector3.Dot. Saatavissa: https://docs.unity3d.com/ScriptReference/Vector3.Dot.html. Hakupäivä 22.2.2019.

Unity 2019e. GeometryUtility.CalculateFrustumPlanes. Saatavissa: https://docs.unity3d.com/ScriptReference/GeometryUtility.CalculateFrustumPlanes.html. Hakupäivä 22.3.2019.

Unity 2019f. GeometryUtility.TestPlanesAABB. Saatavissa: https://docs.unity3d.com/ScriptReference/GeometryUtility.TestPlanesAABB.html. Hakupäivä 22.3.2019.

Unity 2019g. Quaternion. Saatavissa: https://docs.unity3d.com/ScriptReference/Quaternion.html. Hakupäivä 9.4.2019.

Unity 2019h. Transform.eulerAngles. Saatavissa: https://docs.unity3d.com/ScriptReference/Transform-eulerAngles.html. Hakupäivä 12.4.2019.

Unity 2019i. Rotation and Orientation in Unity. Saatavissa: https://docs.unity3d.com/Manual/QuaternionAndEulerRotationsInUnity.html. Hakupäivä 12.4.2019.
Unity 2019j. Quaternion.LookRotation. Saatavissa: https://docs.unity3d.com/ScriptReference/Quaternion.LookRotation.html. Hakupäivä 12.4.2019.

Unity 2019k. User Interfaces for VR. Saatavissa: https://unity3d.com/learn/tutorials/topics/virtual-reality/user-interfaces-vr. Hakupäivä 23.4.2019.

Unity 2019l. Cursor locking and full-screen mode in WebGL. Saatavissa: https://docs.unity3d.com/Manual/webgl-cursorfullscreen.html. Hakupäivä 25.4.2019.

Unity Answers 2019a. How to find the vertices of each edge on mesh. Saatavissa: https://answers.unity.com/questions/566779/how-to-find-the-vertices-of-each-edge-on-mesh.html. Hakupäivä 11.2.2019.

Unity Answers 2019b. How to get 8 vertices from bounds properties. Saatavissa: https://answers.unity.com/questions/29797/how-to-get-8-vertices-from-bounds-properties.html. Hakupäivä 18.2.2019.

Unity Answers 2019c. Compare arrays? Saatavissa: https://answers.unity.com/questions/380548/compare-arrays-1.html. Hakupäivä 20.2.2019.

Unity Answers 2019d. Edit camera culling mask. Saatavissa: https://answers.unity.com/questions/348974/edit-camera-culling-mask.html. Hakupäivä 21.2.2019.

Unity Answers 2019e. Is there a shader to only add an outline? Saatavissa: https://answers.unity.com/questions/60155/is-there-a-shader-to-only-add-an-outline.html. Hakupäivä 22.2.2019.

Unity Answers 2019f. Holding down UI mobile button. Saatavissa: https://answers.unity.com/questions/1262535/holding-down-ui-mobile-button.html. Hakupäivä 8.3.2019.

Unity Answers 2019g. Adding event to EventTrigger on runtime. Saatavissa: https://answers.unity.com/questions/1190581/adding-event-to-eventtrigger-on-runtime.html. Hakupäivä 14.3.2019.

Unity Answers 2019h. How can I know if a gameObject is 'seen' by a particular camera? Saatavissa: https://answers.unity.com/questions/8003/how-can-i-know-if-a-gameobject-is-seen-by-a-partic.html?childToView=928938#answer-928938. Hakupäivä 21.3.2019.

Unity Forums 2019. Smooth transition between perspective and ortographic modes. Saatavissa: https://forum.unity.com/threads/smooth-transition-between-perspective-and-orthographic-modes.32765/#post-217708. Hakupäivä 25.2.2019.