Oğuzhan Arslan

3D MODELLEN GEVISUALISEERD MET DE MICROSOFT HOLOLENS

Afstudeerscriptie over het visualiseren van 3D modellen met de Microsoft HoloLens in opdracht van de Haagse Hogeschool en Quintor

Quintor

DE HAAGSE HOGESCHOOL

3D MODELLEN GEVISUALISEERD MET DE MICROSOFT HOLOLENS

AFSTUDEERSCRIPTIE OVER HET VISUALISEREN VAN 3D MODELLEN MET DE MICROSOFT HOLOLENS IN OPDRACHT VAN DE HAAGSE HOGESCHOOL EN QUINTOR.

Auteur:	Oğuzhan Arslan
Project:	Afstudeerscriptie
	3D Modellen gevisualiseerd met de Microsoft HoloLens
Periode:	Februari – Juni 2017
Onderwijsinstelling:	De Haagse Hogeschool
Opdrachtgever:	Quintor

INFORMATIEPAGINA

ONDERWIJSINSTELLING

Onderwijsinstelling	De Haagse Hogeschool
	Afdeling IT & Design
	Opleiding Informatica
Adres	Johanna Westerdijkplein 75
	2521 EN Den Haag
Telefoon	070 – 445 8888
Afstudeercoördinator	E.M. van Doorn
1e Examinator	J.J. van der Hoek
2e Examinator	O. Zor

OPDRACHTGEVER

Afstudeerbedrijf	Quintor
Adres	Afdeling Den Haag
	Koninginnegracht 19
	2514 AB Den Haag
Telefoon	050 – 711 52 40
Contactpersoon	J. Tillema
Functie	CEO

STUDENT

Naam	Oğuzhan Arslan
Studentnummer	13018604

VOORWOORD

Voor u ligt de afstudeerscriptie "*3D modellen gevisualiseerd met de Microsoft HoloLens.*". Deze scriptie is geschreven in het kader van mij afstuderen aan de opleiding Informatica aan de Haagse Hogeschool en in opdracht van Quintor. Van Februari 2017 tot en met Juni 2017 is er over een periode van zestien weken gewerkt aan het project en het schrijven van de scriptie.

Samen met mijn bedrijfsbegeleider, Johan Tillema, is er gezocht naar toepassingsmogelijkheden voor de Microsoft HoloLens en is de opdracht voor deze scriptie bedacht. Het uitvoeren van het project was niet eenvoudig, mede omdat er gewerkt is met nieuwe technieken. Na een onderzoek, waarin meerdere problemen naar voren kwamen, is het gelukt een applicatie op te leveren die voldeed aan de opdracht.

Bij dezen wil ik graag mijn stagebegeleider, Johan Tillema, en mij begeleiders vanuit de opleiding, J.J. van der Hoek en O. Zor, bedanken voor de fijne begeleiding en ondersteuning tijdens dit project. Tevens wil ik mijn collega's en mede afstudeerders bij Quintor bedanken voor een fijne samenwerking.

Tot slot wil ik mijn ouders in het bijzonder bedanken. Hun wijsheid en ondersteuning motiveerde mij om door te zetten in mijn opleiding en deze scriptie tot een goed einde te brengen.

Ik wens u veel leesplezier toe.

Oğuzhan Arslan
's-Gravenhage, 2 Juni 2017

REFERAAT

Deze scriptie gaat over het koppelen van 3D modellen met de echte wereld door middel van de Microsoft HoloLens. Dit is een “Mixed Reality” headset van Microsoft en was op het moment van het project een nieuwe techniek.

Om het project uit te voeren is er door middel van de Spatial Mapping functionaliteit van de Microsoft HoloLens een scan van de omgeving van de gebruiker gemaakt, waarna benodigde data uit de scan gehaald kon worden. Deze data is vervolgens gebruikt om in een 3D model een passend kamer te vinden.

Vervolgens is deze kamer gekoppeld aan de echte wereld, wat vervolgens gevisualiseerd is met de Microsoft HoloLens. Bij het koppelen is er gebruik gemaakt van een verschillende wiskundige stellingen, zoals goniometrie, waarmee waardes zoals locatie en rotatie berekend zijn.

DESCRIPTOREN

.NET
3D
Agile
Augmented Reality
AutoDesk Maya
Base64
C#
CAD
DAE
Entity Framework
FBX
Goniometrie
HoloLens
Mesh Processing
Mixed Reality
MVC Framework
OBJ
ORM
Restful API
Spatial Mapping
Unity
UWP
Virtual Reality
Visual Studio

SAMENVATTING

De laatste tijd komen bepaalde termen als Virtual- en Augmented Reality steeds vaker voor. Sinds kort heeft Microsoft daar een nieuwe term bij gevoegd, namelijk Mixed Reality. Bij deze term is er ook een nieuwe Headset op de markt gebracht door Microsoft die gebruik maakt van Mixed Reality, namelijk de Microsoft HoloLens. Quintor heeft al vroeg een HoloLens weten te bemachtigen en zoekt dan ook naar toepassingsmogelijkheden van deze HoloLens. Eén van deze projecten is hoe een model (van een gebouw) gekoppeld kan worden aan de echte wereld door middel van de HoloLens.

Om deze vraag te kunnen beantwoorden is er onderzoek gedaan naar nieuwe technieken en hoe deze technieken ingezet kunnen worden om het doel te behalen. Deze onderzoeken zijn opgezet met een deskresearch, waarna de bevindingen met een praktijkgericht onderzoek, onderzocht zijn en een conclusie getrokken is.

Uit deze onderzoeksbevindingen is een architectuur gemaakt voor de uiteindelijke applicatie. Tijdens dit proces is een andere bevinding gemaakt, namelijk dat de HoloLens geen filesystem heeft, waardoor er ook een server gemaakt moest worden. Op deze server kan de gebruiker modellen kunnen uploaden, die de HoloLens vervolgens van de server kan opvragen.

Na het maken van de architectuur van de verschillende systemen, is er begonnen met het bouwen. Hierbij is als eerst de server gemaakt. Om de modellen van de server te kunnen halen is er in de HoloLens applicatie als eerst gewerkt aan de Data Package. Deze package zorgt ervoor dat de modellen van de server gehaald kunnen worden en dat de modellen in het vervolg van de applicatie gebruikt kunnen worden.

Nadat de Data Package gebouwd was, zijn de andere onderdelen van de applicatie gebouwd zoals de Interaction Package (waarmee de gebruiker acties kan uitvoeren), Building Factory (die ervoor zorgt dat het model ingeladen en gebruikt kan worden), Spatial Package (verantwoordelijk voor de technieken die de HoloLens zelf biedt zoals Spatial Scanning en Processing) en de Controllers (die zorgen voor het selecteren van de kamers).

In een periode van zestien weken is het project gerealiseerd en de uitkomst van het project is dan ook dat het mogelijk is modellen te koppelen aan de echte wereld door middel van de Microsoft HoloLens. Het project heeft uiteindelijk een applicatie voor de HoloLens opgeleverd, waarin er een model van de server gehaald kan worden en vervolgens uitgelezen kan worden. Als een kamer in het model overeenkomt met de kamer waar de gebruiker zich bevindt, wordt de kamer ingeladen en gevisualiseerd met de HoloLens.

INHOUDSOPGAVE

1 INLEIDING

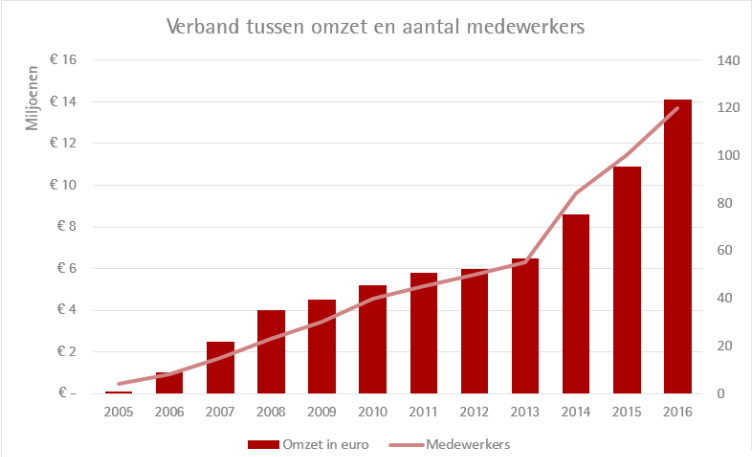
De afgelopen jaren komen termen als Virtual Reality en Augmented Reality steeds vaker voor. Hierbij kan er natuurlijk gevraagd worden, wat is het verschil tussen de twee. De makkelijkste uitleg zou zijn dat je bij Virtual Reality in een computer gegenereerde wereld zit en bij Augmented Reality, Hologrammen in de echte wereld geplaatst worden.

Hierbij heeft Microsoft medio 2015 (Myerson, 2015) een nieuwe term aan toegevoegd, namelijk Mixed Reality. Dit houdt in dat de gebruiker een wereld evenaart dat zowel gevormd is uit de echte wereld als een computer gegeneerde wereld. Dit lijkt veel op Augmented Reality, maar er zit een subtiel verschil tussen en dat is dat bij Mixed Reality, ook rekening gehouden kan worden met de echte wereld. Hierdoor kunnen hologrammen bijvoorbeeld op vooraf berekende locaties geplaatst worden.

Deze nieuwe techniek is voor het eerst toegepast door de Microsoft HoloLens, waarvan medio 2016 (Kipman, 2016) de eerste Developer Edition is uitgebracht. Deze Developer Edition is bedoeld voor ontwikkelaars, met als doel het uitzoeken van toepassingsmogelijkheden van Mixed Reality door middel van de Microsoft HoloLens.

1.1 QUINTOR

Quintor is opgericht in 2005 en hierbij waren 4 medewerkers betrokken. Sindsdien is er een gezonde groei gemaakt en zijn er momenteel 120 medewerkers die samen sterk zijn voor een omzet van 14 miljoen euro, zie figuur 1.1. Hiernaast zijn er drie vestigingen door heel Nederland heen, namelijk Groningen (hoofdvestiging), Amersfoort en de recentere Den Haag.



Figuur 1.1: Verband omzet en aantal medewerkers (Quintor, n.d.)

Quintor is gericht op het gebied van Agile Software Development, Enterprise Java/ .NET technologie en mobile development. Hierbij is er focus op drie pijlers: Techniek en Architectuur, een hoogwaardig ontwikkelstraat en het Agile/Scrum proces.

Deze drie pijlers worden gebundeld in de zogenaamde Quintor Software Factory. In deze Software Factory worden inhouse projecten voor klanten uitgevoerd. De klanten van Quintor zijn werkzaam in veel verschillende sectoren, zoals de publieke-, energie- en financiële sector. Met klanten als ING, KPN, DUO, Eneco en TomTom laat Quintor zien een veelzijdig klantenbestand te hebben.

Naast de projecten die aangenomen worden voor klanten is Quintor ook geïnteresseerd in nieuwe technieken. Daarom wordt er bij Quintor ook onderzocht hoe deze nieuwe technieken werken en waar ze voor ingezet kunnen worden. Een voorbeeld van een nieuwe techniek is de Microsoft HoloLens. Om de toepassingsmogelijkheden van de Microsoft HoloLens bij klanten voor te kunnen stellen wordt er onderzocht hoe de HoloLens gebruikt zou kunnen worden in de verschillende sectoren.

Hier zijn een aantal onderzoeken opgezet bij Quintor met elk een eigen invalshoek, zoals de financiële markt en object herkenning. In dit document zal er ingegaan worden op een van deze onderzoeken, namelijk het koppelen van 3D modellen met de Microsoft HoloLens. Dit onderzoek is dan ook vooral gericht op de bouwsector.

1.2 MICROSOFT HOLOLENS

De Microsoft HoloLens is een Head-Mounted Display (HMD) dat samen met de nieuwe term “Mixed Reality” is geïntroduceerd. Deze term kan vergeleken worden met andere termen als Virtual- en Augmented Reality. Bij Virtual Reality ervaart de gebruiker een computer gegenereerde wereld (vrs, n.d.) en er is op zich geen affiniteit met de echte wereld. Augmented Reality daarentegen speelt af in de echte wereld (Mark Graham, 2013), waarbij er door middel van computer gegenereerde hologrammen een extra realiteit aan de wereld gevoegd kan worden.

Mixed Reality neemt dit nog een stap verder door de computer gegenereerde hologrammen ook de mogelijkheid te geven te reageren op de omgeving (Microsoft (Why HoloLens), n.d.). Hierbij kan er bijvoorbeeld gedacht worden aan een hologram van een schilderij. Bij Augmented Reality zou deze in de wereld getoond worden, waarbij de gebruiker het gevoel krijgt dat het echt in de wereld staat, maar vaak is de plaatsing van deze hologram op een manier waardoor het duidelijk wordt dat het geen echt object is. Bij Mixed Reality wordt deze schilderij hologram bijvoorbeeld getoond op een muur in de buurt van de gebruiker. Hierdoor voelt het hologram als een onderdeel van de echte wereld aan.

De HoloLens is een computer met vele extra’s om Mixed Reality mogelijk te maken. De extra sensoren en technieken (HoloSpecs, n.d.) naast de CPU, RAM en opslag zijn:

- Een HPU ontwikkeld door Microsoft
- 4 camera’s voor omgeving herkenning
- Een camera voor dieptemetingen
- Een camera voor opnames
- Een omgevingslicht sensor
- 2 beeldschermen (1 per oog)
- 4 microfoons

Figuur 1.2: HoloLens ontleed

1.3 LEESWIJZER

Dit verslag is geschreven op volgorde van uitvoering. Dit komt erop neer dat eerst de uitgevoerde onderzoeken besproken zullen worden, waarna de uiteindelijke gekozen tools en technieken uitgewerkt zullen worden. Vervolgens zal de architectuur toegelicht worden, waarmee vervolgens de uitvoering beschreven zal worden. Tot slot zal er een aanbeveling en conclusie gegeven worden.

Naast de beschrijving van de uitvoering zal er door het verslag heen ook een reeds aan hoofdstukken getroffen worden, waarin het einde van een sprint toegelicht zal worden. Deze hoofdstukken zullen extra informatie bevatten met betrekking tot keuzes die gemaakt zijn tijdens de ontwikkeling van het product. Deze keuzes zijn voort gekomen uit de demo’s die aan het einde van sprints gegeven zijn aan de opdrachtgever. Tijdens deze demo’s zijn koerswijzigingen en probleemoplossingen besproken met de opdrachtgever en deze keuzes zullen na een sprint hoofdstuk meegenomen worden in het vervolg van het verslag.

2 OPDRACHT

Quintor heeft verschillende onderzoek opdrachten voor de Microsoft HoloLens, maar dit document gaat over het visualiseren van 3D modellen met de Microsoft HoloLens. De probleemstelling en doelstelling van deze opdracht zal in dit hoofdstuk beschreven worden, evenals de criteria waar het eindresultaat aan moet voldoen.

2.1 PROBLEEMSTELLING

De bouw kent vele problemen, maar deze opdracht heeft betrekking tot bestaande bouwwerken. Het komt namelijk vaak voor dat oudere gebouwen gerenoveerd en of aangepast worden om te voldoen aan nieuwe wensen. Hierbij kan een gebouw gezien worden als een menselijk lichaam, waar niet alle informatie met het blote oog te zien is. Achter de muren van gebouwen lopen namelijk meters aan bekabeling en leidingen.

Om te voorkomen dat deze leidingen en kabels beschadigd raken, moeten de locaties van deze “elementen” vastgelegd worden, maar met de huidige gereedschappen is dit niet echt mogelijk. Het beste wat er gedaan kan worden is volgens schetsen die gemaakt zijn tijdens de bouw de locatie van deze elementen vastleggen en dit herhalen voor elk aanwezig element.

Om dit efficiënter te maken is er gedacht aan de Microsoft HoloLens en met name de “Mixed” Reality technologie, waarbij de huidige wereld aangevuld kan worden met computer gegenereerde elementen. Dit leidt tot de vraag of het mogelijk is modellen te koppelen aan de Microsoft HoloLens, zodat de gebruiker zonder te veel moeite meer informatie kan krijgen van de ruimte waar hij of zij zich op dat moment bevindt.

2.2 DOELSTELLING

Aan het eind van het project wordt een applicatie verwacht voor de Microsoft HoloLens, waarbij de gebruiker een model kan selecteren en dit model vervolgens op de juiste manier kan inladen. Door het inladen zal de gebruiker de informatie die zich in het model bevindt, zien in de echte wereld.

De opgeleverde applicatie moet werkend en presenteer baar zijn, waarbij het uiteindelijk wel gaat om een “Proof of Concept”. Verder wordt er verwacht dat de applicatie zo veel mogelijk verschillende technieken van de Microsoft HoloLens kan presenteren.

2.3 REQUIREMENTS

Voor het project zijn er een aantal requirements verkregen vanuit de opdrachtgever deze requirements zullen hier beschreven worden (tabel 2.1) en hoe belangrijk ze zijn voor het eindresultaat zal aangegeven worden. De Requirements zijn opgedeeld in drie soorten, namelijk User Requirements (UR) die duidelijk maken wat de uiteindelijke gebruiker moet kunnen doen, System Requirements (SR) welke de werkzaamheden van het systeem/applicatie beschrijven en Process Requirements (PR) die ervoor zorgen dat het project uitgevoerd wordt op/met een methodiek dat past bij het bedrijf. In tegenstelling tot de andere requirements zal er bij de laatste requirements geen prioritering zijn, aangezien dit niet functionele requirements zijn zal het gehele project eraan moeten voldoen. Daarom kunnen deze requirements ook gezien worden met de hoogste prioritering.

De prioritering zal bestaan uit Hoog, Midden en Laag. Deze prioriteit zal gebaseerd zijn op de functionaliteit van het systeem, waar functionaliteit dat essentieel is voor de werking een Hogere prioritering zal krijgen dan requirements die zullen zorgen voor een betere ervaring/luxe requirements.

Ook zal er aangegeven worden wanneer de Requirement toegevoegd is. Aangezien er gebruik gemaakt zal worden van een Agile Methodiek is het mogelijk dat de opdrachtgever de requirements aanpast of nieuwe requirements toevoegt. Deze requirements zullen na de sprint waar ze toegevoegd zijn meegenomen worden in het vervolg van het verslag.

ID	**Requirement**	**Toevoeging**	**Prioriteit**
UR01	De gebruiker moet modellen kunnen selecteren	Sprint 0	Hoog
UR02	De gebruiker moet een kamer kunnen scannen	Sprint 0	Hoog
UR03	De gebruiker moet commando's kunnen uitvoeren	Sprint 0	Midden
UR04	De gebruiker moet kamers kunnen selecteren	Sprint 0	Midden
UR05	De gebruiker moet stappen terug kunnen zetten	Sprint 6	Midden
UR06	De gebruiker moet zelf de hoeken kunnen aangeven	Sprint 6	Midden
SR01	De applicatie moet modellen kunnen uitlezen	Sprint 0	Hoog
SR02	De modellen moeten achteraf ingeladen kunnen worden	Sprint 0	Hoog
SR03	De modellen moeten de juiste data bevatten	Sprint 1	Hoog
SR04	De modellen moeten te gebruiken zijn in .NET en Unity	Sprint 1	Hoog
SR05	De modellen moeten toegankelijk zijn voor gebruikers	Sprint 1	Laag
SR06	De modellen moeten makkelijk in gebruik zijn	Sprint 1	Laag
PR01	De ontwikkeling moet op een Agile methodiek	Sprint 0	-
PR02	De ontwikkeling moet volgens het .NET framework	Sprint 0	-
PR03	De ontwikkeling moet gericht zijn op het eindresultaat	Sprint 0	-
PR04	De ontwikkeling zal zonder budget zijn (aangezien het gaat om een Proof of concept)	Sprint 0	-
PR05	De ontwikkeling zal alleen gericht zijn op rechthoekige kamers	Sprint 4	-

Tabel 2.1: Requirements

3 AANPAK

Het project zal worden gepland over een periode van 16 weken. Hierbij zal er gebruik gemaakt worden van een Agile methodiek en zullen er dus sprints van 2 weken gehouden worden. Aan het einde van elke sprint zal er met de producteigenaar besproken worden wat er in de laatste sprint gedaan is, wat er de komende sprint gedaan zal worden en wat voor problemen er zijn voorgekomen. Ook zal er geprobeerd worden een demo te geven van het resultaat zodat er zoveel mogelijk feedback verkregen kan worden waarmee het uiteindelijke product verbeterd kan worden. Uiteindelijk zullen er in totaal 8 sprints gehouden worden van elk 2 weken.

3.1 SCOPE

Gedurende het project zullen een aantal randvoorwaarden zijn waaraan voldaan moet worden. Deze randvoorwaarden zijn er om de scope van het project in een aanneembaar vorm te houden en zullen er dus vooral zijn om het eindresultaat te kunnen halen. Deze randvoorwaarden zijn:

- Er moet ontwikkeld worden met .NET C# (PR02)
  - De HoloLens is een product van Microsoft dus de development zal, met een groot kans, met een Microsoft Taal uitgevoerd worden.
- Gericht op werkend HoloLens applicatie (PR03)
  - De bijkomende producten zoals servers en 3D modellen zullen ontwikkeld worden, maar hierbij zal de opzet en uitvoering niet in acht genomen worden.

Om de doelstelling te behalen zullen er verschillende activiteiten uitgevoerd moeten worden zoals:

- Onderzoek naar de te gebruiken methodieken en tools
- Systeemarchitectuur
- Applicatie bouwen
- Oplevering van verschillende (tussen)producten

Door het project op te delen in verschillende categorieën kan er een duidelijker beeld verkregen worden van de uit te voeren taken. Hierbij is het belangrijk de doelstelling in gedachte te houden.

Dit betekent dus dat er ten eerste onderzoek gedaan moet worden naar de te gebruiken tools, zoals de engine waarmee ontwikkeld kan worden voor de Microsoft HoloLens. Vervolgens moet er een dataset gekozen worden die de data, dat uiteindelijk getoond gaat worden, vast kan houden. Als laatst zal er onderzocht worden hoe de datasets gekoppeld kunnen worden met de echte wereld.

Nadat deze onderzoeken uitgevoerd zijn, zal er een architectuur gemaakt worden van de verschillende systeemdelen, zoals de HoloLens applicatie. Hierbij kunnen tijdens het onderzoek ook nieuwe systeemdelen naar voren komen, die vooraf aan het project niet vast te stellen zijn. Indien dit het geval is zal ook hier een architectuur van gemaakt worden, waarbij het systeemdeel zo optimaal mogelijk is voor de werking van de HoloLens applicatie.

Nadat de onderzoeken en architectuur uitgevoerd zijn, zal er gebouwd worden volgens het architectuur.

Uit de uitvoering van deze activiteiten zullen vervolgens de volgende producten opgeleverd worden:

- Onderzoeksrapport 3D Engines
- Onderzoeksrapport 3D Modellen uitlezen met .NET
- Onderzoeksrapport koppeling van 3D modellen
- Architectuur systeemdelen
- Werkende Applicatie
- Handleiding Applicatie
- Documentatie Applicatie

3.2 PLANNING

Om dit allemaal in 8 sprints te doen zal er vooraf een globale planning gemaakt worden, zodat er bijgehouden kan worden of het project op schema loopt.

Sprint:	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**
Onderzoek* 3D engine HoloLens	X	X						
Onderzoek* 3D Modellen uitlezen .NET	X	X						
Onderzoek* koppeling van 3D modellen	X	X						
Architectuur 3D modellen		X	X					
Architectuur HoloLens Applicatie		X	X	X				
Overige architectuur				X	X			
Bouwen Systeem				X	X	X	X	
Testen & Verbetering							X	X
Schrijven documentatie en handleiding							X	X
Schrijven Afstudeerdossier	X	X	X	X	X	X	X	X

** Inclusief schrijven van onderzoeksrapport*

Tabel 3.1: Sprint Planning

4 ONDERZOEK

Tijdens het project zullen er meerdere tools en methodieken onderzocht worden. Deze onderzoeken zullen in dit hoofdstuk in detail opgenomen worden.

4.1 ONDERZOEKSOPZET

Zoals beschreven in paragraaf 3.1 zijn er een aantal onderzoeken die uitgevoerd moeten worden. Deze onderzoeken zullen antwoord geven op een aantal deelvragen die beantwoord moeten worden om de hoofdvraag te kunnen beantwoorden. De deelvragen zijn:

- Welke 3D Engine kan gebruikt worden om te ontwikkelen voor de Microsoft HoloLens?
- Welke 3D modellen kunnen gebruikt worden om gevisualiseerd te worden?
- Hoe kunnen de 3D modellen gekoppeld worden aan de echte wereld?

Deze deelvragen zullen gezamenlijk het antwoord geven op de hoofdvraag:

Hoe kunnen 3D modellen gevisualiseerd worden met de Microsoft HoloLens?

De antwoorden op deze vragen zullen niet direct uit literatuur of eerder uitgevoerde projecten beantwoord kunnen worden, aangezien de Microsoft HoloLens nog vrij nieuw is en de uiteindelijke onderzoek specifiek is voor de Microsoft HoloLens. Hierom zal er tijdens het onderzoek literatuuronderzoek en deskresearch gebruikt worden voor informatieverzameling, waarop verschillende gevonden bevindingen volgens praktijkonderzoek getest zullen worden. Hierbij zal er rekening gehouden worden met de requirements, waar aan het project moet voldoen, zodat er een goede keuze gemaakt kan worden voor het verloop van het project.

4.2 3D ENGINE IN GEBRUIK MET DE MICROSOFT HOLOLENS

Bij het bouwen van een applicatie voor de Microsoft HoloLens zal er gebruik gemaakt worden van een engine of framework. Het doel van deze engine of framework is de omgeving in te stellen die de gebruiker te zien zal krijgen door de HoloLens op te zetten.

4.2.1 VOORONDERZOEK

Bij het onderzoek naar welke engines of frameworks gebruikt zouden kunnen worden zijn er eigenlijk maar 3 naar voren gekomen en dat zijn Unity, Unreal Engine en DirectX. Bij het vooronderzoek is opgemerkt dat er veel informatie te vinden was over Unity. Over de andere twee was eigenlijk weinig van te vinden en de reden dat ze opgenomen zijn in het onderzoek is dat er beweerd werd (DaveVoyles, 2016), (DirectX, n.d.) dat ze werkend te krijgen waren met de HoloLens.

Naast een aantal requirements die in dit onderzoek meegenomen zullen worden, zullen ook een aantal extra requirements meegenomen worden. Deze requirements zullen geen invloed hebben op het resultaat van het project, maar wel op de ontwikkeling zelf. De extra requirements zullen er namelijk voor zorgen dat de ontwikkeling soepeler zal verlopen en dat problemen die zullen voorkomen makkelijker verholpen zullen worden.

	Requirement	**Prioritering**
1	Support voor de Microsoft HoloLens (PR03)	Hoog
2	Gebruiksvriendelijkheid	Midden
3	.NET C# ondersteuning (PR02)	Hoog
4	Te vinden informatie	Laag

Tabel 4.1: Engine Requirements

De laatste is vooral handig in het geval dat er problemen zouden optreden, aangezien een groter community betekent dat problemen sneller opgelost kunnen worden.

4.2.2 UITVOERING

UNITY

Als er gezocht wordt naar ontwikkeling voor de Microsoft HoloLens komt snel de HoloLens Academy naar voren. In deze academie worden alle verschillende technieken (die beschikbaar zijn in de HoloLens) uitgelegd en ook hoe ze te gebruiken zijn. Hierbij valt het op dat Unity de engine is die wordt aangeraden en gebruikt door Microsoft zelf. Aangezien Microsoft zelf de engine aanraadt is er samengewerkt met Unity om ondersteuning voor de HoloLens te bieden, dit is ook te merken. Vanuit Unity kan er namelijk direct gebouwd worden en uitgegeven worden naar de HoloLens.

Uit eerdere ervaringen met Unity was het ook al bekend dat Unity een van de eenvoudigste engines in gebruik is en dat het een lage leer curve heeft. Door een eenvoudige interface is het makkelijk verschillende taken uit te voeren en zal het dus makkelijk zijn om te ontwikkelen voor de HoloLens.

Alhoewel Unity zelf geschreven is in C++ (Unity Answers, 2010) heeft de ontwikkelaar de mogelijkheid te programmeren in C# of Javascript. Hierdoor voldoet de engine ook aan de .Net C# support requirement die is vastgesteld.

De laatste Requirement is de te vinden informatie en dit is ook een requirement waar Unity aan voldoet. De HoloLens heeft namelijk een online forum, waar ontwikkelaars die bezig zijn met de Microsoft HoloLens alle problemen en oplossingen kwijt kunnen. Buiten de forums die specifiek zijn voor de HoloLens heeft Unity zelf ook een hoog gebruikers hoeveelheid, waardoor problemen die Unity gerelateerd zijn makkelijk opgelost kunnen worden.

UNREAL ENGINE

Unreal Engine is een engine die ook vrij bekend is binnen de game ontwikkeling, maar door eerdere ervaringen met Unreal Engine was het al bekend dat het niet zou voldoen aan de derde requirement, namelijk C# ondersteuning. Om toch een duidelijk beeld te geven is er wel gekeken naar de andere requirements.

Volgens de bron was het mogelijk een applicatie te ontwikkelen voor de HoloLens, maar hierbij gaat het om een simpele Universal Windows Platform applicatie, wat een applicatie is dat zal werken op alle platformen die Windows gerelateerd zijn, zoals desktop, mobiel en de HoloLens. Deze applicatie zal echter niet de mogelijkheid hebben gebruik te maken van de technieken die de HoloLens te bieden heeft. Juist deze technieken zorgen ervoor dat er gebruik gemaakt wordt van “Mixed” Reality en is het support voor de HoloLens niet zoals verwacht.

Net als bij Unity heeft Unreal Engine een interface waar verschillende taken uitgevoerd kunnen worden, waardoor de gebruiksvriendelijkheid gelijk zal zijn aan Unity.

Als laatst is er de te vinden informatie, maar dit zag er niet veel belovend uit. Net als Unity heeft Unreal Engine een hoog gebruikers hoeveelheid, waardoor Unreal Engine gerelateerde problemen eenvoudig opgelost kunnen worden. Problemen die HoloLens gerelateerd zijn, zullen echter niet makkelijk achterhaald kunnen worden, aangezien de problemen die HoloLens gerelateerd zijn vooral voorkomen in Unity en de HoloLens ontwikkelaars geen gebruik maken van Unreal Engine.

DIRECTX

DirectX is in tegenstelling tot de andere twee engines, niet echt een engine. Het is een framework, waarbij er geen interface is. Hierdoor zullen alle taken die uitgevoerd moeten worden codematig uitgevoerd moeten worden. Dit heeft betrekking tot het tweede requirement, namelijk de gebruiksvriendelijkheid. De andere twee engines bieden namelijk de mogelijkheid een groot deel van de taken door middel van een interface uit te voeren, waardoor het makkelijker wordt voor de gebruiker.

Naast de HoloLens Academy die gericht is op Unity heeft Microsoft ook een aantal handleidingen die de ontwikkeling voor DirectX beschrijven. Deze handleidingen leggen uit hoe de technieken van de HoloLens gebruikt kunnen worden, waardoor ook echt “Mixed” Reality applicatie gebouwd kunnen worden, in tegenstelling tot Unreal Engine.

Naast de beschikbare documentatie dat geleverd is door Microsoft is er niet echt veel informatie te vinden over de ontwikkeling met DirectX. DirectX specifieke problemen kunnen geheid wel opgelost worden met behulp van de ontwikkelaars die bezig zijn met DirectX. Problemen die HoloLens specifiek zijn, kunnen echter niet makkelijk opgelost worden. Dit komt doordat er maar een aantal ontwikkelaars zijn die DirectX gebruiken voor HoloLens ontwikkeling.

De laatste requirement is de .NET C# ondersteuning. In de documentatie die Microsoft levert wordt gebruik gemaakt van C++, de taal waar DirectX ook in geschreven is. Het support voor C# is er niet en kan alleen bereikt worden door een omhulsel te schrijven, waarmee de C++ functionaliteit te bereiken is met C# code.

4.2.3 CONCLUSIE EN AANBEVELING

Het eindresultaat was na het onderzoek duidelijk en in tabel 4.2 is te zien hoe de engines in vergelijking met elkaar voldoen aan de requirements.

Requirement	**Unity**	**Unreal Engine**	**DirectX**
Support voor Microsoft HoloLens	Ja	Gedeeltelijk	Ja
Gebruiksvriendelijkheid	Ja	Ja	Nee
.NET C# ondersteuning	Ja	Nee	Nee
Te vinden informatie	Ja	Nee	Gedeeltelijk

Tabel 4.2: Engine Resultaten

Het is duidelijk dat Unity momenteel de beste engine is om te gebruiken bij het ontwikkelen voor de Microsoft HoloLens. Dit was ook wel te verwachten aangezien het de engine is dat ook door Microsoft wordt aangeraden. De andere twee engines waren werkend te krijgen met de Microsoft HoloLens, maar hiernaast hadden ze nog meer minpunten ten opzichte van Unity. Daarom kan er het beste gekozen worden om te ontwikkelen met Unity.

4.3 3D MODELLEN UITLEZEN MET .NET

Tijdens de opstart van het project zou er eigenlijk gewerkt worden met CAD-modellen, zodat de standaard voor de weergave van de bouw gebruikt kon worden. Dit was echter niet mogelijk, aangezien CAD-modellen geen data bevatten over de ligging van bijvoorbeeld leidingen en kabels. In CAD-modellen worden hooguit de locaties van kamer elementen, zoals deuren en ramen weergegeven.

SPRINT OVERGANG

Nadat ontdekt was dat CAD-modellen niet gebruikt konden worden was het einde van de eerste sprint bereikt. Hierbij is er een demo gegeven aan de opdrachtgever en is er besproken wat er in het volgende sprint uitgevoerd zou worden

SPRINT 1 (SPRINT 0)

De eerste sprint stond in het teken van uitzoeken van wat er gedaan moest worden en was dus vooral een initiatie fase, waarin het vervolg van het project uitgewerkt zou worden. Hierbij is er uitgezocht wat de opdracht eigenlijk inhoud en hoe deze opdracht verdeeld moest worden over het project periode.

Deze sprint is afgesloten met een demo aan de opdrachtgever waarin de uitgevoerde taken zijn besproken, zoals de product backlog, plan van aanpak en natuurlijk de eerste onderzoek. Hierbij is er ook besproken hoe het probleem met de CAD-modellen aangepakt zou kunnen worden.

Na een overleg met de opdrachtgever is er besloten geen gebruik te maken van CAD-modellen maar van 3D modellen die wel gebruikt kunnen worden. Hierbij zijn een aantal nieuwe requirements ten orde gekomen en is het onderzoek aangepast, waardoor eerst een bruikbaar model gevonden moest worden, waarna het project hervat zou worden.

SPRINT 2

De volgende sprint staat in het teken van het uitzoeken van verschillende modellen die zullen voldoen aan de requirements. Deze onderzoek zal tot in detail van het gebruik van de verschillende modellen gaan om te kunnen garanderen dat het project succesvol uitgevoerd kan worden.

Uit een voorafgaand onderzoek tijdens het project was er besloten gebruik te maken van Unity als visualisatie engine en zal er dus gekeken worden welke 3D modellen Unity ondersteund. Dit omdat Unity gebruikt kan worden om gestructureerde en bruikbare objecten te maken van de 3D modellen. Deze geïmporteerde modellen worden GameObjecten genoemd en deze GameObjecten kunnen makkelijk benaderd worden in .NET code.

In het vervolg van het onderzoek zal gebruik gemaakt worden van Unity. Om deze engine beter te begrijpen kan er gekeken worden naar paragraaf 5.1, waarin softwareontwikkeling in Unity wordt beschreven.

4.3.1 VOORONDERZOEK

Nadat er besloten was gebruik te maken van 3D modellen in plaats van CAD-modellen zijn er een aantal requirements toegevoegd die ervoor zullen zorgen dat een model gekozen wordt waarmee het doel behaald kan worden. Dit zijn requirements als:

- Kan het de benodigde data opslaan? (SR03)
- Is het te gebruiken met .NET en Unity? (SR04)
- Is het toegankelijk voor gebruikers? (SR05)

De eerste criteria is dat het benodigde data kan opslaan en deze vraag was gesteld nadat er gebleken was dat CAD-modellen niet gebruikt konden worden. Het probleem met CAD-modellen was dat het een 2D plattegrond was, waarin de benodigde informatie niet terug gevonden kon worden. Daarom zal er gekeken worden naar 3D modellen die dus de data wel moeten kunnen opslaan en dit zijn er gelukkig meerdere.

De tweede criteria is dat de modellen aan te passen/manipuleren moeten zijn met .NET. Uit de vorige onderzoek was naar voren gekomen dat Unity gebruikt zou worden en Unity heeft een eigen implementatie voor de modellen, waarbij geïmporteerde modellen worden geconverteerd naar GameObjects. Hierdoor zouden de objecten, die Unity kan uitlezen en gebruiken, ook te manipuleren zijn met .NET code.

De laatste nieuwe criteria is dat het toegankelijk is voor gebruikers en dit betekent of de modellen bekend zijn onder gebruikers. Hiermee wordt bedoeld dat de gebruikte modellen geaccepteerd zijn als “standaard” en dus ondersteund worden door de verschillende applicaties die te maken hebben met 3D modellen.

Als er gezocht zou worden naar 3D modellen zullen veel verschillende modellen naar voren komen, maar om een selectie te krijgen dat te onderzoeken is kunnen de laatste twee genoemde criteria meegenomen worden in de zoekopdracht. Hierdoor zullen alleen 3D modellen naar voren komen die breder geaccepteerd zijn en ondersteund worden door Unity. Als een model namelijk ondersteund is door Unity (Unity, 2017), kan het model benaderd worden met .NET code.

Uit deze zoektocht komen eigenlijk drie modellen naar voren, namelijk FBX, DAE en OBJ. AutoDesk is een van de eerste softwarebedrijven gericht op 3D ontwikkeling en is de eigenaar van het bekende model FBX. Dit model zal daarom meegenomen worden in het onderzoek. Het probleem met deze modellen is dat ze niet volledig opensource zijn en er dus een strikte controle over is. Hierom zal er ook een model dat lijk op FBX onderzocht worden en dat zijn DAE-modellen (Unity Users, 2015). Deze modellen zijn opensource (Khronos, n.d.) en dus toegankelijk voor iedereen.

Het laatste model dat onderzocht zal worden zijn OBJ-modellen. Deze modellen zijn ook opensource en zijn dus ook toegankelijk voor gebruikers. Een ander voordeel van OBJ-modellen is dat een groot deel van de bestaande 3D applicaties de model extensie al ondersteunen, waardoor het één van de breder geaccepteerde type is voor 3D modellen.

Om een duidelijker beeld te kunnen krijgen van deze modellen zal er een toegepast onderzoek uitgevoerd worden. In dit onderzoek zullen de verschillende modellen dezelfde processen ondergaan, waaruit duidelijk zal worden hoe de modellen gebruikt kunnen worden om het doel te kunnen behalen. Hierbij zullen de modellen moeten voldoen aan de requirements die opgenomen zijn in tabel 4.3.

	Requirement	Prioritering
1	Het model kan de juiste data bevatten (SR03)	Hoog
2	Het model is makkelijk in gebruik (SR06)	Laag
3	Het model is te manipuleren door middel van .NET C# (SR04)	Hoog
4	Het model is achteraf/dynamisch in te laden (SR02)	Hoog
5	Het model is gratis in gebruik (PR04)	Laag

Tabel 4.3: Model Requirements

Naast de nieuwe requirements die verkregen zijn tijdens de sprintovergang, zijn er ook twee andere requirements meegenomen die al eerder vastgesteld waren. De eerste is de mogelijkheid om de modellen achteraf/dynamisch in te laden. Dit betekent dat het model dat gekozen wordt ook achteraf, dus nadat de applicatie uitgegeven is en op de HoloLens van de gebruiker staat, ingeladen kan worden. Dit is belangrijk omdat niet alle modellen van alle gebouwen op de wereld in de applicatie inbegrepen kunnen zijn. Daarom moet de gebruiker de mogelijkheid krijgen de modellen zelf te selecteren (Requirement UR01).

De tweede requirement is dat het gratis in gebruik moet zijn. Dit heeft vooral te maken met het feit dat het om een Proof of Concept gaat, waarin het belangrijk is dat de werking aangetoond wordt en de applicatie niet direct klaar hoeft te zijn voor uitgave. Hierdoor zullen bij keuzes tussen betaalde en gratis wegen eerst de gratis wegen gebruikt worden om te zien of het mogelijk is het doel te behalen.

4.3.2 UITVOERING

Om een 3D model te kunnen aanbevelen zijn de modellen die uit het vooronderzoek naar voren zijn gekomen gebruikt in de praktijk. Hierbij zijn er verschillende technische problemen voortgekomen, die een groot invloed hadden op het uiteindelijke gekozen 3D model. Deze problemen zullen in dit hoofdstuk beschreven worden en indien het mogelijk was deze problemen op te lossen zullen de oplossingen ook beschreven worden.

OPZET

Om de modellen te kunnen testen moest er een 3D model gemaakt worden dat gebruikt kon worden als testobject. Dit model kan in elke 3D modelleer software gemaakt worden, maar door eerdere ervaringen met AutoDesk Maya is er gekozen voor dit softwarepakket. In AutoDesk Maya is er een gebouw gemaakt, waarbinnen een aantal verschillende kamers geplaatst zijn met allemaal verschillende afmetingen. Ook hebben de kamers een variërend aantal en soorten attributen toegekend gekregen. Deze opzet zorgt ervoor dat de verschillende aspecten die gespecificeerd zijn in de requirements getest kunnen worden voordat het geïmplementeerd wordt.

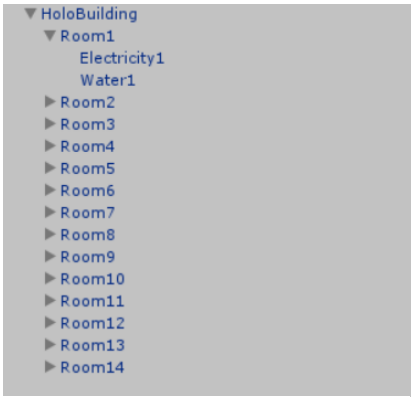
Om de gemaakte gebouw te testen is het vervolgens geëxporteerd naar de verschillende model types, namelijk FBX, DAE en OBJ. Deze modellen zijn geïmporteerd in Unity waarmee vervolgens verder getest is.

BESCHIKBARE DATA

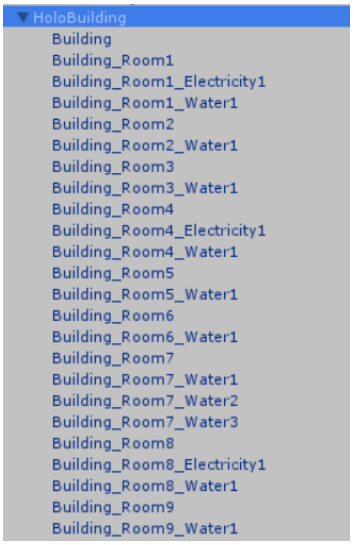
De eerste requirement is of het model de juiste data kan bevatten, maar tijdens het vooronderzoek was deze requirement eigenlijk al aan voldaan. Het probleem met de CAD-modellen was dat ze 2D waren en dus niets anders waren dan een plattegrond van een ruimte, waarin de afmetingen weergegeven werden. Aangezien er nu gebruik gemaakt wordt van 3D modellen is deze requirement al gedekt.

MODEL GEBRUIK

De verschillende modellen kunnen makkelijk in Unity gebruikt worden, aangezien Unity deze modellen ondersteund. Hierbij is er wel een verschil te merken tussen de modellen en dat is in de structuur. Hier was namelijk te merken dat Unity de OBJ-modellen anders interpreteert dan de andere twee. Bij de FBX en DAE-modellen is er namelijk een goed structuur terug te vinden (zie figuur 4.1), waarbij de root element het gebouw is. Deze root element heeft weer kinderen wat de kamers zijn en vervolgens konden ook de kamer-attributen onder de kamers als kinderen terug gevonden worden. Hierdoor was er een duidelijk gestructureerde object terug te vinden waarmee makkelijk gewerkt kon worden.



Figuur 4.1: FBX & DAE-structuur



Figuur 4.2: OBJ-structuur

Bij het OBJ-model daarentegen was de structuur niet handig om mee te werken, zie figuur 4.2. Als root element was er namelijk een leeg object gemaakt die alle andere elementen bevat. Alleen deze root element had kinderen en dat waren alle andere objecten die in het model te vinden waren. Hierdoor zaten het gebouw, kamers en attributen in één groot lijst. Er was echter een manier om de structuur terug te vinden en dat was door te kijken naar de namen. Deze hadden wel de structuur behouden, namelijk gescheiden door middel van laagstreepjes.

UITLEZEN MET .NET

Om GameObjecten met .NET C# te kunnen manipuleren is het nodig gebruik te maken van de Unity library. Deze wordt UnityEngine genoemd en is volledig gedocumenteerd op de website van Unity (Unity Scripting, 2017).

Het eerste waar deze library voor gebruikt is om de structuur van de OBJ-modellen gelijk te krijgen met de andere twee modellen. Hiervoor is er een methode geschreven die de verschillende elementen in het OBJ-model volgens naam op de juiste manier structureert. Hierdoor kon de structuur die in figuur 4.2 te zien is, omgezet worden naar de structuur in figuur 4.1. De methode die geschreven is, is te zien in Bijlage Code 01.

MODELGROOTTES

In de uiteindelijke applicatie moet de juiste kamer gekozen worden gebaseerd op de dimensies. Tijdens de creatie van het model was er een grootte van 15m x 15m x 15m gegeven aan het gebouw, waarin verschillende kamers zijn met verschillende dimensies. Deze dimensies waren echter niet terug te vinden in Unity en moesten op een speciale manier terug gevonden worden.

Bij de FBX en DAE-modellen waren de dimensies relatief aan de Parent, oftewel het gebouw. Hierdoor was een kamer met een breedte van 0.36 gelijk aan 5.6 meter. Deze waarde was te verkrijgen door de breedte van de kamer met de breedte van het gebouw te vermenigvuldigen. Het was dus mogelijk de dimensies van deze modellen terug te krijgen door middel van een simpele berekening.

Bij OBJ-modellen was dit helaas niet mogelijk, omdat het gebouw, de kamers en de attributen dimensies hadden van 1 x 1 x 1. Deze implementatie van dimensies is in veel gevallen handig, aangezien 1 de originele staat betekent, maar in de huidige implementatie was het verwarrend en onhandig. Om toch de afmetingen van de objecten te kunnen achterhalen is er gekeken naar de UnityEngine documentatie (Unity Scripting, 2017). Hierin is de Mesh gevonden en de Mesh is eigenlijk de buitenkant/huid van een model. Dit is dus het fysieke deel van een object wat gezien kan worden.

Deze Mesh heeft verschillende eigenschappen, maar de eigenschap waarnaar gezocht werd, was de bounds eigenschap. Dit zijn de afmetingen van het fysieke object in de wereld van Unity. Door deze bounds property op te vragen was het dus mogelijk de dimensies van een kamer te achterhalen.

Na wat verder onderzoek is gebleken dat deze bounds property wat betrouwbaarder is dan een calculatie tussen parent en child elementen en kon deze property dus ook gebruikt worden op de andere modellen. De implementatie van deze property is te zien in Bijlage Code 02.

KAMER SELECTEREN

Om andere problemen met afmetingen te voorkomen is er ook geprobeerd een kamer te selecteren doormiddel van een vooraf aangegeven dimensie. Hierbij zou de applicatie kijken naar alle kamers binnen het gebouw (die zelf in het model staat) en de overeenkomende kamers, binnen een toegestane tolerantie, aangeven. Hierbij kwam het probleem naar voren dat een kamer niet gevonden kon worden, terwijl de juiste kamer er wel bij zat.

Na wat uitzoek werk bleek dat het probleem te maken had met de rotatie van de kamer, waardoor de afmetingen wel klopten, maar de breedte en diepte omgewisseld waren. Dit was vrij simpel op te lossen, door de controle ook met omgewisselde breedte en diepte uit te voeren. Hierdoor werd het model direct gevonden en was het probleem verholpen.

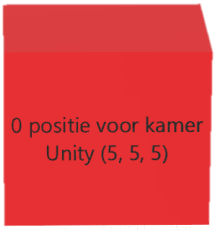
MODELPOSITIES

Nadat een kamer geselecteerd is moet de kamer geplaatst worden op de juiste positie. Deze positie zou in een later stadium berekend worden, maar om te testen werden de kamers op verschillende posities geplaatst. De eerste waarde die gebruikt werd is de 0 positie (x: 0, y: 0, z: 0). Dit is het middelpunt van de Unity wereld en het is ook dus de makkelijkste plaats om objecten te testen.

Bij het plaatsen van objecten op de 0 positie kwam er snel nog een probleem naar voren en het had weer betrekking tot de OBJ-modellen. De FBX en DAE-modellen werden namelijk correct geplaatst op de aangegeven positie, waarbij het middelpunt van het object op de aangegeven positie stond.

Het probleem met de OBJ-modellen is eigenlijk gerelateerd aan het probleem dat bij de modelgroottes optrad. Namelijk dat Unity de losse objecten allemaal een originele waarde toedeelt. Hiermee wordt bedoeld dat het object op zijn originele positie wordt gezien als zijn 0 positie. Dus als een object op positie (5, 5, 5) staat dan ziet Unity dat als de 0 positie van dat object. Voor een voorbeeld zie figuur 4.3, waarbij het kubus eigenlijk staat op positie (5, 5, 5) en volgens Unity het object staat op de 0 positie. Maar vergeleken met de echte 0 positie van de Unity wereld is het duidelijk te zien dat de kubus niet staat op de 0 positie.

Dit is net als bij de modelgroottes handig in sommige situaties, maar niet in dit geval. Aangezien de modellen niet specifiek gepositioneerd kunnen worden. Om dit probleem op te lossen is het model voor het exporteren aangepast, zodat alle kamers in het model op de 0 positie staan. Dit is bij het zien van de hele gebouw geen mooi gezicht, maar voor de functionaliteit van de applicatie maakt de positie binnen het gebouw niets uit. Dit is echter een probleem dat op een later stadium tijdens de ontwikkeling nogmaals onderzocht kan worden, aangezien dit geen goede oplossing is voor het probleem.



VERBETERING MODELPOSITIES

Nadat een groot deel van de applicatie gebouwd was, was er tijd voor wat meer onderzoek en verbeteringen in onderdelen die niet optimaal uitgevoerd waren. De positionering was een van deze onderdelen die een betere oplossing nodig had.

Figuur 4.3: OBJ-model positionering

Zoals al aangegeven was, kunnen OBJ-modellen niet makkelijk geplaatst worden op de gewenste positie. Dit komt doordat de werkelijke positie van het model anders is dan Unity denkt. Het probleem tijdens het onderzoek was dat er niet genoeg ervaring was met Unity, waardoor er geen goede oplossing bedacht kon worden. Tijdens de ontwikkeling van de applicatie is meer ervaring opgedaan en is er een betere oplossing bedacht.

Het was namelijk eerst niet bekend hoe de werkelijke positie van modellen achterhaald konden worden, maar tijdens de ontwikkeling is hier een manier voor gevonden. Door middel van de UnityEngine library kunnen de hoeken van een object opgevraagd worden. Deze hoeken kunnen opgeteld worden en vervolgens gedeeld worden door de aantal hoeken die het model heeft. Dit levert de werkelijke positie van het model op.

Vervolgens kan er door middel van een simpele berekening een relatieve locatie berekend worden, namelijk met de volgende formule:

RELATIEVE POSITIE = GEWENSTE POSITIE – WERKELIJKE POSITIE

Als voorbeeld kan de situatie in figuur 4.3 gebruikt worden, waarbij de gewenste positie de 0 positie (0, 0, 0) is en door de berekeningen de werkelijke positie achterhaald kan worden, wat in dit geval (5, 5, 5) is. Dit levert een relatieve positie op van (0, 0, 0) – (5, 5, 5) = (-5, -5, -5). Door deze positie in Unity te gebruiken als de positie van de kubus, zou de kubus staan op de 0 positie, waardoor het dus staat op de gewenste positie.

Door deze implementatie komt echter nog een ander probleem naar boven en dat is bij het roteren van het object, het object zal roteren om de relatieve positie, dus (-5, -5, -5). Hierdoor zal het object na de rotatie niet meer staan op de gewenste positie (0, 0, 0). Om dit te voorkomen kan er gebruik gemaakt worden van een methode in de UnityEngine die ervoor zorgt dat het object roteert om de gewenste positie. Deze positie zou in het voorbeeld dus (0, 0, 0) zijn, wat na het positioneren van het object volgens de formule, ook het middelpunt zou zijn van het object. Hierdoor zal het object roteren om zijn eigen as, wat ook de bedoeling is.

DYNAMISCH INLADEN

Om de gebruiker in staat te stellen eigen modellen in te laden (Requirement UR01), ontbrak er nog maar één onderdeel en dat is het achteraf/dynamisch inladen van de modellen. Dynamisch inladen is het inladen van een model tijdens het gebruik van de applicatie. Dit komt erop neer dat het model die de gebruiker wilt gebruiken niet tijdens de publicatie mee gepubliceerd wordt. Dit is ook wel logisch, aangezien het onmogelijk is de modellen van alle gebouwen op de wereld met de applicatie mee te publiceren. Door modellen dynamisch in te laden kan de gebruiker tijdens het gebruik van de applicatie een model selecteren, wat vervolgens gebruikt kan worden als het model. Dit was helaas niet direct mogelijk met Unity, aangezien dynamisch inladen niet wordt ondersteund.

Een manier om modellen dynamisch in te laden in Unity is door een Unity Pro account te verkrijgen en vervolgens gebruik te maken van Assetbundles. Dit zijn speciale modellen voor Unity die makkelijk uitgelezen kunnen worden. Om een FBX, DAE of OBJ-model om te zetten in een AssetBundle moet het model in Unity zelf omgezet worden. Dit neemt de toegankelijkheid criterium van het onderzoek weg en is er dus gezocht naar alternatieve manieren.

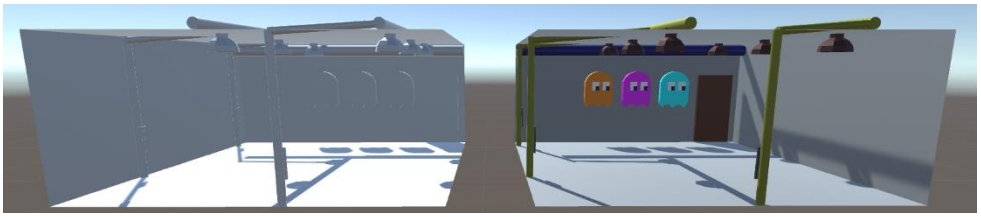
Een alternatieve methode was het gebruik van een library die het mogelijk maakt de modellen dynamisch in te laden. Er zijn geen libraries in overvloed die dit mogelijk maken en doordat er geen budget is moest er hard gezocht worden naar libraries die pasten bij de requirements.

De eerste en misschien wel beste library die gevonden was, was Assimp (Assimp, 2016). Deze library is geschreven in C++, maar het is ook mogelijk een versie met een .NET wrapper te verkrijgen. Assimp is een library die vele verschillende modellen kan importeren en het zou een hele goede library geweest zijn, maar na verschillende implementatie pogingen is het niet gelukt Assimp werkend te krijgen.

Hierna is gekeken naar andere libraries met als focus een library die het mogelijk maakt FBX of DAE-modellen in te laden, aangezien deze eenvoudiger in gebruik waren dan de OBJ-modellen. Na een lange zoektocht naar mogelijke libraries is er geen gevonden en is er gekeken naar importeurs voor de OBJ-modellen.

In tegenstelling tot de andere twee waren er een aantal importeurs voor OBJ-modellen. De OBJ-modellen zijn blijkbaar makkelijker uit te lezen, waardoor er ook een aantal verschillende importeurs waren en ook nog eens gratis in gebruik waren. De reden dat OBJ makkelijker uit te lezen zijn, is omdat de OBJ-modellen alleen informatie over de model zelf bevatten. Alle overige informatie, zoals texturen en animaties, worden niet in OBJ-modellen opgeslagen. Hierdoor is de data die in OBJ-modellen te vinden is ook echt alleen model gerelateerd.

Hierbij kwam bij de OBJ-importeur wel een extra requirement, namelijk dat ze ook texturen moeten kunnen uitlezen. Deze requirement was bij de andere modellen niet nodig, omdat deze modellen de data al bevatten. Bij OBJ-modellen worden de texturen niet in het model zelf opgeslagen, maar in aparte MTL-bestanden. Deze bestanden bevatten alleen de texturen die gebruikt zijn in het model en zonder deze bestand mee te importeren zullen de modellen alleen bestaan uit grijswaarden. Dit is ook te zien in figuur 4.4, waarin een kamer te zien is die in AutoDesk Maya gemaakt is en vervolgens is geïmporteerd in Unity.



Figuur 4.4: OBJ-model zonder (links) en met (rechts) MTL/textuur bestand

De eerste vondst was een ObjImporter (Anonimo, 2012) dat te vinden was op een wiki dat was opgezet voor Unity. Deze importeur was echter oud en kon alleen de model importeren, terwijl het ook mogelijk moet zijn de texturen van het model te importeren.

Hierom is gekeken naar andere importeurs en is er een importeur gevonden met de naam “Runtime OBJ Importer”, geschreven door aaro4130. Deze importeur was te vinden in de Asset Store van Unity, waardoor het makkelijk te gebruiken is in het project. Een voordeel van deze importeur ten opzichte van de ObjImporter is dat de “Runtime OBJ Importer” ook de MTL-bestanden kan importeren. Dit doet hij door op de locatie van de OBJ-bestand te kijken of er een MTL-bestand aanwezig is met dezelfde naam. Als dit het geval is, wordt deze textuur bestand ook geïmporteerd, waardoor de geïmporteerde model niet meer bestaat uit alleen grijs waarden, zie figuur 4.4.

4.3.3 CONCLUSIE EN AANBEVELING

De verschillende modellen hebben allemaal hun eigen plus en minpunten (tabel 4.4), maar er is niet echt een model die zonder twijfel gekozen kan worden. Hierdoor moest er een keuze gemaakt worden om mee door te gaan in het vervolg van het project.

	Requirements	FBX	DAE	OBJ
1	Het model kan de juiste data bevatten (SR03)	Ja	Ja	Ja
2	Het model is makkelijk in gebruik (SR06)	Ja	Ja	Gedeeltelijk
2.1	- Goed Unity structuur	Ja	Ja	Gedeeltelijk
2.2	- Dimensies Correct	Ja	Ja	Ja
2.3	- Locatie Correct	Ja	Ja	Gedeeltelijk
3	Het model is te manipuleren met .NET C# (SR04)	Ja	Ja	Ja
4	Het model is achteraf/dynamisch in te laden (SR02)	Nee	Nee	Ja
5	Het model is gratis in gebruik (PR04)	Nee	Nee	Ja

Tabel 4.4: 3D model resultaten

Tijdens het onderzoek is gekeken of de verschillende modellen zouden voldoen aan de requirements die opgesteld waren. De FBX en DAE-modellen waren passender bij de tweede requirement dan de OBJ-modellen, maar dit was een requirement die een lage prioritering had, aangezien deze requirement niet uitmaakte voor het eindresultaat. Deze requirement was vooral bedoeld om de ontwikkeling zelf eenvoudiger te houden.

Het OBJ-model was niet zo makkelijk in gebruik als de andere twee maar voldeed wel aan de andere requirements die een hogere prioritering hadden gekregen. Deze requirements waren in tegenstelling tot de tweede requirement wel belangrijk voor het eindresultaat en was het dus belangrijker dat de modellen aan deze belangrijkere requirements voldeden. De FBX en DAE-modellen waren totaal niet passend bij deze requirements en was het eigenlijk de één of het ander situatie.

Gezien de tijdsduur van het project is er gekozen voor OBJ-modellen, aangezien dit de enigste model is die alle requirements van het project kan ondersteunen. Mocht het later mogelijk zijn een van de andere modellen dynamisch in te laden, zal er vooral ook aangeraden worden die modellen te gebruiken, aangezien deze veel minder inspanning nodig hebben om werkend te krijgen. Ook wordt er aangeraden om de opzet van de modellen op een generieke manier te doen, zodat het later mogelijk is ondersteuning voor de andere modellen makkelijk in te kunnen bouwen.

SPRINT OVERGANG

Tegen de tijd dat het duidelijk was welke 3D model gebruikt zou worden was de tweede sprint afgelopen. Dit was dus een goed moment om aan de opdrachtgever voor te stellen wat voor bevindingen waren gemaakt en met welk 3D model verder gewerkt zou worden.

SPRINT 2

Om de opdrachtgever aan te geven welke keuzes gemaakt zijn, is er een demo gegeven met de resultaten van het onderzoek dat in sprint 2 uitgevoerd was. Hierbij is uitgelegd dat twee van de drie modellen voor minder problemen zouden kunnen zorgen tijdens de ontwikkeling, maar dat deze modellen ook tegenstrijdig waren met verschillende requirements die opgesteld waren. Daarom is er gekozen voor OBJ-modellen, omdat de problemen die deze modellen opleverden al opgelost waren en er geen grote problemen meer verwacht werden met deze modellen. Dit werd positief ontvangen en was het goed om door te gaan met de opdracht.

SPRINT 3

Tijdens de verschillende onderzoeken die uitgevoerd waren, was er al een applicatie dat ongeveer deed wat de uiteindelijke applicatie moest doen maar zonder structuur, daarom zou er in sprint 3 begonnen worden aan de architectuur van de uiteindelijke applicatie. Ook zou er een begin gemaakt worden aan het laatste onderzoek, namelijk het koppelen van de modellen. Deze onderzoek zou uit twee delen bestaan, waarbij het tweede deel uitgevoerd zou worden nadat de applicatie gebouwd was, aangezien dit de optimalisatie was van het project.

4.4 KOPPELEN VAN 3D MODELLEN

Om een model te kunnen koppelen aan de echte wereld, zullen een aantal stappen uitgevoerd moeten worden. Hoe deze stappen het beste en meest accuraat uitgevoerd kunnen worden zullen in dit hoofdstuk onderzocht en beschreven worden.

4.4.1 VOORONDERZOEK

Om te kunnen onderzoeken hoe de modellen gekoppeld kunnen worden aan de echte wereld, moet er eerst vastgelegd worden welke data beschikbaar is en wat berekend moet worden.

Door het gebruik van de UnityEngine is het mogelijk de afmetingen van objecten binnen Unity op te meten. Deze afmetingen kunnen dus van zowel de gescande kamer, zie hoofdstuk 5.5, als het model berekend worden.

Om het koppelen verder te kunnen afhandelen moet er ook onderzocht worden hoe een kamer uit een model geselecteerd en ingeladen kan worden. Dit zodat het overeenkomt in de echte wereld. Hierbij moeten wel bepaalde waardes als positie en rotatie berekend worden.

HOLOLENS WERKING

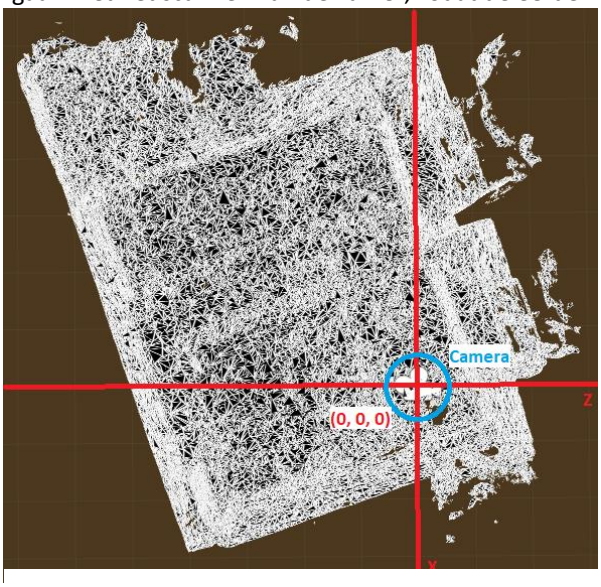
Om een indicatie te kunnen geven hoe de HoloLens te werk gaat is er een test applicatie gemaakt, waarin de HoloLens een scan maakt van de kamer. Deze scan is vervolgens geëxporteerd naar Unity volgens de aanwijzingen van Microsoft (MS Scanning, n.d.). Het vervolg van dit paragraaf is een observatie van de test applicatie.

Op het moment dat de applicatie wordt opgestart in de HoloLens wordt de positie en rotatie van de HoloLens opgeslagen met de waarde 0. Als de gebruiker vanaf dit moment veranderd van positie of een andere richting op kijkt, dan worden de positie en rotatie veranderd van de camera. In Unity wordt de gebruiker/HoloLens namelijk gerepresenteerd als een camera.

Nadat de applicatie is opgestart kan de HoloLens de kamer scannen. Als de kamer nog niet eerder gescand is wordt er een nieuwe scan gemaakt en als de kamer wel al eerder gescand was, wordt de opgeslagen scan van de kamer ingeladen en vervolgens kan de HoloLens door gaan met het scannen van de kamer, zodat de eerder gemaakte scan verbeterd kan worden.

Deze scan wordt om de camera heen geladen, zie figuur 4.5. Zoals te zien is in het figuur is de camera het gecentreerde object, waardoor de scan die gemaakt wordt een rotatie kan bevatten. Deze rotatie zal ook berekend moeten worden, om de kamer die ingeladen wordt met de juiste rotatie te kunnen positioneren.

Een ander punt is dat het middelpunt van de gescande ruimte niet gelijk is aan de camera. De camera staat namelijk op (0, 0, 0), wat het middelpunt van de scene in Unity is. Hierdoor kan het ingeladen object niet geplaatst worden op (0, 0, 0) en moet dus het middelpunt van de gescande ruimte berekend worden.



Figuur 4.5: Scan positionering

4.4.2 UITVOERING

Bij het vooronderzoek is door middel van een simpele test applicatie bepaald wat voor data beschikbaar zal zijn vanuit Unity en wat voor data berekend zal moeten worden. In dit hoofdstuk zal er uitgelegd worden hoe de benodigde data verkregen kan worden. Hierbij gaat het om een eerste versie, waarbij in een later paragraaf (zie paragraaf 4.4.3) de methoden die gebruikt zijn geoptimaliseerd zullen worden.

VERKIJRGEN VAN KAMER DIMENSIES

De eerste stap bij het koppelen is het vinden van de juiste kamer in het 3D model. Hiervoor zijn echter de dimensies van de kamer nodig, waar de gebruiker zich op dat moment in bevindt. Tijdens het onderzoek hiernaar zijn niet veel toepassingen gevonden, waarbij het lukt de afmetingen van een gescand object te bepalen. In een van de gevallen had het meer betrekking tot object herkenning (Extraction, 2016) in een ruimte en het bepalen van de dimensies van het herkende object. Hierbij kon de kamer het herkende object zijn, maar de source code van de technieken die gebruikt werd door het bedrijf kon niet verkregen worden. Hierom is er verder gezocht.

Een tweede oplossing en eigenlijk wel de meest accurate oplossing is een techniek die was toegepast in een applicatie dat al uitgebracht was voor de Microsoft HoloLens, namelijk HoloMeasure (Hajjaj, n.d.). In deze applicatie heeft de gebruiker de mogelijkheid punten in de wereld te plaatsen. Deze punten worden vervolgens gebruikt om de afstanden te berekenen. Dit is natuurlijk een oplossing die handig is bij een accuratere dimensie bepaling, maar tijdens dit project is het gewenst de dimensies automatisch te bepalen. In het geval dat dit niet mogelijk is zou er natuurlijk gebruik gemaakt kunnen worden van punten in de wereld.

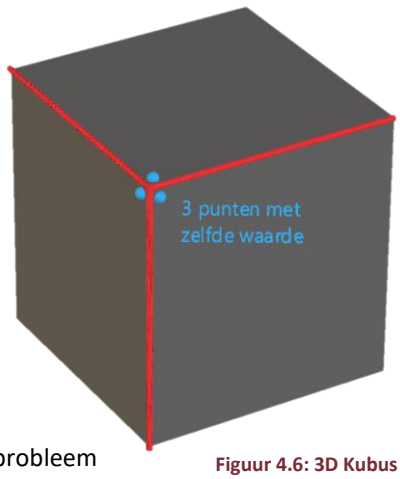
Om toch de dimensies automatisch te kunnen berekenen moest er dus een eigen implementatie gemaakt worden. Hierom is gekeken of het mogelijk is om de dimensies te berekenen van object binnen Unity. Hierbij zou de Scale waarde gebruikt kunnen worden, dat gebonden is aan een object, maar dit is geen representatie van de grootte van het object. Dit is namelijk een waarde dat gebonden is aan de grootte van het object voor makkelijke aanpassing. Dus als een object dimensies heeft van (5, 5, 5) dan is de scale (1, 1, 1), wat originele grootte is. Als de scale vermenigvuldigd zou worden met de originele afmetingen van het object (dus 1 x 5) dan zou de huidige grootte van het object berekend kunnen worden. Dit komt erop neer dat als de scale naar 2 veranderd zou worden dat het object in de scene niet een afmeting van 5 maar van 10 zou hebben.

Hierom moest er een andere manier gevonden worden om de afmetingen te verkrijgen en is er geprobeerd de berekening te gebruiken die ook in paragraaf 4.3.2 gebruikt was. Omdat de scan van de kamer een object oplevert in Unity was er dus ook een Mesh aanwezig. Hierdoor konden de buitenste hoeken van het object verkregen worden. Hierbij was er alleen een probleem dat de gescande kamer niet alleen bestaat uit de muren die echt aanwezig zijn, maar ook uit gedeeltes buiten de kamer die volgens de HoloLens aanwezig zijn. Deze delen komen vaak voor bij het scannen in de buurt van ramen. Dit zou er natuurlijk voor zorgen dat de berekening op de gescande Mesh niet accuraat zouden zijn. Dit was ook wel te merken uit de verkregen afmetingen, aangezien er werd getest in een kamer van vijf bij zes meter en de resultaten een kamer van acht bij zeven meter opleverden.

Aangezien de afmetingen niet direct uit de scan verkregen kunnen worden, moest er een ander manier gevonden worden om de afmetingen te bepalen. Een idee hierbij was om de gescande ruimte te verwerken in rechte delen, waardoor de kleine onnauwkeurige delen eruit gefilterd zouden worden. Dit was gelukkig ook een techniek die geleverd werd door Microsoft. De techniek wordt Mesh Processing genoemd en er zal meer over Mesh Processing verteld worden in Hoofdstuk 5.6. Kort gezegd zorgt Mesh Processing ervoor dat de gescande ruimte wordt omgezet in rechte panelen, waarbij de panelen onder verschillende categorieën kunnen vallen, zoals muren, tafels, vloer of plafond.

Door middel van deze techniek was het dus mogelijk de kleine onnauwkeurigheden eruit te halen, en gebruik te maken van rechte panelen die geplaatst zijn op de positie van de gescande Mesh. Wat hierbij opvalt is dat er veel muren en plafonds aanwezig zijn, terwijl er maar één vloer wordt aangemaakt. Omdat de vloer wordt geplaatst onder de gescande ruimte was het dus een goede representatie van de gescande ruimte en door de hoeken van de vloer op te vragen kon dus de afmeting berekend worden.

Aangezien de panelen eigenlijk uitgerekte 3D vierkanten zijn, werden er acht hoeken/Vertices verwacht. Omdat er gebruik gemaakt wordt van een 3D kubus, hebben de hoeken coördinaten van drie waardes, namelijk de breedte (X), hoogte (Y) en diepte (Z). Van de acht hoeken die verwacht werden zouden er vier een unieke samenstelling van de breedte en diepte hebben. De hoogte maakt in dit geval niet uit, aangezien de hoogte niet berekend zal worden uit de dimensies van de vloer.



Figuur 4.6: 3D Kubus in Unity

Nadat de bedachte methode getest werd, waren er niet acht maar vierentwintig Vertices. In deze lijst met Vertices waren er maar acht met een unieke samenstellingen van een breedte, hoogte en diepte. De andere zestien waren duplicaten van de unieke waardes. Dit zorgde voor verwarring, aangezien er maar acht vertices verwacht werden. Na het uitzoeken van het probleem bleek het zo te zijn dat vierentwintig Vertices klopt in Unity, bij een het gebruik van een 3D kubus (Eric5h5, 2011). Een kubus heeft namelijk zes zijdes, waarvan elke zijde vier verschillende hoeken heeft. En elke hoek van de kubus is eigenlijk het trefpunt van drie verschillende zijdes, zie figuur 4.6. Dit is geen grote hindering, aangezien alleen de unieke waardes gebruikt kunnen worden.

Nadat er vier Vertices gevonden waren met een unieke samenstelling van breedte en diepte kon door middel van afstand berekening berekend worden hoe lang de zijdes waren. Hier was ook een methode voor onder de Vector3 class, namelijk de “Distance” methode. Hierdoor hoeft de afstand niet zelf berekend te worden door middel van de stelling van Pythagoras (aangezien het een afstand is in een 2D ruimte, namelijk een breedte en diepte). Hierbij is het volgende probleem dat je niet weet of je twee aanliggende zijdes hebt berekend, namelijk de breedte en diepte.

Om er zeker van te zijn dat de twee aanliggende zijdes gebruikt worden, kunnen alle punten met elkaar vermenigvuldigd worden. Hieruit zullen twaalf afmetingen naar voren komen (4 hoeken x 3 andere hoeken). Van deze afmetingen zullen er maar zes uniek zijn, namelijk de vier zijdes van de vier hoeken en de twee diagonalen. Aangezien de twee diagonalen niet meegenomen moeten worden en alleen twee aanliggende zijdes nodig zijn om aan te geven wat de breedte en diepte is kan de lijst met afmetingen gesorteerd worden op hoogte. De twee hoogste afmetingen zullen de diagonalen zijn en de laagste twee afmetingen zullen tegengestelde zijdes zijn, bijvoorbeeld de linker en rechterzijde. Van deze twee zijdes kan één zijde meegenomen worden als de eerste zijde. De tweede zijde zal één van de overige zijdes zijn.

Op deze manier is de breedte en diepte van de kamer achterhaald, maar ook de hoogte is nodig. Deze hoogte kan ook berekend worden door middel van de hoogste en laagste punt van het verwerkte scan te, maar dit is niet nodig aangezien er tijdens het Spatial Processing wordt opgeslagen wat de hoogste en laagste punt, of terwijl vloer- en plafondniveau, van de gescande ruimte zijn. Door het verschil tussen deze twee punten te berekenen kan de hoogte van de kamer achterhaald worden.

BEREKENING VAN HET MIDDELPUNT

Bij het veranderen van de positie van een object binnen Unity wordt het middelpunt van het verplaatste object gelijkgesteld aan de gewenste positie. Hierdoor zou de aangegeven positie dus na het verplaatsen gelijk zijn aan het middelpunt van het verplaatste object. Aangezien een model precies over de gescande ruimte heen geplaatst moet worden moet het middelpunt van de gescande ruimte berekend worden.

Dit kan ook op meerdere manieren en de eerst gebruikte manier was door het gebruik van een methode van de UnityEngine, namelijk de bounds property (Unity Bounds, 2017). Deze methode haalt het middelpunt van het GameObject (wat het gescande ruimte is) op. Het probleem hierbij is een terugkomend probleem, namelijk dat de gescande ruimte veel delen heeft die buiten de kamer vallen. Hierdoor was de verkregen middelpunt van het gescande ruimte niet echt het middelpunt van de werkelijke ruimte.

De tweede manier was door het gebruik van een aantal waardes die in de vorige paragraaf berekend waren. Door de berekende hoeken/vertices bij elkaar op te tellen en te delen door vier (de aantal hoeken) kon het middelpunt van de breedte en diepte berekend worden. Om vervolgens ook de hoogte op de helft te krijgen, kon de berekende hoogte gedeeld worden door twee. Op deze manier kon er een nieuwe coördinaat gecreëerd worden dat gelijk is aan het middelpunt van de kamer.

BEREKENING VAN DE ROTATIE

Nadat de dimensies en het middelpunt berekend waren, moest de rotatie berekend worden. Zonder rotatie zou de ingeladen model namelijk niet overeenkomen met de echte ruimte. Om de rotatie van een kamer te kunnen berekenen zijn er eigenlijk maar twee aanliggende hoeken nodig. Door de hoekberekening in het voorafgaande paragraaf op een gestructureerd manier uit te voeren, kan er makkelijk achterhaald worden welke twee punten aanliggend zijn.

Nadat de twee aanliggende punten bekend waren, kon de rotatie berekend worden. Hiervoor is gebruik gemaakt van goniometrie, met name van de “arc tangent” functie. Om de rotatie te berekenen tussen de twee aanliggende punten is de volgende formule gebruikt:

$$Radian = Tan^{-1}(\Delta Y / \Delta X)$$

Om deze functie in .NET C# te kunnen gebruiken is er gebruik gemaakt van de Mathf library van Unity. Dit is een onderdeel van de UnityEngine library wat een extensie is op het .NET framework. De vertaling van de bovengenoemde formule is vertaald naar C# op de volgende manier:

$$var\ Radian = Mathf.Atan2(punt2.X – punt1.x, punt2.y – punt1.y)$$

Door middel van deze formule kan de rotatie (in radialen) berekend worden. In Unity worden radialen echter niet gebruikt als de rotatie waarde, maar graden. Om radialen om te zetten in graden is er ook een wiskundige formule, namelijk:

$$Degree = Radian * (180 / \pi)$$

Deze formule kan in C# geïmplementeerd worden op de volgende manier:

$$var\ Degree = Radian * (180 / Mathf.PI)$$

Door middel van deze berekening kan de rotatie berekend worden dat toegepast moet worden op de geselecteerde kamer, zodat het overeenkomt met de ruimte waar de gebruiker zich bevindt. Voor een voorbeeld zie Bijlage Code 04.

SELECTEREN VAN KAMER(S)

Nadat de benodigde data berekend is kan er gezocht worden naar de kamer in het model van het gebouw. Het zoeken van kamers in het model kan zeer eenvoudig uitgevoerd worden. Door het gebruik van dezelfde formule als in paragraaf 4.3.2 kan de afmeting van elke kamer achterhaald worden. Vervolgens kan deze afmeting vergeleken worden met de afmetingen van de gescande Mesh. Hierbij is het wel handig gebruik te maken van een tolerantie, aangezien de HoloLens niet specifiek is tot in de millimeters. Tijdens het onderzoek is het berekenen van de Mesh afmetingen een aantal keer uitgevoerd en is achterhaald dat de HoloLens tot ongeveer tien centimeter nauwkeurig is. Hierbij is er elke keer een andere scan van de omgeving gemaakt en waren de gemeten afmetingen zeer consistent.

Om zeker te zijn dat de juiste kamers gevonden zullen worden in het model zal er gebruik gemaakt kunnen worden van een tolerantie van tien centimeter en kan er als het nodig is een hoger tolerantie gebruikt worden (Bijlage Code 05).

INLADEN VAN EEN KAMER

De laatste stap is het inladen van de gevonden kamer. Hierbij zal er gebruik gemaakt worden van de data dat berekend is in de voorafgaande paragrafen.

Om de kamer op de juiste locatie te krijgen zal het middelpunt dat berekend is toegewezen worden aan de positie van de gevonden kamer. Hierdoor zullen de middelpunten van de geselecteerde en echte kamer met elkaar gelijk staan.

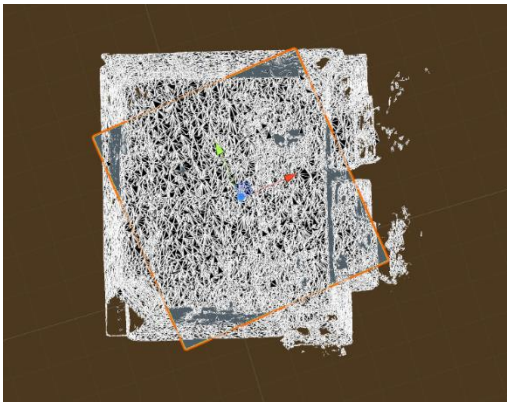
Vervolgens zal de kamer geroteerd moeten worden, zodat de muren van de geselecteerde en echte kamer gelijk zullen staan met elkaar. Hiervoor kan de berekende rotatie gebruikt worden en hierbij moet opgemerkt worden dat de berekend rotatie niet betekent dat de kamer goed ingeladen wordt. De kamer kan nog teveel gedraaid zijn met stappen van negentig graden en de reden hiervoor is dat berekening in de voorafgaande paragraaf alleen berekend hoever de gescande kamer gedraaid is. Hierbij wordt er geen rekening gehouden met waar de deuren of ramen van een kamer zich bevinden. Hierdoor zorgt de rotatie er wel voor dat de muren van de kamers met elkaar uitlijnen, maar kan het zijn dat de niet de juiste muren met elkaar uitgelijnd zijn.

Tijdens het zoeken naar een oplossing hiervoor is geen gevonden. Met name doordat de HoloLens alleen simpele attributen zoals tafels kan herkennen. De deuren worden als onderdeel van de muren gezien en kan er dus niet afgeleid worden waar de deur in een Mesh zou kunnen zitten. De ramen kunnen ook niet gescand worden, aangezien de sensoren geen reflectie krijgen van ramen worden de ramen als gaten in de muur gezien. Er is geprobeerd gebruik te maken van deze gaten als referentie punten, maar deze gaten konden ook gezien worden als deuren die open staan. Hierdoor is het momenteel niet mogelijk dit soort elementen te herkennen. Dit probleem zal met de opdrachtgever aan het einde van sprint 3 besproken worden.

Om de geselecteerde kamer toch op de juiste manier te kunnen plaatsten zal de gebruiker, na het inladen van de kamer, de mogelijkheid krijgen de kamer naar links of naar rechts te roteren. Hierbij zullen stappen van 90 graden gebruikt worden, zodat de berekende calculatie niet aangepast wordt.

4.4.3 OPTIMALISATIE

Nadat de applicatie was ontwikkeld is gebleken dat een aantal technieken die gebruikt zijn niet volledig werkten zoals verwacht. Met name het berekenen van de hoeken, afmetingen en rotatie. De reden hiervoor was dat voor deze methode gebruik gemaakt was van de vloer dat verkregen was door het verwerken van de Mesh. Zoals te zien is in figuur 4.7, wordt de vloer niet altijd op de juiste manier aangemaakt.

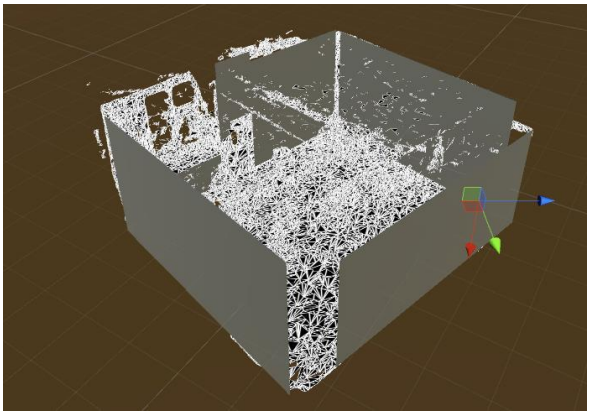


Figuur 4.7: Vloer plaatsing

Tijdens het onderzoeken was de vloer toevallig meerdere malen op de juiste manier geplaatst, waardoor het probleem niet tijdens het onderzoeken ontdekt was. De plaatsing van de vloer had meerdere problemen, namelijk dat de rotatie van de vloer niet overeenkomt met de echte ruimte en ten tweede waren de afmetingen niet correct. Zoals bij het verkrijgen van de afmetingen van de gescande Mesh waren de verkregen afmetingen hoger dan de afmetingen van de echte kamer.

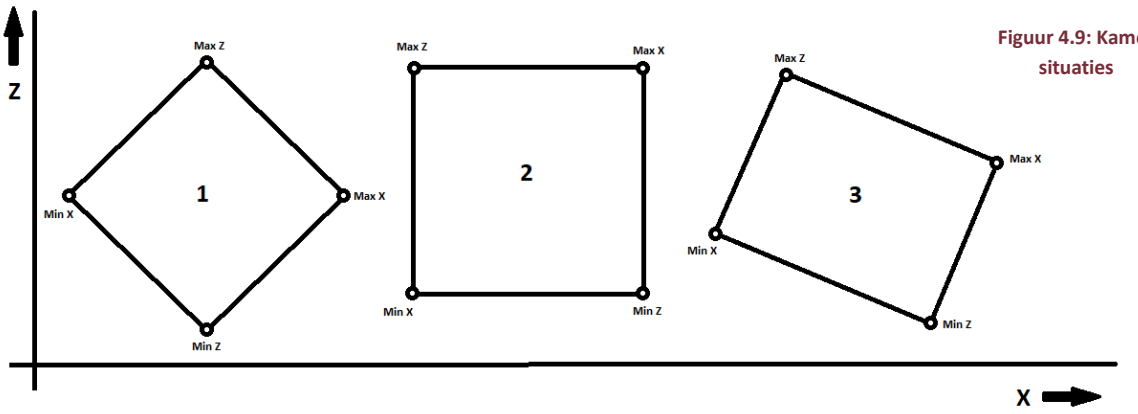
Dit bleek te komen door de manier waarop de vloer wordt geplaats door de Mesh Processing techniek. Hierbij wordt de vloer namelijk niet relatief aan de geplaatste muren, maar relatief aan de gescande Mesh geplaatst. Dit is ook te zien in figuur 4.7, waar een hoek van de scan begint op een gescande deel dat de HoloLens heeft herkend tijdens het scannen bij de raam van de kamer. Deze problemen hebben geleid tot het verkeerd inladen van de kamer.

Hierbij werd wel het volgende opgemerkt, namelijk dat de muren die geplaatst worden wel overeenkomen met de gescande Mesh, zie figuur 4.8. Om niet weer in hetzelfde valkuil te vallen zijn er meerdere verschillende scans van de kamer gemaakt, waarbij er gekeken is of de muren wel altijd overeenkomen en dit was wel het geval. Daarom is er een nieuwe berekening gemaakt om de hoeken van de kamers te berekenen. Deze wijziging had alleen betrekking tot het berekenen van de hoeken van de kamer. De andere berekeningen die uitgevoerd werden, hoefden niet aangepast te worden. Alleen een input waarde, namelijk de berekende hoek coördinaten, moeten aangepast worden.



Figuur 4.8: Muur plaatsing

Bij het berekenen van de hoeken is het niet belangrijk op welke hoogte de hoek geplaatst is. Hierdoor is er alleen gekeken naar de breedte (X) en diepte (Z). Door alle hoeken van alle geplaatste muren te vergelijken met elkaar zijn de hoeken eruit gehaald, door het vinden van de maximale X, minimale X, maximale Z en minimale Z-waarde. Hierbij is uit gegaan van situaties die afgebeeld staan in figuur 4.9.



Figuur 4.9: Kamer situaties

Zoals te zien is in het figuur zijn er een aantal situaties waarbij de redenering klopt, de eerste en derde situatie. Maar er kunnen gevallen zijn zoals situatie twee waarbij een waarde op twee verschillende hoeken voor kan komen. Een voorbeeld hiervan is dat bijvoorbeeld de hoek van Min X ook Min Z kan zijn en dat Max Z, Min X kan zijn. Dit zal echter geen probleem zijn, aangezien er gebruik gemaakt wordt van de variabele “Float”. Deze floats hebben een reikwijdte van -3.4×10^{38} tot $+3.4 \times 10^{38}$ (Microsoft Float, 2017). Zoals in paragraaf 3.4.2 beschreven was is de HoloLens nauwkeurig tot tien centimeter, waardoor de float waardes voor de hoeken in situatie 2 altijd wel zullen verschillen.

De nieuwe hoek herkenning werkt in de meeste gevallen goed, maar er waren een aantal scenario’s waar het niet handig was. Deze situaties hadden meer te maken met de kwaliteit van de scan dan het te maken had met de berekening. Een probleem was namelijk dat tijdens het scannen de deuren en ramen ook echt dicht moeten zijn. Dit komt omdat bij het scannen van de muur om een deur of raam heen de HoloLens onderdelen scant van wat achter de deuren of ramen aanwezig kunnen zijn. Hierdoor was er dus een kans dat een extra muur herkend werd, bijvoorbeeld van de kamer ernaast.

Dit zorgt er niet voor dat de hoek herkenning niet meer werkt, maar dat de herkende hoek niet de gewenste hoek is. Dit is niet alleen een probleem dat voorkomt als de gebruiker de deur open heeft laten staan, want de HoloLens maakt gebruik van “Caching” bij Spatial Mapping. Hierdoor worden de gescande ruimtes opgeslagen, zodat bij een later gebruik deze data makkelijk terug gehaald kan worden en de gebruiker de omgeving niet opnieuw hoeft te scannen. Een kans hierbij is dat de gebruiker in een ander applicatie door meerdere kamers heen is gelopen en dat de scan van alle kamers met elkaar verbonden zijn.

Hiervoor is gekeken of het mogelijk is of de “Caching” uit te zetten in de applicatie, maar dit was niet mogelijk. De enigste controle die over de scanner aanwezig is, is de mogelijkheid de scanner aan of uit te zetten. Hierbuiten is er totaal geen controle over de scanner of over eerder gescande ruimtes. Dit betekent dat alleen de gebruiker de eerder gemaakte scans kan wissen via de instellingen scherm van de HoloLens. Hierdoor zou de gebruiker dus per se scans moet verwijderen om de applicatie te gebruiken, terwijl de gemaakte scans handig kunnen zijn bij het gebruik van andere applicaties.

Om dit probleem op te lossen moest er een oplossing gevonden worden en is het besproken met de opdrachtgever. Uiteindelijk is besloten een optie toe te voegen, waarbij de gebruiker markers kan plaatsen op de hoeken van de kamer. Hierdoor heeft de gebruiker meer controle over de hoeken die gebruikt gaan worden en in het geval dat de indeling van een kamer voor verwarring zorgt, waardoor de automatische herkenning niet correct werkt, heeft de gebruiker de optie de hoeken handmatig op te geven.

Deze toevoeging zal als extra mogelijkheid naast het automatisch berekenen toegevoegd worden, zodat de functionaliteit van de applicatie niet verminderd zal worden.

4.4.4 CONCLUSIE EN AANBEVELING

Uit het onderzoek is gebleken dat het mogelijk is de dataset te koppelen met de omgeving van de gebruiker door middel van de HoloLens. Aangezien het een nieuwe techniek is, was het niet duidelijk wat er te verwachten was en dit heeft ook geresulteerd in een aantal verschillende aanpassingen om de koppeling in de meeste gevallen correct te kunnen laten verlopen.

Hierbij kan wel gezegd worden dat het overduidelijk is dat de techniek nog heel nieuw is en nog in de eerste fase zit. De afmetingen die verkregen worden moeten handmatig berekend worden en er is geen controle over de scan die gemaakt wordt, buiten het feit dat de scanner aan of uit gezet kan worden. Hierdoor is de applicatie afhankelijk van de gebruiker en kunnen nieuwe technieken zoals automatische hoek detectie niet makkelijk gebouwd worden.

Al met al is het wel gelukt een dataset te koppelen aan de omgeving van de gebruiker door middel van automatische hoek detectie, mits de gemaakte scan goed overeenkomt met de omgeving en de ruimtes om de kamer heen niet gescand zijn. Als deze scan niet goed is kan de gebruiker handmatig de hoeken opgeven, waardoor het vervolg van de applicatie wel correct uitgevoerd kan worden.

5 TOOLS EN TECHNIEKEN

Tijdens het onderzoek zijn er meerdere nieuwe tools en technieken onderzocht die gebruikt kunnen worden tijdens de ontwikkeling. In dit hoofdstuk zullen al deze gekozen tools en technieken genoemd en beschreven worden, zodat het later in het verslag duidelijk is wat er gebruikt is en wat de werking hiervan is.

5.1 UNITY

Unity of Unity3D is een Multi-platform game engine ontwikkeld door Unity Technologies. De eerste versie was aangekondigd in 2005 en was origineel bedoeld voor ontwikkeling voor de OS X platform (Unity WWDC, 2005). Sindsdien zijn er vijf grote Unity versies uitgebracht en worden meer dan twintig platformen ondersteund (Unity Multi Platform, n.d.).

Zoals tijdens het onderzoek in hoofdstuk 4.2 al was beschreven is Unity de enige engine die out-of-the box werkt met de Microsoft HoloLens. De tools die beschikbaar zijn in Unity zorgen ervoor dat de ontwikkelaar makkelijk acties kan ondernemen, maar dit zorgt ook voor veel verwarring aangezien het een andere indeling is dan dat de meeste ontwikkelaars gewend zijn. Om het makkelijker te begrijpen zal er in het kort uitgelegd worden wat voor schermen er beschikbaar zijn in Unity en wat het doel is van elk van de schermen.

De eerste scherm is het “Project” scherm, zie Bijlage Unity 01, en dit scherm bevat alle objecten of Assets zoals Unity ze noemt. Dit zijn objecten zijn de objecten die beschikbaar zijn voor direct gebruik in het project en aangezien ze in een scherm te vinden zijn, zijn ze makkelijk terug te vinden. Ook is het mogelijk de verschillende soorten objecten op een gestructureerde manier op te slaan.

De objecten die je in het “Project” scherm hebt staan, worden niet direct gebruikt in de scene (wereld in Unity) van de applicatie. Hiervoor moeten deze objecten eerst in de scene geplaatst worden. De objecten die in de scene geplaatst zijn, zijn terug te vinden in het “Hierarchy” scherm, zie Bijlage Unity 02. Dit scherm heeft als doel de objecten die in de wereld staan weer te geven en de gebruiker de optie te geven deze objecten te selecteren en te structureren zonder ze in de wereld te hoeven vinden en selecteren.

Als een object geselecteerd wordt in het “Project” of “Hierarchy” scherm komen de object attributen tevoorschijn in het “Inspector” scherm die te zien is in Bijlage Unity 03. Dit scherm is er om aanpassingen aan objecten te brengen zonder code te hoeven schrijven. Dit kunnen simpele aanpassingen zoals posities, rotatie en schaal zijn van het object in de scene, maar ook object specifieke instellingen die door middel van code toegevoegd kunnen worden. Hierdoor hoeft een waarde van een parameter in een stuk code niet in de code zelf aangepast te worden, maar kan het direct in het “Inspector” scherm aangepast worden.

Het laatste scherm is eigenlijk een vorm van twee verschillende perspectieven, namelijk het “Scene” en “Game” scherm. Deze schermen zijn te vinden in Bijlage Unity 04 en Unity 05. Het doel van deze schermen is het resultaat van de ontwikkeling te weergeven. Het verschil tussen de twee is dat in het “Scene” scherm de wereld in een vrije weergave bekeken kan worden. Hierdoor kan een object dat in de scene geplaatst is vanuit verschillende hoeken bekeken worden.

Het “Game” scherm daarentegen laat de scene zien vanuit de perspectief van de Camera die geplaatst is in de wereld. Deze camera is eigenlijk de positie van de gebruiker tijdens het uitvoeren van de applicatie, dus in dit geval de positie van de gebruiker die de Microsoft HoloLens op heeft. Hierdoor kan de ontwikkelaar voordat de applicatie gepubliceerd wordt de applicatie testen en zien wat de gebruiker uiteindelijk te zien krijgt.

5.2 VISUAL STUDIO

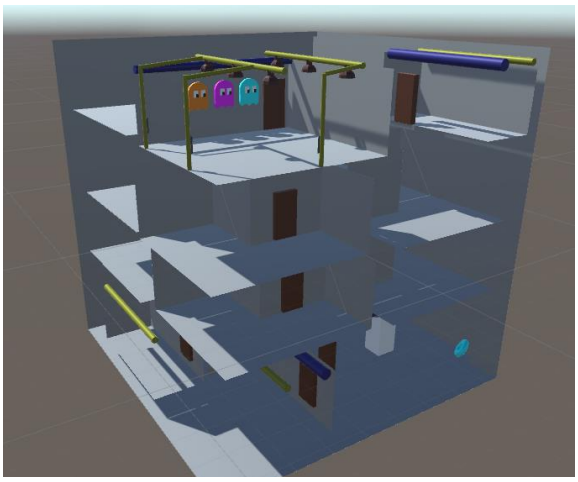
Een IDE is een tool voor het schrijven voor Code. De toevoeging van het gebruiken van een IDE in plaats van een simpele tekstverwerker is dat een IDE vele extra's heeft. Door de automatische toevoeging van kleuren en witregels wordt de code ten eerste beter leesbaar en ten tweede helpt de ingebouwde code controle om fouten in code op te sporen en te verbeteren.

De IDE die gebruikt is, is Visual Studio of Microsoft Visual Studio. Dit is een IDE van Microsoft waarvan de eerste versie aangekondigd en uitgebracht is in 1997 (Microsoft Systems Journal, 1997). Deze versie had de versie nummer 5.0. Dit komt omdat Visual Studio daarvoor veel verschillende namen heeft gehad en de eerste versie die Visual Studio heet versie 5.0 is. Sinds de eerste Visual Studio in 1997 zijn er veel verschillende nieuwe versies uitgebracht, waarvan de laatste versie tijdens het project uitgebracht is, namelijk Visual Studio 2017.

Aan het begin van het project is er gebruik gemaakt van Visual Studio 2015, aangezien de nieuwste versie nog niet officieel uitgebracht was. Ook waren de andere tools die nodig waren voor de ontwikkeling voor de Microsoft HoloLens nog niet compatibel met de nieuwste versie van Visual Studio. Nadat dit wel het geval was is er overgestapt naar de nieuwste versies. Dit was niet zonder problemen, zie paragraaf 7.8.1.

5.3 AUTODESK MAYA

Het doel van het project is het koppelen van een 3D model met de Microsoft HoloLens, maar deze 3D modellen bestaan nog niet. Daarom zullen deze modellen gemaakt moeten worden in een 3D modelleer softwarepakket. Zoals in paragraaf 4.3.2 al beschreven was, is er gebruik gemaakt van AutoDesk Maya. Dit is een programma om 3D modellen te maken en dit is één van de vele verschillende programma's die bestaan voor het creëren van 3D modellen. De reden dat er voor deze softwarepakket gekozen is, is door eerdere ervaringen en buiten het proces van het creëren van de modellen zullen de uitkomsten bij de meeste 3D modelleer applicaties hetzelfde zijn. Voor een voorbeeld van de interface zie Bijlage Maya 01.



Figuur 5.1: Model van het gebouw, gemaakt in AutoDesk Maya

Het model dat hiermee gemaakt is, is een gebouw met meerdere kamers. Elke kamer in het gebouw heeft een verschillende afmeting en ook verschillende attributen. Ook zijn er kamers met dezelfde afmetingen, zodat dit ook getest kan worden in de applicatie. In figuur 5.1 kan het gecreëerde model gezien worden.

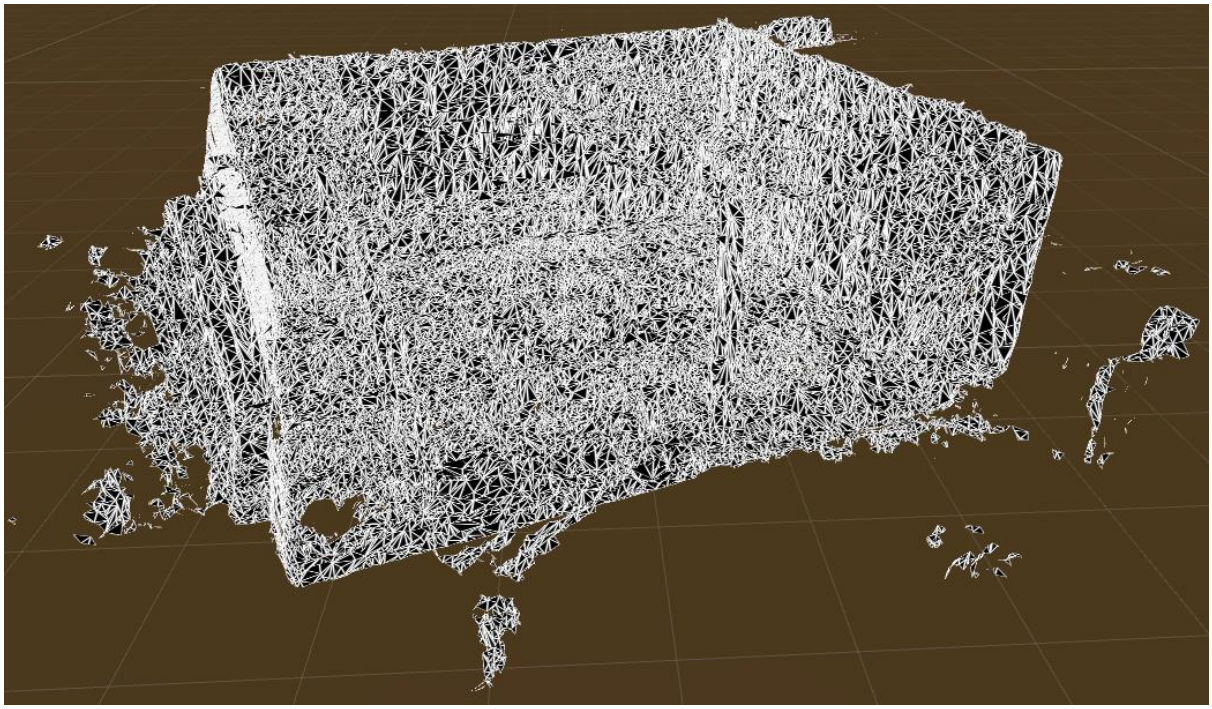
5.4 GIT(HUB)

Tijdens het ontwikkelen is er gebruik gemaakt van git als versiebeheer, waarbij GitHub is gebruikt als host. Dit is een tool die alleen gebruikt is voor beheer tijdens de ontwikkeling en zal dan ook niet verder terugkomen in het vervolg van het verslag.

5.5 HOLOLENS SPATIAL MAPPER

Een van de technieken die de Microsoft HoloLens uniek maakt is de mogelijkheid de omgeving van de gebruiker te scannen. Dit wordt Spatial Mapping genoemd door Microsoft (Microsoft Spatial Mapping, n.d.). In essentie is dit het onderdeel dat het verschil is tussen Augmented- en Mixed Reality. Doordat de HoloLens de mogelijkheid heeft de omgeving van de gebruiker vast te leggen, heeft de ontwikkelaar meer data van de omgeving waar de applicatie gebruikt gaat worden. De ontwikkelaar kan de omgeving van de gebruiker kan gebruiken, om bijvoorbeeld te zoeken worden naar de dichtstbijzijnde muur om hier vervolgens een Hologram van een schilderij te projecteren.

De werking van de scanner wordt niet uitgelegd door Microsoft, maar door het gebruik van de scanner kan de gebruiker een idee krijgen hoe het ongeveer in zijn werk gaat. De HoloLens heeft namelijk verschillende camera's voor het herkennen van de omgeving en het meten van de diepte. Hieruit valt op te maken dat de HoloLens door middel van de omgeving herkenning camera's zijn eigen positie kan bepalen en wat voor obstakels op verschillende richtingen aanwezig zijn. Vervolgens wordt er met de diepte camera gekeken hoeveel afstand tot de obstakels aanwezig is. Door deze data te combineren kan er vervolgens een "Mesh" gecreëerd worden zoals Microsoft het noemt. Deze Mesh bestaat uit heel veel driehoeken die geplaatst worden op de posities van de obstakels. In figuur 5.2 kan de uiteindelijk gescande Mesh van de HoloLens gezien worden.



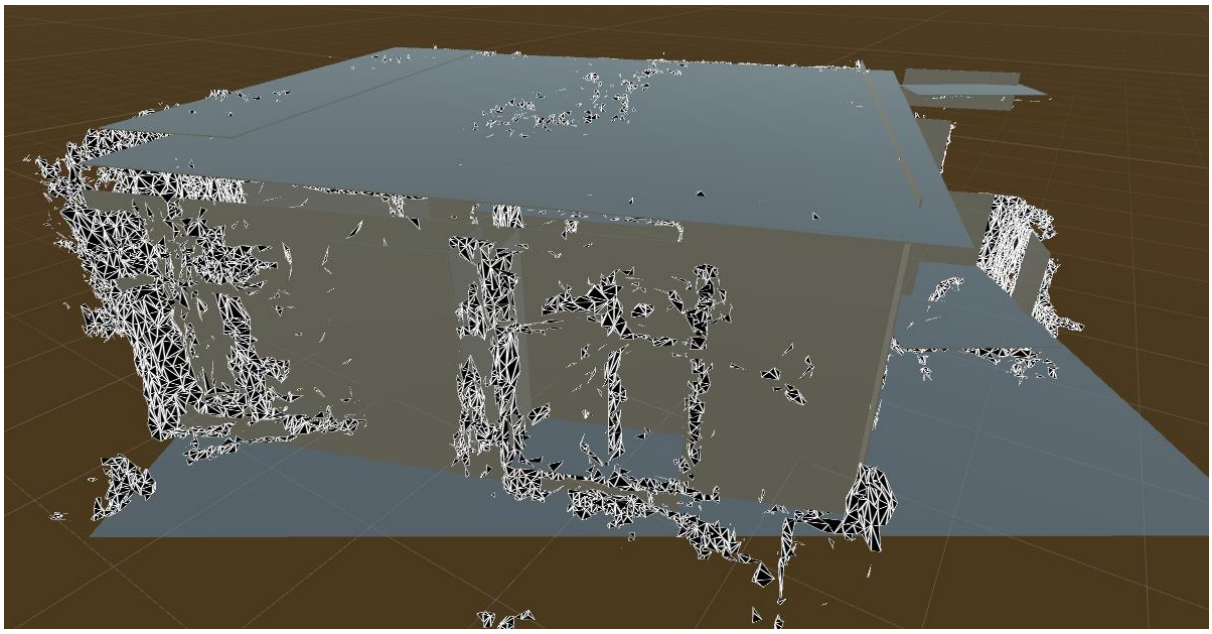
Figuur 5.2: HoloLens Mesh Scan

5.6 HOLOLENS MESH PROCESSING

Als de HoloLens klaar is met het scannen van de omgeving van de gebruiker, is er een object/Mesh waar alles met elkaar verbonden is. Deze Mesh kan gebruikt worden voor het berekenen van de afmetingen, maar omdat de HoloLens zo veel mogelijk informatie probeert vast te leggen komt het vaak voor dat er driehoekjes geplaatst worden buiten de werkelijke scan van de omgeving, zie Figuur 5.2. Hierdoor kan het voorkomen dat de berekeningen die uitgevoerd worden niet accuraat zijn. Ook kan er niet opgevraagd worden waar de muren of tafels in de kamer zich bevinden.

Om dit soort data op te kunnen vragen moet de HoloLens nog een laatste stap uitvoeren, namelijk het Processen van de Mesh (Microsoft Mesh Processing, n.d.). Het processen/verwerken zorgt ervoor dat er volgens de gescande Mesh extra panelen geplaatst worden. Deze panelen worden geplaatst over de gescande Mesh, zodat het dezelfde vorm blijft behouden en omdat het gevormd is door meerdere panelen kan er meer informatie opgevraagd worden.

De panelen die geplaatst worden kunnen verschillende categorieën hebben, zoals vloer, muur, tafel of plafond. Door de panelen die geplaatst zijn kan de ontwikkelaar makkelijk opvragen of er bijvoorbeeld een tafel aanwezig is in de kamer. Ook kan er per paneel afmetingen opgevraagd worden, waardoor alleen de afmetingen van een muur of vloer van de kamer opgevraagd kan worden. In figuur 5.3 kunnen de panelen over de gescande Mesh gezien worden en ook de overeenkomsten qua positie, rotatie en groottes zijn in dit figuur te zien.



Figuur 5.3: Panelen over Mesh heen

6 ARCHITECTUUR

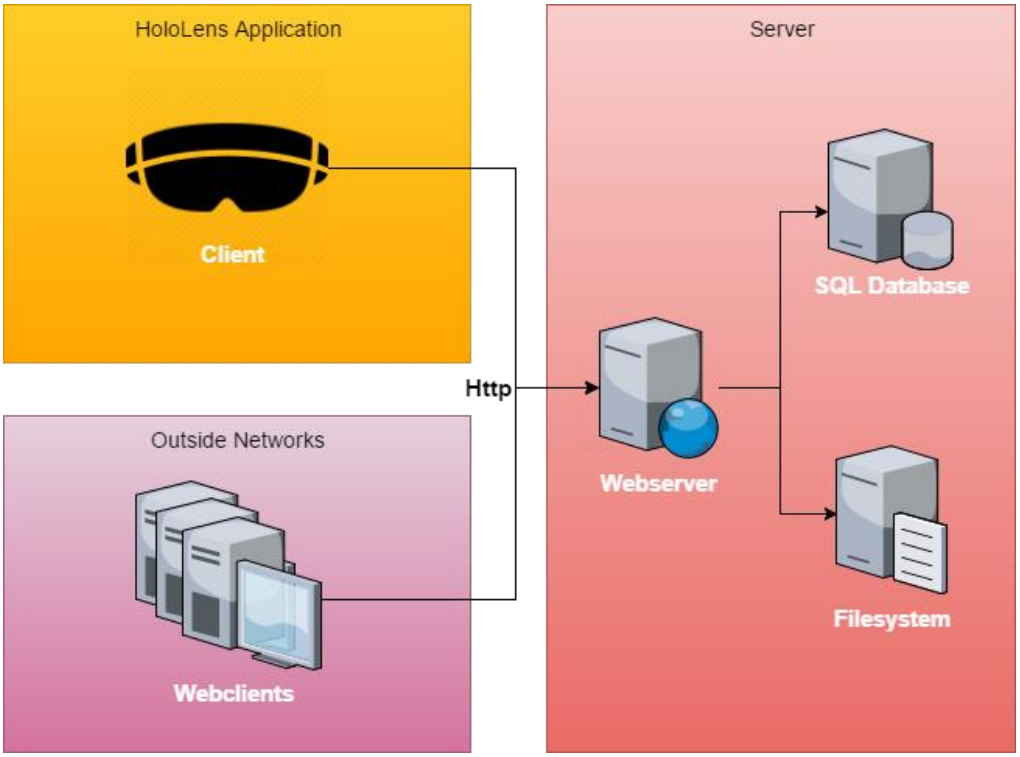
Om tijdens de ontwikkelfase zo min mogelijk problemen tegen te komen, zal er geprobeerd worden zoveel mogelijk aspecten van de applicatie in dit hoofdstuk te beschrijven. Dit zal gedaan worden door middel van verschillende diagrammen en onderzoeksbevindingen.

Tijdens dit proces zullen ook kleinere onderzoeken uitgevoerd worden, maar deze zullen niet zo uitgebreid een grootschalig zijn als de onderzoeken in hoofdstuk 4. Het doel van deze kleinere onderzoeken is dan ook de laatste paar aspecten van de applicatie te dekken, zodat er tijdens de ontwikkeling geen keuzes gemaakt hoeven te worden.

6.1 SYSTEEM ARCHITECTUUR

Tijdens het uitwerken van de architectuur van de applicatie is er ontdekt dat HoloLens geen logische filesystem heeft, waardoor een oplossing bedacht moest worden. Deze oplossing was het gebruik van een server. Maar omdat dit een keuze was dat niet zomaar gemaakt kon worden zou er aan het einde van de sprint met de opdrachtgever besproken moeten worden of het een goede oplossing zou zijn binnen het project. Om voorlopig door te kunnen en omdat dit de beste oplossing leek is er besloten voorlopig door te gaan op deze pad. De toevoeging van een server zou ervoor zorgen dat er een extra systeem gebouwd moest worden en hiervoor kan gekeken worden naar figuur 6.1, waarin de verbanden te zien is tussen de verschillende systemen, die tijdens dit project ontwikkeld zullen worden.

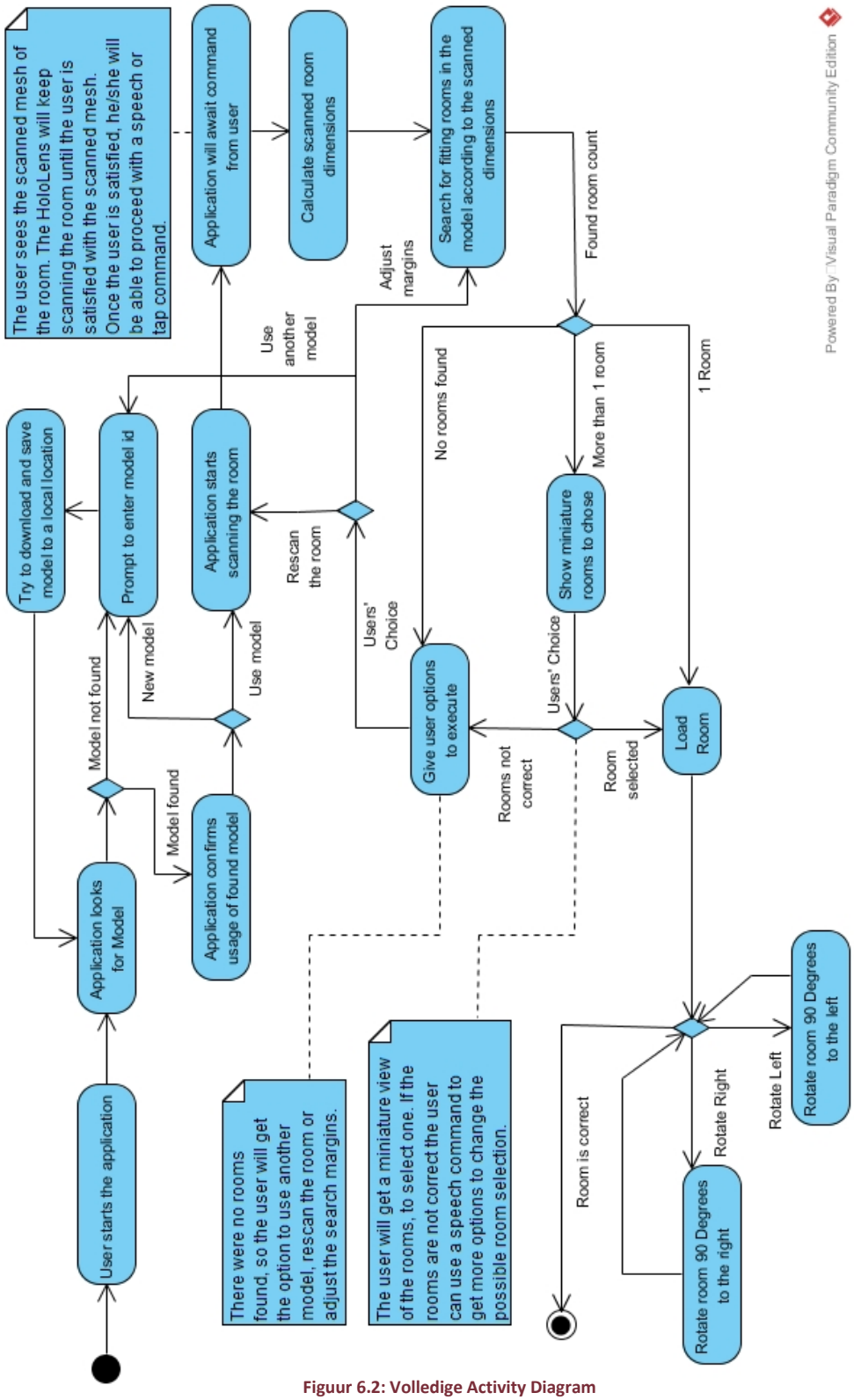
Hierbij zijn webclients te zien die via de webserver een pagina binnen kunnen halen, waarmee vervolgens de modellen geüpload kunnen worden. Deze modellen worden op de juiste manier opgeslagen door de webserver. De geüploade modellen zijn vervolgens door HoloLens (client) op te vragen aan de webserver door middel van een http request.



Figuur 6.1: High Level System Architecture

6.2 HOLOBUILDING WORKFLOW

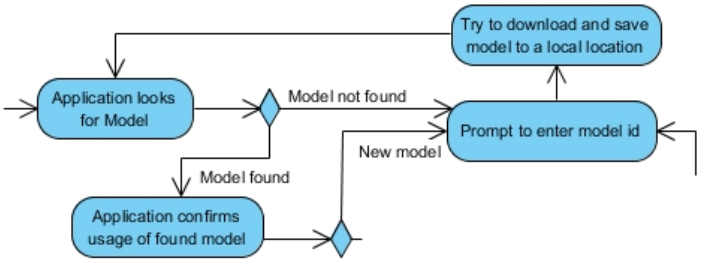
Dit paragraaf zal de uiteindelijke workflow van de HoloLens applicatie beschrijven, die in figuur 6.1 afgebeeld staat. Hierbij kan Figuur 6.2 gebruikt worden als ondersteuning, dit is een Activiteiten diagram waarin alle mogelijke paden van de applicatie afgebeeld staan.



Figuur 6.2: Volledige Activity Diagram

6.2.1 MODEL

Bij het opstarten zal de applicatie kijken of er een model aanwezig is. Deze model kan bij vorig gebruik al opgehaald zijn en als dit het geval is, is er een grote kans dat de gebruiker het model nogmaals wilt gebruiken. Indien dit niet het geval is, zal de gebruiker een model ID moeten invoeren, die te verkrijgen is via de server die in paragraaf 6.4 beschreven zal worden. Door middel van deze ID kan de applicatie de modellen opvragen bij de server en indien deze bestaat, zal deze opgehaald en opgeslagen worden op de HoloLens.



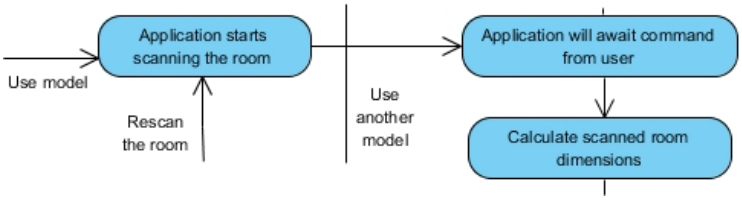
Figuur 6.3: Model selectie

Indien het model al aanwezig is, kan er aan de gebruiker gevraagd worden of deze model gebruikt moet worden. Als de gebruiker dit niet wilt zal dezelfde flow als hierboven beschreven uitgevoerd worden, waarbij er door middel van een model ID een model opgehaald zal worden. Hiermee voldoet de applicatie aan UR01, namelijk de mogelijkheid voor de gebruiker om modellen te kunnen selecteren.

Nadat een model gereed is, dus door een model dat hergebruikt kan worden of een nieuw opgehaald model, zal de volgende proces uitgevoerd worden, namelijk het scannen van de kamer.

6.2.2 SCANNEN VAN DE KAMER

Volgens UR02 moet de gebruiker een kamer kunnen scannen en dit is ook een essentieel onderdeel van de applicatie. Bij het scannen van de kamer, zie hoofdstuk 5.6, zal de HoloLens in een scan modus staan, waarbij er afgewacht zal worden tot de gebruiker tevreden is met het resultaat. Door de lenzen van de HoloLens zal de gebruiker een Wireframe zien over de kamer heen, zie figuur 5.2, waarmee de gebruiker kan zien hoe de kamer gescand is. Zodra deze scan compleet is kan de gebruiker de volgende stap initiëren, namelijk het verwerken (“processing”, zie hoofdstuk 5.6) van de kamer.



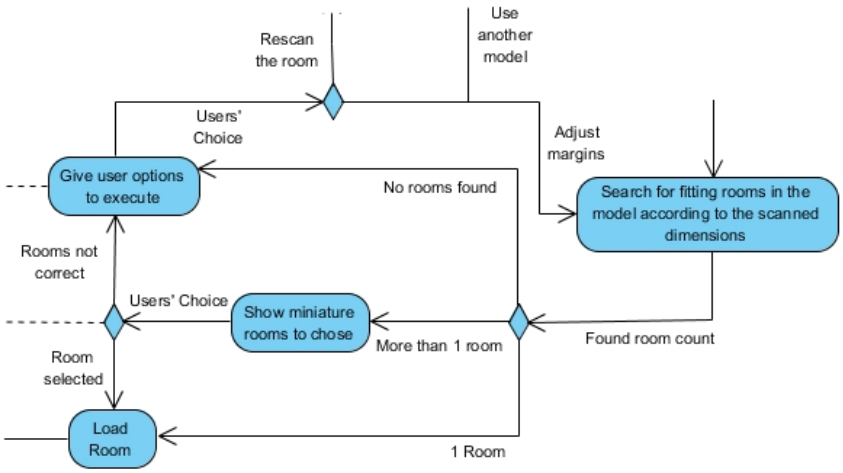
Figuur 6.4: Scan Activity

Nadat de gebruiker het verwerken heeft gestart, wordt het scannen gestopt en worden de muren, vloer en plafond van de kamer berekent. Deze berekeningen zullen helpen bij andere berekenen, zoals de afmetingen, positie en rotatie van de kamer. Door middel van deze data kan er in het model dat is opgehaald en ingeladen, een kamer geselecteerd worden volgens het proces dat in paragraaf 4.4.2 is beschreven.

6.2.3 SELECTEREN VAN EEN KAMER

Naast het selecteren van een model en het scannen van een kamer moet de gebruiker ook de mogelijkheid krijgen een kamer te selecteren die ingeladen moet worden, wat gespecificeerd is in UR04. Om een kamer te kunnen selecteren heeft de applicatie twee datasets nodig, namelijk een model om in te kunnen zoeken en een dataset waar data als afmetingen, rotatie en posities in staan. Door middel van deze datasets kan de gebruiker in het model zoeken naar kamers die overeenkomen.

Uit deze zoekopdracht kunnen drie uitkomsten komen, namelijk geen kamer gevonden, één kamer gevonden en meerdere kamers gevonden.



Figuur 6.5: Kamer Selectie

GEEN KAMER

Indien er geen kamers gevonden zijn, krijgt de gebruiker een aantal mogelijkheden om de zoekopdracht te verbeteren. De eerste mogelijkheid is het ophalen van een ander model. Dit is handig in het geval dat de gebruiker een model heeft gebruikt dat bijvoorbeeld niet bij het gebouw hoort waar de gebruiker zich op dat moment bevindt.

De tweede mogelijkheid is het opnieuw scannen van de kamer. Door de werking van de scanner kan het zijn dat de uiteindelijke berekende afmetingen van de kamer niet kloppen. Dit kan natuurlijk meerdere oorzaken hebben, maar in de meeste gevallen is het omdat het gescande model niet klopt.

De laatste mogelijkheid is het aanpassen van de toegestane tolerantie bij de zoekopdracht. Aangezien de HoloLens nog vrij nieuw is, zijn de afmetingen die verkregen worden van de kamer waar de gebruiker zich bevindt niet honderd procent nauwkeurig. Daarom zal er door middel van een tolerantie een marge vastgesteld worden, waarmee de verschillen in afmetingen onder de tolerantie acceptabel zijn. Hierdoor zal het toch mogelijk zijn de kamer te vinden die hoort bij de kamer waar de gebruiker zich op dat moment bevindt.

MEERDERE KAMERS

In het geval dat er meerdere kamers gevonden zijn, die voldoen aan de afmetingen zal de applicatie al deze kamers in een miniatuurformaat weergeven, kan de gebruiker selecteren welke kamer ingeladen moet worden. Als een gebruiker een van de kamers selecteert, wat door middel van de interactiemogelijkheden zal gebeuren die in paragraaf 6.3.4 uitgelegd zal worden, zal deze kamer ingeladen worden. De andere kamers zullen hersteld worden naar de originele waardes en zullen onzichtbaar gemaakt worden, waardoor de gebruiker alleen de geselecteerde kamer te zien krijgt.

Indien de gebruiker de gevonden kamers niet juist vindt kunnen de opties, die in het vorige paragraaf toegelicht waren, uitgevoerd worden waarbij de gebruiker de mogelijkheid krijgt de uitkomsten van de zoekopdracht te verbeteren.

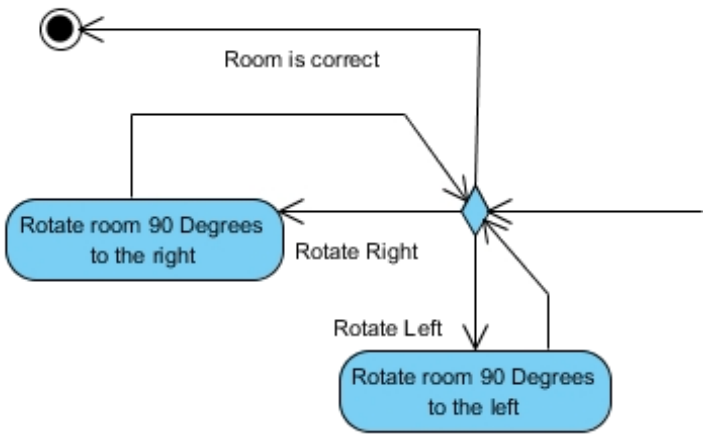
ÉÉN KAMER

Zodra een kamer gevonden of geselecteerd is zal deze ingeladen worden, waarbij een aantal aanpassingen aan de kamer gebracht zullen worden, zodat het ingeladen wordt over de kamer waar de gebruiker zich bevindt. Deze aanpassingswaarden zullen verkregen worden volgens de berekeningen in hoofdstuk 4.4. Dit zijn waardes zoals het middelpunt van de kamer waar de gebruiker zich bevindt, en de rotatie van de kamer. Deze rotatie is belangrijk, aangezien deze rotatie ervoor zal zorgen dat de muren van de kamer en model met elkaar overheen komen.

6.2.4 VERBETERINGEN

Door middel van de verschillende berekeningen zal de kamer zo correct mogelijk ingeladen worden, maar in de huidige stadium kan de HoloLens een aantal elementen niet onderscheiden, zoals deuren en ramen. Hierdoor is het niet mogelijk een zeer nauwkeurige berekening te maken en zoals beschreven in paragraaf 4.4.2 zal dit aan het einde van sprint 3 besproken worden (de oplossing waarbij de gebruiker de laatste zeggenschap krijgt over de rotatie werd uiteindelijk geaccepteerd en er werd een nieuwe requirement opgesteld, namelijk PR05, die vanaf sprint 4 meegenomen zou worden. Meer hierover in de volgende sprintovergang).

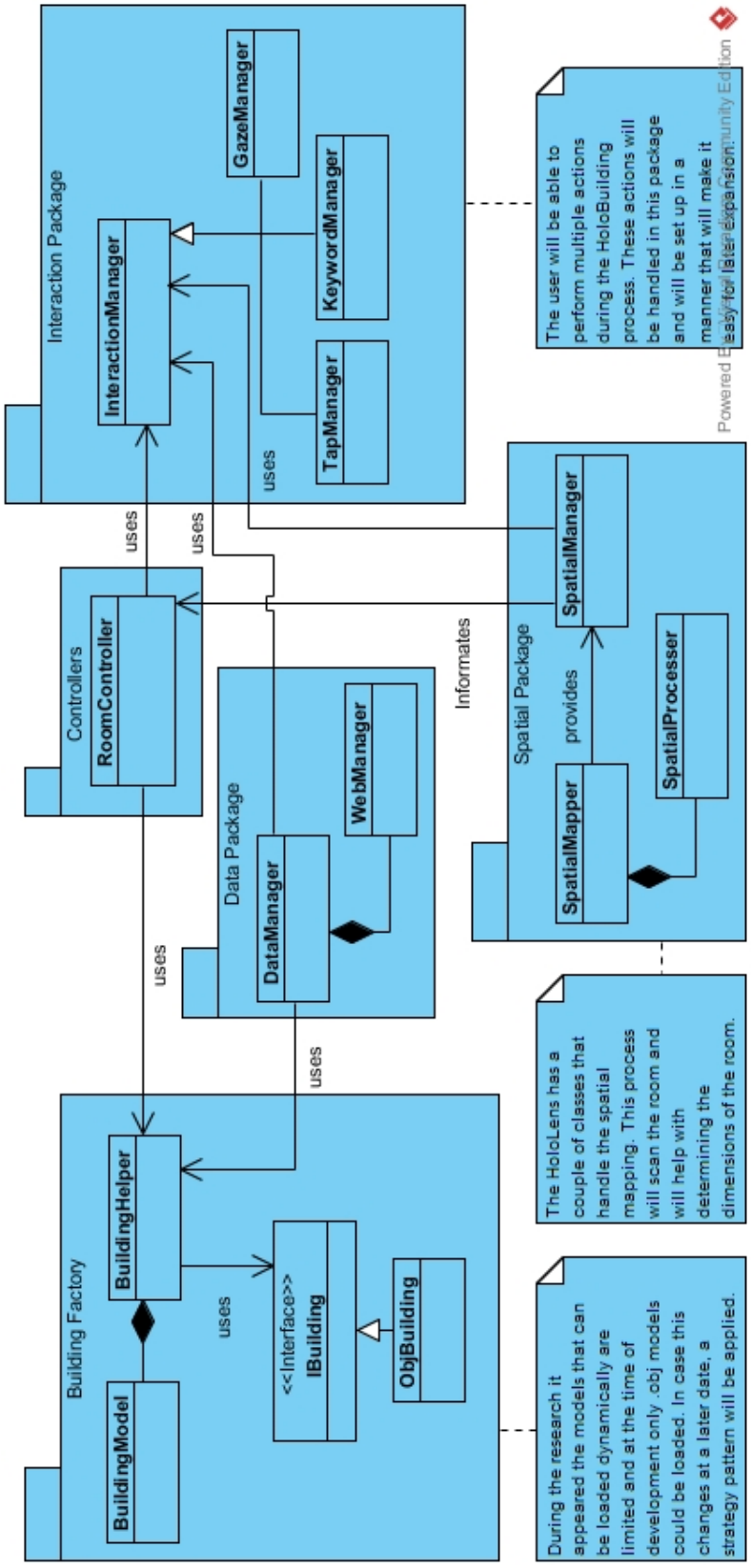
Dit zou dus betekenen dat de ingeladen kamer uiterlijk negentig of honderdtachtig graden (twee keer negentig graden) te ver gedraaid is, waardoor de deur bijvoorbeeld weergegeven kan worden aan de andere kant van de kamer. Om dit soort problemen te kunnen verhelpen zal de gebruiker als laatst de mogelijkheid krijgen om de kamer te draaien. Deze acties kunnen oneindig uitgevoerd worden tot de gebruiker tevreden is met het resultaat.



Figuur 6.6: Laatste acties

6.3 HOLOBUILDING

Om alle benodigde objecten en klassen van de applicatie in kaart te brengen is er een klassendiagram gemaakt die te zien is in figuur 6.7. Als aanvulling van dit diagram zal er in dit hoofdstuk beschreven worden wat het doel is van alle klassen en wat de verbanden onderling zullen zijn. Volgens het figuur 6.1 zal dit hoofdstuk betrekking hebben tot de client/HoloLens applicatie.



Figuur 6.7: Volledige Klassen Diagram

6.3.1 DATA PACKAGE

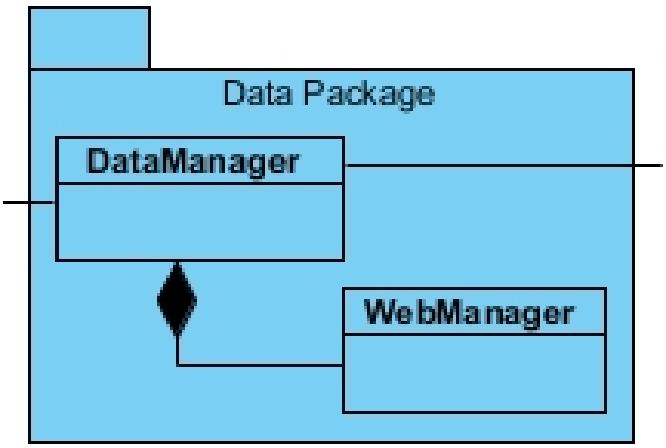
Tijdens de uitwerking van de architectuur was het duidelijk geworden dat een essentieel onderdeel van de functionaliteit niet onderzocht was, namelijk hoe modellen op de HoloLens terecht zouden komen. Hierdoor is er een kort onderzoek gedaan, naar of het mogelijk is modellen direct op de HoloLens te plaatsen. Dit was mogelijk, maar omdat er geen logische filesystem aanwezig is, kan de gebruiker geen eenvoudig stappenplan volgen om de modellen op de juiste locatie te plaatsen.

Om de modellen toch op de HoloLens te kunnen plaatsen is er dus gekeken naar alternatieve manieren, namelijk het gebruik van een server-cliënt model, zie figuur 6.1. Hierbij kan de gebruiker via een website het model uploaden naar een server, waarna een code gebruikt worden om het model te verkrijgen van de server. Voor de uitwerking van deze server zie hoofdstuk 6.4.

De data package is eigenlijk wat als eerst actief is binnen de applicatie. Dit omdat de datapackage de DataManager class bevat. Deze class weet waar het model te vinden is en regelt ook de interacties die nodig zijn met de gebruiker om een nieuw model op te halen. Als een nieuw model nodig is, kan er aan de InteractionManager een code opgevraagd worden, die de gebruiker de optie zal geven een code in te voeren. Vervolgens zal de code gebruikt worden om het model op te halen van de HoloBuilding Server.

Als de gebruiker een code heeft ingevoerd en deze code aangekomen is bij de DataManager kan er door middel van de WebManager, die overigens alleen te bereiken is door de DataManager, een verzoek gestuurd worden naar de server. Als het model gevonden is, zal het model teruggestuurd worden. Deze wordt vervolgens door de WebManager naar de DataManager gestuurd, waar het model wordt opgeslagen op een locatie dat toegankelijk is voor de applicatie.

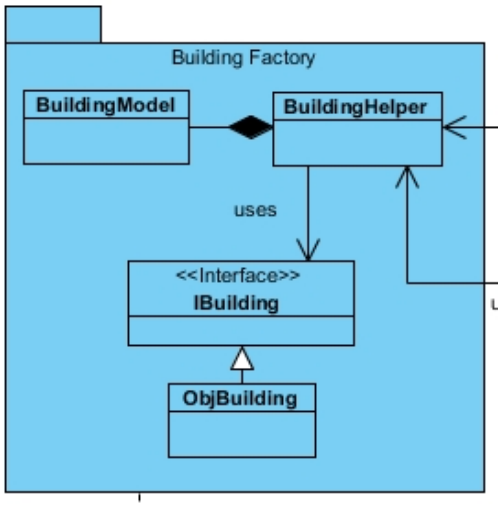
Zodra er een model is opgehaald en dit model in gebruik genomen kan worden, zal de locatie van het model doorgestuurd worden naar de BuildingHelper in de Building Factory. Hier zal het model ingeladen en uitgelezen worden.



Figuur 6.8: Data Package

6.3.2 BUILDING FACTORY

De Building Factory is eigenlijk een pakket dat ervoor zorgt dat modellen op een eenvoudige manier ingeladen en uitgelezen kunnen worden. Tijdens het project zullen alleen OBJ-modellen ondersteund worden, maar de mogelijkheid is er nog om later de ondersteuning voor meerdere modelsoorten toe te voegen. Hierom zal de Building Factory zo generiek mogelijk opgesteld worden.



Figuur 6.9: Building Factory

De eerste stap om een generiek package op te stellen begint met de IBuilding interface. Deze interface is er om de structuur die verschillende importeurs moeten leveren na te komen. Door de verschillende importeurs te laten overerven van de interface kan de BuildingHelper een doel bereiken, terwijl deze doel door de importeurs op verschillende manieren behaald zal worden. Dit is dan ook een implementatie van het “Factory Pattern”.

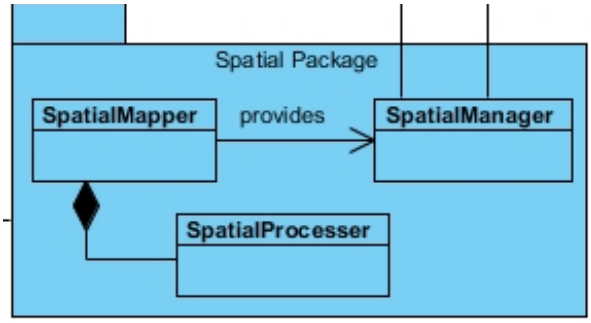
Een voorbeeld hiervan is dat de ObjBuilding op manier X een OBJ-model inlaadt, terwijl een toekomstige FBXBuilding op manier Y een FBX-model inlaadt. Maar omdat ze allebei volgens de IBuilding interface te werk gaan, kan de BuildingHelper in beide gevallen de objecten op dezelfde manier opvragen.

Als de BuildingHelper een model binnenkrijgt zal er gekeken worden wat voor modeltype het is. Gebaseerd op deze type, zal de juiste class geïnstantieerd worden. Zo zal er bij een OBJ-model de ObjBuilding gebruikt worden. Vervolgens zal de ObjBuilding een model leveren die makkelijk uit te lezen is door de BuildingHelper. Op deze manier zullen de requirements SR01 en SR02 verwerkt zijn in de applicatie.

Het model dat uiteindelijk verkregen wordt, zal door BuildingHelper in een BuildingModel geplaatst worden. Deze BuildingModel zal alleen toegankelijk zijn voor de BuildingHelper en alle andere classes die informatie over de BuildingModel willen hebben kunnen via de BuildingHelper deze data op een eenvoudige manier opvragen. Hierdoor zal alle logica dat benodigd is voor de BuildingModel op één centrale locatie staan. Dit zal ook zorgen voor consistentie door de applicatie heen.

6.3.3 SPATIAL PACKAGE

De Spatial Package is een package dat ervoor zal zorgen dat de gebruiker de requirement kan uitvoeren die opgesteld staat in UR02. Deze package wordt grotendeels geleverd door Microsoft en zorgt ervoor dat de scan functionaliteit van de HoloLens aangesproken kan worden en dat deze scan verwerkt kan worden. Alles wat benodigd is in dit package, kan verkregen worden via Microsoft. Om deze Spatial Package te kunnen managen is het wel nodig een SpatialManager te maken, waarin de scan en verwerk functionaliteit van de HoloLens makkelijk aangesproken en aangestuurd kan worden.

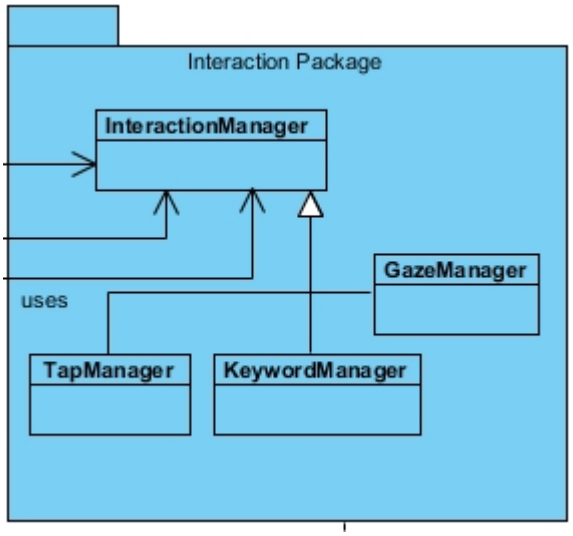


Figuur 6.10: Spatial Package

Ook moet er in deze class een aantal berekeningen uitgevoerd worden, die gebruikt zullen worden bij het inladen van een kamer. In hoofdstuk 4.4 staat beschreven hoe deze berekeningen uitgevoerd kunnen worden.

6.3.4 INTERACTION PACKAGE

De HoloLens heeft een aantal manieren om te communiceren met de gebruiker. Deze manieren zullen in de Interaction Package op een generieke manier geïmplementeerd worden, zodat de rest van de applicatie op een eenvoudige wijze een voorstel kan voorleggen aan de gebruiker. Deze voorstel kan de gebruiker dan beantwoorden, waarop een actie ondernomen wordt door de applicatie.



Figuur 6.11: Interaction Package

De eerste manier om met de gebruiker te communiceren is door middel van de KeywordManager. Deze class stelt de gebruiker in staat door middel van spraak commando's acties te ondernemen. Hierbij kunnen een aantal commando's van de gebruiker gedefinieerd worden en de acties die uitgevoerd moeten worden als deze commando's aangeroepen worden. Deze spraakcommando's moeten op een eenvoudige wijze tijdens de looptijd van de applicatie aangepast kunnen worden, waardoor er dus op bepaald punten in de applicatie extra commando's toegevoegd en verwijderd kunnen worden.

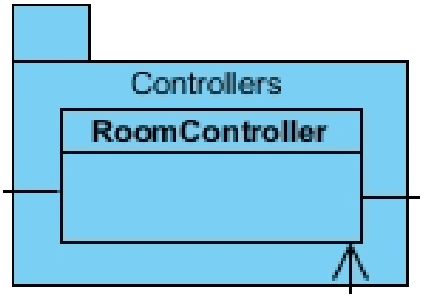
De tweede communicatiemethode is doormiddel van de kijkpunt van de gebruiker. Aangezien de HoloLens een headset is dat zicht bevindt op de hoofd van de gebruiker, weet de HoloLens waar de gebruiker ongeveer naar kijkt. Dit punt kan door middel van een cursor ook in de HoloLens aan de gebruiker getoond worden, waardoor de gebruiker ook weet waar de HoloLens zich op dat moment op concentreert. Hierdoor kan er bijvoorbeeld een Graphical User Interface in de wereld geplaatst worden waar de gebruiker door middel van knoppen commando's kan laten uitvoeren.

De derde en laatste manier van communiceren is door middel van een "Tap". Dit is een handbeweging die de gebruiker voor zijn gezicht kan uitvoeren, waaruit de HoloLens een "Klik" actie opmaakt. Door de kijkpunt hiermee te combineren kan de HoloLens begrijpen dat de gebruiker op een bepaald element klikt en hier dus bepaalde acties op uitvoeren.

Om deze communicatiemethoden bij te houden zal er een InteractionManager zijn. Deze InteractionManager zal de class zijn die door de rest van de applicatie gebruikt wordt om de verschillende interactie mogelijkheden te initiëren. Deze InteractionManager moet daarom op een eenvoudige wijze de applicatie in staat stellen de communicatiemogelijkheden te gebruiken. Om dit doel te bereiken zal de InteractionManager de verschillende communicatiemethoden tegelijkertijd inzetten. Dus als een antwoord wordt verwacht van de gebruiker kan de InteractionManager door middel van een GUI een vraag voor de ogen van de gebruiker plaatsen en de vraag uitspreken. De gebruiker kan vervolgens met spraak een van mogelijke commando's uit voeren of door middel van de kijkpunt en de "tap" een actie ondernemen. Op deze manier wordt ervoor gezorgd dat de gebruiker commando's kan uitvoeren en requirement UR03 gedekt is.

6.3.5 CONTROLLERS

De controllers zijn de knooppunten van de applicatie en dit is er maar één, namelijk de RoomController. Aangezien dit de class is die de kamer selectie begint op het moment dat het model is ingeladen en de SpatialManager klaar is met het berekenen van de benodigde data.



Figuur 6.12: Controllers

Op dit moment zal de RoomManager de afmetingen (gemeten in de SpatialManager) doorsturen naar de BuildingHelper, waar alle kamers binnen het model vergeleken zullen worden met de afmetingen. De overeenkomende kamers zullen teruggestuurd worden in een lijst.

Op het moment dat de RoomManager de overeenkomende kamers heeft ontvangen kunnen er een aantal acties ondernomen worden. Deze acties zijn gebonden aan de aantal kamers dat gevonden zijn. Als er namelijk geen kamers gevonden zijn kan er door middel van de InteractionManager gevraagd worden wat gedaan moet worden, zoals een nieuw model ophalen, opnieuw scannen of de toleranties verhogen. Gebaseerd op de keuze van de gebruiker zal de RoomManager de applicatie in de juiste staat plaatsen om de gekozen activiteit opnieuw uit te voeren.

Indien er meerdere kamers gevonden zijn, zullen deze kamers als miniaturen getoond worden aan de gebruiker, waarbij een klikbare interactie aan de gebruiker gesteld wordt. Hier kan de gebruiker dan de keuze maken welke kamer er ingeladen moet worden.

Als er uiteindelijk één kamer overblijft, of als er maar één kamer gevonden was, kan deze ingeladen worden. De kamer zal als eerst gecentreerd op het punt dat berekend is in de SpatialManager en vervolgens zullen ook de rotaties toegepast worden die ook berekend zijn in de SpatialManager. Hierbij is het belangrijk dat de positionering en rotatie op de geselecteerde kamer gebeurt via de BuildingHelper. Dit omdat deze methoden kunnen verschillen per model type, zie paragraaf 4.3.2. Door dit via de BuildingHelper te doen kan de applicatie er zeker van zijn dat de kamers op de juiste manier ingeladen worden.

SPRINT OVERGANG

Nadat de architectuur van de applicatie uitgewerkt was en een groot deel van de onderzoeken uitgevoerd was, was het einde van sprint 3 inzicht. Net zoals bij de vorige sprints is er een demo gegeven aan de opdrachtgever, waarbij de opgeleverde producten getoond zijn en aangegeven is wat er gepland staat voor de volgende sprint.

SPRINT 3

Aan het einde van het derde sprint was een groot deel van de applicatie al uitgewerkt en waren een aantal problemen naar voren gekomen die met de opdrachtgever besproken konden worden. Het eerste onderdeel dat besproken was, waren de keuzes die gemaakt zijn in de laatste onderzoek. Hierbij is uitgelegd hoe de benodigde data, zoals het middelpunt en rotatie, berekend werd en hoe deze data ingezet werd om de kamer op de juiste manier in te laden.

Hierbij werd ook aangegeven dat er een limiet zat op de Spatial Mapping techniek van de HoloLens, namelijk het probleem met de uitlijning van verkeerde muren bij de rotatie. Hierbij is uitgelegd wat voor oplossing bedacht is en dit werd geaccepteerd als de beste oplossing volgens de huidige mogelijkheden. Hierbij is er wel een nieuwe requirement geïntroduceerd, namelijk dat er alleen rekening gehouden hoeft te worden met rechthoekige kamers en deze requirement zou vanaf sprint 4 meegenomen worden met de ID PR05.

Ook zijn de gemaakte workflow- en klassendiagrammen getoond, om te laten zien hoe de applicatie gerealiseerd zou worden. Tijdens het uitwerken van de architectuur van de uiteindelijke applicatie kwam de vraag naar boven hoe de modellen op de HoloLens terecht zouden komen. Hierbij is bedacht gebruik te maken van een client-server model. Het was alleen niet duidelijk of dit een goede oplossing zou zijn voor de uitwerking van het project. Daarom is ook dit besproken met de opdrachtgever en dit was geen probleem, zolang er rekening gehouden werd met de requirements PR02 en PR03. Hierdoor zou de server in .NET gebouwd moeten worden en zou er niet te veel aandacht gestoken worden in de server, zolang het er maar voor zorgt dat de HoloLens applicatie functioneel is.

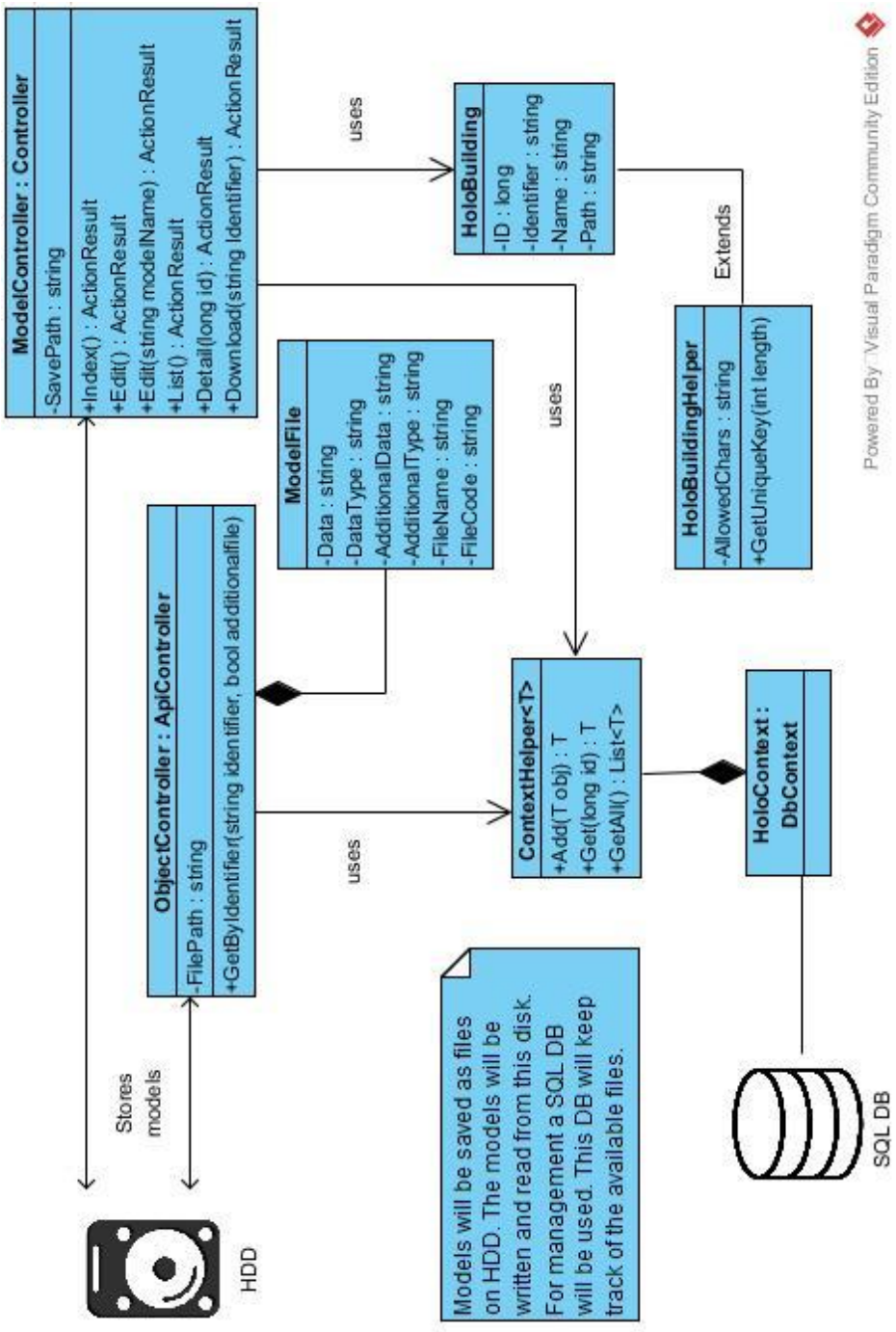
SPRINT 4

In de demo van sprint 3 is aangegeven dat een server gebouwd moest worden, daarom zal er in sprint 4 een architectuur van de server gemaakt worden en zal de server zelf ook gebouwd worden. Indien er bij het maken van de architectuur rekening gehouden werd met alle aspecten die de server zou moeten dekken, zal het maken van de server niet veel tijd in beslag moeten nemen. Dit vooral omdat er al ervaring was met het maken van servers in .NET en omdat alle features die de server moet kunnen vooraf gedocumenteerd zou zijn

Nadat de server gebouwd is zal er ook begonnen worden met de applicatie zelf. Hierbij zal de eerste onderdeel die gebouwd zal worden de Data Package zijn, Hiermee kan de server getest worden in samenwerking met de HoloLens applicatie zelf. Hierna zullen er zoveel mogelijk onderdelen van de applicatie gebouwd worden, zodat er bij de volgende demo (einde van sprint 4) zoveel mogelijk getoond kan worden aan de opdrachtgever voor meer/betere feedback.

6.4 HOLOBUILDING SERVER

Tijdens het testen met de modellen is er gebleken dat de HoloLens geen filesystem heeft, waardoor de te gebruiken modellen niet geplaatst kunnen worden op de HoloLens. Hierom zal er een server, zie Server in figuur 6.1, opgezet worden om te demonstreren hoe de applicatie zou kunnen werken als het gerealiseerd zou worden. Aangezien het hier alleen gaat om de functionaliteit van de echte HoloLens applicatie te ondersteunen is er alleen rekening gehouden met de functionaliteit van de server en niet op andere aspecten als security en schaalbaarheid. De gehele klassendiagram van deze server is hieronder te zien in figuur 6.13 en in het verloop van dit hoofdstuk zal deze architectuur toegelicht worden.

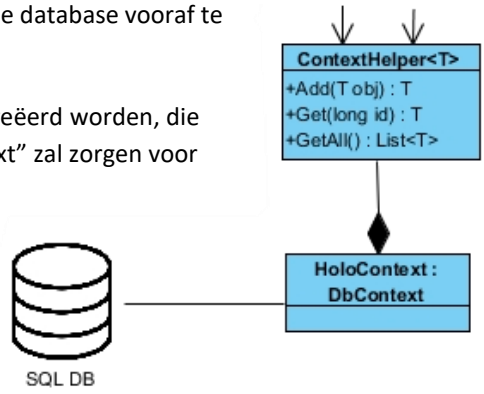


Figuur 6.13: Volledige Server Klassendiagram

6.4.1 DATABASE CONTEXT

Voor een eenvoudige implementatie zal er gebruik gemaakt worden van een ORM genaamd Entity Framework. Hierbij zal gewerkt worden met een Code-First methodiek, waardoor de database automatisch aangepast wordt, gebaseerd op de aangegeven modellen die geschreven zijn in code. Hierdoor zal de database gecreëerd worden naar de wensen van de applicatie en is het dus niet nodig de database vooraf te bouwen en vervolgens identieke modellen in code te definiëren.

Om een generieke implementatie te behouden zal er een class gecreëerd worden, die ook te zien is in figuur 6.14, genaamd HoloContext. Deze “DbContext” zal zorgen voor alle activiteiten die te maken hebben met de Database, dit kunnen activiteiten als toevoegen, ophalen en wijzigen van data zijn. Om deze activiteiten in code te kunnen uitvoeren zal er een “ContextHelper” gemaakt worden, die tijdens het gebruik een class zal verwachten. Door deze class mee te geven is deze “ContextHelper” generiek, waardoor later meerdere modellen toegevoegd kunnen worden, zonder de helper class aan te passen.



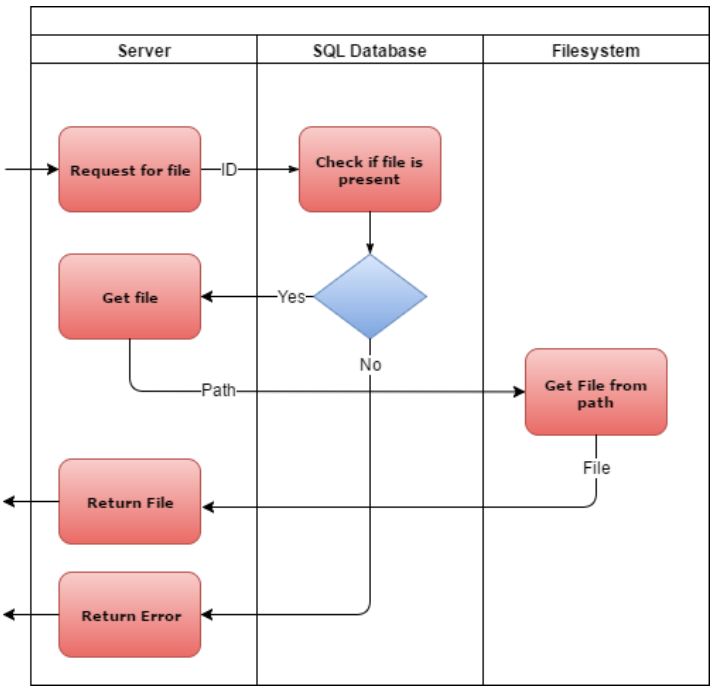
Figuur 6.14: Database Context

6.4.2 OPSLAG

Voor opslag is ervoor gekozen om twee opslagmethoden te gebruiken, namelijk een filesystem en een database. De filesystem, aangegeven als “HDD” in figuur 6.13, dient als opslag voor de 3D modellen. Deze modellen kunnen via een Webapplicatie, paragraaf 6.4.3, geüpload worden en zullen vervolgens op een structurele wijze op deze filesystem opgeslagen worden.

Om de modellen bij te houden die opgeslagen zijn op de filesystem, zal er gebruik gemaakt worden van een Database. Deze database zal gecreëerd en bijgehouden worden door de ORM die uitgelegd is in het vorige paragraaf. Het doel van deze database is dan ook de beschikbare modellen bij te houden, door middel van een naam, unieke code en pad naar het bestand dat bij het model hoort.

Door deze opslagstructuur aan te houden (figuur 6.15), kan de server makkelijk bestanden opvragen aan de database door middel van een ID. Als het model bestaat, zal de server alle informatie over het model terugsturen, zoals de naam en pad. Door middel van deze pad kan de server vervolgens het model in de filesystem vinden en deze model terugsturen. Als de database het model niet kan vinden zal er een error terug gestuurd worden, waardoor het duidelijk wordt voor de applicatie dat de opgevraagde model niet bestaat.



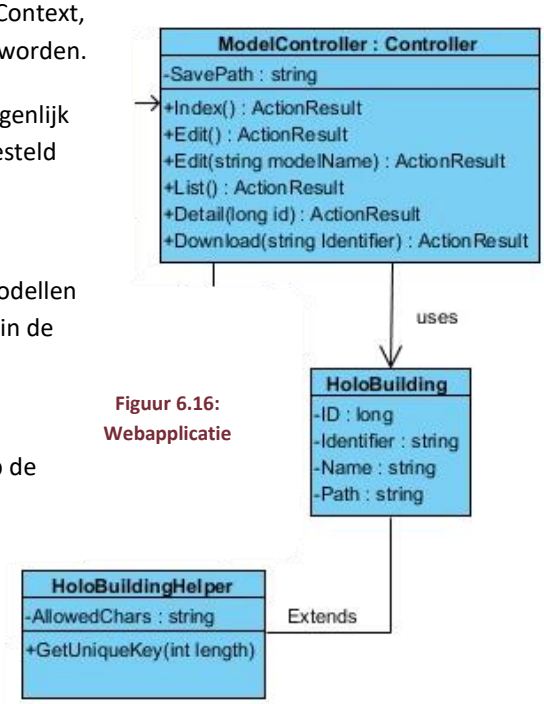
Figuur 6.15: Flow van het ophalen van de modellen

6.4.3 WEBAPPLICATIE (MVC)

Om de gebruiker van de HoloLens applicatie in staat te stellen modellen daadwerkelijk te uploaden naar de server zal er een Webapplicatie gebouwd worden. Hierbij zal gebruik gemaakt worden van het MVC framework. Dit is een framework waarmee er op een eenvoudige wijze een applicatie te gebouwd kan worden en aangezien de modellen al gemaakt zijn voor de Database Context, hoeft er alleen maar een view en een controller gebouwd te worden.

De view is de pagina die de gebruiker te zien krijgt en dit is eigenlijk niets meer dan een HTML-pagina, waarin de webpagina opgesteld staat.

De controller daarentegen is de logica achter de view. In de controller worden functies als het opslaan en ophalen van modellen gedefinieerd. Deze functies staan gedefinieerd in figuur 6.16 in de ModelController class. De belangrijkste twee functies zijn de Edit(string modelName) en Download(string identifier).



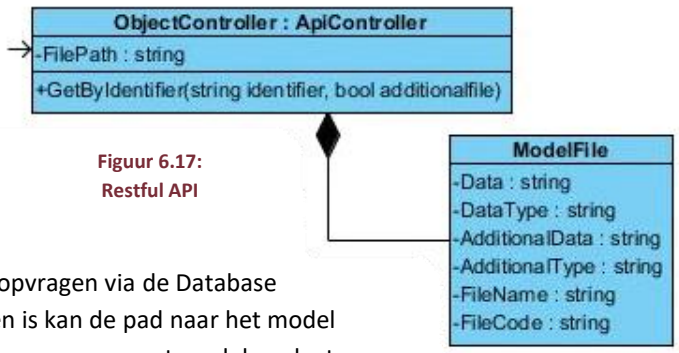
Figuur 6.16: Webapplicatie

De eerste functie zorgt ervoor dat de geüploade modellen op de juiste wijze worden opgeslagen. Dus als eerst wordt er een unieke identifier gecreëerd door middel van de HoloBuildingHelper class en met deze identifier worden de modellen opgeslagen op de filesystem. Vervolgens worden de identifiers, uploadnaam van het bestand en de pad naar het bestand in de database opgeslagen, zodat het model later makkelijk terug te vinden is.

De tweede functie zorgt ervoor die het model die geüpload is weer te downloaden is. Deze functie verwacht een identifier, waarmee de pad naar het bestand makkelijk opgevraagd kan worden.

6.4.4 RESTFUL API

De laatste en belangrijkste onderdeel van de HoloBuilding Server is de Restful API. Dit onderdeel stelt de HoloLens applicatie instaat een model op te vragen van de server. Deze API heeft eigenlijk maar een functie en dat is door middel van de meegestuurde identifier een model op te halen en vervolgens op te sturen.



Figuur 6.17: Restful API

Zoals in paragraaf 6.4.2 beschreven staat zal het opvragen via de Database Context gaan en als het gezochte model gevonden is kan de pad naar het model opgevraagd worden. Hierbij zullen de modellen in een aangepast model geplaats worden, namelijk het “ModelFile” model (zie figuur 6.17). Dit model zal de data/model zelf, datatype, aanvullende model/data, aanvullende datatype, modelnaam en modelcode bevatten. Het model en aanvullende model zullen als base64 string het ModelFile opgenomen worden, waarna het gehele model als een JSON-bestand opgestuurd kan worden naar de HoloLens. Hierbij kan de HoloLens het model makkelijk verwerken en uitlezen. Het aanvullende data en type die hier beschreven zijn, zijn tevens optioneel en zijn een toevoeging na het vervolg van paragraaf 4.3.2, waarbij het bleek dat de texturen van de OBJ-modellen in een ander bestand worden opgeslagen. Indien dit het geval is met andere modeltypes die later toegevoegd zullen worden, kan er gebruik gemaakt worden van deze twee extra velden.

7 ONTWIKKELING

Nadat de architectuur afgerond was, is er begonnen aan de bouw van de server en applicatie zelf. Dit was in sprint 4 en in dit hoofdstuk zal de ontwikkeling van de applicaties en bijkomende systemen uitgewerkt worden. Hierbij zal er volgens de architectuur in hoofdstuk 6 ontwikkeld worden en zullen afwijkingen toegelicht worden. Ook zullen overige problemen die tijdens de ontwikkeling naar voren zijn gekomen toegelicht worden.

7.1 SERVER

Bij het bouwen van de server is er eerst gekeken wat de server moet uitvoeren en volgens de architectuur, moet er een mogelijkheid zijn voor de gebruiker om modellen te uploaden naar een server. Verder moeten de geüploade modellen via een http request op te halen zijn, vanaf de HoloLens.

7.1.1 OPSLAG EN DATABASE

Om een eenvoudig pad aan te houden is er voor de filesystem een simpel pad gebruikt, namelijk direct op de “C:” schijf. Op deze “C:” locatie is een map gemaakt, waarin de modellen opgeslagen kunnen worden en deze map zal in het vervolg gebruikt worden om alle model bestanden op te slaan.

Voor de database is het HoloBuilding model aangemaakt en er is vervolgens door middel van de code-first implementatie van het Entity Framework een database aangemaakt, waarin ook gelijk een tabel gemaakt is waar de HoloBuilding modellen opgeslagen kunnen worden. Voor deze implementatie is gebruik gemaakt van de documentatie op de Microsoft-website (EF Code-First, 2016).

Verder is ook de generieke ContextHelper class aangemaakt, waarin de modellen die in de database staan meegegeven kunnen worden, zoals het HoloBuilding model. Door middel van deze class kan er op een eenvoudige manier rij(en) toegevoegd en opgehaald worden vanuit de database. Deze acties zullen via de HoloContext gaan, die de acties op de gemaakte database uitvoert.

7.1.2 WEBAPPLICATIE

Om de opslag en database te vullen moet de gebruiker de mogelijkheid krijgen modellen te uploaden. Hiervoor is een simpel formulier gemaakt in HTML, waarin gebruikers modellen en aanvullende bestanden kunnen selecteren. Nadat een gebruiker een formulier heeft ingevuld en opgestuurd zal de ModelController op de server een unieke ID aanmaken voor het model met behulp van de HoloBuildingHelper. Deze ID zal gebruikt worden om een map aan te maken in de map dat in paragraaf 7.1.1 is aangemaakt op de “C:” schijf. Deze nieuwe map zal als naam de ID krijgen en de modellen die geüpload zijn zullen in deze map opgeslagen worden. Nadat de modellen opgeslagen zijn, zal er een nieuwe HoloBuilding model aangemaakt worden, waarin de naam van het model, ID en pad opgenomen zullen worden. Het HoloBuilding model zal vervolgens met behulp van de ContextHelper opgeslagen worden in de Database.

Nadat een model is opgeslagen zal de gebruiker doorgestuurd worden naar de pagina dat aangemaakt is voor het model. Deze pagina heeft informatie, zoals de modelnaam en de ID die gegenereerd is voor het model. Ook heeft de gebruiker hier de optie het model te downloaden.

7.1.3 RESTFUL API

Als de HoloLens een model ophaalt kan een request gestuurd worden naar de Restful API, namelijk de ObjectController. Deze controller zal via de ContextHelper aan de database vragen of het model bestaat met de aangeven ID. Als dit het geval is zal een ModelFile aangemaakt en opgestuurd worden, waarbij het aanmaken van het model volgens paragraaf 6.4.4 gaat.

7.2 OPBOUW VAN DE APPLICATIE IN UNITY

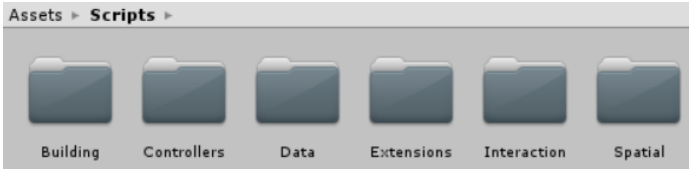
Om de structuur van het klassendiagram terug te vinden in de applicatie zijn alle klassen gecreëerd en opgenomen volgens de structuur van het klassendiagram. Aangezien dit in Unity gedaan is, ziet dit er anders uit dan het in een standaard applicatie eruit zou zien. Om dit beeld te verduidelijken zal er toegelicht worden hoe de structuur in Unity overeenkomt met het klassendiagram.

Zoals in paragraaf 5.1 al toegelicht is, kent Unity een aantal verschillende schermen waarin het project beheerd kan worden. Om de structuur in het klassendiagram aan te kunnen houden zijn er twee schermen waar rekening mee gehouden moeten worden, namelijk het “Project” en “Hierarchy” scherm. Het project scherm zal alle objecten/Assets bevatten die in het project gebruikt zullen worden. Dit zijn objecten als modellen, scripts, materialen, texturen, etc.

Figuur 7.1: Assets Structuur

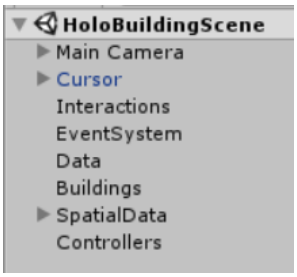
Een groot deel van deze objecten zijn niet opgenomen in het klassendiagram, omdat dit ondersteunende objecten zijn die aangestuurd zullen worden door de scripts. De scripts zijn de code objecten van het project en dit is ook waar de structuur van het klassendiagram toegepast zal worden. De andere object soorten zijn in andere folders naast de scripts geplaatst, zodat ook deze objecten een goed structuur blijven houden (zie figuur 7.1).

De packages in het klassendiagram zullen als folders opgenomen worden in de scripts folder, zie figuur 7.2. Vervolgens zullen de folders die de packages representeren scripts bevatten die de klassen binnen de packages zijn. Hierdoor is de architectuur van het klassendiagram terug te vinden in het “Project” scherm en zal het makkelijker zijn de klassen terug te vinden in de verdere ontwikkeling.



Figuur 7.2: Scripts Structuur

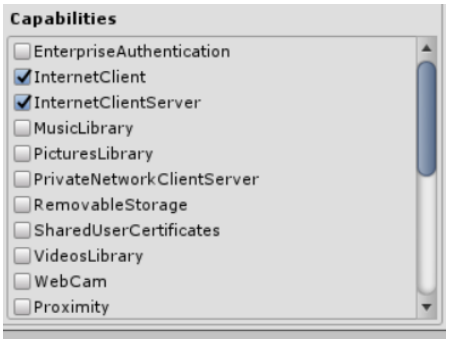
Naast het “Project” scherm is er ook een “Hierarchy” scherm waar de structuur van het klassendiagram terug gevonden moet worden. In dit scherm is het echter niet mogelijk folders te maken, maar alleen GameObjecten. Dit zijn Unity specifieke objecten waar verschillende attributen aan gegeven kunnen worden. Deze GameObjecten zullen in dit geval de packages in het “Hierarchy” scherm voorstellen. Door de packages als aparte GameObjecten op te nemen zullen de verschillende taken die behoren bij de packages geïsoleerd uitgevoerd worden en zal de structuur dus ook terug gevonden kunnen worden. Deze “Hierarchy panel is terug te vinden in figuur 7.3. Hierbij zijn ook een aantal andere GameObjecten te zien die niet in het klassendiagram opgenomen zijn. Deze zullen in de volgende paragrafen toegelicht worden.



Figuur 7.3: Hierarchy structuur

7.3 DATA PACKAGE

Nadat de structuur aangelegd was en de verschillende scripts/klassen waren toegevoegd moest de functionaliteit gebouwd worden. Hierbij is er begonnen met het dataonderdeel, omdat dit ook het eerste onderdeel is dat gebruikt zal worden in de applicatie.



Figuur 7.4: Unity Toestemmingen

In hoofdstuk 6 is aangegeven dat de HoloLens geen filesystem heeft en dat de modellen dus van een webserver gehaald moeten worden. Om de applicatie de toegankelijkheid te geven een connectie naar buiten te maken moeten er in de instellingen van de Unity omgeving een aantal wijzigingen aangebracht worden. De wijzigingen die aangebracht moeten worden vallen onder het scherm dat te vinden is op Edit > Project Settings > Player. Hierbinnen is een gedeelte genaamd “Capabilities” te vinden. Hier kunnen de toestemmingen, die de applicatie nodig kan hebben, aan en uit gezet worden, zie figuur 7.4. De instellingen die hier aangezet moeten worden zijn: InternetClient, InternetClientServer, Microphone en SpatialPerception. Alleen de eerste twee opties zijn nodig om een verbinding met buiten te maken, maar de andere toestemmingen zullen later nodig zijn, dus die zijn ook gelijk aangezet. De doelen van deze toestemmingen zullen ook bij het passende hoofdstuk toegelicht worden.

Nadat deze toestemmingen aangezet waren, kon de HoloLens een connectie naar buiten maken en is er begonnen aan de echte functionaliteit. De eerste controle die volgens het activiteitendiagram (figuur 6.2) uitgevoerd wordt is de controle op beschikbare modellen. Hiervoor moet eerst een locatie naar het model vastgelegd worden. De UnityEngine kent een aantal verschillende paden die vaak gebruikt worden in development, maar enkel één van deze paden is op een locatie waar de applicatie zonder problemen data kan opslaan. Deze pad is te bereiken onder de naam “persistantDataPath”. Dit is een locatie waar een aantal andere bestanden aanwezig zijn van de applicatie. Hierom zal er op de verkregen pad een map aangemaakt worden waar alle bestanden gerelateerd aan de modellen opgeslagen kunnen worden. Door deze bestanden in een map op te slaan zal het later makkelijker worden deze bestanden te beheren.

De volgende stap is het verkrijgen van data vanaf de server. Aangezien het mogelijk is dat een model bestaat uit verschillende bestanden moest het verkrijgen van data op een efficiënte manier uitgevoerd worden. Hierom is er een aangepast object gemaakt, gelijk aan de ModelFile in paragraaf 6.4.4, waarin verschillende data opgeslagen kan worden zoals model- en aanvullende bestanden, modelnaam en types. Dit laatste is om het mogelijk te maken later verschillende soorten modellen toe te voegen. De bestanden zullen als base64 strings verkregen worden, aangezien het bestanden zijn die verstuurd moeten worden via http.

Door het gebruik van dit aangepaste model zal alle benodigde data in een keer opgehaald kunnen worden, waardoor de applicatie na een antwoord direct door kan met het volgende proces. Deze aanvraag zal uitgevoerd worden met behulp van de UnityWebRequest methode (UnityWebRequest, 2017). Dit is een onderdeel van de UnityEngine en heeft als functie het opstellen van http Request.

Nadat een model is opgehaald kan deze door middel van het JsonConvert methode van NewtonSoft terug omgezet worden naar een daadwerkelijk model. Dit model bevat namelijk een base64 string die het model- en aanvullende bestand voorstellen. Deze kunnen vervolgens geconverteerd worden naar een daadwerkelijk bestand op de persistantDataPath locatie, deze conversie is te zien in Code sample 07.

7.4 INTERACTION PACKAGE

Bij het ophalen van het model vanaf de server moet de gebruiker een code invoeren. Het invoeren van deze code zal via de Interaction Package gaan, net zoals alle andere activiteiten waarbij de gebruiker acties kan ondernemen. In dit paragraaf zullen alle componenten beschreven worden die gebruikt zijn qua interactie met de gebruiker.

Een groot deel van de interacties kan verkregen worden van de HoloToolkit (HoloToolkit-Unity, 2017) dat beschikbaar is voor ontwikkelaars die bezig zijn met de HoloLens. Deze toolkit bevat alle interactie technieken die de HoloLens ondersteund en ze kunnen dan ook eenvoudig geïmplementeerd worden. De implementatie van deze technieken zal niet uitgelegd worden, maar wel wat het doel is van een techniek en ook hoe de technieken gecombineerd zijn om de gebruiksvriendelijkheid te verbeteren.

Naast de interactie technieken die de toolkit aanbiedt zijn er ook een aantal eigen componenten gemaakt en ook zijn geïmplementeerde componenten aangepast, zodat ze passender waren binnen het project. Ook al deze toevoegingen en aanpassingen zullen in dit paragraaf beschreven worden.

7.4.1 INTERACTIE TECHNIEKEN

KEYWORDMANAGER

De KeywordManager is het onderdeel binnen de Interaction Package die het mogelijk maakt voor de gebruiker commando's uit te voeren door middel van spraak. Deze commando's kunnen door middel van een lijst doorgegeven worden aan de KeywordManager, waarin de spraakcommando en de actie opgenomen zijn. Als deze lijst wordt doorgegeven aan de KeywordManager, wordt de lijst uitgelezen en gekoppeld aan een KeywordRecognizer (KeywordRecognizer, 2017). Deze KeywordRecognizer luistert naar de gebruiker gedurende de applicatie actief is. Om de gebruiker te kunnen verstaan heeft de KeywordRecognizer toestemming nodig om de microfoon te gebruiken, die reeds is toegewezen in paragraaf 7.3.

Afhankelijk van de staat van de applicatie kunnen de spraakcommando's aangepast worden. Zodra de HoloLens een commando verwacht van de gebruiker, kan de gebruiker de commando's ook met spraak kunnen uitvoeren. Dit is echter een deel van de beschikbare commando's die uitgevoerd kunnen worden, want er kunnen ook een aantal generieke commando's beschikbaar zijn die de gebruiker altijd of na een mijlpunt van het proces moet kunnen gebruiken. Hierdoor zijn er twee verschillende lijsten met commando's die gedurende de applicatie blijven veranderen en de huidige opzet van de commando's ondersteund geen dynamische lijst met commando's. Dit is dan ook de eerste aanpassing die gemaakt is, zodat de KeywordManager beter voldoet aan de wensen van het project.

De aanpassing is namelijk dat er twee verschillende lijsten beschikbaar zijn in de KeywordManager. De eerste is voor de generieke commando's en de tweede lijst is voor commando's die de HoloLens tijdens de applicatie kan verwachten van de gebruiker. Zodra een van deze lijsten wordt aangepast kan een methode aangeroepen worden, waaraan een nieuwe lijst doorgegeven kan worden. Deze methode vervangt de lijst die in de KeywordManager opgenomen is en verwijderd de eerder aangemaakte KeywordRecognizer. Vervolgens worden de twee verschillende lijsten gecombineerd in een lijst en wordt er een nieuwe KeywordRecognizer aangemaakt.

TEXTTOSPEECHMANAGER

De TextToSpeechManager is het onderdeel waarmee de HoloLens kan praten tegen de gebruiker. Hierbij kan er door middel van een string (een variabele dat tekst bevat) een uitspraak doorgegeven worden aan de HoloLens, waarna deze door de HoloLens uitgesproken wordt.

TAGALONG

Als een object in de wereld geplaatst wordt kan deze met de gebruiker mee bewegen, dit is de taak van de TagAlong. Als een object een TagAlong toegewezen krijgt blijft het object in de buurt van de gebruiker. De positie van het object zal niet constant aangepast worden, maar zodra het object uit het gezichtsveld van de gebruiker is zal de positie aangepast worden, waardoor de object weer te zien zal zijn.

BILLBOARD

In tegenstelling tot de TagAlong zorgt de Billboard er niet voor dat een object met de gebruiker mee beweegt maar dat het object altijd in de richting van de gebruiker kijkt. Hierdoor kan de gebruiker om het object heen lopen en zal de zijde dat geselecteerd is altijd te zien zijn door de gebruiker. Deze techniek is vooral handig bij een userinterfaces.

CURSOR

De cursor is het onderdeel dat ervoor zorgt dat de gebruiker weet waar de HoloLens op gefocust is. Om dit werkend te krijgen zijn er wel een aantal verschillende klassen nodig uit de HoloToolkit, zoals de CursorManager, GazeManager, GazeStabilizer en GestureManager.

CURSORMANAGER

Om de cursor, die de gebruiker te zien krijgt, te kunnen beheren is er een manager nodig, namelijk de CursorManager. Deze manager zorgt ervoor dat de positie en rotatie van de cursor aangepast wordt als de gebruiker van positie veranderd of een andere kant op kijkt.

De cursor kent ook twee verschillende visuele vormen, namelijk één voor als er niet gekeken wordt naar een hologram en één voor als er wel gekeken wordt naar een hologram. Dit wordt beslist door de manager door aan de GazeManager op te vragen of de gebruiker kijkt naar een hologram. Gebaseerd op de situatie zorgt de CursorManager ervoor dat de cursor de juiste visuele vorm aanneemt.

GAZEMANAGER EN GAZESTABILIZER

Het berekenen van de kijkrichting en of de gebruiker kijkt naar een hologram wordt berekend in de GazeManager, waarbij gebruik gemaakt wordt van een Raycast. Deze Raycast is ook onderdeel van de UnityEngine en wat het doet is berekenen of er een object aanwezig is in de kijkrichting van de gebruiker.

Omdat de HoloLens zich op de hoofd van de gebruiker bevindt, zullen de waardes die de kijkrichting van de gebruiker voorstellen constant veranderen. Dit komt omdat er ook hier gebruik gemaakt wordt van floats en deze floats zijn, zoals in paragraaf 4.4.3 beschreven is, heel nauwkeurig. Hierdoor zou de kleinste beweging resulteren in het veranderen van de locatie van de cursor. Daarom is er een GazeStabilizer, die ervoor zorgt dat een verandering onder een bepaalde waarde genegeerd wordt. Hierdoor wordt de positie van de cursor alleen veranderd als dit ook echt de bedoeling is van de gebruiker.

GESTUREMANAGER

Op dit moment weet de gebruiker waar de cursor op gefocust is en of er dus een commando uitgevoerd kan worden. Hiervoor heeft de HoloLens een commando wat de gebruiker met zijn eigen handen kan uitvoeren, namelijk een air-tap. Deze air-tap is uit te voeren door een vinger op te tillen voor de HoloLens en vervolgens te “tappen”. Dit herkent de HoloLens als een klik event, zoals een computer een muisklik herkent. Deze klik event wordt opgevangen in de GestureManager en hierop kan dus een actie afgevuurd worden.

SPRINT OVERGANG

De interactie technieken die beschikbaar waren voor de HoloLens bleken er meer te zijn dan verwacht en tegen de tijd dat de beschikbare technieken toegevoegd waren aan het project was het einde van sprint 4 al bereikt.

SPRINT 4

In sprint 4 is de overige architectuur afgerond en gebaseerd op deze architectuur is er een server gemaakt. Om deze server te kunnen testen is er ook een begin gemaakt aan de HoloLens applicatie, waarbij als eerst de Data Package is geïmplementeerd. Hierdoor kon de server getest worden en getoond worden tijdens de demo.

Om de ID van modellen te kunnen invoeren met de HoloLens was er ook een begin gemaakt aan de Interaction Package, waarbij eerst alle beschikbare interactie technieken geïmplementeerd zouden worden. Tegen de tijd dat alle verschillende interactie technieken geïmplementeerd waren, was het einde van sprint 4 bereikt en moest er gepresenteerd worden. Aangezien de interactie technieken alleen geïmplementeerd waren en nog niet gebruikt werden in de applicatie zelf, kon er tijdens de demo niet veel getoond worden over de Interacties.

Hierom is er alleen een demo gegeven van de server en hoe de server werkt met de HoloLens. Hierbij is aangegeven dat de request nog handmatig gaat en dat in de volgende sprint de interactie volledig operationeel zou moeten zijn. Volgens de opdrachtgever was het project op het goede pad en kon de ontwikkeling dus doorgaan.

SPRINT 5

In de volgende sprint zou de interactie afgerond worden, waarbij de gebruiker de mogelijkheid zou krijgen een ID door te geven en zou er visuele feedback toegevoegd worden, waardoor de gebruiker kon zien wat er gedaan moest worden. Ook zou er een begin gemaakt worden aan de overige onderdelen van de applicatie.

7.4.2 INTERACTIONMANAGER

Tijdens het zoeken naar een toetsenbord voor gebruik binnen de HoloLens zijn er geen gevonden die makkelijk te implementeren waren binnen Unity en ook functioneel waren voor gebruik met de HoloLens. Daarom is er een eigen toetsenbord gemaakt en is er ook een User Interface gemaakt die de gebruiker kan zien. Hiernaast zijn ook de technieken die in paragraaf 7.4.1 geïmplementeerd waren in gebruik gezet, waardoor de interactie met de gebruiker op een eenvoudige wijze afgehandeld kon worden.

HOLOKEYBOARD

Bij het maken van het toetsenbord voor de HoloLens is er gebruik gemaakt van de UI-elementen die Unity aanbiedt. Hiermee is een paneel gemaakt met twee onderdelen, namelijk een output en een input. De output is het bovenste gedeelte van het toetsenbord en hier komt de ingevoerde tekst tevoorschijn. De tekst zal door de knoppen in de input gedeelte ingevoerd worden, waarbij er een toetsenbord volgens het “Qwerty” lay-out ingedeeld is. In figuur 7.5 is te zien hoe de uiteindelijke toetsenbord er voor de gebruiker uitziet.

De toetsenbord leek nu op een toetsenbord, maar werkte niet als een toetsenbord. Hiervoor moesten een aantal scripts geschreven worden, die ervoor zouden zorgen dat de knoppen ook daadwerkelijk effect hadden op de tekst-uitvoer. Hiervoor is er een script geschreven die gebonden zijn aan de knoppen op het toetsenbord. Dit is één script, genaamd “KeyBehaviour”, die controleert wat voor toets is ingedrukt en hierop gebaseerd acties onderneemt. Bij de getallen en letters zal het karakter naar het tekst-uitvoer gestuurd worden, bij het pijltje naar links(backslash) zal het laatst ingevoerde karakter verwijderd worden en bij de “Done” knop zal de gebonden actie uitgevoerd worden.

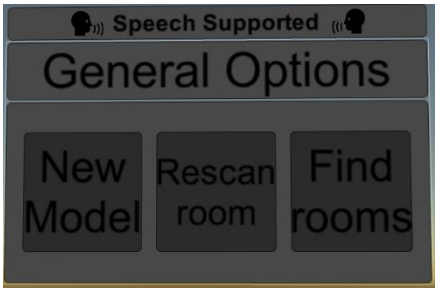
Figuur 7.5: HoloKeyboard

Al deze acties worden niet direct vanaf de KeyBehaviour script uitgevoerd, maar worden door gestuurd naar het hoofdscript, namelijk “KeyboardManager”. Deze manager houdt de ingevoerde tekst bij en update te tekst in de tekst-uitvoer. Bij het aanmaken van het toetsenbord wordt er ook een actie doorgegeven aan de KeyboardManager, deze actie wordt uitgevoerd als de gebruiker op de “Done” knop drukt en zorgt ervoor dat de aanvrager van de code niet hoeft te wachten tot de gebruiker een ID heeft ingevoerd, maar dat het toetsenbord de volgende actie initieert als de gebruiker klaar is met het invoeren.

Op deze toetsenbord is ook de TagAlong en Billboard techniek toegepast, waardoor het toetsenbord het kijkveld van de gebruiker blijft volgen en het toetsenbord altijd te zien is door de gebruiker, ongetwijfeld de locatie van de gebruiker. Deze hele toetsenbord pakket is vervolgens opgeslagen als een “Preset”, zodat het later makkelijk gebruikt kan worden in de applicatie en heeft het de naam HoloKeyboard gekregen.

HOLOUI

Naast het toetsenbord is er ook een userinterface gemaakt met de UI-elementen die Unity aanbiedt. De userinterface is net zoals het toetsenbord in verschillende onderdelen opgesplitst (zie figuur 7.6), namelijk het speech-, question- en optionspaneel. Het speechpanel is een statisch paneel, waarin wordt aangegeven dat de acties die in het paneel aangeven staan ook uitgevoerd kunnen worden met een spraakcommando, waarvan de implementatie in het volgende paragraaf toegelicht zal worden. De question paneel stelt de vraag aan de gebruiker waarop een antwoord verwacht wordt, die beschikbaar zijn in het optionspaneel.



Figuur 7.6: HoloUI

Om deze userinterface operationeel te maken is er een HoloUIManager script geschreven. Deze script zit vast aan de HoloUI en dit is het aanspreekpunt van de HoloUI. Bij het aanmaken van een HoloUI verwacht de manager een aantal waardes, zoals de vraag(Question), de opties en of het een HoloUI is dat constant bijgewerkt moet worden. De vraag zal geplaatst worden in de question paneel en geeft aan wat voor antwoord de HoloUI verwacht. De opties zijn de antwoorden die de gebruiker kan geven. Deze opties worden verwacht in een lijst vorm, waarbij er per optie een antwoord formulering is opgenomen en ook de actie die uitgevoerd moet worden. Als laatst kan het zijn dat de userinterface is dat langer moet blijven bestaan en dat de antwoorden die de gebruiker kan geven wel constant moeten veranderen.

Door middel van deze waardes kan de manager ervoor zorgen dat de interface om de aantal seconden bijgewerkt wordt of eenmaal wordt uitgevoerd. In beide gevallen wordt dezelfde methode uitgevoerd, met het verschil dat bij het bijwerken dezelfde methode vaker wordt aangeroepen.

De methode die aangeroepen wordt zorgt ervoor dat er knoppen verschijnen in de HoloUI die gebaseerd zijn op de lijst met opties. Dit gebeurd door een knop object aan te maken en deze object te plaatsen in de options paneel. Deze knoppen krijgen verder als waarde de antwoord formulering en de actie wordt gebonden aan de knop door middel van een ander script die speciaal voor de knoppen geschreven is, namelijk de HoloUIButton script. Deze script voert de actie uit die bij het aanmaken gebonden is aan het knop.

INTERACTIE SAMENHANG

Om al deze verschillende interacties beschikbaar te stellen voor de applicatie en om bepaalde technieken samen te laten werken is er als laatst een InteractionManager gemaakt. Deze manager is het centrale punt van de Interaction Package en het is de manager die aangesproken zal worden vanuit verschillende onderdelen van de applicatie. Doormiddel van deze verschillende onderdelen in de InteractionManager kunnen alle acties waarbij de gebruiker betrokken is op een eenvoudige wijze uitgevoerd worden.

Het eerste wat in deze manager gebouwd is, is de mogelijkheid om de HoloLens een vraag uit te laten spreken. Hiervoor is gebruik gemaakt van de TextToSpeechManager. Deze manager heeft een methode die ervoor zorgt dat vanuit de HoloLens een vraag gesteld kan worden. Om deze methode makkelijk te kunnen benaderen is er dus een Announce functie geschreven in de InteractionManager. Snel werd al opgemerkt dat het vaker aanroepen van de functie ervoor zorgt dat de vorige wordt geannuleerd. Hierdoor wordt de eerste uitspraak halverwege gestopt om vervolgens de volgende uitspraak te doen.

Om dit probleem op te lossen is er een Stack gemaakt, wat een lijst is met een first in first out structuur, waarin alle teksten die uitgesproken worden geplaatst worden. Vervolgens is er in de Update-methode, een methode die elke frame door Unity wordt aangeroepen, een check geplaatst waarin wordt gecontroleerd of de TextToSpeechManager aan het praten is. Als dit niet het geval is wordt er gekeken of er nog teksten zijn die uitgesproken moeten worden en indien deze teksten aanwezig zijn worden ze uitgesproken. Als de TextToSpeechManager nog bezig is met een uitspraak wordt er geen nieuwe uitspraak gedaan en wordt er bij de volgende uitvoering van de Update-methode nogmaals gecontroleerd of het mogelijk is de volgende uitspraak te doen.

Om de HoloKeyboard te kunnen gebruiken is er ook een methode geschreven in de InteractionManager. Deze methode start de HoloKeyboard op en zorgt ervoor dat als er op de “Done” knop gedrukt is de ingevoerde tekstwaarde in een tijdelijke variabele in de InteractionManager geplaatst worden. Vervolgens wordt de methode aangeroepen die de waarde nodig heeft. Deze methode kan vervolgens met de GetUserInput methode de ingevulde tekst opvragen.

Het derde en laatste onderdeel dat gebouwd is, is de Questioning methode. Deze methode wordt aangeroepen als een onderdeel van de applicatie een direct antwoord nodig heeft vanuit de gebruiker. Deze Questioning methode zorgt er namelijk voor dat er een HoloUI tevoorschijn komt waarop de vraag gesteld wordt en antwoordmogelijkheden te zien zijn. Naast deze HoloUI zorgt de Questioning methode er ook voor dat de KeywordManager de nieuwe lijst met spraakcommando’s krijgt, waardoor de gebruiker de acties ook met spraak kan initiëren. In het geval dat een actie ervoor moet zorgen dat de HoloUI weer verdwijnt moet ook de StopQuestioning methode in de actie aangeroepen worden. Deze methode zorgt ervoor dat de spraakcommando’s en HoloUI op de juiste manier hersteld en verwijderd worden.

7.5 BUILDING FACTORY

Nadat het model opgehaald en klaar is voor gebruik moet het model ingeladen worden. Hier moet alleen rekening gehouden worden met modeltypes die in een later stadium toegevoegd kunnen worden. Om dit te vereenvoudigen in de toekomst is er gebruik gemaakt van een interface, namelijk de IBuilding interface. In deze interface zijn een aantal verschillende methoden opgenomen, die gebruikt zullen worden om de processen van de Building Factory vast te stellen. Deze methoden zullen ervoor zorgen dat nieuwe modeltypes makkelijk toegevoegd kunnen worden.

Onder de IBuilding interface zullen andere klassen geplaatst worden die specifiek per model soort opgesteld zullen worden. Deze klassen zullen de methoden die in de interface opgesteld staan, zelf moeten definiëren met de eigen afhandelingsmanier. Hierdoor kan een methode aangeroepen worden en gebaseerd op de type model kan de methode specifiek voor dat model uitgevoerd worden. In de huidige aanpak is er alleen een ObjBuilding klasse geplaatst onder de interface, aangezien alleen OBJ-modellen worden ondersteund.

Onder de klassen die geplaatst zijn (onder de interface), kunnen ook andere klassen toegevoegd worden die nodig zijn om de taken voor de desbetreffende model soort uit te kunnen voeren. Een voorbeeld hiervan is de ObjImporter (zie paragraaf 4.3.2) die geplaatst is onder de ObjBuilding klasse.

Door de ObjImporter onder de ObjBuilding te plaatsten kunnen de specifieke methodes in ObjImporter verbonden worden met de methoden in ObjBuilding (die komen vanuit de interface). Hierdoor hoeft er bij het inladen van een OBJ-model niet uitgezocht te worden welke methode aangeroepen moet worden, aangezien het één specifieke methode is die opgesteld staat in de interface.

Naast de klassen die voor het inladen van het model zorgen, is er ook een BuildingHelper toegevoegd aan de Building Package. Dit is een klasse die aangesproken wordt als er een actie, dat betrekking heeft tot de ingeladen model, uitgevoerd moet worden. De eerste taak die deze BuildingHelper uitvoert is het instantiëren van het BuildingController object, wat van het type IBuilding is. Gebaseerd op het type model dat verkregen is vanaf de server, wordt de juiste klasse geïnstantieerd in de BuildingController object. Door middel van deze object worden de acties die betrekking hebben tot de ingeladen modellen op de juiste manier afgehandeld.

De eerste actie die met deze BuildingController uitgevoerd wordt is het inladen van het model. De locatie van het model wordt doorgeven aan de controller, die er vervolgens ervoor zorgt dat het in de juiste subklasse van de interface terecht komt. Naast het inladen van het model worden ook andere acties die betrekking hebben tot het model uitgevoerd met behulp van de BuildingController, zoals het roteren en positioneren.

Naast de acties die uitgevoerd worden via de BuildingController worden ook een aantal andere acties in de BuildingHelper zelf uitgevoerd. Een van deze taken is het vergelijken van kamers binnen het model met de verkregen dimensies. Hierbij wordt er een waarde met dimensies en de tolerantie die toegestaan is verwacht. Door middel van deze waardes wordt er in het ingeladen model gezocht naar kamers die passen bij de opgegeven waardes, waarna de gevonden kamers teruggestuurd worden.

Door middel van de beschreven acties kan zullen de verschillende model types, die in de toekomst toegevoegd kunnen worden, op een eenvoudige manier ingeladen en gemanipuleerd kunnen worden. Dit zal allemaal achter de schermen gebeuren, waarbij de verschillende processen op één enkele manier in de rest van de applicatie uitgevoerd zullen worden.

7.6 SPATIAL PACKAGE

De Spatial Package is het onderdeel dat voor een groot deel geleverd is door Microsoft. In tegenstelling tot de gemaakte architectuur zijn er niet drie maar negen klassen nodig om het Spatial Mapping en Spatial Processing techniek uit te voeren. De functie van deze klassen is in paragraaf 4.4 en 4.5 al uitgelegd en de klassen zelf zijn te verkrijgen op de HoloLens Academy 230 (Microsoft Holograms 230, n.d.). Ook is er een toestemming nodig voor de werking van het Spatial Mappen, namelijk de SpatialPerception toestemming die in paragraaf 7.3 al toegekend was.

Hiernaast zijn er nog twee andere klassen aangemaakt om de Spatial Package eenvoudig te kunnen beheren. De eerste klas is de SpatialManager, hierin wordt de Spatial Mapper aan en uit gezet. Ook zullen de waardes die berekend zijn in de SpatialCalculations klas alleen toegankelijk zijn via de SpatialManager, zodat alleen deze klas dient als toegangspunt van de Spatial Package.

De SpatialCalculations is de tweede klas die aangemaakt is in de Spatial Package en deze klas voert de calculaties uit die in hoofdstuk 4.4 zijn beschreven. Deze calculaties worden uitgevoerd nadat de Spatial Mapping is afgerond en de berekende waardes worden in variabelen opgeslagen, zodat ze makkelijk opgevraagd kunnen worden door de SpatialManager.

7.7 CONTROLLERS

Het laatste onderdeel dat voor de werking van de applicatie gemaakt moet worden, is de RoomController. Deze klas zorgt voor het laatste onderdeel, namelijk het selecteren van de kamer. Zodra de calculaties zijn uitgevoerd in de SpatialCalculations, vraagt de RoomController aan de SpatialManager de afmetingen op en geeft deze door aan de BuildingHelper om vervolgens een lijst met kamers te verkrijgen die voldoen aan de afmetingen. Gebaseerd op de aantal kamers in deze lijst kunnen verschillende acties ondernomen worden.

Als de lijst leeg is kan de RoomController de gebruiker vragen om een keuze te maken tussen een aantal verschillende opties, zoals selecteren van een ander model, opnieuw scannen van de kamer of het verhogen van de toleranties. Indien deze laatste gekozen wordt, zal er tot een tolerantie van vijf meter gezocht worden naar kamers. Hierbij zal het zoeken wel in stappen van tien centimeter gebeuren, zodat de gebruiker ook echt alleen kamers te zien krijgt die het meest overeenkomen met de kamer waar de gebruiker zich op dat moment bevindt.

In het geval dat er meerdere kamers gevonden zijn, zal het gehele model in miniatuur in het midden van de kamer geplaatst worden. Hierbij zullen alleen de kamers die voldoen aan de afmetingen getoond worden, waardoor de gebruiker kan zien hoe de kamers eruitzien en op welke verdiepingen de kamers zich bevinden binnen het gebouw. Vervolgens kan de gebruiker op een van kamers klikken om de kamer in te laden.

Als er maar één kamer is gevonden of één kamer is geselecteerd uit meerdere gevonden kamers zal deze kamer ingeladen moeten worden. Om te beginnen zulle andere kamers teruggezet worden naar de beginwaarden en uitgezet worden. Hierdoor blijft alleen de kamer over die ingeladen moet worden. Via de BuildingHelper zal de kamer vervolgens geroteerd en gepositioneerd worden.

Nadat de kamer ingeladen is zal er via de Interaction Package een HoloUI getoond worden waarin de gebruiker de optie krijgt de kamer negentig graden naar links of rechts te draaien. Deze rotatie zal ook via de BuildingHelper gaan.

SPRINT OVERGANG

Zoals aangegeven in de vorige demo zou er verder gewerkt worden aan de applicatie, waarbij als eerst de Interaction Package afgerond zou worden en vervolgens door gewerkt zou worden aan de overige onderdelen.

SPRINT 5

Aan het einde van sprint 5 was de applicatie zo goed als afgerond. Alle requirements waren geïmplementeerd en de applicatie was klaar om getoond te worden aan de opdrachtgever. Tijdens de demo is er daarom een volledige demo gegeven van de applicatie, waarin alle aspecten van de applicatie getoond zijn. Dit werd positief ontvangen en de applicatie deed wat volgens de beschrijving moest gebeuren.

Hierbij waren echter wel een aantal opmerkingen, namelijk dat het scannen niet altijd even goed gaat en de gebruiker dit tijdens het proces al kan merken. Daarom moesten er acties beschikbaar zijn om stappen terug te zetten. Deze acties zijn dezelfde acties die de gebruiker kan kiezen als er geen kamers gevonden zijn, waardoor het dus een eenvoudige aanpassing zou zijn. Deze aanpassing is opgenomen als requirement UR05 en zou in de volgende sprint geïmplementeerd worden.

De tweede aanpassing is de mogelijkheid voor het handmatig plaatsen van hoeken. Dit heeft vooral te maken met de gemaakte scan en daarom moest er een alternatief komen voor het detecteren van de hoeken, namelijk dat de gebruiker zelf de hoeken kan markeren. Dit is opgenomen als de nieuwe requirement UR06.

SPRINT 6

De applicatie was zo goed als af en er waren een paar kleine aanpassingen die gemaakt moesten worden en deze zullen dus in sprint 6 gemaakt worden.

7.8 OPTIMALISATIE

Voordat de aanpassingen van de opdrachtgever geïmplementeerd werden, was er een ander probleem opgedoken, namelijk het systeem waarop ontwikkeld werd. Hierbij zijn verschillende problemen boven water gekomen die eerst opgelost moesten worden, voordat de nieuwe requirements gebouwd konden worden.

7.8.1 SYSTEEM PROBLEMEN

Aan het begin van sprint 6 was er een technisch probleem met het systeem waarop ontwikkeld werd, waardoor de omgeving opnieuw opgezet moest worden. Tijdens het opnieuw opzetten van de omgeving is er gebruik gemaakt van de nieuwe versies van een aantal tools die gebruikt werden. Deze tools hadden vooral betrekking tot Visual Studio en Unity. Tijdens de ontwikkeling was er namelijk gebruik gemaakt van Visual Studio 2015 en Unity versie 5.5.0. Na de problemen met het systeem, is er gebruik gemaakt van Visual Studio 2017 en Unity versie 5.6.0.

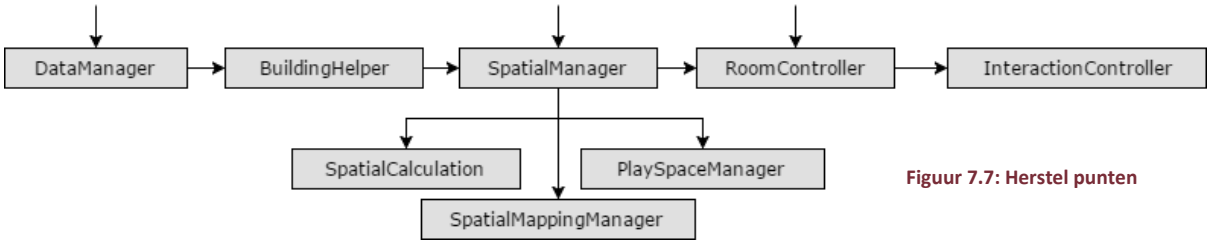
Door deze verandering werkte de gebouwde applicatie niet meer en werden er problemen aangegeven tijdens het uitvoeren en bouwen van de applicatie. Aangezien Visual Studio alleen een code editor is, is er gekeken of het kan komen door het updaten van Unity. Tijdens het zoeken naar een oplossing kwam er een forum post van een andere ontwikkelaar naar voren die hetzelfde probleem had. Op deze post was als antwoord aangegeven dat het te maken heeft met de UWP-versie (thebanjomatic, n.d.) en niet Unity. UWP is een verzamelnaam dat staat voor Universal Windows Platform. Dit is een platform waar alle producten van Microsoft van draaien, zoals Windows 10, Windows Mobile, Windows Holographic en zelfs Xbox.

Deze versie was blijkbaar niet bijgewerkt tijdens de installatie van Visual Studio, waardoor Unity de oude UWP-versie aan het gebruiken was, waarin onderdelen opgenomen waren die niet meer ondersteund werden. Daarom moest de nieuwe UWP-versie ook geïnstalleerd worden, waarna de applicatie direct weer werkte.

7.8.2 GENERALE ACTIES

De eerste nieuwe requirement die gebouwd is, is requirement UR05. Deze requirements is aangepakt in de Interaction Package. Hier is er een nieuw onderdeel aan toegevoegd, genaamd GeneralUI, wat eigenlijk een HoloUI is die constant beschikbaar zal zijn voor de gebruiker. Zodra de gebruiker een bepaald mijlpunt bereikt heeft binnen de applicatie, komt er in deze HoloUI een actie tevoorschijn waarmee de gebruiker stap(pen) terug kan zetten.

Zodra een gebruiker een actie kiest wordt er in de desbetreffende onderdeel een actie uitgevoerd die alle instellingen binnen het onderdeel herstelt en de methodes aanroept die behoort in onderdelen na de gekozen punt. Dit wordt duidelijk in figuur 7.7, waarin te zien is dat de gebruiker kan herstellen op de DataManager, SpatialManager of RoomController. Als een van deze punten gekozen wordt, zullen de andere onderdelen, die na het gekozen punt komen, als een soort van kettingreactie hersteld worden.



Figuur 7.7: Herstel punten

7.8.3 HOEK MARKERING

Het laatste onderdeel dat gebouwd moest worden is de mogelijkheid om hoeken handmatig te markeren. Dit speelt zich af in de Spatial Package, waarbij er gelet wordt of de gebruiker klikt op een gescand onderdeel. Als een gebruiker klikt op een gescand onderdeel, wordt er een markering geplaatst, waarmee duidelijk wordt waar de gebruiker geklik heeft. Als de gebruiker de markeringen wilt gebruiken moet er in totaal vier markeringen geplaatst worden, waarna de “gebruik markering” optie gekozen kan worden in de HoloUI dat op dat moment te zien is.

Door het gebruik van markeringen hoeft de SpatialCalculations klas niet meer uit te rekenen waar de hoeken van de kamer zich bevinden. In plaats hiervan worden de coördinaten van de geplaatste markers gebruikt waardoor het vervolg van de processen op dezelfde manier uitgevoerd worden.

SPRINT OVERGANG

SPRINT 6

Na het afronden van de aanpassingen die verkregen waren tijdens de vorige demo, was er nog tijd voor het einde van sprint 6. Daarom is er een video gemaakt van de applicatie en heeft de applicatie de naam HoloBuilding gekregen. Deze video laat de mogelijkheden van de applicatie zien en deze video is dan ook gebruikt om de applicatie aan het einde van sprint 6 te tonen aan de opdrachtgever. Hierbij zijn er geen nieuwe functionaliteiten naar voren gekomen en was het project dus afgerond.

SPRINT 7 & 8

De laatste twee sprints zijn gebruikt om de documentatie over de applicatie af te ronden en te werken aan de afstudeerscriptie.

8 CONCLUSIE EN AANBEVELING

Tijdens het project is het duidelijk geworden dat de technieken die de HoloLens aanbiedt nog in zijn babyschoentjes staan. De technieken doen wat ze moeten doen, maar het is te merken dat veel onderdelen nog niet geoptimaliseerd zijn en dus niet altijd werken zoals ze bedoeld zijn. Ook zorgt de strikte controle van Microsoft ervoor dat de HoloLens niet op zijn volle potentie gebruikt kan worden. Hiermee wordt bedoeld dat de applicaties die gebouwd worden, draaien in een afgesloten omgeving en dat alleen de technieken zoals Spatial Mapping en Processing beschikbaar zijn voor gebruik. De overige instellingen van de HoloLens zullen niet toegankelijk zijn voor aanpassing binnen de applicatie en is het dus gebruiker afhankelijk.

Desondanks deze strikte controle en het feit dat de technieken nog in zijn babyschoentjes staan, kunnen wel mooie nieuwe producten opgeleverd worden met de Microsoft HoloLens. Deze producten hebben wel veel inspanning nodig, aangezien het niet altijd mogelijk is alle wensen op de beste manier uit te voeren.

Als het op de onderzoeksvragen van dit project aankomt, kan gezegd worden dat momenteel Unity de beste engine is voor ontwikkeling voor de Microsoft HoloLens. Deze engine is meegenomen in het onderzoek naar de te gebruiken 3D modellen, maar hierbij is er uiteindelijk alleen gebruik gemaakt van de OBJ-modellen. Deze modellen kunnen uitgelezen en ingeladen worden, waardoor er dus een kamer gekoppeld kan worden aan de echte wereld.

Deze verschillende bevindingen hebben geleid tot de conclusie dat het mogelijk is 3D modellen te visualiseren met de Microsoft HoloLens, mits het acceptabel is dat de applicatie zal aanvoelen als een Alpha versie. Hiermee wordt bedoeld dat de technieken die de HoloLens aanbiedt nog niet optimaal zijn en het dus nog niet duidelijk is wat voor problemen kunnen voorkomen.

Als vervolgonderzoek zou er gekeken kunnen worden naar de verschillende modellen die tijdens het project niet meegenomen zijn. Indien deze modellen ook kunnen voldoen aan de verschillende requirements zal het mogelijk zijn een volwaardig applicatie te bouwen waarin verschillende modeltypes ingeladen kunnen worden.

Ook is het aan te raden het gebruik van de Spatial Mapping technologie van Microsoft te optimaliseren voor het project. Hiermee wordt bedoeld dat de hoekherkenning verbeterd zou kunnen worden, waardoor de juiste hoeken herkend kunnen worden, ook als de scan niet optimaal is.

9 EVALUATIE

In dit hoofdstuk zullen de resultaten van het project worden geëvalueerd. Hierbij zal er gekeken worden naar de producten die opgeleverd zijn, de aanpak en de competenties die behaald moesten worden. De producten die opgeleverd moesten worden en de competenties staan tevens beschreven in Bijlage Afstudeerplan.

9.1 EVALUATIE VAN PRODUCTEN

Voor de aanvang van het project was een afstudeerplan opgesteld, waarin een aantal op te leveren producten beschreven waren, namelijk:

- Onderzoeksrapport 3D Engines
- Onderzoeksrapport 3D Modellen uitlezen met .NET (was CAD-modellen)
- Onderzoeksrapport Koppelen van 3D modellen (was Koppelen data)
- Documentatie werking applicatie
- Documentatie applicatie

Onderzoeksrapport 3D Engines

Het onderzoek dat bij dit rapport hoort, is te zien in paragraaf 4.2. Binnen dit onderzoek is er onderzocht welke engines gebruikt konden worden en hoe deze verschillende engines zouden passen binnen het project. Uiteindelijk is er gekozen voor één engine en is beargumenteerd waarom. In het verloop van het project heb ik het een goede keuze gevonden, aangezien het project vereenvoudigd is door het gebruik van de engine. Een aantal sterke punten van de engine zijn zeker van pas gekomen, zoals de online community die beschikbaar was voor vragen en de interface van de engine die een aantal taken eenvoudig maakte.

Onderzoeksrapport 3D modellen uitlezen met .NET

Dit onderzoek zou eigenlijk gaan over het gebruik van CAD-modellen, maar al vroeg werd het duidelijk dat deze modellen niet gebruikt konden worden. Hierom is er met de opdrachtgever besproken over een oplossing en is het onderzoek aangepast, zodat er gebruik gemaakt zou worden van 3D modellen, zie paragraaf 4.3. De keuze die als gevolg van het onderzoek gemaakt is, paste het beste bij de requirements die opgesteld waren. Zoals in hoofdstuk 8 beschreven is, zou een vervolgonderzoek het gebruik van de andere modellen kunnen zijn. Dit kon helaas door tijdsgebrek niet tijdens dit project uitgevoerd worden.

Als opmerking heb ik bij dit onderzoek dat voorafgaande het project beter uitgezocht kon worden wat de verschillende taken precies inhouden. Het was al snel duidelijk dat CAD-modellen niet gebruikt konden worden en dit had ook voor de aanvang van het project gezien worden. Dit was dan ook een goed leermoment, om in toekomstige projecten vooraf beter te uitzoeken of het project haalbaar is, zoals het beschreven staat.

Onderzoeksrapport Koppelen van 3D modellen

Dit onderzoeksrapport is te zien in paragraaf 4.4 en dit is ook een onderzoek dat aangepast is, als gevolg van het vorige onderzoek. Doordat er gebruik gemaakt zou worden van 3D modellen en geen CAD-modellen moest er specifiek onderzocht worden hoe de modellen die gekozen waren in vorige onderzoek gekoppeld konden worden aan de echte wereld door middel van de Microsoft HoloLens. De omvang van dit onderzoek was voldoende en tijdens de ontwikkeling van de applicatie kon de koppeling er makkelijk ingebouwd worden.

Documentatie werking applicatie

De werking van de HoloLens applicatie is beschreven in paragraaf 6.2, waarbij alle acties die uitgevoerd kunnen worden beschreven staan.

Documentatie applicatie

De documentatie van de applicatie bestaat uit twee onderdelen, namelijk de architectuur en uitvoer. Deze zijn tevens te vinden in paragraaf 6.3 en hoofdstuk 7.

9.2 EVALUATIE VAN AANPAK

Terugkijkend op het project is het duidelijk geworden dat het een goede zet was het project uit te voeren op een Agile methodiek. Dit omdat de technieken die gebruikt zijn nieuw zijn en het dus niet altijd duidelijk was, wat mogelijk was en wat niet. Door aan het eind van elke sprint een demo te geven aan de opdrachtgever, was de opdrachtgever op de hoogte hoe ver het project was en wat voor problemen opgetreden waren.

De problemen konden op deze manier makkelijker opgelost worden, aangezien het mogelijk was de oplossingen, voor de problemen, voor te leggen aan de opdrachtgever en te bespreken of de oplossing zou passen binnen de verwachtingen van het project. Ook was het mogelijk voor de opdrachtgever wijzigingen en toevoegingen te brengen aan het project en dus de uitkomst van het project te veranderen.

Buiten de aanpak methodiek van het project was ook de planning van het project goed ingedeeld. De onderdelen die ingepland waren, waren op een goede volgorde ingedeeld en ook de tijds toewijzing was voldoende per onderdeel. Alleen was het niet duidelijk dat een deel van een onderzoek later in het project uitgevoerd zou worden en dat komt dan ook vooral doordat er vooraf niet voldoende uitgezocht is over het project.

Hiermee komen we op een punt dat slecht was uitgevoerd en dat was dat het vooraf het project niet precies duidelijk was wat gedaan zou worden om het project te realiseren. Een aantal problemen die al vroeg in het project naar voren kwamen, hadden vooraf al opgelost kunnen worden. Hiermee wordt het probleem met de CAD-modellen bedoeld en ook was het niet duidelijk wat de koppeling van kamers precies inhield en kon er dus geen rekening gehouden worden met onderdelen van het onderzoek dat achteraf uitgevoerd zouden worden. De reden dat deze onderdelen later uitgevoerd moesten worden is omdat de kennis vooraf niet groot genoeg was om bepaalde betere oplossingen een aankomende problemen te kunnen inzien. Hiervoor moest er eerst gewerkt worden met de tools en technieken, waarna betere oplossingen en implementaties ingezet konden worden.

Al met al, is het project soepel verlopen. Er zijn heel wat problemen opgetreden, maar door een goede communicatie met de opdrachtgever waren deze problemen snel opgelost.

9.3 EVALUATIE VAN COMPETENTIES

De competenties ook wel beroepstaken genoemd, zal ik zelf in dit paragraaf oordelen. Hierbij zal ik gebruik maken van een tabel dat voortkomend is uit de handleiding “Beroepstaken van de Opleiding Informatica” dat onderdeel is van de opleiding Informatica aan de Haagse Hogeschool.

	Geleid	**Zelfstandig**	**Sturend**
Simpel	1	2	3
Lastig	2	3	4
Complex	3	4	5

Tabel 9.1: Niveau Competenties (Academie voor ICT & Media, 2009)

In het Afstudeerplan dat te vinden is in de bijlage zijn er ook een aantal competenties opgenomen die behaald moesten worden tijdens het project. Hierbij was er ook aangegeven aan welke niveau deze competentie moest voldoen. Het niveau zal in dit hoofdstuk beoordeeld worden en ook zal er beschreven worden hoe de competentie aangetoond is door middel van de scriptie.

1.1 Selecteren van methoden, technieken en tools

Selecteren en adviseren over methoden en technieken om een systeem te ontwikkelen, beheren en testen.

Tijdens de ontwikkeling zijn er verschillende technieken en tools gekozen en gebruikt. Zoals .NET en Spatial Mapping. Er is onderzocht wat nodig was om het project waar te kunnen maken en de keuzes van technieken en tools die als gevolg van dit onderzoek naar voren zijn gekomen staan beschreven in hoofdstuk 5. Het project is zelfstandig uitgevoerd en er is rekening gehouden met de te gebruiken nieuwe technieken en natuurlijk de wensen van de opdrachtgever. Hierdoor valt deze competentie op niveau 3 of 4. Bij niveau 4 is het te argumenteren dat er meerdere stakeholders aanwezig moeten zijn, maar tijdens dit project was alleen de opdrachtgever een stakeholder.

1.3 Selecteren van standaardsoftware

Adviseren over en/of selecteren van software, zoals een applicatie (commerciële software of open source), framework (al dan niet gebaseerd op open standaarden), databasemanagementsystemen of besturingssysteem.

Tijdens het afstuderen is er onderzocht wat voor tools en technieken gebruikt kunnen worden. Hierbij is er rekening gehouden met de wensen van de opdrachtgever, evenals de comptabiliteit met de nieuwe technieken die gebruikt zijn. Dit heeft geleid tot de selectie van applicaties als Unity en AutoDesk Maya en ook is er een selectie gemaakt van het gebruikte model. Hierbij is er zelfstandig gewerkt in een lastige omgeving, wat uitkomt op niveau 3.

3.1 Ontwerpen van Softwarearchitectuur

Bepalen en beschrijven van de verschillende subsystemen en componenten waarin een applicatie is onderverdeeld en de relaties tussen die delen.

Het ontwerpen van de software heeft geleid tot meerdere systemen, waarbij de hoofdapplicatie bestaat uit meerdere systeemdelen. Hierbij is er rekening gehouden met de requirements die voldaan moesten worden en is er zelfstandig gewerkt. Het rekening houden met meerdere systemen en hoe de systeemdelen met elkaar werken zorgde voor een complexe omgeving en komt daarmee uit op niveau 4.

3.2 Ontwerpen systeemdeel

Beschrijven van systeemdelen (subsystemen, componenten, modules), hun onderliggende structuur en het gedrag in detail, zodanig dat bouwen van het systeemdeel mogelijk is.

Om het gedrag van de applicatie weer te geven is er een activiteitendiagram gemaakt, waarin de verschillende paden van de applicatie weergegeven worden. Ook is er per onderdeel aangegeven waar het voor dient en hoe het onderdeel opgebouwd is. Verder is er rekening gehouden met toekomstige uitbreidingen door het gebruik van design patterns. Hierdoor kan ook dit onderdeel beschouwd worden als een complexe aanpak, waarbij rekening is gehouden met uitbreiding, wat neerkomt op niveau 4.

3.3 Bouwen Applicatie

Bouwen en documenteren van de systeemdelen. Systeemdelen samenstellen tot een werkende applicatie.

Bij het bouwen van de applicatie is er gewerkt volgens de architectuur en is er incrementeel gebouwd (per systeemdeel). Hierbij is er gebruik gemaakt van verschillende frameworks, zoals .NET. Ook is er gebruik gemaakt van GitHub als versiebeheer. Na het bouwen van de verschillende systeemdelen was het makkelijk de systeemdelen samen te stellen tot een werkende applicatie. Dit heeft ook een complexe applicatie opgeleverd, die gebouwd is met nieuwe technieken. Hierdoor komt dit onderdeel uit op niveau 4.

LIJST VAN FIGUREN EN TABELLEN

WOORDENLIJST

.NET - Een framework, ontwikkeld door Microsoft, voor het ontwikkelen van applicaties. 2, 6, 15, 18, 24

Agile - Een iteratieve aanpak voor software ontwikkeling. 2, 5, 6

AssetBundle - Speciaal model dat alleen gemaakt en gebruikt kan worden in Unity. 18

Augmented Reality - Een live, direct of indirect, beeld van de werkelijkheid waaraan elementen worden toegevoegd door een computer. 2

AutoDesk Maya - Programma voor het modelleren van objecten in een 3D omgeving. 14, 29

Base64 - Een codering, voor het coderen van binaire bestanden naar een tekstformaat. 38, 45, 48

C# - Programmeertaal gebruikt binnen het .NET framework van Microsoft. 6, 10, 11, 15, 24

C++ - Een programmeertaal dat object-georiënteerd en generiek is. Verder is het bekend voor het aanbieden van low-level geheugen manipulatie. 10, 11, 18

Caching - Opslaan van data voor later gebruik. 27

CAD *(Computer-aided Design)* - Gebruik van computers voor het ontwerpen. 12

CAD-modellen - Model in een CAD bestandsformaat. 12

CPU *(Central Processing Unit)* - Brein van een computer, waarin alle calculaties uitgevoerd worden. 3

DAE *(Digital Asset Exchange)* - Bestandsformaat dat ook wel bekend staat als COLLADA, wat gebruikt wordt voor het behouden van digitale content. 13, 15, 16, 18, 19

DirectX - Een collectie van methoden voor het afhandelen van taken gerelateerd aan multimedia, voornamelijk spelletjes en video, op Microsoft Platformen. 9, 11

Entity Framework - Een open source ORM voor gebruik binnen het .NET framework. 44, 46

FBX *(FilmBoX)* - Bestandsformaat voor het behouden van digitale content dat onderdeel is van AutoDesk. 13, 15, 16, 18, 19

Filesystem - Opslagsysteem 32, 38, 43, 44, 48

GameObjecten - Benaming van objecten binnen Unity. 12, 15, 47

Goniometrie - Tak vanuit de wiskunde voor het berekenen van vorm, afmeting, relatieve posities en eigenschappen van ruimtes. 24

GUI *(Graphical User Interface)* - Grensvlak/interface waarin de gebruiker acties kan uitvoeren. 40, 52

HDD *(Hard Drive Disk)* - Schijf voor dataopslag. .. 44

HMD *(Head-Mounted Display)* - Een beeldscherm dat gekoppeld is aan de hoofd van een gebruiker. 3

Hologrammen - Computer gegenereerde objecten 2

HPU *(Holograpic Processing Unit)* - Processor voor holografische taken, zoals Spatial Mapping en het herkennen van gebaren. 3

HTML *(HyperText Markup Language)* - Standaard opmaaktaal voor het maken van webpagina's en -applicaties. 45, 46

HTTP *(HyperText Transfer Protocol)* - Protocol voor communicatie tussen informatiesystemen. 32, 48

IDE *(Integrated Development Environment)* - Omgeving voor het ontwikkelen van computersoftware. 29

LITERATUUR

Academie voor ICT & Media. (2009, 06). Beroepstaken van de opleiding Informatica. Den Haag, Zuid-Holland, Nederland.

Anonimo, e. (2012). *Unity3D Wiki*. Retrieved from http://wiki.unity3d.com/index.php?title=ObjImporter

Assimp. (2016). *Assimp*. Retrieved from http://assimp.sourceforge.net/

DaveVoyles. (2016, 07 26). *forums.unrealengine*. Retrieved from unrealengine: https://forums.unrealengine.com/showthread.php?118375-Unreal-Engine-4-is-available-for-Win10-UWP-app-dev-now

DirectX. (n.d.). *developer.microsoft*. Retrieved from Microsoft: https://developer.microsoft.com/en-us/windows/mixed-reality/creating_a_holographic_directx_project

EF Code-First. (2016, Oktober 23). *msdn.microsoft*. Retrieved from microsoft: https://msdn.microsoft.com/en-us/library/jj193542(v=vs.113).aspx

Eric5h5. (2011, July 13). Retrieved from http://answers.unity3d.com/questions/143109/why-does-a-primitive-cube-contain-24-verts.html

Extraction, C. S. (2016, July 22). *CurvSurf*. Retrieved from Youtube: https://www.youtube.com/watch?v=C7mLH_5QzvU

Hajjaj, M. E. (n.d.). *HoloMeasure*. Retrieved from Microsoft: https://www.microsoft.com/en-us/store/p/holo-measure/9nblggh52bq1

HoloSpecs. (n.d.). *developer.microsoft*. Retrieved from microsoft: https://developer.microsoft.com/en-us/windows/mixed-reality/hololens_hardware_details

HoloToolkit-Unity. (2017). Retrieved from github: https://github.com/Microsoft/HoloToolkit-Unity

KeywordRecognizer. (2017). *unity3d*. Retrieved from docs.unity3d: https://docs.unity3d.com/ScriptReference/Windows.Speech.KeywordRecognizer.html

Khronos. (n.d.). *Khronos*. Retrieved from Khronos: https://www.khronos.org/collada/

Kipman, A. (2016, 02 29). *blogs.windows*. Retrieved from windows: https://blogs.windows.com/devices/2016/02/29/announcing-microsoft-hololens-development-edition-open-for-pre-order-shipping-march-30/#5wMQWy0bxje1IMzS.97

Mark Graham, M. Z. (2013). Augmented reality in urban places: contested content and the duplicity of code. Transactions of the Institute of British Geographers, 38: 464–479. . Transactions of the Institute of British Geographers.

Microsoft (Why HoloLens). (n.d.). *why-hololens*. Retrieved from microsoft: https://www.microsoft.com/en-us/hololens/why-hololens

Microsoft Float. (2017, March 29). Retrieved from docs.microsoft.com: https://docs.microsoft.com/en-us/dotnet/articles/csharp/language-reference/keywords/float

Microsoft Holograms 230. (n.d.). *developer.microsoft*. Retrieved from microsoft: https://developer.microsoft.com/en-us/windows/mixed-reality/holograms_230

Microsoft Mesh Processing. (n.d.). *developer.microsoft*. Retrieved from microsoft: https://developer.microsoft.com/en-us/windows/mixed-reality/spatial_mapping#mesh_processing

Microsoft. (n.d.). *Microsoft HoloLens*. Retrieved from Microsoft: https://www.microsoft.com/microsoft-hololens/en-us

Microsoft Spatial Mapping. (n.d.). *developer.microsoft*. Retrieved from microsoft: https://developer.microsoft.com/en-us/windows/mixed-reality/spatial_mapping

Microsoft Systems Journal. (1997). *microsoft*. Retrieved from microsoft: https://www.microsoft.com/msj/0597/visualstudio97.aspx

MS Scanning. (n.d.). *developer.microsoft*. Retrieved from microsoft: https://developer.microsoft.com/en-us/windows/mixed-reality/holograms_230#chapter_1_-_scanning

Myerson, T. (2015, 01 21). *blogs.windows*. Retrieved from windows: https://blogs.windows.com/windowsexperience/2015/01/21/the-next-generation-of-windows-windows-10/#Cp5uxMmZgYDYH7sy.97

Quintor. (n.d.). Retrieved from https://www.quintor.nl/wie-we-zijn/

thebanjomatic. (n.d.). *forums.hololens*. Retrieved from hololens: https://forums.hololens.com/discussion/7194/wack-errors-after-updating-to-visual-studio-2017-and-unity-5-6-0f3

Unity. (2017). *3D Formats*. Retrieved from unity3d: https://docs.unity3d.com/Manual/3D-formats.html

Unity Answers. (2010). *answers.unity3d*. Retrieved from unity3d: http://answers.unity3d.com/questions/9675/is-unity-engine-written-in-monoc-or-c.html

Unity Bounds. (2017). *docs.unity3d.com*. Retrieved from Unity3D: https://docs.unity3d.com/ScriptReference/Renderer-bounds.html

Unity Multi Platform. (n.d.). *unity3d*. Retrieved from https://unity3d.com/unity/multiplatform/

Unity Scripting. (2017). *Scripting Reference*. Retrieved from Unity: https://docs.unity3d.com/ScriptReference

Unity Users. (2015). *forum.unity3d*. Retrieved from unity3d: https://forum.unity3d.com/threads/pros-and-cons-of-different-3d-model-formats-fbx-dae-ect.344009/

Unity WWDC. (2005). *unity3d*. Retrieved from https://unity3d.com/company/public-relations/news/unity-stage-wwdc

UnityWebRequest. (2017). *docs.unity3d*. Retrieved from unity3d: https://docs.unity3d.com/Manual/UnityWebRequest.html

vrs. (n.d.). *virtual-reality*. Retrieved from vrs: https://www.vrs.org.uk/virtual-reality/what-is-virtual-reality.html

BIJLAGEN

CODE

CODE 01

```
1 reference
public GameObject StructorizeBuilding(GameObject model)
{
    GameObject root = null;

    var maxTries = 10;
    var tryCount = 0;
    while (model.transform.childCount > 2)
    {
        root = StructorizeChild(model, root, "_- ,.");
        if (maxTries >= tryCount++) continue;

        Debug.Log("Couldn't parse building.");
        break;
    }

    Destroy(model);
    if (root == null) return root;
    var building = Instantiate(root);
    building.name = "BuildingModel";
    return building;
}

1 reference
private GameObject StructorizeChild(GameObject building, GameObject root, string delimeters)
{
    foreach (Transform child in building.transform)
    {
        var objStructure = child.name.Split(delimeters.ToCharArray());

        switch (objStructure.Length)
        {
            case 1:
                if (objStructure[0] == "default") break;
                root = root == null ? child.gameObject : root;
                break;
            case 2:
                child.parent = root.transform;
                child.name = objStructure[1];
                break;
            case 3:
                foreach (Transform room in root.transform)
                {
                    if (room.name.Equals(objStructure[1]))
                    {
                        child.parent = room;
                        child.name = objStructure[2];
                    }
                }
                break;
        }
    }
    return root;
}
```

CODE 02

```
1 reference
public static Vector3 GetActualSize(this Transform elem)
{
    Vector3 actual = Vector3.zero;
    MeshFilter mf = elem.GetComponent(typeof(MeshFilter)) as MeshFilter;
    if (mf == null) return actual;

    Mesh mesh = mf.sharedMesh;
    if (mesh == null) return actual;

    Vector3 size = mesh.bounds.size;
    Vector3 scale = elem.localScale;

    actual = Vector3.Scale(size, scale);
    actual.Abs();

    return actual;
}
```

CODE 03

```
1 reference
private void CalculateRotation(Point side)
{
    var point1 = side.point1;
    var point2 = side.point2;

    var rad = Mathf.Atan2(point2.z - point1.z, point2.x - point1.x);
    if (rad < 0) rad += 360;

    CalculatedRotation = rad * (180 / Mathf.PI);
}
```

CODE 04

```
2 references
public List<Room> FindMatchingRooms(Vector3 dimensions, float multiplier, float tolerance)
{
    Debug.Log(string.Format("looking for x: {0}, y: {1} and z: {2}", dimensions.x, dimensions.y, dimensions.z));
    var matchingRooms = new List<Room>();
    foreach (Transform room in BuildingModel.transform)
    {
        Destroy(room.gameObject.GetComponent<BoxCollider>());
        room.gameObject.SetActive(true);
        var actualSize = room.GetActualSize() * multiplier;
        var original = dimensions - actualSize;
        original = original.Abs();

        if (original.y > tolerance) continue;

        var rotated = dimensions - new Vector3(actualSize.z, actualSize.y, actualSize.x);
        rotated = rotated.Abs();
        if (original.x < tolerance && original.z < tolerance ||
            rotated.x < tolerance && rotated.z < tolerance)
        {
            var roomModel = new Room(room.gameObject, actualSize);
            matchingRooms.Add(roomModel);

            room.gameObject.AddComponent<BoxCollider>();
            room.gameObject.AddComponent<RoomSelector>();
            room.gameObject.GetComponent<RoomSelector>().model = roomModel;

            if (matchingRooms.Count == 1)
                room.gameObject.GetComponent<RoomSelector>().Chosen = true;
        }
        else
        {
            room.gameObject.SetActive(false);
        }
    }

    if (matchingRooms.Count > 0)
        BuildingModel.SetActive(true);

    return matchingRooms;
}
```

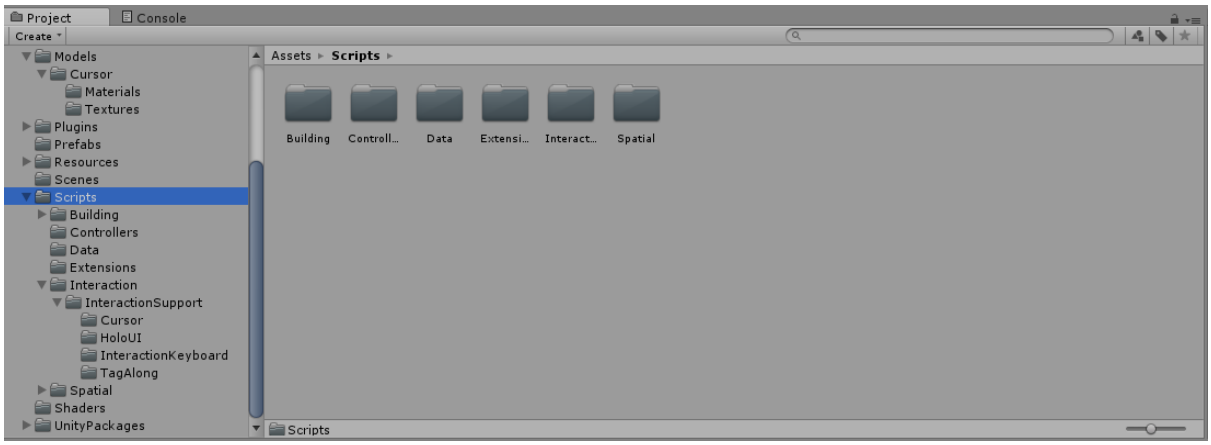
CODE 05

```
1 reference
public void SaveNewData(string data)
{
    DeleteOldModels();

    ModelFile modelFile = JsonConvert.DeserializeObject<ModelFile>(JsonConvert.DeserializeObject<string>(data));

    if (!string.IsNullOrEmpty(modelFile.Data))
    {
        FileLocation = Path.Combine(ModelLocation, modelFile.FileName + modelFile.DataType);
        byte[] bytes = Convert.FromBase64String(modelFile.Data);
        File.WriteAllBytes(FileLocation, bytes);
    }

    if (!string.IsNullOrEmpty(modelFile.AdditionalData))
    {
        byte[] bytes = Convert.FromBase64String(modelFile.AdditionalData);
        File.WriteAllBytes(Path.Combine(ModelLocation, modelFile.FileName + modelFile.AdditionalType), bytes);
    }

    DataReady = true;
}
```

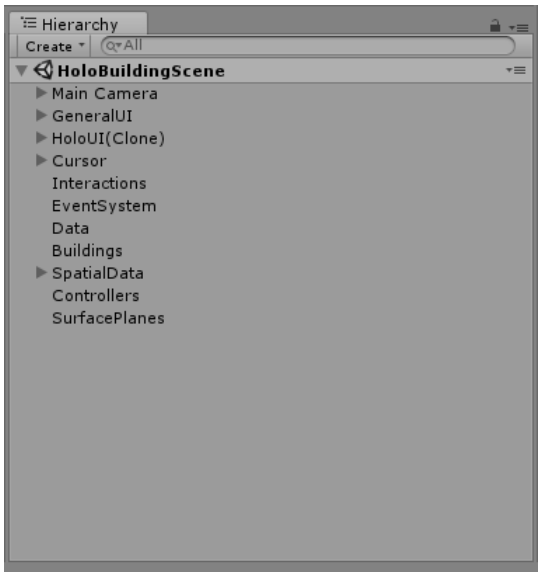
OVERIGE AFBEELDINGEN

UNITY

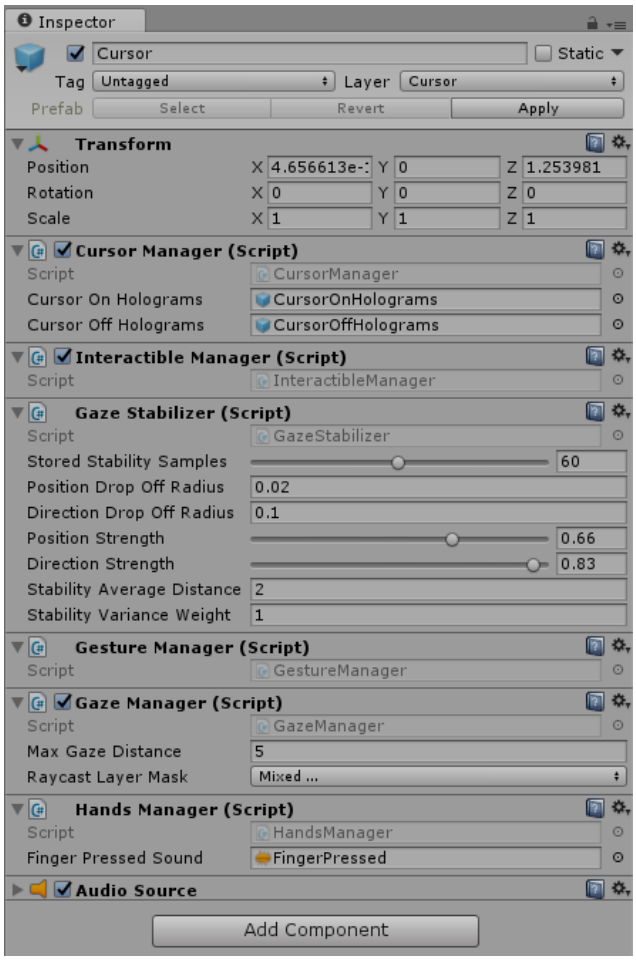
UNITY 01



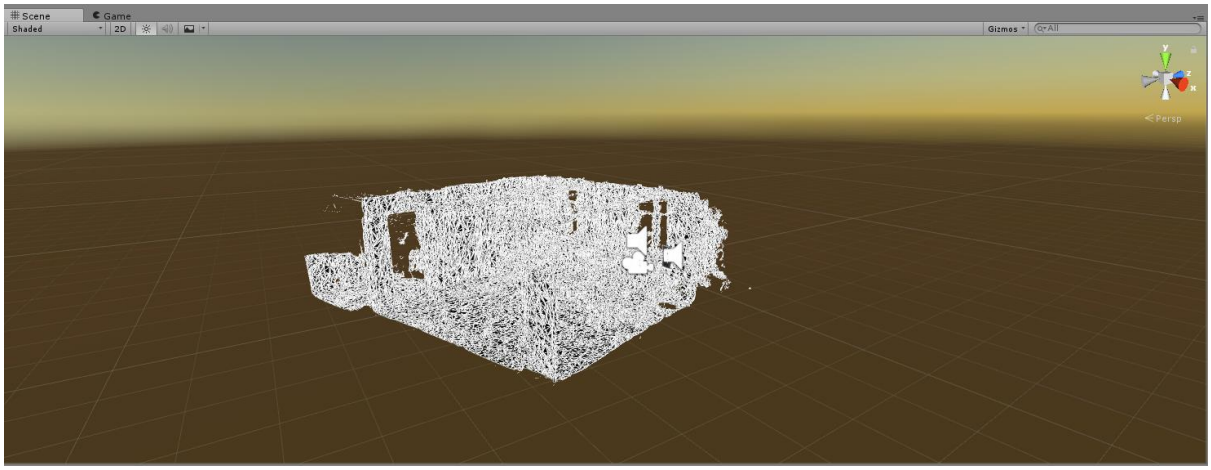
UNITY 02



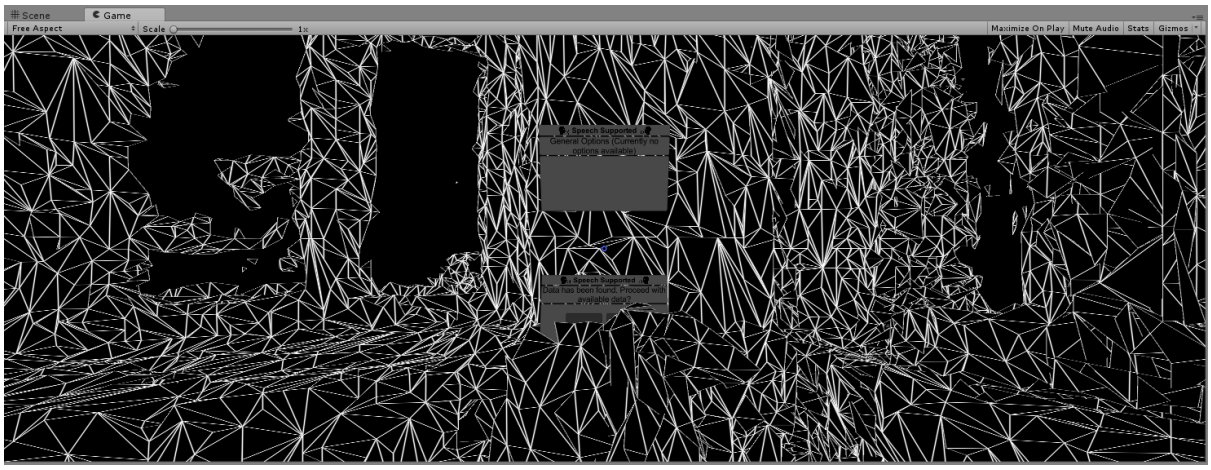
UNITY 03



UNITY 04

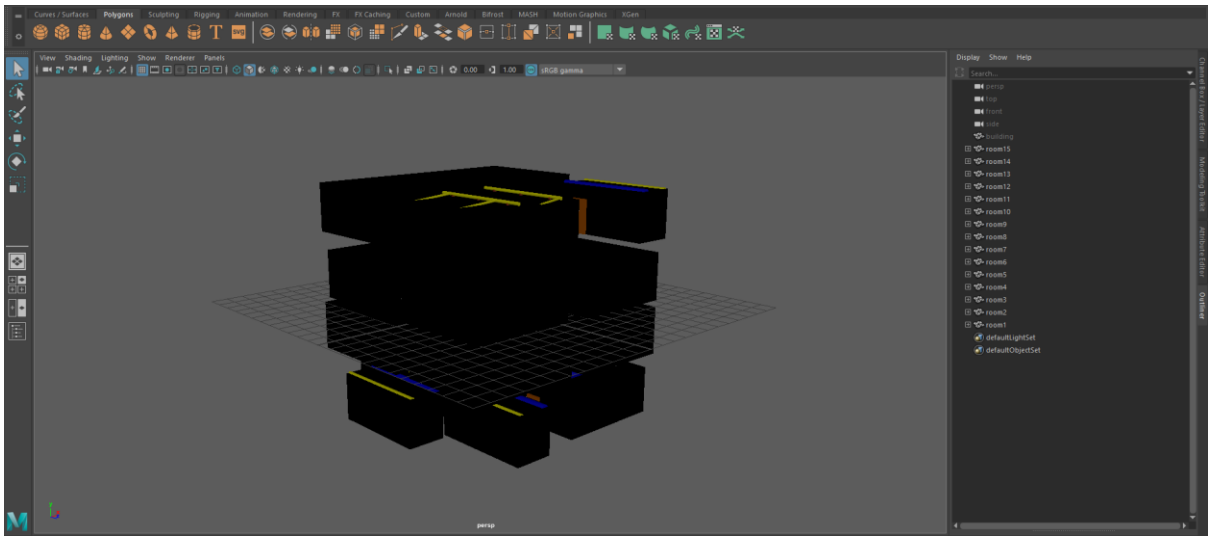


UNITY 05



MAYA

MAYA 01



Titel afstudeeropdracht: HoloLens gekoppeld aan 3D CAD-modellen.

Opdrachtomschrijving

1. Bedrijf

Quintor is een toonaangevend bedrijf op het gebied van Agile Software Development, Enterprise Java/ .NET technologie en mobile development. Tijdens de oprichting in 2005 waren er 4 medewerkers en er is sindsdien een gezonde groei gemaakt. Er zijn ondertussen vestigingen in Groningen, Amersfoort en Den Haag met meer dan 100 medewerkers in dienst

Verder beschikt Quintor over een Software Factory waarin inhouse projecten voor klanten worden uitgevoerd. In deze Software Factory wordt er geprobeerd altijd gebruik te maken van de nieuwste technieken. Daarom wordt er bij Quintor ook onderzocht hoe deze technieken werken en waar ze voor ingezet kunnen worden.

2. Probleemstelling

Een probleem bij de reconstructie van gebouwen, zijn de leidingen en kabels die achter de muren lopen. Doordat deze niet zichtbaar zijn, kan het soms moeilijk zijn om te bepalen waar er bijvoorbeeld geboord mag worden. Hierom zal een applicatie ontwikkeld moeten worden, dat het mogelijk maakt 3D CAD-modellen voor de ogen van gebruikers te renderen.

Hiervoor zal gekeken worden naar de Microsoft HoloLens, aangezien dit een goede oplossing is voor het probleem. Omdat de HoloLens nog nieuw is, is het nog niet duidelijk in welke scenario's de HoloLens precies ingezet kan worden. Door middel van de opdracht wordt er gehoopt dit te verduidelijken.

3. Doelstelling van de afstudeeropdracht

De opdracht is het ontwikkelen van programmatuur voor de HoloLens, waarbij een 3D CAD-model gekoppeld kan worden aan de ruimte waar de gebruiker zich op dat moment bevindt.

Tijdens de ontwikkeling kan er verwacht worden dat de documentatie over de HoloLens schaars zal zijn, aangezien het nog vrij nieuwe technologie is. Een doelstelling van opdracht is dus onderzoeken welke tools en methoden ingezet kunnen worden tijdens de ontwikkeling voor de HoloLens. Een voorbeeld hiervan is de omgeving om in te renderen, bijvoorbeeld Unity3D of een ander ondersteund 3D Engine.

De applicatie die opgeleverd zal worden zal de gebruiker de optie geven een 3D CAD-Model te selecteren, van het gebouw waar de gebruiker zich op dat moment bevindt. Vervolgens kan de applicatie, door middel van de HoloLens, de afmetingen van de kamer waar de gebruiker zich op dat moment bevindt berekenen. Door middel van deze berekeningen zal de juiste kamer in het 3D CAD-model gevonden moeten worden. Als de kamer gevonden is, kan de gebruiker kiezen uit de verschillende lagen die bij de kamer horen, waardoor de gebruiker bijvoorbeeld de leidingen die achter de muren lopen kan zien of bepaalde kenmerken, zoals stopcontacten.

4. Resultaat.

Het uiteindelijke resultaat is een applicatie dat het mogelijk maakt onzichtbare kenmerken van een kamer zichtbaar te maken. Deze kenmerken kunnen bijvoorbeeld de leidingen achter de muren zijn.

5. Uit te voeren werkzaamheden, inclusief een globale fasering, mijlpalen en bijbehorende activiteiten

Om de opdracht succesvol af te ronden zullen de volgende activiteiten uitgevoerd worden:

- Alle requirements samen met de opdrachtgever documenteren.
- Plan van aanpak schrijven
- Opzetten van werkomgeving
- Beschikbare literatuur en documentatie onderzoeken.
- Onderzoeken hoe CAD Modellen uitgelezen kan worden met .Net.
- Onderzoeken welke 3D Engine gebruikt kan worden om te visualiseren
- Onderzoeken hoe de data die verkregen wordt gekoppeld kan worden, dus de ruimte afmetingen van de HoloLens koppelen aan de juiste kamer van een 3D CAD Model. Hierbij kunnen design patterns en algoritmes onderzocht worden.
- Project uitvoering
  - Programma opzetten
  - Koppeling van CAD Modellen inbouwen
  - Koppeling van HoloLens inbouwen
  - Koppeling van Data inbouwen
  - Visualisatie van een 3D Engine bouwen
    - Hierbij kan het zijn dat er d.m.v. een 3D Modelling software een aantal 3D Objecten gemaakt moeten worden.
- Testen (en optimalisatie)

Verder zal er ontwikkeld worden op volgens de Agile filosofie, waarbij er backlogs opgesteld zullen worden met sprints van twee weken. Verder zal er één dag in de week gereserveerd worden om te werken aan de documentatie die uiteindelijk opgeleverd zal worden.

De applicatie zal gebouwd worden in Visual Studio d.m.v. .Net. De programmeertaal is C# en de libraries die gebruikt zullen worden zullen onderzocht moeten worden.

6. Op te leveren (tussen)producten

- Onderzoeksrapport CAD-modellen → Competentie 1.1, 1.2
- Onderzoeksrapport 3D Engines → Competentie 1.1, 1.2
- Onderzoeksrapport koppeling data (passende design patterns en algoritmes) → Competentie 1.1, 1.2
- Documentatie werking applicatie → Competentie 3.1, 3.2, 3.3
- Documentatie applicatie (werking van modules en gebruikte technieken en methodieken) → Competentie 3.1, 3.2, 3.3

7. Te demonstreren competenties en wijze waarop

1.1 Selecteren methoden, technieken en tools → Niveau 3
1.2 Selecteren van standaardsoftware → Niveau 3
3.1 Ontwerpen softwarearchitectuur → Niveau 4
3.2 Ontwerpen systeemdeel → Niveau 4
3.3 Bouwen applicatie → Niveau 4

DE HAAGSE HOGESCHOOL

3D MODELLEN GEVISUALISEERD MET DE MICROSOFT HOLOLENS

PLAN VAN AANPAK OVER HET VISUALISEREN VAN 3D MODELLEN MET DE MICROSOFT HOLOLENS IN OPDRACHT VAN DE HAAGSE HOGESCHOOL EN QUINTOR.

Auteur: Oğuzhan Arslan
Project: Plan van Aanpak
3D Modellen gevisualiseerd met de Microsoft HoloLens
Periode: Februari – Juli 2017
Onderwijsinstelling: De Haagse Hogeschool
Opdrachtgever: Quintor

1

DE HAAGSE HOGESCHOOL

INHOUDSOPGAVE

DE HAAGSE HOGESCHOOL

1 INLEIDING

De bouw kent vele vormen, zoals renovatie, nieuwbouw, reconstructie, etc. Hierbij heb je vele hinderingen binnen het gebouw die niet met het blote oog te zien zijn, denk bijvoorbeeld aan de kabels en leidingen die achter de muren lopen. Het kan natuurlijk vervelend zijn als er per ongeluk een leiding geraakt wordt, waardoor je een lekkage krijgt. Hierbij is de vraag hoe kan je ervoor zorgen dat deze "onzichtbare kenmerken" vooraf zichtbaar worden.

Een oplossing kunnen hologrammen zijn. Hiermee zou je op de locaties waar leidingen lopen een hologram kunnen tonen, waardoor het duidelijk wordt waar de leiding precies loopt. Momenteel is de Microsoft HoloLens de enigste oplossing hiervoor, aangezien het hologrammen in de echte wereld kan tonen.

De Microsoft HoloLens (Microsoft, n.d.) is een nieuwe techniek, wat nog niet toegankelijk is voor de consument. Het is nog volop in ontwikkeling, maar wat zijn nou de toepassingsmogelijkheden ervan en is momenteel alleen beschikbaar voor ontwikkelaars. De techniek valt onder Mixed Reality en houd in dat er in de echte wereld objecten getoond kunnen worden die niet echt zijn. De Microsoft HoloLens is op dit moment de enigste techniek die dit mogelijk maakt.

2 AANLEIDING EN CONTEXT

Quintor is een toonaangevend bedrijf op het gebied van Agile Software Development, Enterprise Java/ .NET technologie en mobile development. Tijdens de oprichting in 2005 waren er 4 medewerkers en er is sindsdien een gezonde groei gemaakt. Er zijn ondertussen vestigingen in Groningen, Amersfoort en Den Haag met meer dan 100 medewerkers in dienst

Verder beschikt Quintor over een Software Factory waarin inhouse projecten voor klanten worden uitgevoerd. In deze Software Factory wordt er geprobeerd altijd gebruik te maken van de nieuwste technieken. Daarom wordt er bij Quintor ook onderzocht hoe nieuwe technieken werken en waar ze ingezet kunnen worden.

Een van deze nieuwe technieken is de Microsoft HoloLens en daarom wordt er bij Quintor verschillende toepassingsmogelijkheden onderzocht die hier verduidelijking in zouden kunnen brengen.

3 OPDRACHTGEVER

De opdrachtgever, Quintor, is constant op zoek naar nieuwe technieken en hierbij willen ze weten wat de toepassingsmogelijkheden van deze nieuwe technieken zijn. Hierdoor wordt er veel onderzoek binnen Quintor gedaan. Een van deze nieuwe technieken is de Microsoft HoloLens en de vraag is dan waar deze nieuwe techniek ingezet zou kunnen worden.

DE HAAGSE HOGESCHOOL

4 PROBLEEMANALYSE EN PROBLEEMSTELLING

Een probleem bij de reconstructie van gebouwen, zijn de "onzichtbare attributen" zoals leidingen en kabels die achter de muren lopen. Omdat deze niet zichtbaar zijn, kan het soms moeilijk zijn om te bepalen waar er bijvoorbeeld geboord mag worden. Hierom zal een applicatie ontwikkeld moeten worden, dat het mogelijk maakt deze attributen voor de ogen van gebruikers te projecteren.

Momenteel kunnen bouwvakkers verschillende hulpmiddelen gebruiken (Jackson, n.d.) om te achterhalen waar bijvoorbeeld een kabel loopt. Het probleem bij deze hulpmiddelen is dat ze met de hand de gehele kamer moeten scannen om een inzicht te krijgen van wat waar loopt. Vervolgens hebben ze bijvoorbeeld alleen een inzicht van één type attribuut, terwijl er nog vele andere soorten kunnen zijn.

Hiervoor zal gekeken worden naar de Microsoft HoloLens, aangezien dit een goede oplossing is voor het probleem. Omdat de HoloLens nog nieuw is, is het nog niet duidelijk in welke scenario's de HoloLens precies ingezet kan worden. Door middel van de opdracht wordt er gehoopt dit te verduidelijken.

Om dit werkend te krijgen zal er gekeken worden naar bestaande modellen die hierbij zouden kunnen helpen. Dit kunnen bijvoorbeeld CAD-modellen zijn, die al gebruikt worden in de bouw.

DE HAAGSE HOGESCHOOL

5 DOELSTELLING EN EINDRESULTAAT

5.1 DOELSTELLING

De opdracht is het ontwikkelen van programmatuur voor de Microsoft HoloLens, waarbij een 3D model gekoppeld kan worden aan de ruimte waar de gebruiker zich op dat moment bevindt.

Tijdens de ontwikkeling kan er verwacht worden dat de documentatie over de HoloLens schaars zal zijn, aangezien het nog vrij nieuwe technologie is. Een doelstelling van opdracht is dus onderzoeken welke tools en methoden ingezet kunnen worden tijdens de ontwikkeling voor de HoloLens. Een voorbeeld hiervan is de omgeving om in te renderen, bijvoorbeeld Unity3D of een ander ondersteund 3D Engine.

De applicatie die opgeleverd zal worden zal de gebruiker de optie geven een model te selecteren van het gebouw waar de gebruiker zich op dat moment bevindt. Vervolgens kan de applicatie, door middel van de HoloLens, de afmetingen van de kamer waar de gebruiker zich op dat moment bevindt berekenen. Waarmee vervolgens de juiste kamer in het model gevonden moeten worden. Als de kamer gevonden is, kan de gebruiker kiezen uit de verschillende lagen die bij de kamer horen, waardoor de gebruiker bijvoorbeeld de leidingen die achter de muren lopen kan zien of bepaalde kenmerken, zoals stopcontacten.

5.2 EINDRESULTAAT

Het eindresultaat is een applicatie voor de Microsoft HoloLens, waarmee een 3D model van een gebouw uitgelezen kan worden en de juiste kamer voor de gebruiker geprojecteerd kan worden. Hiermee kunnen bijvoorbeeld de kabels achter de muren getoond worden. Hierbij moeten ook de volgende deelvragen beantwoord zijn:

- Welke modellen voldoen aan de requirements?
- Hoe moeten modellen gestructureerd zijn?
- Hoe kunnen modellen ingeladen worden?
- Hoe nauwkeurig zijn de metingen?

Door middel van de bovenste deelvragen zal er een conclusie getrokken worden op de volgende hoofdvraag:

Hoe en welke modellen kunnen gebruikt worden om kamer kenmerken te weergeven in de Microsoft HoloLens?

DE HAAGSE HOGESCHOOL

6 RISICOANALYSE

Het kan voorkomen dat het niet mogelijk is modellen van gebouwen te verkrijgen of dat de verkregen modellen niet voldoen aan de eisen. Ook kan het zijn dat er geen beschikbare hulpmiddelen zijn om de opdracht te voldoen. Gezien de grootte en veelzijdigheid van de opdracht is het niet de bedoeling alle aspecten van de opdracht zelf te maken. In het geval dat een van deze risico's voorkomen zal er met de opdrachtgever besproken worden hoe de opdracht aangepast kan worden om deze problemen op te lossen.

Dit zijn echter een van de meerdere risico's die kunnen voorkomen tijdens het project. Aangezien het een nieuwe techniek is, is het nog niet duidelijk wat wel en niet mogelijk is. Dit is dan ook de reden dat de requirements van het project niet vast zullen staan en aangepast kunnen worden aan de mogelijkheden van de gebruikte technieken.

Naast deze ontwikkel risico's zijn er ook materiele risico's, zoals de gebruikte hardware. De HoloLens die gebruikt zal worden is dan de belangrijkste hardware die gebruikt zal worden. De HoloLens is te emuleren, maar deze heeft niet alle mogelijkheden die de HoloLens ook heeft. Daarom neemt de HoloLens risico's met zich mee die niet echt te dekken zijn.

7 AANPAK

Het project zal worden gepland over een periode van 16 weken. Hierbij zal er gebruik gemaakt worden van scrum en zal er dus een backlog opgesteld worden van sprints van 2 weken. Aan het einde van elke sprint zal er met de producteigenaar besproken worden wat er in de laatste sprint gedaan is, wat er de komende sprint gedaan zal worden en wat voor problemen er zijn voorgekomen. Ook zal er geprobeerd worden een demo te geven van het resultaat zodat er zoveel mogelijk feedback verkregen kan worden waarmee het uiteindelijke product verbeterd kan worden.

7.1 ONDERZOEK

Aangezien de gebruikte technieken nog nieuw zijn zal er niet veel literatuur over te vinden zijn en de bronnen die gevonden zullen worden zullen eenzijdig of niet betrouwbaar zijn. Hierdoor is er gekozen voor een kwalitatieve onderzoek en aangezien het nog niet duidelijk is welke technieken precies gebruikt zullen worden zal er ook exploratief onderzocht worden, zodat technieken die tijdens de onderzoek gevonden worden ook opgenomen kunnen worden in het onderzoek.

Uiteindelijk zullen er conclusies getrokken worden uit de resultaten die, tijdens de toepassing van de technieken, naar voren zijn gekomen. Hiermee zal vervolgens een vergelijking gemaakt kunnen worden, waarmee de beste keuzes aangeraden kunnen worden aan de opdrachtgever.

DE HAAGSE HOGESCHOOL

7.2 PLANNING

Voor het hele project is er dus ruimte voor 8 sprints van 2 weken. In deze sprints moeten de volgende taken uitgevoerd worden:

- Onderzoek te gebruiken methodieken en tools
- Systeemarchitectuur
- Applicatie bouwen
- Oplevering van verschillende (tussen)producten

Aan het einde van het project wordt gehoopt de volgende producten op te leveren:

- Werkende Applicatie
- Handleiding Applicatie
- Documentatie Applicatie
- Onderzoeksrapport 3D Modellen
- Onderzoeksrapport 3D Engines
- Onderzoeksrapport koppeling van data

Om dit allemaal in 8 sprints te doen is er vooraf een globale planning gemaakt, zodat er bijgehouden kan worden of het project op traject loopt.

Sprint:	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**
Onderzoek 3D engine HoloLens*	X	X						
Onderzoek Model uitlezen .NET*	X	X						
Onderzoek koppeling van data*	X	X						
Architectuur te gebruiken modellen		X	X					
Architectuur 3D engine systeem			X	X				
Architectuur koppelen van systemen				X	X			
Bouwen Systeem					X	X	X	
Testen & Verbetering							X	X
Schrijven documentatie en handleiding							X	X
Schrijven Afstudeerdossier	X	X	X	X	X	X	X	X

** Inclusief schrijven van onderzoeksrapport*

DE HAAGSE HOGESCHOOL

8 COMPETENTIES

Tijdens het project zullen er een aantal competenties uitgevoerd worden. Deze zullen vervolgens ook verantwoord moeten worden in de opgeleverde producten. Hieronder zullen deze competenties aangegeven worden en in welk document de verantwoording afgelegd zal worden.

1.1 Selecteren methoden, technieken en tools
1.2 Selecteren van standaardsoftware
3.1 Ontwerpen softwarearchitectuur
3.2 Ontwerpen systeemdeel
3.3 Bouwen applicatie

	1.1	1.2	3.1	3.2	3.3
Onderzoeksrapport 3D Engines	■	■			
Onderzoeksrapport 3D modellen uitlezen met .NET	■	■			
Onderzoeksrapport koppelen van data	■	■			
Documentatie werking applicatie			■	■	■
Documentatie applicatie			■	■	■

Voortgang

Op het moment van schrijven van het voortgangsverslag ben ik op ongeveer 40% van het afstuderen. In deze periode heb ik een groot deel van mijn onderzoeken en architectuur afgerond. Qua documentatie moeten de onderzoeksrapporten afgerond worden en toegevoegd worden aan de uiteindelijke afstudeerdossier. Zoals besproken bij de bedrijfsbezoek ben ik van plan de volledige onderzoeksrapporten toe te voegen in de bijlagen en in de afstudeerdossier zelf een korte herhaling te schrijven over de uitgevoerde onderzoek(en).

De applicatie zelf was zelf ook al voor een groot gedeelte afgerond en dit was het resultaat van de uitvoering van de verschillende onderzoeken. Hierbij was er alleen geen structuur meer terug te vinden en is het nu dus de bedoeling de applicatie te schrijven op de structuur die uitgedacht is bij de architectuur van de applicatie. Dit is niks meer dan het combineren van bestaande code en de nieuwe juiste structuur.

Problemen

Tijdens de onderzoeken zijn meerdere problemen naar voren gekomen die het bereiken van het doel verhinderden. De eerste grote hindering was dat CAD modellen niet geschikt waren voor het project. Dit was in het afstudeerplan opgenomen als de modellen die uitgelezen moeten worden, maar deze modellen bevat de data niet die nodig is om te tonen. Hierdoor is er met de opdrachtgever gesproken en was de term CAD modellen eigenlijk te specifiek en mocht ik gewoon kijken naar geschikte modellen om het uit te voeren. Hierbij was het vooral belangrijk dat er door middel van de Microsoft HoloLens een model gekoppeld kan worden aan de echte wereld. Hier vanuit heb ik uiteindelijk een onderzoek gedaan naar een aantal verschillende modellen die geschikt zouden kunnen zijn en wat de voor en nadelen zijn van deze verschillende modellen. Uiteindelijk heb ik hier een model uit gekozen waar ik mee verder ben gaan werken.

De modellen die onderzocht waren hadden allemaal hun eigen problemen, maar de meeste problemen waren wel oplossingen voor te vinden. De grootste probleem was dat de modellen dynamisch ingeladen moeten worden en dit was alleen mogelijk met 1 van de onderzochte modellen en dit model was dan ook de model die de meeste problemen had. Deze problemen en oplossingen zijn beschreven in de onderzoeksverslag.

Nog een ander probleem was dat de HoloLens geen filesystem heeft en dus geen logische pad heeft naar een model. Hierom is er een webapplicatie geschreven waar modellen opgeslagen kunnen worden. Deze modellen worden vervolgens via een restapi opgehaald en gebruikt. Deze webapplicatie zal dan ook kort beschreven worden in de afstudeerdossier.

Te doen

Op ongeveer 60% is de bespreking van het concept afstuderen. Mijn bedoeling is dat ik voor de bespreking het structuur van het verslag heb staan en de onderzoeken en architectuur van het project voor 95% heb afgerond. Hierna heb ik ongeveer 2 weken om de documentatie over de applicatie zelf te schrijven en het verslag grotendeels afgerond te hebben voor de tussentijdse assessment.

CONCEPT BESPREKING

Bespreking concept	Tussentijds assessment	Eerste beoordeling

Formulier bespreking concept afstudeerdossier

Student: Oguzhan Arslan **Studentnummer:** 13018604

Datum: 20-4-2017

Tijdens de bespreking is het volgende geconstateerd:		ja	nee
a	*Het voortgangsverslag is ontvangen*	X	
b	*Het afstudeerdossier is digitaal beschikbaar*	X	
c	*Het afstudeerdossier is opgebouwd conform de richtlijnen*	X	
d	*Het goedgekeurde afstudeerplan is aanwezig*	X	
e	*Het plan van aanpak is aanwezig*	X	
f	*Reeds geleverd commentaar is aanwezig*	X	
g	*Het afstudeerdossier geeft voldoende inzicht in de stand van zaken*	X	
h	*De afstudeeropdracht is tot nu toe naar behoren uitgevoerd*	X	

Verbeterpunten:

Vermeld in het afstudeerverslag ook de resultaten van de demo's en hoe dit van invloed is geweest op de product backlog.

Opmerkingen:

Afspraak voor tta: maandag 15 mei, 10:00

Nam begeleidend examinator:

Datum: J.J. van der Hoek

Dit formulier wordt door de begeleidend examinator digitaal ingevuld en per email naar de student verstuurd met een cc naar de coördinator van ICT & Media @ Work (A.M.Schipper@hhs.nl). Het formulier dient door de student te worden opgenomen in het afstudeerdossier.

TUSSENTIJDS ASSESSMENT

	Formulier Tussentijds assessment afstudeerdossier		

Student:	**Oguzhan Arslan**	Onderwerp:	HoloLens gekoppeld aan 3D CAD-modellen	
Studentnummer:	13018604	Datum:	**15-05-2017**	Ingevuld door: Okan Zor
Opleiding:	Informatica			

Aanpak	O	T	V	G	Toelichting per beoordelingscriterium
Passend		**x**			*Werkelijke aanpak is nog te weinig beschreven.*
Theoretisch verantwoord		**x**			*De aanpak zoals gepland komt niet overeen met de overeenkomstige theorie. Scrum is onvoldoende terug te zien in de aanpak. Alle requirements worden vooraf tot in detail achterhaald.*
Samenhang uitvoering beroepstaken		x			*De samenhang tussen de requirements en het onderzoek is onvoldoende gelegd.*

Beroepstaken op afgesproken niveau uitgevoerd?		O	T	V	G	**Toelichting per beoordelingscriterium**
1.1	Selecteren methoden, technieken en tools		x			**De onderbouwing van de gekozen methoden , technieken en tools is niet helder. De student heeft SCRUM toegepast, maar het blijft onduidelijk waarom dit nou een methodiek is die het beste past bij dit project. Overigens is het raar dat de student allerlei tools gebruikt en pas achteraf beschrijft welke tools hij gaat gebruiken tijdens het afstudeerproces.**
1.2	Voorbereiden en opstarten software-ontwikkeltraject		x			**De student heeft een onderzoek uitgevoerd om zodoende te achterhalen wat er ontwikkeld gaat worden. De opgestelde eisen aan dit onderzoek is in verslag niet voldoende beargumenteerd. Het lijkt wel alsof de student in zijn hoofd meer criteria heeft zitten dan dat dit op papier is vastgelegd. Hiernaast zijn de gekozen onderzoeksvragen ook erg overlappend waardoor het niet te achterhalen is welke vragen nou precies zijn beantwoord.**
3.1	Ontwerpen softwarearchitectuur		x			**Uit de opgestelde diagrammen is het niet traceerbaar hoe de requirements zich nou verhouden op de aangedragen architectuuroplossing.**
3.2	Ontwerpen systeemdeel		x			**Zie commentaar hierboven.**
3.3	Bouwen applicatie		x			**We zien wel dat de student heeft gebouwd, echter of dit conform de requirements en dus ook architectuur is, uitgevoerd is niet helder.**

Producten	O	T	V	G	Toelichting per beoordelingscriterium
Tussenproducten		x			Deze komen nog niet voldoende terug in het eindverslag.
Eindproducten		x			*Rapport geeft geen indicatie.*

Effectief communiceren	O	T	V	G	Toelichting per beoordelingscriterium
Binnen afstudeerbedrijf		x			Nog geen evaluatieformulier ontvangen van de opdrachtgever
Afstudeerdossier		x			De structuur van het verslag moet verbeterd worden. De beargumentatie van het uitgevoerd werk moet beter beschreven worden.

Reflectie	O	T	V	G	Toelichting per beoordelingscriterium
Inzicht in eigen functioneren	x				Ontbreekt.
Inzicht in eigen leerproces	x				Ontbreekt.

Resultaat TTA

Advies		**Inleverdatum**	**Besluit student**		**Inleverdatum**
x	**Inleveren (bindend advies**	**2 juni 2017**		**Afstudeerdossier wordt ingeleverd op:**	**2 juni 2017**
	Verlengen (vrijblijvend advies)			**Afstudeerperiode wordt verlengd en afstudeerdossier wordt ingeleverd op:**	
	Stoppen (vrijblijvend advies)			**Student stopt met afstudeeropdracht**	

Nb. Gebruik Resultaat TTA OF Resultaat Eerste beoordeling. Wat niet gebruikt wordt svp weghalen.

Resultaat Eerste beoordeling