HAUTE TECHNIQUE

Ontwikkelen van een applicatie voor het aansturen van DMX apparaten aan de hand van locatie updates

Afstudeerverslag

Student : Jan de Boer
Studentnummer : 11071133
Onderwijsinstelling : Haagse Hogeschool te Delft
Opleiding : Technische Informatica
Afstudeerperiode : 1 september 2015 t/m 8 januari 2016
Begeleider : Dhr. A. Andrioli
Tweede examinator : Dhr. J. van Peski

Bedrijf : HAUTE TECHNIQUE
Bedrijfsmentor : Sander ter Braak

Datum : 7 januari 2016
Versie : V1.0

Versiebeheer

Versie	Datum	Opmerking
0.1	03-09-2015	Eerste concept versie opgezet.
0.2	10-10-2015	Opmerkingen van de heer Andrioli verwerkt.
0.3	15-12-2015	Opmerkingen van tussentijds assessment verwerkt.
1.0	06-01-2016	Verslag tot de laatste versie bijgewerkt.

Referaat

Jan de Boer, “Ontwikkelen van een applicatie voor het aansturen van DMX apparaten aan de hand van locatie updates”. Afstudeerverslag opleiding Technische Informatica, Haagse Hogeschool te Delft, 2016

Dit verslag beschrijft de afstudeeropdracht die naar aanleiding van het afronden van de opleiding Technische Informatica te Delft is uitgevoerd. De afstudeerperiode betrof zeventien weken en liep van 1 september 2015 tot en met 8 januari 2016. De opdracht is uitgevoerd bij Haute Technique te Utrecht.

Haute Technique heeft een software applicatie ontwikkeld voor een bekende Nederlandse DJ. Met deze applicatie is de DJ in staat om lichten en of visuals aan de hand van zijn armbewegingen aan te sturen. Haute Technique heeft de wens om dit concept verder uit te breiden met locatiegevoeligheid. De afstudeeropdracht zal zich richten op het verwezenlijken van dit concept.

Tijdens het afstudeertraject wordt er gewerkt met de ontwikkelmethodiek Scrum. Tijdens de sprints is onderzoek gedaan naar verschillende technieken en frameworks. De applicatie biedt een uitbreidbare basis voor het aansturen van DMX apparaten volgens een bepaald effect. Hierbij is er een scripting interface ontwikkeld die gebruikers in staat stelt om zelf effecten voor de applicatie te ontwikkelen. Voor het weergeven en aansturen van de verschillende apparaten maakt de applicatie gebruik van de Unity3D engine.

Descriptoren

- *Softwareontwikkeling*
- *Model View Presenter*
- *User interface*
- *Volglampen*
- *Scripting*
- *Unity3D*
- *Locatie*
- *Scrum*
- *UML*
- *DMX*
- *DJ*
- *C#*

Voorwoord

Ter afsluiting van de studie Technische Informatica aan de Haagse Hogeschool te Delft heb ik een afstudeeropdracht uitgevoerd. Deze opdracht is uitgevoerd bij het bedrijf Haute Technique te Utrecht in de periode van 1 september 2015 tot en met 8 januari 2016. Dit document beschrijft het doorlopen proces en de verschillende keuzes die gemaakt zijn tijdens het traject. Met dit document wil ik aantonen dat ik werkzaamheden kan verrichten op HBO niveau.

Bij het lezen van dit document is enige kennis vereist over softwareontwikkeling en de verschillende zaken die hierbij komen kijken zoals programmeertalen en ontwerptechnieken zoals UML. Een verklarende woordenlijst is te vinden aan het einde van dit document

Ik wil graag mijn collega's bij Haute Technique bedanken voor een zeer leuke en leerzame tijd. In het bijzonder wil ik Sander ter Braak bedanken voor het beschikbaar stellen van een afstudeerplek en de begeleiding tijdens het project. Ook wil ik de heer Andrioli en de heer Peski bedanken voor het begeleiden vanuit de opleiding en de feedback op mijn verslag. Tot slot wil ik ook Niels de Boer en Maartje de Boer bedanken voor het 'nalopen' van mijn verslag.

Jan de Boer,
Stompwijk, 7 januari 2016

Samenvatting

Als afronding van studie Technische Informatica aan de Haagse Hogeschool te Delft is er een afstudeeropdracht uitgevoerd. In een periode van zeventien weken is gekomen tot het resultaat dat in dit verslag beschreven staat. De afstudeeropdracht is uitgevoerd bij het bedrijf Haute Technique te Utrecht door afstudeerder Jan de Boer.

Haute Technique heeft een software applicatie ontwikkeld voor de Nederlandse DJ Armin van Buuren. Met deze applicatie is Armin van Buuren in staat om lichten en of visuals aan de hand van zijn arm bewegingen aan te sturen. Haute Technique heeft de wens om dit concept verder uit te breiden met locatiegevoeligheid. Tijdens de afstudeerperiode is er gewerkt aan het verwezenlijken van dit concept.

Voordat de afstudeerperiode begon was de afstudeeropdracht beschreven in het afstudeerplan. Bij aanvang van het project bleek echter dat de opdracht voor een groot deel was veranderd. De eerste week van het project is dan ook besteed aan het formuleren van een nieuwe opdracht samen met de opdrachtgever en het maken van een plan van aanpak. Er is gekozen om tijdens dit project gebruik te maken van de ontwikkelmethodiek Scrum. In totaal zijn er negen sprints geweest die zich elk op een onderdeel van het systeem focuste.

In sprint één is er begonnen met het achterhalen van de eisen van het systeem. Tevens is er onderzoek gedaan naar de verschillende technieken die tijdens het project gebruikt gaan worden. Hieruit kwam naar voren dat er meer onderzoek nodig was naar het te gebruiken framework om de applicatie in te ontwikkelen. Dit onderzoek heeft plaats gevonden in sprint twee. Er zijn verschillende frameworks onderzocht en tegen elkaar afgewogen, uiteindelijk is er gekozen om de applicatie te ontwikkelen in samenwerking met de Unity Engine. In sprint twee is er ook een basis gelegd voor de architectuur van de applicatie die in verdere sprints uitgebreid kan worden. De derde sprint focuste zich op het implementeren van het effectgedeelte van de applicatie. Er is gekozen om C# als scripttaal in te zetten voor het schrijven van effecten. Om het ontwikkelen van de user interface te versimpelen is er in sprint vier een keuze gemaakt voor het MarkUX framework. Hiernaast vond er deze sprint een testdag plaats waarbij in samenwerking met Sendrato de applicatie en de hardware werden getest. De resultaten van de testdag zijn in sprint vijf verder verwerkt. In sprint zes is er een eerste versie van de user interface ontwikkeld, hierbij werd gebruik gemaakt van het MarkUX framework. De hierop volgende sprint focuste zich op het makkelijker aanpasbaar maken van de positie en rotatie van een lamp door de gebruiker. Dit kostte meer tijd dan verwacht waardoor een aantal taken doorschoven naar sprint acht. Hiernaast focuste sprint 8 zich op het loskoppelen van de user interface laag met de business logica. De laatste sprint focuste zich op het overdragen van het project.

Tijdens het project is er een uitbreidbare basis gelegd voor het aansturen van lampen volgens een bepaald effect. Er kan dan ook gezegd worden dat het project succesvol is verlopen. De applicatie is echter nog niet in zijn eindversie, vooral de user interface vergt nog verder werk. Door de verschillende documenten die geschreven zijn is het software project goed overdraagbaar.

Inhoudsopgave

Figuren

Tabellen

1 Inleiding

Dit afstudeerverslag is één van de resultaten van de afstudeeropdracht die ter afsluiting van de studie Technische Informatica te Delft is uitgevoerd. Deze afstudeeropdracht heeft plaatsgevonden in een periode van twintig weken bij het bedrijf Haute Technique te Utrecht. Dit verslag beschrijft hoe het project is verlopen en welke keuzen er gaandeweg het project zijn gemaakt.

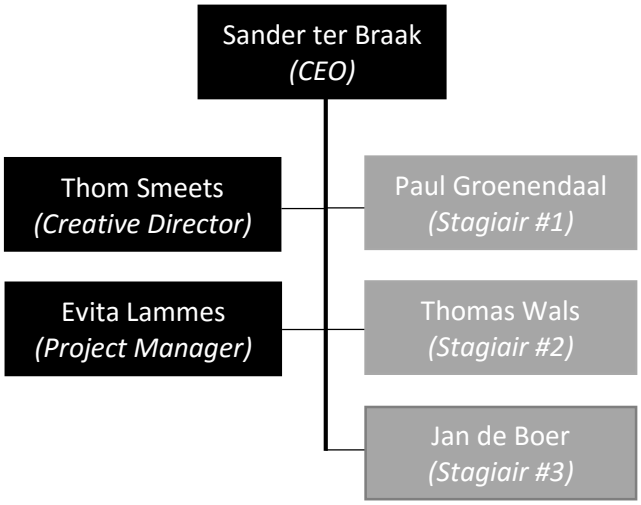
Haute Technique heeft een applicatie ontwikkeld voor de Nederlandse DJ Armin van Buuren. Met deze applicatie is Armin van Buuren in staat om lichten en of visuals aan de hand van zijn armbewegingen aan te sturen. Hierdoor kunnen de shows interactiever worden gemaakt. Haute Technique heeft de wens om dit concept verder uit te breiden met locatiegevoeligheid, waardoor het mogelijk zou moeten worden om lampen en of visuals aan te sturen aan de hand van de positie van een persoon. Er is al een prototype applicatie ontwikkeld die aantoont dat het mogelijk moet zijn. Tijdens deze afstudeeropdracht wordt er een basis gelegd aan een applicatie die dit concept verder implementeert.

Dit verslag is op te delen in grof weg drie onderdelen. Het eerste deel van het verslag beschrijft de aanpak van het project, hierbij wordt ook het bedrijf Haute Technique en de opdracht verder beschreven. Het tweede onderdeel beschrijft in chronologische volgorde het verloop van het project en de belangrijkste keuzen die tijdens dit traject zijn gemaakt. De structuur van dit onderdeel is sterk beïnvloed door de gekozen ontwikkelmethodiek Scrum. Het laatste onderdeel van het verslag blikt terug op de verloop van het project. Hiernaast wordt er een conclusie getrokken en worden er verschillende aanbevelingen voor een vervolgtraject gedaan.

Tijdens dit project zijn er nog tal van andere documenten geschreven. Deze zijn te vinden in het externe bijlageboek. In sommige stukken van het verslag wordt er naar een specifieke bijlagen verwezen.

2 Haute Technique

De afstudeeropdracht wordt uitgevoerd bij het bedrijf Haute Technique. Haute Technique is een jong bedrijf dat zich bezighoudt met de zogenoemde nieuwe media en werkt op project basis. De focus van Haute Technique ligt in het creëren van unieke en passende oplossingen voor zijn klanten. Zo hebben ze ervaring met het maken van games, interactieve installaties, aanraakgevoelige tafels en verschillende andere zaken. Haute Technique bestaat uit een vast team van drie personen. Hiernaast worden afhankelijk per project verschillende freelancers ingehuurd. Zie onderstaand figuur voor het organogram.



Figuur 1 Organogram

Haute Technique tekent zich als een jong en dynamisch bedrijf. Doordat het bedrijf klein is verloopt de communicatie vaak direct ook kent het bedrijf geen afdelingen zoals grotere bedrijven.

Een van de projecten die Haute Technique heeft lopen is een project samen met Armin van Buuren. In dit project helpt Haute Technique Armin van Buuren om zijn shows interactiever te maken. Hierbij maakt Haute Technique gebruikt van de Myo [1]. De Myo is een Bluetooth armband die de positie van de armen kan volgen en deze doorstuurt via bluetooth. Haute Technique heeft een software systeem ontwikkeld die de Myo kan uitlezen en aan de hand van deze data lampen en of video beelden kan aansturen. Deze techniek wordt op dit moment toegepast bij de sets van Armin van Buuren en wordt goed ontvangen door het publiek.

Samen met Sendrato [2] wil Haute Technique dit systeem naar het volgende niveau tillen door het uit te bereiden met locatie gevoeligheid. Sendrato heeft veel expertise op het gebiedt van locatie bepaling van grote groepen mensen. Zo leveren ze het toegangssysteem van een aantal grote festivals. Met dit toegangssysteem wordt het mogelijk om de locatie van iedere bezoeker te volgen.

HAUTE TECHNIQUE SENDRATO

3 Opdrachtomschrijving

3.1 Probleemstelling

Het door Haute Technique ontworpen systeem om de shows van Armin van Buuren interactiever te maken is op dit moment alleen nog maar in te zetten als de locatie van Armin tijdens de show niet veranderd. Dit omdat een aantal van de visuele effecten die gebruikt worden afhankelijk zijn van de positie van Armin van Buuren. Daarom is het gewenst om het systeem zodanig uit te breiden zodat er kan worden gereageerd op de positie van Armin van Buuren. Dit nieuwe systeem kan in de toekomst ook zorgen voor nieuwe elementen in de show waardoor de shows steeds 'interactiever' kan worden.

3.2 Doelstelling

De doelstelling van dit project is om een software applicatie te ontwikkelen die gebruikmakend van de hardware van Sendrato lampen en of visuals laat reageren op de positie veranderingen van een persoon of personen. Veder zijn aan het einde van het project tests uitgevoerd om de werking van de hardware te valideren.

3.3 Opdrachtomschrijving

Aan het begin van het afstuderen was de opdracht als volgt geformuleerd: het ontwikkelen van een softwaresysteem dat gebruikmakend van de sensoren die Sendrato levert de locatie van een persoon op een podium kan berekenen. Deze locatie gegevens zouden vervolgens via een gebruiksvriendelijke interface worden gedeeld met andere applicaties. Ook zou er een applicatie worden ontwikkeld die gebruikmakend van de interface die de ontwikkelde applicatie aanbood een volglicht kon aansturen waardoor dit volglicht een persoon automatisch kon volgen. Zie BIJLAGEN A voor verdere informatie over de originele afstudeeropdracht.

Echter bij aanvang van het project bleek dat een groot deel van deze opdracht al uitgevoerd was. Haute Technique en Sendrato hadden samen een testdag gehad en deze was veel belovend verlopen. Ze kregen het voor elkaar om een volglicht een persoon te laten volgen. Hierbij droeg de persoon een tracking device bij zich. Een groot deel van de afstudeeropdracht was toen al voltooid. Haute Technique had namelijk een prototype applicatie geschreven die de sensoren kon uitlezen en vervolgens een volglamp kon aansturen. De moeilijkheidsgraad van de initiële afstudeeropdracht is aan het begin verkeerd ingeschat. Dit kwam mede doordat Sendrato zijn werk beter deed dan verwacht. In eerste instantie was het plan dat de te schrijven applicatie de sensor data nog moest omzetten naar bruikbare waarden dit was echter niet nodig aangezien Sendrato dit al deed. Het systeem van Sendrato leverde dus al een interface waarmee de *X*, *Y* en *Z* waarden van een persoon konden worden ontvangen. Het was vervolgens een kleine moeite om deze data uit te lezen en daarmee een volglamp aan te sturen. Een andere reden dat het initiële project moeilijker werd ingeschat dan het was kwam doordat de afstudeerder zeer weinig ervaring en kennis heeft over de technieken die voor het project worden gebruikt.

De prototype applicatie die ontwikkeld was voor de testdag is echter verre van af. Hier moet nog tijd en onderzoek in worden gestoken om de applicatie bruikbaar te maken. Dit is dus ook waar het project zich op gaat focussen. Hierbij komen hele nieuwe problemen naar boven. Zoals: hoe worden de volglampen aan het systeem toegevoegd en gekalibreerd? Hoe kunnen meerdere personen gevolgd worden? Hoe reageren de volglampen op locatie veranderingen? Wat wordt er gedaan met de visuals, hoe worden deze locatie gevoelig gemaakt? Wat wordt de rol van de applicatie binnen de shows van bijvoorbeeld Armin van Buuren? Hiernaast zullen tijdens de loop van het project ook een hoop nieuwe vragen ontstaan. Naast het antwoord vinden op deze vragen moet ook het systeem dat Sendrato levert getest worden.

3.4 Rol van Sendrato

Haute Technique heeft zelf niet de know how in huis om hardware te produceren die het mogelijk maakt om personen te tracken. Hiervoor werkt Haute Technique samen met Sendrato. Sendrato levert binnen dit project de hardware. Er worden waarschijnlijk een aantal testdagen georganiseerd waar bij Haute Technique en Sendrato het systeem dat Sendrato levert zullen testen. Op deze testdagen kan Haute Technique feedback geven over het systeem. Sendrato heeft verder geen inspraak of inzage in het verloop van het project.

3.5 Projectgrenzen

Tijdens dit project worden niet de visuals of lampschema's bedacht en gemaakt die gebruikt kunnen worden voor de shows. Er is echter wel kennis nodig over de werking zulke schema's zodat de applicatie op de juiste manier kan worden ontwikkeld.

3.6 Op te leveren producten

Tijdens en aan het einde van dit afstudeerproject worden verschillende producten aan de opdrachtgever geleverd. Deze op te leveren producten zijn tot stand gekomen in samenspraak met de opdrachtgever en met de afstudeerbegeleider.

- Een onderzoeksrapport met daarin de verschillende onderzoeken die tijdens het project zijn uitgevoerd. Zoals bijvoorbeeld de analyse naar het kalibreren van de volglampen.
- Het software systeem met de daarbij horende source code van de software.
- Documentatie die het software systeem beschrijft hieronder vallen de volgende documenten: requirements document ontwerpdocument met daarin de UML diagrammen die het systeem beschrijven, een testrapport en een handleiding van het systeem.
- Een testrapport met betrekking op het systeem dat Sendrato levert.

3.7 Organisatie

Dit project dient als afstudeeropdracht voor de afstudeerder Jan de boer. Hierbij zijn vanuit de opleiding twee docenten aangesteld namelijk: de heer A. Andrioli en de heer J. van Peski hier heeft de heer Peski de rol van tweede beoordelaar en de heer Andrioli die van begeleider.

Vanuit Haute Techniek is Sander ter Braak de opdrachtgever en tevens de begeleider van de afstudeerder. Tijdens de loop van de afstudeer periode zijn een aantal contactpunten met de opleiding ingepland om de voortgang van het afstudeerproject door te nemen.

4 Aanpak

In dit hoofdstuk wordt de aanpak van het afstudeer project behandeld. Hierin wordt de keuzen voor de gekozen methodiek verantwoord en worden de gebruikte standaarden verder toegelicht.

4.1 Ontwikkelmethodiek

Het kiezen van de juiste methodiek voor dit project is van belang voor het (op de juiste manier) behalen van het einddoel van de afstudeeropdracht. Er zijn verschillende methodieken die gebruikt kunnen worden bijvoorbeeld: RUP, XP of Scrum. Elke methodiek is in meerdere of in mindere mate geschikt voor dit project. Het is daarom belangrijk om eerst de kenmerken van dit project te beschrijven. Hier volgen de kenmerken van het project:

- Tijdens dit project moeten er verschillende onderzoeken (analyses) worden gedaan.
- Tijdens het project wordt er software ontwikkeld.
- Het precieze einddoel van de te schrijven software staat nog niet vast. Hierdoor kunnen in de loop van het project de eisen veranderen.
- Tijdens dit project moet er documentatie worden opgeleverd.
- Er is sprake van een derde partij die de hardware levert.
- De kennis van de afstudeerder over het probleemdomein is nog miniem.

Vanuit deze kenmerken zijn een aantal eisen opgesteld. Deze eisen zijn terug te vinden in FIGUUR 2. Per methodiek wordt er gekeken hoe deze zich aan de gestelde eis houdt. Vervolgens kan er een gegronde keuze worden gemaakt over een passende methodiek voor dit project. De eisen in onderstaand figuur zijn gesorteerd naar prioriteit. Hierbij staat de eis met de hoogste prioriteit boven in. De lijst met methodieken in FIGUUR 2 is beperkt tot de meest bekende methodieken en of methodieken die al bij de afstudeerder bekend waren.

Eis	Nr.	Waterval	RUP	openUp	Scrum	TDD	XP
Kan solo worden uitgevoerd	E1	+	~	~	~	+	-
Geschikt voor software ontwikkeling	E2	+	+	+	+	+	+
Werkt met iteraties	E3	-	+	+	+	+	+
Eisen kunnen worden aangepast tijdens het project	E4	-	~	~	+	+	+
Kan (goed) overweg met een onduidelijk einddoel	E5	-	~	~	+	+	+
Biedt ruimte voor het uitvoeren van onderzoeken	E6	+	+	+	+[1]	~	~
Heeft plek voor documentatie	E7	+	+	+	~	~	~
Ervaring van de afstudeerder met de methodiek	E8	~	+	~	+	-	-

\+ *methodiek voldoet aan deze eis*
~ *methodiek voldoet deels aan deze eis*
\- *methodiek voldoet niet (of nauwelijks) aan deze eis*

Figuur 2 Methodieken

Notitie (1): In tegenstelling tot RUP en OpenUp biedt SRUM standaard geen plek aan voor onderzoek. Dit is echter prima binnen Scrum te passen zie de bronnen [3] *en* [4]

Kijkend naar FIGUUR 2 valt de keuzen op Scrum [5]. De waterval methode valt af omdat deze methode niet overweg kan met veranderende eisen [6]. Test driven development (TDD) [7] valt af omdat deze methoden vooral handvatten biedt voor het ontwikkelen van de software aan de hand van tests. Scrum biedt naast handvatten voor het ontwikkelen van de software ook tal van handvatten om het proces te beheren. Tevens is het schrijven van tests prima in te passen binnen de Scrum methodiek. Extreme programming (XP) [8] valt af omdat deze niet solo is uit te voeren. XP is namelijk gebaseerd op pair programming hierbij zitten twee programmeurs tegelijkertijd achter de zelfde computer. RUP [9] en OpenUp [10] zouden goede kandidaten zijn alleen kunnen ze minder goed omgaan met een onduidelijk einddoel aangezien de meeste eisen vroeg in het traject worden bepaald. Scrum kan dit beter aangezien Scrum met kortere iteraties (sprints) werkt. Aan het begin van iedere sprint wordt er aandacht geschonken aan veranderend eisen. Scrum biedt echter niet zoals RUP of OpenUp een standaard set van documenten aan, deze moeten binnen Scrum zelf naar gelang het project bepaald worden.

4.2 Scrum

Scrum is een methodiek die valt onder de Agile vlag. Scrum maakt gebruik van sprints van één tot vier weken. In een sprint worden de items van de sprint backlog afgewerkt. De sprint backlog wordt aan het begin van de sprint gevuld met items uit de product backlog tijdens de zogenoemde sprint planning. In de product backlog bevinden zich de verschillende user stories van het project. User stories vormen de eisen waaraan de software die ontwikkeld wordt moet voldoen. Het resultaat van elke sprint is een werkend (of deels werkend) product. Zie FIGUUR 3.

Figuur 3 Scrum proces [11]

Scrum is in eerste instantie bedoeld voor een team van meerdere mensen er is daarom een vertaal slag nodig om Scrum bruikbaar te maken voor een project met een persoon.

- Binnen Scrum is het gebruikelijk om elke dag een zogenoemde daily Scrum te houden. Dit is een meeting waarbij het ontwikkelteam de voortgang van de vorige dag bespreekt en een plan maakt voor de te komen dag. Het doel van deze meeting is het op tijd opsporen van problemen die het halen van de sprint deadline in gevaar brengen. Aangezien dit een klein project is wordt er maar door een persoon aan gewerkt. Er is daarom gekozen om deze meeting niet te houden aangezien er maar een ontwikkelaar aan dit project werkt die dus ook te allen tijde de voortgang van de sprint kent. Op het moment dat er problemen worden gedetecteerd wordt de opdrachtgever zo snel mogelijk op de hoogte gesteld en wordt er samen naar een passende oplossing gezocht.

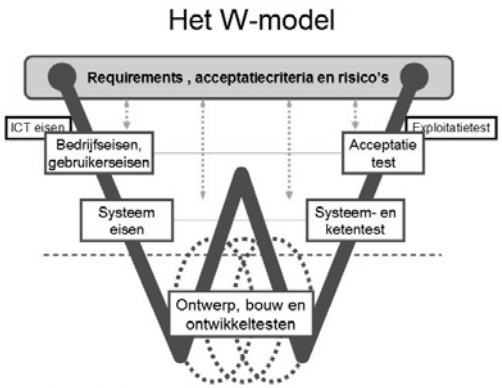
- Scrum definieert verschillende rollen binnen het ontwikkelteam. Aangezien dit een klein project is worden de verschillende rollen samengenomen. Het samennemen van deze rollen leidt in principe tot conflicten maar aangezien dit een klein project is en er maar een persoon aan werkt zijn de risico's beperkt. Om een extra laag van zekerheid toe te voegen is er gekozen om tijdens de sprint planning extra stil te staan bij deze conflicten. Hierdoor zou het geen probleem moeten vormen dat de rollen worden samengenomen.
- Aan het begin van een sprint moeten er bepaald worden welke backlog items er tijdens de sprint worden geïmplementeerd. Samen met de opdrachtgever wordt er bepaald welke backlog items er in de sprint worden geïmplementeerd.

De productbacklog van dit project is te vinden in het scrum rapport (BIJLAGEN F) In dit document wordt per sprint behandeld welke user stories worden geïmplementeerd en welke nieuwe user stories er aan de product backlog zijn toegevoegd. Delen uit dit document zijn terug te vinden in dit verslag.

4.3 Test strategie

Zoals al eerder vermeld werkt Scrum met user stories; wanneer een user story afgerond is hangt af van de definition of done van het Scrum team. De definition of done beschrijft dus wanneer een user story als af kan worden beschouwd. Voor dit project geldt dat de user story af is als hetgeen is geïmplementeerd en getest wat de desbetreffende user story beschrijft. Dit kan nogal verschillen per user story, zo kan het zijn dat dat een user story pas af is als er een bepaald document is geschreven of als er een bepaald stuk software is ontwikkeld en getest. Voor user stories binnen een klein project als dit, is het over het algemeen duidelijk wanneer een user story als af kan worden beschouwd. Dit komt doordat er maar door één persoon actief aan het project wordt gewerkt.

Voordat een user story die een stuk software implementeert als af kan worden gezien moeten er afhankelijk van de user story eerst een aantal tests worden uitgevoerd. Hiervoor wordt de methode SmartTest als naslagwerk gebruikt. SmartTest is een testmethode die veel verschillende testsoorten en documenten beschrijft die kunnen worden gebruikt in het testproces. De SmartTest methode maakt gebruik van het zogenoemde W model voor een iteratieve aanpak van testen, zie hiervoor FIGUUR 4. Aangezien het hier gaat om een klein project, worden maar een aantal van de testsoorten die SmartTest beschrijft gebruikt.



Figuur 4 Het W model volgens SmartTest [12]

Het gaat hier dan voornamelijk om de testsoorten die voorkomen in de groep ontwikkeltesten uit het W model. Deze tests focussen zich voornamelijk op de correcte werking van een klasse of een methode. Hiernaast kan ook de samenwerking tussen verschillende klassen of subsystemen worden getest, dit worden ook wel integratie tests genoemd. Al deze verschillende tests zijn uitgevoerd op het moment dat een user story als af kan worden gezien. Aangezien het hier om een klein project gaat is er gekozen om voor de complexere klassen in het systeem unit tests te ontwikkelen. Het idee is dat op

deze manier de tes basis per sprint groeit. Het voordeel van unit tests is dat deze over het algemeen automatisch kunnen worden uitgevoerd waardoor de werking van het systeem dus snel te valideren is. Voordat een user story als af kan worden gezien moet er ook een systeem test worden uitgevoerd. Over het algemeen is uit de beschrijving van een user story af te leiden wat de systeem test is.

4.4 Risicoanalyse

Door van te voren de risico's van het project te beschrijven en passende maatregelen tegen deze risico's te nemen is de kans dat het project slaagt groter. De risicoanalyse diende ook als belangrijke input voor de planning in het plan van aanpak. De risicolijst is tot stand gekomen door de verschillende kenmerken van het project te onderzoeken. Deze kenmerken zijn reeds beschreven in HOOFDSTUK 4.1. De risico's kunnen worden afgeleid vanuit deze kenmerken. Hiernaast is er ook gekeken naar veel voorkomende problemen binnen projecten zoals miscommunicatie en het uitlopen van het projecten.

In TABEL 1 volgen een aantal van de grootste risico's van het project. De rest van de risico's zijn te vinden in het plan van aanpak (BIJLAGEN B). Deze risico's zijn geprioriteerd door de volgende berekening: elk risico krijgt een kans in procenten en een impact (een waarde tussen een en tien). De prioriteit wordt vervolgens berekend door de kans te vermenigvuldigen met de impact.

Tabel 1 Deel van de risico's uit de risicoanalyse

Nr.	Beschrijving	Kans	Impact	Prioriteit	Maatregel
R1	Niet alle aspecten van de opdracht zijn nog even duidelijk. Zo is er nog niet duidelijk wat de applicatie allemaal moet doen.	100%	8	800	Door de juiste ontwikkel methodiek te kiezen kan het doel stap voor stap achterhaald worden. De ontwikkel methodiek moet flexibel genoeg zijn om in korte iteraties een werkend (deel) systeem op te leveren. Zodat de opdrachtgever snel ziet in welke richting het project gaat en zo nodig kan bijsturen. Ook moet de planning rekening houden met dit feit. Door vroeg in het project de eisen duidelijker te krijgen.
R2	De opdrachtgever heeft een drukke agenda en is mogelijk veel op tour. Waardoor er geen vergaderingen kunnen plaats vinden.	100	6	600	Door van te voren vaste dagen af te spreken om te vergaderen kan de opdrachtgever hier rekening mee houden.

4.5 Technologieën

Binnen dit project zijn verschillende technologieën gebruikt. Hier volgt een lijst van de belangrijkste technologieën die binnen dit project zijn gebruikt:

GIT

Binnen dit project wordt git [13] gebruikt als versie control systeem. Git is geschreven door Linus Torvalds de ontwikkelaar van Linux en wordt gebruikt door de Linux Kernel. Haute Technique maakt intern gebruik van GitHub voor het hosten van hun verschillende git repositories. Binnen dit project is ook gebruik gemaakt van GitHub.

Icescrum

Icescrum [14] is een tool voor het beheren van Scrum projecten. Binnen Icescrum worden onder andere de productbacklog, sprintbacklog en de voortgang van de sprint bijgehouden. Er is gekozen om een tool als icescrum te gebruiken zodat er niet zelf een omgeving hoeft te worden opgezet voor het beheren van het scrum proces. Icescrum is een van de weinige gratis te gebruiken tools. Ook had de afstudeerder al ervaring met Icescrum.

Microsoft office

De documenten en grafieken die binnen dit project worden gecreëerd zijn gemaakt met applicaties uit de Microsoft office familie onder anderen Word en Excel.

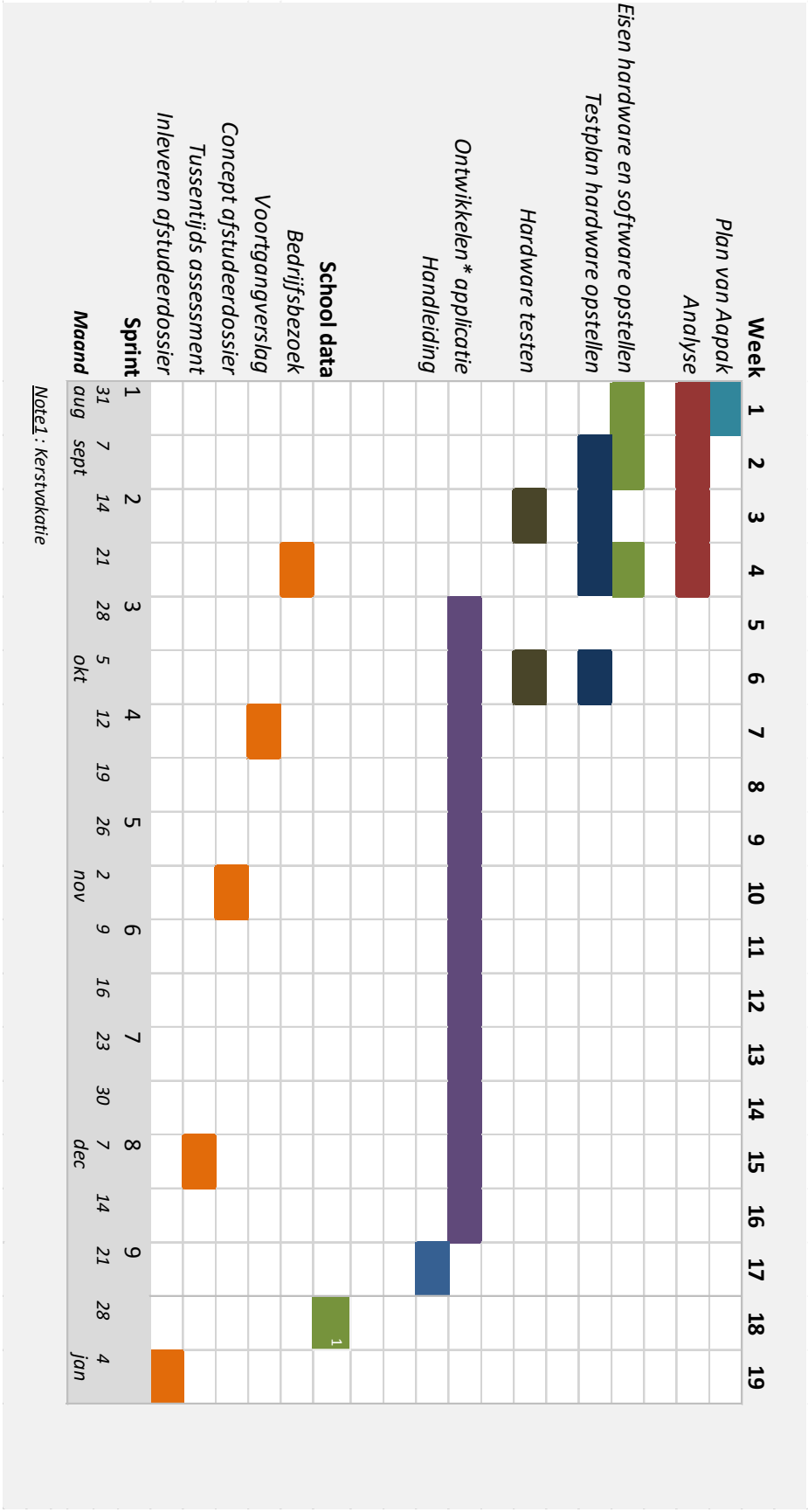
4.6 Planning

Het opstellen van een planning aan het begin van het project is belangrijk voor het goed verlopen van het project. De gekozen methodiek in dit geval Scrum heeft invloed hoe de planning er uit ziet. Scrum werkt in het algemeen alleen maar met sprint planningen. Deze worden aan het begin van een sprint bepaald en daarna uitgevoerd. Kennis over hoe de volgende sprint of sprints er precies uit gaan zien is er dan vaak nog niet. Toch is het gewenst om aan het begin van het project een globale planning te maken om zo beter grip te krijgen op de tijd en te controleren of het project haalbaar is. Deze planning is tot stand gekomen rekening houdend met de risico's die beschreven staan in het plan van aanpak. Dit is terug te zien in de planning door de risico's met de hoogste prioriteit vroeg in het project te tackelen.

Er is gekozen om een sprint lengte van twee weken aan te houden. Dit zorgt er voor dat de verschillende contact momenten van Scrum telkens op de zelfde dag van de week vallen dit maakt het makkelijker voor de opdrachtgever om de momenten in te plannen. Tevens zorgt een kortere sprint er voor dat er sneller kan worden bijgestuurd mocht de verkeerde richting zijn ingeslagen.

In de eerste sprint van het project worden de eisen van de te maken software bepaald ook worden er verschillende analyses uitgevoerd. De tweede sprint staat in het teken van het testen van de hardware die Sendrato levert. Ook worden er nog verschillende analyses uitgevoerd. Na dat sprint één en sprint twee zijn voltooid is er voor een groot deel duidelijk wat de te schrijven software moet gaan doen. Er kan in de hierna geplande sprints dan ook ontwikkeld worden aan de applicatie. De precieze indeling van de verschillende sprints is op dit moment echter nog niet bekend. De planning zal gaande het project tot stand komen.

De planning die te zien is in FIGUUR 5 is in de eerste week van het project tot stand gekomen. Het idee was toen dat er in de derde week van het project een testdag kon worden georganiseerd. Dit bleek later echter nog niet mogelijk omdat Sendrato nog niet klaar was met het implementeren van de verbeteringen die door Haute Technique de vorige testdag waren doorgegeven. Het testen is later in het project alsnog uitgevoerd. Door het wegvallen van het testen van de hardware in SPRINT 2 kon SPRINT 2 zich focussen op het leggen van een basis van de applicatie.



Figuur 5 Globale project fasering

5 Sprint 1 | Analyse

De eerste week van het project werd gebruikt om het project op te starten. De nieuwe opdracht werd vastgesteld en deze werd verder beschreven in het plan van aanpak. Aangezien de keuze voor scrum in de eerste week is gemaakt is deze week niet meegenomen in de sprint planning. De eerste sprint startte de tweede week van het project en had een lengte van één week. Er is gekozen om deze sprint één week korter te maken dan normaal aangezien er deze sprint voornamelijk analyses werden gedaan die niet meer tijd nodig hadden. De sprint stond in het teken van het helder krijgen van de eisen van de te maken software. Ook werden verschillende technieken die binnen het project worden gebruikt nader uitgezocht. Zie voor de user stories die tijdens deze sprint zijn geïmplementeerd het scrum rapport.

5.1 Eisen onderzoeken

Een belangrijk onderdeel in deze sprint was het achterhalen van de functionele en niet functionele eisen van de software en de hardware. Er is een use case diagram met bijhorende scenario's gemaakt. Dit use case diagram met bijhorende scenario's is vervolgens samen met de opdrachtgever doorgelopen om te kijken of de globale functionaliteiten van de applicatie correct waren. Uit de beschreven scenario's waren een groot deel van de eisen af te leiden. Het andere deel van de eisen (voornamelijk de niet functionele eisen) heeft de opdrachtgever aangegeven tijdens het gesprek.

Ook aan de hardware die Sendrato levert moesten eisen worden gesteld. Deze eisen kunnen later in het project worden gebruikt om te testen of de hardware voldoet aan de eisen van de opdrachtgever. De eisen zijn geprioriteerd volgens de MoSCoW methode.

Must haves	M	Aan deze eisen moet worden voldaan anders is het product niet bruikbaar.
Should haves	S	Deze eisen zijn zeer gewenst maar als er aan deze eisen niet wordt voldaan is er nog wel een bruikbaar product.
Could haves	C	Deze eisen komen alleen aan bod als er genoeg tijd is om deze eisen te implementeren
Won't haves	W	Deze eisen zullen in dit project niet aan bod komen maar kunnen voor eventuele vervolg projecten interessant zijn.

Hier volgt een kleine selectie van zowel de software eisen als die van de hardware. De rest van de eisen en het use case diagram met de daarbij horende scenario's is te vinden in het Requirements document in BIJLAGEN C.

Software functioneel

Nr.	Beschrijving	Prioriteit
F1.1	Er moeten meerdere lampen aan het systeem kunnen worden toegevoegd.	Must
F1.2	Er moeten verschillende typen lampen aan het systeem worden toegevoegd.	Should
F1.3	Een lamp moet in staat zijn om zich volgens een effect te gedragen.	Must

Software niet functioneel

Nr.	Beschrijving	Prioriteit
NF1.1	Het systeem stuurt de lampen aan via het DMX protocol gebruik makend van ArtNet	Must

NF1.2	Het systeem moet zo geschreven worden dat het relatief makkelijk is om in de toekomst nieuwe effecten toe te voegen	Should
NF1.3	De applicatie draait onder zowel Apple OSX als onder Windows	Must

Hardware functioneel

Nr.	Beschrijving	Prioriteit
F2.1	Het locatie systeem moet binnen vier uur opgezet en klaar voor gebruik zijn.	Must
F2.2	De locatie data moet realtime worden doorgestuurd. Dit houdt in dat er genoeg samples per seconden worden doorgestuurd zodat er hoog genoege resolutie ontstaat. Hoeveel samples er per seconden moeten zijn moet nog worden bepaald.	Must

Hardware niet functioneel

Nr.	Beschrijving	Prioriteit
NF2.1	Het systeem moet klein genoeg zijn om mee te nemen in een flight case	Must
NF2.2	Het systeem moet te bedienen zijn door een leek	Won't

De software eisen in het requirements document zijn in deze sprint ook vertaald naar user stories. Deze user stories zijn vervolgens toegevoegd aan de product backlog. Zo is bovenstaande functionele eis F1.2 vertaald naar onderstaande user story. Voor een overzicht van de verschillende backlog items kan hoofdstuk twee van het Scrum rapport worden geraadpleegd. De beschreven user stories zijn in deze fasen van het project nog erg globaal beschreven naar mate het project vordert worden de verschillende user stories specifieker.

Als gebruiker van de applicatie wil ik dat lampen worden aangestuurd volgens een bepaald effect.

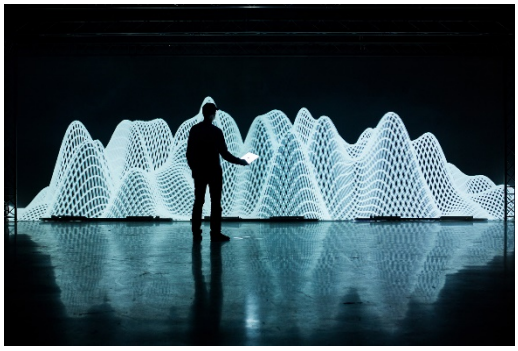
5.2 Ontwikkelomgeving

Haute Technique maakt veel gebruik van het framework openFrameworks [15] voor het creëren van interactie applicaties. Zo maakt ook de applicatie die gebruikt is tijdens de testdag met Sendrato gebruik van dit framework. Deze applicatie ontving aan de ene kant X, Y en Z waarden van de sensoren en stuurde aan de andere kant DMX uit. De applicatie is in staat om een volglicht zodanig aan te sturen dat deze een tracking device volgde.

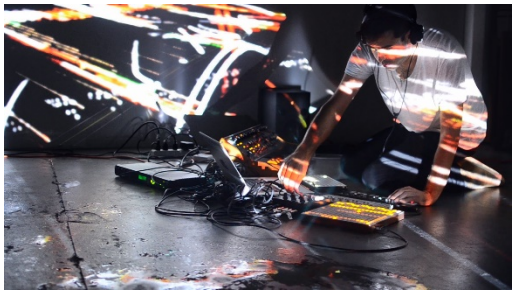
openFrameworks is een open source C++ framework voor het creëren van interactieve applicaties. In FIGUUR 6 tot en met FIGUUR 9 zijn een aantal verschillende toepassing te zien die gebruik maken van openFrameworks. Zo is in FIGUUR 6 een ecosysteem te zien dat reageert op mensen. Zo kunnen kinderen bijvoorbeeld zaadjes planten in het virtuele ecosysteem. In FIGUUR 9 wordt een applicatie getoond die aan de hand van audio input visuals laat veranderen. openFrameworks noemt zichzelf een:

an open source C++ toolkit designed to assist the creative process by providing a simple and intuitive framework for experimentation.

Figuur 6 Connected worlds [16]



Figuur 7 NO_THING [17]



Figuur 8 This city [18]

Figuur 9 Deserve [19]

In eerste instantie werd gedacht om de te schrijven applicatie voor dit project te ontwikkelen met openFrameworks aangezien de prototype applicatie ook van openFrameworks gebruik maakte en omdat er binnen Haute Technique veel kennis beschikbaar is over openFrameworks. Maar na het opzetten van een ontwikkelomgeving voor openFrameworks op Windows en er een test applicatie voor te ontwikkelen bleek dat openFrameworks waarschijnlijk niet de meest geschikte kandidaat was voor dit project. Dit komt doordat openFrameworks niet per definitie gemaakt is om 3D werelden mee te creëren. Het kan wel maar dit is niet de intentie van het framework. Hierdoor beschikt het niet over de functies van bijvoorbeeld een volwaardige game engine. In overleg met de opdrachtgever is er vervolgens bepaald om in de volgende sprint onderzoek te doen naar mogelijke andere kandidaten.

5.3 Onderzoek naar de werking van volglampen

Binnen dit project wordt er veel gewerkt met volglampen de kennis van de afstudeerder is echter vrij beperkt. Het is daarom nodig om verder onderzoek te doen naar volglampen. Dit onderzoek is begonnen met een vergadering met de opdrachtgever. De opdrachtgever heeft al ervaring met volg lampen (moving heads in het Engels) en het aansturen hiervan. Na het gesprek is het onderzoek verder gegaan op internet. Hierbij is de kennis die tijdens het gesprek was opgedaan gebruikt om gericht te kunnen zoeken. Naast de werking van moving heads te hebben beschreven is de protocol omschrijving van het DMX en het ArtNet protocol doorgelezen. Hier volgende de bevindingen van het onderzoek.

Volglampen ook wel moving heads zijn beweegbare lampen die vaak rond de 540 graden kunnen draaien (ook wel pan genoemd en 270 graden kunnen kantelen (ook wel tilt genoemd). Hiermee kan de plek waar de lamp schijnt worden aangepast. Moving heads kunnen vaak naast het veranderen van de kleur en helderheid van de lichtstraal ook een bepaalde effect op de licht straal toepassen bijvoorbeeld: het snel laten knipperen van de lamp (ook wel strobing) of het laten zien van een bepaald patroon.

Moving heads worden aan gestuurd doormiddel van het DMX protocol. Dit protocol specificeert 512 kanalen (ook wel een DMX universe). Afhankelijk van het type moving head gebruikt deze een tal van kanalen om zijn parameters zoals bijvoorbeeld de pan of tilt van de lamp te kunnen aan passen. Een

moving head 'luistert' afhankelijk van zijn adres naar een bepaalde offset binnen de 512 kanalen. Het DMX protocol is gebouwd boven op het seriële RS-485 protocol. De verschillend DMX apparaten zijn met elkaar door gelust door een middel van een XLR kabel.

Er zijn bepaalde hubs beschikbaar die het mogelijk maken om het DMX protocol via UDP door te sturen. Hierdoor wordt het bijvoorbeeld makkelijker om met bijvoorbeeld een laptop DMX apparaten aan te sturen. Dit gebeurt via het ArtNet protocol [20]. In FIGUUR 10 is een weergaven te zien van de opstelling om DMX apparatuur aan te sturen via het ArtNet protocol.

Figuur 10 Weergaven opstelling

5.4 Onderzoek positie analyse

Een van de eisen die naar voren kwam bij onderzoeken van de eisen was F1.4 'de positie van een volglamp wordt semi automatisch bepaald'. Het implementeren van deze eis zou op zich zelf al een hele afstudeeropdracht kunnen zijn. Er is daarom in samenspraak met de opdrachtgever besloten om deze eis een prioriteit te geven van should. De analyse voor deze eis is echter interessant voor de applicatie aangezien de kennis die tijdens het onderzoeken van deze eis wordt opgedaan kan worden gebruikt in de applicatie.

5.4.1 Omschrijving van het probleem

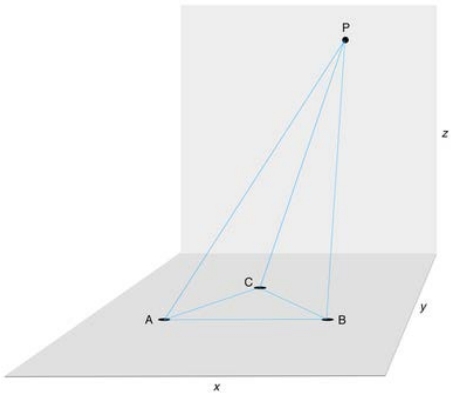
Het volgende probleem doet zich voor: tijdens een show hangen niet zelden 50 moving heads. Om deze moving heads aan te sturen zijn er een aantal eigenschappen van de moving head nodig namelijk: in welk DMX universe de lamp zich bevind, welk adres de lamp heeft, de positie van de lamp in de ruimte en de rotatie van de base van de moving head. Het DMX universe en het adres van de lamp wordt van te voren bekend gemaakt in het show plan deze zijn dus bekend. Het probleem doet zich echter voor bij de positie en de rotaties van de moving head.

De positie en de rotatie van de moving head worden ook van te voren in het show plan doorgestuurd. Maar deze waarden wijken echter vaak af van de werkelijke positie en rotatie. Dit komt doordat de lampen niet altijd op de gewenste locatie kunnen worden bevestigd. Dit is niet het enige probleem de lampen worden namelijk ook niet altijd even secuur opgehangen waardoor de lampen iets meer naar links of iets meer naar rechts hangen. Dit is voor de normale toepassingen geen probleem aangezien deze niet heel erg afhankelijk zijn van de positie van de moving heads. Maar bij bijvoorbeeld het volgen van een tracking device is wel de exacte positie en rotatie nodig. Dit aangezien de lamp anders te veel gaat afwijken.

5.4.2 Mogelijke oplossing

Er zijn een aantal verschillende oplossingen te bedenken voor dit probleem bijvoorbeeld elke moving head positie aware maken door bijvoorbeeld een tracking device aan de moving head toe te voegen of er voor zorgen dat de positie van een moving head wel van te voren klopt in het show plan. Beide oplossingen hebben hun problemen. Het grootste probleem is dat Haute Technique niet direct zeggen schap heeft over de positie en het gebruik van de lampen. Deze worden bepaald door de het bedrijf dat de shows organiseert en plant. Haute Technique 'reist' met Armin mee en moet het doen met wat er staat. Er kan dus niet vanuit worden gegaan dat de posities van de lampen kloppen. Binnen Haute Technique is het volgende idee ontstaan (dit is niet door de afstudeerder bedacht).

De positie van de moving head (punt P in FIGUUR 11) kan worden bepaald door de moving head achter een volgens op de punten A, B, en C te richten. De posities van deze punten zijn bekend even als de pan en de tilt van de moving head als die op een van de punten gericht is. Door gebruik te maken van een wiskundige formule kan punt P worden berekent. Er is binnen Haute Technique al een proof of concept gemaakt. Nu is alleen nog de vraag hoe de moving head op punten A, B en C wordt gezet. Hiervoor wordt waarschijnlijk een ander systeem ontwikkeld die moet samen werken met de applicatie die in dit project wordt ontwikkeld.



Figuur 11 Positie moving head in de ruimte [21]

5.5 Sprint evaluatie

Het doel van deze sprint was om de opdracht helder te krijgen en dit is gelukt. Door het maken van een use case (zie het requirements document) met bijhorende scenario's zijn de eisen bepaald. Tijdens deze sprint is naar voren gekomen dat er een onderzoek nodig is naar betere alternatieven voor openFrameworks er is voor gekozen om dit onderzoek plaats te laten vinden in SPRINT 2. Verder zijn in deze sprint verschillende risico's van het project geëlimineerd. Zo is er een onderzoek gedaan naar moving heads en het DMX protocol.

6 Sprint 2 | Architectuur

Het eerste deel van SPRINT 2 was gereserveerd voor het onderzoek naar het te gebruiken framework. De te schrijven applicatie zou gebruik maken van dit framework. Het tweede deel van deze sprint was ingericht om een basis architectuur van de applicatie te bedenken en deze te implementeren. De volgende user stories horen bij deze sprint.

Tabel 2 User stories sprint 2

Nr.	Story	Uren
20	Als ontwikkelaar wil ik onderzoeken welk framework het beste is geschikt om de applicatie in te ontwikkelen.	21
21	Als ontwikkelaar wil ik een basis ontwerp van de applicatie creëren om vandaaruit verder te ontwikkelen.	14
5	Als gebruiker wil ik meerdere lampen van mogelijk verschillende typen aan het systeem kunnen toevoegen.	7
8	Als gebruiker wil ik de verschillende lampen in een 3D omgeving zien.	7
6	Als gebruiker wil ik dat lampen worden aangestuurd volgens een bepaald effect.	7
9	Als gebruiker wil ik zien waar de verschillende tracking devices zich bevinden in de 3D ruimte.	1
11	Als gebruiker wil ik de mogelijkheid hebben om tracking devices te kunnen toevoegen.	3
13	Als gebruiker wil ik de mogelijkheid om de locatie van een of meerdere tracking devices op te nemen zodat deze later weer kan worden afgespeeld.	3
14	Als gebruiker wil ik de mogelijkheid om de opgenomen locatie data van een of meerdere tracking devices weer af te spelen.	4
15	Als ontwikkelaar wil ik dat de locatie data die binnen komt kan worden gesimuleerd.	2

6.1 Frameworks

In SPRINT 1 is gebleken dat openFrameworks mogelijk niet de beste kandidaat is voor het ontwikkelen van de applicatie. Dit komt omdat een groot deel van de applicatie leunt op het feit dat er een 3D wereld moeten worden getoond met daarin de verschillende lampen en tracking devices. Er moet daarom een onderzoek worden gedaan naar betere alternatieve. Voordat dit kon worden gedaan zijn er verschillende kenmerken opgesteld waarmee de verschillende gevonden frameworks met elkaar konden worden vergeleken.

Zie TABEL 3 voor de verschillende kenmerken. De kenmerken K4 tot en met K6 komen direct uit het requirements document. K4 correspondeert met de eis *F1.14*, K5 met de eisen *F1.1*, *F1.6* en *F1.18* en K6 met *NF1.3*. De overige kenmerken zijn kenmerken die (als er aan voldaan wordt) positieve gevolgen hebben voor het ontwikkelen met het framework. Zo is een framework met goede documentatie over het algemeen makkelijker in gebruik dan een framework met slechte documentatie of is het zoeken naar hulp bij een framework met een grote community makkelijker dan bij een framework met een kleine community.

Tabel 3 Kenmerken frameworks

Nr.	Beschrijving
K1	Opensource: Dit kenmerk geeft aan of een framework al dan niet opensource is. Een opensource framework kan een pre zijn aangezien het op deze manier mogelijk wordt om alles van het framework in te zien en mogelijk aan te passen of uit te breiden.
K2	Documentatie: Onder de documentatie van een framework wordt het volgende verstaan: hoe uitgebreid en duidelijk is de officiële documentatie over het gebruik van het framework, zijn er officiële tutorials beschikbaar en behandeld de documentatie de verschillende aspecten van het framework van beginner gebruik tot meer geavanceerd gebruik. Aan de hand van deze zaken zijn de verschillende frameworks vergeleken.
K3	Community: Een framework met een grote en actief community heeft en aantal voordelen: zo kan je eerder met je vragen worden geholpen, is er meer informatie online te vinden, zijn er meer plugins en of libraries voor het framework beschikbaar en kan het framework langer worden ondersteund. Voor elk framework is onderzocht hoe groot en actief het officiële forum is hiernaast is ook de git repository waar mogelijk bekeken.
K4	3D Werelden: Zoals als al eerder behandel moet er een 3D wereld worden getoond met daarin de verschillende lampen en tracking devices. Het te kiezen framework moet hiervoor ondersteuning hebben.
K5	GUI: De applicatie moet beschikken over een GUI zodat de eindgebruiker bijvoorbeeld lampen en dergelijk kan toevoegen aan de applicatie. Er is onderzocht of de frameworks standaard functionaliteit bieden voor een GUI.
K6	Platformen: Een van de niet functionele eisen is dat de applicatie minimaal moet draaien op Windows en OSX.

Game engines zijn toegespitst op het creëren en laten zien van 3D werelden. Er werden dus naast andere creative coding frameworks als openFrameworks verschillende game engines onderzocht als alternatief op openFrameworks. Een deel van de onderzochte engines waren al bekend bij de afstudeerder. Dit waren GameKit, Unreal, en Unity hiernaast zijn andere engines onderzocht. De andere engines uit TABEL 3 zijn gevonden door verschillende game forums en blogs door te lezen over game engines. Hierbij viel het op dat een tal van triple A engine makers hun engines met bijhorende ontwikkel tools sinds kort gratis of tegen een zeer lage prijs aanbieden voor indie developers[W4]. Vroeger waren de triple A engines alleen maar weggelegd voor rijke game studio's, dit is echter verleden tijd.

Naast de verschillende game engines zijn er ook nog twee creative coding frameworks onderzocht. De opdrachtgever gaf aan ooit gekeken te hebben naar Cinder en via internet is het Greenhouse framework gevonden.

De verschillende gevonden frameworks zijn op te delen in twee verschillende typen namelijk: creative coding frameworks hieronder vallen: openFrameworks, Cinder en Greenhouse en game engines hier onder vallen TorQue 3D, GameKit, IRRLicht, Unity en Unreal. In FIGUUR 12 is het tabel te zien met daarin de onderzochte frameworks en hoe deze zich tot elkaar verhouden. In dit figuur zijn ook andere kenmerken opgenomen naast de genen die beschreven zijn in TABEL 3.

Naam	Nr.	Creative coding			Engines				
		openFrameworks	*Cinder*	*Greenhouse*	*TorQue 3D*	*GameKit*	*IRRLicht*	*Unity3D*	*Unreal*
Opensource	**K1**	ja	ja	nee	ja	ja	ja	nee[3]	ja
Documentatie	**K2**	+	-	+	+	-	+	++	+
Community	**K3**	++	+	+	+	-	++	++	++
3D Werelden	**K4**	+	+	+	++	++	++	++	++
GUI	**K5**	-	-	-	+	-	+	++	++
Platformen	**K6**	ja	ja	nee	ja	ja	ja	ja	ja
Ondersteunde platformen		Windows, Linux, OSX, iOS, Android	Windows, OSX, iOS	OSX	Windows, Linux, OSX	Windows, Linux, OSX	Windows, Linux, OSX	Windows, Linux, OSX, iOS, Android, HTML5 + 17 meer	Windows, Linux, OSX, iOS, Android, HTML5 + 6 meer
Programmeertaal		C, C++	C, C++	C, C++	C, C++, TorqueScript	C, C++	C, C++	C#, Javascript, Boo script	C, C++, blueprints
Bekend bij afstudeerder		nee	nee	nee	nee	ja	nee	nee	nee
Licentie kosten		nee	nee	ja	nee	nee	nee	ja[1]	ja[2]
		- Voldoet slecht		*+ Voldoet goed*		*++ Voldoet erg goed*			

Figuur 12 Vergelijking van verschillende frameworks

Notitie (1): Unity biedt twee verschillende versies van de engine aan. Voor de professional edition moet 75$ per maand worden betaald. Deze versie biedt echter meer functionaliteit dan de personal edition en biedt de mogelijkheid om het spalsh screen aan te passen. Applicaties die gebouwd zijn met de gratis personal edition bevatten namelijk een spalsh screen dat niet kan worden aan gepast. [22]

Notitie (2): De Unreal engine is gratis te gebruiken, als er meer dan 3.000$ dollar per kwartaal wordt verdient moet er een royalty van 5% worden betaald over de bruto inkomsten van het product. [23]

Notitie (3): De engine is op dit moment (nog) niet open source maar het bedrijf achter Unity3D maakt steeds meer tools en libraries voor de engine open source. Bijvoorbeeld het nieuwe ontwikkelde user interface systeem. [24]

6.1.1 Toelichting van gevonden frameworks

Hier volgt per gevonden framework een korte omschrijving die het framework verder toelicht aan de hand van FIGUUR 4. Na deze toelichting volgt er een keuze voor een framework.

- openFrameworks: Is een framework dat zich richt op het zogenoemde creative coding. openFrameworks wordt veel gebruikt door Haute Technique voor het maken van interactieve installaties. Het framework biedt een uitgebreide set aan functionaliteiten zo heeft het standaard support voor de Kinect. Hiernaast beschikt openFrameworks over een grote actieve community en kan de functionaliteit snel worden uitgebreid door gebruik te maken van een van de vele plugins die beschikbaar zijn. [15]
- Cinder: Lijkt kwa functionaliteit heel erg op openFrameworks maar dan met minder functionaliteit. Hiernaast is de documentatie ook van mindere kwaliteit. [25]
- Greenhouse: Is gratis voor niet commercieel gebruik. Greenhouse richt zich op het zelfde gebied als Cinder en OpenFrameworks en richt zich dan voornamelijk op multi-user input. [26]
- TorQue 3D: Is een gratis te gebruiken game engine. TorQue 3D beschikt over een level editor waarin levels kunnen worden gecreëerd in een grafische omgeving. Naast ondersteuning voor C en C++ biedt TorQue 3D ondersteuning voor Torquescript. Dit is een scripttaal speciaal ontwikkeld voor de TorQue engine en maakt het mogelijk om snel gedrag aan 3D objecten toe te voegen. [27]

- GameKit: Combineert verschillende open source software projecten om zo een engine te creëren. Tijdens een minor game development heeft de afstudeerder met deze engine gewerkt. GameKit heeft een kleine community en de beschikbare documentatie is vaak niet volledig. [28]
- IRRLicht: is een open source game engine geschreven in C++. IRRLicht focust zich vooral op het creëren van een high performance 3D game engine. IRRLicht biedt niet net zoals TorQue 3D een scripttaal of een level editor aan. Deze kunnen wel worden gedownload van 3e partijen. [29]
- Unity3D: ook wel Unity is samen met Unreal een zogenoemde triple A engine. Dit betekent dat een paar van de beste games gebruik maken van deze engines. Unity is in de lijst van onderzochte engines de enige die niet gebruik maakt van de programeer taal C, of C++. De kern van Unity engine is geschreven in C++ maar de games worden ontwikkeld in C# of javascript. Unity biedt een compleet pakket aan ontwikkel tools aan waarmee level kunnen worden ontworpen en getest. Unity ondersteund van de onderzochte frameworks de meeste platformen. [30]
- Unreal: staat bekend om zijn grafische kracht. De game logica wordt geschreven in C of C++ hiernaast biedt Unreal ook een visual scriptingtaal (blueprints) aan. Hiermee wordt het mogelijk om zonder gebruikt te maken van een programmeertaal de game van gedrag te voorzien. De documentatie van Unreal is minder uitgebreid als die van Unity. Unreal biedt net als Unity een complete set aan ontwikkeltools aan. [31]

6.1.2 Keuze tussen type framework

De eerste keuze die moest worden gemaakt was de keuze voor een echte game engine of een creative coding framework kijkend naar de eisen uit het requirements document is het visualiseren van de wereld een belangrijke eis. Het visualiseren van de wereld zou in zekere maten mogelijk moeten zijn met een van de creative coding frameworks. De keuze zou dan vallen voor openFrameworks. Aangezien dit framework standaard het meeste ondersteund en de actiefste community heeft. Hiernaast is er binnen Haute Technique kennis beschikbaar over dit framework. Een game engine is echter meer geschikt voor het creëren van 3D werelden en is hier op toegespitst. Ook zijn er game engines beschikbaar die een nog grotere community hebben dan openFrameWorks. Verder kan er door gebruik te maken van een volwaardig game engine sneller een werkend product worden opgeleverd. De creative coding frameworks moeten het dus afleggen tegen de game engines. Er moet dus uit een van de vijf onderzochte games engine worden gekozen.

6.1.3 Keuze tussen game engines

GameKit viel als eerste af aangezien deze engine simpel weg niet van voldoende kwaliteit is. De engine beschikt over een kleine community en slechte documentatie ook beschikt de engine niet over een level editor. IRRLicht is vergeleken met de overgebleven engines minder volledig. Zo ontbreekt ook voor deze engine een level editor. Voor de opdrachtgever was het betalen van licentie kosten in de toekomst geen probleem hierdoor bleven zowel Unity als Unreal goede kandidaten. TorQue 3D valt in het niet als deze wordt vergeleken met Unity of Unreal deze twee triple A engine bieden een grote community, uitgebreide ontwikkel tools, veel plugins en libraries om de functionaliteit van de engine uit te breiden, support van de ontwikkelaars en een zekerdere toekomst aangezien de engines een stuk groter zijn dan TorQue 3D wat er voor zorgt dat het project niet zomaar stopt. Unity en Unreal bleven dus over. Beide engines bieden hun ontwikkeltools gratis aan.

Figuur 13 Unity logo [30]

Figuur 14 Unreal Logo [31]

Unreal en Unity zijn beide goede keuzes voor het creëren van games. Het was lastig om een keuze te maken tussen de twee engines aangezien ze beide aan de gestelde eisen voldeden. Hierbij bleef de documentatie van Unreal wel achter dat van Unreal. Er was daarom besloten om in beide engines een test applicatie te bouwen. Op deze manier konden de engines beter onderzocht worden. Er was per engine een halve dag uitgetrokken om deze beter te leren kennen. De test applicatie moest een kubus in de wereld plaatsen en deze een aantal keer kopiëren hiernaast moest er een simpel UDP server worden gemaakt.

6.1.4 Keuze voor Unity

Unity kwam als beste uit de bus bij het schrijven van een test applicatie. Het ontwikkelen met Unity was goed gedocumenteerd en er kon snel worden begonnen met daadwerkelijk programmeren. Ontwikkelen binnen Unreal was lastiger aangezien de documentatie op sommige punten te kort schoot. Ook zat de ontwikkelomgeving minder duidelijk in elkaar. Een halve dag is natuurlijk te weinig om alle ontwikkel tools en de engine te bekijken maar er kan wel een inschatting worden gemaakt over de werking van het platform. Hierbij kwam Unity in samenwerking met C# er beter uit dan Unreal. Er is dus gekozen om de applicatie te ontwikkelen met de Unity engine. Naast de resultaten van de testapplicaties zijn de volgende reden ook doorslag gevend geweest in de keuze voor Unity.

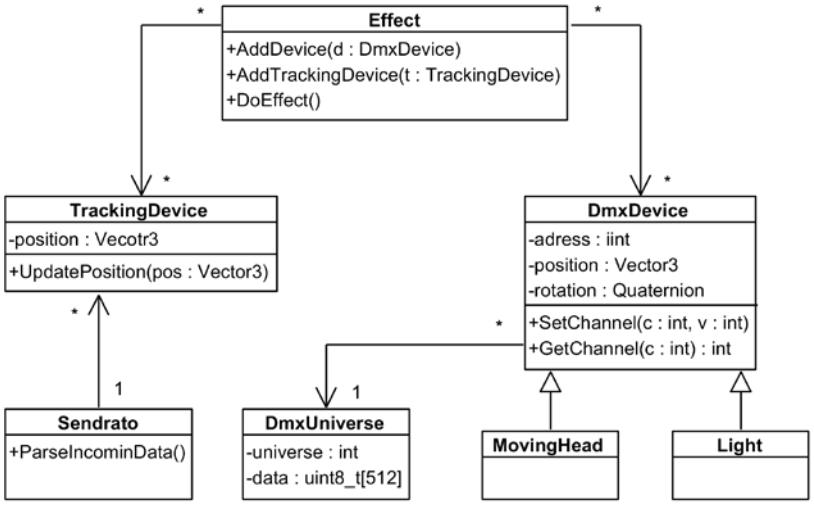
- Unity maakt gebruik van C# dit is een voordeel aangezien C# een grotere standaard library heeft waardoor het werken met bijvoorbeeld threads of sockets een stuk makkelijker wordt.
- C# maakt gebruik van een garbage collector hierdoor hoeft het geheugen niet met de hand worden beheerd waardoor er minder snel memory leaks optreden. In tegenstelling tot C++ waar dit wel moet. De nieuwe C++ standaard biedt tegenwoordig wel zogenoemde auto pointers die objecten automatisch verwijderen als deze niet meer worden gebruikt maar deze zijn niet te vergelijken met de garbage collector van C#.
- C# wordt over het algemeen langzamer bevonden dan C of C++. Dit komt deels door de garbage collector. De garbage collector moet af en toe namelijk de ongebruikte objecten opruimen (dit kost relatief veel tijd). Het andere deel komt omdat C# vaak net als bij Java draait op een virtual machine. Deze virtual macihne vertaald de C# byte code naar machine code en voert deze uit. Dit is bij C++ code niet het geval. Mocht performance binnen de applicatie een probleem worden dan biedt Unity de mogelijkheid om bepaalde stukken code te schrijven in C of C++ [32]
- Door de hierboven beschreven voordelen van C# ten opzichte van C++ is C# een relatief makkelijkere taal om te leren dan bijvoorbeeld C++. Aangezien er binnen Haute Technique (nog) niet veel programeer kennis aanwezig is kan de stap naar C# makkelijker worden gemaakt.

De keuze is dus gevallen op de Unity engine. De programmeertaal C# is nog niet bij de afstudeerder bekend. Er is dus onderzoek nodig naar de werking van C#. Nu er gebruik wordt gemaakt van een game engine zou er veel tijd kunnen worden gespendeerd in het grafisch mooi maken van de applicatie door bijvoorbeeld hoge resolutie 3D modellen van de lampen in te laden. Dit is echter niet de focus van dit project en zal hoogstwaarschijnlijk dus ook niet worden gedaan. Dit neemt niet weg dat dit in de toekomst als nog kan worden toegevoegd.

Voor zover bekend is heeft niemand nog Unity ingezet voor het aansturen van DMX lampen. In samenwerking met de hardware van Sendrato opent dit nieuwe deuren voor het aansturen van lampen. Een voorbeeld hiervan kan zijn dat volglampen voortaan worden aangestuurd met gebruik van de physics engine van Unity. De physics engine is verantwoordelijk voor het simuleren van het fysieke gedrag van objecten bijvoorbeeld hoe de zwaartekracht objecten manipuleert of hoe objecten met elkaar botsen. Door de physics engine in te zetten voor het aansturen van volglampen kunnen er nieuwe interessante effecten worden gecreëerd die met regulieren lichttafels niet of nauwelijks te creëren zijn. Hiernaast maakt de 3D wereld het mogelijk om snel de status van de verschillende lampen te zien of te simuleren. Het gebruik van een game engine opent dus nieuwe interessante deuren voor het aansturen van DMX apparaten.

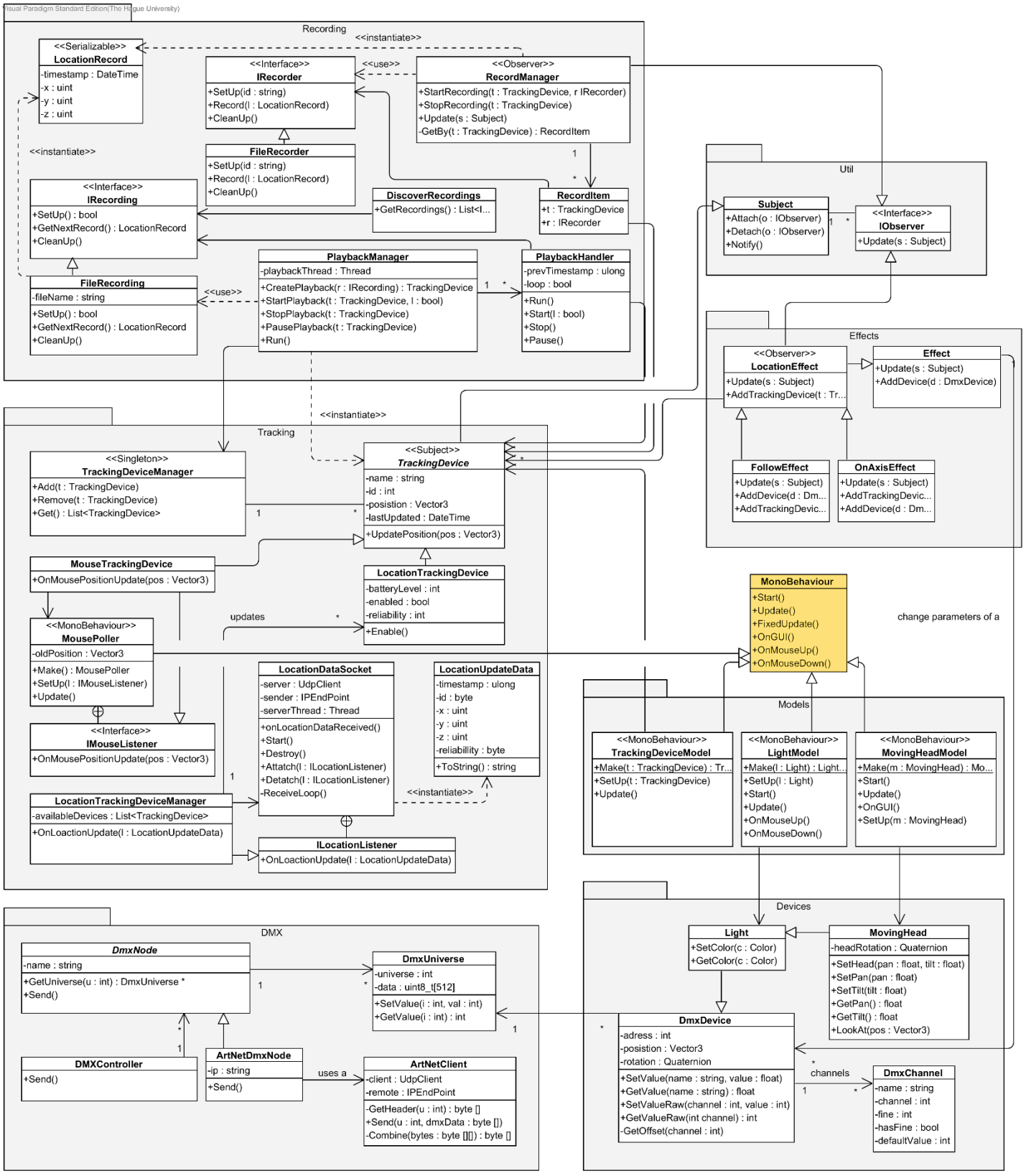
6.2 Ontwerp

Zoals eerder beschreven is wordt in deze sprint ook de basis architectuur van de applicatie ontworpen. Deze architectuur wordt gaande het project steeds verder uitgebreid. De architectuur is beschreven aan de hand van verschillende UML diagrammen [33]. De in dit hoofdstuk weergegeven diagrammen zijn in de loop van deze sprint meerdere malen aangepast. Op het moment dat er namelijk begonnen wordt met het implementeren blijken bepaalde aspecten van het diagram verkeerd in elkaar te zitten of missen nog bepaalde functionaliteit. Deze zaken worden dan alsnog aan de diagrammen toegevoegd. Voor de eerste versie van het ontwerp kan onderstaand figuur worden geraadpleegd.



Figuur 15 Analyse klassendiagram

Er is tot het ontwerp gekomen door te kijken naar mogelijke objecten die in het probleem domein voorkomen. Denk hier bijvoorbeeld aan DMX, moving head of tracking device. Deze objecten zijn om gezet naar klassen met verschillende attributen en operaties. Deze klassen zijn vervolgens verder uitgewerkt. In dit proces van uitbreiding zijn er ook nieuwe klassen ontstaan. Het uiteindelijke ontwerp van deze sprint is te vinden in FIGUUR 16. Om een betere structuur aan het klassendiagram aan te brengen zijn de verschillende klassen opgedeeld in packages. Elke package heeft zijn eigen functionaliteit. Bij het indelen van de packages wordt er gestreefd naar een hoge cohesie binnen de package en een lage koppeling naar andere packages. Soms was de afweging voor een duidelijkere structuur echter hoger. Dit is vooral te zien in de *Util* package en alle op volgende versies hiervan, zie het ontwerp document in BIJLAGEN H.



Figuur 16 Klassendiagram

Om het ontwerp leesbaar te houden zijn de meeste getters en setters uit de klassen weggelaten. Ook zijn op sommige plekken de methode en variabele namen ingekort en de constructors weggelaten.

In het ontwerp uit FIGUUR 16 is de klasse *MonoBehaviour* gemarkeerd. Dit is om aan te duiden dat dit een klasse is uit de *UnityEngine* package. De *MonoBehaviour* klasse is de base klasse voor (bijna) alle scripts binnen de Unity engine. Binnen Unity worden scripts gebruikt om gedrag aan game objecten te koppelen, later meer hierover. Er is gekozen om de koppeling met deze klasse zo laag mogelijk te houden. Het overerven van een *MonoBehaviour* zorgt er namelijk voor dat het object mee gaat 'draaien' in de engine en dit zorgt voor meer overhead.

Hier volgt een korte omschrijving van een tweetal packages. Voor de omschrijving van het gehele systeem kan het ontwerp rapport geraadpleegd worden. Het ontwerp rapport beschrijft de eindversie van het ontwerp.

6.2.1 DMX package

De DMX package is verantwoordelijk voor het versturen van DMX data aan de hand van de verschillende nodes en universes. De *DmxController* beheert verschillende *DmxNodes*. *DmxNode* is een klassen die een abstracte methode *Send()* definieert. Op deze manier kunnen er in de toekomst makkelijker nieuwe manieren van verzenden worden toegevoegd. Op dit moment wordt de klasse *ArtnetDmxNode* gebruikt om de data via het ArtNet protocol te versturen. Dit gebeurt door middel van de klasse *ArtNetClient*. Deze klasse genereert een ArtNet pakketje en verstuurd deze via het UDP protocol over het netwerk naar de ArtNet hub.

6.2.2 Tracking package

Een ander onderdeel van het systeem is het tracking deel. Dit deel is verantwoordelijk voor het beheren van de verschillende tracking devices binnen het systeem. *TrackingDevice* is een abstracte klassen die het gedrag van een tracking device definieert. Er zijn twee verschillende type tracking device namelijk *MouseTrackingDevice* dit is een 'virtueel' tracking device. Deze volgt namelijk de positie van de muis in de 3D wereld. Het andere typen tracking device is de *LocationTrackingDevice*. Deze klassen komt overeen met een daadwerkelijk tracking device van Sendrato.

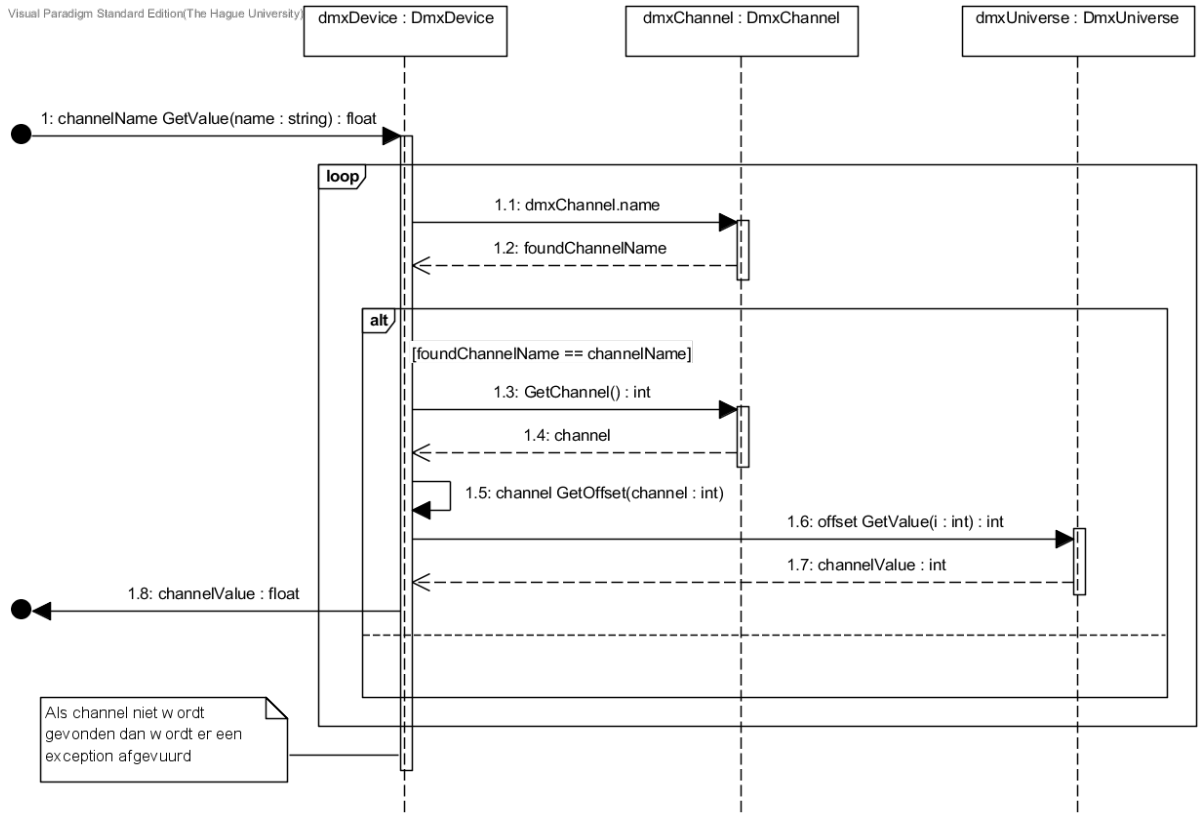
Om de *MouseTrackingDevice* te implementeren moest er een gebruik worden gemaakt van een zogenoemde *MonoBehaviour* script. Binnen Unity moeten alle scripts die iets willen doen in de 3D wereld overerven van de klasse *MonoBehaviour*. Hierbij krijgt de klassen ook toegang tot verschillende events bijvoorbeeld: *OnMousePressed*, *Update* enzovoort [34]. De methode *Update* wordt constant door de engine aangeroepen om objecten te updaten. Binnen de *Update* methode kan dus gecontroleerd worden of de positie van de muis is veranderd en zo nodig een actie worden uit gevoerd. Het is echter binnen C# niet mogelijk om van twee klassen te overerven. Daarom is er een speciale *MousePoller* klasse geïmplementeerd die gebruik maakt van een zogenoemde listener om de *MouseTrackingDevice* klasse op de hoogte te stellen als de muis positie veranderd. Door gebruik te maken van raycasting[W1] kan vervolgens de positie van de plek waar de muis naar wijst worden gevonden.

Een tracking device maakt gebruikt van het *Observer* pattern [35] voor het door geven van positie veranderingen. Op het moment dat de methode *UpdatePosition* wordt aangeroepen van een tracking device bijvoorbeeld door het verplaatsen van de muis bij een *MouseTrackingDevice* worden alle *Observers* op de hoogte gesteld. De *TrackingDeviceManager* houdt een lijst bij van alle actieve tracking devices. In de toekomst kan een GUI klassen hier gebruik van maken om een lijst te tonen van de actieve tracking devices.

Sendrato stuurt de tracking device data door via UDP. Het protocol dat Sendrato hiervoor gebruikt is beschreven in het hardware testrapport in BIJLAGEN D. Om deze data te kunnen ontvangen moet er dus een UDP server worden geïmplementeerd. Aangezien het ontvangen en parsen van UDP pakketje relatief lang kan duren is het niet geschikt om het ontvangen en parsen van UDP pakketje in de zelfde thread als Unity te doen. Er is daarom gekozen om een losse thread te creëren waarin de UDP server draait. De methode *ReceiveLoop* van de klassen *LocationDataSocket* draait in deze thread. Het sequentiediagram in HOOFDSTUK 6.4 ligt dit proces verder toe.

6.3 Sequentiediagram DMX

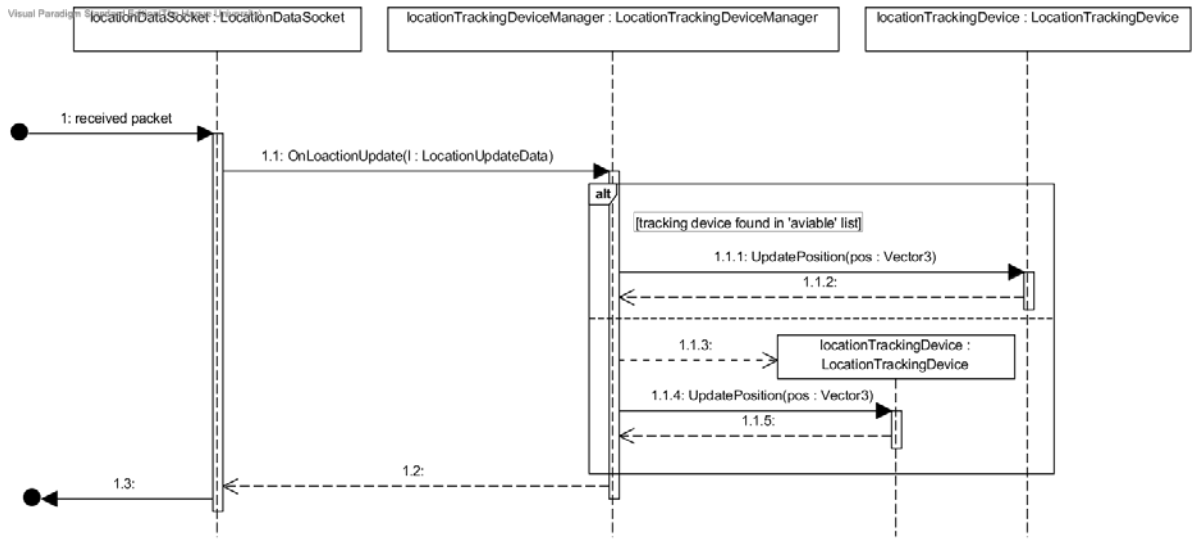
In onderstaand sequentie diagram is te zien hoe de waarde van een kanaal van een DMX apparaat kan worden opgevraagd.



Figuur 17 Sequentiediagram: waarde van een DMX kanaal opvragen

6.4 Sequentiediagram Tracking

In FIGUUR 18 is een sequentiediagram te zien die het ontvangen van locatie data van de sensoren van Sendrato weergeeft. Er zijn twee mogelijkheden die zich kunnen voordoen als er een locatie update binnen komt. Namelijk: het locatie tracking device is al eens eerder gezien dit resulteert in een locatie update van het desbetreffend tracking device of het tracking device is nog niet bekend. In het laatste geval wordt er een nieuw tracking device gecreëerd en toegevoegd aan de lijst met beschikbare tracking devices.



Figuur 18 Sequentiediagram: tracking device update

6.5 Effecten

Een effect veranderd verschillende parameters van een lamp of lampen aan de hand van locatie veranderingen van een tracking device. Er zijn in deze sprint twee effecten ontworpen. Het eerste effect is een effect dat een lamp aan zet op het moment dat een tracking device zich in de buurt van een bepaalde as bevindt. Het andere effect is een volg effect dat moving heads in staat stelt om een tracking device te volgen. Hierbij wordt de methode *LookAt*() van de *MovingHead* klassen gebruikt. Deze methode berekent aan de hand van de rotatie van de head en base en de positie waar de moving head naar moet wijzen de correcte pan en tilt voor de lamp. Alle rotaties binnen de applicatie worden opgeslagen als Quaternionen.

6.6 Locatie sensoren

De locatie sensoren die Sendrato levert communiceren via het UDP protocol. De applicatie beschikt over een UDP server. Deze bevindt zich in de klassen *LocationDataSocket*. Aangezien de sensoren in deze sprint nog niet aanwezig zijn en de code die de verantwoordelijk is voor het parsen van het Sendrato protocol en toevoegen van de tracking devices wel getest moesten worden is er een simulatie applicatie geschreven. Deze applicatie gedraagt zich als het systeem van Sendrato en stuurt elke 100 milliseconden een locatie update. Met deze applicatie kon de code worden getest zonder dat de sensoren fysiek aanwezig waren.

6.7 Unit Testen

Zoals beschreven in het HOOFDSTUK 0 is er besloten om voor delen van het systeem die zich lenen voor het schrijven van unit tests unit tests te schrijven. Unit tests worden vaak geautomatiseerd uitgevoerd. Unity beschikt over een test framework die het mogelijk maakt om unit tests automatisch te laten uitvoeren. Dit framework maakt onder anderen gebruik van het C# framework NUnit [36] voor het schrijven van unit tests.

```
[Test]
public void SetAndGetChannel([NUnit.Framework.Range(0.0F, 1.0F, 0.1F)] float value)
{
    // Get a DMX device and set the RED channel
    DmxDevice device = GetDmxDevice();
    device.SetValue(DmxChannel.RED, value);

    // Receive the actual value
    float actualValue = device.GetValue(DmxChannel.RED);

    // Make sure the value and the actual value are within 0.01F of each other
    Console.WriteLine("Value: {0} found: {1}", value, actualValue);
    Assert.That(actualValue, Is.EqualTo(value).Within(precisionNormal));
}
```

Figuur 19 Unit tests voor het testen van de methode SetValue() van de klasse DmxDevice

De focus van de unit test lag tijdens deze sprint op de packages DMX en Devices. Deze packages vormen de basis van het systeem. Per methode zijn er vaak verschillende tests geschreven om zo alle cases te testen. Een methode kan namelijk met de juiste waarden worden aangeroepen maar ook met de verkeerde waarden. Bij beide scenario's moet de methode correct blijven werken. Zie FIGUUR 19 voor een van de geschreven unit tests

6.8 Sprint evaluatie

Deze sprint had als doel om een keuze te maken voor een framework dat zou worden gebruikt voor het ontwikkelen van de applicatie. Hiernaast moest de basis structuur van de applicatie ontworpen worden. Verder heeft deze sprint een opzet gemaakt voor het aansturen van lampen volgens een bepaald effect maar dit onderdeel van de applicatie moet nog verder worden uitgebreid. Zo is het op dit moment niet mogelijk om gemakkelijk nieuwe effecten toe te voegen. De opdrachtgever heeft aangegeven dat dit een belangrijke wens is. De volgende sprint zal zich hier op gaan focussen.

Zoals in het ontwerp uit HOOFDSTUK 6.2 te zien is wordt er op sommige plekken gebruik gemaakt van zogenoemde *Listener* klassen om events door te sturen naar andere klassen. Halverwege de sprint kwam naar voren dat C# hier standaard ondersteuning voor biedt namelijk in de vorm van *delegates* Aangezien een groot deel van de architectuur toen al geschreven was (gebruik makend van een eigen implementatie) is er besloten om deze *Listeners* gaandeweg het project om te zetten naar *delegates*.

7 Sprint 3 | Effecten

SPRINT 3 staat in het teken van het verder uitwerken van het effecten gedeelte van de applicatie. In de vorige sprint is hier al een kleine opzet voorgemaakt. De volgende userstories zijn deze sprint uitgewerkt.

Tabel 4 Sprint planning sprint 3

Nr.	Story	Uren
25	Als ontwikkelaar wil ik onderzoeken wat een effect precies inhoudt.	8
26	Als ontwikkelaar wil ik onderzoeken welke scriptaal het meest geschikt is om te gebruiken voor het schrijven van effecten.	21
27	Als gebruiker wil ik effecten kunnen toevoegen aan een lamp.	12
28	Als gebruiker wil ik een effect kunnen verwijderen van een lamp.	2
29	Als effect schrijver wil ik een effect script kunnen toevoegen aan de applicatie.	7
30	Als effect schrijver wil ik een effect script kunnen verwijderen uit de applicatie.	2
31	Als student wil ik het bedrijfsbezoek voorbereiden.	2

7.1 Configuratie

In SPRINT 1 is er requirements document gecreëerd. In dit document zijn ook eisen te vinden met betrekking op het configuratie onderdeel van de applicatie. In eerste instantie was het plan om in één van de komende sprints dit onderdeel van het systeem verder te onderzoeken en te implementeren. Maar door de verandering van prioriteiten is dit hoogstwaarschijnlijk niet meer haalbaar. De opdrachtgever heeft namelijk aangegeven dat het schrijven en toevoegen van effecten een hogere prioriteit heeft. Er is daarom in samenspraak met de opdrachtgever besloten om vooral te focussen op het aansturen van de lampen aan de hand van effecten. Het configuratie deel van de applicatie is op dit moment nog een stap te ver.

7.2 Wat is een effect

Om het effecten gedeelte van de applicatie te kunnen ontwerpen en implementeren is het belangrijk om eerst helder te krijgen wat een effect precies in houdt. In de vorige sprint werd de volgende omschrijving van een effect gebruikt: een effect veranderd verschillende parameters van een lamp of lampen aan de hand van locatie veranderingen van een tracking device. Deze beschrijving was echter niet ruim genoeg en er is daarom verder onderzoek nodig om de definitie van een effect te bepalen.

Een startpunt van het onderzoek was het analyseren van films van licht shows. Hierbij werd gekeken hoe de lampen zich gedragen tijdens een show. Hierbij is naar voren gekomen dat er tijdens een licht show veel verschillende effecten worden gebruikt. De lichtman bepaald welk effect wanneer wordt gebruikt en bepaald de overgangen tussen de effecten.

Een ander aangrijpingspunt was de ervaring van de opdrachtgever. Zoals al eerder beschreven heeft de opdrachtgever een applicatie geschreven zodat er lampen kunnen worden aan gestuurd aan de hand van de arm bewegingen van Armin van Buuren (ook wel de Myo applicatie genoemd). De applicatie die hiervoor geschreven is al eerder in dit project doorgenomen. Hierbij legde de opdrachtgever de werking van de applicatie uit. Tijdens dit gesprek kwam naar voren dat een effect bepaalde parameters heeft die ‘on the fly’ kunnen worden aangepast. Dit kan bijvoorbeeld de kleur of de snelheid van het effect zijn. Om deze parameters van een effect aan te passen maakt de opdrachtgever gebruik van een MIDI controller[W2]. Vanuit de verschillende analyses kon de definitie van een effect gedetailleerder worden beschreven. Er is tot de volgende beschrijving gekomen:

1. Een effect heeft betrekking op een of meerdere apparaten die bij de zelfde groep horen. Hierbij wordt met groep bedoeld dat de apparaten allemaal over de zelfde optie beschikken zoals bijvoorbeeld: de kleur van de lamp veranderen of dat de lamp beweegbaar is.
2. Een effect kan gebruik maken van een bepaalde gebeurtenis. Dit kan zijn: het veranderen van de positie van een tracking device of het voorbijgaan van de tijd.
3. Aan de hand van deze gebeurtenis past een effect bepaalde eigenschappen van een apparaat aan.
4. Tijdens de loop van een effect kunnen bepaalde parameters van een effect ‘on the fly’ worden aangepast. Zo kan bijvoorbeeld de snelheid of de kleur van het effect worden aangepast.
5. Een effect kan bij het initialiseren bepaalde extra gegevens nodig hebben. Denk bijvoorbeeld aan een gebied waarbinnen het effect werkt.
6. Een effect kan worden gestart en worden gestopt.
7. Een effect kan worden toegevoegd en worden verwijderd.

7.3 Scripting

Een wens van de opdrachtgever is dat het toevoegen van nieuwe effecten aan de applicatie en het aanpassen van bestaande effecten gemakkelijk is. Bij de bestaande Myo applicatie heeft de opdrachtgever namelijk ondervonden dat het schrijven en aanpassen (of tweaken) van effecten een belangrijk onderdeel is van de applicatie.

In de huidige generatie van de Myo applicatie moet telkens als een effect moet worden toegevoegd of moet worden aangepast de gehele applicatie opnieuw worden gecompileerd. Hierbij gaat veel tijd verloren. Ook biedt de Myo applicatie geen eenduidige interface voor het creëren van een effect.

Om het creëren van nieuwe effecten en het aanpassen van bestaande effecten makkelijker te maken kan er voor worden gekozen om een zogenoemde effecten bouwer te implementeren. Met deze effecten bouwer kunnen effecten worden gecreëerd door middel van een GUI. De gebruiker hoeft dan geen code te tikken. Een effecten bouwer gaat echter ver voorbij aan de scope van dit project en is ook niet wat de opdrachtgever zoekt. Een betere oplossing zou het gebruik van scripting zijn. Door gebruik te maken van scripting hoeft de applicatie niet constant opnieuw te worden gecompileerd als er een effect wordt gecreëerd of aangepast. De engine interpreteert de code namelijk run time en voert deze vervolgens uit. Ook is het toevoegen van script support voor de applicatie makkelijker dan dat het bouwen van een GUI is.

Er zijn veel verschillende scriptalen geschreven door de jaren heen maar niet al deze talen zijn geschikt voor het gebruik binnen een Unity applicatie. Er is daarom onderzoek gedaan naar scriptalen die ‘out of the box’ werken met Unity. Het schrijven of compatibel maken van een scriptaal voor Unity gaat de scope van dit project namelijk voorbij. Er zijn een aantal verschillende scriptalen gevonden die voldoen aan deze eis. De drie eerste scripttalen in onderstaand tabel zijn gevonden door te zoeken in de Unity Asset store. Dit is een online winkel waarin ontwikkelaars plugins voor Unity aanbieden. Het was bij de afstudeerder al bekend dat C# ook als ‘scripttaal’ kon worden gebruikt.

Tabel 5 Overzicht scripttalen

Scriptaal	Implementatie	Omschrijving
Lua	NLua	NLua is een C# implementatie van de scripttaal LUA. Lua is een veel gebruikte scriptaal in de game industrie. [37]
Python	IronPython	IronPython is een open source C# implementatie van de programmeertaal Python. [38]
Javascript	Jint of Jurassic	Jint [39] en Jurassic [40] zijn beide implementaties van de scriptaal Javascript voor Unity.
C#	-	Bij de C# runtime wordt standaard ook de C# compiler geleverd. Hierdoor wordt het mogelijk om runtime C# code te compileren en toe te voegen aan de applicatie. [41]

De belangrijkste eis om een scriptaal te gebruiken was het feit dat er op deze manier effecten 'on the fly' konden worden toegevoegd en aangepast. Dit is mogelijk met al de hierboven beschreven talen. De talen verschillen echter in hoe gemakkelijk ze zijn te implementeren in Unity voor de eerste drie talen (LUA, IronPython en Javascript) moeten er bindings in C# worden gemaakt om de scripts te kunnen laten werken met de applicatie. Dit is voor C# niet nodig. Hierdoor is C# het snelst te implementeren in de applicatie. Verder biedt C# nog een aantal andere voordelen:

- Door gebruik te maken van C# kunnen de effecten zonder extra werk gebruik maken van de gehele standaard library van C#. Dit leidt er toe dat effecten zo complex kunnen worden gemaakt als nodig.
- De effecten worden gecompileerd naar Assemblys (of DLL's) hierdoor hoeft het effect maar één keer te worden gecompileerd door de applicatie (als het niet wordt aangepast). Dit zorgt voor een snelheidswinst bij het volgende keer opstarten van de applicatie.

Op dit moment lijkt C# de beste keuzen om te gebruiken als scripttaal. Het is echter mogelijk dat de wensen van de opdrachtgever in de toekomst veranderen en dat er een andere scriptaal moet worden gebruikt. Door dit mee te nemen in het ontwerpen van de software kan dit in de toekomst gemakkelijk worden toegevoegd.

Om het schrijven van effecten gemakkelijk te maken moet er een set van helper methodes en of klassen komen die het gebruik van veel gebruikte onderdelen binnen effecten makkelijker maken. Denk bijvoorbeeld aan de vraag of een positie binnen een bepaald gebied ligt. Door het schrijven van deze klassen kan er sneller een effect worden ontwikkeld.

7.4 Ontwerp

Nadat de verschillende analyses waren afgerond kon er begonnen worden met het ontwerpen van het effect onderdeel van de applicatie. In FIGUUR 20 is het ontwerp van de *Effects* en *Follow Effect* package te zien. De *Effect* package bestaat uit twee onderdelen: het zoeken, compileren en beheren van de effect scripts en verschillende interfaces en abstracte klassen die het mogelijk maken om gemakkelijk nieuwe effecten toe te voegen.

Zoals te zien in de package *Follow Effect* moet er om een nieuw effect te creëren twee klassen worden geïmplementeerd. De eerste klasse *AbstractEffectDescriptor* is een klasse die het effect beschrijft en beheert. Zo wordt deze klasse gebruikt om op te vragen welke apparaten het effect ondersteund en hoeveel. Ook geeft deze klasse aan of het effect ondersteuning heeft voor tracking devices en hoeveel.

Verder is deze klasse verantwoordelijk voor het creëren van het effect. De tweede klasse die nodig is om een effect te creëren is de klasse *AbstractLocationEffect* of *AbstractEffect*. Het verschil tussen deze klassen is dat de klasse *AbstractLocationEffect* ondersteuning biedt voor tracking devices op deze manier kan er worden geabonneerd op locatie verandering van een of meer tracking devices. De klasse *AbstractEffect* heeft hiervoor geen ondersteuning.

Veel van de beschreven kenmerken van een effect zoals beschreven in HOOFDSTUK 7.2 zijn direct terug te zien in het ontwerp. Hier volgt een tabel met verdere uitleg.

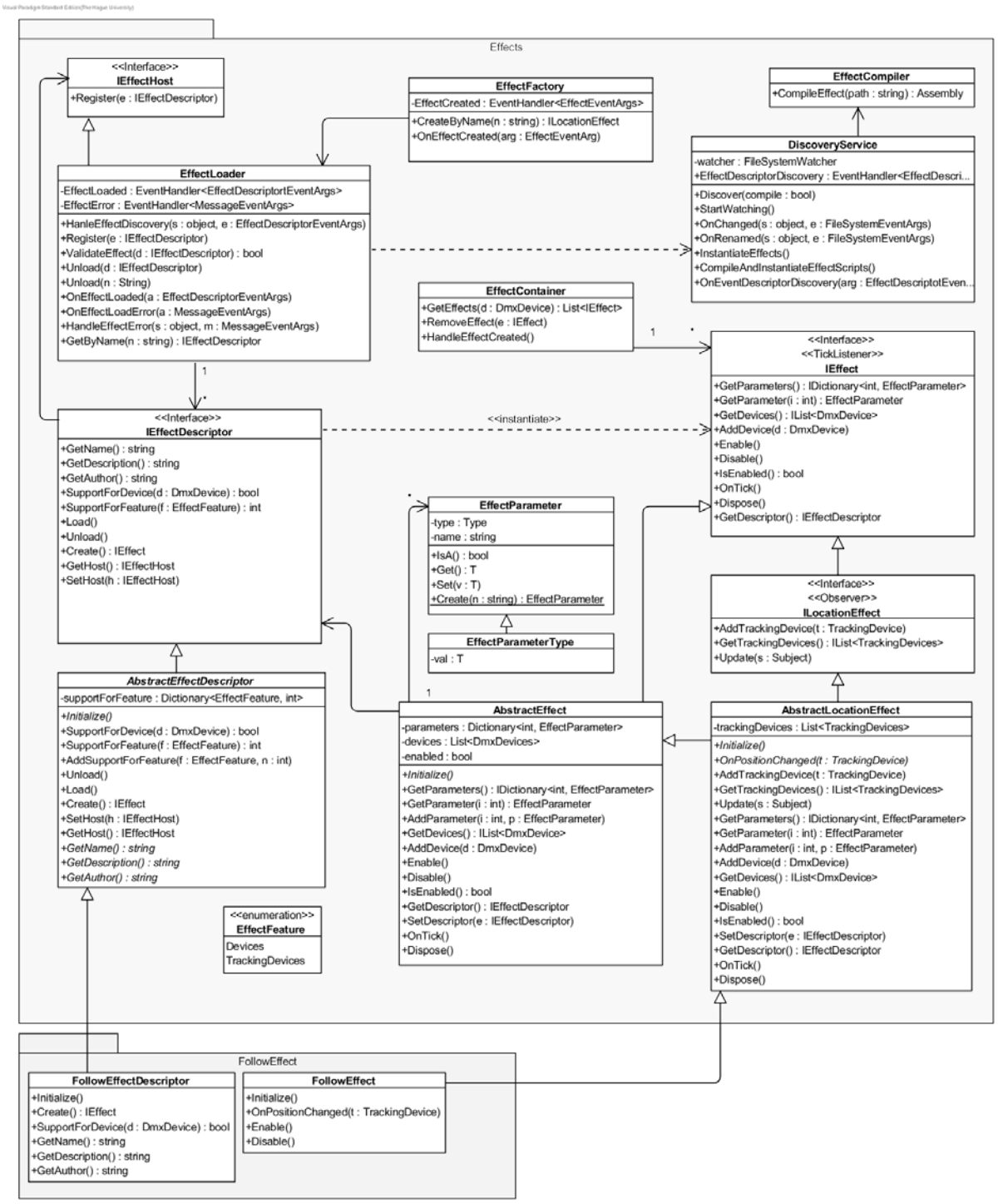
Tabel 6 Link tussen effect definitie en ontwerp

Nr.	Omschrijving
1	Hiervoor wordt de methode *SupportForDevice()* gebruikt om te bepalen of het apparaat compatible is. De methode *SupportForFeature()* wordt gebruikt om te bepalen voor hoeveel apparaten het effect ondersteuning heeft.
2	De methode *OnTick()* wordt bij elk frame dat de engine tekent aangeroepen. Deze methode kan dus worden gebruikt om 'tijd' te meten. De methode *OnPositionChanged()* wordt aangeroepen op het moment dat de positie van een gekoppeld tracking device veranderd. Zie de interface *IEffect* en *ILocationEffect* voor de methodes.
4, 5	Zie klassen *EffectParameter* en *EffectParameterType* en de methodes *GetParameters()* en *GetParameter()*.
6	Respectievelijk *Enable()* en *Disable()* van *IEffect*.
7	Respectievelijk *Load()* en *Unload()* van *IEffectDescriptor*.

Het implementeren van het ontwerp ging zeer voorspoedig. Dit kwam mede doordat de door Microsoft geleverde documentatie om C# als scriptaal in te zetten zeer uitgebreid was. Ook waren er online al tal van ontwikkelaars die C# al als scriptaal hadden ingezet en hier uitgebreide tutorial over hadden geschreven.

7.5 Sprint evaluatie

In deze sprint is er een keuze gemaakt om C# als scripttaal te gebruiken voor het schrijven van effecten. Hiernaast is er een uitbreidbaar systeem opgezet waardoor het mogelijk word om tal van verschillende typen effecten aan het systeem te kunnen toevoegen. Hierbij is ook rekening gehouden met de toekomstige user interface. De verschillende user stories die voor deze sprint stonden gepland zijn succesvol afgrond.



Figuur 20 Package Effects

8 Sprint 4 | User interface framework

In deze sprint was het tijd om te onderzoeken hoe er in Unity het beste een user interface kon worden ontwikkeld. Hiervoor moest eerst worden bepaald wat de verschillende functies van de user interface zouden worden. Tevens werd er deze sprint een dag meegelopen met de opdrachtgever bij het opzetten en inregelen van de Myo applicatie voor het Amsterdam Music Festival.

Tabel 7 Sprint planning sprint 4

Nr.	Story	Uren
37	Als ontwikkelaar wil ik onderzoeken welke functionaliteit de user interface krijgt.	14
38	Als ontwikkelaar wil ik de verschillende mogelijkheden onderzoeken voor het tonen van een user interface binnen Unity.	28
39	Als ontwikkelaar wil ik een dag meelopen met de opdrachtgever om te kijken in wat voor soort situatie de applicatie later zou kunnen worden ingezet.	8

De tweede week van de sprint werd duidelijk dat de eerst komende maandag een testdag met Sendrato kon plaatsvinden. Na een gesprek met de opdrachtgever over wat het doel van de testdag was werd duidelijk dat de applicatie nog verder moest worden uitgebreid met ondersteuning voor het inlezen van XML bestanden met daarin de configuratie van lampen. Er is toen besloten om de taken die nog stonden gepland voor deze sprint (dit waren vooral beschrijvende taken) door te schuiven naar de volgende sprint om zo plaats te maken voor de nieuwe user story. Deze zijn hieronder te lezen. Er is ook besloten om de sprint met twee dagen te verlengen zodat er tijd beschikbaar werd gemaakt om de resultaten van de testdag te beschrijven.

Tabel 8 Sprint planning sprint 4 toevoegingen

18	Als gebruiker wil ik een lamp met de daarbij horende DMX kanalen automatisch laten inladen door gebruik te maken van het des betreffende XML bestand.	14
43	Als ontwikkelaar wil ik de eerste testdag met Sendrato bijwonen.	7
44	Als ontwikkelaar wil ik de resultaten van de eerste testdag verwerken.	4

8.1 Keuze user interface framework

In eerste instantie zijn er twee opties overwogen voor het creëren van een user interface voor de applicatie. De eerste optie was om een front end applicatie te schrijven in een bestaand UI framework bijvoorbeeld met QT of JavaFX dit zijn respectievelijk twee UI frameworks in C++ en Java. Deze applicatie zou vervolgens kunnen communiceren via bijvoorbeeld een socket interface met de Unity applicatie, dit wordt ook wel IPC of Interprocess communication genoemd [42]. In eerste instantie werd namelijk gedacht dat het creëren van een UI binnen Unity vrij lastig was, het gebruik van een ander UI framework zou het creëren van de UI dan makkelijker moeten maken. Deze optie heeft echter ook een aantal zware nadelen. Zo moet er bijvoorbeeld een protocol worden opgezet zodat de twee applicaties met elkaar kunnen communiceren hiernaast zijn er nu twee applicaties die ontwikkelt en onderhouden moeten worden. Na verder onderzoek bleek dat Unity zelf ook een aantal goede oplossingen heeft voor het creëren van een UI. Aangezien er veel nadelen aan de eerste optie kleven is er gegaan voor de tweede optie. Namelijk de user interface binnen de Unity applicatie ontwikkelen.

Er zijn verschillende mogelijkheden om een UI binnen Unity te creëren. Er kan gebruik worden gemaakt van Unity's eigen UI framework of er kan gebruik worden gemaakt van een UI framework van een 3e party ontwikkelaar. Maar voordat er een gegronde keuze kon worden gemaakt moest eerst de user interface worden beschreven. Dit is gedaan aan de hand van de use case scenario's uit SPRINT 1. De

verschillende onderdelen van de UI zijn beschreven in het user interface document in BIJLAGEN E. Uit dit document volgeden een aantal punten die konden worden meegenomen bij het zoeken naar een UI framework. Hier volgt een tweetal van de punten (voor de rest kan de bijlagen worden geraadpleegd).

- De UI maakt gebruik van windows (net zoals in de bekende besturingssystemen) om de gebruiker content te laten zien. Deze windows kunnen door de gebruiker worden rond geschoven en gesloten.
- Er komen binnen de applicatie verschillende soorten lijsten voor onder anderen: tracking devices, effecten, apparaten of opnamen. Deze lijsten moeten zo nodig scrol baar zijn en moeten de data realtime weergeven, dus de content moet veranderen als de data veranderd.
- Het is een grote pre als de data en de view (die bij de data hoort) kan worden gescheiden. Hiermee wordt de UI van de applicatie makkelijker uitbreidbaar.

Het UI framework dat Unity standaard meelevert biedt een set aan standaard UI componenten. Deze componenten kunnen wordt gebruikt om geavanceerdere componenten te maken bijvoorbeeld lijsten, tabellen of windows. Standaard heeft Unity UI daar geen ondersteuning voor. Er is daarom onderzoek gedaan naar andere frameworks die mogelijk meer functionaliteit aanbieden en dus het ontwikkelen van een UI makkelijker zouden kunnen maken.

Unity biedt op zijn website een online winkel aan waar ontwikkelaars assets (plugins) voor Unity met elkaar kunnen delen. Deze asset store (zoals Unity hem zelf noemt) zit vol met toevoegingen voor Unity van hoge resolutie 3D modellen tot UI frameworks tot complete games. Zo zijn er ook tal van UI frameworks te vinden in deze online winkel. Voor dit onderzoek zijn de meest gebruikte frameworks onderzocht. Hieronder volgen twee van de in totaal zeven onderzocht UI frameworks.

Tabel 9 Userinterface frameworks

NGUI [43]	
Waardering	5 (2800 reviews)
Prijs	$95 (per seat)
Unity versie	4.6.7 of hoger
Datum	10 december 2011 – 7 oktober 2015
Omschrijving	NGUI ook wel next-gen UI is een van de oudste UI frameworks voor Unity. Het biedt een tal van standaard elementen om je snel op weg te helpen met het ontwikkelen van een UI. De maker van NGUI heeft een tijd bij Unity gewerkt voor het creëren van de nieuwe Unity UI. De nieuwe Unity UI lijkt daarom erg op die van NGUI. Verder heeft NGUI de grootste community van de verschillende onderzochte frameworks.

MarkUX [44]	
Waardering	5 (16 reviews)
Prijs	$30 (per seat)
Unity versie	4.6.0 of hoger
Datum	26 juni 2015 – 16 oktober 2015
Omschrijving	MarkUX is een relatief nieuw UI framework voor Unity. Achter de schermen maakt MarkUX gebruik van de nieuwe Unity UI API. De UI wordt gecreëerd in de vorm van XML bestanden en kan door middel van data bindings met de applicatie logica communiceren. MarkUX biedt een tal van verschillende standaard UI elementen en deze kunnen gemakkelijk worden uitgebreid.

Tijdens het onderzoek bleek dat de UI frameworks grofweg in twee groepen waren onder te verdelen namelijk frameworks die gebruik maken van Unity's (nieuwe) UI framework en framework die dit niet doen. Voor Unity versie 4.6 beschikte Unity over een omslachtige en niet efficiënte manier voor het creëren van een user interface. Er waren toen der tijd een aantal UI frameworks gecreëerd die het creëren van een UI binnen Unity makkelijker maakte hier is NGUI er een van. De ontwikkelraar achter NGUI is door Unity een voor een bepaalde periode ingehuurd om mee te helpen met het ontwikkelen van het nieuwe UI framework (dit nieuwe framework wordt ook wel UGUI, Unity GUI genoemd).

Om de keuze voor een framework te kunnen maken werd er gebruik gemaakt van de hierboven beschreven punten. De plugins konden niet getest worden aangezien er geen gratis versies beschikbaar waren. De keuze kon alleen gemaakt worden door te kijken naar de ervaring van andere mensen met het framework en de documentatie van het framework. Veel van de onderzocht frameworks boden niet de gewenste functionaliteit in de zin dat ze wel extra componenten aanboden maar deze waren van onvoldoende kwaliteit of het waren er te weinig voor de prijs die betaalt moest worden voor het framework. Hiernaast waren er een aantal frameworks die simpel weg verouderd waren. Deze frameworks waren relevant voor Unity versie 4.6 maar door introductie van het nieuwe UI systeem was het de investering niet waard. Dit zelfde gelde voor NGUI.

De frameworks MarkUX en NoesisGUI 1.1 bleven als kandidaten over. MarkUX is met zijn dertig euro een kleine investering en biedt veel waarde voor zijn geld. Het framework maakt gebruik van het MVVM [45] patroon om data en view te scheiden. De UI wordt gedefinieerd in XML bestanden dit zorgt er voor dat de UI gemakkelijk is aan te passen. Hiernaast biedt het framework een tal van standaard componenten (waaronder: windows, tabellen en lijsten) en zijn deze gemakkelijk uit te breiden. De keuze is dus ook gevallen voor dit framework. Hiervoor heeft het framework NoesisGUI 1.1 (beschreven in het user interface document) het van wege zijn hoge licentie kosten van 250 euro moeten afleggen tegen MarkUX. Er is dus gekozen om de user interface te ontwikkelen gebruik makend van het framework MarkUX. Dit framework zal de komende sprints gebruikt gaan worden.

8.2 Instellingen inladen uit XML

Een veel gebruikte lichttafel voor de optredens van Armin van Buuren is de GrandMA. Met deze lichttafel is de lichtman in staat om de lampen tijdens de show aan te sturen. In deze lichttafel bevinden zich de instellingen van de lampen, ook wel fixtures genoemd. Deze instellingen kunnen worden geëxporteerd naar XML bestanden zodat ze gebruikt kunnen worden door andere programma's. De interessantste informatie die in dit soort XML bestanden te vinden is zijn de beschrijvingen van de verschillende DMX kanalen zoals bijvoorbeeld de naam en het adres van het kanaal.

De applicatie moest ook ondersteuning krijgen voor het inladen van deze XML bestanden. Voorheen moesten deze instellingen namelijk met de hand worden ingevuld. Tot nu toe was dat geen probleem aangezien er nog niet met veel lampen werd gewerkt maar voor de testdag was het gewenst om de instellingen uit de XML bestanden te halen. Hiervoor zijn een tweetal nieuwe klassen ontworpen namelijk de *FixtureLoader* en de *Fixture*. De *FixtureLoader* parsed de verschillende XML bestanden en zet deze om naar instanties van de *Fixture* klassen. De *Fixture* klasse bezit informatie onder de lamp zoals de naam en de verschillende kanalen van de lamp. Een verandering ten opzichte van het klassendiagram uit FIGUUR 16 is dat een DmxDevice niet langer meer direct een relatie heeft met een *DmxChannel* maar nu indirect via de Fixture klasse de kanalen bereikt. Voor het ontwerp van deze klassen kan hoofdstuk 3.4 van het ontwerprapport geraadpleegd worden.

8.3 Testdag

Deze sprint vond de eerste testdag met Sendrato plaats. Deze testdag was zoals al eerdere beschreven niet van te voren in de sprint gepland. Hierdoor moesten een aantal taken doorschuiven naar de volgende sprint om plaats te maken voor de testdag en de daarbij horende voorbereidingen.

De testdag vond plaats in het Amsterdam theater. Nadat het systeem van Sendrato was opgezet en gekalibreerd kon de applicatie met het systeem worden verbonden. Al snel bleek dat de X, Y en Z coördinaten die Sendrato doorstuurde niet goed door de applicatie werden geïnterpreteerd. Waardoor de positie van het tracking device in de 3D wereld niet klopte. Nadat dit probleem was opgelost kon het volg effect worden getest. Dit effect bevatte een aantal problemen waardoor de volglamp niet naar de juiste positie keek. Zo klopte de implementatie van de *LookAt()* methode van de klasse *MovingHead* niet. Deze is vervolgens aangepast. De volgende zaken zijn uiteindelijk bereikt op de eerste testdag:

- Een tracking device weergeven in de 3D wereld en deze de tracker laten volgen.
- Een moving head zodanig aansturen dat deze met zijn lichtstraal de tracker volgt.
- Een moving head zodanig aansturen dat deze van kleur verandert zodra er een bepaald gebied wordt binnengegaan met de tracker.

8.3.1 Bevindingen

De resultaten en bevindingen van de testdag zijn beschreven in de bijlagen van het hardware testrapport uit BIJLAGEN D. De bevindingen van de testdag zijn samen met de opdrachtgever doorlopen en geprioriteerd. De twee belangrijkste bevindingen van de testdag zijn hieronder te vinden. Deze bevindingen kunnen in de komende sprints worden omgezet naar bruikbare oplossingen.

1. Tijdens de testdag bleek dat de precieze betekenis van de X, Y en Z coördinaten per opstelling kan veranderen. Zo betekend een positieve Z waarde de ene keer dat een object voor je ligt en de andere keer dat een object achter je ligt. Ook zouden de assen geheel van betekenis kunnen veranderen. Dus dat een X as de diepte aangeeft in plaats van de lengte. Het is van belang om deze waarden te converteren naar voor de applicatie bruikbare waarden. De applicatie werkt namelijk altijd met het zelfde coördinaten systeem.

2. De applicatie bood nog geen ondersteuning voor het automatisch weergeven van de anker punten (antennes) in de applicatie. Het toevoegen van de anker punten is echter wel gewild omdat deze referentie punten vormen voor de gebruiker.

8.4 Sprint evaluatie

Deze sprint is er onderzoek gedaan naar de user interface van de applicatie. Er is gekeken naar de verschillende functionaliteiten van de applicatie hieruit zijn een tal van nieuwe user stories ontstaan. De user stories zijn toegevoegd aan de product backlog. Hiernaast is er onderzocht hoe de user interface het beste kan worden ontwikkeld. Hiervoor zijn verschillende UI frameworks onderzocht, uiteindelijk is er gekozen om de UI in samenwerking met het MarkUX framework te implementeren. Ook is er deze sprint een dag meegelopen met de opdrachtgever om zo kennis te maken met de omgeving waarin de applicatie wordt ingezet.

Tevens is er deze sprint een testdag met Sendrato geweest. Hieruit zijn een aantal nieuwe user stories ontstaan die zijn toegevoegd aan de product backlog. Een aantal geplande werkzaamheden zijn door deze testdag door geschoven naar de volgende sprint. Dit zijn vooral de beschrijvende taken dus het bijwerken van het afstudeerverslag en het verder uitwerken van het UI document.

9 Sprint 5 | Documentatie

In deze sprint wordt user story 38 uit de vorige sprint afgemaakt. Deze story kon vanwege de testdag niet worden afgemaakt. Verder richt deze sprint zich op het aanvullen van het hardware testrapport met de resultaten van de testdag. Hiernaast worden verschillende documenten die belangrijk zijn voor het afstudeerdossier aangevuld en aangepast waar nodig het gaat hierbij om het afstudeerverslag en het scrum rapport. Deze sprint wordt er namelijk een eerste concept versie van het afstudeer dossier naar de begeleidende docenten gestuurd. De vorige sprint duurde door de testdag twee dagen langer. Hierdoor zijn er in deze sprint twee dagen minder beschikbaar om in te plannen.

Tabel 10 Sprint planning sprint 5

Nr.	Story	Uren
38	Als ontwikkelaar wil ik de verschillende mogelijkheden onderzoeken voor het tonen van een user interface binnen Unity.	12
76	Als opdrachtgever wil ik dat de testdag wordt verwerkt in het hardware rapport.	7

Notitie: In bovenstaand tabel is de tijd die aan het afstudeer dossier is gewerkt niet weergegeven.

Het schrijven van de documentatie ging sneller dan verwacht hierdoor was de sprint eerder klaar dan gepland. Er is toen besloten om alvast een aantal user stories die gepland stonden voor de volgende sprint te implementeren. Deze user stories zijn hieronder te vinden.

58	Als gebruiker wil ik een overzicht hebben van de verschillende tracking devices binnen de applicatie.	10
75	Als gebruiker wil ik inzicht in verschillende parameters van de applicatie zoals: het aantal frames per seconde en het aantal locatie updates per seconde.	4

9.1 Hardware testrapport

Het hardware testrapport heeft twee doelen: het beschrijven van de werking van de hardware die Sendrato levert en het beschrijven en bijhouden van de verschillende testcases en de bijhorende resultaten. Het rapport was voor een groot deel al opgezet in SPRINT 1. Deze sprint is het testrapport uitgebreid met de bevindingen van de eerste testdag.

Aan de hand van de verschillende beschreven test cases is de werking van de hardware gevalideerd. Er waren een aantal cases die tijdens deze eerste testdag niet konden worden getest. De resultaten van de testdag waren echter veelbelovend waardoor er niet wordt verwacht dat deze testen problemen zullen vormen. Er was echter wel een test waaraan (nog) niet voldaan is. Dit was test case 6 deze test beschrijft:

Het systeem moet niet te verstoren zijn met materialen of omstandigheden die tijdens een show aanwezig (kunnen) zijn.

Op dit moment was het signaal van het tracking device nog te verstoren door een ‘body block’ waardoor het signaal wegviel. Hier moet de volgende testdag verder onderzoek naar worden gedaan. Zie onderstaand tabel voor het resultaat van test case 6.

Tabel 11 Hardware testrapprot test case 6

Testcase	Voldaan	Opmerking
T6	N	Het systeem was te verstoren door een 'body block'. Een body block vind plaats als je lichaam zich tussen het tracking device en een van de antennes bevind waardoor het tracking device dus niet meer kan communiceren met de desbetreffende antenne. Tijdens de testdag moet er dus worden rondgelopen met het tracking device boven je hoofd.

9.2 Sprint evaluatie

De sprint was begonnen met het afmaken van user story 38 die door de testdag in de vorige sprint niet kon worden afgemaakt. Ook zijn de resultaten van de testdag verwerkt in het hardware testrapport. Verder is de documentatie van het project verder uitgebreid en als concept versie naar de begeleidende docenten op gestuurd.

Het bijwerken van de documentatie ging sneller dan verwacht hierdoor bleef er tijd over in de sprint. Deze tijd is opgevuld met twee user stories die stonden gepland voor de volgende sprint. Dit waren user stories die betrekking hadden op de user interface. In samenwerking met MarkUX is de eerste versie van de GUI tot stand gekomen.

10 Sprint 6 | User interface implementeren

Deze sprint staat in het teken van het implementeren van de bevindingen van de testdag. Zo wordt er gewerkt aan een manier om de antennes binnen de applicatie weer te geven en de opzettijd van de applicatie te versnellen. Tevens wordt er deze sprint voor het eerst 'echt' gewerkt met het UI framework MarkUX dat in SPRINT 4 is gekozen. In de vorige sprint was al een begin gemaakt met het creëren van een user interface voor de tracking devices deze werd deze sprint verder afgemaakt.

Tabel 12 Sprint planning sprint 6

Nr.	Story	Uren
10	Als gebruiker wil ik zien waar de verschillende antennes zich bevinden in de 3D ruimte.	14
73	Als gebruiker wil ik een grid zien waarover de verschillende tracking devices zich bewegen.	4
74	Als gebruiker wil ik dat ik snel het systeem van Sendrato kan koppelen met de applicatie ongeacht de opstelling van de hardware.	10
42	Als gebruiker wil ik dat de manier hoe een tracking device wordt geüpdatet (bijvoorbeeld door een muis of door een tracker) kan veranderen zonder dat ik hiervoor de effecten opnieuw moet instellen die aan het tracking device zijn gekoppeld.	20
56	Als gebruiker wil ik een tracking device kunnen toevoegen aan de applicatie via de UI.	4
58	Als gebruiker wil ik een overzicht hebben van de verschillende tracking devices binnen de applicatie.	2
59	Als gebruiker wil ik de input methode van een tracking device binnen de UI kunnen veranderen.	8

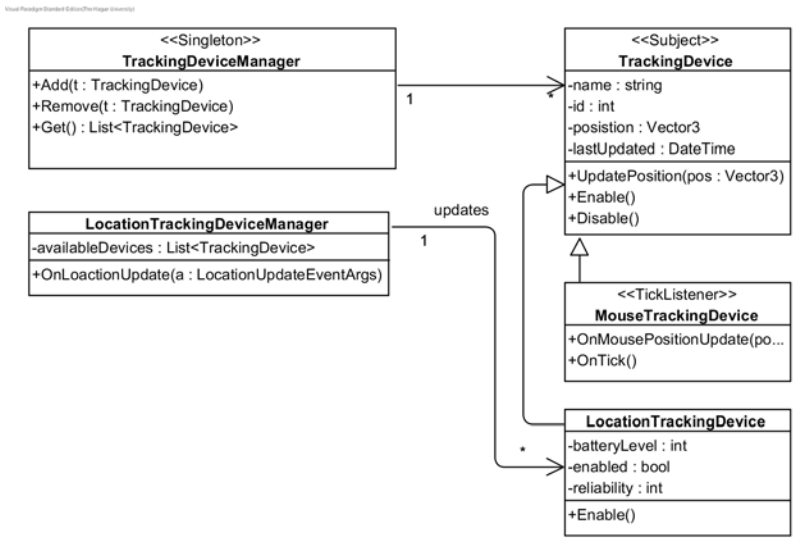
Het implementeren van de bovenstaande user stories ging sneller dan verwacht. Dit kwam deels doordat er extra tijd was ingeschat voor het maken van de user interface aangezien dit onderdeel nieuw was. Maar deze tijd bleek niet nodig te zijn. Daardoor bleef er extra tijd over. Er is gekozen om een handleiding te schrijven voor het ontwikkelen van een effect.

33	Als effect schrijver wil ik een handleiding over hoe ik een effect kan schrijven voor de applicatie.	10

10.1 TrackingDevice update methode loskoppelen

In SPRINT 2 is de eerste versie van het ontwerp gemaakt. Dit ontwerp maakte gebruik van verschillende typen *TrackingDevices* zoals te zien is in FIGUUR 21. Elk subtype werd op zijn eigen manier geüpdatet (van positie veranderd) zo volgde het *MouseTrackingDevice* de positie van de muis in de 3D wereld en volgde de *LocationTrackingDevice* de positie van een fysieke Sendrato sensor. Tevens maakte de *PlaybackManager* gebruik van een *TrackingDevice* om een opname te kunnen afspelen.

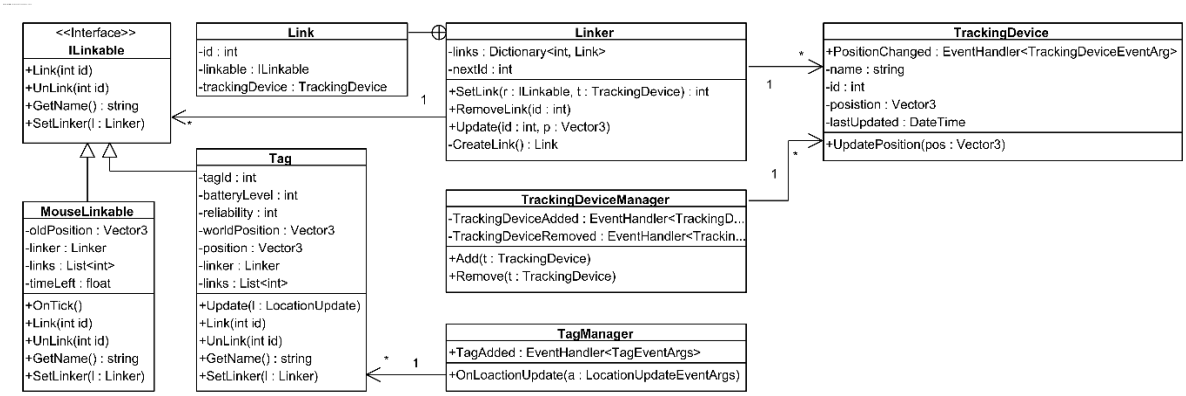
Tijdens het opzetten van het user interface rapport kwam echter naar voren dat deze implementatie niet voldoende vrijheid bood. Zo was het niet mogelijk om de wijze waarop een *TrackingDevice* werd geüpdatet te veranderen zonder een nieuwe instantie van een *TrackingDevice* aan te maken. Hiernaast moesten effecten of andere klassen die een referentie hadden naar het 'oude' TrackingDevice worden geüpdatet zodat ze het nieuw gecreëerde *TrackingDevice* gebruikten. Dit was niet gewenst. Er moest daarom een nieuwe manier ontwikkeld worden waarmee de wijze waarop een *TrackingDevice* werd geüpdatet kon worden aangepast zonder een *TrackingDevice* te verwijderen en een nieuwe te creëren.



Figuur 21 Ontwerp: TrackingDevice eerste versie

Er zijn op dit moment drie verschillende manieren waardoor een *TrackingDevice* van positie kan veranderen namelijk: het veranderen van de positie van de muis, het afspelen van een opnamen of het ontvangen van een locatie update van een van de trackers van Sendrato. In de toekomst zouden er nieuwe methodes kunnen worden toegevoegd.

In de vorige implementatie waren de klassen die een *TrackingDevice* wilden updaten direct gekoppeld aan het desbetreffend *TrackingDevice*. Hierdoor was er om een tracking device te kunnen updaten kennis nodig over het bestaan van de *TrackingDevice* klassen en de betreffende implementatie daarvan. Dit is echter niet gewenst aangezien dit het in de toekomst moeilijker maakt om nieuwe manieren van updaten toe te voegen. Dit nadeel in het vorige ontwerp moest in het nieuwe ontwerp worden opgelost naast het al hierboven beschreven probleem.

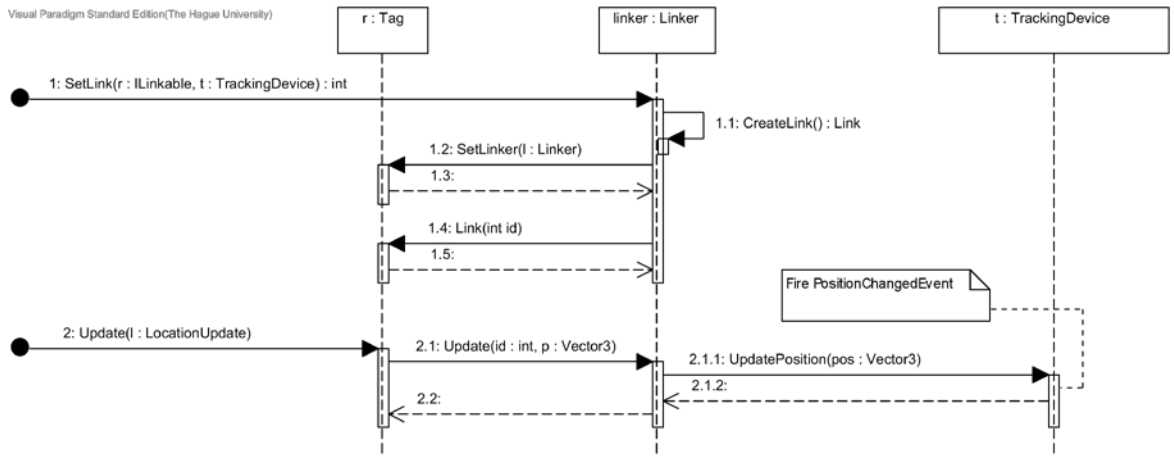


Figuur 22 Klassendiagram: tracking device linker

Er moest dus een klasse ontworpen worden die de verschillende update methodes kan verbinden met *TrackingDevices*. Hierbij is het van belang dat het gemakkelijk moet zijn om in de toekomst nieuwe update methodes toe te voegen. Deze functionaliteit is ontworpen in de *Linker* klassen zie FIGUUR 22. Deze klasse verbindt (linkt) een update methode aan een *TrackingDevice*. De verschillende update methodes implementeren de ILinkable interface. Deze interface wordt door de *Linker* gebruikt om de verschillende Links te beheren. Een klasse die de *ILinkable* interface implementeert heeft geen kennis over welk *TrackingDevice* er wordt geüpdatet. Het enige waar de klasse kennis over heeft is een unieke id van de link. Met deze id kan de klasse de link updaten doormiddel van de *Update()* methode van de Linker.

De *LocationTrackingDeviceManager* en de *LocationTrackingDevice* hebben plaats gemaakt voor de *TagManager* en de *Tag*. Een *Tag* representeert een Sendrato tracker. Op het moment dat een *TrackingDevice* wordt gelinkt met een *Tag* zal het *TrackingDevice* de locatie van de *Tag* overnemen.

Zie onderstaand sequentiediagram voor het updaten van een tracking device. Hierbij wordt eerst de betreffende *Tag* gelinkt met het *TrackingDevice* via de methode *SetLink()*. Vervolgens wordt het *TrackingDevice* geüpdatet als er een nieuwe *LocationUpdate* wordt ontvangen door de *TagManager*.



Figuur 23 Sequentiediagram: tracking device updaten

10.2 Eerste versie van de UI gemaakt

In deze sprint is de user interface van de applicatie verder uitgewerkt. Hierbij is het framework dat in SPRINT 4 is gekozen gebruikt. MarkUX maakt zoals al eerder vermeld gebruik van het MVVM (Model-view-viewmodel) ontwerp patroon dit is een variatie op het MVC (Model View Controller) patroon.

In de volgende alinea's wordt er gesproken van een view. Een view is een component van de user interface. Dit kan een label zijn maar dit kan ook een compleet venster zijn met daarin verschillende andere views zoals bijvoorbeeld: stukken tekst, knoppen of invoer velden. In MarkUX wordt de view geschreven in XML. Om gedrag aan een view toe te voegen is er een *ViewModel* nodig. De *ViewModel* zorgt er voor dat een view wordt voorzien van data en dat er gedrag aan gebruiker acties wordt toegevoegd bijvoorbeeld als de gebruiker op een knop drukt.

Doordat de user interface in XML wordt beschreven is het later aanpassen van de lay-out van de user interface gemakkelijk. Een ander voordeel van het gebruiken van XML voor het beschrijven van de user interface is dat ontwerpers hier makkelijker mee overweg kunnen. Een ontwerper hoeft weinig te weten van de achter liggende programmatuur om de user interface te kunnen creëren of aan te passen. Dit is binnen dit project nog niet zo zeer van belang aangezien er door een persoon wordt gewerkt aan het project. Maar in de toekomst wordt dit zeker relevant. Binnen dit project wordt er niet gefocust op een grafisch aantrekkelijke user interface maar op een functionele user interface.

De eerste versie van de user interface is tot stand gekomen in samenwerking met het user interface rapport. In dit rapport werd beschreven dat er gebruik zou worden gemaakt van verschillende beweegbare vensters (windows) om de verschillende soorten content te laten zien. Er is begonnen met het implementeren van de verschillende vensters in XML. Nadat de vensters voor het grootse deel werkte is er gedrag aan toegevoegd door de betreffende *ViewModels* te implementeren. Deze *ViewModels* communiceerde met de andere klassen in het systeem om zo de *TrackingDevices* en *Tags* te kunnen op vragen. Het implementeren van deze functionaliteit ging sneller dan verwacht er waren echter wel wat problemen met het live herladen van de lijsten meer hierover in het volgende hoofdstuk.

Zie FIGUUR 24 en FIGUUR 25 voor in game screenshots van de user interface. Het eerste figuur geeft het tracking device venster aan. Vanuit dit venster zijn nieuwe tracking devices aan te maken en is het mogelijk om de positie van de devices af te lezen. Het tweede figuur toont een venster met daarin de verschillende tags in het systeem. Deze lijst wordt automatisch bijgewerkt op het moment dat er een nieuwe tag aan het systeem van Sendrato wordt toegevoegd. Tevens is het vanuit dit venster mogelijk om een tag met een bepaald tracking device te koppelen en te ontkoppelen.

Figuur 24 User interface: tracking device venster

Figuur 25 User interface: tag venster

10.3 Lijsten live herladen

Zoals te zien is in TABEL 9 is MarkUX een relatief nieuw framework. Doordat het nog niet zolang bestaat kan het gebruik maken van de nieuwste technieken die Unity en C# te bieden hebben. Een nadeel van een jong framework kan echter ook zijn dat nog niet alle onderdelen even volwassen zijn.

Zo wordt er binnen de applicatie veel gewerkt met lijsten een aantal van deze lijsten moeten live worden bijgewerkt. Zo moeten de posities van de verschillende tracking devices live worden weergegeven. De manier waarop MarkUX het verversen van een lijst heeft geïmplementeerd is echter niet efficiënt genoeg waardoor het verversen van de lijst relatief veel tijd kost. Dit vormt een probleem op het moment dat een lijst vaak moet worden bijgewerkt wat bij het weergeven van de posities van de tracking devices het geval is deze worden namelijk meerdere keren per seconden geüpdatet. Een deel van dit probleem kan worden opgelost door de lijsten minder vaak bij te werken maar dit is enkel een work around en geen echte oplossing van het probleem. Er is daarom gekozen om voor nu een herlaad (refresh) knop te implementeren die handmatig de data herlaad. Tevens is de ontwikkelaar op de hoogte gesteld [46]. De ontwikkelaar van MarkUX heeft aangegeven dit ‘probleem’ in de volgende versie van het framework op te lossen.

10.4 Assen configureren

Na de testdag die plaats vond aan het einde van SPRINT 4 waren er een aantal bevindingen gedaan. Een van die bevindingen was dat de betekenis van de X, Y en Z coördinaten die het systeem van Sendrato doorstuurt kunnen veranderen van betekenis. De betekenis van de X, Y en Z coördinaten hangt namelijk af van de positie en oriëntatie van de antennes in de opstelling. Het moet dus mogelijk zijn om de betekenis van de X, Y en Z waarden zo om te zetten dat deze kloppen binnen de (Unity) applicatie.

Unity maakt gebruik van het linkshandige coördinaten systeem[W6]. Het systeem dat Sendrato gebruikt hangt (zoals al eerder gezegd) af van de opstelling van de antennes. Zo kan het bijvoorbeeld voorkomen dat Sendrato de Y as gebruikt om de diepte in de wereld door te sturen terwijl de Y as binnen Unity staat voor de hoogte in de wereld. Hierdoor kunnen de coördinaten niet zomaar aan elkaar ‘gekoppeld’ worden. Er zijn een aantal verschillende parameters die per as moeten worden ingesteld dit zijn:

- Welke Sendrato as moet worden gelinkt aan de Unity as. Zo is het in bovenstaand voorbeeld gewenst dat de Y as die Sendrato gebruikt voor het doorsturen van de diepte wordt gekoppeld aan de X as van Unity aangezien deze as door Unity wordt gebruikt voor het weer geven van de diepte.
- Een waarde waarmee de ontvangen coördinaat van Sendrato moet worden gedeeld. Sendrato werkt namelijk met millimeters terwijl de applicatie werkt in meters.
- Een bepaalde offset die bij het coördinaat wordt opgeteld of afgetrokken.
- Of het ontvangen coördinaat moet worden geïnverteerd.

Om de gewenste functionaliteit te kunnen implementeren waren er een aantal aanpassing aan het systeem nodig. Zo moest er een user interface worden gemaakt waardoor de gebruiker in staat werd gesteld om de verschillende parameters per as te kunnen aanpassen. Tevens moest de klasse die verantwoordelijk was voor het parsen van de Sendrato pakketjes worden uitgebreid zodat deze de locatie data op de juiste manier kan converteren. Zie onderstaand figuur voor de user interface die de gebruiker in staat stelt om de verschillende parameters te kunnen inzien en aanpassen.

Figuur 26 User interface: configuratie venster voor assen

Voor het converteren van de locatie data is een nieuwe klasse ontworpen genaamd: SendratoAxisConverter maar om de locatie data te kunnen converteren moest de klasse bij de verschillende parameters kunnen komen die de gebruiker via de user interface had ingesteld. Hiervoor is de *SettingsManager* klasse ontworpen die deze parameters kon opslaan. Deze klasse is zo ontworpen dat er ook andere instellingen (parameters) mee kunnen worden opgeslagen. Er is onderzoek gedaan naar de verschillende mogelijkheden voor het opslaan van instellingen binnen de applicatie. Er kwam naar voren dat er twee manieren zijn om instellingen op te slaan namelijk:

1. C# heeft een ingebouwde klasse die het mogelijk maakt om instellingen te kunnen opslaan.

2. Door gebruik te maken van een eigen implementatie die bijvoorbeeld gebruik maakt van een bestand waar de instellingen naar weg worden geschreven in bijvoorbeeld een XML of INI vorm zodat ze later weer kunnen worden ingeladen.

De eerste manier zou de minste tijd kosten om te implementeren deze manier is dus ook verder onderzocht. Deze bleek echter niet te werken in samenwerking met Unity. Dit omdat deze klasse niet beschikbaar is binnen Mono [47]. Mono is een open source implementatie van het .NET framework die door Unity wordt gebruikt. Het was daarom noodzakelijk om een eigen klassen te implementeren. Hiervoor is gekeken naar de implementatie van de instellingen klasse van C#. Er is getracht om de klassen zo op te zetten dat het mogelijk is om een tal van verschillende data typen op te slaan. Voor nu kan de klassen ints, floats, bools en enums opslaan. De instellingen worden opgeslagen in een XML bestand er is gekozen voor XML omdat er in SPRINT 4 al een aantal methodes waren geschreven die het werken met XML bestanden versimpelde hierdoor kon de klasse sneller worden geïmplanteerd.

10.5 Effect handleiding

Het implementeren van de user interface ging sneller dan verwacht hierdoor bleef er tijd over om een handleiding te schrijven deze is te vinden in BIJLAGEN G. In deze handleiding wordt beschreven hoe het ontwikkelen van een effect in zijn werk gaat. Deze handleiding beschrijft de verschillende stappen die nodig zijn om de ontwikkelomgeving zodanig op te zetten zodat er een effect kan worden ontwikkeld. Aan de hand van beschrijvingen en code voorbeelden legt de handleiding de verschillende mogelijkheden uit. Hier volgen twee beschrijvingen die uitleggen hoe een bepaalde methode en een variabelen gebruikt moeten worden. Voor alle beschrijvingen kan de handleiding geraadpleegd worden.

10.5.1 OnPositionChanged()

Template:	void OnPositionChanged(TrackingDevice t)
Omschrijving:	Wordt aangeroepen op het moment dat een tracking device van positie veranderd.
Opmerking:	Wordt alleen aangeroepen als het effect actief is (*Enabled*)
Voorbeeld:	

In onderstaand voorbeeld worden alle moving heads die zijn gekoppeld met het effect geüpdatet zodat ze wijzen naar de (nieuwe) positie van het tracking deivce. Hierbij is er van uitgegaan dat alle *devices* van de klassen *MovingHead* zijn.

```
public override void OnPositionChanged(TrackingDevice trackingDevice) {
    Vector3 pos = trackingDevice.position;

    foreach (MovingHead movingHead in devices) {
        movingHead.LookAt(pos);
    }
}
```

10.5.2 Devices

Template:	List<DmxDevice> devices
Omschrijving:	Een lijst met alle gekoppelde apparaten.
Opmerking:	-
Voorbeeld:	

Als het effect ondersteuning heeft voor Devices (zie *AddSupportForFeature()*) kunnen deze worden benaderd door gebruik te maken van de *devices* variabelen die binnen het effect toegankelijk zijn. Hieronder staan twee voorbeelden voor het aanpassen van apparaten.

```
foreach(DmxDevice d in devices) {
    d.SetValue(DmxChannel.RED, 0.5F);
}
```

```
foreach(MovingHead h in devices) {
    h.LookAt(Vector3.zero);
}
```

10.6 Sprint evaluatie

In deze sprint is er begonnen met het implementeren van een user interface aan de hand van het framework MarkUX dat in SPRINT 4 is gekozen. Hiernaast is het makkelijker geworden om het systeem van Sendrato te koppelen met de applicatie door een aantal instellingen in de applicatie aan te passen. Tevens zijn delen van de tracking package herschreven zodat het makkelijker werd om de manier waarop een tracking device wordt geüpdatet te veranderen. Ook worden nu de anker punten die het systeem van Sendrato doorsturen weergegeven in de applicatie.

Alle user stories die aan het begin van deze sprint waren in gepland zijn succesvol geïmplementeerd. Er bleef nog tijd over om een handleiding voor het schrijven van een effect te maken. Deze handleiding kan worden gebruikt tijdens het ontwikkelen van een effect voor het systeem.

11 Sprint 7 | Positie en rotatie aanpassen

Deze sprint stond in het teken van het aanpasbaar maken van de positie en of rotatie van een lamp door de gebruiker. Al vanaf SPRINT 1 bleek dat deze wens er was. Na de eerste testdag werd deze wens verder versterkt. Er is daarom besloten om deze functionaliteit in deze sprint te implementeren. De user story die deze functionaliteit beschreef was reeds al beschreven in SPRINT 4 in het User Interface document. Zie het tabel hieronder voor de desbetreffende user story.

47	Als gebruiker wil ik de mogelijkheid om de verschillende instellingen van een lamp (zoals bijvoorbeeld de positie en de rotatie) te kunnen aanpassen via de UI.	UI Document

De bovenstaande user story is ten tijden van schrijven globaal opgeschreven. Met de kennis die er nu is kan de story specifieker worden beschreven. Hierbij is de story in vier verschillende (sub) stories te verdelen namelijk:

79	Als gebruiker wil ik vanuit de applicatie de rotatie van een apparaat aanpassen.	User Story 47
80	Als gebruiker wil ik vanuit de applicatie de positie van een apparaat aanpassen.	User Story 47
81	Als gebruiker wil ik vanuit de applicatie de DMX instellingen van een lamp veranderen.	User Story 47
82	Als gebruiker wil ik dat de instellingen van een apparaat worden opgeslagen zodat ik handmatige bijstellingen niet telkens opnieuw hoef te doen.	User Story 47

Het aanpassen van de DMX instellingen en het opslaan van de instellingen had een lagere prioriteit dan het aanpasbaar maken van de positie en of rotatie van een lamp. Dit omdat het aanpassen van DMX instellingen niet vaak voorkwam en het opslaan van de instellingen meer betrekking had op het configuratie sub onderdeel. In SPRINT 3 is reeds aangegeven dat er niet op dit onderdeel wordt gefocust. Daarom zal er deze sprint worden gefocust op het aanpassen van de positie en of rotatie van een lamp. Deze sprint vindt ook het tussentijds assessment plaats er is in deze sprint daarom meer tijd voor het maken en bijwerken van de documentatie gereserveerd.

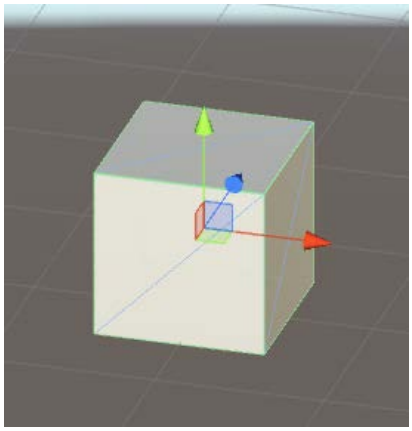
Tabel 13 Sprint planning sprint 7

Nr.	Story	Uren
77	Als effectschrijver wil ik dat als ik de pan en tilt van een moving head in de code aanpas dit ook zichtbaar wordt in de 3D wereld (fix).	4
79	Als gebruiker wil ik vanuit de applicatie de rotatie van een apparaat aanpassen.	20
80	Als gebruiker wil ik vanuit de applicatie de positie van een apparaat aanpassen.	20

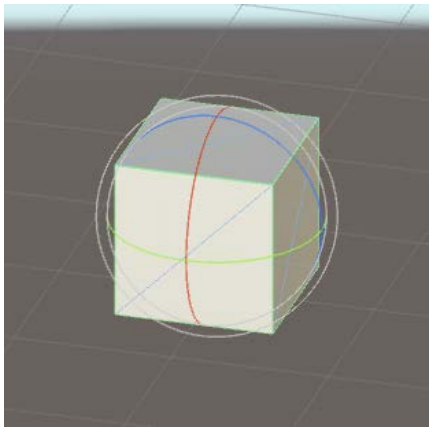
11.1 Positie en of rotatie aanpassen

In deze sprint werd de functionaliteit geïmplementeerd waardoor het mogelijk zou moeten worden om de positie en of rotatie van een apparaat te kunnen aanpassen. Zoals al in SPRINT 1 te lezen was is het van belang dat de positie en rotatie van de lampen (apparaten) in de applicatie correct zijn ingevuld voor het goed laten werken van effecten. De exacte positie en rotatie van een lamp zijn over het algemeen niet beschikbaar de globale positie echter wel. Deze zijn over het algemeen te vinden in het show plan. De gebruiker van de applicatie moet dus de mogelijkheid hebben om de positie en rotatie van een lamp te kunnen aanpassen om zo handmatig de juiste positie van de lamp te vinden. In de toekomst moet de locatie worden bepaald door het systeem dat is beschreven in SPRINT 1.

Het aanpassen van de positie of rotatie van objecten in een 3D wereld is niet nieuw. In de meeste 3D programma's zit zulke functionaliteit ingebouwd. Zo zit dit ook ingebouwd in de Unity Editor[W5]. Door binnen de Unity Editor op een object in de 3D wereld te klikken worden afhankelijk van de status van de applicatie (of de gebruiker de rotatie of de positie van een object wilt aanpassen) de zaken in FIGUUR 27 voor het aanpassen van de positie en FIGUUR 28 voor het aanpassen van de rotatie zichtbaar. Waarna de positie of rotatie van een object kan worden aangepast. Het idee was om deze functionaliteit te kopiëren naar de applicatie. Het aanpassen van de positie of rotatie van een object met een muis is echter niet in elk scenario even gemakkelijk. Zo is het lastig om tegelijker tijd op de positie van de muis te letten en te letten op de lamp in de (fysieke) wereld. Het gebruik van een toetsenbord kan dan handiger zijn. Er is daarom in samenspraak met de opdrachtgever gekozen om het veranderen van de positie en rotatie van een lamp zowel mogelijk te maken met de muis als met het toetsenbord.



Figuur 27 Positie van een object binnen de Unity Editor aanpassen



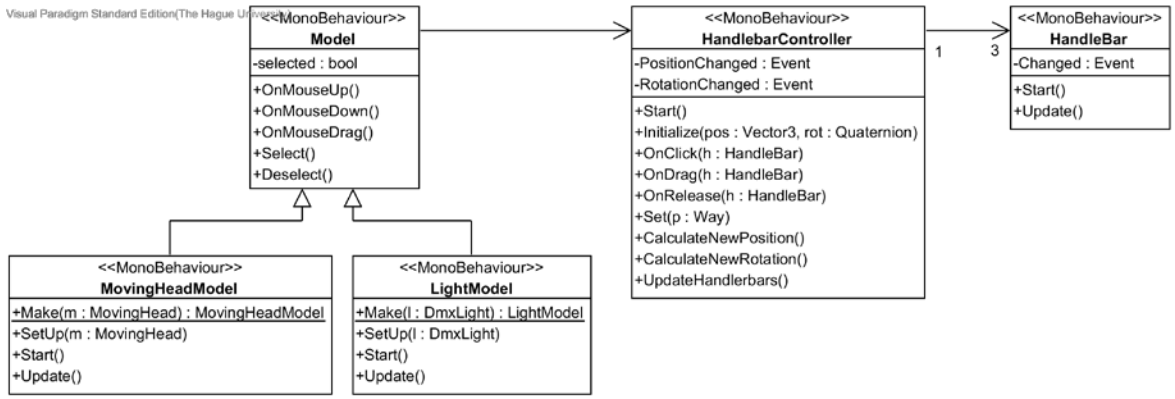
Figuur 28 Rotatie van een object binnen de Unity Editor aanpassen

Uit bovenstaande beschrijving zijn een aantal verschillende eisen van dit (sub) systeem af te leiden namelijk:

1. Het moet mogelijk zijn om de positie van een apparaat aan te passen.
2. Het moet mogelijk zijn om de rotatie van een apparaat aan te passen.
3. Het moet mogelijk zijn om de positie of de rotatie van een apparaat aan te passen door gebruik te maken van een muis of door toets combinaties op een toetsenbord.
4. Het moet voor alle typen apparaten mogelijk zijn om zowel de positie of de rotatie van aan te passen.
5. Op het moment dat er op een apparaat wordt geklikt moet het mogelijk worden om de positie of rotatie (afhankelijk van de status van de applicatie) van een apparaat aan te passen.

11.2 Eerste ontwerp Handlebar klasse

Vanuit de hierboven beschreven eisen en kijkend naar hoe de Unity Editor de functionaliteit heeft geïmplementeerd is er tot een eerste versie van het ontwerp gekomen. Het is voor te stellen dat de drie kegels uit FIGUUR 27 om de positie van een object aan te passen fungeren als handvatten (handlebars) het zelfde geld voor de lijnen in FIGUUR 28 om de rotatie van een object aan te passen maar dit is minder zichtbaar in FIGUUR 28. In het onderstaande ontwerp zijn deze handvatten terug te vinden in de *Handlebar* klassen. De *Handlebars* worden beheerd door de *HandlebarController* deze klassen zorgt er voor dat het mogelijk wordt om de positie of rotatie van een *Model* aan te passen.



Figuur 29 Klassendiagram: analyse handlebar klassen

Om het mogelijk te maken dat een apparaat kan worden geselecteerd is er een *Model* klassen ontworpen. Deze klassen bezit de selecteer functionaliteit. Zoals te zien is in het ontwerp uit FIGUUR 16 zijn er een aantal verschillende apparaten namelijk: lampen en volg lampen (*Light*[1] en *MovingHead* in de DMX package) deze apparaten worden gebruikt door de verschillende *Model* klassen uit de model package. De models representeren de verschillende apparaten in de 3D wereld. Om het mogelijk te maken om verschillende typen apparaten te kunnen selecteren is de *Model* klassen toegevoegd. Alle apparaat models moeten in het vervolg van deze klassen overerven om het mogelijk te maken om de positie en of rotatie van het apparaat te veranderen.

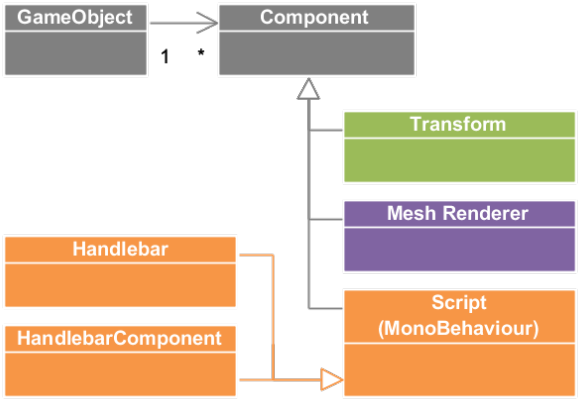
Het eerste idee was rond een 'fat' *HandlebarController* klassen gebouwd. De drie *Handlebar* klassen waren enkel klassen die events doorstuurde naar de *HandlebarController*. Er was een extra iteratie nodig om dit ontwerp te verbeteren. De *Handlebar* klasse had te veel verschillende verantwoordelijk heden deze moesten beter worden opgesplitst zodat de klassen beter te beheren en uit te breiden werd. Hiernaast was er nog een andere zaak die verder moest worden uitgewerkt namelijk: wanneer en wie (welke klassen) de *HandlebarController* creëert.

Notitie1: De naam van de klasse Light is in SPRINT 5 veranderd naar DmxLight om verwarring met de klasse Light uit de Unity package te voorkomen.

11.3 Game objecten en componenten

In SPRINT 2 is al kort de werking van de Unity klasse *MonoBehaviour* uitgelegd. In dit hoofdstuk wordt de verhouding tussen game objecten en *MonoBehaviours* verder uitgelegd aan de hand van onderstaand figuur.

Alle objecten die zich in de gamewereld bevinden zijn game objecten [48]. Deze game objecten hebben van zichzelf geen gedrag. Het zijn enkel containers voor verschillede componenten die het gedrag van een game object aanpassen [49]. Er zijn veel meer componenten binnen Unity beschikbaar dan degenen die in het figuur hiernaast zijn weergegeven. Maar de belangerijksten zijn: *Transform*, *Mesh Renderer* en Script (of *MonoBehaviour*). Het *Transform* component bepaald de positie en rotatie van het game object in de wereld. Het *Mesh Renderer* component bepaald hoe het game object er uit ziet(1).



Figuur 30 Relaties van een Unit game object(versimpeld)

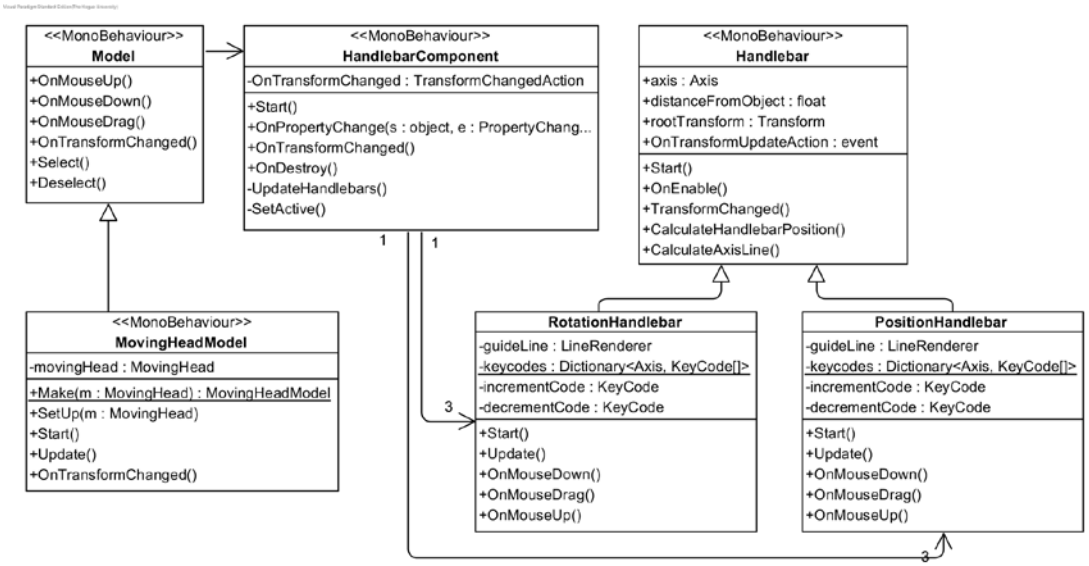
Het script component is een component waarvan het gedrag kan worden geprogrammeerd. In het figuur hierboven zijn twee klassen weergeven die van de klassen *MonoBehaviour* overerven. Deze klassen worden in het volgende hoofdstuk verder uitgelegd. Binnen de applicatie zijn meer klassen die overerven van de klasse *MonoBehaviour* deze klassen dragen het stereotypen '*<MonoBehaviour>*'.

Notitie1: In werkelijkheid ligt dit complexer dan beschreven maar het is genoeg om te weten dat een Mesh Renderer zorgt voor de visuele kant van het game object.

11.4 Tweede ontwerp Handlebar klasse

Kijkend naar de tekortkomingen van de eerste versie van het ontwerp is er een tweede versie van het ontwerp gemaakt die deze tekortkomingen oploste. Het resultaat hiervan is te zien in FIGUUR 31. Hierbij is het grote verschil dat de *Handlebars* nu zelf de kennis bezitten om de rotatie of positie van een object aan te passen, de klasse *HandlebarComponent* voegt enkel de verschillende *Handlebars* samen.

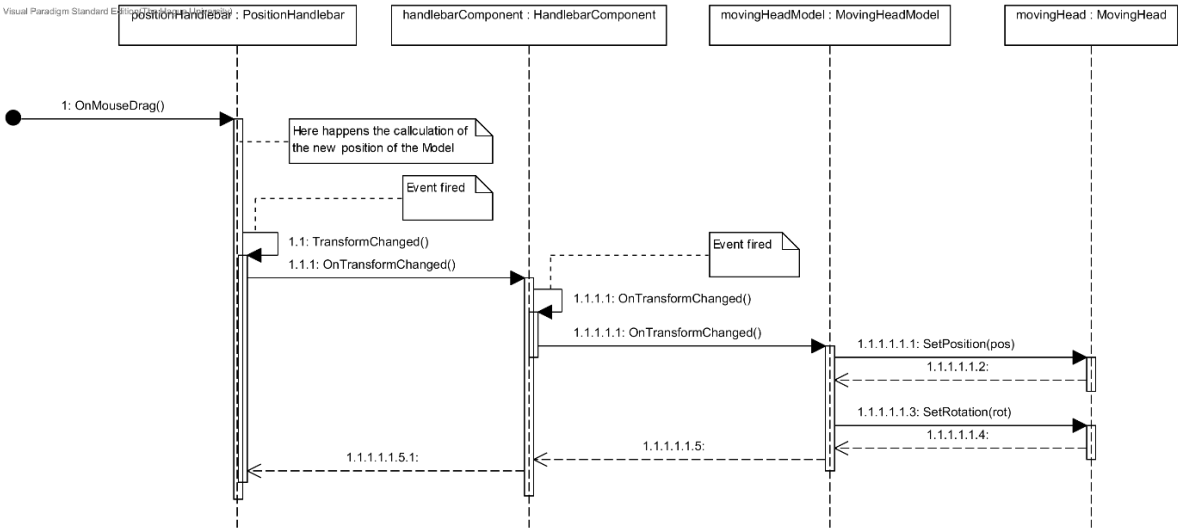
De klasse *Handlebar* definieert de basis functionaliteit van een handvat namelijk: op welke as het handvat zich bevindt, het game object (of Transform) dat moet worden aangepast en een event dat wordt afgevuurd op het moment dat het game object is aangepast. De klasse *HandlebarComponent* is een klasse die de verschillende handvatten aanmaakt en koppelt met de *Model* klassen door middel van het *OnTransformChanged* event. Daarnaast zorgt de klassen *HandlebarComponent* dat de correcte handvatten worden getoond. De applicatie kent namelijk twee verschillende statussen: positie, hierbij moeten de positie handvatten worden getoond en rotatie, hierbij moeten de rotatie handvaten worden getoond. De gebruiker kan tussen deze statussen switchen door middel van toets combinaties.



Figuur 31 Handlebar ontwerp[1]

In de eerste versie van het ontwerp was nog niet duidelijk welke klasse verantwoordelijk zou moeten zijn voor het creëren van de klassen *HandlebarComponent* (voorheen nog *HandlebarController*). Zoals af te leiden valt uit Figuur 31 bestaat de klasse *HandlebarComponent* uit zeven game objecten. Het zou zeer inefficiënt zijn om bij het creëren van een *Model* ook direct een *HandlebarComponent* te creëren dit aangezien het aantal game objecten in de engine dan onnodig groot zou worden. Zo bestond de show, die in SPRINT 4 bezocht was, uit ongeveer 830 verschillende lampen dit zou dan resulteren in 830 * 7 = 5810 extra game objecten. Een betere optie is om het component te creëren op het moment dat het *Model* wordt geselecteerd en weer te verwijderen op het moment dat het Model wordt geselecteerd. Hiermee wordt voorkomen dat er onnodig veel game objecten worden gecreëerd.

Zie onderstaand sequentiediagram voor hoe de positie van zowel de 3D representatie van een volg lamp (*MovingHead*) als de volg lamp zelf wordt aangepast. Het event *OnMouseDrag* wordt door de Unity engine aangeroepen op het moment dat een gebruiker een game object aan het slepen is.



Figuur 32 Sequentiediagram: update de positie van een MovingHead

<u>Notitie1</u>: De klasse LightModel is weggetalen voor de overzichtelijkheid

11.5 Handlebar klasse implementeren

Nadat het ontwerpen van de verschillenden klassen was afgerond kon er begonnen worden met het implementeren van de verschillende klassen. Eén van de vragen die beantwoord moest worden was hoeveel een object precies moest verplaatsen bij een druk op een toets. Een gebruiker is namelijk in staat om een apparaat zowel met de muis als met het toetsenbord van positie of rotatie te veranderen. Afhankelijk van welke toets de gebruiker in drukt wordt er een bepaalde waarde bij de positie of rotatie opgeteld of afgetrokken. De waarde die bij de positie of rotatie wordt opgeteld of afgetrokken mag niet te klein of te groot zijn. Is de waarde te klein dan moet de gebruiker heel vaak op de toets drukken om de bijvoorbeeld de positie met een meter te verschuiven. Is de waarde te groot dan kan de gebruiker de positie niet precies genoeg instellen. Om dit probleem op te lossen maken de *Handlebars* gebruik van een damping methode [50]. Deze methode zorgt er voor dat de waarde die per keer wordt toegevoegd lineair groter wordt naarmate de toets langer wordt ingedrukt. Deze methode zorgt er voor dat de gebruiker de positie of rotatie van een apparaat tot op de millimeter nauwkeurig kan instellen maar ook gemakkelijk snel de positie of rotatie in grotere stappen kan aanpassen door de toets langer in te drukken.

Het implementeren van de functionaliteit dat de gebruiker de positie en rotatie van een apparaat met de muis kon veranderen duurde langer dan verwacht. Dit kwam doordat er van te voren werd gedacht dat dit al eerder door andere ontwikkelaars was geïmplementeerd binnen Unity aangezien dit een veel gebruikte manier is in 3D programma's maar dit bleek niet het geval. Hierdoor koste het implementeren van de algoritmes langer dan van te voren was gepland. Het probleem zat hem in het vertalen van een 2D cursor positie naar een 3D positie in de wereld die gelijk lag aan de as die werd aangepast. Dit zorgde ervoor dat de twee user stories die gepland stonden voor deze sprint niet geheel af waren aan het einde van de sprint.

11.6 Sprint evaluatie

Voor deze sprint stond er gepland om het mogelijk te maken om de positie en rotatie van een lamp te kunnen aanpassen vanuit de applicatie. De tijd die de verschilleden user stories kosten was echter onderschat waardoor niet alle gewenste functies van het aanpassen van de positie en rotatie van een lamp konden worden geïmplementeerd.

Het implementeren van user story 77 kostte meer tijd dan verwacht. Dit kwam doordat het implementeren van een algoritme dat vanuit een gegeven pan en tilt de lamp in de 3D wereld op de juiste rotatie zette langer duurde. Verder werd er tijdens het inplannen van deze sprint vanuit gegaan dat het aanpassen van de positie en rotatie van een object (lamp) zoals hoe dat binnen de Unity Editor werkt al eerder zou zijn gedaan. Dit bleek echter niet het geval, hierdoor moest de functionaliteit vanaf de grond af worden geïmplementeerd. User story 79 en 80 zijn daarom doorgeschoven naar de volgende sprint.

In deze sprint heeft het tussentijds assessment plaats gevonden. Deze is goed verlopen. De feedback op het afstudeerdossier die tijdens het assessment is ontvangen wordt in de verder verloop van het project doorgevoerd.

12 Sprint 8 | UI laag loskoppelen

In de vorige sprint is het niet gelukt om alle geplande user stories te implementeren. Deze user stories zijn dan ook doorgeschoven naar deze sprint. Het gaat hier om user story 79 en 80. Verder wordt er deze sprint onderzocht hoe de koppeling tussen de UI laag en de rest van het systeem kan worden gereduceerd. Hiernaast zullen de verschillende packages van het systeem worden beschreven zodat toekomstige ontwikkelaars beter grip kunnen krijgen op de applicatie.

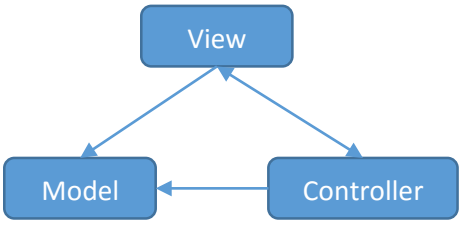
Tabel 14 Sprintplanning sprint 8

Nr.	Story	Uren
79	Als gebruiker wil ik vanuit de applicatie de rotatie van een apparaat aanpassen.	5
80	Als gebruiker wil ik vanuit de applicatie de positie van een apparaat aanpassen.	10
78	Als ontwikkelaar van het systeem wil ik een omschrijving van de verschillende packages van het systeem.	20
83	Als ontwikkelaar wil ik dat het systeem gemakkelijk te initialiseren is.	5
86	Als ontwikkelaar wil ik dat de koppeling tussen de UI package en de rest van het systeem zo laag mogelijk is.	20

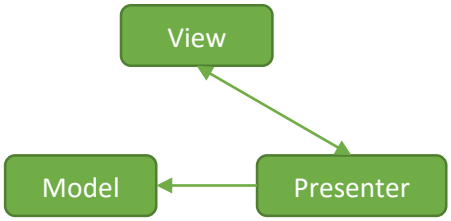
12.1 Loskoppelen van de user interfase

In SPRINT 6 is een deel van de user interface geïmplementeerd. De implementatie van SPRINT 6 had echter een hoge koppeling met de andere packages van het systeem. Dit was niet gewenst aangezien hierdoor veranderingen in de business logica van de applicatie zouden leiden tot veel veranderingen in de UI laag. Ook was de UI laag verantwoordelijk voor het vertalen van de verschillende *Models* (niet te verwarren met de *Model* klasse uit de *Models* package) naar *Models* waarmee de verschillende views gemakkelijk mee konden werken. Dit alles zorgde ervoor dat er moest worden nagedacht over een andere aanpak voor de UI laag. Er is hiervoor gekeken naar verschillende ontwerp patronen en best practices.

Om bovenstaand probleem op te lossen waren er een aantal verschillende oplossingen onderzocht. Het eerste idee was om voor elke package een klassen te ontwerpen die toegang bood tot de verschillende functies van de package een soort van facade dus. De verschillende views in de UI laag zouden dan via deze facade de verschillende functionaliteiten van de package benaderen. Deze oplossing had alleen een probleem het loste maar een deel van de twee problemen op. Door deze oplossing toe te passen werd de koppeling met de verschillende package verlaagd maar er bleven klassen over die een koppeling bleven houden met de UI laag dit waren naast de verschillende facades van de packages de zogenoemde business models van de packages. Dit zijn klassen die de data uit de business logica bezitten dit zijn bijvoorbeeld de klassen: *Tag* of *TrackingDevice* in FIGUUR 22. Dit probleem zou kunnen worden opgelost als de facades de verschillende business models uit de package zou om zetten naar speciale models voor de views. De klassen is dan echter geen facade meer te noemen maar lijkt dan veel meer op een Controller of Presenter uit het Model View Controller of Model View Presenter ontwerppatroon. Er is dan ook gekozen om te gaan voor een van deze bestaande oplossingen om de UI los te koppelen van de business logica.

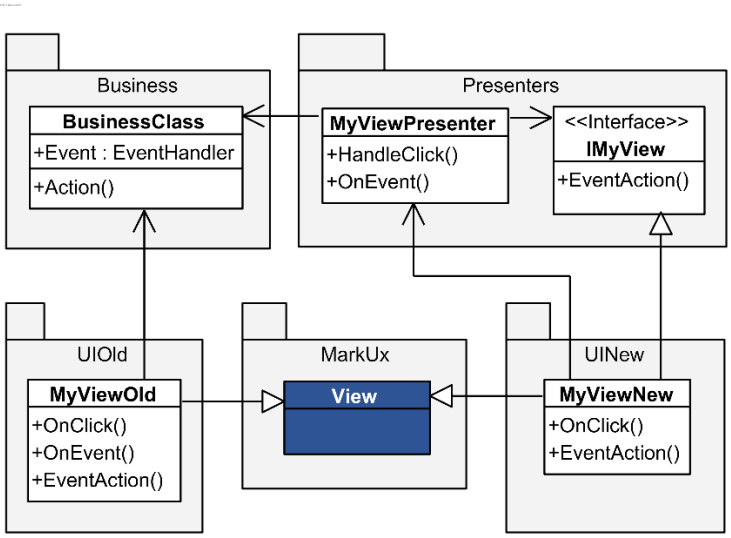


Figuur 33 Model View Controller



Figuur 34 Model View Presenter

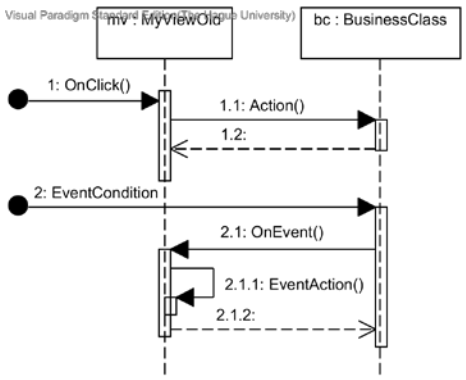
Het verschil tussen MVC en MVP zit hem in hem in het feit dat bij MVC de view en de controller met de model communiceren zie FIGUUR 33. Bij MVP communiceert de view via de presenter met de model zie FIGUUR 34. Het voordeel van MVP ten opzichte van MVC is dat de views nog minder kennis nodig hebben van het onderliggende systeem. Hierdoor worden de views enkel doorgeef luiken naar de presenters toe. Dit is in deze situatie gewenst aangezien de code dan beter testbaar wordt. Dit komt doordat de views zelf alleen maar logica bezitten voor het afhandelen van de verschillende gebruiker acties zoals bijvoorbeeld een druk op de knop. De view vertaald deze druk op een knop naar een methode aanroep op de presenter en de presenter doet het 'echte' werk. Aangezien het 'echte' werk zich in de presenter bevindt kan deze los worden getest zonder tussen komst van een gebruiker. Zie onderstaand figuur voor een versimpeld overzicht hoe de UI package in SPRINT 4 was geïmplementeerd (*UIOld* package) en hoe deze in deze sprint is ontworpen en geïmplementeerd (*UINew* package). Voor een verdere omschrijvingen kan hoofdstuk 4.11 worden geraadpleegd uit het ontwerprapport.



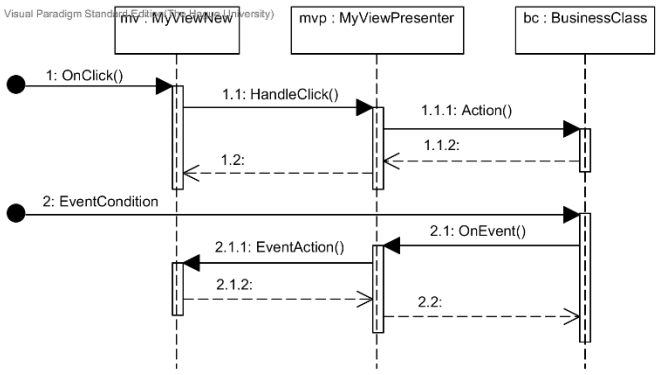
Figuur 35 Klassendiagram: versimpelde weergaven oude en nieuwe ontwerp UI

De presenter communiceert over het algemeen met de view via een interface zoals te zien is in FIGUUR 35. Deze interface definieert acties die er op de view kunnen worden gedaan bijvoorbeeld het verversen van een lijst of het toevoegen van een item aan een lijst. De presenter is ook verantwoordelijk voor het omzetten van de verschillende business models naar speciale view models. Dit is niet weergegeven in het figuur hierboven. Deze view models presenteren de data op zo'n manier dat de view er gemakkelijk mee overweg kan. Hiernaast zorgen deze models er voor dat de views geen weet hebben van de verschillende business models die er zijn.

Zie de sequentiediagrammen uit FIGUUR 36 en FIGUUR 37 voor een verdere uitleg van het hierboven weergegeven klassendiagram. In de sequentiediagrammen wordt de werking van de *OnClick()* methode weergegeven. Hiernaast worden ook de aanroepen weergegeven die optreden als het event *Event* van de BusinessClass wordt afgevuurd.



Figuur 36 Sequentiediagram: UI communicatie voorheen



Figuur 37 Sequentiediagram: UI communicatie nu

Al met al zorgt deze nieuwe implementatie van de UI package er voor dat de koppeling van de UI laag met de verschillende business klassen uit het systeem is verwijderd. Dit zorgt er voor dat toekomstige veranderingen in de business klassen minder snel zullen te leiden tot veranderingen in de UI package (dit geldt ook andersom).

12.2 Initialiseren van het systeem

Gaandeweg het project zijn er tal van klassen aan het systeem toegevoegd. Sommige van deze klassen moeten worden geïnitialiseerd voordat ze gebruikt kunnen worden. Zo hangen ze bijvoorbeeld af van andere klassen of moeten ze instellingen uit bepaalde bestanden lezen voordat ze gebruikt kunnen worden. Nu het systeem groter is geworden en om te voorkomen dat het initialiseren van de verschillende klassen versnippert over meerdere klassen is het nodig om deze sprint het initialiseren van de verschillende klassen beter te structureren. Daarom is er deze sprint een klasse ontworpen die verantwoordelijk is voor het initialiseren van de verschillende klassen van het systeem. Deze klasse creëert en verbindt de verschillende klassen aan elkaar zodat deze kunnen worden gebruikt door het systeem. Deze klassen is terug te vinden als de *PositionApplication* klasse in het ontwerprapport.

Een aantal klassen in het systeem maakte achter de schermen gebruik van singletons of creëerde instanties van andere klassen in hun constructor. Hierdoor was het voor deze klassen niet direct duidelijk hoe deze moesten worden geïnitialiseerd. Zo was het voor sommige klassen namelijk nodig om eerste een singleton klasse te initialiseren voordat de klasse zelf kon worden aangemaakt. Om deze onduidelijke relaties met andere klassen te vermijden is er gebruik gemaakt van het Dependency Injection (DI) [51] patroon. Dus in plaats van indirecte relaties met klassen is er voor gekozen om de relaties mee te geven in bijvoorbeeld de constructor van de klasse. Hierdoor is bij het aanmaken van de klassen direct duidelijk welke andere klassen de klasse nodig heeft om te functioneren.

12.3 Singletons vermijden

Voor een aantal klassen in het systeem geldt dat het niet zinvol is om meerdere instanties van de klassen te hebben. Zo is het bijvoorbeeld niet nodig om meerdere TagManagers of RecordManagers te hebben. Er zou kunnen worden gezegd dat dit dus prima kandidaten zijn voor Singletons maar dat is dit geval niet waar. Singletons zijn niet meer dan verkapte globale variabelen hierdoor zijn de singleton klasse vanuit iedere andere klasse binnen de applicatie te gebruiken. Dit heeft op sommige momenten zijn nut maar voor veel van de klassen binnen het systeem is het beter om de klassen via het DI patroon aan de benodigde klassen mee te geven dit is precies wat de initialisatie klasse die in het vorige hoofdstuk werd beschreven is doet.

Sommige klassen zijn echter wel goede kandidaten voor Singletons. Dit zijn klassen die door veel andere klassen worden gebruikt. Binnen de applicatie wordt er geen gebruik gemaakt van het Singleton patroon om de klassen bereikbaar te maken in de globale scope. Hiervoor wordt namelijk een static *App* klasse gebruikt. Aangezien het systeem vier potentiele singletons kent namelijk

UiManager, *TickProvider*, *SettingsManager* en *ResourceLoader* is er voor gekozen om deze samen te bundelen in een *App* klasse. Deze klasse is volkomen static en is dus te bereiken door de andere klassen in het systeem. Op deze manier wordt bijna het zelfde effect behaald als een Singleton maar dan met het voordeel dat de desbetreffende klassen niet elk het Singleton patroon hoeven te implementeren. Tevens zijn niet singleton klassen over het algemeen makkelijker te Unit testen.

12.4 Sprint evaluatie

In deze sprint is de koppeling van de UI package met de rest van het systeem verminderd. Er is een Presenter package toegevoegd. Deze package zorgt voor een bridge tussen de UI en de business logica. Hierbij is gebruik gemaakt van het Model View Presenter ontwikkelpatroon. Tevens is er een klasse ontworpen die het initialiseren van de verschillende packages van het systeem op zich neemt.

Hiernaast zijn deze sprint de verschillende packages van het systeem nader toegelicht. Deze beschrijvingen dienen als input voor het ontwerprapport dat gepland staat voor de volgende sprint.

13 Sprint 9 | Afronding

SPRINT 9 is met zijn drie weken de langste sprint van dit project, tevens is dit de laatste sprint van het project. Er is gekozen om deze sprint een week langer te laten duren dan de rest van de sprints aangezien er deze sprint een aantal feestdagen in het midden van de week vallen. Het aantal indeelbare dagen blijft ongeveer hetzelfde als dat van de vorige sprint. Er kan dus niet per se meer werk in deze sprint worden gestopt. Deze sprint zal zich focussen op het ontwerpdocument en het afmaken van het afstudeerverslag met de bijhorende bijlagen. Hiernaast zal ook de geschreven applicatie worden overgedragen.

Tabel 15 Sprint planning sprint 9

Nr.	Story	Uren
87	Als ontwikkelaar van het systeem wil ik een ontwerpdocument waarin de structuur van de applicatie wordt beschreven.	10

13.1 Ontwerprapport

In SPRINT 8 is er al een begin gemaakt aan de beschrijvingen van de verschillende packages van het systeem. Aan de hand van verschillende soorten UML diagrammen worde de werking van de applicatie nader toegelicht hierbij wordt vooral gebruik gemaakt van de package, klassen, sequentie en state diagrammen. Per package wordt de werking van de belangrijksten klassen toegelicht dit gebeurt aan de hand van sequentiediagrammen of statediagrammen. De diagrammen die in het ontwerprapport zijn gebruikt zijn gaande weg het project tot stand gekomen. Deze sprint zal zich focussen op het verder uitbreiden van dit document met daarin een beschrijving van de ontwikkelomgeving en beschrijvingen van de projectstructuur.

Het doel van het ontwerprapport is om het toekomstige ontwikkelaars makkelijker te maken om het systeem uit te breiden. Het ontwerprapport dient dan als beschrijving van het systeem. Hiernaast zijn er nog tal van andere zaken die het systeem makkelijker onderhoudbaar en overdraagbaar maken. Deze zullen in het volgende hoofdstuk verder worden toegelicht.

13.2 Overdragen

Het is van belang dat de geschreven software gemakkelijk te over dragen is. Hierdoor zijn toekomstige ontwikkelaars sneller instaat om met het systeem aan de slag te gaan. Dit gebeurt aan de ene kant door verschillende soorten documentatie, hier is het ontwerprapport een voorbeeld van. Maar dit kan bijvoorbeeld ook door commentaar aan de code toe te voegen. In dit hoofdstuk worden de verschillende aspecten behandeld die er voor zorgen dat dit project goed overdraag baar is.

13.2.1 Projectstructuur

Naast de uitleg over de ontwikkelomgeving en de structuur van het project zijn er in de git repository verscheidende *README* bestanden te vinden. Deze bestanden beschrijven de verschillende onderdelen van het project zoals bijvoorbeeld waar de effectscripts of de fixture bestanden moeten worden geplaatst. Tevens is de git repository logisch gestructureerd in mappen waardoor de verschillende onderdelen van de applicatie gemakkelijk te vinden zijn.

13.2.2 Documentatie

Naast het ontwerprapport en de beschrijvingen van de project structuur is ook de code voorzien van documentatie. Deze documentatie komt in twee vormen, de eerste vorm is door gebruik te maken van een door Microsoft ontwikkelde documentatie structuur [52]. Door het toevoegen van bepaalde XML

attributen aan de code kan de werking van klassen en methodes worden toegelicht zie FIGUUR 38 voor een voorbeeld. Deze methode wordt ook ondersteund door Doxygen [53] hierdoor is het gemakkelijk om in de toekomst automatisch documentatie te generen. De belangrijksten klassen en methodes zijn voorzien van dit soort commentaar. De tweede manier is door het toevoegen van commentaar in de code zelf. Door het gebruik van commentaar worden complexe stukken code nader toegelicht. Zo worden bijvoorbeeld *if else* constructies waar van niet direct duidelijk is wat er wordt gecontroleerd voorzien van een uitleggend stukje commentaar. Dit zorgt er voor dat toekomstige ontwikkelaars de werking van de code sneller kunnen doorgronden.

Figuur 38 XML commentaar van een C# methode

```
/// <summary>
/// Get the name of the effect
/// </summary>
/// <remarks>
/// The name of the effect should be unique inside the application otherwise it
/// will not be loaded.
/// </remarks>
/// <returns>The name of the effect</returns>
abstract public string GetName();
```

13.2.3 Ontwerp

In de loop van het project zijn er tal van diagrammen gemaakt. Deze diagrammen staan beschreven in het ontwerpdocument. Door gebruik te maken van deze diagrammen kan de werking van de applicatie sneller achterhaald worden.

13.2.4 Unit tests

Er zijn tal van unit test geschreven die verschillende stukken van het systeem testen. Deze unit test zijn gaandeweg het project tot stand gekomen en worden gebruikt om de werking van de software te valideren. Een groot voordeel van unit test is dat ze geautomatiseerd kunnen worden uitgevoerd. Hierdoor is het vrij snel te bepalen of het systeem naar behoren werkt. Bij overdragen en onderhouden van software kunnen unit tests een belangrijke rol innemen. Zo kan de werking van een klasse of methode verduidelijkt worden door te kijken naar desbetreffende unit test. Ook is het met unit tests makkelijker om te controleren of de herschreven methode correct werkt en of een nieuwe functionaliteit van het systeem geen bestaande functionaliteiten kapot maakt.

13.2.5 Hardware

In SPRINT 1 is er een hardware testrapport opgezet met daarin verschillende test cases waar aan de hardware moet voldoen. Dit rapport is gaandeweg het project verder uitgebreid met de werking van de hardware en de resultaten van de testdag. Dit rapport kan gebruikt worden als basis om de hardware verder te testen. Tevens kan dit rapport dienen als naslag werk voor de werking van het protocol dat gebruikt wordt om te communiceren met het systeem van Sendrato.

13.3 Sprint evaluatie

In deze sprint is het ontwerprapport gemaakt. Dit rapport beschrijft de applicatie aan de hand van de verschillende UML beschrijvingen die gaande weg het project zijn gecreëerd. Tevens is de applicatie met de daarbij horende bron code en documentatie overgedragen aan de opdrachtgever.

14 Conclusie en aanbevelingen

In dit hoofdstuk wordt de conclusie van het project beschreven, tevens worden er een aantal aanbevelingen gedaan. Deze aanbevelingen komen voor een groot deel uit het scrumrapport.

14.1 Conclusie

Bij aanvang van het project bleek dat de opdracht die beschreven was in het afstudeerplan niet meer relevant was. Er moest dus een nieuwe opdracht worden gedefinieerd deze is samen met de opdrachtgever gedefinieerd. Deze nieuwe opdracht omschrijving lag in het verlengde van de oude opdracht. De doelstelling van het project luide als volgt:

> *"De doelstelling van dit project is om een software applicatie te ontwikkelen die gebruikmakend van de hardware van Sendrato lampen en of visuals laat reageren op de positie veranderingen van een persoon of personen. Verder zijn aan het einde van het project tests uitgevoerd om de werking van de hardware te valideren."*

Kijkend naar de doelstelling en de resultaten van het project kan worden gezegd dat de doelstelling voor een deel is gehaald. Al vroeg in het project bleek dat de focus van het project zou worden gelegd bij het aansturen van lampen in plaats van videobeelden. Er is tijdens dit project een uitbreidbare applicatie ontwikkeld die in staat is om lampen aan de hand van positie veranderingen aan te sturen. Hierbij zijn tal van keuzes gemaakt die het gebruik van de applicatie makkelijker en in zijn geheel makkelijker uitbreidbaar maken. De applicatie is echter nog niet in zijn eind versie. Vooral aan de kant van de user interface moet er nog werk worden verricht. Voor de hardware van Sendrato zijn verschillende testcases beschreven en deze zijn tijdens de testdag gevalideerd.

14.2 Aanbevelingen

Elke sprint zijn er een aantal user stories geïmplementeerd tevens zijn er in sommige sprints nieuwe user stories bijgekomen die soms weer in vervolg sprints zijn geïmplementeerd. Het scrum rapport uit BIJLAGEN F geeft hier een duidelijk overzicht van weer. Niet alle user stories uit de product backlog zijn tijdens dit project aan bod gekomen. Deze user stories kunnen dus ook in een vervolg project worden meegenomen. Hier volgen een aantal beschrijvingen van onderdelen van het systeem waarvan nog een of meerdere user stories open staan. Niet alle open user stories zijn met onderstaande beschrijvingen beschreven. Voor een totaal overzicht kan het scrum rapport worden geraadpleegd.

14.2.1 User Interface

Niet voor alle functionaliteit van de applicatie is er een user interface geschreven. Zo is het nog niet mogelijk op bijvoorbeeld opnamen of effecten vanuit de UI te beheren. De applicatie is echter wel zo ontworpen dat het toevoegen van nieuwe user interface elementen in de toekomst makkelijker gaat. Dit is mede te danken aan de duidelijke scheiding van data en view in de applicatie. Verder is er tijdens dit project gefocust op een functionele user interface in plaats van een grafisch aantrekkelijke interface. Er kan dus nog worden na gedacht over een grafisch aantrekkelijk user interface.

14.2.2 Effecten

Het is mogelijk om het gedrag van lampen aan de hand van zogenoemde effect scripts aan te passen. Op dit moment laad de applicatie deze effect scripts bij het opstarten van de applicatie in. Om dit onderdeel van de applicatie nog bruikbaarder te maken kan er voor worden gekozen om deze effect scripts bij veranderingen opnieuw te compileren en toe te voegen aan de applicatie. Om deze functionaliteit te implementeren is er echter nog meer onderzoek nodig.

14.2.3 Configuratie

Zoals beschreven in SPRINT 3 is er tijdens dit project niet gefocust op het configuratie onderdeel van de applicatie aangezien de prioriteit van het gemakkelijk toevoegen van effecten hoger was. Dit onderdeel is echter wel nodig in de eind versie van de applicatie. Hier moet dus nog meer onderzoek naar worden gedaan.

14.2.4 Hardware

Tijdens dit project is er gebruik gemaakt van hardware geleverd door een 3e partij namelijk Sendrato. Het was van belang om de werking van de hardware te valideren aan de hand van de eisen die de opdrachtgever stelde. Er zijn verschillende eisen van de hardware opgesteld. Om deze eisen te kunnen valideren zijn er verschillende testcases gecreëerd. De verschillende testcases zijn tijdens de testdag gevalideerd. De testcases en de resultaten van de testdag zijn te vinden in het hardware testrapport dat te vinden is in BIJLAGEN D. Er is echter nog niet voldaan aan alle testcases zo zijn de sensoren van Sendrato bijvoorbeeld nog te verstoren door een body block. Het is nodig Sendrato nog meer tijd te geven om de hardware correct te implementeren tevens is het noodzakelijk om nog verschillende testdagen te organiseren zodat de werking van de hardware verder kan worden gevalideerd.

15 Project evaluatie

In dit hoofdstuk wordt het project geëvalueerd. Hierbij is het hoofdstuk opgedeeld in drie onderdelen namelijk de product evaluatie, de proces evaluatie en een evaluatie naar de aan te tonen competenties.

15.1 Productevaluatie

Aan het begin van het project zijn er een aantal producten beschreven die tijdens de loop van dit project zouden worden opgeleverd tijdens of aan het einde van het project. Het grootste product hierin is het softwaresysteem. Dit systeem in de loop van het project tot stand gekomen.

Of het software systeem af is een daarmee dus het doel van het project gehaald is, is lastig te stellen. Dit komt doordat het vanaf het begin al snel duidelijk werd dat niet alle aspecten van het doel konden worden geïmplementeerd in de tijd die stond voor dit afstudeerproject. Samen met de opdrachtgever zijn de verschillende onderdelen van het systeem verdeeld in belangrijke onderdelen en minder belangrijke onderdelen zoals te lezen valt in het requirements document uit BIJLAGEN C. Er is vervolgens begonnen met het onderdeel met de hoogste prioriteit. In SPRINT 3 kwam naar voren dat de opdrachtgever graag de mogelijkheid zou willen hebben om effecten in de vorm van scripts aan de applicatie te kunnen toevoegen. Hierdoor zou er minder tijd over blijven voor de andere onderdelen, er is toen besloten om het configuratie onderdeel van de applicatie een lagere prioriteit te geven. Mocht er later nog tijd overblijven dan kon dit onderdeel nog worden opgepakt. Hier is uiteindelijk niet meer van gekomen doordat er onderdelen aan het systeem moesten worden toegevoegd die een hogere prioriteit hadden. Er kan dus nog niet worden gesproken van een af software systeem. Maar er is een uitbreidbare basis gelegd waarop in de toekomst gemakkelijk kan worden door gebouwd.

De applicatie wordt beschreven aan de hand van verschillende UML diagrammen die te vinden zijn in het ontwerprapport uit BIJLAGEN H. De software van de applicatie is uitbreidbaar en onderhoudbaar opgezet. Zo is er rekening gehouden dat er in de toekomst gemakkelijk nieuwe DMX apparaten aan de applicatie kunnen worden toegevoegd en is het bijvoorbeeld gemakkelijk om de manier van opslaan en uitlezen van opnamen later te veranderen. Ook is het door de keuze van het MarkUX framework gemakkelijk om later de user interface van de applicatie aan te passen zonder dat hiervoor programma code hoeft te worden aangepast. Tevens zijn er voor een groot deel van de applicatie unit tests geschreven. Deze unit test valideren de werkingen van de verschillende klassen en methodes in de software. Er is bewust gekozen om niet voor alle klassen en methodes unit tests te schrijven aangezien niet elke klasse of methode even complex is, waardoor het schrijven van een unit test niet veel toevoegt. Hiernaast bieden de unit tests naast de geschreven documentatie, toekomstige ontwikkelaars een duidelijk beeld van de werking van een klassen of een methode.

Naast de hierboven beschreven producten zijn er nog een aantal andere producten tijdens dit project opgeleverd. Zo zijn de verschillende user stories van de applicatie te vinden in het scrumrapport. Dit rapport beschrijft alle voltooide en onvoltooide user stories van het project. Dit document kan als input gebruikt worden in een vervolg project. Verder is er een handleiding geschreven die uitlegt hoe een effectscript aan de applicatie kan worden toegevoegd.

Een ander product was het hardware testrapport dit rapport beschreef de verschillende test cases die gebruikt werden om de functionaliteit van de hardware te valideren. Aan de hand van de hardware eisen uit het requirements document zijn de testcases opgesteld. In eerste instantie was het idee dat de hardware vroeg in het project zou worden getest maar dit bleek uiteindelijk niet mogelijk doordat Sendrato nog niet ver genoeg gevorderd was met de hardware. De testdag heeft uit eindelijk aan het einde van SPRINT 4 plaats gevonden. De resultaten van deze testdag bevinden zich ook in het rapport.

De hardware voldeed tijdens de testdag niet aan alle opgestelde test cases. Het was dus nodig om de hardware een volgende testdag nogmaals te testen. Het was niet mogelijk om deze testdag nog voor de deadline van het verslag in te plannen. Aangezien het niet constant mogelijk was om fysiek met de hardware te testen is er een simulatietool ontwikkeld. Deze tool implementeert het protocol dat Sendrato gebruikt om data door te sturen. Door deze tool was het mogelijk om zonder de hardware toch de applicatie te testen.

Aan het begin van het project was het idee er om ook een los onderzoeksrapport te creëren zoals te lezen valt in HOOFDSTUK 3.6. Het idee was dat hierin de verschillende onderzochte technieken die gebruikt worden in de applicatie zouden worden beschreven. Tijdens de loop van het project bleek echter dat er geen behoefte was aan dit document. Dit omdat de werking van de verschillende technieken niet voldoende complex was om dit op te schrijven. Er is via andere bronnen al veel informatie over de technieken te vinden.

15.2 Procesevaluatie

Tijdens dit project is er gewerkt volgens de Scrum methodiek. Aangezien Scrum er van uit gaat dat het in een team verband wordt uitgevoerd moesten een aantal aspecten van de methodiek anders worden ingedeeld. Zo zijn de verschillende rollen die Scrum beschrijft samengenomen en is de invulling van een aantal van de voorgeschreven meetings veranderd. Tijdens dit project zijn er in totaal negen sprint geweest. Elke sprint had ongeveer een lengte van twee weken. Tijdens de sprint planningen werden er user stories uit de backlog gehaald om in de sprint geïmplementeerd te worden. Af en toe kostte een user story meer tijd dan van te voren was ingepland, dit kwam bijvoorbeeld doordat de complexiteit was onderschat of omdat er tegenslagen waren tijdens implementeren van de user story. De betreffende user story werd dan doorgeschoven naar de volgende sprint. Soms kwam het ook voor dat een user story werd overschat, hierdoor bleef er tijd in de sprint over, deze tijd werd dan opgevuld met nieuwe user stories.

Doordat Scrum goed om kan gaan met veranderende requirements was het gemakkelijk om gaande weg het project waar nodig bij te sturen. Ook was het gemakkelijk om inzichten die in voorgaande sprints opgedaan waren te verwerken in de nog uit te voeren sprints. Zoals bijvoorbeeld het gebruik van delegates in SPRINT 2. Dit alles zorgde ervoor dat het proces heel natuurlijk verliep waarbij input van de opdrachtgever werd gebruikt om de sprints in te delen. Er kan worden gezegd dat Scrum de juiste methodiek was voor dit project.

Tijdens dit project heeft er één testdag samen met Sendrato plaats gevonden. De afstudeerder had graag meerdere van deze dagen gehad om de software nog verder te verbeteren. Het was echter lastig voor de opdrachtgever om in de tijd die beschikbaar was een geschikte datum en locatie voor een vervolg testdag te vinden. Het ontbreken van meerdere testdagen heeft echter niet geleid tot een mindere kwaliteit van de software.

Voordat een user story als af kon worden bestempeld moeten er afhankelijk van de user story verschillende soorten tests worden uitgevoerd. Deze tests zijn uitgevoerd maar zijn echter niet altijd beschreven waardoor het lastig is voor een andere ontwikkelaar om deze tests nogmaals uit te voeren. Achteraf hadden deze tests en dan met name de systeemtests beter gedocumenteerd kunnen worden zodat ze makkelijker reproduceerbaar zouden zijn. Dit probleem is deels opgelost door het scrum rapport waarin de verschillende user stories staan beschreven en waaruit de systeem tests dus af te leiden zijn. Ook de aanwezigheid van verschillende unit tests helpt dit probleem deels op te lossen.

15.3 Competenties

Bij aanvang van het project zijn er een aantal competenties gekozen die in het bijzonder worden aangetoond binnen dit project. Deze competenties zijn voor aanvang van het project beschreven in het afstudeerplan. Bij aanvang van het project bleek echter dat de opdracht moest worden herzien. Voor de nieuwe opdracht waren dezelfde competenties van toepassing als die van de oude.

15.3.1 Ontwerpen van een technisch informatie systeem (C8)

Tijdens dit project zijn er verschillende ontwerpen van het software systeem gecreëerd. De te schrijven onderdelen van het systeem werden telkens eerst ontworpen voordat er werd begonnen met het implementeren er van. Het ontwerpen gebeurde over het algemeen aan de hand van verscheidende UML diagrammen. Tijdens het implementeren van de software ontstonden soms nieuwe inzichten, deze werden dan verwerkt in het ontwerp. Op deze manier bleven de ontwerpen consistent met de code die ze beschreven. De verschillende ontwerpen van het software systeem zijn iteratief tot stand gekomen. Dit is bijvoorbeeld goed te zien bij het effecten gedeelte. In SPRINT 2 wordt dit onderdeel voor het eerst ontworpen waarna het in SPRINT 3 verder uitgewerkt wordt. Ook is dit iteratieve proces goed terug te zien bij het ontwerpen en implementeren van de user interface.

Er is gebruik gemaakt van een aantal verschillende soorten UML diagrammen om de software te beschrijven. Om de diagrammen beter leesbaar te maken is er soms afgeweken van de standaard UML notatie. Om deze diagrammen beter overdraagbaar te maken zijn deze samengenomen in het ontwerprapport. Hierin wordt de samenhang en de werking van de verschillende diagrammen nader beschreven. Dit leidt er toe dat het software systeem beter overdraagbaar is.

15.3.2 Kiezen van een ontwikkelstrategie en ontwikkelmethodiek (A4)

Bij aanvang van het project is er gekozen met welke ontwikkelmethodiek er gewerkt zou worden. Om een goede keuze te kunnen maken zijn eerste de verschillende kenmerken van het project beschreven. Een van de belangrijkste kenmerken was dat het precieze einddoel van het project nog niet gedefinieerd was. Aan de hand van de verschillende kenmerken van het project zijn vervolgens eisen opgesteld waaraan de te kiezen ontwikkelmethodiek aan moest voldoen. Vervolgens is er onderzoek gedaan naar verschillende ontwikkelmethodieken.

De verschillende gevonden ontwikkelmethodieken zijn vervolgens in een tabel met elkaar vergeleken. Op deze manier kon goed worden weergegeven welke methodieken wel en welke methodieken niet bij het project pasten. Hierna zijn de verschillende methodieken met elkaar vergeleken er is uit eindelijk gekozen voor de ontwikkelmethodiek Scrum. Aangezien de Scrum ontwikkelmethodiek over het algemeen uit gaat van een team verband moest er een vertaal slag worden gemaakt aangezien dit project door één persoon wordt uitgevoerd. Hierbij zijn een aantal verschillende rollen van Scrum samengevoegd en zijn een aantal vaste vergader momenten anders ingedeeld.

15.3.3 Professioneel werken: zelfstandig werken (H5)

Dit project is zelfstandig uitgevoerd. Bij aanvang van het project bleek dat de beschreven opdracht uit het afstudeerplan niet meer relevant was. Er zijn toen gesprekken gevoerd met de opdrachtgever om een nieuwe opdracht te definiëren. Deze opdracht is vervolgens door gecommuniceerd met de begeleidend docent. Het verloop van het project was niet van te voren gedefinieerd, de afstudeerder was dus vrij om het proces zoals gewenst in te richten. Er is gekozen om tijdens dit project te werken volgens de Scrum ontwikkelmethodiek. Bij deze methodiek is een korte lijn met de opdrachtgever noodzakelijk. Het initiatief van de verschillende meetings die binnen Scrum gepland zijn lag bij de afstudeerder. Ook beschreef de afstudeerder tijdens bijvoorbeeld de sprint planningen de verschillende mogelijkheden voor de vervolg sprint. In samenspraak met de opdrachtgever werd deze

vervolg sprint vervolgens ingericht. Tevens nam de afstudeerder ook het initiatief voor de verschillende ontmoetingsmomenten met de begeleidende docenten.

Naast het beheren van het project proces dacht de afstudeerder ook mee met de opdrachtgever. Dit is bijvoorbeeld terug te zien in de keuze voor het gebruik van Unity in plaats van openFrameworks of in de keuze voor scripting. Hierbij heeft de afstudeerder de opdrachtgever verschillende alternatieve voorgelegd. In samenspraak met de opdrachtgever is er vervolgens tot een keuze gekomen.

15.3.4 Analyseren van het probleemdomein (A1)

Bij aanvang van het project was er nog weinig kennis over het probleemdomein van de opdracht bekend bij de afstudeerder. Een van de eerste werkzaamheden die dan ook gepland stond was het verhelderen van het probleemdomein. Dit is gedaan aan de hand van een aantal verschillende zaken. Zo zijn er verschillende gesprekken met de opdrachtgever gevoerd om het domein te verhelderen. Deze gesprekken boden een tal van handvatten waarmee de afstudeerder zelf aan de slag kon. Zo kwamen er tijdens de gesprekken een aantal technieken naar voren die nader onderzocht moesten worden. Dit waren bijvoorbeeld de protocollen DMX en ArtNet. Naast de technieken moest ook het probleem dat moest worden opgelost worden geanalyseerd.

De opdrachtgever had bij aanvang van het project een bepaald beeld voor zich van het probleem dat moest worden opgelost. Aan de hand van de hierboven beschreven gesprekken is er door de afstudeerder kennis opgedaan over dit probleem. De afstudeerder heeft dit probleem vervolgens vertaald naar een software probleem. Dit is mede gedaan door gebruik te maken van een use case diagram met bijhorende use case scenario's. Dit diagram diende als start punt voor het verdere ontwerp van het systeem. Tevens konden uit dit diagram de eerste eisen van het systeem worden bepaald.

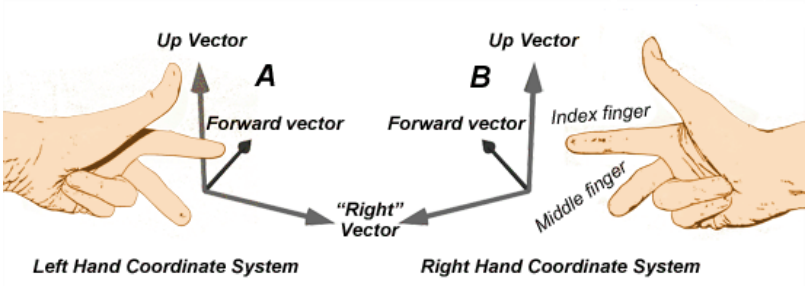
15.3.5 Creëren en innoveren (H8)

De laatste competentie die voor dit project gekozen was is de competentie: creëren en innoveren. De setting van de opdracht leent zich erg voor deze competentie aangezien de opdracht veel vrijheid bood er was namelijk nog weinig beschreven en uitgezocht. Er was dus veel ruimte voor eigen invulling van het project. Dit is ook terug te zien in de verschillende keuzes die tijdens het project zijn gemaakt.

Een van de belangrijkste keuzes binnen dit project was wel de keuze om een game engine te gebruiken voor het aansturen van de lampen. Er is onderzoek gedaan naar een aantal verschillende game engines. Deze zijn onderling met elkaar afgewogen en uiteindelijk bleek Unity de beste kandidaat voor dit project te zijn. Het gebruik van een game engine opent nieuwe deuren voor het aansturen van lampen. Zo zouden lampen in de toekomst kunnen worden aangestuurd met physics effecten zoals beschreven staat in HOOFDSTUK 6.1.4. Tevens is het gebruik van een game engine voor zover bekend nog niet eerder gebruikt om lampen mee aan te sturen. Bij het maken van een keuze zijn vaak ook alternatieve overwogen dit is bijvoorbeeld terug te zien bij de keuze voor scripting of de keuze voor een user interface framework.

Verklarende woordenlijst

Nr.	Woord	Omschrijving
W1	Raycasting	Raycasting is een techniek die binnen game engine wordt gebruikt om een 2D positie te vertalen naar een 3D positie. Zo kan raycasting worden gebruikt om te bepalen naar welk 3D object de muis wijst.
W2	MIDI controller	Een MIDI controller is een stuk hardware waarmee parameters van een programma of ander stuk hardware kunnen worden aangestuurd. Een MIDI controller wordt veel gebruikt door DJ's en VJ's (personen die de visuals van een show aansturen) om hun software programma's mee aan te sturen.
W3	Assemblies	Dit zijn de bouw stenen van het .NET framework. In een assembly bevinden zich de verschillende typen en resources van een applicatie. In de praktijk nemen assemblies de vorm in van een uitvoerbaar bestand (.exe) of een dynamic link library (.dll) onder Windows.
W4	Indie developer	Hieronder worden game studio's verstaan die een game ontwikkelen met een relatief klein budget en ontwikkelteam (1 – 25 man). Het budget van een indie game ligt stukken lager dan dat van triple A games.
W5	Unity Editor	De Unity Editor wordt meegeleverd met de Unity engine. Met behulp van dit programma kunnen er applicaties voor de Unity engine worden ontwikkeld.
W6	Links handig coördinaten systeem	Het links handige coordinaten systeem is het best uit te leggen aan de hand van onderstaand figuur. Het systeem defineert naar welke kanten de forward, up en right vector wijzen.



Figuur 39 Coördinatensystemen [54]

Bibliografie

[1] „Myo Gesture Control Armband," Thalmic Labs, [Online]. Available: https://www.myo.com/. [Geopend 14 September 2015].

[2] „MANAGING CONNECTED CROWDS," [Online]. Available: http://sendrato.com/. [Geopend 14 September 2015].

[3] OrenD, „How can a team working on research based items work in an Agile project?," [Online]. Available: http://pm.stackexchange.com/questions/1358/how-can-a-team-working-on-research-based-items-work-in-an-agile-project. [Geopend 16 September 2015].

[4] M. Hicks, „Adapting Scrum to Managing a Research Group," Department of Computer Science, [Online]. Available: http://www.cs.umd.edu/~mwh/papers/score.pdf. [Geopend 16 September 2015].

[5] „Scrumalliance," Scrumalliance, [Online]. Available: https://www.scrumalliance.org/. [Geopend 16 September 2015].

[6] MeLearning, „Waterval versus Agile (cyclisch) projectmanagement," MeLearning, [Online]. Available: https://www.projectmanagement-training.nl/boek/waterval-versus-agile/. [Geopend 16 September 2015].

[7] Ambysoft, „Introduction to Test Driven Development (TDD)," Ambysoft Inc, [Online]. Available: http://www.agiledata.org/essays/tdd.html. [Geopend 16 September 2015].

[8] D. Wells, „Extreme Programming:," [Online]. Available: http://www.extremeprogramming.org/. [Geopend 16 September 2015].

[9] P. Kruchten, The Rational Unified Process An Introduction, Pearson Education, 2007.

[10] „What is OpenUP?," Eclipse Foundation, [Online]. Available: http://epf.eclipse.org/wikis/openup/. [Geopend 16 September 2015].

[11] „What is Scrum," Scrumalliance, [Online]. Available: https://www.scrumalliance.org/why-scrum. [Geopend 16 September 2015].

[12] „Het W-model," [Online]. Available: http://www.smartest.nl/verdieping/wmodel. [Geopend 18 September 2015].

[13] „Git," [Online]. Available: https://git-scm.com/. [Geopend 18 September 2015].

[14] „iceScrum, Open Source Scrum & Agile project management tool," Kagilum SAS, [Online]. Available: https://www.icescrum.com/. [Geopend 18 September 2015].

[15] „openFrameworks," [Online]. Available: www.openframeworks.cc. [Geopend 21 September 2015].

[16] „Connected Worlds – Interactive ecosystem for NYSCI by Design I/O," Cambridge (US), [Online]. Available: http://www.creativeapplications.net/openframeworks/connected-worlds-interactive-ecosystem-for-nysci-by-design-io/. [Geopend 28 September 2015].

[17] I. Lab, „NO_THING – An infrared light framework that turns (almost) anything into a device," Milla & Partner GmbH, [Online]. Available: http://www.creativeapplications.net/openframeworks/no_thing/. [Geopend 28 September 2015].

[18] M. Wheeler, „This City – Audio-visual performance of a simulated world," [Online]. Available: http://www.creativeapplications.net/openframeworks/this-city-audio-visual-performance-of-a-simulated-world/. [Geopend 28 September 2015].

[19] N. Hardeman, „Deserve – Interactive visual experiment for Bruzed by Nick Hardeman," [Online]. Available: http://www.creativeapplications.net/openframeworks/deserve-interactive-visual-experiment-for-bruzed-by-nick-hardeman/. [Geopend 28 September 2015].

[20] „Specification for the Art-Net 3 Ethernet Communication Protocol," Artistic Licence Holdings, [Online]. Available: http://www.artisticlicence.com/WebSiteMaster/User%20Guides/art-net.pdf. [Geopend 28 September 2015].

[21] T. Smeets, *Positie moving head in de ruimte,* Haute Technique.

[22] „Get Unity," Unity, [Online]. Available: http://unity3d.com/get-unity. [Geopend 21 September 2015].

[23] „FREQUENTLY ASKED QUESTIONS (FAQ)," EPIC GAMES, [Online]. Available: https://www.unrealengine.com/faq. [Geopend 21 September 2015].

[24] „4.6 is released with source for UI system," Unity, [Online]. Available: http://blogs.unity3d.com/2014/11/26/4-6-is-released-with-source-for-ui-system/. [Geopend 21 September 2015].

[25] „Cinder | The library for professional-quality creative coding in C++," Cinder, [Online]. Available: http://libcinder.org/. [Geopend 21 September 2015].

[26] „Greenhouse SDK," oblong industries, [Online]. Available: http://greenhouse.oblong.com/. [Geopend 21 September 2015].

[27] „Torque 3D," GarageGames, [Online]. Available: http://www.garagegames.com/products/torque-3d/. [Geopend 21 September 2015].

[28] „Gamekit," [Online]. Available: https://github.com/gamekit-developers/gamekit. [Geopend 21 September 2015].

[29] „Irrlicht Engine - A free open source 3D engine," [Online]. Available: http://irrlicht.sourceforge.net/. [Geopend 21 September 2015].

[30] „Unity - Game Engine," Unity, [Online]. Available: http://unity3d.com/. [Geopend 21 September 2015].

[31] „Unreal Engine," Unreal, [Online]. Available: https://www.unrealengine.com/. [Geopend 21 September 2015].

[32] „Native Plugins," Unity, [Online]. Available: http://docs.unity3d.com/Manual/NativePlugins.html. [Geopend 5 Oktober 2015].

[33] „Unified Modeling Language," Object Management Group, [Online]. Available: http://www.uml.org/. [Geopend September 21 2015].

[34] „MonoBehaviour," Unity, [Online]. Available: http://docs.unity3d.com/ScriptReference/MonoBehaviour.html. [Geopend 10 Oktober 2015].

[35] E. Gamma, Design Patterns Elements of Reusable Object-Oriented Software, Pearson Education, 1994.

[36] „NUnit," NUnit, [Online]. Available: http://www.nunit.org/. [Geopend 12 Oktober 2015].

[37] NLua, „Nlua," [Online]. Available: http://nlua.org/. [Geopend 15 Oktober 2015].

[38] I. Community, „the Python programming language for the .NET Framework," IronPython, [Online]. Available: http://ironpython.net/. [Geopend 15 Oktober 2015].

[39] sebastienros, „Javascript Interpreter for .NET," [Online]. Available: https://github.com/sebastienros/jint. [Geopend 15 Oktober 2015].

[40] UnifyCommunity, „Jurassic Javascript Runtime Interpreter for Unity," UnifyCommunity, [Online]. Available: https://www.assetstore.unity3d.com/en/#!/content/2345. [Geopend 15 Oktober 2015].

[41] L. Kojecký, „Compiling C# Code at Runtime," [Online]. Available: http://www.codeproject.com/Tips/715891/Compiling-Csharp-Code-at-Runtime. [Geopend 15 Oktober 2015].

[42] „Interprocess Communications,” Microsoft, [Online]. Available: https://msdn.microsoft.com/en-us/library/windows/desktop/aa365574(v=vs.85).aspx. [Geopend 26 Oktober 2015].

[43] „NGUI: Next-Gen UI kit,” Tasharen Entertainment, [Online]. Available: http://www.tasharen.com/?page_id=140. [Geopend 26 Oktober 2015].

[44] „MarkUX,” eXmakina, [Online]. Available: http://www.markux.com/. [Geopend 26 Oktober 2015].

[45] „The MVVM Pattern,” Microsoft, [Online]. Available: https://msdn.microsoft.com/en-us/library/hh848246.aspx. [Geopend 26 Oktober 2015].

[46] „Update items in a DataGrid at runtime,” Reddit, [Online]. Available: https://www.reddit.com/r/markuxdev/comments/3rr1fc/update_items_in_a_datagrid_at_runtime/. [Geopend 23 November 2015].

[47] „Cross platform, open source .NET framework,” Mono, [Online]. Available: http://www.mono-project.com/. [Geopend 23 November 2015].

[48] „GameObjects,” Unity, [Online]. Available: http://docs.unity3d.com/Manual/GameObjects.html. [Geopend 10 December 2015].

[49] „Using Components,” Unity, [Online]. Available: http://docs.unity3d.com/Manual/UsingComponents.html. [Geopend 10 December 2015].

[50] „Mathf.SmoothDamp,” Unity, [Online]. Available: http://docs.unity3d.com/ScriptReference/Mathf.SmoothDamp.html . [Geopend 10 December 2015].

[51] „The Dependency Injection Design Pattern,” Microsoft, [Online]. Available: https://msdn.microsoft.com/library/hh323705(v=vs.100).aspx. [Geopend 28 December 2015].

[52] „XML Documentation Comments,” Microsoft, [Online]. Available: https://msdn.microsoft.com/library/b2s063f7(v=vs.100).aspx . [Geopend 28 December 2015].

[53] „Doxygen,” Doxygen, [Online]. Available: http://www.stack.nl/~dimitri/doxygen/. [Geopend 28 December 2015].

[54] G. Fiedler, „How The Go Stone Moves: Rigid Body Dynamics,” GAFFER ON GAMES, [Online]. Available: http://gafferongames.com/virtual-go/how-the-go-stone-moves/. [Geopend 30 December 2015].