

HoGent

Faculteit Bedrijf en Organisatie

React 360, een framework om mobile virtual reality applicaties te bouwen

Michiel Glibert

Scriptie voorgedragen tot het bekomen van de graad van
professionele bachelor in de toegepaste informatica

Promotor:
Johan Van Schoor
Co-promotor:
Jasper Dansercoer

Instelling: —

Academiejaar: 2017-2018

Tweede examenperiode

Faculteit Bedrijf en Organisatie

React 360, een framework om mobile virtual reality applicaties te bouwen

Michiel Glibert

Scriptie voorgedragen tot het bekomen van de graad van
professionele bachelor in de toegepaste informatica

Promotor:
Johan Van Schoor
Co-promotor:
Jasper Dansercoer

Instelling: —

Academiejaar: 2017-2018

Tweede examenperiode

Woord vooraf

Deze bachelorproef werd gemaakt om mijn opleiding toegepaste informatica met succes te kunnen voltooien. Het was voor mij niet gemakkelijk om een goed onderwerp te kunnen definiëren. Maar door de hulp van Karine Samyn, lector aan de hogent, heb ik toch een goed onderwerp kunnen vinden. Zij bracht mij namelijk op de hoogte van React VR. Een framework voor virtual reality applicaties te bouwen in de web browser. Aangezien ik al een gezonde interesse had voor javascript, was het onderwerp dus zeker iets voor mij. In mei 2018 werd de naam van het framework aangepast naar React 360. Dit was vooral een 'rebranding' maar had toch een kleine impact waardoor ik bepaalde delen van mijn bachelorproef moest aanpassen.

Bij het schrijven van deze bachelorproef heb ik hulp gekregen uit meerdere hoeken.

Ik wil beginnen met mijn promotor Johan Van Schoor te bedanken voor de feedback op mijn bachelorproef. Daarnaast wil ik ook zeer graag mijn co-promotor Jasper Danseroer bedanken. Hij heeft mij zeer veel bijgeleerd over het React framework en stond altijd klaar voor mijn vragen.

Uiteraard wil ik ook graag mijn goede vrienden Arne Aers en Maxim Hendrickx bedanken voor het nalezen van mijn bachelorproef en feedback hierop te geven.

Ten slotte wil ik graag nog de mede-studenten Niels Blanckaert en Giel De Clercq bedanken waarmee ik 3 jaar lang de opleiding met plezier heb gevuld en meerder projecten met succes heb afgewerkt.

Samenvatting

React 360 is een virtual reality (VR) framework ontwikkeld door facebook dat gebaseerd is op het al bestaande en frequent gebruikte React JS framework dat tevens ook ontwikkeld is door facebook. Het concept achter virtual reality bestaat al lang maar heeft pas sinds de laatste jaren zijn intrede gemaakt op de consumentenmarkt. Aangezien een virtual reality wereld zich afspeelt in een wereld waar de gebruiker drie dimensionaal en in 360 graden rond zich kan kijken is het een grote uitdaging voor een ontwikkelaar om hiervoor applicaties te gaan ontwikkelen. Het React 360 framework zou hier de oplossing voor moeten bieden en geeft aan de ontwikkelaar de mogelijkheid om aangename en solide virtual reality applicaties te creëren.

In deze bachelorproef werd onderzocht of React 360 een goed framework is voor de ontwikkeling van een volwaardige VR applicatie. We bekeken voor welke doeleinden we het zouden kunnen gebruiken, wat de performantie is, hoe een gebruiker een applicatie ervaart dat gemaakt is met dit framework en of het framework gemakkelijk aan te leren is voor een ontwikkelaar.

In eerste instantie werd een stand van zaken opgesteld over de concepten van virtual reality. Het is belangrijk een goed beeld te hebben van wat deze technologie precies inhoud voor we dieper op React 360 konden ingaan. Een VR applicatie verschilt namelijk veel van een gewone web applicatie. De programmeertalen waarmee React 360 is opgebouwd en de mogelijke alternatieven voor het framework waren hier ook onderdeel van.

Na de stand van zaken begon het eerste deel van het onderzoek. We bekeken kort de belangrijkste componenten van het React JS framework. React 360 is hier op gebaseerd en dit is dus essentieel om het te kunnen verstaan. Hierna werd het React 360 framework ontleedt en overliepen we alle concepten.

Vervolgens gingen we over tot het tweede deel van het onderzoek. Hiervoor werden twee applicaties ontwikkeld en getest bij 15 personen tussen de 20 en 25 jaar. Elke persoon kreeg 5-10 minuten om beide applicaties uit te testen met een virtual reality headset in combinatie met de smartphone. Tijdens het uitvoeren van dit onderzoek bekeken we ook hoe deze applicaties presteerden op de smartphones en of ze veel van de capaciteiten van de smartphone vroegen. Tot slot werden de resultaten en het besluit van dit onderzoek weergegeven.

Uiteindelijk kunnen we er dan een conclusie opgemaakt die een antwoord formuleert op al de onderzoeks vragen.

Inhoudsopgave

1	Inleiding	15
1.1	Probleemstelling	16
1.2	Onderzoeksvraag	16
1.3	Onderzoeksdoelstelling	17
1.4	Opzet van deze bachelorproef	17
2	Stand van zaken	19
2.1	De virtuele realiteiten	19
2.1.1	Virtual reality	19
2.1.2	3D en 4D	21
2.1.3	Augmented Reality	21
2.2	De werking van virtual reality	22
2.2.1	Stereoscopie	22

2.2.2	De virtuele ervaring	23
2.2.3	Interactie met de virtuele wereld	24
2.3	De gevolgen voor de gebruiker	25
2.3.1	User Experience	25
2.4	JavaScript, HTML en CSS	26
2.4.1	HTML	27
2.4.2	CSS	27
2.4.3	Javascript	28
2.5	De mogelijke alternatieven	32
2.5.1	A-Frame	32
2.5.2	Primrose	32
2.5.3	Agron.js	33
3	Methodologie	35
3.1	Het framework bestuderen	35
3.2	Ervaring van virtual reality voor een gebruiker	35
3.3	De doeleinden en stand van zaken van React 360	36
4	React 360	37
4.1	React.js	38
4.1.1	JSX	38
4.1.2	Components en props	38
4.1.3	State en levenscyclus	40
4.1.4	Events	41
4.2	Werking van React 360	42

4.3	Enkele belangrijke componenten	43
4.3.1	View	43
4.3.2	Text	44
4.3.3	Image	44
4.4	Concepten	45
4.4.1	360° foto's en video's	45
4.4.2	Het ruimtelijke systeem	46
4.4.3	Geluid	49
4.4.4	Interactie	49
5	React 360 applicaties testen	53
5.1	Applicatie 1: Omgeving simuleren	53
5.2	Applicatie 2: Een interactieve applicatie	54
5.3	Prestaties	55
5.4	Resultaten bevraging	55
6	Conclusie	59
A	Onderzoeksvoorstel	61
A.1	Introductie	61
A.2	State-of-the-art	61
A.3	Methodologie	62
A.4	Verwachte resultaten	62
A.5	Verwachte conclusies	62

B	Bijlagen	63
B.1	Resultaten React 360 onderzoek	63
	Bibliografie	67

Lijst van figuren

2.1	Stereoscopie voorbeeld.	23
2.2	De hiërarchie van de benodigdheden in virtual reality.	26
2.3	Code fragment 2.3 ingeladen in de browser.	28
4.1	De opbouw van componenten en sub-componenten in React.	39
4.2	De software stack van React 360 (React VR).	42
4.3	Code fragment 4.8 uitgevoerd in de browser.	44
4.4	Een equidistante cilinderprojectie.	46
4.5	Een flexbox container met flex items.	48
4.6	De X, Y en Z-assen.	48
4.7	Een cursor, raycaster en controller	51
5.1	Applicatie 1	54
5.2	Applicatie 2	54
5.3	Een momentopname van het geheugen verbruik van applicatie 1 (links) en applicatie 2 (rechts)	57
5.4	Het batterijverbruik van de 1ste applicatie na 1u en 20 minuten.	58

Lijst van code fragmenten

2.1	HTML Element voorbeeld 1	27
2.2	HTML Element voorbeeld 2	27
2.3	Voorbeeld van een HTML bestand	27
2.4	Voorbeeld van een CSS bestand	28
2.5	Hond object aanmaken in javascript	29
2.6	Eigenschap toevoegen aan een object	29
2.7	Het nieuwe hond object	29
2.8	Een functie met var	30
2.9	Een functie met let	30
2.10	Een functie die twee getallen optelt	31
2.11	Een functie die de som functie oproept.	31
2.12	Voorbeeld van Primrose omgeving	32
4.1	Voorbeeld van een JSX tag	38
4.2	Het component HelloWorld wordt gedefinieerd	38

4.3 Het component Hello wordt gedefinieerd waarbij de property 'name' wordt gebruikt	39
4.4 Het component Hello met de property name die een waarde toegewezen krijgt	39
4.5 Het component Hello die meerdere malen met andere props wordt gebruikt. .	39
4.6 Het temperature component met de bijhorende state.	40
4.7 Een event afhandelen	41
4.8 Een component die een View teruggeeft.	43
4.9 Het Text component met kinderen.	44
4.10 Het Image component met als bron een-afbeelding.jpg.	44
4.11 Een afbeelding als achtergrond instellen in de runtime.	46
4.12 Voorbeeld van een surface	47
4.13 Voorbeeld van een location	48
4.14 Voorbeeld van een Entity component	49
4.15 Voorbeeld van geluid in React 360	49

1. Inleiding

Virtual Reality, of in het Nederlands virtuele werkelijkheid, is een technologie die het mogelijk maakt om een bepaalde omgeving zowel auditief als visueel te simuleren en het gevoel te geven aan de gebruiker dat hij/zij zich werkelijk in die omgeving bevindt. Men gaat letterlijk op de menselijke zintuigen gaan inspelen door middel van elektronica om de gebruiker een gevoel van realiteit te geven die er eigenlijk niet is. Het voornaamste apparaat dat hiervoor gebruikt wordt is een virtual reality bril. Dit is een bril waarmee voor elk oog een beeld van de virtuele wereld wordt weergegeven. Deze brillen hebben meestal ook extra sensoren zoals bijvoorbeeld een gyroscoop die ook alle bewegingen met het hoofd opvolgt en weergeeft in de virtuele wereld.

De toepassingen waar dat virtual reality voor kan gebruikt worden zijn zeer uitgebreid. De bekendste hiervoor is entertainment, zoals gaming. Hiermee kan men simuleren alsof de persoon zich werkelijk in het spel bevindt. Daarnaast kan het ook gebruikt worden in de medische wereld. Men kan virtual reality of kort VR, gaan gebruiken als behandeling tegen bepaalde aandoeningen zoals fobieën en posttraumatische-stressstoornis. Vervolgens kan virtual reality gebruikt worden voor bepaalde ingrepen te gaan simuleren als training. Deze training kan dan een goede voorbereiding zijn op de echte ingreep. Het gebruik van VR als training komt ook voor in andere gebieden dan het medische, zoals in het leger, de ruimtevaart, vliegsimulators ...

Virtual reality heeft een ruim aanbod van toepassingsgebieden waarvoor het gebruikt kan worden. Toch is het ontwikkelen van een virtual reality applicatie een grote uitdaging. Bijgevolg is er nood aan een goed framework die voldoende componenten aanbiedt voor de gewenste doeleinden van de VR-applicatie. React 360 is een door Facebook ontwikkeld framework dat gebruik maakt van React JS (ook door Facebook) dat voor zowel 360° applicaties gebruikt kan worden als voor VR applicaties. Deze bachelorproef legt vooral

de focus op de ontwikkeling van mobile virtual reality applicaties.

1.1 Probleemstelling

Virtual reality is een nog opkomende trend die nog in zijn kinderschoenen staat. De prijzen voor een geavanceerde VR headset kunnen soms oplopen tot 900 euro. Dit is momenteel nog een struikelblok voor de consument om de aankoop van een VR headset uit te stellen, echter is dit niet de enigste. In het artikel van (Abarrera, 2017) wordt goed aangehaald wat precies de redenen zijn waarom VR headsets nog niet volledig aangeslagen zijn bij het publiek. Een ander zeer belangrijk probleem momenteel met VR is het tekort aan ontwikkelaars voor deze technologie. Een virtual reality applicatie is niet hetzelfde als een gewone webapplicatie. Ten eerste moet er letterlijk in 360° gedacht worden aangezien een persoon in VR perfect rond zich moet kunnen kijken. Daarnaast mogen we niet vergeten dat bepaalde beelden die ongepast zijn voor VR een negatieve weerslag kunnen geven aan de gebruiker en zelfs kan leiden tot fysieke pijn. Ten slotte gaat het ontwikkelen van een virtual reality applicatie niet zomaar. Er is al snel nood aan een uitgebreide kennis over het ontwikkelen van 3D omgevingen zoals in videospellen. Dit kan voor een ontwikkelaar geen gemakkelijke taak zijn om dit zomaar aan te leren en dit zorgt er dan weer voor dat ontwikkelaars moeilijker de stap naar het ontwikkelen van virtual reality applicaties nemen.

React 360 zou hier een oplossing voor moeten bieden. Dit framework biedt niet alleen een groot aantal componenten aan waarmee er al snel een solide VR applicatie kan worden ontwikkeld, maar daarnaast wordt er bij React 360 ook gebruik gemaakt van React JS. React JS is al een gekend framework voor het ontwikkelen van interactieve webapplicaties. Dit zorgt ervoor dat de kloof tussen het ontwikkelen van een webapplicatie en een virtual reality applicatie heel wat kleiner wordt. Ook is het zo dat men voor deze applicaties uit te voeren niet verplicht een virtual reality headset hoeft te gebruiken aangezien deze applicaties in 360° via een webbrowser kunnen uitgevoerd worden en het VR gedeelte ervan eerder een uitbreiding erop is. Daarnaast zijn de mobiele virtual headsets, waarop we de nadruk gaan leggen in dit onderzoek, veel goedkoper. Hier spreken we eerder over prijzen van 20 tot 50 euro. De vraag is echter of React 360 wel een goed framework is in zijn huidige vorm. Kunnen we er wel goede VR applicaties mee ontwikkelen die gebruiksvriendelijk zijn en daarnaast weten we nog niet precies voor welke doeleinden het framework zou kunnen gebruikt worden. Een computer moet veel meer berekeningen doen voor een 3D omgeving te kunnen creëren. Is er dan ook nood aan goede en performante hardware?

1.2 Onderzoeksraag

Deze bachelorproef zoekt antwoorden op volgende onderzoeksraag:

- Hoe ervaart een gebruiker een VR-applicatie ontwikkeld met React 360 ten opzichte

van een gewone webapplicatie?

- Hoe voelt de user experience aan?
- Zijn er fysieke klachten aanwezig?
- Wat is de performantie van React 360 in de browser?
 - Is er nood aan dure hardware voor React 360?
- Voor welke doeleinden kan React 360 gebruikt worden?
- Is React 360 al een goed framework voor volwaardige VR-applicaties?
- Is React 360 voor ontwikkelaars makkelijk aan te leren?

1.3 Onderzoeksdoelstelling

In dit onderzoek zochten we een antwoord op al de onderzoeks vragen en vooral of React 360 een goed framework is voor gebruikersvriendelijke en solide virtual reality applicaties te ontwikkelen en voor welke doeleinden. Het antwoord hoeft niet noodzakelijk positief te zijn aangezien het belangrijkste was om een duidelijk antwoord te formuleren. We wouden vooral het framework gaan ontleden en alle belangrijkste aspecten dat het aanbiedt bekijken terwijl we ook rekening hielden met alle concepten van de wereld van virtual reality. We hielden ook in gedachten dat virtual reality een nieuwe technologie is. Het concept hierachter bestond al lang op het moment van het onderzoek, maar het was pas sinds de laatste jaren dat het zijn intrede had gemaakt op de markt.

1.4 Opzet van deze bachelorproef

De rest van deze bachelorproef is als volgt opgebouwd:

In hoofdstuk 2 wordt een overzicht gegeven van de stand van zaken binnen het onderzoeks domein, op basis van een literatuurstudie. Onder andere, de technologie virtual reality zelf en de programmeertalen waarmee React 360 gebouwd is: HTML, CSS en Javascript.

In hoofdstuk 3 wordt de methodologie toegelicht en worden de gebruikte onderzoekstechnieken besproken om een antwoord te kunnen formuleren op de onderzoeks vragen.

In hoofdstuk 4 wordt React 360 bestudeert en bekijken we enkele belangrijke onderwerpen van het framework.

In hoofdstuk 5 worden de resultaten van de bevraging gegeven. Dit onderzoek bestond uit twee applicaties die getest werden door een 15-tal personen. Daarnaast wordt ook kort weergegeven hoe deze applicaties presteerden.

In hoofdstuk 6 wordt tenslotte de conclusie gegeven en een antwoord geformuleerd op de onderzoeks vragen. Daarbij wordt ook een aanzet gegeven voor mogelijke toekomstige onderzoeken binnen dit domein.

2. Stand van zaken

Vooraleer React 360 kan worden bekeken is het belangrijk een goed beeld te scheppen van de technologie virtual reality zelf. In dit hoofdstuk wordt er dieper ingegaan op de concepten van virtual reality, hoe het werkt en wat de gevolgen zijn voor de gebruiker van deze soort applicaties. Daarna worden de programmeertalen bekeken waarmee het framework werkt: HTML, CSS en Javascript. Tot slot wordt er nog een overzicht gegeven van de mogelijke alternatieven voor React 360 met daarbij een korte uitleg van deze frameworks.

2.1 De virtuele realiteiten

We weten al wat virtual reality is, maar wanneer is nu iets 'virtual reality'. Er zijn namelijk nog een aantal andere vormen van valse realiteiten die al snel verward kunnen worden met de term virtual reality. Wat zijn deze nu precies en wanneer kunnen we iets als VR aanzien?

2.1.1 Virtual reality

Om gemakkelijk te kunnen aantonen wanneer nu iets virtual reality is, maken we gebruik van de vier sleutel elementen van virtual reality die Sherman in zijn boek aanhaalt. (Sherman, 2000)

Een virtuele wereld

Mensen kunnen in hun hoofd dingen voorstellen die enkel zij op een bepaalde manier kunnen voorstellen. Als iemand aan een olifant denkt, zal dit voor elke andere persoon een ander beeld zijn in zijn of haar hoofd. Dit kunnen ook bepaalde omgevingen of objecten zijn die niet bestaan, die fictie zijn. Dit noemt men ook wel een virtuele wereld. Een virtuele wereld hoeft niet noodzakelijk afgebeeld te worden op een computer, men kan deze virtuele wereld al gaan beschrijven op andere manieren zoals een film een script heeft die de film verteld, maar dat is natuurlijk nog niet de afgewerkte film.

Het eerste sleutel element is zeer duidelijk en vloeit rechtstreeks van de naam virtual reality zelf. Met een virtuele wereld wordt niet noodzakelijk bedoeld dat men iets gaat gaan maken dat niet bestaat, maar eerder iets gaan bedenken dat niet binnen handbereik is. Bijvoorbeeld de planeet Mars. Die bestaat natuurlijk wel echt, echter is Mars 227 900 000 km van ons verwijderd. Door virtual reality kan een persoon toch de oppervlakte van Mars trachten te ervaren zoals het werkelijk is. Het is natuurlijk ook toegestaan om iets te gaan bouwen dat niet bestaat, maar dat heeft ook zo zijn limieten en hier geven we in hoofdstuk 2.3.1 een duidelijke verklaring voor.

Immersion/Onderdompeling

Bij virtual reality is het belangrijk dat de gebruiker wordt ondergedompeld in iets anders dan de realiteit. Men moet zich eigenlijk van de echte wereld kunnen afscheiden en ten volle kunnen opgaan in de virtuele wereld. De gebruiker moet zich echt aanwezig kunnen voelen in de virtuele wereld waarbij prikkels van buitenaf zo beperkt mogelijk moeten blijven. Wat ook belangrijk is, is dat men op deze wereld interacties kan uitvoeren. Als men een boek leest kan men zich de wereld wel inbeelden, maar zal alles volgens een vaste lijn gaan. In een virtuele wereld heeft de gebruiker zelf de keuze wat hij/zij doet. Er is een communicatie aanwezig in twee richtingen. De gebruiker reageert op de virtuele wereld, terwijl de virtuele wereld terug reageert op de gebruiker. Bij een boek is dit natuurlijk niet mogelijk. De onderdompeling kan onderscheiden worden in twee soorten:

- **De mentale onderdompeling** is waarbij de gebruiker met het hoofd diep geëngageerd zit in de wereld en alles wat er in gebeurt, begint te geloven.
- **De fysieke onderdompeling** is waarbij het lichaam van de gebruiker in de virtuele wereld terechtkomt. Hierbij gaat men vooral op de menselijke zintuigen en bewegingen inspelen.

Feedback voor de zintuigen

Om de fysieke onderdompeling te ondersteunen, is er nood aan feedback voor de zintuigen. Een droom bijvoorbeeld, is ook niet de realiteit, nochtans is het mogelijk om in een droom bepaalde gevoelens of zelfs pijn te voelen. Ook bij virtual reality moet de ontwikkelaar ervoor zorgen dat het lichaam van de gebruiker zoveel mogelijk denkt dat het zich echt in de virtuele wereld bevindt. Momenteel, bij de huidige generatie van virtual reality apparaten, is er al een heleboel feedback aanwezig. Ten eerste is er de feedback op de

ogen, de gebruiker ziet de wereld. Dan is er de feedback op het gehoor, de gebruiker hoort de wereld. Ten slotte zijn er de bewegingen die de gebruiker met de handen uitvoert. Deze kunnen dan gereflecteerd worden in de virtuele wereld.

Interactiviteit

Hier komen de drie voorgaande elementen samen. Zoals eerder vermeld, heeft virtual reality nood aan communicatie die in twee richtingen gaat. De gebruikers willen interacties kunnen uitvoeren op de virtuele wereld en deze ook te zien krijgen. Net zoals er bij videospelletjes op een beeldscherm bepaalde knoppen kunnen worden ingeduwd om zo een respons te krijgen. Het is zeer belangrijk dat deze interactie op een correcte manier gebeurt zodat dit geen ongemakken veroorzaakt bij de gebruiker. Dit wordt in hoofdstuk 3.2 uitgebreid aangehaald.

2.1.2 3D en 4D

3D is de afkorting voor driedimensionaal. Hiermee bedoelt men dat iets drie meetkundige dimensies heeft, namelijk diepte, breedte en hoogte. Deze technologie kan men dan gaan toepassen op beeldschermen a.d.h.v. stereoscopie, waar in hoofdstuk 2.2 wat meer uitleg over gegeven wordt. Op deze manier krijgt de gebruiker een illusie van diepte. Meestal worden hier dan speciale brillen voor gebruikt, ook wel 3D-brillen genoemd. De klassiekere brillen hebben een kleurfilter, meestal rood en blauw. Rood laat enkel de rode kleuren door en blauw enkel de blauwe kleuren. Hiermee kan men dan vanaf twee perspectieven een blauwe afbeelding en een rode afbeelding gaan weergeven waardoor de diepte van een object zichtbaar is.

3D wordt al veel toegepast in de filmwereld, vooral in de cinema dan. Hierdoor ontstond dan ook de uitgebreidere 4D, maar dit is eerder een marketingterm dan een technologie en wordt bijna uitsluitend in de filmwereld gebruikt als entertainment. Bij 4D gaat men de beelden nog altijd driedimensionaal gaan weergeven, maar zorgt men ook voor extra fysieke effecten die synchroon lopen met de film. Als bijvoorbeeld een film zich in het water afspeelt, kan men de kijker af en toe kleine spatjes water laten aanvoelen.

Het grote verschil hier met virtual reality is de 'immersion'. Deze is amper aanwezig en afhankelijk van de content is er ook weinig feedback voor de zintuigen. Er is wel een extra dimensie, echter kan er niet rondgekeken worden in tegenstelling tot met een VR headset. Interactie met de wereld is ook in de meeste gevallen niet aanwezig. Spelconsoles zoals de Nintendo 3DS vormen dan weer een uitzondering. 3D of 4D is zeker geen volwaardige virtual reality, maar is wel al een stap in de goede richting. (Peniche, 2016)

2.1.3 Augmented Reality

Augmented Reality wordt heel veel verward met virtual reality, maar zijn toch twee andere technologieën. Augmented reality kan vertaald worden naar 'toegevoegde realiteit'. In tegenstelling tot virtual reality gaat men bij augmented reality letterlijk iets gaan toevoegen

aan wat al bestaat. Bij virtual reality gaat men een volledige nieuwe virtuele wereld gaan maken (Augment, 2015). Aangezien augmented reality iets toevoegt via een beeldscherm, is er ook niet verplicht nood aan een bril. Vandaag de dag wordt augmented reality regelmatig gebruikt bij de smartphone. Bijvoorbeeld een bepaald meubilair die virtueel gegeneerd en getoond wordt in een kamer.

Er is niet echt een definitie van wat augmented reality precies inhoud. Eigenlijk is een scorebord bij een live uitzending van sport op de televisie al een vorm van augmented reality. Volgens Azuma (Azuma, 1997) zou iets aan volgende drie karakteristieken moeten voldoen vooraleer er over augmented reality kan gesproken worden:

- Combineert de echte wereld met de virtuele wereld
- Men kan interacties doen met het virtuele
- De virtuele wereld bezit de drie dimensies (diepte, breedte en hoogte)

Als deze drie karakteristieken worden toegepast op het voorbeeld van het scorebord, zien we dat een scorebord bij een live uitzending van sport geen augmented reality is. De eerste voorwaarde wordt wel voldaan, maar de tweede en de derde niet. Men kan namelijk geen interacties gaan uitvoeren op dit scorebord. De programmamaker zou dit dan wel kunnen doen, voor hem is enkel de derde regel niet voldaan. Zou het beeld voor de programmamaker dan ook nog in 3D (een 3D scorebord, niet noodzakelijk een 3D beeldscherm) worden weergegeven, dan kan men stellen dat dit een vorm van augmented reality is.

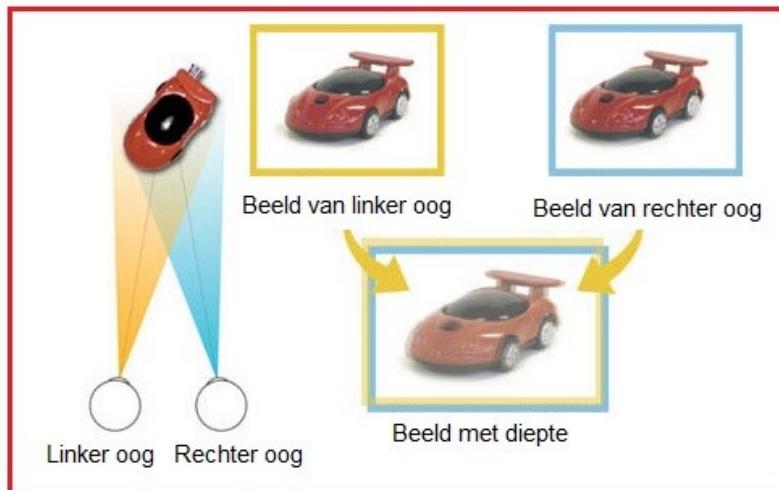
2.2 De werking van virtual reality

Er werd al duidelijk gemaakt wat virtual reality precies is en welke andere gelijkaardige vormen er zijn, maar het is ook belangrijk te weten hoe de technologie werkt. Hier gaan we in tegenstelling tot het vorige hoofdstuk, eerder de nadruk leggen op het technische gedeelte van virtual reality.

2.2.1 Stereoscopie

Bij virtuele werkelijkheid wordt er gebruik gemaakt van illusies. Dit doet men ten eerste a.d.h.v. stereoscopie. Hierbij gaat men diepte meegeven aan een afbeelding. Dit doet men door twee afbeeldingen vanaf een verschillende afstand (meestal de afstand tussen de ogen) te maken. Hierna gaat men dit combineren tot één stereoafbeelding. (Rouse, 2011).

Door stereoscopie kan men de illusie van diepte gaan maken. Zo kan er beter worden ingeschat hoe ver en hoe groot een object is in de virtuele wereld. Deze stereoscopische afbeelding kan dan met dieptezicht getoond worden aan de gebruiker door middel van een VR headset. In deze bachelorproef ligt de focus vooral op React 360 voor mobile VR applicaties. We gaan bijgevolg gebruik maken van een mobile VR headset in dit onderzoek. Bij deze soort VR headset maakt men gebruik van een smartphone die in een hoofddeksel wordt gestoken met speciale lenzen, de VR headset. Op de smartphone worden er dan



Figuur 2.1: Stereoskopie voorbeeld.

twee beelden geprojecteerd. Één voor het linker oog en één voor het rechter oog. De lenzen zorgen ervoor dat de omgeving ruimer lijkt dan het werkelijk is. In combinatie met een gyroscoop kan men dan gaan rondkijken in deze wereld terwijl de smartphone al het rekenwerk doet. Hiermee kan er dan op een goedkope manier, virtuele realiteit getoond worden. Dit is helaas vaak ten koste van de kwaliteit doordat de resolutie op smartphones meestal te laag is voor de virtuele wereld scherp te kunnen weergeven. Er zijn ook meer geavanceerde VR headsets. Hierbij is er per oog een scherm aanwezig met een hoge resolutie. Deze tonen dan elk hun beeld en simuleren de virtuele omgeving. Deze headsets worden het meest gebruikt bij VR videospelletjes. Deze zijn automatisch ook een pak duurder en vereisen een krachtige computer aangezien deze ermee worden verbonden.

2.2.2 De virtuele ervaring

In hoofdstuk 2.1.2 hebben we al vermeld dat het niet enkel door de extra dimensie is dat iets als virtual reality kan gezien worden. Er zijn nog meerdere aspecten waarmee rekening moet gehouden worden om ervoor te zorgen dat de gebruiker een positieve virtuele ervaring beleeft (Mullis, 2016).

Frames per second

De beelden per seconde, ofwel frames per second (FPS) van de beeldschermen in VR headsets, zijn zeer belangrijk. De twee duurste virtuele headsets die momenteel op de markt zijn en dan vooral naar gaming gericht, zijn de HTC vive en de Oculus Rift. Beide headsets maken gebruik van een 90Hz scherm. Dat wil zeggen dat er maximaal 90 beelden per seconde kunnen worden weergegeven. Hoe sneller, hoe realistischer. Een andere bekende headset, de Playstation VR, draait op *slechts* 60Hz, 60 beelden per seconde. Bij smartphones is dit meestal ook 60Hz maar dit kan soms nog lager liggen, tot 30Hz zelfs. Het is duidelijk dat de HTC vive en Oculus Rift een meer realistische ervaring zullen geven dan de Playstation VR of de smartphone.

Latency

Latency heeft ook een zeer grote impact op de ervaring. Latency is de hoeveelheid tijd er tussen zit als de gebruiker een bepaalde input geeft en deze dan wordt weergegeven in de virtuele wereld. Een goed voorbeeld hiervan is het moment dat de gebruiker een stap vooruit zet. Er zal dan een bepaalde hoeveelheid tijd zijn tot men vooruit beweegt in de virtuele wereld. Bij een te hoge latency zal de virtuele ervaring als heel slecht worden ervaren en zelfs vanaf er al een latency is van 20ms zal het menselijk brein duidelijk onderscheid kunnen maken tussen iets dat echt is en iets dat vals is. Dit zal niet alleen leiden tot een mindere ervaring voor de gebruiker, maar kan ook tot fysieke ongemakken leiden zoals motion sickness, bewegingsziekte (Pappas, 2016). Dit is een fysiek ongemak dat ook voorkomt bij onze moderne manieren van transport bij voorbeeld autorijden (wagenziekte). Het is zeer belangrijk dat deze latency zo laag mogelijk is voor een positieve ervaring en om ongemakkelijkheid bij de gebruiker te vermijden.

Field of view

Field of view (FOV), ofwel het gezichtsveld, is zeer belangrijk voor een goede virtuele ervaring. Een mens ziet ongeveer 180° rond zich, maar dit kan oplopen tot 270° als er met de ogen bewogen wordt. De meeste headsets hebben slechts een gezichtsveld tussen de 90° en 110°, wat in principe niet genoeg is. Dit heeft dan ook een impact op de virtuele ervaring en kan ook hier opnieuw leiden tot motion sickness.

Als er niet wordt voldaan aan een hoge FPS, voldoende FOV en een lage latency zal dit een impact hebben op de virtuele ervaring. Hierdoor ontstaan dan fysieke ongemakken omdat het menselijk brein weet dat er iets niet klopt. Het is zeer belangrijk voor ontwikkelaars van virtuele applicaties om hiermee rekening te houden. Als een applicatie de gebruiker ziek maakt, dan gaat de gebruiker het automatisch ook minder of zelfs niet meer gebruiken.

2.2.3 Interactie met de virtuele wereld

Ontwikkelaars zijn al in staat om een virtuele wereld zeer realistisch weer te geven, maar interactie is momenteel nog een moeilijk aspect van virtual reality. Wat men al zeer goed doet, is dat de gebruiker rond zich kan kijken in de virtuele wereld aan de hand van een gyrocoop. Wat dan met rondlopen in die virtuele wereld?

Hiervoor kan de gebruiker simpele gamecontrollers met een joystick gebruiken waarbij men in de virtuele wereld gaat bewegen als hoe men dit zou doen in een videogame. Dit is natuurlijk niet de perfecte ervaring aangezien er weinig rekening wordt gehouden met hoe de handen of voeten bewegen. Om dit dan te gaan oplossen wordt er gebruik gemaakt van motion controllers, bewegingsbesturing. Met deze controllers kan de ontwikkelaar zeer accuraat de bewegingen van de handen van de gebruiker gaan volgen waarbij men ook gebruik maakt van een gyrocoop. Voor de voeten bestaan er ook al oplossingen, maar deze zijn natuurlijk minder praktisch. Er zijn al zogenaamde loopbanden die ervoor zorgen dat de gebruiker kan stappen in de virtuele wereld zonder dat men in een bepaalde richting stapt in de echte wereld. Dit zijn natuurlijk vrij onhandige werktuigen en zijn daarnaast

niet goedkoop. Daarom wordt er voor verplaatsing nog steeds regelmatig gekozen voor een eenvoudige joystick, meestal in combinatie met een motion controller.

Voor mobile is het natuurlijk wat moeilijker om een gamecontroller te gebruiken, aangezien niet elke gebruiker hierover beschikt. Daarom wordt er bij mobile VR de zogenaamde 'gaze' techniek gebruikt. In plaats van dat de gebruiker op een knop gaat drukken is het de bedoeling dat men bepaalde knoppen in de virtuele wereld gaat aanstaren voor een vooraf bepaalde tijd. Dit is een zeer handige oplossing die gemakkelijk kan toegepast worden om het doelpubliek voor een VR applicatie groter te maken. Deze gaze er wel voor dat de gebruiker minder snel kan reageren op bepaalde acties die in de virtuele wereld gebeuren.

2.3 De gevolgen voor de gebruiker

Een virtual reality app is erg verschillend van een gewone webapplicatie. Hierbij is het doel om de gebruiker zich in de wereld te laten voelen. Zoals we al in hoofdstuk 2.2 hebben aangehaald kan een slechte opbouw van een virtuele applicatie lijden tot fysieke ongemakken bij de gebruiker. Hier gaat het dan eerder over onschuldige klachten zoals hoofdpijn, desoriëntatie, misselijkheid, ... Daarom is het belangrijk voor de ontwikkelaar dat hij/zij hiermee rekening houdt bij het ontwikkelen van een VR applicatie. Het is immers niet de bedoeling dat een gebruiker de applicatie niet kan gebruiken omwille van deze fysieke klachten.

2.3.1 User Experience

Het is van belang dat de user experience bij een virtual reality applicatie optimaal is. Hiervoor heeft Beau Cronin (Cronin, 2015) inspiratie gehaald van de piramide van Maslow om zo op te lijsten wat de belangrijkste puntjes zijn voor een ontwikkelaar om ongemak te vermijden.

Ten eerste is er het **comfort**. Deze is eigenlijk al zeer duidelijk aangehaald in hoofdstuk 3.2. Met comfort heeft men het over de ervaring voor de gebruiker die bepaald wordt door de menselijke zintuigen. De manier waarop de zintuigen gaan reageren op de virtuele prikkels is zeer belangrijk voor een positieve ervaring bij deze soort applicaties. De hardware is hiervoor het meeste verantwoordelijk.

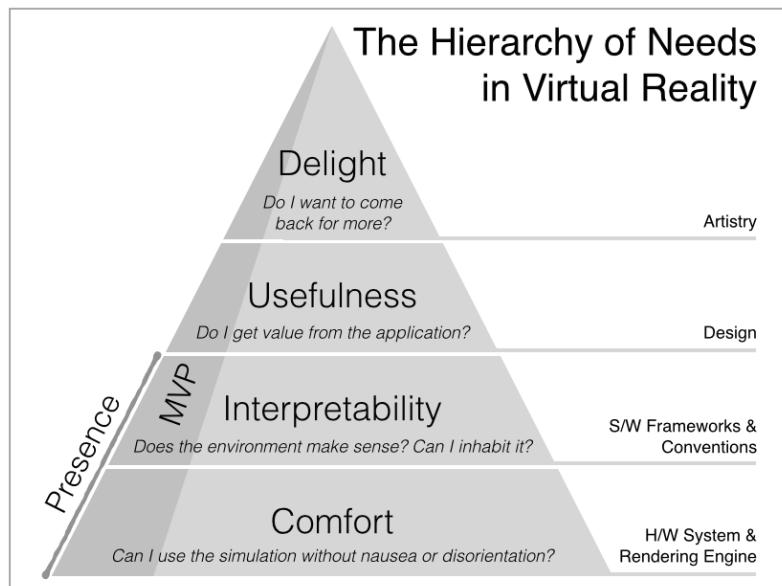
Dan komt de **interpreteerbaarheid**. Hoe realistisch is de virtuele wereld? Hier wordt eerder het verschil bedoeld tussen wat nog non-fictie en fictie is. Een ervaring waarbij de gebruiker door het heelal wordt gezogen op een enorme snelheid met beelden die hij/zij nog nooit eerder gezien heeft zal te overweldigend zijn. De ontwikkelaar kan wel enkele regels van de fysica gaan aanpassen, echter blijft het belangrijk dat deze ervaringen uitbreidingen vormen op het echte leven. Een VR applicatie moet immers nog steeds iets onbestaand, virtueel bezitten.

Vervolgens is er de **bruikbaarheid**. Dit hangt af van applicatie tot applicatie, maar hier gaat het over hoe nuttig iets was, de waarde achter de VR ervaring. Enkele voorbeelden

hiervan zijn: Was het verhaal achter de film realistisch? Was de virtuele wandeling door het huis een aanzet om het huis te kopen? Het gaat hier vooral over de design van de applicatie.

Ten slotte is er nog het **genot**. Dit is dan eigenlijk een uitbreiding op de bruikbaarheid. Hier heeft men het over het feit of de gebruiker meer wilt van de virtuele ervaring. Hier is de aandacht voor detail zeer belangrijk.

Het is duidelijk dat de twee belangrijkste componenten van een goede user experience bij een VR applicatie het comfort en de interpreteerbaarheid zijn aangezien deze vooral de nadruk leggen op de aanwezigheid in de virtuele wereld, de onderdompeling. Bij bruikbaarheid en genot ligt de nadruk meer op het design en de details van de applicatie.



Figuur 2.2: De hiërarchie van de benodigdheden in virtual reality.

2.4 JavaScript, HTML en CSS

Javascript is de taal waarop het framework React JS gebaseerd is. De taal is een objectgeoriënteerde taal, het kan echter ook gebruikt worden als een functionele programmeertaal. Samen met HTML, CSS en Javascript vormt het de basis van een groot aantal websites op het huidige wereldwijde web. Bij HTML gaat het over de inhoud van de pagina, bij CSS over hoe het eruitziet en javascript zorgt dan voor de interactiviteit met de webpagina. Een goed voorbeeld die de werking van javascript aantoont is de animatie bij een uitschuifbaar menu of tekst die verandert als er op een knop wordt geklikt

2.4.1 HTML

HTML staat voor **HyperText Markup Language** en is een opmaaktaal voor documenten die werd uitgebracht in 1993. De taal HTML bestaat uit verschillende HTML elementen. Met behulp van deze elementen, ook wel tags genoemd, kan men structuur gaan creëren in stukken tekst. Een tag ziet er als volgt uit: ``. Met dit element wil men een afbeelding aanhalen in het document. Daarnaast hebben deze elementen attributen. Deze attributen kunnen dan een bepaalde waarde hebben die het HTML element zal beïnvloeden. Bij deze `` tag kunnen we een attribuut `src` toevoegen. Dit attribuut verwijst naar een source, een bron. Deze bron is de locatie van een afbeelding op de computer. Door het element en het attribuut weet HTML precies wat hij moet weergeven. Een HTML element zal altijd als volgt opgebouwd zijn:

```
1 <tag attribute1="value1" attribute2="value2">content  
2 </tag>
```

Code fragment 2.1: HTML Element voorbeeld 1

```
1 <tag attribute1="value1" attribute2="value2" />.
```

Code fragment 2.2: HTML Element voorbeeld 2

HTML elementen kunnen zich ook bevinden binnen andere elementen. Dit wordt ook wel 'nesting' genoemd. De HTML elementen die zich dan binnen in een ander element bevinden noemt men ook wel de kinderen. Dit is een voorbeeld van hoe een zeer eenvoudig en klein HTML document er uit ziet.

```
1 <!DOCTYPE html>  
2 <html>  
3   <head>  
4     <title>Titel van de pagina</title>  
5   </head>  
6   <body>  
7     <h1>Een hoofdtitel</h1>  
8     <h2>Een subtitel</h2>  
9     <p>Een stukje tekst</p>  
10    </body>  
11 </html>
```

Code fragment 2.3: Voorbeeld van een HTML bestand

Het stukje HTML code van hierboven zou er in de browser als volgt uit zien:

2.4.2 CSS

Het enigste wat HTML nog mist is een manier om het document ook mooi voor het oog te maken. De oplossing hiervoor is Cascading Style Sheets, zoals de naam zegt, bladeren met de stijl op. Dit zijn bestanden waar er voor alle HTML elementen een stijl kan worden gegeven. Op deze manier kan men een pagina vormgeving geven dat veel aantrekkelijker is voor het oog. De manier waarop CSS een bepaald HTML element



Figuur 2.3: Code fragment 2.3 ingeladen in de browser.

aanspreekt is door middel van een klasse. Door deze klasse kan het uitzicht van een specifiek HTML element worden aangepast. Het is ook bijvoorbeeld mogelijk om voor alle `<h1>` elementen een globale stijl te voorzien. Men kan een klasse toevoegen aan een HTML element door `class=""` toe te voegen als attribuut. Bij een `<p>` zou dat er dan zo uitzien: `<p class="grootBlauw">`.

Een CSS document is net zoals HTML vrij eenvoudig opgebouwd. In het volgende voorbeeld is er een stijl gegeven aan een klasse en een HTML tag:

```
1 h1 {  
2   font-size: 36pt;  
3   color: red;  
4 }  
5  
6 .grootBlauw {  
7   font-size: 20pt;  
8   color: blue;  
9 }
```

Code fragment 2.4: Voorbeeld van een CSS bestand

In het voorbeeld hebben we ervoor gezorgd dat alle `<h1>` elementen rood zijn en een tekstgrootte zullen hebben van 36pt (point-size). Daarnaast zullen alle elementen die de klasse `.grootBlauw` bevatten een tekstgrootte van 20pt en een blauwe kleur hebben.

2.4.3 Javascript

Javascript is de meest geavanceerde van de drie en ook het belangrijkste voor deze bachelorproef. Met javascript kan men een pagina gemaakt met HTML, interactief gaan maken en is om die reden ook een volwaardige programmeertaal. Javascript heeft zoals elke programmeertaal bepaalde aspecten waar het zich onderscheidt van de andere talen. Javascript is vooral populair om het feit dat het door zeer veel browsers ondersteund wordt. Daarnaast is javascript ook een zeer flexibele taal. Daar waar een andere taal al snel een error voor zou aanrekenen, gaat javascript dit net niet doen. We bekijken de belangrijkste onderdelen die javascript aanbiedt.

Objecten

Zoals eerder al vermeld is javascript een object georiënteerde taal. Dat wil zeggen dat de taal met zogenaamde objecten werkt. Deze objecten kunnen vergelijkt worden met objecten van het echte leven en zijn verzamelingen van bepaalde eigenschappen die dan op zich een eigen name/key (sleutel) en value (waarde) hebben. De waarde van een eigenschap kan onder andere een getal zijn, maar ook een functie of een ander object kan perfect de waarde van een eigenschap zijn. Javascript heeft al enkele vooraf gedefinieerde objecten, maar men kan ook eigen objecten aanmaken.

Laten we dit beter tonen a.d.h.v. een voorbeeld. We gaan een hond object gaan definiëren in javascript. In code ziet dat er als volgt uit:

```

1 var hond = {
2   naam: "Cyra",
3   leeftijd: 3,
4   ras: "Labrador",
5   kleur: "Zwart"
6 }
```

Code fragment 2.5: Hond object aanmaken in javascript

In dit stukje code hebben we een hond object aangemaakt met de eigenschappen: naam, leeftijd, ras en kleur. Bij andere object georiënteerde talen zou het moeilijker zijn om nu nog een extra eigenschap aan het hond object toe te voegen. Er zouden al snel andere programmeer technieken moeten gebruikt worden zoals overerving. Hier kan de flexibiliteit van javascript goed bewezen worden aangezien we de eigenschap oogkleur zeer gemakkelijk aan het hond object kunnen toevoegen:

```

1 hond.oogkleur = "Blauw"
```

Code fragment 2.6: Eigenschap toevoegen aan een object

Het eerder aangemaakte hond object zal er dan zo uitzien:

```

1 {
2   naam: "Cyra",
3   leeftijd: 3,
4   ras: "Labrador",
5   kleur: "Zwart",
6   oogkleur: "Blauw"
7 }
```

Code fragment 2.7: Het nieuwe hond object

Variabelen

Bij hoofdstuk 2.4.3 werd het hond object aangemaakt door het stukje code '`var` hond' te gebruiken. Dit is een variabele. Variabelen zijn vergelijkbaar met de letters van uit de wiskunde, ze houden een bepaalde waarde bij. Een variabele kan aangemaakt worden door eerst aan te geven welke soort variabele men wilt. In vele andere talen moet voor een

nummer een andere soort declaratie gebruikt worden dan als men een stuk tekst in een variabele wilt onderbrengen. Ook hier is opnieuw de flexibiliteit zichtbaar die javascript te bieden heeft. Javascript heeft drie verschillende soorten declaraties voor een variabele waarbij het niet uitmaakt of het tekst of een nummer (met of zonder komma) is:

- **var**: Declareert een variabele, maar een waarde is optioneel.
- **let**: Declareert een lokale, 'block-scoped' variabele, maar een waarde is optioneel.
- **const**: Declareert een lokale, 'block-scoped' variabele, maar een waarde is verplicht en read-only.

Var is de meest flexibele declaratie. Bij een var zal de 'scope' binnen de gehele functie zijn. Bij een let daarentegen, zal dit enkel binnen het blok zichtbaar zijn. Met dit volgend voorbeeld is het goed aantoonbaar wat men precies bedoelt met *de scope*:

```

1 function eenFunctieMetVar() {
2   //eenVar is hier zichtbaar
3   for (var eenVar = 0; eenVar < 10; eenVar++) {
4     //eenVar is hier zichtbaar
5   }
6   //eenVar is hier zichtbaar
7 }
```

Code fragment 2.8: Een functie met var

```

1 function eenFunctieMetLet() {
2   //eenLet is hier NIET zichtbaar
3   for (var eenLet = 0; eenLet < 10; eenLet++) {
4     //eenLet is hier zichtbaar
5   }
6   //eenLet is hier NIET zichtbaar
7 }
```

Code fragment 2.9: Een functie met let

In het voorbeeld kunnen we zien dat een let een lokale declaratie is, ook wel een block-scoped variabele genoemd. Een const is hetzelfde als een let, maar daarbij is het verplicht om een waarde te geven als het gedeclareerd wordt. Dit is omdat een const read-only is. Het kan enkel gelezen worden en achteraf niet meer worden gewijzigd.

Indien een variabele geen waarde krijgt, zoals mogelijk is bij een var, dan zal dit de waarde `undefined` toegewezen krijgen.

Functies

De echte core functionaliteit van javascript zijn functies. Een functie is een verzameling van verschillende declaraties, statements die bepaalde taken uitvoeren. Functies zorgen voor een duidelijke structuur in een javascript bestand en vermijden duplicate code. Javascript zelf heeft al een aantal standaard gedefinieerde functies, maar zoals bij objecten kunnen deze functies ook zelf geschreven worden. Dit is een voorbeeld van een eenvoudige functie die twee nummers optelt:

```
1 function som(nummer1, nummer2) {  
2     var oplossing = nummer1 + nummer 2;  
3     return oplossing;  
4 }
```

Code fragment 2.10: Een functie die twee getallen optelt

Men kan aanhalen dat iets een functie is door 'function' in de code te gebruiken. Daarna geeft men een naam voor de functie, in dit geval is dat 'som'. Vervolgens gaat men aan deze functie twee parameters meegeven, nummer1 en nummer2. Uiteindelijk kan men de functie gaan uitvoeren en het resultaat van de optelling teruggeven. Om gebruik te gaan maken van deze functie, kan de functie worden opgeroepen in andere functies door '()' met de naam van de functie te gebruiken met de nodige parameters.

```
1 function eenAndereFunctie() {  
2     var optelling = som(5,10);  
3     console.log(optelling)  
4 }
```

Code fragment 2.11: Een functie die de som functie oproept.

Deze oproep van de functie zou dan de optelling gaan maken van $5 + 10$ en vervolgens dit gaan weergeven in de console. Doormiddel van de `console.log(...)` functie kunnen we een bepaalde waarde gaan weergeven in de console. Dit is zeer handig om fouten te gaan opsporen binnenin de code. De console kan gezien worden als de uitlaatklep van javascript. Als de code zou crashen, kan javascript a.d.h.v. de console weergeven wat de fout precies was.

Asynchroon programmeren

Een iets geavanceerder aspect van javascript, maar wel een zeer groot pluspunt van de taal, is dat er zeer eenvoudig asynchroon kan geprogrammeerd worden. Asynchroon programmeren wil zeggen dat men meerdere taken tegelijkertijd kan uitvoeren. Dit gebeurt dan doormiddel van een 'Promise'. Dit is een object dat aangeeft dat het een waarde gaat ontvangen, ofwel niet gaat ontvangen. We gaan hier in deze bachelorproef niet verder op in.

Frameworks

De allergrootste reden dat javascript zoveel gebruikt wordt is het grote aanbod aan frameworks. Een framework is eigenlijk een grote verzameling van softwarecomponenten die gebruikt kan worden bij het programmeren van applicaties. Een aantal voorbeelden van bekende javascript frameworks zijn: Angular, React, Vue, Ember en Meteor

Aan het gebruik van frameworks zijn een groot aantal voordelen verbonden (Korotya, 2017):

- **Efficiëntie:** Met een javascript framework kan men veel sneller een goed werkende

website gaan opbouwen met een zeer goede structuur.

- **Veiligheid:** Actieve javascript frameworks worden goed onderhouden door de community en worden ook veel getest, daardoor scoort de veiligheid en robuustheid bij javascript frameworks zeer hoog.
- **Kost:** De meeste javascript frameworks zijn open-source en gratis. Om deze reden en het feit dat het ontwikkelen zo snel gaat, kost het een pak minder voor bedrijven om een goede website te ontwikkelen.

2.5 De mogelijke alternatieven

Het ontwikkelen van virtual reality applicaties gaat natuurlijk niet zomaar en vereisen een goede technische kennis. Zeker het feit dat men in een wereld zit waar er volledig om zich kan worden gekeken en eventueel in bewogen kan worden is natuurlijk een grote uitdaging. Om dit allemaal makkelijker te maken zijn er momenteel al een aantal VR frameworks beschikbaar zoals React 360. Naast dit framework bestaan er nog een aantal andere mogelijke frameworks (Lab, 2017). Wat opvalt is dat enkel A-Frame en React360 momenteel nog actieve projecten zijn.

2.5.1 A-Frame

Het Mozilla team begon met virtual reality in de browser te tonen door middel het A-Frame framework. Met A-Frame kunnen ontwikkelaars 3D werelden gaan bouwen die bestemd zijn voor virtual reality. Doordat A-Frame beschikbaar is in de browser door HTML-elementen te gaan gebruiken is het ook voor meerdere toestellen beschikbaar zoals mobile en desktop. Net zoals React 360, maakt A-Frame gebruik van javascript.

2.5.2 Primrose

Primrose is net zoals A-Frame en React360 een VR framework dat ook in de browser draait door behulp van javascript. Het doel bij Primrose is om letterlijk een omgeving object te gaan maken. Deze code hiervoor ziet er als volgt uit:

```

1 var env = new Primrose.BrowserEnvironment({
2   backgroundColor: 0x000000,
3   groundTexture: "deck.png",
4   useFog: true
5 });

```

Code fragment 2.12: Voorbeeld van Primrose omgeving

Aan deze omgeving kan men dan UI-elementen toevoegen zodat men hiermee interacties kan uitvoeren. Net zoals A-Frame en React360 ondersteund het meerdere toestellen omwille van het feit dat het in de browser wordt uitgevoerd.

2.5.3 Agron.js

AgronJS is niet een VR framework, maar wel een AR (Augmented Reality) framework. Met AgronJS is het mogelijk om AR applicaties in elke browser uit te voeren. Het werd ontwikkeld voor een onderzoeksproject en daarom wordt het momenteel ook niet meer actief geüpdatet. Het werkt zoals alle andere opgenoemde frameworks hier, met javascript.

3. Methodologie

3.1 Het framework bestuderen

We bekeken eerst kort React.JS zelf. Hiermee wordt React 360 namelijk opgebouwd en het was dus cruciaal om hier ook kennis over te hebben. Daarna bekeken we de belangrijkste concepten van het React 360 framework. Zowel de manier van opbouw in de applicatie, als de extra beschikbare componenten in het framework.

3.2 Ervaring van virtual reality voor een gebruiker

Met al de informatie die we verworven hadden, konden we door middel van React 360 enkele eenvoudige applicaties maken. Deze applicaties werden getest door een aantal personen en zij hadden dan hun mening kunnen geven a.d.h.v. een korte vragenlijst. Deze gaf dan een beter inzicht in hoe een persoon een virtual reality applicatie gemaakt met React 360, ervaart.

Applicatie 1

De eerste applicatie was de meest eenvoudige. Dit is een applicatie waarbij de nadruk lag op het simuleren van een echte omgeving. Hiervoor werd een gebied dat echt bestaat in 360° weergegeven met het bijhorende omgevingsgeluid. De gebruiker kon dan ervaren alsof hij echt op die locatie aanwezig was.

Applicatie 2

De tweede applicatie legde meer de nadruk op de interactie met de virtuele wereld. Er werd hier dus vooral gekeken naar de input mogelijkheden en hoe de gebruiker dit ervarde. Zo was de applicatie opgebouwd met verschillende knoppen waarop de gebruiker kon drukken a.d.h.v. een fysieke knop die aanwezig was op de virtuele headset die gebruikt werd voor het onderzoek. Er paste zich dan bepaalde elementen aan in de gebruikersomgeving afhankelijk van wat de gebruiker gekozen had.

Prestaties

Tijdens het testen van deze applicaties hebben we ook gekeken wat de prestaties waren van beide applicaties. Zo hebben we antwoorden kunnen formuleren op vragen zoals: Hoeveel geheugen verbruikten ze? Hoeveel aanslag legden ze op de batterij? Blokkeerde de browser soms?

3.3 De doeleinden en stand van zaken van React 360

Ten slotte werd er dan een besluit opgemaakt van dit onderzoek. Hier werd een stand van zaken opgesteld over hoe React 360 scoort als framework voor virtual reality. Zo konden we een antwoord formuleren op alle onderzoeks vragen.

4. React 360

React 360 is een framework voor het ontwikkelen van VR en 360° applicaties in javascript dat ontwikkeld is door facebook. React 360 wordt gebruikt om gemakkelijk een complexe gebruikersomgeving te kunnen maken in een 360° omgeving. Het grote voordeel aan React 360 is dat het in de browser kan worden uitgevoerd. Dat wil zeggen dat gebruikers met een recente versie van hun browser, React 360 applicaties zonder problemen zouden kunnen uitvoeren. Daarnaast is React 360 op meerdere platformen beschikbaar en is er ook geen verplichting om een VR headset te gebruiken. Men kan React 360 gaan ervaren op zowel een vaste computer, smartphone als op andere VR apparaten (Lehr, 2017).

In React 360 wordt er een combinatie gemaakt van tweedimensionale (enkel breedte en hoogte) elementen met een 3D omgeving. Men gaat 2D interfaces in een 3D wereld gaan plaatsen, echter is er ook de mogelijkheid om 3D objecten hieraan toe te voegen. Enkele voorbeelden waar React 360 kan worden voor gebruikt is panoramische foto's en video's, 360° rondleidingen en zelfs eenvoudige games.

Het grootste voordeel aan React 360 is dat het al gebruik maakt van het bestaande React JS. Voor ontwikkelaars die dit framework al kennen, dat trouwens ook ontwikkeld is door facebook, is de overstap naar het ontwikkelen van VR applicaties een pak eenvoudiger. Bepaalde concepten van React Native komen ook terug in React 360. React Native, ook door facebook ontwikkeld, is een framework dat eveneens gebaseerd is op React JS. Met React Native kunnen mobiele applicaties worden ontwikkeld voor zowel iOS als voor Android toestellen.

4.1 React.js

React JS is een framework dat gemaakt is om user interfaces (gebruikersomgevingen) te gaan ontwikkelen. Hiermee kan er vrij eenvoudig een complexe user interface worden gebouwd waarbij alle data op de juiste manier wordt aangepast door het framework. Het is belangrijk dat we bekijken wat de belangrijkste onderdelen zijn van het React JS framework voor we verder kunnen gaan op React 360. React 360 is immers gebouwd op React JS.

4.1.1 JSX

JSX is een uitbreiding op javascript en is iets dat ongeveer hetzelfde doet als wat een HTML tag doet. Het is een soort van samenvoeging van HTML en javascript. Dit is hoe een JSX tag er uit ziet in javascript code:

```
1 const element = <h1>Hello, world!</h1>;
```

Code fragment 4.1: Voorbeeld van een JSX tag

Men kan aan de hand van JSX, HTML gaan gebruiken in javascript code. Dit maakt het een heel wat eenvoudiger om interfaces te gaan bouwen. In het voorbeeld hebben we de HTML tag `<h1>` met bijhorende tekst in een variabele gestopt, wat natuurlijk niet mogelijk is in een gewoon HTML of javascript bestand zonder JSX. Daarnaast verhoogt JSX de leesbaarheid van de code en is het niet moeilijk om aan te leren als men al HTML en javascript kent.

4.1.2 Components en props

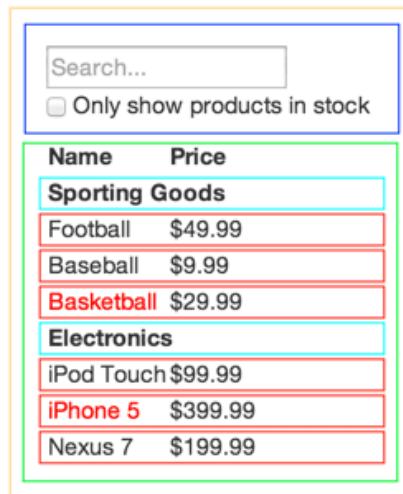
React is opgebouwd uit components die onafhankelijk en herbruikbaar zijn. Met deze components kan een user interface worden opgebouwd. Deze componenten bestaan op zichzelf uit een verzameling van JSX tags. Een user interface is meestal een soort van boom die opgebouwd is uit andere kleinere onderdelen. Aan deze componenten kunnen kleinere sub-componenten worden toegevoegd en zo is er een duidelijke opbouw en structuur aanwezig.

In het voorbeeld zien we hoe een component kan opgebouwd zijn uit meerdere herbruikbare sub-componenten. Dit zorgt voor minder duplicate en meer consistente code. Een component gaan definiëren in React gaat als volgt:

```
1 class HelloWorld extends React.Component {  
2   render() {  
3     return <h1>Hello world!</h1>  
4   }  
5 }
```

Code fragment 4.2: Het component HelloWorld wordt gedefinieerd

Een component moet altijd iets kunnen tonen. Het is verplicht om de `render()` functie te initialiseren. Deze render functie zal dan JSX teruggeven.



Figuur 4.1: De opbouw van componenten en sub-componenten in React.

Het is ook mogelijk om net zoals bij een gewoon HTML element, attributen mee te geven aan een component. In React noemt men dit properties of kort, props. Met deze props is het mogelijk om ervoor te zorgen dat componenten dynamisch zijn en niet enkel vaste data teruggeven. De props van een component kan men aanspreken door `this.props` te gaan gebruiken. In dit voorbeeld maken we een component die gebruik maakt van de property `name`:

```

1 class Hello extends React.Component {
2   render() {
3     return <h1>Hello {this.props.name}!</h1>
4   }
5 }
```

Code fragment 4.3: Het component Hello wordt gedefinieerd waarbij de property 'name' wordt gebruikt

Om dan de property te kunnen gebruiken, kan men dit net zoals bij een gewone HTML tag in het element van het component gaan zetten:

```
1 <Hello name='Arne' />
```

Code fragment 4.4: Het component Hello met de property name die een waarde toegewezen krijgt

Op de webpagina zouden we dan het resultaat 'Hello Arne!' terugkrijgen. Hier tonen we de herbruikbaarheid aan van de React componenten. Als we dit meerdere malen met andere namen zouden willen doen, kunnen we de property name aanpassen zonder dat we een volledig nieuw component hoeven te definiëren:

```

1 <Hello name='Arne' />
2 <Hello name='Maxim' />
3 <Hello name='Rony' />
```

Code fragment 4.5: Het component Hello die meerdere malen met andere props wordt gebruikt.

4.1.3 State en levenscyclus

Men kan door middel van props componenten dynamisch opbouwen met verschillende data. Wat als we nu een component Temperature zouden hebben die het aantal graden teruggeeft en regelmatig verandert. Props zijn niet meer aan te passen nadat het component weergegeven is op de webpagina. Men zou dan steeds een nieuw component moeten gaan aanmaken wat natuurlijk niet handig is. Hiervoor biedt het state object van een component de oplossing. Een state van een component is eigenlijk een object (data) van het component waarvan de eigenschappen naargelang de levenscyclus van het component kunnen veranderen. Zo is het mogelijk om het aantal graden (degree) van het Temperature component te gaan aanpassen elke keer dat deze wijzigt. Zo zou het temperatuur component met een state er dan uitzien:

```

1 class Temperature extends React.Component {
2   constructor() {
3     super()
4     this.state = {
5       degree: 0
6     };
7   }
8
9   setTemperature(amountCelsius) {
10    this.setState({
11      degree: amountCelsius
12    });
13  }
14
15  render() {
16    return <p>{this.state.degree}</p>
17  }
18}
```

Code fragment 4.6: Het temperature component met de bijhorende state.

We zien dat het Temperature component een `constructor()` functie bevat. Dit is een speciale methode die gebruikt wordt om objecten te gaan maken voor de componenten (Chima, 2017). In dit geval gaan we het state object gaan aanmaken met de eigenschap `degree` dat de initiële waarde 0 bevat. We zien ook de functie `setTemperature(amountCelsius)`. Deze functie heeft een parameter en bevat een oproep naar de functie `setState()` waarmee de waarde van de state eigenschappen kan aangepast worden. Stel dat deze functie wordt opgeroepen elke keer dat de temperatuur zou wijzigen, dan kunnen de gewijzigde waarden worden doorgegeven door middel van de parameter `amountCelsius`. Ten slotte wordt de `render()` functie gebruikt om de eigenschap `degree` van het state object te kunnen tonen.

Elk component heeft een zogenaamde levenscyclus. Deze cyclus start vanaf dat het component bestaat tot dat het verdwijnt. Tijdens de levenscyclus worden er automatisch een aantal methodes opgeroepen, de zogenaamde lifecycle hooks. Enkele voorbeelden van deze hooks zijn de `componentDidMount` en `componentWillUnmount`. De `componentDidMount` wordt opgeroepen als de `render()` methode is uitgevoerd. De `componentWillUnmount` wordt uitgevoerd als het component gaat verdwijnen van de pagina.

4.1.4 Events

In React gaat het verwerken van events, interacties van de gebruiker, zeer makkelijk. Bij een element kan je een `onClick` property gaan toevoegen die verwijst naar de methode die het event afhandelt. Daarnaast kan men eventueel ook parameters meegeven aan deze methode:

```
1 class Leeftijd extends React.Component {
2   constructor() {
3     super();
4     this.state = {
5       leeftijd: 0
6     };
7
8     this.handleClick = this.handleClick.bind(this);
9   }
10
11   handleClick() {
12     this.setState({
13       leeftijd: this.state.leeftijd + 1
14     });
15   }
16
17   render() {
18     return (
19       <button onClick={this.handleClick}>
20         Verhoog leeftijd
21       </button>
22     )
23   }
24 }
25 }
```

Code fragment 4.7: Een event afhandelen

Bij dit voorbeeld hebben we een button gedefinieerd met de toegevoegde prop `onClick`. Hierbij verwijzen we naar de methode `handleClick`. Omdat in javascript de methodes nog niet standaard 'gebind' zijn aan een klasse, is het belangrijk dat we dit nog doen zodat we correct naar de methode `handleClick` kunnen verwijzen. Anders zal dit een `undefined` teruggeven.

4.2 Werking van React 360

React 360 is een framework waarmee men op een eenvoudige manier VR applicaties kan gaan maken. Deze VR applicaties spelen zich af in een 3D omgeving. Het is natuurlijk niet zomaar dat React 360 de mogelijkheid heeft om React JS componenten toe te voegen aan deze 3D omgeving. Hiervoor heeft het framework een uitgebreide stack van software libraries (bibliotheeken) om dit mogelijk te maken (Bieronski, 2016).



Figuur 4.2: De software stack van React 360 (React VR).

Een React 360 applicatie bestaat uit twee delen, waaronder de applicatie zelf (de geschreven code) en de 1ste laag onder React 360 op de afbeelding, namelijk de React Runtime. Dit zorgt ervoor dat de React code geconverteerd wordt naar code die de andere lagen begrijpen, echter kan men ook niet-react code in deze runtime schrijven. Daarnaast zorgt het ervoor dat de browser niet vastloopt omdat de react code apart wordt uitgevoerd en dan pas aan de runtime wordt toegevoegd waardoor de beelden per seconde consistent blijven en zo het gevoel van onderdompeling in de virtuele wereld niet verdwijnt. Ten slotte bevat deze runtime nog zogenaamde 'executors'. Deze executors zorgen ervoor dat er nog code in de achtergrond kan worden uitgevoerd, dus niet noodzakelijk enkel in de browser.

Op de 2de laag is OVRUI (Oculus VR User Interface) en Three.js aanwezig. De OVRUI is een library die ervoor zorgt dat bepaalde elementen gemakkelijk kunnen weergegeven worden via Three.js. Dit zijn onder andere tekst, knoppen... Three.js is dan weer een

library die het mogelijk maakt om 3D objecten op een meer eenvoudige manier te kunnen weergeven in de browser.

De laatste laag bestaat uit WebVR en de browser. WebVR is hetgeen wat ervoor zorgt dat een virtual reality experience mogelijk is in de browser en hangt nauw samen met WebGL (waar Three.js ook gebruik van maakt). Dat is een standaard voor 3D objecten in de browser te verkrijgen. Zo kan het herkennen welk type toestel de gebruiker bezit, zorgt het voor de communicatie met VR toestellen en toont het de 'VR Frame', het stuk op de webpagina dat in VR moet worden weergegeven.

React 360 is een verzameling van meerdere technologieën waar men a.d.h.v. React JS gemakkelijk mee kan communiceren. Dit toont duidelijk aan dat React 360 het voor een ontwikkelaar veel eenvoudiger maakt om virtual reality applicaties te gaan maken.

4.3 Enkele belangrijke componenten

React 360 heeft nog enkele belangrijke componenten die niet beschikbaar zijn in de gewone React JS.

4.3.1 View

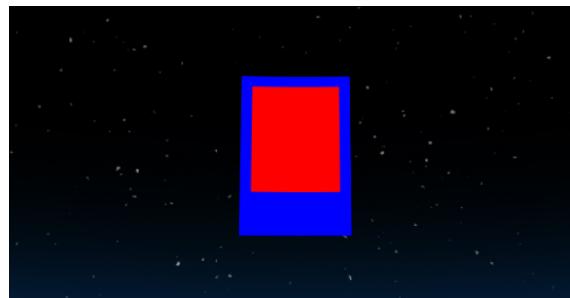
De view is het fundamentele component om in React 360 een user interface te kunnen bouwen. De view is een container die ervoor zorgt dat men elementen kan gaan tonen, stijl kan geven, elementen sorteren en/of groeperen en om interacties op uit te voeren. Een view is vergelijkbaar met een `<div>` element van HTML, maar dan geoptimaliseerd voor React 360. Een view kan als volgt worden gebruikt:

```
1 class BasicView extends React.Component {  
2     render() {  
3         return (  
4             <View  
5                 style={{  
6                     backgroundColor: 'blue',  
7                     height: 150,  
8                     width: 100,  
9                     padding: 10  
10                }}  
11            >  
12            <View style={{backgroundColor: 'red', height: 100}}/>  
13        </View>  
14    );  
15 }  
16 }
```

Code fragment 4.8: Een component die een View teruggeeft.

Hier maken we een simpel component aan die een view retourneert waarbij de achtergrond blauw is, de hoogte 150 pixels, de breedte 100 pixels en bevat een padding van 10 pixels.

De padding is de afstand in pixels tussen de rand aan de binnenkant van het object en de rand van de kinderen van dat object. Dit is mogelijk door een style attribuut toe te voegen aan het view component. Dit is gewoon CSS, maar dan met CamelCasing. In deze view zit nog een andere view met een rode achtergrond met een hoogte van 100 pixels. De breedte hoeft hier niet worden aangegeven omdat de flexbox (zie hoofdstuk 4.4.2).



Figuur 4.3: Code fragment 4.8 uitgevoerd in de browser.

4.3.2 Text

Het Text component is zoals de naam aangeeft om tekst te gaan weergeven in een React 360 omgeving. Dit is vergelijkbaar met het `<p>` element van HTML. Net zoals bij een View kan men hier ook stijl op toepassen.

```

1 <Text>
2   <Text>Dit is</Text>
3   <Text>een stukje tekst</Text>
4 </Text>

```

Code fragment 4.9: Het Text component met kinderen.

4.3.3 Image

Het Image component geeft een afbeelding terug en is vergelijkbaar met het `` element van HTML. Het is gebruikelijk om deze images op te slaan in de 'static_assets' folder die wordt gegenereerd bij het aanmaken van een React 360 project. In volgend voorbeeld gaan we een Image gaan definiëren die zich in de static_assets folder bevindt met de naam een-afbeelding.jpg:

```

1 <View>
2   <Image source={asset('een-afbeelding.jpg')} />
3 </View>

```

Code fragment 4.10: Het Image component met als bron een-afbeelding.jpg.

4.4 Concepten

In deze sectie worden de belangrijkste concepten van React 360 overlopen (Mangialardi, 2017) (Facebook, 2017).

4.4.1 360° foto's en video's

Waar React 360 zeer goed in is, is het tonen van 360° foto's en video's. Deze zijn natuurlijk essentieel voor een realistische omgeving te kunnen creëren. Met deze foto's en video's kan men de gebruikers naar andere omgevingen in de virtuele wereld gaan 'transporteren' en eventueel in combinatie met een interactieve ervaring. Bijvoorbeeld een tour in een museum waarbij de gebruiker naar de verschillende ruimtes kan teleporteren.

React 360 biedt een aantal vormen van formaten aan voor zowel foto's en video's te tonen in 180° of 360°

Foto's

- 360° mono equirectangular
- 360° stereo equirectangular (boven/onder)
- 180° mono equirectangular
- 180° stereo equirectangular (boven/onder)
- 180° stereo equirectangular (links/rechts)

Video's

- 360° mono equirectangular
- 360° stereo equirectangular (boven/onder)
- 180° mono equirectangular
- 180° stereo equirectangular (links/rechts)

Een equirectangular, of in het Nederlands equidistante cilinderprojectie is een projectie van een gebogen oppervlak op een vlak oppervlak. Deze techniek wordt vooral gebruikt bij de wereldkaart.

Het verschil tussen een mono en stereo afbeelding is zoals er in hoofdstuk 2.2.1 werd gesproken over stereoscopie. Bij mono gaat men slechts één afbeelding gaan tonen vanaf één perspectief terwijl men bij stereo twee afbeeldingen vanaf een verschillend perspectief gaat tonen om dieptegevoel mogelijk te maken. Bij React 360 is dit mogelijk door de afbeelding voor het linkse oog van boven (of links bij 180°) te zetten en de afbeelding voor het rechtse oog aan de onderkant (of rechts bij 180°) te zetten en dit dan in één afbeelding te plaatsen. Het is natuurlijk enkel mogelijk om stereoafbeeldingen te bekijken met een VR headset. Indien de gebruiker de applicatie zou uitvoeren zonder een VR headset dan kan men enkel hetgeen zien wat men in het linkse oog zou zien.



Figuur 4.4: Een equidistante cilinderprojectie.

Panorama foto's video's tonen

De ontwikkelaar kan foto's of video's zowel in de runtime (zie hoofdstuk 4.2) als in de React code zelf gaan tonen. Dit kan men voor afbeeldingen doen in de runtime met volgende code:

```
1 r360.compositor.setBackground('./static_assets/image.jpg', {
2   format: '2D',
3 }) ;
```

Code fragment 4.11: Een afbeelding als achtergrond instellen in de runtime.

De code is zeer vanzelfsprekend, echter is er wel een extra argument nodig, namelijk het formaat. Er zijn vier mogelijke formaten die men kan opgeven als argument in de code die overeenstemmen met de eerder genoemde formaten:

- **2D** voor mono afbeeldingen/video's.
- **3DTB** voor stereo afbeeldingen/video's waarbij de afbeelding voor het *linkse* oog aan de bovenkant staat en de afbeelding voor het *rechtse* oog aan de onderkant.
- **3DBT** voor stereo afbeeldingen/video's waarbij de afbeelding voor het *rechtse* oog aan de bovenkant staat en de afbeelding voor het *linkse* oog aan de onderkant.
- **3DLR** voor stereo afbeeldingen/video's waarbij de afbeelding voor het *linkse* oog links staat en de afbeelding voor het *rechtse* oog rechts.

Om video's af te spelen is er een gelijkaardig proces. Hier heeft men ook de keuze om de code in de runtime of in React te schrijven. Het grote verschil bij een afbeelding is dat men bij een video in twee delen zal gaan werken. Aanvankelijk maakt men de videospeler aan die dan een unieke naam moet krijgen en vervolgens gaat men deze videospeler instellen als achtergrond op dezelfde manier als bij een afbeelding.

4.4.2 Het ruimtelijke systeem

Het grootste verschil met een gewone webapplicatie en een VR applicatie is dat deze VR applicaties zich in een 3D ruimte bevinden. Er moet niet enkel worden rekening gehouden

met hoogte en breedte, maar ook met diepte. Dit kan al vrij snel ingewikkeld worden. React 360 heeft ervoor gezorgd dat dit een veel eenvoudiger gaat.

Surfaces

Surfaces (oppervlaktes) bieden de mogelijkheid om 2D user interfaces in een 3D wereld te gaan onderbrengen. Men kan het vergelijken met een stuk van een webpagina in een 3D wereld weer te geven. Dit zorgt ervoor dat het een wat eenvoudiger is voor de ontwikkelaar om een ingewikkelde interface in React 360 te kunnen maken. In de code ziet een surface er zo uit:

```
1 import {Surface} from 'react-360-web';
2
3 const eenSurface = new Surface(
4     500,
5     400,
6     Surface.SurfaceShape.Cylinder
7 );
```

Code fragment 4.12: Voorbeeld van een surface

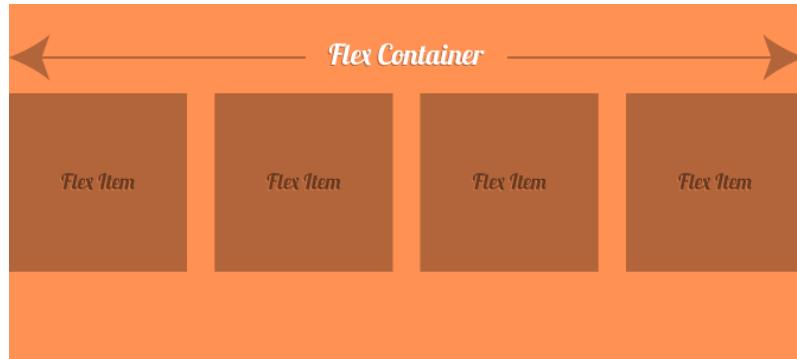
Er wordt een breedte (500) en een hoogte (400) voor de surface gedefinieerd. Daarnaast wordt er ook nog een vorm toegevoegd als argument. Er zijn twee vormen van een surface, namelijk een 'Cylinder' of 'Flat' surface. Bij een cylinder surface gaat men de 2D inhoud gaan projecten in de vorm van een cilinder rondom de gebruiker. Het grote voordeel hieraan is dat de gebruiker vanaf elk perspectief recht op de 2D interface gaat kijken. Bij een flat of vlakke surface gaat men de 2D interface vlak gaan plaatsen. Het verschil hier is dat de inhoud niet altijd vanaf hetzelfde perspectief wordt bekeken door de gebruiker.

Surfaces zijn zeer flexibel. Men kan ze gemakkelijk gaan bewerken door functies zoals de functie `setShape()` te gebruiken. Deze functie zorgt ervoor dat men de surface kan aanpassen van een cilinder surface naar een vlakke surface of omgekeerd. Ook de hoogte en breedte van surfaces kan perfect in de code worden aangepast. Het is perfect mogelijk om meerdere surfaces naast elkaar te gaan gebruiken of om eventueel ook meerdere componenten in één surface onder te brengen.

Aangezien er meerdere componenten kunnen worden ondergebracht in één surface is er ook nood aan een duidelijke manier om deze componenten te ordenen. Dit is op dezelfde manier als bij React Native, namelijk door middel van een flexbox. Een flexbox wordt ook al veel gebruikt in gewone webpagina's en zorgt vooral voor een consistente layout op alle schermgroottes. De belangrijkste eigenschap van een flexbox is de richting van de items die ofwel in een rij of een in kolom geordend zijn. Op het web is dit standaard een rij, terwijl dit bij React Native en React 360 standaard een kolom is.

3D-objecten

Het is ook mogelijk om 3D objecten te gaan gebruiken in React 360. Dit zijn objecten die op voorhand al gemaakt zijn in een ander extern programma van het formaat .obj, .mtl of



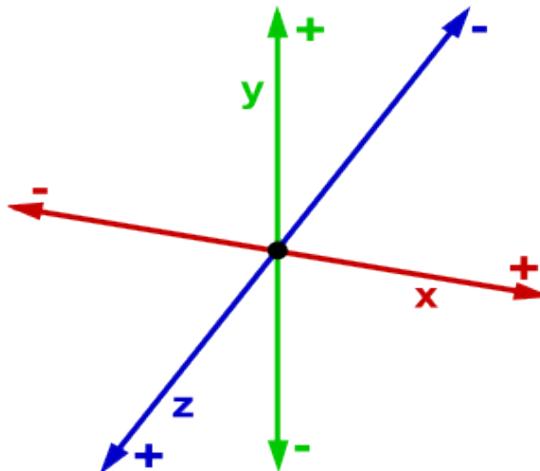
Figuur 4.5: Een flexbox container met flex items.

.gltf2.

Het verschil met 3D objecten en 2D user interfaces, is dat deze niet aan een surface moeten worden toegevoegd, maar aan een 'Location'. Deze locations zijn punten in de omgeving waar men de 3D objecten kan gaan plaatsen. Deze gaan declareren in de code lijkt ook veel op het declareren van een surface, enkel hier is er ook nood aan de X, Y en Z coördinaten. Waarbij X+ hetgeen rechts van de gebruiker voorstelt, Y+ de hoogte en Z+ hetgeen achter de gebruiker. De eenheid die hier gebruikt wordt is meter.

```
1 import {Location} from 'react-360-web';
2
3 const location = new Location([0, -1, -2]);
```

Code fragment 4.13: Voorbeeld van een location



Figuur 4.6: De X, Y en Z-assen.

Om 3D objecten van de opgenoemde formaten in een React 360 applicatie te krijgen moet men gebruik maken van het Entity component. Met dit component kan men ook de grootte en richting van dit 3D object aanpassen. Dit gebeurt door translate of rotateX/rotateY/rotateZ toe te voegen aan de style van het entity component. Bij

rotate worden de graden opgegeven en bij translate de coördinaten ten opzichte van de gebruiker.

```

1 <Entity
2   style={{transform: [
3     {translate: [0, 0, -1]},
4     {rotateY: 90}
5   ]}}>
6   source={{obj: asset('een3DObject.obj')}}
7 />
```

Code fragment 4.14: Voorbeeld van een Entity component

Hierbij draaien we het 3D object met 90 graden volgens de Y-as en verplaatsen we het -1 meter volgens de Z-as. Nu zal het object voor de gebruiker tevoorschijn komen.

4.4.3 Geluid

Net zoals het mogelijk is om afbeeldingen en video's op de achtergrond te gebruiken is het ook perfect mogelijk om de applicatie van achtergrond geluid te voorzien. Bij video's kan men eveneens functies gaan uitvoeren die het volume kunnen aanpassen, audio dempen... De implementatie hiervan is ook zeer gelijkaardig:

```

1 import { asset, NativeModules } from 'react-360';
2 const { AudioModule } = NativeModules;
3
4 AudioModule.playEnvironmental({
5   source: asset('audio.mp3'),
6 });
```

Code fragment 4.15: Voorbeeld van geluid in React 360

Omdat een virtual reality wereld steeds moet communiceren in twee richtingen is het ook belangrijk dat bepaalde interacties met de wereld voldoende feedback teruggeven. Het is daarom in React 360 ook mogelijk om een 'One-shot audio' gebruiken. Dit kan men bereiken door `AudioModule.playOneShot` te gebruiken met de verwijzing naar de audio file als parameter. Het kan dan bijvoorbeeld gebruikt worden bij een functie die wordt opgeroepen door de `onClick` van een button (zie hoofdstuk 4.4.4) waardoor er audio feedback mogelijk is.

Aangezien VR zich in een 3D omgeving bevindt. Is het ook mogelijk om aan een geluid component een parameter `is3d` toe te voegen met de bijhorende locatie (x,y,z) van het geluid. Hierdoor zal het geluid zich specifiek vanaf die coördinaten afspelen.

4.4.4 Interactie

De interactie met de virtuele wereld is natuurlijk zeer belangrijk. Er zijn vele mogelijkheden beschikbaar om input door te zenden naar de virtuele wereld. Hierover hebben we al uitgebreid gesproken in hoofdstuk 2.2.3. Hiervoor biedt React 360 meerdere oplossingen

aan zodat de ontwikkelaar zich vooral kan focussen op de logica van de applicatie en minder op welke manier de interacties verwerkt worden.

Button input

Met de titel wordt niet noodzakelijk een `<Button>` component als input bedoelt, maar eerder een fysieke knop als input. Een toetsenbord, muis, controller, ... hebben allemaal fysieke knoppen ter beschikking. Als deze knoppen worden ingedrukt verwacht de gebruiker een bepaalde feedback van de applicatie. Om dit gemakkelijk te kunnen afhandelen voor alle input mogelijkheden biedt React 360 een `onInput` property aan. Dit is zoals `onClick` een event handler, echter hierbij wordt de zogenaamde `inputEvent` meegestuurd. Dit is een eigenschap van het event dat werd verstuurd waarmee de verschillende acties kunnen worden geïdentificeerd. Zowel de knop (boven, onder, links, rechts, ok ...) en de actie (knop indrukken, knop terug omhoog) die wordt uitgevoerd, worden teruggegeven. Het maakt niet uit welke manier van input de gebruiker gebruikt, als er een 'naar onder' knopactie wordt gedetecteerd zal React 360 de juiste knop en actie terugsturen waardoor het mogelijk is voor de ontwikkelaar om de correcte code hiervoor te schrijven. Dit toont zeer duidelijk de cross-platform mogelijkheden aan van React 360 en spaart voor de ontwikkelaar een hoop tijd uit dan moest hij dit voor elk apparaat individueel moeten programmeren.

VrButton

Het `VrButton` component is niet meer dan een `button` waarbij er wordt gereageerd op de 'confirm' actie. Deze actie komt regelmatig terug en is zoals de naam aanhaalt om iets te gaan bevestigen. Dit component is louter om het gemakkelijker voor de ontwikkelaar te maken. Het enigste verschil is dat hier een `onClick` event handler kan toegevoegd worden.

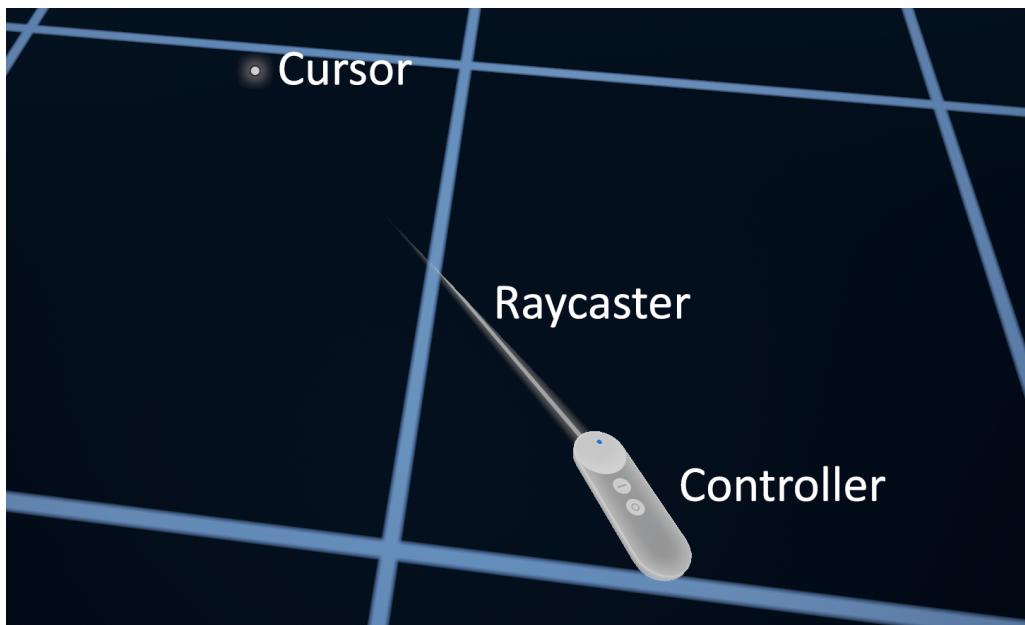
Cursors en raycasters

De gebruiker kan niet correct op de virtuele wereld interacties uitvoeren, als de gebruiker niet weet waar hij/zij de interacties precies uitvoert. React 360 gebruikt hiervoor cursors, net zoals een gewone windows computer een cursor heeft. Sommige controllers worden standaard in React 360 ondersteund en tonen dan een soort van laserstraal in de virtuele wereld. De cursor wordt getoond op het element waar de laserstraal mee snijdt. Om hiervoor ook events op te kunnen uitvoeren, zijn de `onEnter` en `onExit` event handlers aanwezig. De `onEnter` wordt uitgevoerd vanaf dat de cursor het component binnentreedt. De `onExit` doet dan het omgekeerde en wordt opgeroepen als de cursor het component verlaat.

Omdat React 360 niet alle controllers ondersteund, is het ook mogelijk voor een ontwikkelaar om deze zelf te gaan definiëren. Men noemt dit ook wel een raycaster. Een raycaster moet een bepaalde interface gaan volgen om als raycaster gezien te worden:

- De richting van de raycaster
- Of de raycaster een cursor moet tonen op het snijpunt
- Of de raycaster absoluut moet getoond worden tegenover de camera (positie van de

- gebruiker)
- De maximale lengte van de 'laserstraal' van de raycaster
 - Het punt waar de raycaster begint
 - Het type raycaster (touch, mouse, ..)



Figuur 4.7: Een cursor, raycaster en controller

5. React 360 applicaties testen

Het is belangrijk dat we een goed beeld krijgen van hoe de gebruikers een React 360 applicatie ervaren. Het is immers voor deze groep dat we VR applicaties ontwikkelen. Om dit beeld te verkrijgen hebben we een klein onderzoek uitgevoerd a.d.h.v. twee eenvoudige applicaties die ontwikkeld werden in React 360. Deze applicaties leggen vooral de nadruk op de belangrijkste componenten op wat React 360 te bieden heeft.

Dit onderzoek werd afgenoem bij 15 personen tussen de 20 en 25 jaar. Eerst werden de personen onderworpen aan de twee applicaties ontwikkeld met React 360. Zij kregen een mobiele VR headset waar zich een smartphone in bevond en mochten dan beide applicaties gedurende 5 tot 10 minuten uittesten. Daarna werd er een bevraging afgenoem die bestond uit zes korte vragen die vooral een antwoord geven op de user experience van React 360 en dus niet van virtual reality in het algemeen.

5.1 Applicatie 1: Omgeving simuleren

De eerste applicatie was de eenvoudigste waarbij er gebruik werd gemaakt van een mono 360° video (zie hoofdstuk 4.4.1). Een stereo versie zou realistischer geweest zijn omdat er dan dieptegezicht aanwezig was, echter is deze soort video's vinden in goede kwaliteit niet makkelijk. Daarom werd er toch de keuze gemaakt voor een gewone mono 360° video. Naast de aanwezigheid van een video, was er ook natuurlijk omgevingsgeluid aanwezig. De applicatie was vooral bedoelt om de gebruiker zich aanwezig te laten voelen in een realistische omgeving.



Figuur 5.1: Applicatie 1

5.2 Applicatie 2: Een interactieve applicatie

De tweede applicatie was iets geavanceerder. Deze legde vooral de focus op de interactiviteit met de gebruiker. Hier was een user interface aanwezig die volledig gemaakt werd in React 360 met de hulp van React JS. Daarnaast werd er ook gebruik gemaakt van een raycaster (zie hoofdstuk 4.4.4). Deze applicatie is gebaseerd op een voorbeeld die terug te vinden is op de documentatie van React 360, maar had wel lichte aanpassingen gekregen zodat deze offline werkt en volledig compatibel was met de toen huidige versie van Chrome op de smartphone.

De applicatie toont aan de linkerzijde een overzicht van de mogelijke objecten die kunnen gekozen worden met hun titel en auteur. Deze objecten zijn afkomstig van een gratis online database van google genaamd Googly Poly. Men kan één van deze objecten links gaan selecteren en dit object zal dan centraal op het scherm tevoorschijn komen en langzaam ronddraaien. Ten slotte kan men aan de rechterzijde de naam en de auteur nogmaals zien met een bijhorende beschrijving van het object.



Figuur 5.2: Applicatie 2

5.3 Prestaties

De performantie van het framework is ook een belangrijk aspect. We willen immers dat zoveel mogelijk toestellen onze applicatie kunnen gebruiken. Een VR applicatie die met het React 360 framework gebouwd is zou normaal aan 60 beelden per seconde en een lage latency moeten uitgevoerd kunnen worden.

Tijdens het onderzoek maakten we voor applicatie 1 gebruik van volgende smartphone:

OnePlus One

- Schermresolutie: 1920x1080 (Full HD)
- Processor: Qualcomm Snapdragon 801
- RAM geheugen: 3GB
- Besturingssysteem: Android 7.1.1
- Browser: Chrome Canary - versie 68.0.3434.0

Voor applicatie 2 maakten we gebruik van volgende smartphone:

OnePlus 3T

- Schermresolutie: 1920x1080 (Full HD)
- Processor: Qualcomm Snapdragon 821
- RAM geheugen: 6GB
- Besturingssysteem: Android 8.0.0
- Browser: Chrome Canary - versie 68.0.3434.0

Voor beide applicaties op beide toestellen lag het geheugen-verbruik vrij hoog. Voor de eerste applicatie lag het gemiddeld verbruik rond de 580mb. Bij de tweede applicatie lag het nog wat hoger en was het verbruik gemiddeld 900mb. Een gewone webpagina waarbij geen React 360 gebruikt wordt, verbruikt gemiddeld 150-200mb.

Doordat de applicatie dus een grote hoeveelheid van het RAM geheugen innam en ook de processor tot aan het werken zette, was het batterijverbruik van deze applicaties vrij hoog.

We kunnen dus concluderen dat we voor een VR-applicatie uit te voeren een vrij krachtige smartphone nodig hebben met zeker 2GB minimum aan ram geheugen voor een goede virtual reality experience te kunnen ervaren.

5.4 Resultaten bevraging

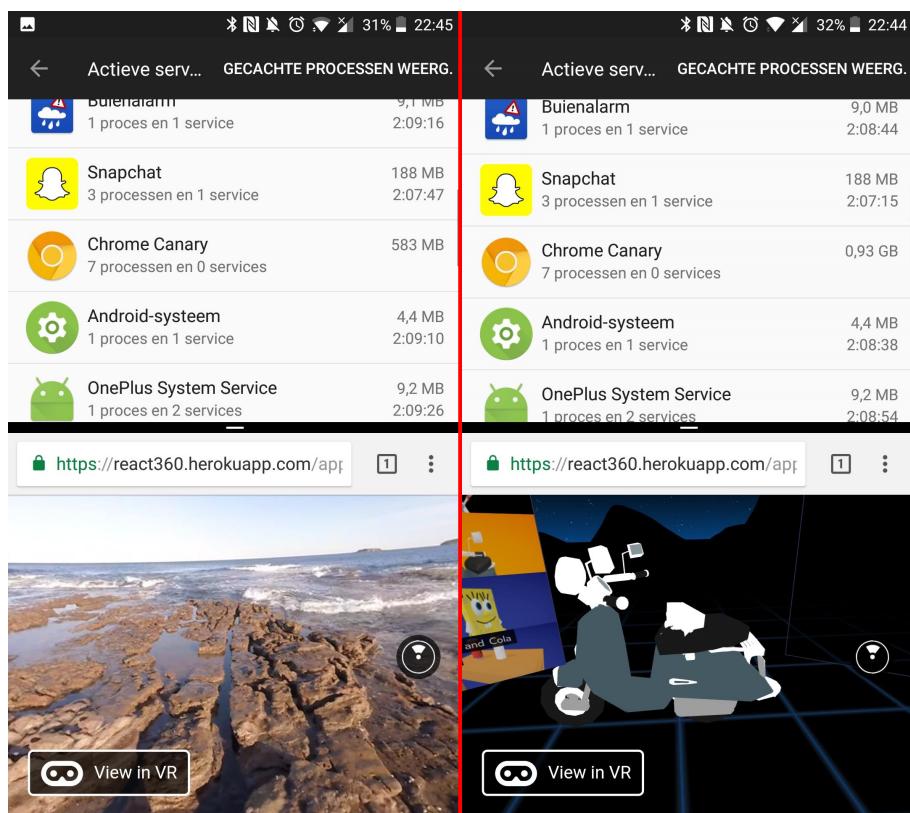
Bij het onderzoek lag de focus grotendeels op de ervaring van de gebruiker bij de React 360 applicaties. Deze vragen werden dus zeker niet te technisch of te algemeen opgesteld. Voor het onderzoek werden volgende zes vragen gesteld:

- Voelde u zich aanwezig op de plaats zelf?

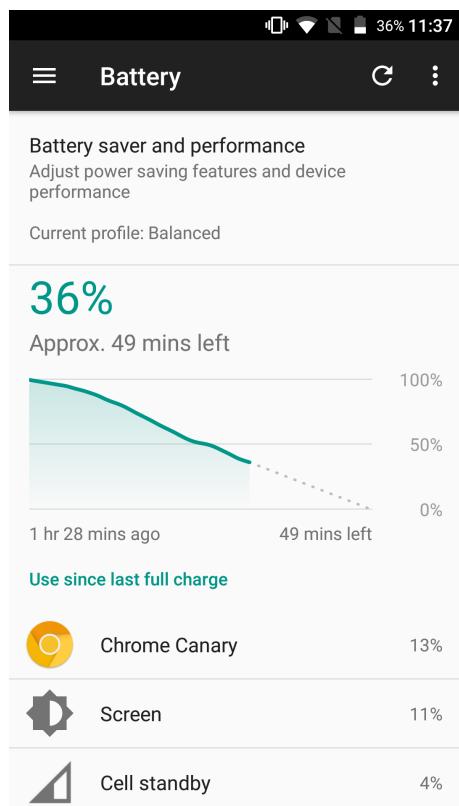
- Vond u de applicatie aangenaam om te gebruiken?
- Heeft u al eerder gebruik gemaakt van virtual reality?
- Vond u het beeld goed van kwaliteit?
- Had u last van fysieke ongemakken? (hoofdpijn, duizeligheid, droge ogen, misselijkheid...)
- Moest u zelf beschikken over een VR headset, zou u dan regelmatig deze soort applicaties gebruiken?

De grafieken en resultaten kan u terugvinden onder bijlage B.1

Volgens de resultaten kunnen we dus enkele dingen gaan concluderen rekening houdend met het feit dat 73,3 % van de ondervraagden al eerder virtual reality had gebruikt. Zo is het ten eerste zeer duidelijk dat React 360 een goede manier is om gebruiksvriendelijke applicaties te maken. 93,3 % van de ondervraagden vond immers dat de tweede applicatie aangenaam was om te gebruiken. Ook al vond dus de meerderheid (73,3 %) dat de kwaliteit van het beeld niet goed was, toch was men nog steeds positief over de gebruikerservaring. Ten tweede zie je zeer duidelijk dat fysieke ongemakken regelmatig voorkwamen. Dit ging van duizeligheid tot ook misselijkheid bij de uitvoering van het onderzoek. Dit was vooral te merken bij de eerste applicatie, waar 60 % zich deels aanwezig voelde. Een mobiele React 360 applicatie is zeker nog niet een goede manier om ervoor te zorgen dat de gebruiker volledig wordt ondergedompeld in de wereld. Dit heeft hoogstwaarschijnlijk te maken met het feit dat de beeldkwaliteit een pak lager is bij mobile VR. Ten slotte zien we wel dat 60 % regelmatig deze soort applicatie zou gebruiken als hij/zij over een VR headset zou beschikken. Dit is positief, maar we kunnen hier ook uit afleiden dat 40 % geen nood heeft aan een VR-ervaring.



Figuur 5.3: Een momentopname van het geheugen verbruik van applicatie 1 (links) en applicatie 2 (rechts)



Figuur 5.4: Het batterijverbruik van de 1ste applicatie na 1u en 20 minuten.

6. Conclusie

Het React 360 framework is een zeer goede aanzet tot de verdere ontwikkeling van virtual reality. Het framework heeft een lage instapdrempel in de zin van dat de ontwikkelaar niet noodzakelijk uitgebreide kennis nodig heeft om een virtual reality applicatie ermee te kunnen ontwikkelen. React JS is een framework dat zeer snel kan worden aangeleerd waardoor het ontwikkelen van deze applicaties ook automatisch heel wat eenvoudiger wordt. Een ingewikkelde applicatie ontwikkelen wordt door middel van het React 360 framework heel wat eenvoudiger gemaakt. Men kan zeer gebruiksvriendelijke interfaces gaan ontwikkelen om hiermee solide applicaties te gaan maken. Dit is echter waar het er momenteel bij React 360 bij blijft. De doeleinden en functionaliteit van applicaties gebouwd met het framework blijven zeer beperkt. We kunnen React 360 vooral gaan gebruiken voor applicaties waarbij nood is aan een goede user interface en die vooral professioneel gericht zijn. Bijvoorbeeld applicaties voor de toeristische sector waarbij men een rondleiding kan geven op bepaalde plaatsen met behulp van VR. Ook immobiliënkantern kunnen gebruik maken van React 360 waarbij men bijvoorbeeld 360° foto's of video's van huizen kan gaan voorstellen. Zelfs de medische sector of scholen in het algemeen kunnen educatieve applicaties ontwikkelen met React 360 waarbij men een 3D weergave kan gaan geven van bepaalde organen. Voor entertainment zoals een film in 360° bekijken, kan men wel React 360 gebruiken, maar veel verder dan dat gaat het niet op vlak van entertainment. Eenvoudige games zijn mogelijk, echter komt React 360 niet in de buurt van wat er al mogelijk is met de andere technologieën en VR headsets (HTC Vive, Oculus Rift, ..) op vlak van gaming.

De user experience was iets waar ik initieel aan twijfelde aangezien er 2D elementen in een 3D wereld worden weergeven. Dit bleek uiteindelijk zeer goed samen te werken. Dit is ook de manier waarop gebruikers het duidelijk graag hebben volgens de resultaten van het onderzoek. Dit is zowel voor de ontwikkelaar als voor de gebruiker een groot

voordeel. Het blijft natuurlijk nog steeds virtual reality. De fysieke opmerkingen werden ook dikwijls duidelijk tijdens het onderzoek. Dit waren regelmatig gebruikers die last hadden van de ogen, maar ook misselijkheid en duizeligheid kwamen een aantal keer voor. Fysieke klachten vermijden bij een VR applicatie is dus een belangrijk werk punt voor de ontwikkelaar.

Bij React 360 is er een actieve community aanwezig. Andere frameworks zoals Primrose hebben geen actieve community en daardoor zal de applicatie ontwikkeling met dat framework ook een pak stroever verlopen door de weinige hulp die aanwezig is bij eventuele problemen. React 360 bezit wel nog een aantal bugs, maar dat is misschien eerder te wijten aan WebVR dan React 360 zelf. Dit was vooral te merken bij het cross-platform gebruik. Zo verliep het uitvoeren van de testapplicaties op een desktop computer met de browser 'Google Chrome' vlekkeloos. Bij de smartphone versie van Chrome werden er dan weer enkele problemen ondervonden zoals het geluid dat niet correct afspeelde. Om deze problemen te kunnen oplossen werd een variant op Chrome gebruikt, namelijk Chrome Canary. Dit toont dan weer aan dat React 360 nog niet 100% cross-platform is. Iets wat Facebook wel aanhaalt op de homepage van het React 360 framework. Wat mij ook duidelijk werd in tegenstelling tot wat ik initieel dacht bij het aanvangen van mijn bachelorproef, is de beperkte documentatie die momenteel aanwezig is op de website. Zo werd in mei 2018 het framework aangepast van React VR naar React 360. De core functionaliteiten bleven hetzelfde en het hele framework kreeg een nieuwe 'look and feel'. Er is een groot aantal functionaliteit die nog van React VR afkomstig is en bijgevolg ook nog steeds werkt in React 360, echter zijn deze nog steeds ongedocumenteerd. React 360 doet het wel goed als hij het doet. Ook al was het batterij- en geheugen verbruik wel hoog. Ik ondervond geen onderbrekingen tijdens het uitvoeren van de applicaties. Daardoor is er niet noodzakelijk nood aan een zeer dure smartphone. Tegenwoordig is de prijs van een smartphone met 2GB RAM-geheugen relatief laag, maar er moet wel een zekere 'kracht' aanwezig zijn.

Al bij al kunnen we stellen dat React 360 zeker een stap in de juiste richting is. Het zet aan tot een verdere ontwikkeling en biedt ook een manier aan om applicaties te ontwikkelen die wel gebruiksvriendelijk zijn. Toch blijft het aantal doeleinden eerder beperkt en is dit enkel bedoelt voor eenvoudige en interactieve applicaties. Willen we een ingewikkeldere applicatie met gebruik van veel 3D elementen en animaties, dan is React 360 niet het ideale framework en zou een beter alternatief het A-Frame framework van Mozilla kunnen zijn. Mijn initiële verwachtingen voor React 360 lagen hoger dan het resultaat van deze bachelorproef aanhaalt.

A. Onderzoeksvoorstel

Het onderwerp van deze bachelorproef is gebaseerd op een onderzoeksvoorstel dat vooraf werd beoordeeld door de promotor. Dat voorstel is opgenomen in deze bijlage.

A.1 Introductie

Virtual reality is een opkomende trend maar staat nog steeds in zijn kinderschoenen. Er is dus nood aan applicaties die gebruik maken van deze technologie. Voor een ontwikkelaar is dit een geheel nieuwe manier van ontwikkelen. React 360 zorgt ervoor dat er aan de hand van de huidige technologie (React JS) een virtual reality webapplicatie kan worden gemaakt. Zo kan de ontwikkelaar dus vooral focussen op de principes die er in de virtual reality wereld heersen en minder op het technische gedeelte. Ik ga onderzoeken of React 360 een geschikt framework is voor gebruiksvriendelijke applicaties te ontwikkelen. Bij virtual reality moet er met een aantal aspecten rekening worden gehouden. Deze applicaties moeten niet alleen een goede user experience bezitten maar ook de gebruiker minimale fysieke last bezorgen. Hier heb ik het dus over duizeligheid, misselijkheid... Ik ga daarnaast ook bekijken voor welke doeleinden het framework gebruikt kan worden. Bijvoorbeeld voor gaming, entertainment, medische sector...

A.2 State-of-the-art

React 360 is nog een vrij nieuwe technologie die nog maar sinds 2017 bestaat. Het is een framework dat net zoals virtual reality zelf, nog volop in ontwikkeling is. De technologie die gebruikt wordt in de browsers om virtual reality via webpagina's te ondersteunen heet

WebVR. De website van React zelf is goed gedocumenteerd en zal dus ook zeker een startpunt vormen voor het verzamelen van informatie. Daarnaast zijn ook enkele boeken ter beschikking, waaronder één van Michael Mangialardi, Learn React 360 (Mangialardi, 2017), die ik ook zal gebruiken zodat ik React 360 leer gebruiken.

A.3 Methodologie

Ik zal mijn onderzoek beginnen door eerst te bekijken wat React 360 allemaal te bieden heeft. Ik zal zelf ook met React 360 leren werken en hiervoor enkele applicaties gaan schrijven. Ik zal deze applicaties testen in zowel 360° als met een VR bril voor mijn smartphone. Deze apps zullen niet alleen door mij getest worden maar ook door een aantal andere mensen waarna ik naar hun meningen zal vragen over enkele onderwerpen die aan virtual reality gebonden zijn. Met deze informatie zal ik gaan kijken wat de mogelijkheden allemaal zijn van het framework en of het dus wel geschikt is in zijn huidige vorm om gebruiksvriendelijke virtual reality applicaties te maken. Hieruit zal ik gaan kijken voor welke doeleinden het framework geschikt is.

A.4 Verwachte resultaten

Ik verwacht dat React 360 voor zowel de ontwikkelaar als voor de gebruiker een geschikt framework is waarmee gebruiksvriendelijke applicaties ontwikkeld kunnen worden die bestemd zijn voor een groot aantal doeleinden. Daarnaast verwacht ik dat de applicaties die ik zal schrijven goed zullen ontvangen worden bij de mensen bij wie ik die laat testen. Maar men zal virtual reality nog eerder zien als een 'gimmick' die niet echt nodig is. Ik verwacht uiteindelijk ook dat er enkele fysieke lasten zullen optreden.

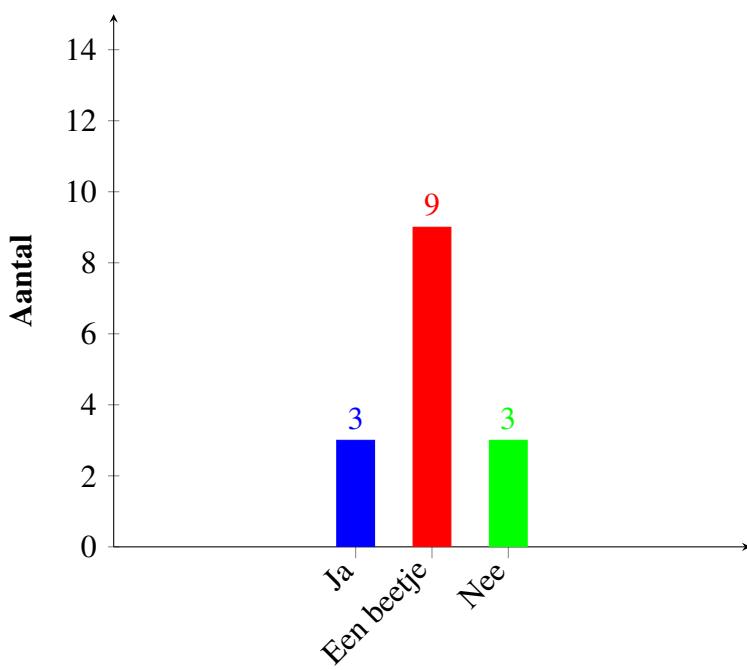
A.5 Verwachte conclusies

Ik denk dat uit mijn onderzoek zal blijken React 360 een geschikt framework is voor gebruiksvriendelijke applicaties te ontwikkelen die gebruikt kunnen worden voor een groot aantal doeleinden. Alhoewel ik niet verwacht dat het framework zelf zeer uitgebreid zal zijn, verwacht ik wel dat het een zeer goede basis is waar zeker nog in de toekomst verder op kan worden gebouwd.

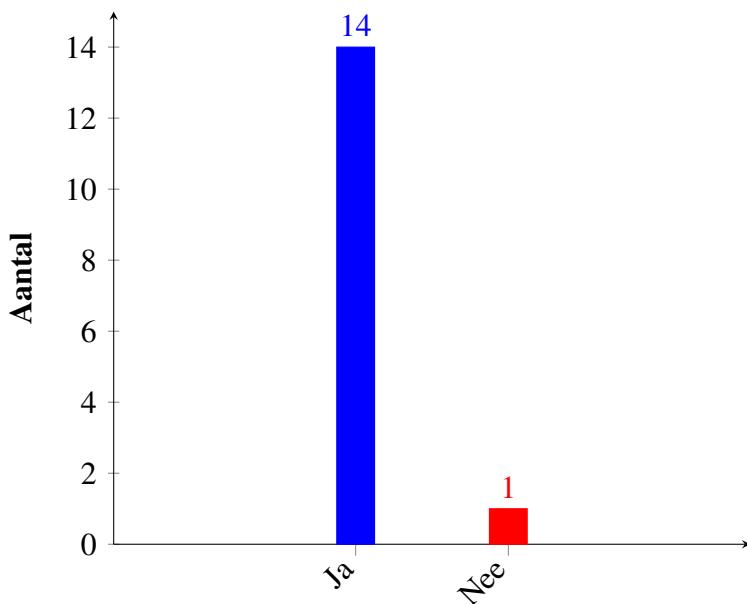
B. Bijlagen

B.1 Resultaten React 360 onderzoek

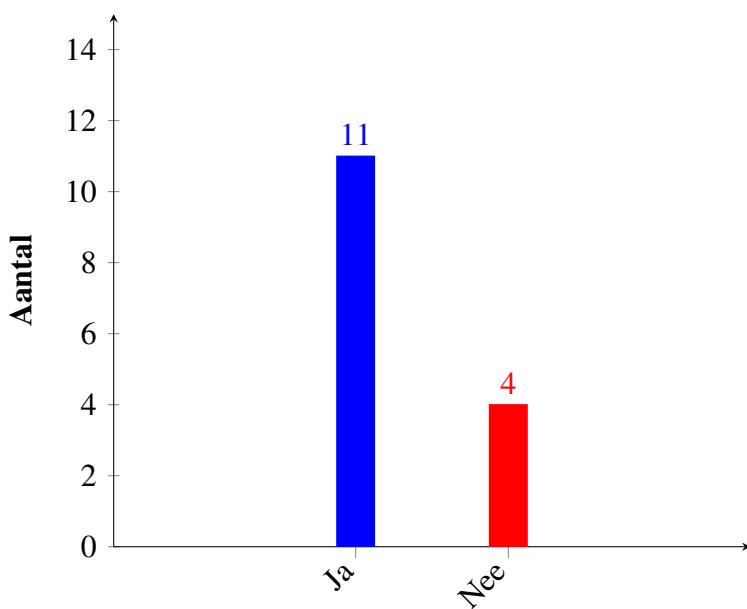
Voelde u zich aanwezig op de plaats zelf?

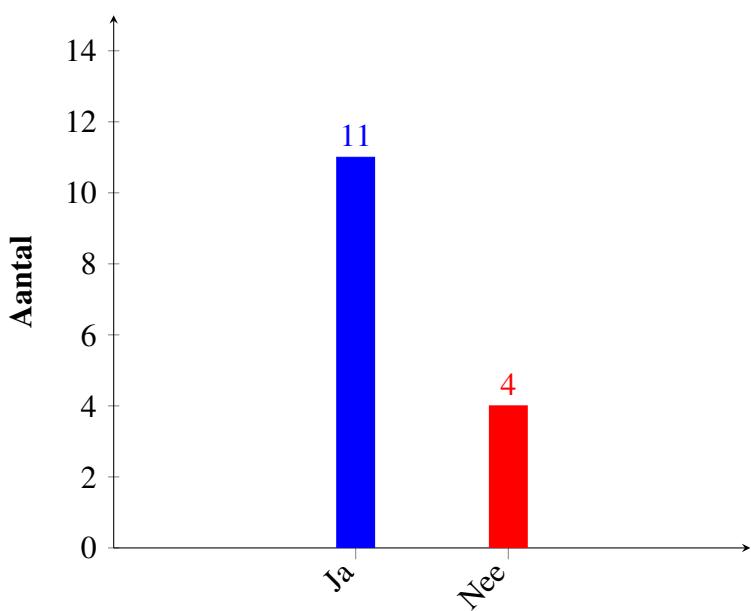


Vond u de applicatie aangenaam om te gebruiken?

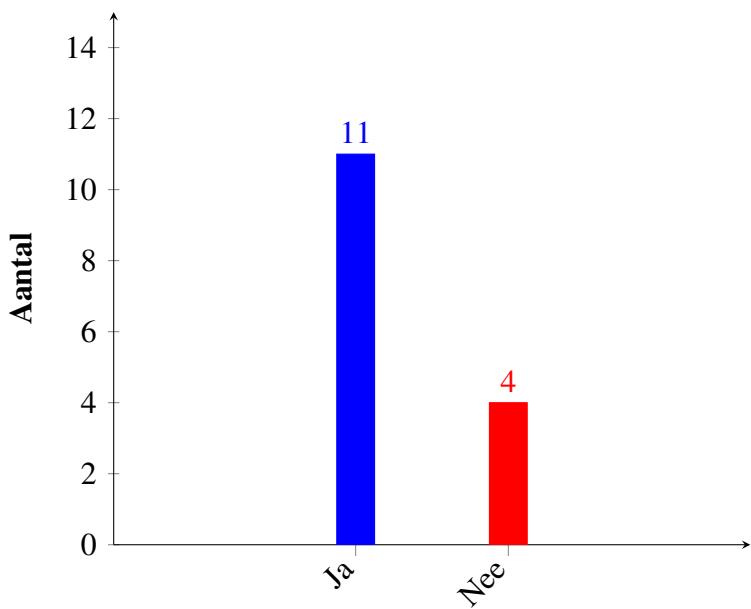


Heeft u al eerder gebruik gemaakt van virtual reality?

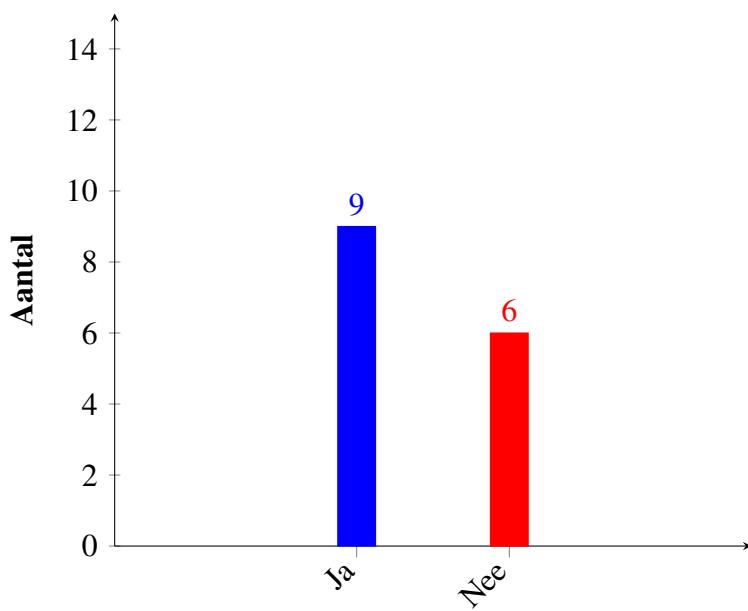


Vond u het beeld goed van kwaliteit?

Had u last van fysieke ongemakken? (hoofdpijn, duizeligheid, droge ogen, misselijkheid, ...)



Moest u zelf beschikken over een VR headset, zou u dan regelmatig deze soort applicaties gebruiken?



Bibliografie

- Abarrera. (2017, november 20). Virtual Reality isn't functional yet, here is why. Verkregen van <https://thealeph.com/articles/2017/11/virtual-reality-vr-oculus-htc-why/>
- Augment. (2015, oktober 6). Virtual Reality vs. Augmented Reality. Verkregen van <http://www.augment.com/blog/virtual-reality-vs-augmented-reality/>
- Azuma, R. T. (1997). *A Survey of Augmented Reality*. Hughes Research Laboratories.
- Bieronski, M. (2016, december 22). How does React VR work? Verkregen van <https://medium.com/@mike.bieronski/how-does-react-vr-work-cf86cd5568a1>
- Chima, S. (2017, oktober 21). Understanding State in React Components. Verkregen van https://dev.to/sarah_chima/understanding-state-in-react-components-1f1
- Cronin, B. (2015, januari 19). The hierarchy of needs in virtual reality development. Verkregen van <https://medium.com/@beaucronin/the-hierarchy-of-needs-in-virtual-reality-development-4333a4833acc>
- Facebook. (2017). React VR: Getting started. Verkregen van <https://facebook.github.io/react-vr/docs/getting-started.html#content>
- Korotya, E. (2017, januari 19). 5 Best JavaScript Frameworks in 2017. Verkregen van <https://hackernoon.com/5-best-javascript-frameworks-in-2017-7a63b3870282>
- Lab, U. (2017, mei 29). 5 Web VR Frameworks to Help Developers Build Interesting Design. Verkregen van <https://uiux.cc/blog/5-web-vr-frameworks-to-help-developers-build-interesting-design/>
- Lehr, M. (2017, maart 16). Goodbye Flatland! An Introduction to ReactVR. Verkregen van <https://www.slideshare.net/geildanke/goodbye-flatland-an-introduction-to-react-vr-and-what-it-means-for-web-developers>
- Mangialardi, M. (2017). *Learn React VR*.
- Mullis, A. (2016, juli 15). How does virtual reality work? Verkregen van <https://www.androidauthority.com/virtual-reality-work-702049/>

- Pappas, S. (2016, april 20). Why Does Virtual Reality Make Some People Sick? Verkregen van <https://www.livescience.com/54478-why-vr-makes-you-sick.html>
- Peniche, M. (2016, oktober 24). Depth in VR: How Three Dimensions Make Virtual Reality Feel Real. Verkregen van <https://www.loading-human.com/depth-vr-three-dimensions-make-virtual-reality-feel-real/>
- Rouse, M. (2011). Stereoscopy. Verkregen van <https://whatis.techtarget.com/definition/stereoscopy-stereoscopic-imaging>
- Sherman, W. (2000). *Understanding Virtual Reality: Interface, Application, and Design.*