

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 het framework hernoemd naar React 360. Dit was vooral een naamswijziging 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 kanten.

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 zijn 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° 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 eenvoudige maar aangename virtual reality applicaties te creëren.

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

We gaan in eerste instantie een stand van zaken opstellen 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 kunnen ingaan. Een VR applicatie verschilt namelijk veel van een gewone web applicatie. We bekijken in deze stand van zaken ook de programmeertalen waarmee het framework is opgebouwd. Tot slot bekijken we enkele mogelijke alternatieven voor React 360.

Hierna beginnen we aan het echte onderzoek. In het 1ste deel van het onderzoek bekijken we kort de belangrijkste componenten van het React JS framework. React 360 is hier op gebaseerd en dus is dit essentieel om het te kunnen verstaan. Hierna gaan we het React 360 framework gaan ontleden. We overlopen alle concepten en geven hier een uitgebreide

uitleg over.

Nadat we het framework kunnen begrijpen, gaan we over tot het 2de deel van het onderzoek. Hiervoor werden 2 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 geven we de resultaten weer van dit onderzoek en maken we hier een besluit van op.

Uiteindelijk kunnen we dan een conclusie opmaken die een antwoord formuleert op al onze 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 VR frameworks naast React 360	32
2.5.1	A-Frame	32
2.5.2	Primrose	32
2.5.3	Agron.js	32
3	Methodologie	33
3.1	Het framework, React 360, bestuderen	33
3.2	Ervaring van virtual reality voor een gebruiker	33
3.3	Conclusie: De doeleinden en stand van zaken van React 360	34
4	React 360	35
4.1	React.js	35
4.1.1	JSX	36
4.1.2	Components en props	36
4.1.3	State en levenscyclus	38
4.1.4	Events	39
4.2	Werking van React 360	39

4.3	Enkele belangrijke componenten	41
4.3.1	View	41
4.3.2	Text	41
4.3.3	Image	42
4.4	Concepten	42
4.4.1	360°foto's en video's	42
4.4.2	Het ruimtelijke systeem	44
4.4.3	Geluid	46
4.4.4	Input	47
5	React 360 applicaties testen	49
5.1	Applicatie 1: Omgeving simuleren	49
5.2	Applicatie 2: Een interactieve applicatie	50
5.3	Prestaties	51
5.4	Resultaten bevraging	51
6	Conclusie	55
A	Onderzoeksvoorstel	57
A.1	Introductie	57
A.2	State-of-the-art	57
A.3	Methodologie	58
A.4	Verwachte resultaten	58
A.5	Verwachte conclusies	58

B	Bijlagen	59
B.1	Resultaten React 360 onderzoek	59
	Bibliografie	63

Lijst van figuren

2.1 Het verschil tussen virtual reality en augmented reality.	22
2.2 Stereoscopie voorbeeld.	23
2.3 De hiërarchie van de benodigdheden in virtual reality.	26
4.1 De opbouw van componenten en sub-componenten in React.	36
4.2 De software stack van React 360 (React VR).	40
4.3 Een equidistante cilinderprojectie.	43
4.4 Een flexbox container met flex items.	45
4.5 De X, Y en Z assen.	45
4.6 Een cursor, raycaster en controller	48
5.1 Applicatie 1	50
5.2 Applicatie 2	50
5.3 Een momentopname van het geheugen verbruik van applicatie 1 (links) en applicatie 2 (rechts)	53
5.4 Het batterijverbruik van de 1ste applicatie na 1u en 20 minuten.	54

Lijst van code fragmenten

2.1	HTML Element voorbeeld	27
2.2	HTML Element voorbeeld	27
2.3	Voorbeeld van een HTML bestand	27
2.4	Voorbeeld van een CSS bestand	28
2.5	Hond object aanmaken	29
2.6	Eigenschap toevoegen aan en 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 2 getallen optelt	30
2.11	Een functie de som functie oproept	31
2.12	Voorbeeld van primrose omgeving	32
4.1	Voorbeeld van een JSX tag	36
4.2	Het component HelloWorld wordt gedefinieerd	37

4.3	Het component Hello wordt gedefinieerd die de property 'name' gebruikt . . .	37
4.4	Het component Hello met de property name	37
4.5	Het component Hello die meerdere malen met andere props wordt gebruikt. .	37
4.6	Het temperature component met de bijhorende state.	38
4.7	Een event afhandelen	39
4.8	Een component die een View teruggeeft.	41
4.9	Het Text component met kinderen.	41
4.10	Het Image component met als bron een-afbeelding.jpg.	42
4.11	Een afbeelding als achtergrond instellen in de runtime.	43
4.12	Voorbeeld van een surface	44
4.13	Voorbeeld van een surface	45
4.14	Voorbeeld van een Entity component	46
4.15	Voorbeeld van een Entity component	46

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 echt bevindt in die omgeving. Men gaat letterlijk op de menselijke zintuigen gaan inspelen door middel van elektronica om dus de gebruiker een gevoel van realiteit te geven die er eigenlijk niet echt 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 gyrocoop die ook alle bewegingen met het hoofd opvolgt en weergeeft in de virtuele wereld.

De toepassingen waar dat virtual reality kan gebruikt worden zijn zeer uitgebreid. De bekendste hiervoor is entertainment, zoals gaming. Hiermee kan men dus gaan simuleren alsof de persoon zich in het spel bevindt. Daarnaast kan het ook gebruikt worden in het medische gebied. Men kan VR gaan gebruiken als behandeling tegen bepaalde aandoeningen zoals bepaalde fobieën en posttraumatische-stressstoornis. Daarnaast kan aan de hand van virtual reality bepaalde ingrepen 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, astronaut, vliegsimulators, ...

Virtual reality heeft dus zeker een ruim aanbod van toepassingsgebieden waarvoor het gebruikt kan worden, maar toch is het ontwikkelen van virtual reality applicaties een grote uitdaging. Er is dus 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 (wat dan weer een javascript framework is, 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. Hierdoor is het nog moeilijk om toe te treden tot deze markt aangezien de prijzen van deze virtual reality brillen soms hoog kunnen oplopen. Dit is momenteel nog een struikelblok voor de consument om de aankoop van een VR headset uit te stellen, maar zeker niet de enigste. In het artikel van (Abarrera, 2017) wordt goed aangehaald wat precies de redenen zijn waarom VR headsets nog niet volledig aangeslaan 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 zeker niet hetzelfde als een gewone webapplicatie. Er moet ten eerste al 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 lijden 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 een virtual reality applicatie gaan 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 app kan worden ontwikkeld, maar daarnaast wordt er bij React 360 ook gebruik gemaakt van React JS. React JS is al een bekend framework voor het ontwikkelen van interactieve webapplicaties. Dit zorgt er dus voor dat de kloof tussen het ontwikkelen van een webapplicatie en een virtual reality applicatie pakken kleiner wordt. Ook is het zo dat men voor deze apps uit te voeren niet verplicht een virtual reality headset hoeft te gebruiken aangezien deze applicaties in een web browser kunnen worden uitgevoerd 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, een pak goedkoper. Maar de vraag is natuurlijk 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. Is er dan ook nood aan goede en performante hardware? Een computer moet namelijk veel meer berekeningen doen voor een 3D omgeving te gaan creëren.

1.2 Onderzoeksraag

De 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 zoeken we een antwoord op al onze onderzoeks vragen en vooral of React 360 een goed framework is voor virtual reality applicaties en voor welke doeleinden. Het antwoord hoeft dus niet noodzakelijk positief te zijn, het belangrijkste is om een duidelijk antwoord te kunnen formuleren. We willen vooral het framework gaan ontleden en alle belangrijkste aspecten dat het aanbiedt gaan bekijken terwijl we ook rekening houden met alle concepten van de wereld van virtual reality. We moeten ook vooral in gedachten houden dat virtual reality een nieuwe technologie is. Het concept erachter bestaat al lang maar het is pas sinds de laatste jaren dat het zijn intreden in de markt heeft gemaakt.

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 wordt de resultaten van het onderzoek gegeven. Dit onderzoek bestond uit 2 applicaties die getest werden door een 15-tal personen. Daarnaast wordt ook kort weergegeven hoe een React 360 applicatie presteert.

In Hoofdstuk 6 wordt tenslotte de conclusie gegeven en een antwoord geformuleerd op de onderzoeks vragen. Daarbij wordt ook een aanzet gegeven voor toekomstig onderzoek binnen dit domein.

2. Stand van zaken

Voordat React 360 kan worden bekeken is het belangrijk een goed beeld te scheppen van de technologie virtual reality zelf. In dit hoofdstuk gaan we dieper in op de concepten van virtual reality, hoe het werkt en wat de gevolgen zijn voor de gebruiker van deze applicaties. Daarna bekijken we de programmeertalen waarmee het framework werkt: HTML, CSS en Javascript. We zullen tot slot nog een overzicht geven van de mogelijke alternatieven voor React 360 met ook 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, maar 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 4 sleutel elementen van virtual reality die (Sherman, 2000) in zijn boek aanhaalt.

Een virtuele wereld

In ons hoofd kunnen we ons dingen voorstellen die enkel wij ons kunnen voorstellen. Dit kunnen bepaalde omgevingen zijn die niet bestaan, fictie zijn. Dit noemt 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 men bij een film een script heeft die de film verteld (maar het is natuurlijk nog niet de afgewerkte film).

Het eerste sleutel element is zeer duidelijk en vloeit natuurlijk rechtstreeks van de naam virtual reality zelf. Met een virtuele wereld wordt niet noodzakelijk bedoelt dat men iets gaat gaan maken dat niet bestaat, maar eerder iets gaan bedenken dat niet binnen handbereik is. Denk maar aan de Chinese muur. Dit bestaat natuurlijk wel echt, maar China is niet een land dat dicht bij de deur is. Door virtual reality kunnen we toch de Chinese muur proberen te ervaren zoals hij in het echt 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 zijn. Wat ook belangrijk is, is dat men met 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 heb je zelf de keuze wat je doet. Er is dus een communicatie aanwezig in 2 richtingen. De gebruiker reageert op de virtuele wereld maar de virtuele wereld reageert ook terug op de gebruiker. Bij een boek is dit natuurlijk niet mogelijk. De onderdompeling kan onderscheiden worden in 2 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 eigenlijk in de virtuele wereld terechtkomt. Hierbij gaat men vooral op de menselijke zintuigen en bewegingen gaan inspelen.

Feedback voor de zintuigen

Om dus de fysieke onderdompeling te ondersteunen, is er nood aan feedback voor de zintuigen. Denk maar aan een droom, waarbij het ook allemaal niet echt is. Toch is het mogelijk om in een droom bepaalde gevoelens of zelfs pijn te voelen. Ook bij virtual reality moeten we het lichaam zoveel mogelijk doen denken 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 3 voorgaande elementen samen. Zoals eerder vermeld, heeft virtual reality nood aan communicatie die in 2 richtingen gaat. We willen interacties kunnen uitvoeren op de virtuele wereld en deze ook te zien krijgen. Net zoals we bijvoorbeeld bij videospelletje op een gewoon beeldscherm bepaalde knoppen kunnen indrukken om zo een respons te zien 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 goed aangehaald.

2.1.2 3D en 4D

3D wil letterlijk zeggen driedimensionaal. Hiermee bedoelt men dat iets 3 meetkundige dimensies heeft, namelijk diepte, breedte en hoogte. Deze technologie kan men dan gaan toepassen op beeldschermen a.d.h.v. stereoscopie, waar we in hoofdstuk 2.2 wat meer uitleg over geven. Op deze manier krijgt de gebruiker dus een illusie van diepte. Meestal worden hier dan speciale brillen voor gebruikt, ook wel 3D-brillen genoemd. De klassiekere brillen hebben dan een kleurfilter, meestal rood en blauw. Rood laat enkel rode kleuren door en blauw enkel blauwe kleuren. Hiermee kan men dan vanaf 2 perspectieven een blauwe afbeelding en een rode afbeelding gaan weergeven waardoor de diepte van een object dus 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 is en afhankelijk van de content is er ook weinig feedback voor de zintuigen. Er is wel een extra dimensie maar er kan niet rondgekeken worden in tegenstelling tot 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 dus zeker geen volwaardige virtual reality, maar is wel al een stap in de goede richting (Peniche, 2016).

2.1.3 Augmented Reality

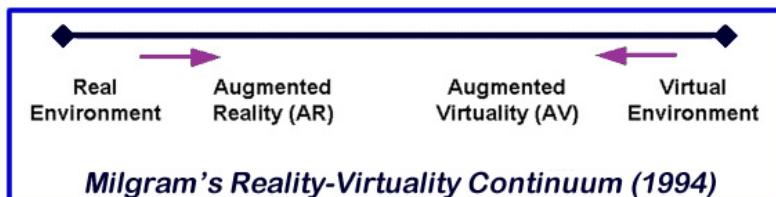
Augmented Reality wordt heel veel verward met virtual reality maar zijn toch 2 andere technologieën. Je kan augmented reality gaan vertalen naar 'toegevoegde realiteit'. In tegenstelling tot virtual reality gaat men bij augmented reality letterlijk iets gaan toevoegen aan wat al bestaat. Bij virtual reality gaat men dan iets volledig virtueel gaan scheppen. Aangezien augmented reality iets toevoegt, is er ook niet verplicht nood aan een bril. Vandaag de dag wordt augmented reality regelmatig gebruikt bij de smartphone. Denk maar aan een applicatie waarbij je een object kan afbeelden op een plaats. Bijvoorbeeld

een bepaald meubilair die virtueel gegeneerd en getoond wordt in een kamer dat echt bestaat.

Er is niet echt een definitie van wat augmented reality precies inhoud. Je kan eigenlijk al een scorebord bij een live uitzending van sport op de televisie zien als een vorm van augmented reality. Maar volgens (Azuma, 1997) zou iets aan volgende 3 karakteristieken moeten voldoen om augmented reality te kunnen zijn:

- Combineert de echte wereld met de virtuele wereld
- Men kan interacties doen met het virtuele
- De virtuele wereld bezit de 3 dimensies (diepte, breedte en hoogt)

Als we deze 3 karakteristieken toepassen op het voorbeeld van het scorebord, zien we dus dat een scorebord bij een live uitzending van sport geen augmented reality is. De 1ste voorwaarde wordt wel voldaan, maar de 2de en de 3de niet. Men kan geen interacties gaan uitvoeren op dit scorebord, de programmamaker zou dit wel kunnen dus voor hem is dan enkel de 3de regel niet voldaan. Zou het beeld voor de programmamaker dan ook nog in 3D (een 3D scorebord dus, niet noodzakelijk een 3D beeldscherm) worden weergegeven, dan kunnen we stellen dat dit een vorm van augmented reality is.



Figuur 2.1: Het verschil tussen virtual reality en augmented reality.

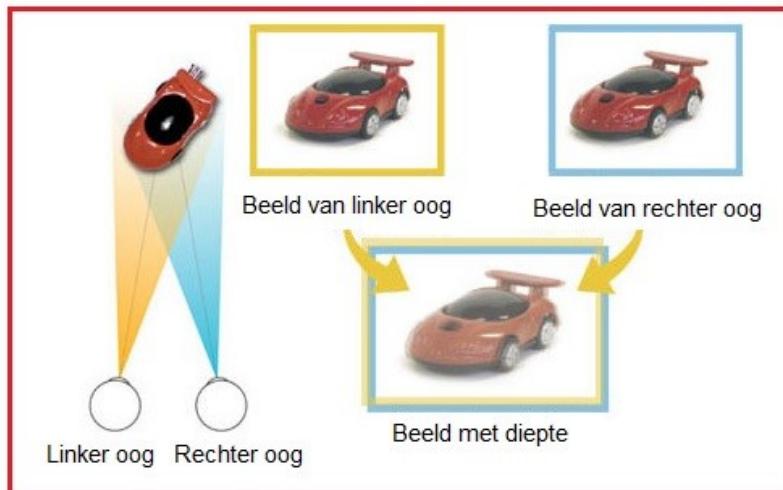
2.2 De werking van virtual reality

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 technische gedeelte van virtual reality.

2.2.1 Stereoscopie

Bij virtuele werkelijkheid wordt er gebruik gemaakt van een illusie. Dit doet men ten eerste a.d.h.v. stereoscopie. Hierbij gaat men diepte meegeven aan een afbeelding. Dit doet men door 2 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 dus diepte gaan simuleren. Zo kan er beter worden ingeschat hoe ver en hoe groot een object is in de virtuele wereld. Men kan dan deze 2 afbeeldingen



Figuur 2.2: Stereoskopie voorbeeld.

gaan tonen door middel van een VR headset. In deze bachelorproef ligt de focus vooral op React 360 voor mobile VR apps. We gaan dus gebruiken moeten maken van een mobile VR headset. 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 2 beelden geprojecteerd. De lenzen zorgen er dan voor dat de omgeving ruimer lijkt dan het werkelijk is. In combinatie met een gyrocoop kan men dan gaan rondkijken en deze wereld terwijl de smartphone al het rekenwerk doet. Hiermee kan er dan op een goedkope manier virtuele realiteit getoond worden. Dit is helaas meestal 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 virtual reality is. Er zijn nog meerdere aspecten waarmee men rekening moet houden om ervoor de 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 2 duurste virtuele headsets die momenteel op de markt zijn en dan vooral naar gaming gericht zijn, 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 maar op 60Hz, 60 beelden per seconde dus. Bij

smartphones is dit meestal ook 60Hz maar soms kan dit nog lager liggen, tot 30Hz zelfs. Het is dus 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 je een stap vooruit zet. Er zal dan een bepaalde hoeveelheid tijd zijn tot je 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 onderscheidt 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 pijn leiden zoals motion sickness (Pappas, 2016), iets dat ook voorkomt bij onze moderne manieren van transport bij bijvoorbeeld autorijden (wagenziekte). Het is dus 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 in de virtuele wereld ook zeer belangrijk. Een mens ziet ongeveer 180° rond zich, maar dit kan oplopen tot 270° als er met de ogen bewogen wordt. De meeste headsets hebben maar een gezichtsveld tussen de 90° en 110°, wat dus eigenlijk niet genoeg is. Dit heeft dan ook een impact op de virtuele ervaring en kan ook hier opnieuw leiden tot motion sickness.

Als er dus 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 dus zeer belangrijk voor ontwikkelaars van virtuele applicaties om hiermee rekening te houden. Als een applicatie je ziek maakt, dan ga je het automatisch ook minder of zelfs niet meer gebruiken.

2.2.3 Interactie met de virtuele wereld

We 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 sowieso goed doet, is rond je kunnen kijken in de virtuele wereld aan de hand van een gyrocoop. Maar wat dan met rondlopen in die virtuele wereld?

Men kan hiervoor simpele gamecontrollers met een joystick gaan gebruiken waarbij men dus gewoon in de virtuele wereld gaat bewegen als hoe men dit zou doen in een videospel. 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 men zeer accuraat de bewegingen van de handen gaan volgen waarbij men ook gebruik maakt van

een gyroscoop. Voor de voeten bestaan er ook al oplossingen, maar deze zijn natuurlijk minder praktisch. Er zijn al zogenaamde loopbanden die ervoor zorgen dat men 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 vooral gekozen voor een simpele joystick, meestal in combinatie met een motion controller.

Voor mobile is het natuurlijk wat moeilijker om een gamecontroller te gebruiken, niet iedereen beschikt hierover. Daarom wordt er bij mobile VR de zogenaamde 'gaze' techniek gebruikt. Gaze is zoals de vertaling zegt, staren. In plaats van dat je dus op een knop gaat drukken is het de bedoeling dat je bepaalde knoppen in de virtuele wereld gaat aanstaren voor een korte tijd. Dit is een zeer handige oplossing die gemakkelijk kan toegepast worden om het doelpubliek voor een VR applicatie groter te maken. Maar helaas zorgt 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 ?? hebben aangehaald kan een slechte opbouw van een virtuele applicatie lijden tot fysieke ongemakken bij de gebruiker. Hier hebben we het dan eerder over onschuldige kwaaltjes 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 omdat van deze fysieke gevolgen.

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 onze zintuigen. De manier waarop onze 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.

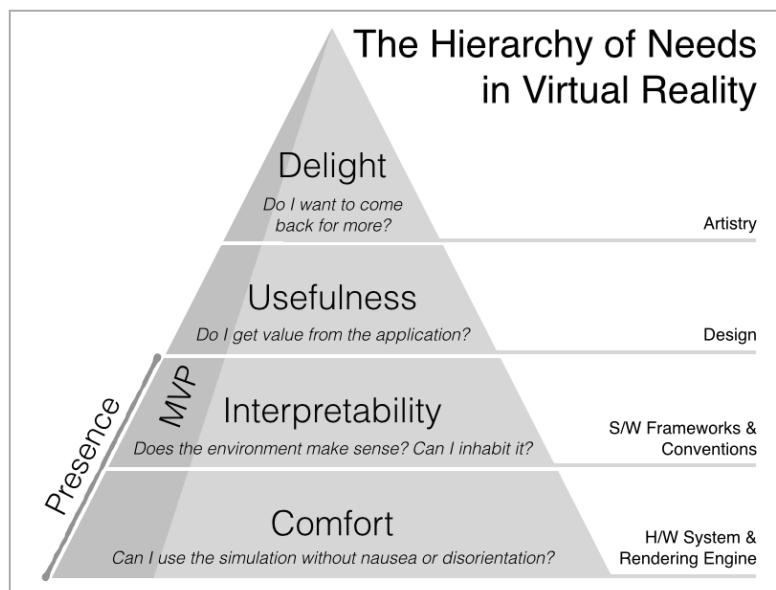
Dan komt de **Interpreteerbaarheid**. Hoe realistisch is de virtuele wereld nog? Hier wordt eerder het verschil tussen wat nog non-fictie is en wat fictie is bedoeld. Een ervaring waarbij de gebruiker door het heelal wordt gezogen op een enorme snelheid met beelden die hij/zij nog nooit eerder gezien heeft zal dus zeer overweldigend zijn. De ontwikkelaar

kan wel enkele regels van de fysica gaan aanpassen, maar het blijft belangrijk dat deze ervaringen uitbreidingen vormen op het echte leven. Een VR applicatie moet natuurlijk nog steeds iets onbestaand, virtueel bezitten. Anders zou een virtuele applicatie ontwikkelen weinig nut hebben.

Vervolgens heb je de **Bruikbaarheid**. Dit hangt af van applicatie tot applicatie, maar hier gaat het dus 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 tot denken om het huis te kopen? Het gaat hier dus vooral over de design van de applicatie.

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

Het is duidelijk dat de 2 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 2.3: 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 maar kan ook zeker 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.

Denk maar aan de animatie bij een uitschuifbaar menu of tekst die verandert als je op een knop klikt.

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 dan naar een source, een bron. Deze bron is dan de locatie van een afbeelding. 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 <tag attribute1="value1" attribute2="value2" />.
```

Code fragment 2.2: HTML Element voorbeeld

Men kan deze HTML elementen ook in een ander HTML element gaan stoppen. Dit wordt ook wel nesting genoemd. De HTML elementen die zich dan binnen in een ander element bevinden noemen we 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

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 voor alle HTML elementen een stijl

kan worden gegeven. Op deze manier kan men dus een pagina vormgeving geven dat veel aantrekkelijker is voor het oog. De manier waarop CSS een bepaald HTML element aanspreekt is door middel van een klasse. Door deze klasse kan men het uitzicht van een specifiek HTML element gaan aanpassen. Het is ook bijvoorbeeld mogelijk om voor alle `<h1>` elementen een globale stijl te voorzien. Je 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 zeer simpel 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 vorige voorbeeld hebben we ervoor gezorgd dat alle `<h1>` elementen een groote zullen hebben van 36pt (point-size) en de kleur rood zal zijn. Daarnaast zullen alle elementen die de klasse `.grootBlauw` bevatten een tekst grootte hebben van 20pt en een blauwe kleur hebben.

2.4.3 Javascript

Javascript is de meest geavanceerde van de 3 en ook het belangrijkste voor deze bachelorproef. Met javascript kan je dus een pagina gemaakt met HTML interactief gaan maken en is daarom 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 even de belangrijkste onderdelen die javascript te bieden heeft.

Objecten

Zoals eerder al vermeld is javascript een object georiënteerde taal. Dit wil zeggen dat de taal met zogenaamde objecten werkt. Deze objecten kan je vergelijkingen met objecten in 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 je kan natuurlijk ook je eigen objecten gaan maken.

Laten we dit beter tonen a.d.h.v. een voorbeeldje. 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 dit stukje code hebben we dus een hond object aangemaakt met de eigenschappen: naam, leeftijd, ras en kleur. Hier kan de flexibiliteit van javascript bewezen worden. Bij andere object georiënteerde talen zou het moeilijker zijn om nu een extra eigenschap aan het hond object toe te voegen, er zouden al snel andere programmeer technieken moeten gebruikt worden zoals overerving. In javascript kunnen we bijvoorbeeld de eigenschap oogkleur zeer gemakkelijk toevoegen als volgt:

```
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 goed voorbeeld van een variabele. Variabelen zijn vergelijkbaar met de X en de Y die we uit de wiskunde kennen, ze houden een bepaalde waarde bij. Een variabele kan je maken door eerst aan te geven welke soort variabele je wilt maken. In vele andere talen moet je voor een nummer een andere soort gebruiken dan als je een stukje tekst in een variabele wilt stoppen. Ook hier zien we opnieuw de flexibiliteit die javascript biedt heeft. Javascript heeft 3 verschillende soorten declaraties voor een variabele:

- **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 dus 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

Zoals je kan zien, is een let een lokale declaratie of ook wel block-scoped, binnen de accolades, 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 dus achteraf niet meer worden gewijzigd.

Indien een variabele geen waarde krijgt, zoals dus mogelijk is bij een var, dan zal dit de waarde undefined 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 kan je deze functies ook zelf gaan schrijven. Dit is een voorbeeld van een simpele functie die 2 nummers optelt:

```

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

Code fragment 2.10: Een functie die 2 getallen optelt

Het voorbeeld toont duidelijk aan dat een functie zeer eenvoudig opgebouwd is. Men kan aanhalen dat iets een functie is door `function` te gebruiken. Daarna geeft men een naam voor de functie, in dit geval is dat 'som'. Dan gaat men aan deze functie 2 parameters

meegeven, nummer1 en nummer2. Uiteindelijk kan men de functie gaan uitvoeren en het resultaat van de optelling teruggeven. Om dan gebruik te gaan maken van deze functie, kan men de functie gaan oproepen in andere functies door '(' en ')' 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 de som functie oproept

Deze 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 binnin je code. De console kan je zien als de uitlaat 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 geavanceerde aspect van javascript, maar wel een zeer groot pluspunt van de taal, is dat er zeer gemakkelijk 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 gaan ontvangen. We gaan hier in deze bachelorproef niet verder op in.

Frameworks

Waarschijnlijk de allergrootste reden dat javascript zoveel gebruikt wordt is het grote aanbod aan frameworks. Een framework is eigenlijk een grote verzameling van software componenten 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 gebruiken 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 VR frameworks naast React 360

Het ontwikkelen van virtual reality applicaties gaat natuurlijk niet zomaar en vereisen een goede technische kennis. Zeker het feit dat je in een wereld zit waar er volledig om zich kan worden gekeken en ook eventueel in bewogen kan worden is natuurlijk een grote uitdaging. Om dit allemaal makkelijker te maken zijn er momenteel al een aantal 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 virtual reality in de browser te tonen aan de buitenwereld door A-Frame. Met A-Frame kunnen ontwikkelaars 3D werelden gaan bouwen die bestemd zijn voor virtual reality. Doordat A-Frame beschikbaar is in de browser door simpelweg 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 en heeft een goede performantie.

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. Dit ziet er in de code zo 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.

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 onderzoeks-project. 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, React 360, bestuderen

We gaan eerst kort React.JS zelf bekijken. Hiermee wordt React 360 namelijk opgebouwd en het is dus cruciaal om hier ook kennis over te hebben. Daarna zullen we de belangrijkste concepten van het React 360 framework gaan bekijken. 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

Daarna kunnen we met al de informatie die we verworven hebben door middel van React 360 enkele eenvoudige applicaties gaan maken. Deze applicaties zullen getest worden door een aantal personen en zij zullen dan hun mening geven a.d.h.v. een korte vragenlijst. Deze zal dan een beter inzicht geven in hoe een persoon een virtual reality applicatie gemaakt met React 360 ervaart.

App 1

De eerste applicatie zal de simpelste zijn. Dit is een applicatie waarbij de nadruk ligt op het simuleren van een echte omgeving. Hiervoor zal een gebied dat echt bestaat in 360° worden weergegeven met het bijhorende omgevingsgeluid. De gebruiker zal dan kunnen ervaren alsof hij echt op die locatie aanwezig is.

App 2

De 2de applicatie zal meer de nadruk leggen op de interactie met de virtuele wereld. Er zal hier dus vooral gekeken worden naar de input mogelijkheden en hoe de gebruiker dit ervaart. Zo zal er een applicatie worden gebouwd met verscheidende knoppen waarop de gebruiker kan drukken a.d.h.v. een knop de virtuele headset. Er zullen dan bepaalde elementen in de gebruikersomgeving zich gaan aanpassen afhankelijk van wat de gebruiker geselecteerd heeft.

Prestaties

We gaan ook tijdens testen van deze applicaties letten op hoe de applicaties presteren. Hoeveel geheugen verbruiken ze? Hoeveel aanslag leggen ze op de batterij? Blokkeert de browser soms?

3.3 Conclusie: De doeleinden en stand van zaken van React 360

Ten slotte zullen we een besluit opmaken van React 360. Hier gaan we dan een stand van zaken opstellen hoe React 360 scoort als framework voor virtual reality. Zo zullen we antwoord formuleren op alle onderzoeks vragen.

4. React 360

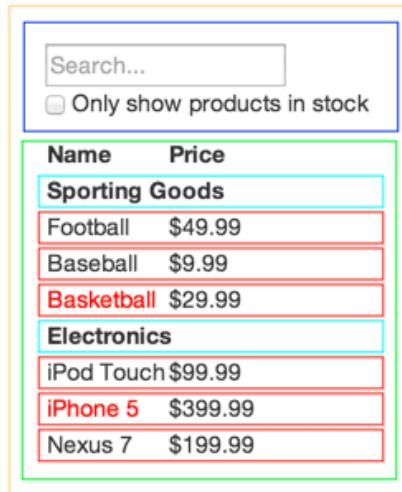
React 360 is een framework voor het ontwikkelen van VR en 360°applicaties in javascript dat ontwikkeld is door facebook. React 360 wordt gebruikt om gemakkelijk een complexe gebruikersomgeving te kunnen maken in een 360°omgeving. Het grote voordeel aan React 360 is dat het gewoon in de browser kan worden uitgevoerd. Dit wil dus zeggen dat gebruikers met een recente versie van hun browser React 360 applicaties zonder problemen 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 pc, smartphone als op andere VR apparaten (Lehr, 2017).

In React 360 gaat men eigenlijk 2D met 3D gaan combineren. Men gaat 2D interfaces in een 3D omgeving gaan plaatsen maar er is ook de mogelijkheid om gebruik te maken van 3D objecten die in de wereld kunnen geplaatst worden. Enkele voorbeelden waar React 360 kan gebruikt voor worden is foto's, videos, 360°rondleidingen en zelfs simpele 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 dus een pak eenvoudiger. Bepaalde concepten van React Native komen ook terug in React 360. React Native is een framework (ook door facebook) dat ook gebaseerd is op React JS. Met React Native kunnen mobiele applicaties worden ontwikkeld voor zowel iOS als voor Android.

4.1 React.js

React JS is een framework dat gemaakt is om user interfaces (gebruikersomgevingen) te gaan ontwikkelen. Hiermee kan er zeer gemakkelijk een complexe user interface worden



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

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. JSX is eigenlijk iets dat 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:

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

Code fragment 4.1: Voorbeeld van een JSX tag

Je kan dus eigenlijk HTML gaan gebruiken in javascript code. Dit maakt het een pak gemakkelijker 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 dus een user interface worden opgebouwd. Deze components bestaan dan op zichzelf uit een verzameling van JSX tags. Een user interface is meestal een soort van boom die opgebouwd is uit andere kleine onderdelen. Hier schijnt React dus in uit. Je kan dus aan deze componenten, sub-componenten gaan toevoegen en zo een duidelijke opbouw en structuur hebben.

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 is zeer eenvoudig gaan 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 dus verplicht om de `render()` functie te initialiseren. Deze render functie zal dan JSX teruggeven.

Het is ook mogelijk om net zoals bij een gewoon HTML element attributen mee te geven aan een component. In React noemt men dit dan properties of kort, props. Met deze props is het mogelijk om ervoor te zorgen dat componenten dynamisch zijn en dus niet enkel hetzelfde moeten teruggeven. De props van een component kan je 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 die de property 'name' gebruikt

Om dan de property te kunnen gebruiken, kunnen we 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

Op de webpagina zouden we dan het resultaat `Hello Arne!` terugkrijgen. Hier tonen we dus de herbruikbaarheid aan de React componenten. Als we dit meerdere malen met andere namen zouden willen doen, kunnen we simpelweg de property name aanpassen zonder dat we een volledig nieuw component moeten 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

We hebben dus al een manier om componenten dynamisch te gaan opbouwen met verschillende data. Maar wat als we nu een component `Temperature` zouden hebben die het aantal graden teruggeeft. Props zijn niet meer wijzigbaar 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 en waarvan de eigenschappen naargelang de levenscyclus van het component kan veranderen. Zo is het dus 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 dus 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 dus 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 call naar de functie `setState()` waarmee de waarde van de state eigenschappen kan worden aangepast. Stel dat deze functie wordt opgeroepen elke keer dat de temperatuur zou wijzigen zal hij deze kunnen doorgeven door middel van de parameter `amountCelsius`. Ten slotte wordt dan de `render` functie gebruikt om de eigenschap `degree` van het state object te kunnen tonen.

Elk component heeft een zogenaamde levenscyclus. Deze cyclus start dus 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 de 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 gaan toevoegen die dan verwijst naar de methode die het event afhandelt. Daarnaast kan je eventueel ook parameters meegeven aan deze methode. Hier zien we een voorbeeld duidelijk de werking ervan toont:

```
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 <button onClick={this.handleClick}>Verhoog leeftijd</
19       button>
20   }
}
```

Code fragment 4.7: Een event afhandelen

Bij dit voorbeeld hebben we een button gedefinieerd met de toegevoegde prop onClick . Hierbij verwijzen we dan 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 we op een eenvoudige manier VR applicaties kunnen 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 2 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 onze code geconverteerd wordt naar code die de andere lagen snappen. Daarnaast zorgt het ervoor dat de browser niet vastloopt omdat de code apart wordt uitgevoerd en dan pas aan de 3D omgeving wordt toegevoegd waardoor de beelden per seconde dus 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 in de browser zelf.

Op de 2de laag hebben we OVRUI (Oculus VR User Interface) en Three.js. De OVRUI is een library die ervoor zorgt dat bepaalde elementen gemakkelijk kunnen weergegeven worden via Three.js. Dit zijn onder andere text, knoppen, ... Three.js is dan weer een library die het mogelijk maakt om 3D objecten te kunnen weergeven in de browser.

De laatste laag bestaat dan 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 je gebruikt, zorgt het voor de communicatie met VR toestellen en toont het de 'VR Frame', het stuk op de webpagina dat dus in VR moet worden weergegeven.

We zien dus dat React 360 een verzameling is van meerdere technologieën waar men dan a.d.h.v. React JS gemakkelijk mee kan communiceren. Dit toont dus 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 306 een user interface te kunnen bouwen. De view is een container die ervoor zorgt dat je 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 style={{backgroundColor: 'blue', height: 150px}}>  
5                 <View style={{backgroundColor: 'red', height: 100px}}+>  
6             </View>  
7         );  
8     }  
9 }
```

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 en de hoogte 150 pixels. Dit is mogelijk door het style attribuut toe te voegen aan het view component. Dit is eigenlijk gewoon CSS maar dan met CamelCasing. In deze view zit dan nog een andere view met een rode achtergrond.

4.3.2 Text

Het Text component is zoals de naam zegt om tekst te gaan teruggeven in React 360. Dit is vergelijkbaar met het `<p>` element van html. Net zoals bij een View kan je hier ook stijl op toepassen, maar dan voor de tekst.

```
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 zegt ook zeer duidelijk wat het doet. Het geeft simpelweg een afbeelding terug en is dus 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 dus 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

4.4.1 360°foto's en video's

Waar React 360 zeer goed in is, is het tonen van 360°foto's en videos. Deze zijn natuurlijk essentieel voor een realistische omgeving te kunnen creëren. Met deze foto's en video's kunnen we dus mensen naar andere bestaande plaatsen in de wereld gaan transporteren en dit in combinatie met een interactieve ervaring. Denk maar aan 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 videos 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, maar dus ook in de wereld van VR.



Figuur 4.3: Een equidistante cilinderprojectie.

Het verschil tussen een mono en stereo afbeelding is zoals er in hoofdstuk 2.2.1 werd gesproken over stereoscopie. Bij mono gaat men dus maar 1 afbeelding gaan tonen vanaf 1 perspectief terwijl men bij stereo 2 afbeeldingen vanaf een verschillend perspectief gaat tonen om dieptegezicht mogelijk te maken. Bij React 360 is dit mogelijk door de afbeelding voor het linkse oog vanboven (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 1 afbeelding te plaatsen. Het is natuurlijk enkel mogelijk om stereoafbeeldingen te bekijken met een VR headset. Indien de gebruiker dus de applicatie zou uitvoeren zonder een VR headset dan kan men enkel hetgeen zien wat men in het linkse oog zou zien.

Panorama foto's video's tonen

Je kan foto's of video's zowel in de runtime (4.2) als in de React zelf gaan tonen. Dit kan je 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 maar er is wel een extra argument nodig, namelijk het formaat. Er zijn 4 mogelijke formaten die je kan opgeven als argument in de code die overeenstemmen met de eerder genoemde formaten:

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

Om video's af te spelen volgen we een gelijkaardig proces. Hier heb je dus ook de keuze

om die in de runtime of in React te gaan doen. Het grote verschil dan bij een afbeelding is dat je hier in 2 delen zal gaan werken. Eerst ga je de videospeler gaan aanmaken die een unieke naam moet krijgen en dan ga je deze videospeler gaan 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 dus niet enkel worden rekening gehouden met hoogte en breedte maar dus ook met diepte. Dit kan al vrij snel ingewikkeld worden. React 360 heeft ervoor gezorgd dat dit een pak makkelijker gaat.

Surfaces

Surfaces (oppervlaktes) bieden de mogelijkheid om 2D user interfaces in een 3D wereld te gaan stoppen. Je kan het gaan vergelijken met een stuk van een webpagina in een 3D wereld te gaan onderbrengen. Dit zorgt ervoor dat het een pak makkelijk is voor een 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 );
```

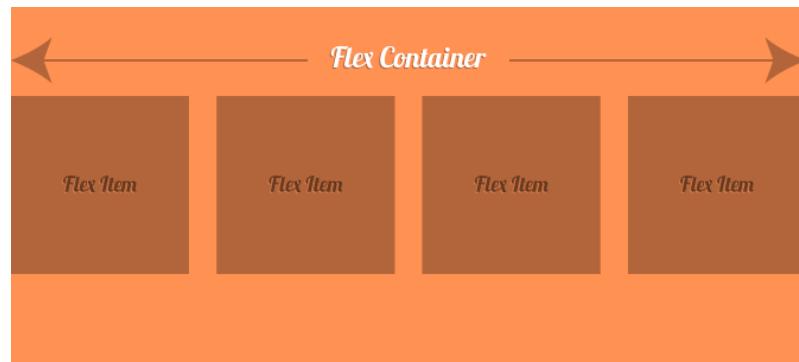
Code fragment 4.12: Voorbeeld van een surface

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

Surfaces zijn zeer flexibel. Je kan ze gemakkelijk gaan bewerken door functies zoals setShape te gaan gebruiken waarbij je de surface van een cylinder naar een vlakke surface kan aanpassen of omgekeerd. Ook de hoogte en breedte van surfaces kan je perfect in React aanpassen. Het is dus ook perfect mogelijk om meerdere surfaces naast elkaar te gaan gebruiken of om eventueel ook meerdere componenten in 1 surface te gaan stoppen.

Aangezien er meerdere componenten kunnen worden gestopt in 1 surface is er ook nood aan een duidelijke manier om deze componenten te ordenen. Dit is net hetzelfde als dat

bij React Native wordt gebruikt namelijk flexbox. Flexbox wordt ook al veel gebruikt in gewone webpagina's en zorgt vooral voor een consistente layout op alle schermgroottes.



Figuur 4.4: Een flexbox container met flex items.

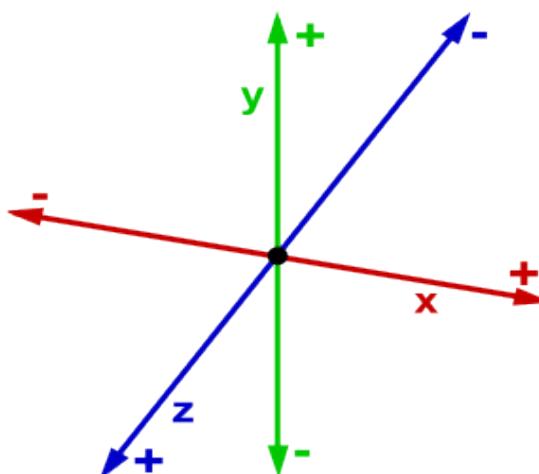
3D-objecten

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 programma van het formaat .obj, .mtl of .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 eigenlijk gewoonweg punten in de omgeving waar je 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+ dus 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 surface



Figuur 4.5: De X, Y en Z assen.

Om 3D objecten van de opgenoemde formaten in onze applicatie te krijgen moeten we gebruik maken van het Entity component. Met dit component kunnen we dan ook de grootte en richting van dit 3D object gaan aanpassen. Dit gebeurt door translate of rotateX/rotateY/rotateZ toe te voegen aan de style van het entity component. Bij rotate worden dan 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 dus voor ons 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. Net zoals bij videos kan je 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 een Entity component

Omdat een virtual reality wereld steeds moet communiceren in 2 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 je doen door `AudioModule.playOneShot` te gebruiken met de verwijzing naar de audio file als parameter. Je kan het dan ook perfect gaan gebruiken bij een functie die wordt opgeroepen door de `onClick` van een button.

Aangezien VR zich in een 3D omgeving bevindt. Is het ook mogelijk om aan een geluid 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 Input

De interactie met de wereld in virtual reality is natuurlijk zeer belangrijk. Maar er zijn ook vele mogelijkheden om input door te sturen naar de virtuele wereld waar we al over gesproken hebben 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.

Button input

Met de titel wordt niet noodzakelijk een `<button>` element als input, maar eerder een fysieke knop als input bedoelt. Een toetsenbord, muis, controller, ... hebben allemaal fysieke knoppen ter beschikking. Als deze knoppen worden ingedrukt verwacht de gebruiker dus 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, maar hierbij wordt dan 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 dus niet uit welk input toestel je gebruikt, als er een naar onder knopactie wordt gedetecteerd zal React 360 de juiste knop en actie terugsturen waardoor het dus 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 eigenlijk niet meer dan een button waarbij er wordt gereageerd op de 'confirm' actie. Deze actie komt regelmatig terug en is zoals de naam zegt om iets te gaan bevestigen. Dit component is dus louter om het gemakkelijker voor de ontwikkelaar te maken. Het enigste verschil is dat hier een `onClick` event handler kan toegevoegd worden.

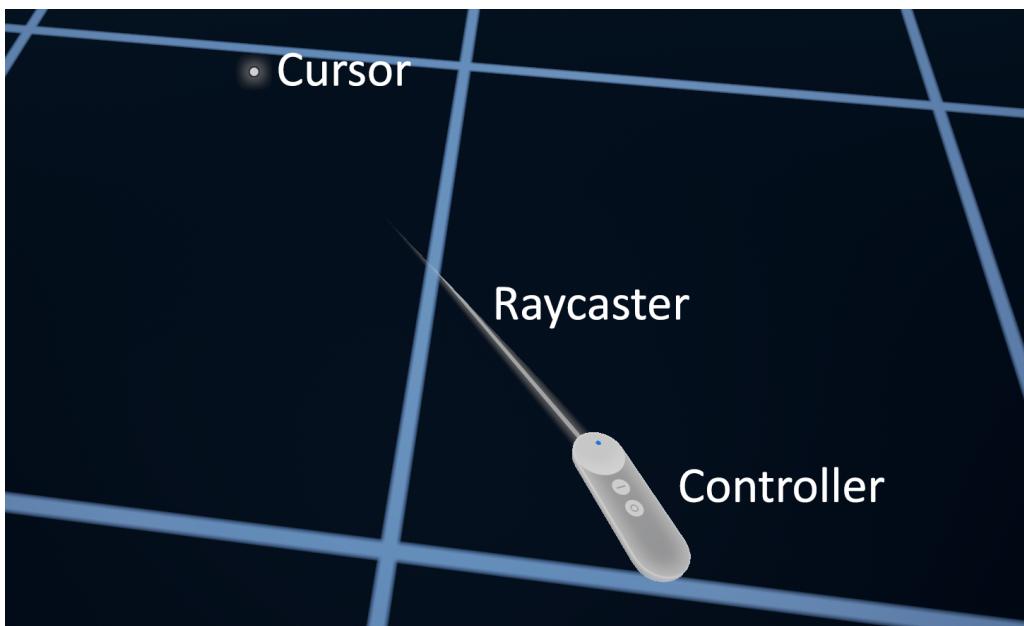
Cursors en raycasters

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. Deze tonen dan een soort van laserstraal in de wereld. De cursor wordt dan getoond op het element waar de laserstraal mee snijdt. Om hiervoor dan 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 dus niet alle controllers ondersteund, is het ook mogelijk voor een

ontwikkelaar om zelfs deze te gaan definiëren. Men noemt dit ook wel een raycaster. Een Raycaster moet een bepaalde interface gaan volgen om als raycaster te worden gezien:

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



Figuur 4.6: 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. 2 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 volgende 2 applicaties. Zij kregen een mobiele VR headset met een smartphone in en mochten dan beide applicaties gedurende 5-10 minuten uittesten. Daarna werd er een bevraging afgenoem die bestond uit een 6-tal 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 1ste 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 is, maar deze soort video's zijn niet gemakkelijk te vinden in goede kwaliteit. 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 2de applicatie was een iets ingewikkeldere. Deze legt vooral de focus op de interactiviteit met de gebruiker. Hier is 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 het offline werkt en volledig compatibel was met de toen huidige versie van chrome op de smartphone.

De applicatie toont aan de linkerkant een overzicht van de mogelijke objecten die kunnen gekozen worden met hun titel en maker. Deze objecten zijn afkomstig van een gratis online database van google genaamd Googly Poly. Men kan 1 van deze objecten links gaan selecteren en dit object zal dan tevoorschijn komen in het midden en traag ronddraaien. Ten slotte kan men dan aan de rechterkant 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 geheugenverbruik vrij hoog. Voor de 1ste applicatie lag het gemiddeld verbruik rond de 580mb. Bij de 2de applicatie ging het nog wat hoger en was het verbruik gemiddeld 900mb. Als we weten dat een gewone webpagina ongeveer gemiddeld 150-200mb verbruikt, is dat een aanzienlijk verschil.

Doordat de applicatie dus een grote hoeveelheid van het RAM geheugen inneemt 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 toch zeker 2GB minimum aan ram geheugen voor een goede virtual reality ervaring te hebben.

5.4 Resultaten bevraging

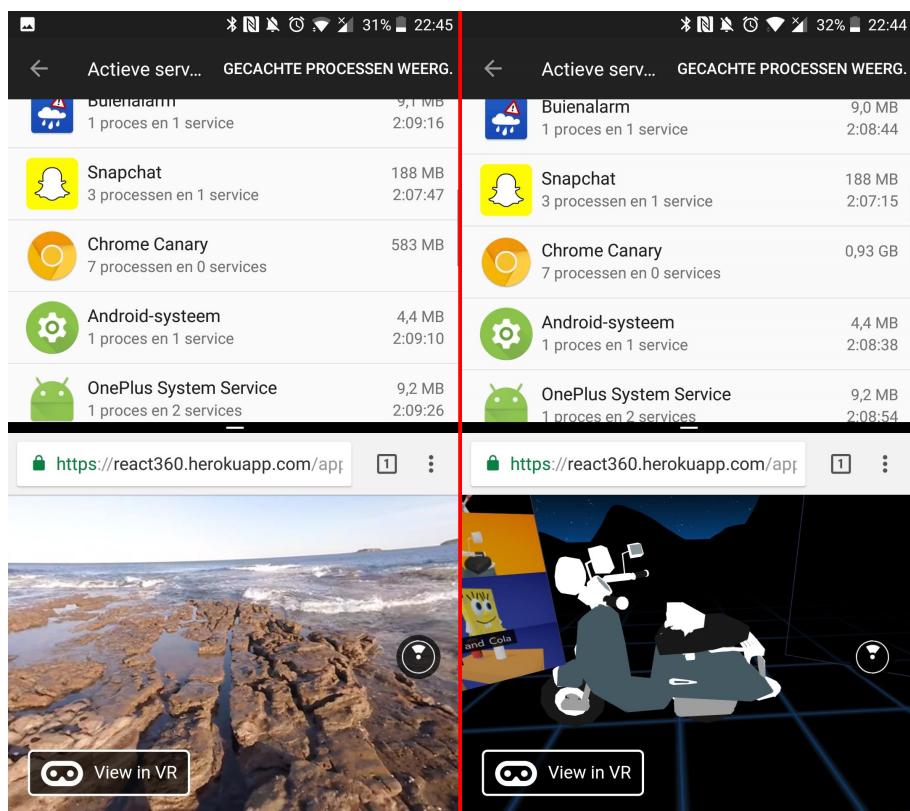
Bij het onderzoek lag de focus vooral op de ervaring van de gebruiker bij de React 360 applicatie. Deze vragen waren dus zeker niet technisch en ook niet over virtual reality in het algemeen. De focus lag dus vooral op het feit of een VR applicatie die gemaakt is met React 360 positief wordt ervaren door de gebruiker. Voor het onderzoek werden volgende

6 vragen gesteld:

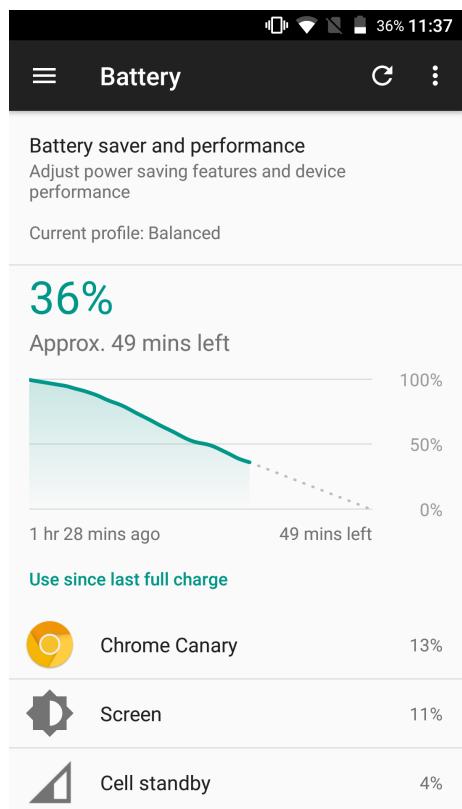
- Voelde u zich aanwezig op de plaats zelf?
- Vond u de applicatie aangenaam om te gebruiken?
- Heeft u al eerder gebruik gemaakt van virtual reality?
- Vond u het beeld goed van kwaliteit?
- Had u last van fysieke ongemakken? (hoofdpijn, duizeligheid, droge ogen, misselijkheid, ...)
- Moest u zelf beschikken over zo een 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 2de applicatie aangenaam was om te gebruiken. Ookal 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 dan vooral te merken bij de 1ste applicatie, waar 60% zich deels aanwezig voelde. Een mobiele React 360 applicatie is dus zeker nog niet een goede manier om ervoor te zorgen dat de gebruiker volledig wordt ondergedompeld in de wereld. Dit heeft dan ook hoogst waarschijnlijk 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 VR headset beschikt. Dit is dus zeker postief, 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 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 een pak eenvoudiger wordt. Een ingewikkelde applicatie ontwikkelen wordt door middel van het React 360 framework een pak makkelijker gemaakt. Men kan zeer gebruiksvriendelijk interfaces gaan ontwikkelen om hiermee solide applicaties te gaan maken. Maar dit is waar het er momenteel bij React 360 bij blijft. De doeleinden voor het framework blijven zeer beperkt. We kunnen React 360 vooral gaan gebruiken voor applicaties waarbij nood is aan een goede user interface. Denk dus maar aan applicaties voor de toeristische sector waarbij men een rondleiding kan geven op bepaalde plaatsen met behulp van VR. Ook de medische sector kan dit eventueel gaan gebruiken, maar dit blijft ook zeer beperkt. Hierbij zou de focus dan eerder liggen op educatieve applicaties waarbij men een 3D weergave kan geven van bepaalde organen. Ook immobiliën kantoren kunnen zeker gebruik maken van React 360 waarbij men 360°foto's of video's van huizen kan gaan voorstellen. De doeleinden van de applicaties die ontwikkeld kunnen worden met React 360 blijven dus vooral 'professioneel' gericht. Voor simpele entertainment zoals een film in een virtuele cinema kan men natuurlijk wel React 360 gebruiken, maar veel verder dan dat gaat het niet. Simpele games zijn mogelijk, maar verre van wat er al mogelijk is met andere technologieën en VR headsets (HTC Vive, Oculus Rift, ..).

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

Het blijft natuurlijk nog steeds virtual reality. De fysieke opmerkingen waren dus ook regelmatig aanwezig. Vooral personen die last hadden van de ogen. Maar misselijkheid en duizeligheid kwam ook een aantal keer voor. Dit blijft dus nog steeds een belangrijk werk punt voor de ontwikkelaar om hiermee rekening te houden.

Bij React 360 is er een actieve community aanwezig, iets wat dan weer een groot voordeel is voor de ontwikkelaar zelf. Andere frameworks zoals Primrose heeft geen actieve community en daardoor zal de ontwikkeling met dat framework ook een pak stroever verlopen. React 360 bezit wel nog een aantal bugs, maar is misschien eerder te wijten aan WebVR dan React 360. Dit was vooral dan te merken bij het cross platform gebruik. Zo verliep het uitvoeren van de testapplicaties op een desktop computer met de browser google chrome vlekkeloos. Alle functionaliteit werkte naar behoren. Bij de smartphone versie van chrome werden er dan weer enkele problemen ondervonden zoals het geluid dat niet correct afspeelde. Hiervoor werd dan ook een andere variant op chrome gebruikt, chrome canary, om dit te gaan oplossen. Maar dit toont dan weer aan dat React 360 nog niet 100% cross-platform is. Iets wat facebook nochtans aanhaalt op de homepage van het framework. Wat mij dan ook opviel 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, maar het hele framework kreeg een nieuwe look and feel. Zo is er een hoop functionaliteit die nog van React VR afkomstig is en dus ook nog steeds werkt in React 360, maar die nog steeds ongedocumenteerd is. React 360 doet het wel goed als hij het doet. Ookal 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 aan RAM geheugen vrij laag. Maar er moet wel een zekere 'kracht' aanwezig zijn.

Al bij al, kunnen we stellen dat React 360 zeker een goede stap in de juiste richting is. Het zet aan tot een verdere ontwikkeling en biedt ook een manier aan om op een gemakkelijke manier applicaties te ontwikkelen die wel degelijk gebruiksvriendelijk zijn, maar het aantal doeleinden blijft beperkt en is dus enkel bedoelt voor eenvoudige interactieve applicaties. Echte volwaardige applicaties? Zolang de applicatie eenvoudig blijft, dan wel. Willen we een ingewikkeldere applicatie, met gebruik van veel 3D elementen en animaties, dan is React 360 niet het juiste framework en zal de ontwikkelaar gebruik moeten maken van moeilijkere ontwikkelingsmethoden. React 360 blijft ten slotte wel een framework voor de browser en een browser heeft ook zo zijn beperkingen.

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 developer 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 developer 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 andere aspecten worden rekening 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, enz.. 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 vrije nieuwe technologie die nog maar sinds 2017 bestaat. Het is dus ook een framework dat zoals virtual reality zelf nog volop in ontwikkeling is. De API die gebruikt wordt in de browsers om virtual reality via webpagina's te ondersteunen

noemt WebVR. Dit zal dus hoogstwaarschijnlijk enkele malen aanbod komen in mijn onderzoek. 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 waarbij ik die van Michael Mangialardi, Learn React 360 (Mangialardi, 2017) ook zal gebruiken voor React 360 aan te leren.

A.3 Methodologie

Ik zal mijn onderzoek beginnen door eerst te bekijken wat React 360 allemaal te bieden heeft. Ik ga dus 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 waar 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. Verder zal ik dan hieruit gaan kijken voor welke doeleinden het dan geschikt is.

A.4 Verwachte resultaten

Ik verwacht dat React 360 voor zowel de developer 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 die ik het 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.

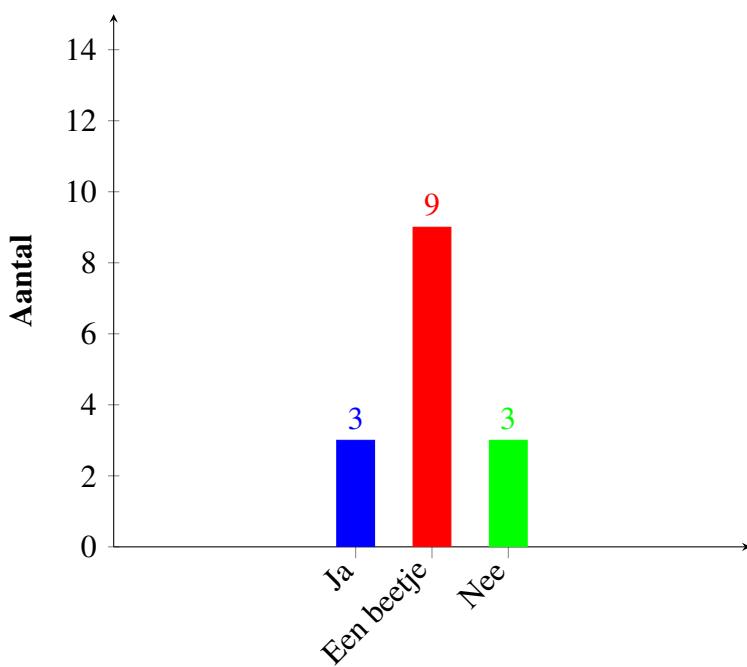
A.5 Verwachte conclusies

Ik verwacht dat React 360 een geschikt framework is voor gebruiksvriendelijke applicaties te ontwikkelen die gebruikt kan 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 gebouwd worden.

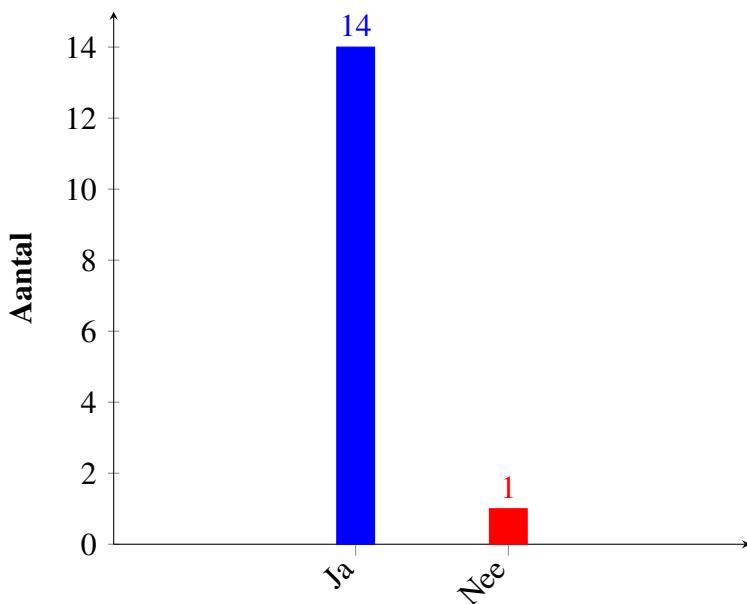
B. Bijlagen

B.1 Resultaten React 360 onderzoek

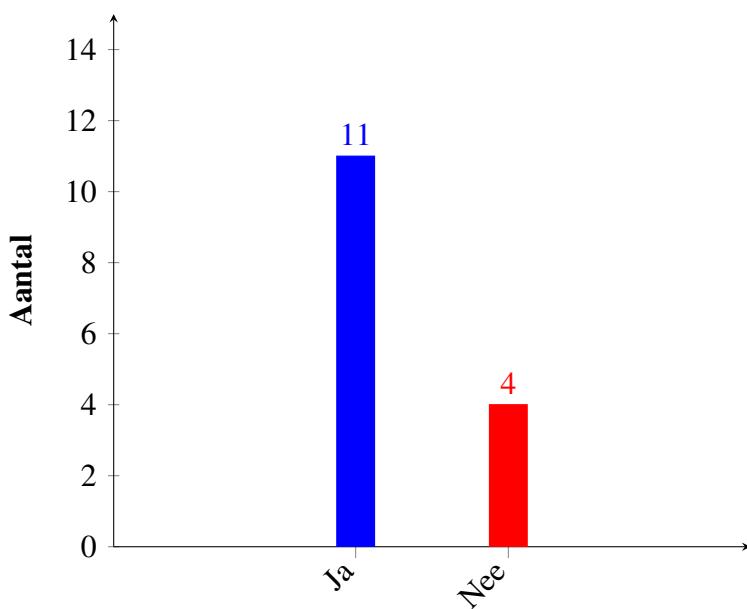
Voelde u zich aanwezig op de plaats zelf?

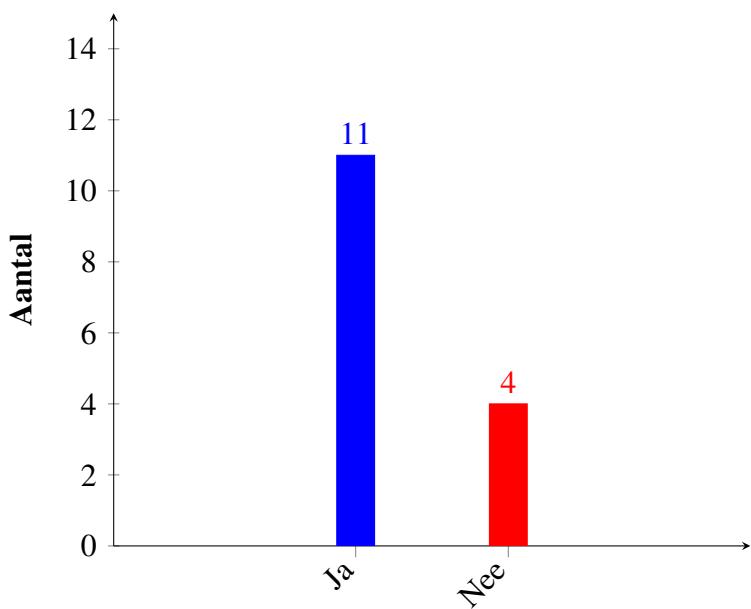


Vond u de applicatie aangenaam om te gebruiken?

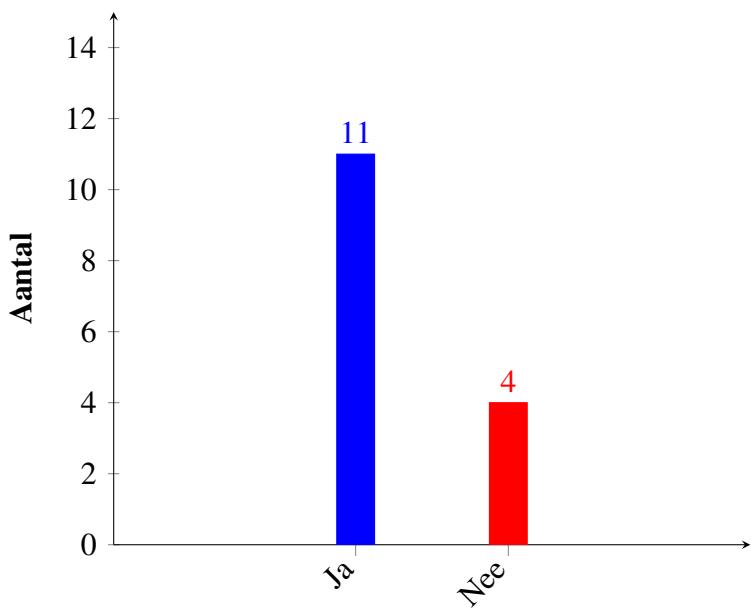


Heeft u al eerder gebruik gemaakt van virtual reality?

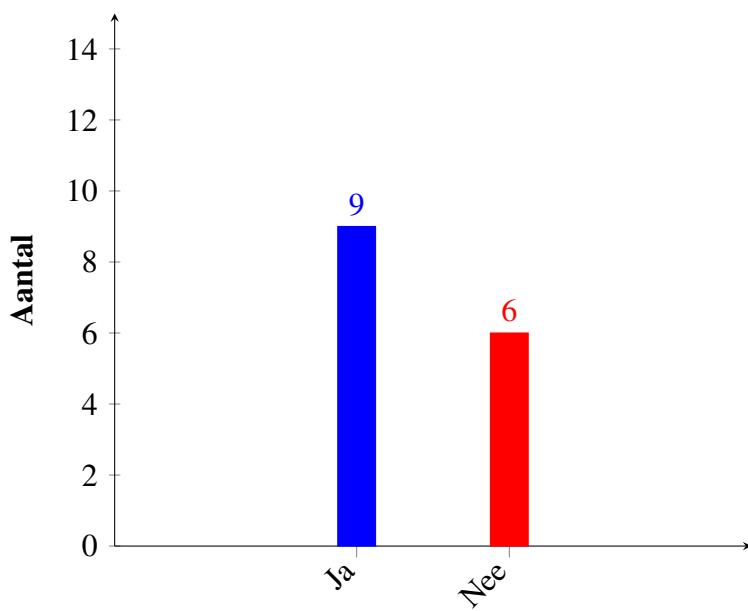


Vond u het beeld goed van kwaliteit?

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



Moest u zelf beschikken over zo een 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/>
- 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>
- 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*.