Informatica — Universiteit van Amsterdam

Bachelor Informatica

University of Amsterdam

Ontwikkelomgeving voor 360 Graden VR voor Training en Therapie

Chris Bras

4 mei 2021

Supervisor(s): Robert Belleman, Iza Scherpbier

Signed: Signees

Samenvatting

Bij de instelling voor jeugdpsychiatrie Levvel wordt een methode ontwikkeld die gebruik maakt van VR om ouders therapiesessies opnieuw thuis te laten afspelen. Om dit mogelijk te maken is een ontwikkelomgeving nodig voor VR trainingen.

Dit onderzoek richt zich op het implementeren van een toegankelijke ontwikkelomgeving voor VR trainingen en therapiesessies. Hierin wordt zowel een editor als een player gebouwd. De editor maakt het mogelijk om VR werelden te bouwen met behulp van 360 graden video en 3D modellen. De player maakt het mogelijk om deze werelden af te spelen via een mobiele telefoon. Beide applicaties zijn te vinden in een overkoepelende ontwikkelomgeving en draaien in de browser.

De ontwikkelomgeving is opgebouwd in React en communiceert via een API met de backend. In de backend worden trainingen opgeslagen in een SQL database. Video's en 3D objecten worden apart opgeslagen op het filesystem van de server als respectievelijk mp4 en glTF bestanden, hiervan wordt vervolgens de locatie in de database opgeslagen.

Voor de player is gebruik gemaakt van Babylon als 3D en VR framework. De player kan gebruikt worden op mobiele telefoons en maakt het mogelijk 360 graden video samen met 3D objecten af te spelen in de browser.

In de uiteindelijke implementatie is het mogelijk een training te bouwen van begin tot einde. De gebruiker kan deze vervolgens ook afspelen op een mobiele telefoon. Er zijn onderdelen die verbeterd kunnen worden. Ondersteuning op verschillende telefoons en besturingssystemen moet beter onderzocht worden. Gebruik van niet ondersteunde browsers en telefoons levert problemen op met zowel het gebruik van volledig scherm in de applicatie als het automatisch afspelen van video's in de browser.

Inhoudsopgave

HOOFDSTUK 1

Introductie

Virtual reality (VR) wordt vaak geassocieerd met videospellen en vergelijkbare entertainment doeleinden. Er zijn echter ook andere toepassingen mogelijk, VR biedt namelijk interactie mogelijkheden die in andere media, zoals bijvoorbeeld video, niet bereikt kunnen worden. Door een virtuele wereld om de gebruiker heen te bouwen kan er intuïtievere interactie met de media plaats vinden. Dit kan gebruikt worden bij bijvoorbeeld trainingen die in de realiteit lastig uit te voeren zijn, zoals het omgaan met gevaarlijke machines.

Naast gebruik in trainingen kan VR ook toegepast worden voor therapie sessies. In Virtual Reality Exposure Therapy (VRET) [13] wordt dit bijvoorbeeld al gedaan. Hierbij worden post traumatische stress stoornis patiënten opnieuw blootgesteld aan een traumatische ervaring met behulp van VR. Door deze situaties in een gecontroleerde omgeving opnieuw te beleven via virtual reality kunnen patiënten geholpen worden deze trauma's te verwerken.

Bij Levvel[9], een instelling voor jeugdpsychiatrie, wordt op dit moment onderzoek gedaan naar de toepassing van VR voor Parent-Child Interaction Therapy (PCIT). PCIT is een vorm van therapie waarin ouders van kinderen met gedragsproblemen onder begeleiding van een therapeut getraind worden in de omgang met hun kind[11]. Om ouders extra te laten oefenen met technieken die zij tijdens therapiesessies leren, wordt VR ingezet om scenario's te creëren die de ouders zelf thuis nog eens kunnen afspelen.

De meest geavanceerde VR ervaringen voor thuis maken gebruik van gespecialiseerde hardware. De hardware bestaat vaak uit een speciale VR bril met een scherm van hoge kwaliteit en hoge resolutie. Om deze brillen aan te sturen is dan ook veel grafische processorkracht nodig. Deze kosten van deze systemen kunnen daardoor hoog oplopen, waardoor het voor veel gebruikers niet haalbaar zal zijn om op deze manier thuis VR te gebruiken. Om er voor te zorgen dat gebruikers toch gebruik kunnen maken van VR zonder aanschaf van kostbare apparatuur wordt in dit onderzoek gekeken naar toepassingen die gebruik maken van een mobiele telefoon als VR hardware. Zo kunnen gebruikers hun eigen telefoon gebruiken en is alleen een houder nodig om de telefoon als VR bril in te zetten.

Scenes in VR kunnen zich het best onderscheiden, van bijvoorbeeld normale video's, door de mogelijkheid voor interactie van de gebruiker. Hier kan gebruik van gemaakt worden door de acties van gebruikers invloed te laten hebben op het verloop van een training. In eerder onderzoek is een systeem gebouwd waarmee non-lineaire verhaallijnen kunnen worden afgespeeld in VR [17]. Hierbij heeft de keuze van de gebruiker invloed op hoe een training zich verder afspeelt. Dit concept wordt in dit onderzoek verder als basis gebruikt en verder uitgebreid door de toevoeging van 3D objecten. 3D objecten maken het mogelijk voorwerpen aan de gebruiker te tonen die in het echt lastig te visualiseren zijn, zoals bijvoorbeeld de binnenkant van een verbrandingsmotor.

Wanneer complexe interactie van de gebruiker voor de verloop van een training ondersteund wordt, kunnen de verhaallijnen die afgespeeld moeten worden snel te complex worden. Hiervoor

is een systeem ontwikkeld waarmee gemakkelijk verhaallijnen gemaakt kunnen worden via een grafische gebruikersomgeving [16]. Deze *timeline editor* maakt het mogelijk losse scenes aan elkaar te koppelen. Om het gebruiksgemak voor het maken van een training te verhogen wordt deze editor samen met een *scene editor* geïntegreerd tot een complete ontwikkelomgeving waarin VR trainingen gemaakt kunnen worden.

De onderzoeksvraag die hieruit volgt is **Wat is er nodig om een ontwikkelomgeving te maken voor 360 graden VR trainingen?** Hierbij zijn de volgende deelvragen te onderzoeken:

- Wat is er nodig voor de backend en frontend van de ontwikkelomgeving?
- Hoe worden VR scenes opgeslagen?
- Wat is er nodig om VR trainingen op een mobiele telefoon mogelijk te maken?

HOOFDSTUK 2

Functionele Analyse

De implementatie van een training in VR kan op meerdere manieren gedaan worden. Een mogelijkheid is om een compleet virtuele wereld te bouwen. Dit kost echter veel tijd aangezien elk detail gemodelleerd zal moeten worden. Een andere mogelijkheid is het maken van een 360 graden video en dit vervolgens afspelen in VR. Deze video's kunnen dan een scene uitspelen, waarna vragen beantwoord kunnen worden door de gebruiker over wat er in deze scene heeft afgespeeld. Deze manier van scenes maken wordt al gebruikt in PCIT-VR[17].

In sommige gevallen kan het voor komen dat een video alleen niet genoeg detail kan bieden voor bepaalde objecten. In dit geval moet er de mogelijkheid zijn om 3D gemodelleerde objecten toe te voegen aan de virtuele ruimte. Aan deze objecten kan vervolgens ook interactiviteit worden toegevoegd wat een video alleen niet zou kunnen.

Om de keuzes van een gebruiker invloed te laten hebben over wat er gebeurt binnen een tijdlijn moet er een definitie zijn van wat er gebeurt bij bepaalde keuzes. Dit is aan de trainingontwerper om van te voren aan te geven in de vorm van een non-lineaire tijdlijn. Een tijdlijn bestaat uit een verzameling scenes met daarbij de koppeling tussen de scenes en de gemaakte keuzes van de gebruiker. Hierin staat aangegeven welke scene geladen moet worden wanneer een gebruiker aan het eind van de huidige scene is gekomen.

Om de drempel voor gebruik laag te houden voor VR, moeten scenes afgespeeld kunnen worden zonder gebruik te maken van gespecialiseerde hardware. Hiervoor bieden smartphones een goede uitkomst. Een moderne smartphone beschikt namelijk over alle onderdelen die nodig zijn voor VR. Zo kan een smartphone de positie van het hoofd van de gebruiker meten via de ingebouwde sensoren en is het beeldscherm voldoende voor VR toepassingen. Een ander voordeel van het gebruik van een smartphone is dat er geen externe hardware nodig is om de VR bril aan te sturen, alle processoren zitten immers als in de smartphone zelf. De VR applicatie kan vervolgens aangeboden worden via de webbrowser, waardoor een zo groot mogelijk aantal telefoons gebruikt kan worden en de drempel tot deelname zo laag mogelijk is. Naast het bekijken van de VR wereld moet de gebruiker ook de mogelijkheid tot interactie hebben. Ook hiervoor is het belangrijk dat geen speciale hardware, zoals VR controllers, nodig is.

2.1 Ontwikkelomgeving

Het maken van een training in VR kan snel ingewikkeld worden. Om deze reden is het belangrijk dat er een overzichtelijke ontwikkelomgeving is waarin een training uitgewerkt kan worden. De volgende onderdelen zijn nodig voor een gestroomlijnde ervaring binnen de ontwikkelomgeving:

- **Scene Editor**: Een plek om scenes te bouwen
- **Scene Player**: Een manier om een scene af te kunnen spelen.

- **Data Opslag**: De manier waarop trainingen worden opgeslagen
- **Account Systeem**: Een account systeem maakt het makkelijk om trainingen te distribueren tussen gebruikers

Hieronder volgt per onderdeel een korte uitleg.

2.1.1 Scene Editor

Een scene bestaat uit een video en eventueel enkele 3D objecten. Deze 3D objecten moeten binnen de virtuele ruimte op de juiste plek gezet kunnen worden. Om dit goed te kunnen zien is er een omgeving nodig waarin de trainingontwerper zowel de video kan zien als 3D objecten kan plaatsen. Dit wordt gedaan binnen de scene editor. De scene editor dient een 3D viewport te hebben waarin de video afgespeeld wordt en 3D objecten verplaatst, geroteerd en geschaald kunnen worden. Hiernaast is ook de mogelijkheid nodig om vragen toe te voegen aan een scene op een bepaald tijdstip in de video.

2.1.2 Scene Player

Na het maken van een scene is het nodig dat deze bekeken kan worden door de eindgebruiker. Hiervoor moet de scene afgespeeld kunnen worden in VR. Belangrijk hierbij is dat de gebruiker dit via een mobiele telefoon zal gaan doen. Dit betekent dat er niet van uit gegaan kan worden dat de gebruiker beschikt over speciale VR controllers. Om toch interactie mogelijk te maken zal er een vorm van *gaze to pick* geïmplementeerd moeten worden. *Gaze to pick* laat de gebruiker objecten selecteren door er langere tijd naar te kijken. Als de gebruiker hun blik richt op een object gaat er een timer lopen. Wanneer deze afloopt en de gebruiker nog steeds naar het te selecteren object kijkt, wordt deze geselecteerd.

2.1.3 Data Opslag

Video's en 3D modellen zijn relatief grote bestanden. Wanneer deze worden hergebruikt in meerdere scenes kan de benodigde data opslag snel oplopen. Om dit te voorkomen zullen scenes moeten bestaan uit referenties naar video's en 3D modellen. Hiervoor is een overkoepelende *asset* opslag nodig waar de video's en 3D modellen kunnen worden opgeslagen om vervolgens aan een scene gekoppeld te worden.

2.1.4 Account Systeem

Bij het gebruik van een centrale applicatie met meerdere gebruikers is een systeem nodig waarbij gebruikers kunnen inloggen op een account. Dit account systeem is de link tussen alle door een gebruiker aangemaakte data en zorgt dat dit niet zonder toestemming benaderd kan worden door andere gebruikers. Belangrijk hiervoor is dat gebruikers kunnen inloggen met een unieke gebruikersnaam en wachtwoord en dat dit veilig wordt opgeslagen in de database.

Het gebruik van accounts maakt het ook mogelijk om per gebruiker analytics op te slaan. Zo kan er gemonitord worden hoe iemand door een training heen loopt.

2.2 Ethiek

Ethiek is een belangrijk onderwerp wanneer het gaat om therapiesessies aangezien deze vaak van persoonlijke aard zijn. Hierom wordt er voorzichtig omgegaan met de data die vergaard kan worden met de resultaten van dit onderzoek. Zo kunnen gebruikers niet zomaar bij therapie sessies van andere gebruikers. Ook wanneer ze de precieze URL weten te gokken is er nog een extra wachtwoord nodig om toegang te krijgen tot een training. Therapeuten kunnen alleen bij trainingen die zij zelf gemaakt hebben. Hiervoor moeten ze inloggen op hun account dat beveiligd is met een wachtwoord. Dit wachtwoord staat als een hash opgeslagen in de database. In het geval van een database lek zullen deze wachtwoorden op deze manier niet leesbaar zijn.

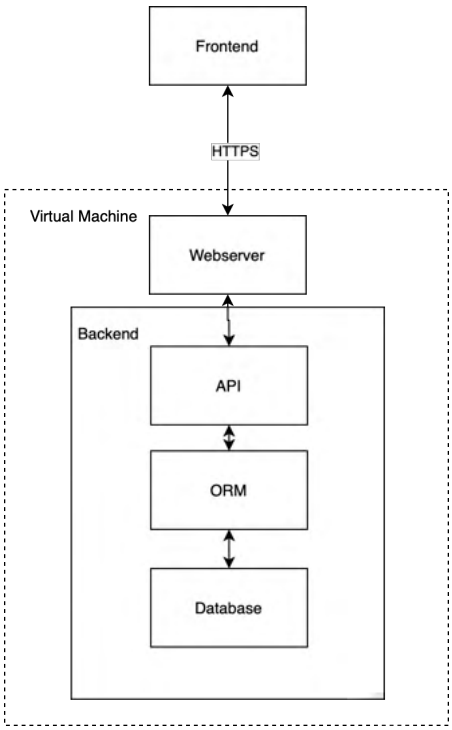
HOOFDSTUK 3

Implementatie

In dit hoofdstuk zal stap voor stap door de implementatie van de VR training omgeving gelopen worden. Hierbij worden keuzes die gemaakt zijn behandeld en specifieke implementatie besproken.

3.1 Software Stack

In deze sectie wordt de software en tooling besproken waarmee de eerder genoemde functionaliteit geïmplementeerd is. De softwarestack bestaat uit twee onderdelen, een frontend en een backend. Deze twee onderdelen communiceren via HTTP requests. Dit alles wordt gehost door een webserver en draait op een virtuele machine. In figuur 3.1 is een schematische weergave van de softwarestack te zien.



Figuur 3.1: Schematische weergave van de softwarestack.

3.1.1 Frontend

De frontend is het gedeelte van de applicatie dat de gebruiker via een interface toegang geeft tot de data op de server. In deze context bestaat de frontend uit een webapplicatie die via HTTPS requests data ophaalt uit de backend en deze beschikbaar stelt voor de gebruiker. Een webapplicatie bestaat in de meest simpele vorm uit een combinatie van HTML, CSS en Javascript. Gezien de complexe aard van dit project is er gekozen voor het gebruik van een framework dat deze elementen op een eenvoudige manier samenvoegt. Bij de keuze voor een framework is gekeken naar de volgende punten:

- Mogelijkheden tot het bijhouden van een state
- Het maken van API calls
- Beschikbaarheid van documentatie en support
- Beschikbaarheid van libraries, packages en design-kits

Voor deze toepassing is gekozen voor VueJS[18] en React[12]. Beide frameworks bieden de mogelijkheid om een state bij te houden. De API calls kunnen in beide frameworks afgehandeld worden via Axios[3], een op *promises* gebaseerde HTTP client voor Nodejs.

Zowel VueJS als React beschikken over goede documentatie, echter is React een populairder framework met meer gebruikers waardoor voor bepaalde problemen makkelijker een oplossing te vinden is op bijvoorbeeld forums. Hiernaast wordt React ontwikkeld en onderhouden door Facebook, wat voordelen heeft voor de lange termijn support.

Door de grootte van de community rondom React zijn er veel libraries en packages beschikbaar. Deze kunnen met de package manager *npm* geïnstalleerd en beheerd worden. De packages die gebruikt moeten worden, zoals Axios bijvoorbeeld, zijn echter in beide frameworks beschikbaar.

Tot slot wordt gekeken naar de beschikbaarheid van design-kits. Een design-kit draagt bij aan het consistent ogen van een website door de styling van veelgebruikte componenten, zoals knoppen en invoervelden, van te voren te definiëren. Material-UI[10] is een veel gebruikt en herkenbaar voorbeeld hiervan en wordt onder andere gebruikt door Amazon en Spotify. Material-UI stelt de componenten uit de design-kit beschikbaar via een package voor React. Samen met de eerder genoemde vergelijkingen is dit de reden dat gekozen is voor React als framework voor de frontend.

3.1.2 API

De frontend communiceert met de backend via een REST API. Deze API is geschreven in *Python* en maakt gebruik van *Flask*[5]. Flask is een framework dat helpt bij het maken van web applicaties. Doordat Flask weinig afhankelijkheden heeft, wordt het gezien als *lightweight* en *barebones*. Een nadeel hiervan is dat de programmeur zelf nog veel moet doen, maar hierdoor wel meer vrijheid heeft. Om dit te verhelpen is naast Flask ook gebruik gemaakt van *Flask RestX* [6]. Dit is een extensie op Flask die taken als invoervalidatie en marshalling van objecten afhandelt.

3.1.3 Database

Alle benodigde data voor de ontwikkelomgeving wordt opgeslagen in een database. De vrije relationele databaseserver PostgreSQL wordt hiervoor gebruikt. PostgreSQL is in veel opzichten vergelijkbaar met SQL server. Beide zijn geschikt om efficiënt data te managen in een database systeem. Een voordeel van PostgreSQL is dat het cross-platform werkt en volledig gratis te gebruiken is[14].

De API communiceert met de database via SQLAlchemy, een ORM voor Python[15]. Het gebruik van een ORM voorkomt dat de programmeur zelf SQL queries moet schrijven, dit bespaart tijd en maakt het uitvoeren van database queries minder foutgevoelig. Ook geeft dit de mogelijkheid om database objecten direct te koppelen aan Python objecten. In combinatie met object marshalling wordt zo het opvragen van data uit de backend versimpeld. Een API call vanuit de frontend kan

zo met weinig code omgezet worden tot database call, waarvan het resultaat automatisch wordt omgezet naar een JSON object dat door de frontend te lezen is.

3.1.4 Webserver

Zowel de frontend als de backend moeten vanaf het internet bereikbaar zijn. Om dit mogelijk te maken is een webserver nodig die de bestanden van de frontend host en de endpoints van de backend bereikbaar maakt. Voor dit onderdeel is gekozen voor *NGINX*. De webserver heeft in dit geval drie verschillende taken. Ten eerste wordt de website voor de frontend gehost. Hiernaast wordt het ook als algemene fileserver gebruikt voor het hosten van assets zoals video's en 3D modellen. Als laatste zorgt de webserver ook voor een proxy naar de backend. Dit wordt gedaan omdat NGINX een uitgebreidere webserver is dan de ingebouwde webserver van Flask. Hierdoor kunnen bijvoorbeeld fouten beter worden afgevangen.

3.1.5 Virtuele machine

Docker wordt gebruikt als virtuele machine om de gehele applicatie in te draaien. Dit zorgt er voor dat alle onderdelen van de applicatie in een gecontroleerde omgeving opgestart kunnen worden. Een ander voordeel van een virtuele machine is dat het de applicatie gemakkelijk te verplaatsen maakt tussen verschillende hardware. Zo zal de backend exact hetzelfde werken op de productieserver als op een ontwikkelmachine.

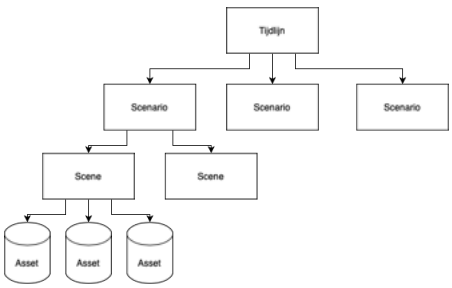
3.2 VR Ontwikkelomgeving

In deze sectie wordt de implementatie van de VR ontwikkelomgeving besproken. Deze ontwikkelomgeving is boven op de eerder genoemde softwarestack geïmplementeerd en bevat onderdelen zowel in de frontend als in de backend.

3.2.1 Projecten

Een van de belangrijkste functionaliteiten voor de trainingen in VR is het maken van niet-lineaire scenario's. De gebruiker krijgt een fragment te zien met daarin een vraag over wat zich afspeelt in het fragment. Aan de hand van welke keuze de gebruiker maakt, wordt bepaald welk fragment deze hierna te zien krijgt. Zo beïnvloedt de keuze dus hoe een gebruiker door het verhaal heen loopt.

Een scenario bestaat uit een niet-lineaire opeenvolging van *scenes*. Deze *scenes* bestaan zelf uit een video en eventueel een aantal 3D objecten. Deze video's en 3D modellen zullen vanaf nu *assets* genoemd worden. Naast deze *assets* beschikt een scene ook over een vraag met mogelijke antwoorden voor de gebruiker waarmee een verhaal voortgezet kan worden. Scenario's kunnen ook achter elkaar getoond worden om zo een *tijdlijn* te creëren. Zo kan bijvoorbeeld onderzocht worden hoe een gebruiker reageert op het meemaken van bepaalde scenario's achter elkaar. In figuur 3.2 is deze structuur weergegeven.



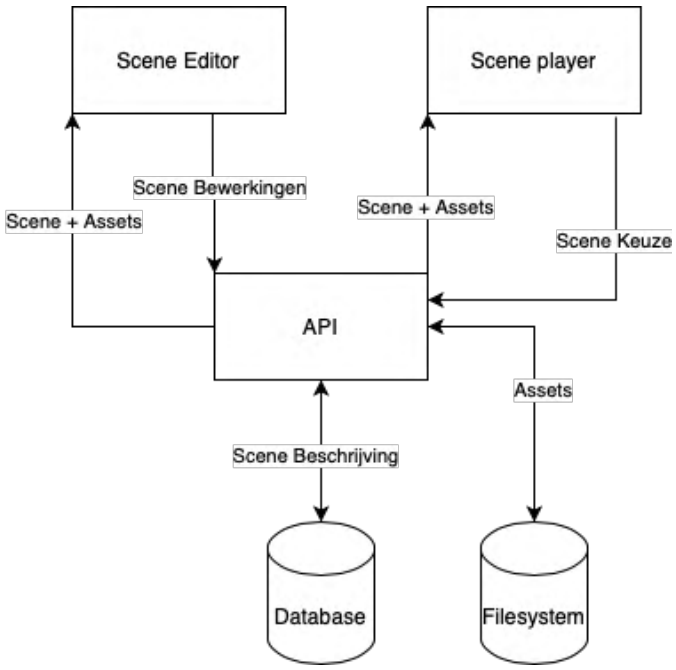
Figuur 3.2: Concepten binnen de VR omgeving en hun onderlinge relaties.

Elk onderdeel van een tijdlijn kan ook gebruikt worden voor elk ander onderdeel in een tijdlijn. Zo kan een asset bijvoorbeeld in meerdere scenes gebruikt worden, een scene in meerdere scenario's en kunnen scenario's in meerdere tijdlijnen voorkomen. Uiteindelijk moet de gebruiker die de verhaallijnen maakt een manier hebben om overzicht te houden, hiervoor zijn *projecten* gemaakt. Een gebruiker kan assets, scenes, scenario's en tijdlijnen toevoegen aan een project, en deze vervolgens vrij gebruiken binnen dit project. Een project is dus een overkoepelende datastructuur. Een gebruiker heeft ook de mogelijkheid meerdere projecten aan te maken, om zo voor verschillende doeleinden een training te kunnen bouwen.

3.2.2 API Koppeling

Projecten, tijdlijnen, scenario's, scenes en assets worden opgeslagen in een database, die te bereiken is via een API. De scene editor en scene player zijn beide gekoppeld via de API en communiceren dus niet direct met elkaar. In figuur 3.3 is een globaal overzicht weergegeven van de API.

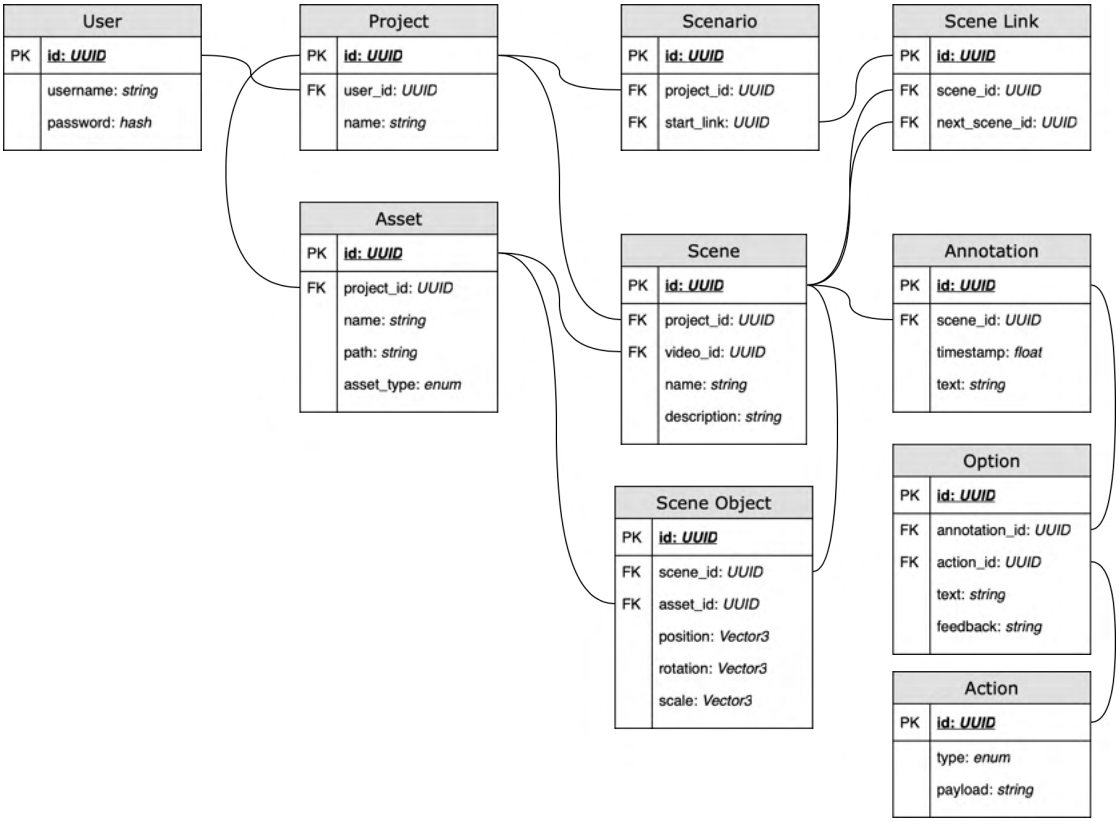
De scene editor ontvangt vanuit de API de beschrijving van een scene met de beschikbare assets binnen het huidige project. Vervolgens stuurt deze dan de bewerkingen van de scene terug naar de API, die deze opslaat in de database. De scene player kan vervolgens deze scenes ophalen uit de API, met bijbehorende assets. De assets zelf worden opgeslagen als referenties in de database. De bestanden zelf staan weggeschreven op het filesystem van de machine waar de API op draait.



Figuur 3.3: Schematische weergave van de API koppeling tussen scene editor en scene player. Een scene wordt opgebouwd via de beschrijving uit de database en de bijbehorende assets uit het filesystem. De editor en player ontvangen deze scenes. De editor kan bewerkingen terugsturen, de player stuurt een aanvraag voor een scene aan de hand van de keuze van de gebruiker.

3.2.3 Database

De database slaat projecten, tijdlijnen en scenes direct op. Assets worden zelf op het filesystem opgeslagen waarbij vervolgens de database referenties naar de locaties op het filesystem bevat. In figuur 3.4 is een versimpelde versie van de *entity relationship diagram* (ERD) te zien.



Figuur 3.4: *Entity relationship diagram* van de database.

3.2.4 Accountsysteem

Voordat de gebruiker iets kan doen met de applicatie, moet deze inloggen. Hiervoor is een account nodig. Via dit account wordt bijgehouden wie er bij welke projecten, video's, objecten en andere onderdelen kan. Een account bestaat uit drie onderdelen, een gebruikersnaam, een wachtwoord en een uniek gegenereerde *identifier*. Deze *identifier* wordt gebruikt om informatie over die gebruiker te anonimiseren. De gebruikersnaam en wachtwoord combinatie wordt gebruikt om in te loggen. De gebruiker wordt hierna geïdentificeerd door de systemen via de unieke *identifier*. Belangrijk hierbij is dat logingegevens veilig worden opgeslagen, hiervoor is het nodig dat de gegevens niet als niet versleutelde tekst worden opgeslagen.

Wachtwoord hash

Om het wachtwoord veilig op te slaan in de database wordt er eerst een *salt* aan het wachtwoord toegevoegd. Hierna wordt de hash van deze combinatie opgeslagen. Bij *salting* van een wachtwoord wordt een reeks aan willekeurige tekens toegevoegd aan het wachtwoord voordat deze door een hash functie gaat. Dit voorkomt dat wachtwoorden die hetzelfde zijn niet dezelfde hash krijgen, en dus in het geval van een database lek hierdoor de wachtwoord hashes niet met elkaar vergeleken kunnen worden. In listing 3.1 is een voorbeeld implementatie in Python gegeven.

Listing 3.1: Voorbeeld implementatie van een wachtwoord hash met salting in Python.

```
import hashlib, binascii, os

def hash_password(password):
    """Hash a password for storing."""
    salt = hashlib.sha256(os.urandom(60)).hexdigest().encode('ascii')
    pwdhash = hashlib.pbkdf2_hmac('sha512', password.encode('utf-8'),
                                  salt, 100000)
```

```
    pwdhash = binascii.hexlify(pwdhash)
    return (salt + pwdhash).decode('ascii')
```

Vervolgens kan een ingevoerd wachtwoord geverifieerd worden door de hash hiervan te vergelijken met de opgeslagen hash. Hiervoor is wel belangrijk dat de salt van het opgeslagen wachtwoord bekend is, deze moet immers toegevoegd worden aan het ingevulde wachtwoord, anders zal er niet dezelfde hash uit komen. In listing 3.2 is een voorbeeld implementatie van deze verificatie gegeven.

Listing 3.2: Voorbeeld implementatie van een wachtwoord verificatie in Python.

```
import hashlib, binascii, os

def verify_password(stored_password, provided_password):
    """Verify a stored password against one provided by user"""
    salt = stored_password[:64]
    stored_password = stored_password[64:] # get the salt from the stored passwora
    pwdhash = hashlib.pbkdf2_hmac('sha512',
                                  provided_password.encode('utf-8'),
                                  salt.encode('ascii'),
                                  100000)
    pwdhash = binascii.hexlify(pwdhash).decode('ascii')
    return pwdhash == stored_password
```

3.3 Scene Editor

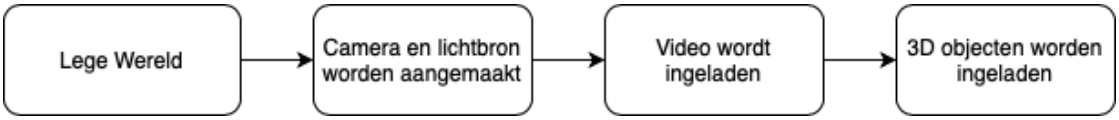
De scenes die aangemaakt zijn in de projecten kunnen worden geopend in de scene editor. Hierin kunnen de scenes verder bewerkt en opgeslagen worden.

Centraal in de scene editor staat de 3D viewport. Voor het implementeren van deze viewport is gebruik gemaakt van Babylon [4] als 3D javascript framework. Babylon biedt een eenvoudige manier om 3D werelden, inclusief 360 graden video's, te renderen in de webbrowser. Hiernaast maakt Babylon het ook mogelijk de wereld door een VR camera te bekijken. Hierbij kunnen de bewegingssensoren op een mobiele telefoon gebruikt worden om de camera te bedienen.

Naast Babylon bestaat ook het framework A-Frame [2]. Dit biedt net als Babylon de mogelijkheid om, al dan niet in VR, een 3D wereld te renderen in een webbrowser. Een groot verschil tussen A-Frame en Babylon is dat A-Frame voornamelijk gebaseerd is op het gebruik van HTML-componenten. Dit betekent dat een 3D ervaring creëren relatief eenvoudig is, hoewel dit weinig controle over de onderliggende engine biedt. Aangezien voor de editor nauwere controle gewenst is, is uiteindelijk voor Babylon gekozen als framework.

3.3.1 Scene Inladen

Wanneer de editor opgestart wordt voor een scene, moet deze vanuit de database geladen worden. Dit gebeurt in drie losse stappen, zoals te zien in figuur 3.5. Als eerste wordt er een wereld gecreëerd. Deze wereld bevat twee onderdelen, een camera en een lichtbron. Nadat de wereld is opgebouwd wordt eerst gekeken of er in de huidige scene al een video geselecteerd is, wanneer dit het geval is, wordt een video-bol aangemaakt. Op deze bol wordt vervolgens aan de binnenkant de video als texture geprojecteerd. Nadat de video is geladen, worden de 3D objecten voor de scene opgehaald en geladen.



Figuur 3.5: Overzicht van de stappen die worden genomen voor het opbouwen van de 3D view in de scene editor.

3.3.2 3D Objecten Laden

Het laden van 3D objecten heeft twee onderdelen. Allereerst moeten de meshes, de sub onderdelen van een object, ingeladen worden. Deze zijn te vinden in een bestand op de server. Nadat deze geladen zijn, moeten ze op de juiste plek in de wereld geplaatst worden. Om de juiste translatie, rotatie en schaal te toe te passen op alle meshes van een object, worden deze aan elkaar gekoppeld via een parent-child hiërarchie. Aan de top van deze hiërarchie staat de zogenaamde *root mesh*. Door de meshes onderling te koppelen kan via Babylon een transformatie uitgevoerd worden op de root mesh, waarna alle child meshes volgen. In listing 3.3 is te zien hoe een object wordt ingeladen.

Listing 3.3: Het laden van 3D objecten en toepassen van transformaties gegeven een lijst aan meshes met Babylon.

```
((meshes) => {

  // set rootmesh for the model with object ID as name
  let rootMesh = new AbstractMesh(object.id);
  rootMesh.alwaysSelectAsActiveMesh = true

  meshes.forEach((mesh) => {
      // set the parent only if the mesh doesn't already have a parent
      // to preserve mesh hierarchy
      if (!mesh.parent) {
          mesh.parent = rootMesh
      }
  })

  // apply stored transformations to rootMesh
  rootMesh.position = new Vector3(object.x_pos,
                                  object.y_pos,
                                  object.z_pos
                                 );
  rootMesh.rotationQuaternion = new Quaternion(object.x_rotation,
                                               object.y_rotation,
                                               object.z_rotation,
                                               object.w_rotation
                                              );
  rootMesh.scaling = new Vector3(object.x_scale,
                                 object.y_scale,
                                 object.z_scale
                                );
})
```

3.3.3 3D Object Bestanden

Er zijn veel verschillende formats beschikbaar voor het opslaan van 3D objecten. Bekende voorbeelden van dit soort formats zijn Wavefront OBJ[1] bestanden en FBX[8] bestanden. Wanneer deze bestanden echter in de webbrowser gebruikt zouden worden, krijgt de gebruiker te maken

met lange laadtijden. Om deze reden is gekozen voor het glTF bestandstype[7]. GlTF is vergeleken de andere efficiënter 3D data overbrengen. Hierdoor is een glTF loader snel en efficiënt te bouwen wat het voor deze toepassing een goede keuze maakt. Naast een efficiënte manier van bestanden laden biedt glTF ook toekomst perspectief en is de verwachting dat support in de toekomst alleen maar zal toenemen[7].

3.3.4 Scene Bewerken

In de scene editor kan een scene bewerkt worden. Zo kan de belichting worden aangepast, maar ook de positie van objecten binnen de 3D ruimte kan worden veranderd door de gebruiker. Om een 3D object aan te kunnen passen moet deze eerst door de gebruiker geselecteerd kunnen worden. Hiervoor wordt de *GizmoManager* van Babylon gebruikt. De GizmoManager biedt functionaliteit om objecten te selecteren en hier vervolgens controls aan toe te voegen. Dit wordt gedaan door bij elke klik een straal te schieten, wanneer deze straal kruist met een object en dit object in de configuratie als *selectable* staat aangegeven wordt deze geselecteerd. Bij het selecteren van een object wordt altijd de root mesh als ankerpunt gekozen, hierdoor komen de eventuele transformaties altijd overeen met het gene wat in de database staat opgeslagen.

3.4 Scene Player

De onderliggende implementatie van de scene player lijkt op dat van de scene editor. Beiden maken ze gebruik van dezelfde onderliggende systemen om scenes in te laden en de wereld te renderen. Het verschil zit voornamelijk in de manier waarop de wereld getoond wordt en hoe de gebruiker interactie heeft met de 3D wereld.

3.4.1 VR Camera

Bij het creëren van de wereld voor de scene player wordt in plaats van een normale camera een VR camera gebruikt. Deze camera is onderdeel van de *VRExperienceHelper* binnen Babylon. Deze helper creëert twee losse camera's, een voor elk oog, en bestuurt deze camera via de bewegingssensor inputs op een mobiele telefoon. In listing 3.4 is de configuratie voor deze helper te zien die gebruikt is in deze implementatie.

Listing 3.4: Gebruik van de Babylon VRHelper om een VR ervaring te creëren.

```
let vrHelper = scene.createDefaultVRExperience();
// Needs to be true to update position
vrHelper.displayGaze = true;
// mobile phones need to enable this
vrHelper.enableGazeEvenWhenNoPointerLock = true;
// exits VR view when user double taps
vrHelper.exitVROnDoubleTap = true;
// lock cursor if user enters fullscreen
vrHelper.requestPointerLockOnFullScreen = true;
// set the camera view to world origin
vrHelper.position = new Vector3(0,0,0);
```

3.4.2 Gebruikersinteractie

Bij het gebruik van een mobiele telefoon kan er niet verwacht worden dat de gebruiker beschikt over VR controllers. Om toch objecten te kunnen selecteren binnen de 3D wereld is een vorm van *gaze to pick* geïmplementeerd. Bij *gaze to pick* moet de gebruiker voor een bepaalde tijd kijken richting een object om dit te selecteren. Dit is geïmplementeerd door voor elke rendercyclus een straal te schieten vanuit het midden van het scherm. Als deze straal kruist met een object wordt de huidige status van de gebruiker naar *selecting* gezet. Hierbij wordt ook een waarde opgehoogd. Wanneer deze waarde een bepaalde drempelwaarde bereikt heeft, wordt de actie als selectie geregistreerd.

Als de straal geen object kruist, wordt deze waarde terug gezet naar nul en is de gebruiker niet meer bezig met selecteren. Hierdoor zal de gebruiker aan een stuk door moeten kijken naar een object voordat deze geselecteerd wordt. In listing 3.5 is een voorbeeld implementatie hiervan gegeven.

Listing 3.5: Voorbeeld implementatie van het selecteren van een object via gaze to pick.

```
let selectedObject = undefined

let selecting = false
let selectionProgress = 0
let selectionThreshold = 100

func onRender() {
  let ray = castRay()

  if (ray.hit) {
    // increase selection progress when object is hit
    selectionProgress += 1
    if (selectionProgress == selectionThreshold) {
      selectedObject = ray.intersectedObject
    }
  } else {
    // reset selection tracking and deselect any selected object
    selectionProgress = 0
    selectedObject = undefined
  }
}
```

3.4.3 Annotaties

In een scene kan een annotatie getoond worden aan de gebruiker. Deze annotatie kan een vraag bevatten met een set aan antwoorden waaruit de gebruiker kan kiezen. Om een annotatie te kunnen tonen op het juiste moment wordt er aan het video object dat zich afspeelt in de scene een callback functie toegevoegd. Deze callback functie wordt elk frame van de video aangeroepen en geeft het huidige tijdstip door. Wanneer dit tijdstip groter of gelijk is aan het tijdstip van de annotatie die bij de video hoort, wordt de video op stop gezet en de annotatie getoond. De annotatie zelf bestaat uit een verzameling 3D objecten. Zowel de vraag als de opties worden gerenderd naar een texture, welke vervolgens op een plane in de ruimte wordt geprojecteerd. De positie van deze planes wordt bepaald door de kijkrichting van de gebruiker. Op deze manier zal de gebruiker altijd doorhebben dat er een vraag beschikbaar is. In listing 3.6 wordt de functionaliteit getoond waarmee de positie van een annotatie berekend wordt.

Listing 3.6: Functie om de positie van een annotatie te berekenen gegeven de huidige camera.

```
let forwardRay = camera.getForwardRay()

// put position of annotation 5.5 units in front of user
let planePosition = camera.position + forwardRay.direction.scale(5.5)

// plane rotation is the flipped version of camera rotation in the Y axis
// this makes sure the plane is always facing the user
let planeRotation = camera.rotation * vector3(1, -1, 1)
```

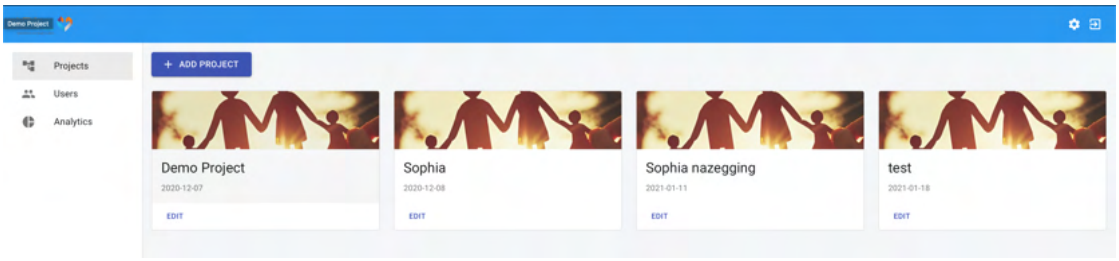
HOOFDSTUK 4

Resultaten

In dit hoofdstuk worden de behaalde resultaten getoond en besproken. De getoonde versie is een implementatie voor de PCIT-VR trainingen die bij Levvel ontwikkeld worden.

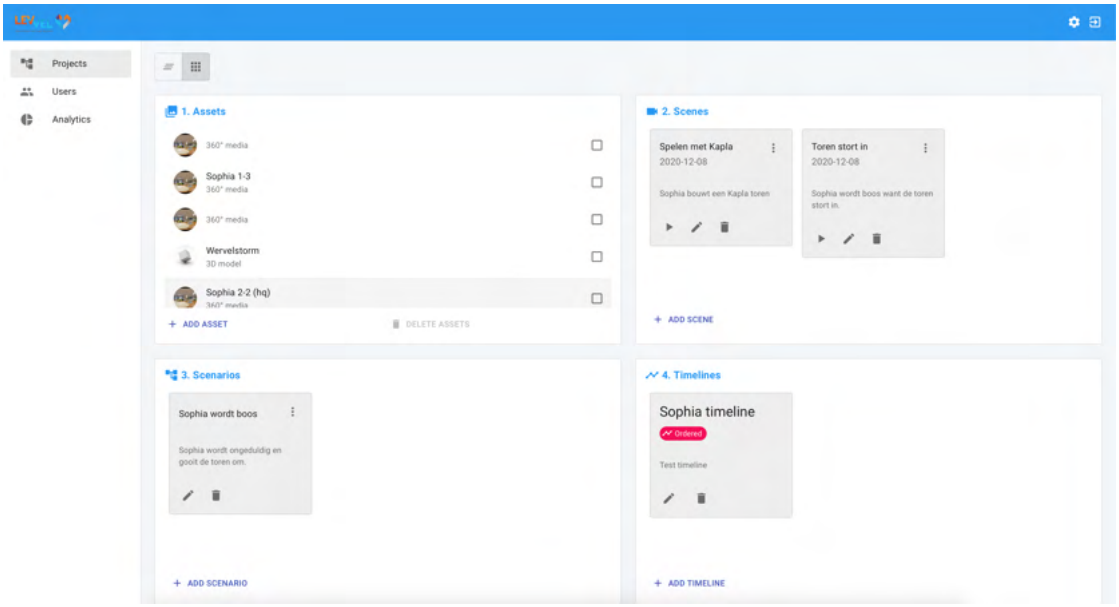
4.1 PCIT-VR Ontwikkelomgeving

Om de gebruikservaring te verbeteren zijn de verschillende onderdelen van de VR trainingen, zoals de scenario editor en de scene editor, gecombineerd tot een omgeving. In deze omgeving kan de gebruiker van begin tot eind een training ontwikkelen en uitleveren aan de eindgebruiker. Een therapeut kan hierop inloggen en krijgt dan meteen alle beschikbare projecten te zien voor het huidige account, zoals te zien in figuur 4.1.



Figuur 4.1: Startpagina voor de PCIT-VR ontwikkelomgeving. Hierin ziet de gebruiker alle beschikbare projecten op het huidige account.

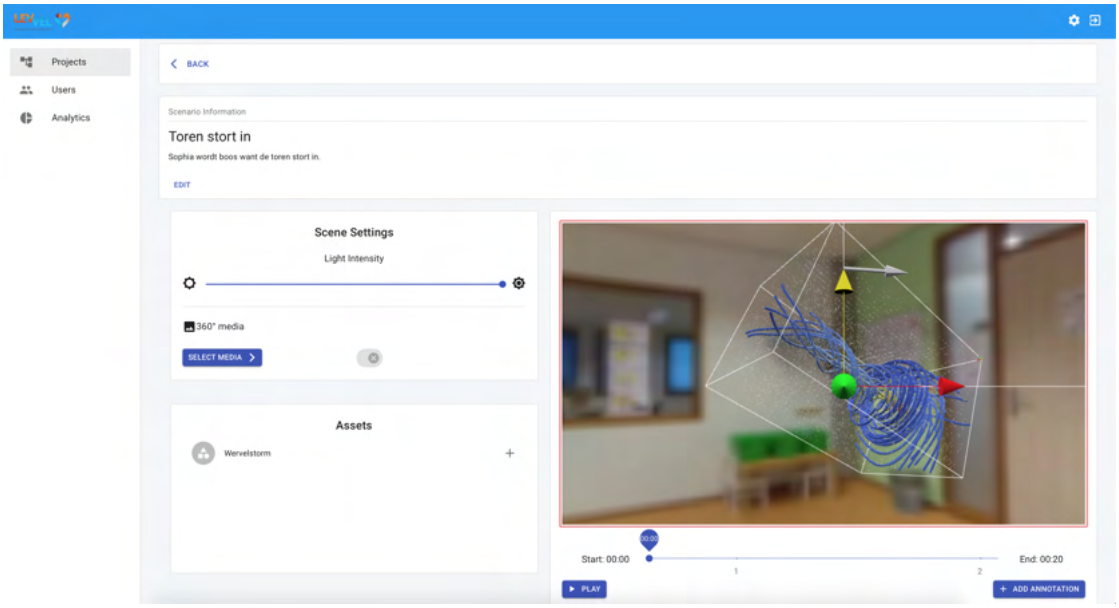
Als de gebruiker een project opent, krijgt deze direct een overzicht te zien van alles binnen dit project, zoals te zien in figuur 4.2. Hier kan vervolgens per onderdeel de bijbehorende editor bereikt worden. Ook kunnen hier nieuwe assets zoals video's en 3D objecten geüpload worden naar het project.



Figuur 4.2: Overzicht van een project.

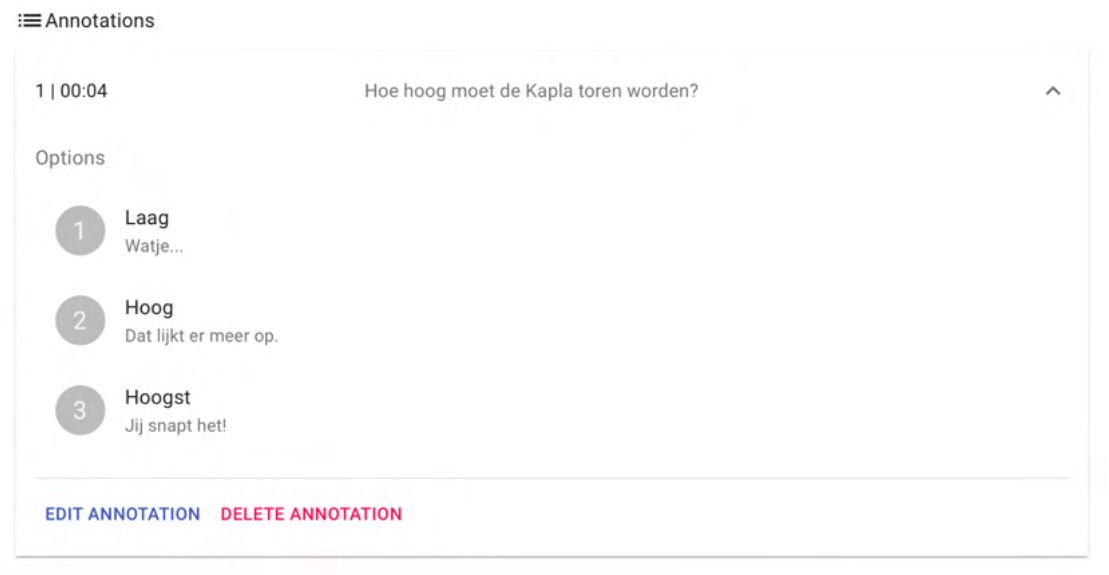
4.2 Scene Editor

De scene editor maakt het mogelijk voor de gebruiker om scenes aan te passen. Hierin kan de video gekozen worden en 3D objecten toegevoegd worden. De editor beschikt ook over simpele video transport functionaliteit om gemakkelijk door de video te bewegen, de getallen onder deze transport geven aan welke annotatie op welk punt in de video verschijnt. In figuur 4.3 is het interface van de scene editor te zien.



Figuur 4.3: Het interface van de scene editor.

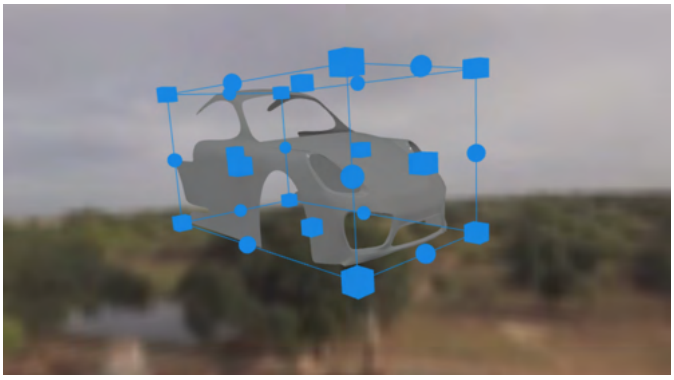
In deze editor kunnen ook de annotaties worden aangepast. Hiervoor is wordt het menu uit figuur 4.4 gebruikt.



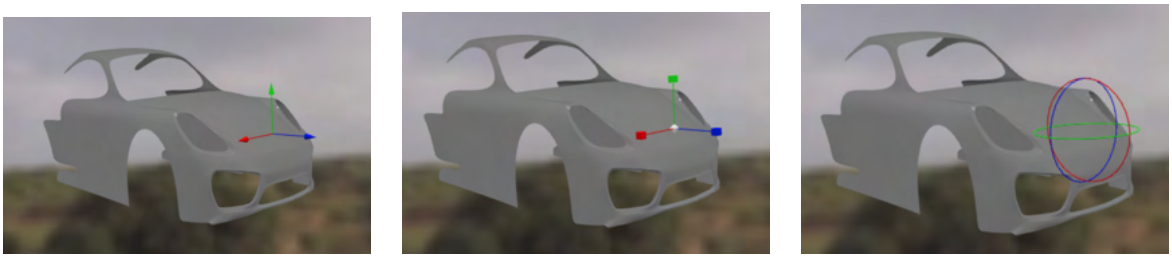
Figuur 4.4: Het menu om annotaties mee te bewerken.

4.2.1 3D viewport

Centraal in de scene editor staat de 3D viewport. Hier kan de gebruiker aanpassingen maken aan de 3D wereld. Objecten kunnen verplaatst, geschaald en geroteerd worden. Bij het selecteren van een object, krijgt de gebruiker enkele controls te zien. Er zijn vier typen controls mogelijk. Standaard zal een object een *box-control* krijgen, zoals te zien in figuur 4.5. Met deze control kan de gebruiker drie transformaties uitvoeren op het object, namelijk rotatie, translatie en schalen. Naast deze *box-control* kan de gebruiker ook kiezen om de drie transformaties als losse controls te gebruiken, te zien in figuur 4.6.



Figuur 4.5: 3D object geslecteerd met box-control.



(a) translate-control (b) scale-control (c) rotate-control

Figuur 4.6: Verschillende basis controls.

4.3 Scene Player

Nadat een scene gemaakt is kan deze getest worden in de scene player. Deze is te bereiken door in het project overzicht bij de scenes op de play knop te drukken. De scene player rendert de scene in VR mode. Dit betekent dat er twee camera's gebruikt worden, zoals te zien in figuur 4.7.

Figuur 4.7: VR beeld van de scene player.

4.3.1 Gebruikersinvoer

De gebruikersinvoer wordt gedaan via de eerder besproken *gaze to pick* functionaliteit. Om de gebruiker extra feedback te geven op wat er geselecteerd wordt, is een cirkel in het midden van het scherm toegevoegd. Deze cirkel wordt groter naarmate het selectieproces vordert. Wanneer de cirkel op het grootste punt gekomen is, is de selectie voltooid. Naast het groter worden van een cirkel zal ook het selecterende object veranderen van kleur om aan te geven waar de gebruiker precies naar kijkt. Het selecteren van een object is te zien in figuur 4.8.

(a) De gebruiker kijkt buiten de kubus, deze wordt nu nog niet geselecteerd.

(b) De gebruiker kijkt richting de kubus, deze zal nu geselecteerd worden.

Figuur 4.8: Het proces van een object selecteren via *gaze to pick*. Wanneer de gebruiker gedurende een bepaalde tijd naar een object kijkt, wordt deze geselecteerd.

HOOFDSTUK 5

Conclusie

Het doel van dit onderzoek was om een ontwikkelomgeving te bouwen die geschikt is voor het maken van generieke VR trainingen. Het resultaat hiervan is een applicatie waarin van begin tot eind een training kan worden ontworpen. Deze omgeving biedt ook de mogelijkheid om deze training vervolgens aan te bieden aan de eindgebruiker. Deze applicatie is online beschikbaar en wordt gebruikt voor de ontwikkeling van PCIT-VR therapiesessies bij de instelling voor jeugd psychiatrie Levvel.

Uit het gebruik bij Levvel is gebleken dat de applicatie niet op elke telefoon goed werkt. Zo zijn er problemen met ondersteuning voor het automatisch afspelen van video met geluid op mobiele webbrowsers, sommige video's worden wel afgespeeld en anderen niet. Hiernaast is gebleken dat het gebruik van volledig scherm niet consistent is tussen verschillende telefoons van verschillende fabrikanten. Op niet elke telefoon kan gebruik gemaakt worden van een scherm vullende weergave, waardoor de adresbalk van de browser altijd in beeld blijft staan. Dit neemt weg van de VR ervaring voor de gebruiker.

In dit onderzoek is de keuze gemaakt voor het gebruik van Babylon in plaats van A-Frame voor het mogelijk maken van 3D in de webbrowser. Uiteindelijk is de extra controle die Babylon biedt ten opzichten van A-Frame alleen nodig gebleken in het maken van de 3D-editor. De player maakt geen gebruik van geavanceerde functionaliteiten die niet mogelijk zijn in A-Frame. Dit betekent dat de player in A-Frame verder ontwikkeld had kunnen worden, wat mogelijk voor snellere ontwikkeling gezorgd zou hebben, gezien het gebruiksgemak van A-Frame.

5.1 Vervolgonderzoek en Algemene Verbeteringen

Analyse is een belangrijk onderdeel bij trainingen en therapiesessies om inzicht te krijgen in wat een gebruiker doet en waar deze op vast loopt. De huidige omgeving biedt hier nog geen volledige ondersteuning voor. In de backend wordt er wel al rekening gehouden met analytics, maar deze worden nog niet getoond in de applicatie. Om de nieuwe speler goed te kunnen gebruiken moet deze op nuttige momenten analytics bijhouden, hier kan vervolg onderzoek zich op richten.

Op dit moment is het gebruik binnen de applicatie nog niet overal consistent. Niet alles wat de gebruiker kan toevoegen kan ook weer verwijderd worden. Hier zou ook verder werk aan gedaan moeten worden om alles binnen de applicatie consistent te maken.

Bibliografie

[1] URL: http://paulbourke.net/dataformats/obj/.

[2] *A-Frame*. URL: https://aframe.io.

[3] *Axios*. URL: https://github.com/axios/axios.

[4] *Babylon JS*. URL: https://www.babylonjs.com.

[5] *Flask*. URL: https://flask.palletsprojects.com/en/1.1.x/.

[6] *Flask RestX*. URL: https://flask-restx.readthedocs.io/en/latest/.

[7] Ben Houston. *glTF: Everything You Need to Know*. Aug 2020. URL: https://www.threekit.com/blog/gltf-everything-you-need-to-know.

[8] Ben Houston. *When should you use FBX 3D file format ?* Aug 2020. URL: https://www.threekit.com/blog/when-should-you-use-fbx-3d-file-format.

[9] *Levvel*. URL: https://levvel.nl/.

[10] *Material-UI*. URL: https://material-ui.com.

[11] Cheryl Bodiford. McNeil e.a. *Parent-child interaction therapy*. Springer, 2010.

[12] *React*. URL: https://reactjs.org.

[13] B. O. Rothbaum, L. Hodges en Rob Kooper. "Virtual reality exposure therapy". English (US). In: *Journal of Psychotherapy Practice and Research* 6.3 (jul 1997), p. 219–226. ISSN: 0002-9564.

[14] *SQL Server vs PostgrSQL*. URL: https://www.educba.com/sql-server-vs-postgresql/.

[15] *SQLAlchemy*. URL: https://www.sqlalchemy.org.

[16] Emil van Veen. "A visual editor for creating non- linear timelines for PCIT-VR". Universiteit van Amsterdam, 2021.

[17] Mick Vermeulen. "Towards a production-ready framework for PCIT-VR." Universiteit van Amsterdam, 2019. URL: https://scripties.uba.uva.nl/search?id=714701.

[18] *Vue JS*. URL: https://vuejs.org.