Deel 5

VIRTUAL REALITY EN AUGMENTED REALITY IN DE BROWSER

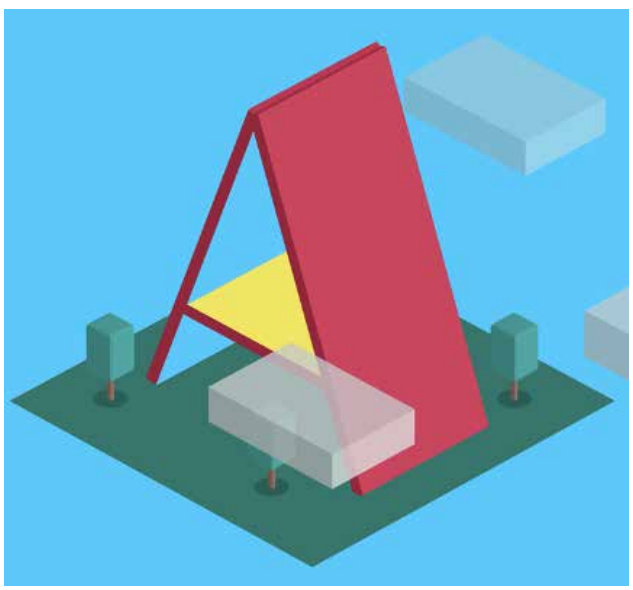
Dit deel over Progressive Web Apps gaat over het implementeren van Virtual Reality (VR) en Augmented Reality (AR) in je webapp. Hiervoor maken we gebruik van A-Frame en de WebXR Web API.

Peter Eijgermans is Frontend Developer en Codesmith bij Ordina JSRoots. Hij deelt graag zijn kennis met anderen middels workshops en presentaties.

VR verplaatst gebruikers naar een andere wereld en laat hen deze beleven alsof ze zichzelf in deze virtuele wereld bevinden. AR daarentegen biedt de mogelijkheid om de echte en de virtuele wereld in elkaar te laten overvloeien. Zowel VR als AR bieden unieke mogelijkheden voor gedragsverandering, kennisoverdracht, training en onderzoek. Denk bijvoorbeeld aan VR-live streaming, social-VR en kunst. Dit alles is nu ook mogelijk in de browser!

{ WAT IS A-FRAME? }

A-Frame is een web framework voor het bouwen van VR-ervaringen en AR-ervaringen. A-Frame is gebaseerd op HTML. De kern is een krachtig framework voor entiteiten dat een uitbreidbare en herbruikbare structuur biedt. A-Frame is gebouwd bovenop WebXR en Three.js en ontwikkelaars hebben onbeperkte toegang tot JavaScript, DOM API's, Three.js, WebXR en WebGL. In deze context is WebXR een Web API voor het creëren van VR/AR-ervaringen in je browser. Three.js is een VR/AR-framework en WebGL zorgt voor de rendering.

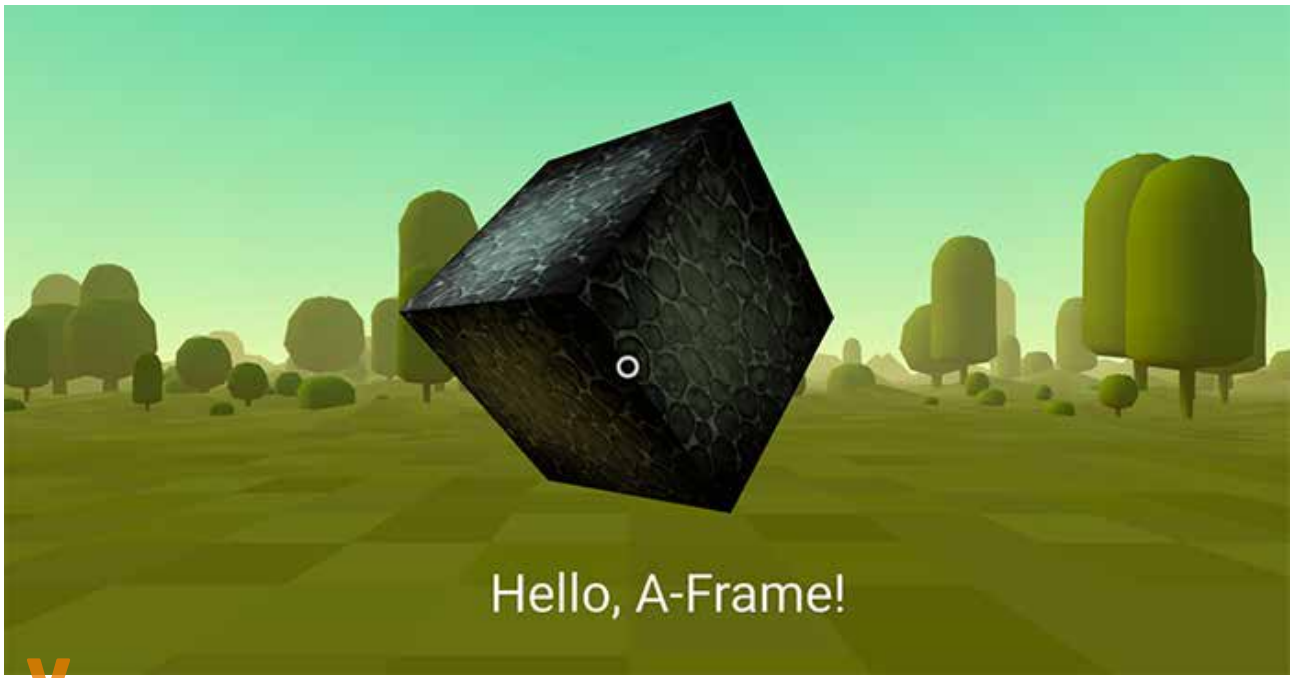


{ WAAR TE STARTEN? }

Om te beginnen met het leren van A-Frame ga je naar A-Frame School: https://aframe.io/aframe-school/#/. A-Frame applicaties kunnen worden ontwikkeld vanuit een HTML-bestand zonder iets te installeren! Een goede manier om A-Frame uit te proberen, is door middel van het 'starters voorbeeld' op 'Glitch': **https://glitch.com/~aframe**. Glitch is een online IDE voor JavaScript en Node.js.

{ ECS IN A-FRAME }

A-frame is gebaseerd op de Entity-Component-System (ECS) architectuur. ECS-architectuur is een veelvoorkomend patroon in 3D- en game-ontwikkeling die het 'composition over inheritance' principe volgt. Dat wil zeggen dat iedere entiteit samengesteld (composed) is uit componenten (zie paragraaf: Abstracte voorbeelden van entiteiten). Entiteiten zijn binnen ECS containerobjecten waaraan componenten kunnen worden gekoppeld.



Afbeelding 1.

En componenten zijn herbruikbare modules die aan entiteiten kunnen worden gekoppeld om entiteiten functionaliteit te bieden. Alle logica wordt geïmplementeerd via componenten en we definiëren verschillende soorten entiteiten door componenten te configureren en te mixen.

A-Frame heeft API's die elk onderdeel van ECS vertegenwoordigen:

- Entiteiten worden vertegenwoordigd door het `<a-entity>` element.
- Components worden vertegenwoordigd door de HTML-attributen op de `<a-entity>`'s.

{ ABSTRACTE VOORBEELDEN VAN ENTITEITEN }

Hieronder zie je enkele abstracte voorbeelden van hoe je een entiteit kan samenstellen (composition) met componenten. 'Sphere' (of bol) is in dit voorbeeld een entiteit. En 'Position', 'Geometry' en 'Material' zijn de componenten waaruit deze entiteit bestaat.

Sphere = Position + Geometry + Material
Ball = Position + Velocity + Physics + Geometry + Material

{ SYNTAX ENTITY IN A-FRAME }

Conform het vorige voorbeeld ziet de syntax van een entity voor een sphere er als volgt uit (Listing 1).

Dus om deze sphere te tekenen, definieer je een `<a-entity>` en voeg je componenten toe als HTML-attributen. De meeste componenten hebben meerdere eigenschappen (properties) die worden weergegeven door een syntax die lijkt op CSS.

L1

```
<a-entity geometry="primitive: sphere; radius:
1.5"
       material="color: white; shader: flat;
        src: glow.jpg"
       position="0 0 -5"></a-entity>
```

L2

```
Primitive:
<a-box color="red" width="3"></a-box>
Entity:
<a-entity geometry="primitive: box; width: 3"
material="color: red"></a-entity>
```

{ PRIMITIVES }

A-Frame biedt een handvol elementen zoals `<a-box>` of `<a-sky>`, primitieven genaamd. 'Primitives' zijn de gebruiksvriendelijke HTML-elementen van A-Frame die de onderliggende entiteitcomponenten omhullen. Ze zijn gemaakt om het ontwikkelen aantrekkelijk te maken voor beginners. In Listing 2 zie je een voorbeeld van de primitive `<a-box>`.
Onderstaande `<a-box>` primitive bestaat onder de motorkap uit een `<a-entity>` met geometrie- en material-componenten.

{ BUILD A VR-SCENE IN DE BROWSER }

Laten we beginnen met het bouwen van een basis VR-scene met A-Frame met entiteiten en animaties (Afbeelding 1).

{ EERSTE STAP }

Als eerste stap maak je een folder aan voor dit project. Vervolgens maak je een index.html document aan in deze 'project folder' en daarin plaats je de HTML uit Listing 3.

Je neemt eerst een recente versie van A-Frame op als een script-tag in de `<head>`. Voor de laatste versie ga je naar: **https://github.com/aframevr/aframe/releases**. Vervolgens neem je de `<a-scene>` tag op in de `<body>`.

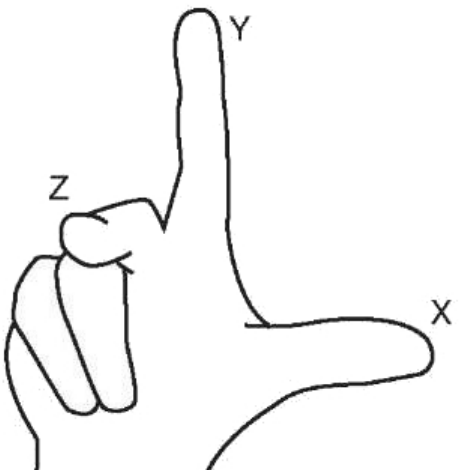
{ WAT IS EEN SCENE? }

Een scene is de plek waar alles gebeurt. Alle entiteiten en componenten dienen allemaal aan de scene te worden toegevoegd, zodat ze gerenderd worden. In A-Frame wordt de scene vertegenwoordigd door een `<a-scene>` entity. `<a-scene>` behandelt alle Three.js and WebXR Boilerplate voor ons, zoals het instellen van WebGL (deze zorgt voor de rendering), het canvas, de camera, het licht, de renderer, de renderloop en kant-en-klare WebXR-ondersteuning op platforms zoals HTC Vive, Oculus Rift en Samsung GearVR. De scene entity erft van de klasse entity. Het erft dus al zijn eigenschappen, de mogelijkheid om componenten te koppelen en het gedrag om te wachten op al zijn onderliggende child entities en nodes (bijv. `<a-assets>` en `<a-entity>`) om te laden, voordat de renderloop wordt gestart.

{ TOEVOEGEN ENTITY }

Binnen onze `<a-scene>` voeg je 3D-entiteiten toe met behulp van een van A-Frame's standaard 'primitives', zoals `<a-box>`. In Listing 4 is de kleur van een `<a-box>` gedefinieerd.

Omdat de standaard camera en de `<a-box>` zich echter op de standaardpositie op de 0 0 0-oorsprong bevinden, kan je de box niet zien tenzij je deze verplaatst. Je kunt dit doen door het 'positiecomponent' te gebruiken om de box in 3D-ruimte te transformeren.



Afbeelding 2.

L3

```
<html>
   <head>
      <script src="https://aframe.io/
      releases/1.2.0/aframe.min.js"></script>
   </head>
   <body>
      <a-scene>
      </a-scene>
   </body>
</html>
```

L4

```
<a-scene>
   <a-box color="red"></a-box>
</a-scene>
```

L5

```
<a-scene>
     <a-box color="red" position="0 2 -5"
     rotation="0 45 45" scale="2 2 2">
     </a-box>
</a-scene>
```

L6

```
<a-scene>
    <a-box position="0 2 0" rotation="0 45 45"
    scale="2 4 2">
        <a-sphere position="1 0 3"></a-sphere>
    </a-box>
</a-scene>
```

{ TRANSFORMING AN ENTITY IN 3D }

Laten we eerst de 3D-ruimte bekijken. A-Frame maakt gebruik van een rechtshandig coördinatensysteem (zie Afbeelding 2). Met de standaard camerarichting: positieve X-as strekt zich naar rechts uit, positieve Y-as strekt zich uit naar boven en de positieve Z-as steekt uit het scherm naar ons toe.

{ ONZE <A-BOX> VOOR DE CAMERA ZICHTBAAR MAKEN }

Laten we de box voor de camera zichtbaar maken, roteren en schalen. Om het zichtbaar te maken, kun je de box vijf meter terugschuiven op de negatieve Z-as met het positiecomponent. Je dient de box ook twee meter omhoog te verplaatsen op de positieve Y-as, zodat de box de grond niet snijdt. Daarnaast pas je ook de rotatie en de schaal aan, die ook gebruik maken van hetzelfde rechtshandige coördinatensysteem. Dit zal onze box onder een hoek draaien en de grootte ervan verdubbelen (Listing 5).

L7

```
<head>
   <script src="https://aframe.io/
   releases/1.2.0/aframe.min.js"></script>
   <script src="https://unpkg.com/aframe-
   environment-component/dist/aframe-
   environment-component.min.js">
   </script>
</head>
```

L8

```
<a-scene>
    <a-box color="red" position="0 2 -5"
    rotation="0 45 45"
           scale="2 2 2"></a-box>
    <!-- Out of the box environment! -->
    <a-entity environment="preset: forest;
    dressingAmount: 500"></a-entity>
</a-scene>
```

L9

```
<a-scene>
    <a-assets>
        <img id="boxTexture" src="https://i.
        imgur.com/mYmmbrp.jpg">
    </a-assets>
    <a-box src="#boxTexture" position="0 2 -5"
    rotation="0 45 45" scale="2 2 2">
        <a-sphere position="1 0 3"></a-sphere>
    </a-box>
    <a-entity environment="preset: forest;
    dressingAmount: 500"></a-entity>
</a-scene>
```

De afstandseenheid van A-Frame is in meters, omdat de WebXR API posities in meters retourneert. Bij het ontwerpen van een scene voor VR is het belangrijk om rekening te houden met de echte wereldschaal van de entiteiten. Een box met `height="10"` ziet er misschien normaal uit op je desktop, maar in VR zal de box enorm lijken. De rotatie-eenheid is in graden, hoewel deze intern wordt geconverteerd naar radialen bij het overgaan naar Three.js.

{ PARENT AND CHILD TRANSFORMATIES }

A-Frame HTML vertegenwoordigt een 3D-scene graph. In een scene graph kunnen entiteiten een enkele parent en meerdere children hebben. Onderliggende entiteiten (childs) erven transformaties (positie, rotatie en schaal) van hun bovenliggende entiteit (de parent).

We kunnen bijvoorbeeld een sphere (bol) hebben als child van een box in Listing 6.

Als we de positie van de sphere berekenen, zou het 1, 2, 3 zijn. Dit wordt bereikt door de bovenliggende positie van de sphere samen te stellen met zijn eigen positie. Eveneens zal de sphere de rotatie en schaal van de box erven.

{ TOEVOEGEN VAN EEN OMGEVING MET EEN OMGEVINGSCOMPONENT }

Met A-Frame kunnen ontwikkelaars herbruikbare componenten maken, die anderen gemakkelijk kunnen gebruiken. Een voorbeeld is de 'environment component'. Dit component genereert een verscheidenheid aan omgevingen voor ons met een enkele regel HTML. De environment component is een gemakkelijke manier om onze VR-toepassing visueel op te starten en biedt meer dan een dozijn omgevingen met tal van parameters.

Voeg eerst de environment component toe met een script tag. Hierbij wordt gebruik gemaakt van unpkg.com. Via unpkg.com krijg je de meest actuele CDN-link naar de betreffende npm-module (Listing 7).

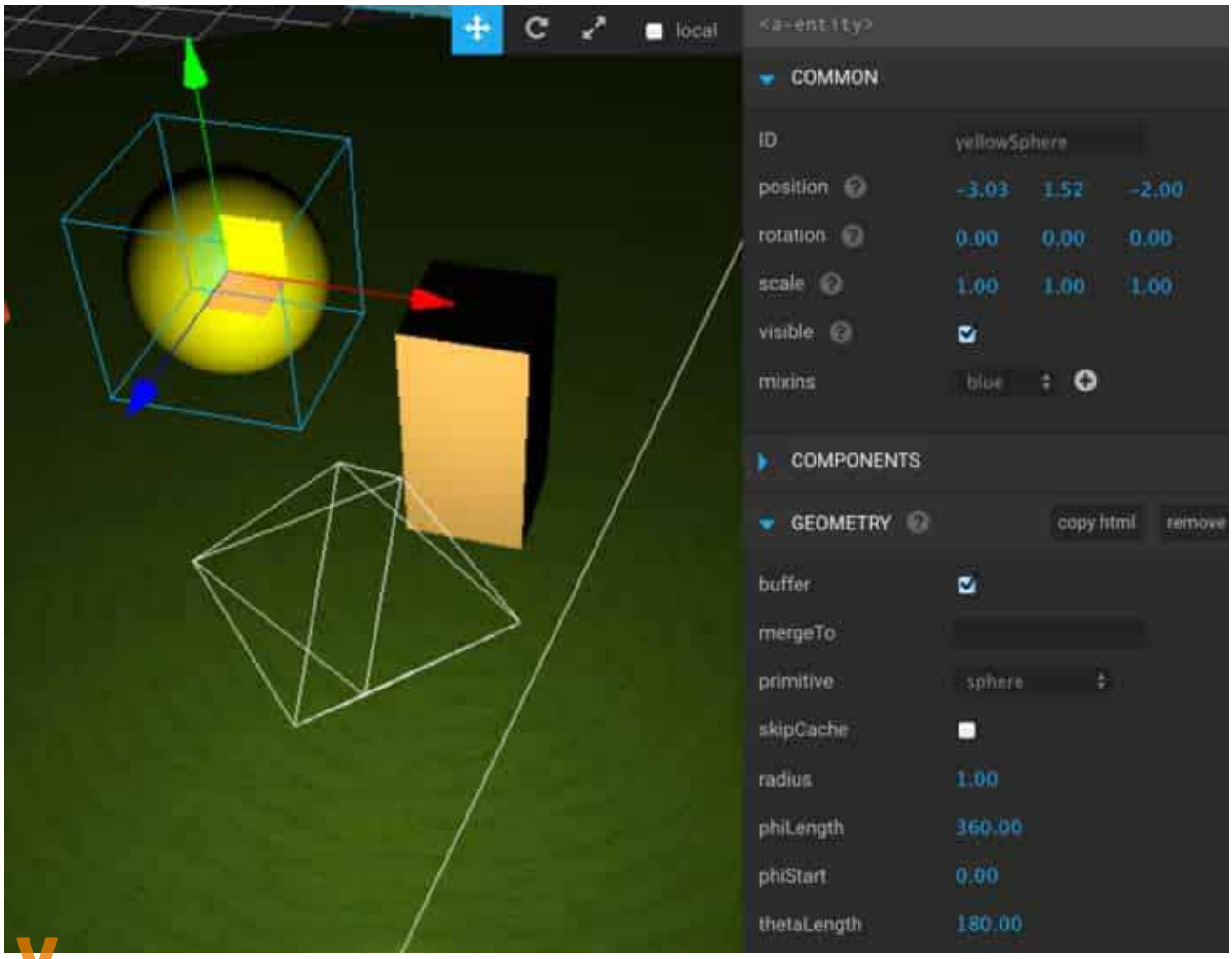
Voeg vervolgens binnen de `<scene>-tag` een a-entity toe waaraan de environment component is gekoppeld (Listing 8). Je kunt een vooraf ingestelde omgeving specificeren (bijvoorbeeld `preset: forest`). En vele andere parameters (zoals 'aantal bomen') middels 'dressing amount'.

Presets zijn een combinatie van parameterwaarden die een bepaalde stijl definiëren, ze zijn een startpunt dat je kunt aanpassen. Deze parameters kun je aanpassen via de inspector (ctrl + alt + i). De inspector (afbeelding 3) is een visueel hulpmiddel voor het inspecteren en aanpassen van je gebruikte entities/componenten. De inspector is vergelijkbaar met de DOM-inspector van de browser, maar is op maat gemaakt voor 3D en A-Frame.

{ TOEVOEGEN VAN EEN ASSET MANAGEMENT SYSTEM }

Je kan een textuur toepassen op de box met behulp van een `<img>`-element. Asset Management maakt het mogelijk om images en video's te laden en te cachen, voordat de scene gerenderd wordt. Het vooraf laden en cachen verbetert de performance. In Listing 9 wordt een voorbeeld gegeven van hoe Asset Management is toegepast op een image.

In Listing 9 wordt een `<a-asset>` tag gedefinieerd, waarbinnen `<img>` met de afbeelding textuur is gedefinieerd. Deze img heeft ook een `id="boxTexture"`. Tenslotte wordt in de box gerefereerd naar dit id middels de `src="#boxTexture"`.



Afbeelding 3.

{ TOEVOEGEN VAN ANIMATIE }

Je kunt animatie toepassen op de box en de sphere met behulp van een animatiecomponent (Listing 10). Om de box op en neer te laten bewegen, stel je het volgende in:
De box op de Y-as tussen 2 en 2.2 meter laten bewegen, stel je in met `property: object3D.position.y; to: 2.2`.
De direction (dir) is alternate. Dat wil zeggen dat er afwisselend op en neer gegaan wordt. De duration geeft aan hoe lang een cycle duurt (2000 milliseconden). En loop geeft aan dat de animatie oneindig herhaald wordt.

Het eindresultaat van bovenstaande kun je vinden op Glitch: https://glitch.com/edit/#!/vr-example-peter.

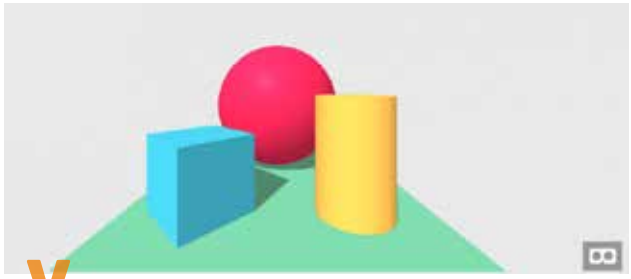
{ AUGMENTED REALITY }

Met behulp van A-Frame is het ook mogelijk om AR-ervaringen te implementeren. Als uitgangspunt voor een AR-implementatie is uitgegaan van het 'starters voorbeeld' op Glitch (https://glitch.com/~aframe), zie Afbeelding 4.

LI0

```
<a-scene>
    <a-assets>
        <img id="boxTexture" src="https://i.
        imgur.com/mYmmbrp.jpg">
    </a-assets>
    <a-box src="#boxTexture" position="0 2 -5"
    rotation="0 45 45" scale="2 2 2"
           animation="property: object3D.
           position.y; to: 2.2; dir: alternate;
           dur: 2000; loop: true">
        <a-sphere position="1 0 3"></a-sphere>
    </a-box>

    <a-entity environment="preset: forest;
    dressingAmount: 500"></a-entity>
</a-scene>
```



Afbeelding 4.

Afbeelding 5.

Dit voorbeeld is uitgebreid voor een AR 'Hit-testing' toepassing. Het resultaat kan je vinden in Listing 11. Wat vet gedrukt is, geeft aan wat er is toegevoegd. Deze implementatie kun je ook vinden via: https://glitch.com/edit/#!/ar-basic-hittest. De final versie test je op je mobiel via: https://ar-basic-hittest.glitch.me. Je maakt voor AR gebruik van je camera!

{ WAT IS HIT-TESTING IN AR? }

Met Hit-testing kun je virtuele objecten inline plaatsen met objecten uit de echte wereld, zoals de vloer, tafels en muren. Met het voorbeeld in de volgende paragraaf kun je de objecten in Afbeelding 4 inline plaatsen met bijvoorbeeld de tafel, door je camera op de tafel te richten (met een richtkruis) en deze tafel dan aan te raken (Hit-testing).

{ UITLEG }

Om Hit-testing in AR voor elkaar te krijgen, moet je eerst de `ar-components.js` in je script tag opnemen. Deze library kun

LI1

```
<html>
<head>
    <script src="https://aframe.io/
    releases/1.2.0/aframe.min.js"></script>
    <script src="/ar-components.js"></script>
</head>
<body>
<a-scene webxr="optionalFeatures: hit-test">
    <a-entity id="world" scale="0.1 0.1 0.1">
        <a-box color="#4CC3D9" position="-1 0.5
        0" rotation="0 45 0"></a-box>
        <a-sphere color="#EF2D5E" position="0
        1.25 -2" radius="1.25"></a-sphere>
        <a-cylinder color="#FFC65D" height="1.5"
        position="1 0.75 -0"
                    radius="0.5"></a-cylinder>
        <a-plane color="#7BC8A4" height="4"
        position="0 0 -1" rotation="-90 0 0"
                 width="4"></a-plane>
    </a-entity>
    <a-entity ar-hit-test="target:#world;">
        <a-plane height="0.2"
        material="transparent:true"
        rotation="-90 0 0"
                 src="./arrow.png"
                 width="0.2"></a-plane>
    </a-entity>
</a-scene>
<script>
    const reticle = document.querySelector("[ar-
    hit-test]");
</script>
</body>
</html>
```

je kopiëren uit de final Glitch implementatie. In de scene tag geef je aan dat je gebruik wil maken van de WebXR feature: hit-test. Vervolgens wordt er een realistische schaal toegepast van de objecten via de a-entity. Tenslotte wordt er een nieuwe entity gedefinieerd met het component: ar-hit-test. Dit component maakt het mogelijk de virtuele objecten inline te plaatsen op een werkelijk object die je door je camera ziet (Afbeelding 5). Ter ondersteuning van Hit-testing definiëer je een 'richtkruis' (reticle) in een vierkant van 20 centimeter. Dit doe je middels de a-plane component.

{ TENSLOTTE }

Er zijn allerlei toepassingen mogelijk met VR en AR in A-Frame. Je kan met Hit-testing bijvoorbeeld een 'AR-basketball game' maken. Een voorbeeld hiervan is te vinden onder Medium: 'making an ar-game with a-frame'. Voor meer voorbeelden en documentatie ga je naar: https://aframe.io. <