Het leren van Three.js

Dit document heb ik gemaakt om mijn proces te laten zien. Dit proces is van het oefen met Three.js. De documentatie ga ik ook gebruiken voor wanneer ik aan mijn code voor project x begin en het kan teruglezen.

Op dinsdag 6 mei heb ik een bootcamp gevolgd van Maikel (een docent die proffesional is in front end development) over Three.js. De volgende dingen werden behandeld in de bootcamp:

- ThreeJS: Installing en running it
- Canvas... scene... lights... camera... (action)
- Geometry, materials & mesh
- Rendering (60fps) & moving
- Helpful tips: clickthrough, transparent canvas, help axis, Orbital controls (camera)

Tijdens de les heb ik mee gevolgd met de code en aantekeningen gemaakt. Ik heb vooral aantekeningen gemaakt van de opzet. Ik merk dat ik de aantekeningen van de opzet vaak goed kan gebruiken wanneer ik het nog eens probeer.

Mijn aantekeningen:

Three.js bootcamp
-node.js nodig (check in terminal met npm -v)
-Eerst gaan we three.js installeren
+ga naar https://www.npmjs.com/package/three
+daar zie je de code voor three.js
+Er is een gedeelte Install: npm i three
+in je mapje in de terminal typ het bovenste
+nu installeerd die de folder
-er staat een versie in package.json wanneer je node_modules verwijderd of je code deelt met andere haalt die daar je versie van dependencies

-https://threejs.org/manual/#en/installation daar zie je hoe je het moet linken

```
<body>
  <script type="module" src="/main.js"></script>
</body>
```

-dan een main.js file aanmaken
-import * as THREE from 'three' (in main.js)

-

```
<script type="importmap">
{
 "imports": {
  "three": "https://cdn.jsdelivr.net/npm/three@<version>/build/three.module.js",
  "three/addons/": "https://cdn.jsdelivr.net/npm/three@<version>/examples/jsm/"
 }
}
</script>
```

-

```
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>threejs op website</title>
  <script type="importmap">
    {
```

```
        "imports": {
          "three": "./node_modules/three/build/three.module.js",
          "three/addons/": "./node_modules/three/examples/jsm/"
        }
      }
      </script>
      <script type="module" src="/main.js"></script>
</head>
-script main.js moet niet in de body maar in de head
-import file location moet veranderd worden
-no more error msg in de live server
```

Challenge

Na de bootcamp is er een challenge geven. Mijn project x gebruikt ook eigen 3D objecten. Ik ga met deze challenge proberen een eigen 3D-object te importeren en op een website te krijgen. Vervolgens deze website op de HERA-hosting zetten en de link naar Maikel sturen.

Process eigen 3D-object in website

Tip van Maikel:
If you are going to load in your own 3D objects, use .gltf files and the GLTF loader - https://threejs.org/docs/?q=gltf#examples/en/loaders/GLTFLoader

Voor deze challenge heb ik gebruik gemaakt van AI, namelijk ChatGPt.

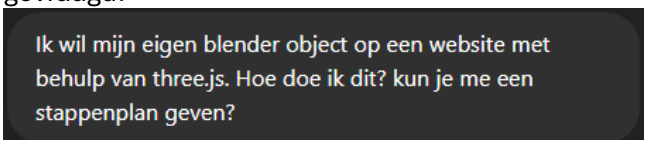
Het process

Ik begon met importeren van het Three.js framework. Met behulp van mijn notities en code die ik gemaakt heb tijdens de Bootcamp ging dit vrij makkelijk. Ik ben naar https://www.npmjs.com/package/three gegaan. En heb hier keek ik opnieuw naar hoe het geinstalleerd moet worden. Vervolgens kwam ik bij de onderste afbeelding.

Install

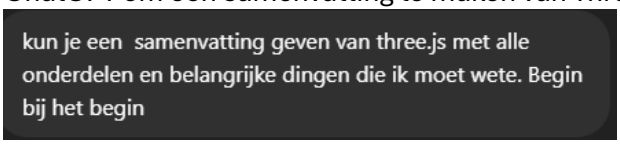
```
> npm i three
```

“npm i three” heb ik in mijn terminal in VS code ingevoerd. Deze installeerde een mapje met het framework. Vervolgens ben ik op https://threejs.org/manual/#en/installation gaan kijken hoe ik de indeling moet maken. Zo heb ik een index.html, style.css en main.js bestand aangemaakt en volgens de voorbeeldcode aangevuld.
Nadat ik dat had gedaan ben ik veel gebruik gaan maken van ChatGPT. Zo heb ik het volgende gevraagd:



Figuur 1: ChatGPT propt 1

Ik kreeg een duidelijk stappenplan hoe ik het kon aanpakken. De uitleg stapte ik voor een groot gedeelte. De bootcamp van Maikel was toen ik deze challenge aan het doen was al weer 3 weken geleden. Ik was een aantal dingen van Three.js vergeten. Daarom heb ik gevraagd aan ChatGPT om een samenvatting te maken van Three.js.

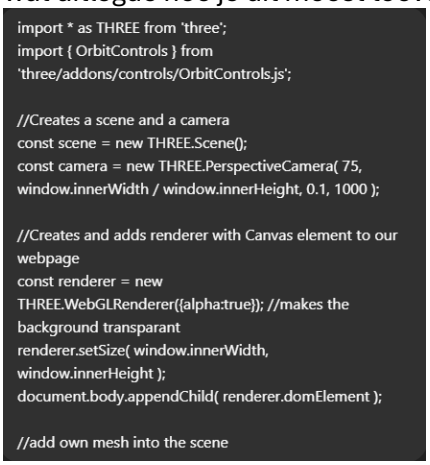


Figuur 2: ChatGPT propt 2

Deze samenvatting heb ik overgetypt in een word document zodat ik er altijd naar terug kan kijken.

Toevoegen eigen object

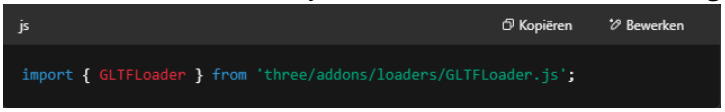
Nadat ik de basis code had voor mijn 3D omgeving heb ik niet naar de tip van Maikel gekeken, wat uitlegde hoe je dit moest toevoegen, maar heb ik het aan ChatGPT gevraagd.



```
import * as THREE from 'three';
import { OrbitControls } from
'three/addons/controls/OrbitControls.js';

//Creates a scene and a camera
const scene = new THREE.Scene();
const camera = new THREE.PerspectiveCamera( 75,
window.innerWidth / window.innerHeight, 0.1, 1000 );

//Creates and adds renderer with Canvas element to our
webpage
const renderer = new
THREE.WebGLRenderer({alpha:true}); //makes the
background transparant
renderer.setSize( window.innerWidth,
window.innerHeight );
document.body.appendChild( renderer.domElement );

//add own mesh into the scene
```

Figuur 3: ChatGPT propt 3

Zo kwam ik erachter dat je een GLTFLoader moet toevoegen.



```
js                                   Kopiëren   Bewerken

import { GLTFLoader } from 'three/addons/loaders/GLTFLoader.js';
```

Figuur 4: code GLTFLoader toevoegen

Ook hoe ik het object in de code gebruiken en zo laten zien op de website.

```js
const loader = new GLTFLoader();
loader.load(
  'model.glb', // pad naar jouw bestand
  function (gltf) {
    const model = gltf.scene;
    model.position.set(0, 0, 0); // optioneel
    scene.add(model);
  },
  undefined,
  function (error) {
    console.error('Fout bij het laden van model:', error);
  }
);
```

Figuur 5: code blender object importeren

ChatGPT gaf daarnaast ook een tip op OrbitControls toe te voegen, zodat je met de muis kunt draaien en zoomen.

```js
const controls = new OrbitControls(camera, renderer.domElement);
```

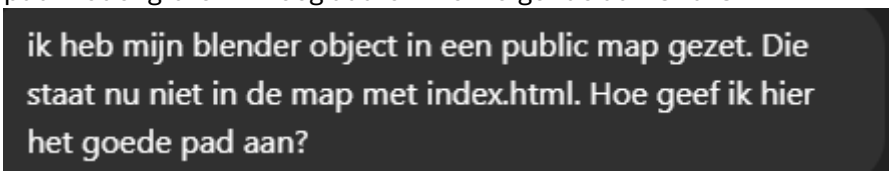
Figuur 6: code controls

Ook het toevoegen van een animatie.

```js
function animate() {
  requestAnimationFrame(animate);
  renderer.render(scene, camera);
}
animate();
```

Figuur 7: code animatie

Tijdens een ander project had ik moeite met in een code linken naar een ander bestand. Dit moet je doen door het goede pad te geven naar het bestand. Bijvoorbeeld in figuur 5 waarbij het pad model.glb is. Ik vroeg daarom het volgende aan ChatGPT.

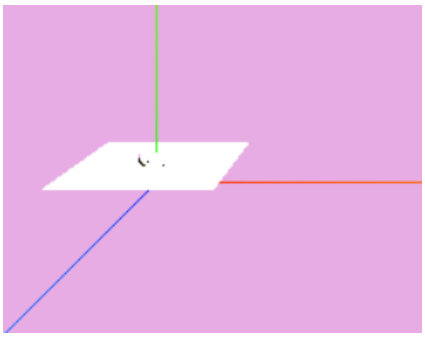


Figuur 8: ChatGPT propt 4

Zo gaf die een uitleg over het pad en de volgende tip:
Let op de / aan het begin – dit betekent: **laad vanaf de website-root**, en zoekt dus in de public map.

Live website

Na het oplossen van een paar kleine syntax fouten en vergeten symbolen ben ik gaan kijken hoe het object nu op de live website eruit ziet. In de afbeelding hieronder staat wat ik zag.



Figuur 9: Eerste render donut op website

Ik vroeg aan ChatGPT hoe dit kon. Dit had waarschijnlijk te maken met het ligt wat ik gebruikte. De code die ik gebruikte was het volgende:

```
//This is pointlight
const pointlight = new THREE.PointLight( 0xffffff, 1000, 100 ); //wit licht pointlight.position.set( 2,2,2 );
scene.add( pointlight );

//Atmospheric light (illuminates everything the same way)
const dirLight = new THREE.DirectionalLight(0xffffff, 1);
dirLight.position.set(5, 5, 5); scene.add(dirLight);
```

ChatGPT dacht dat het aan de kleur/materiaal van het object lag. Ik dacht dat het aan de licht was wat ik gebruikte. Ik heb het pointlight verwijderd, want ik dacht dat dit heel fel licht was. Nu ik erop terug kijk was de sterkte van het licht 100. Dit was veel te vel en dit verklaard waarom alles wit was.

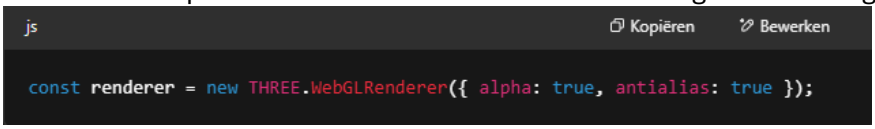
Scherpte en spikkels

Het zag er nu uit zoals in de afbeelding hieronder:

Figuur 10: Donut zonder spikkels en onscherp

Hierin staat ook wat ik precies aan ChatGPT heb gevraagd. De donut zag er niet scherp uit en de spikkels ontbreken.

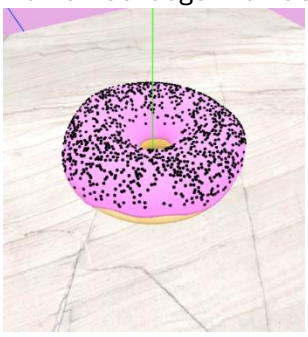
Voor het scherper maken van de donut moest ik het volgende toevoegen:



Figuur 11: code voor betere rederer

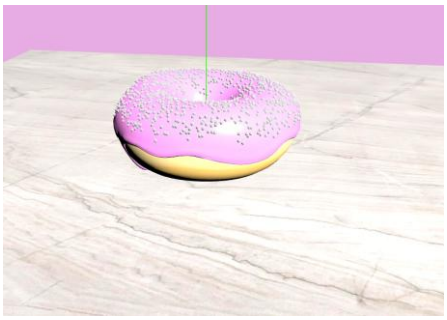
De spikkels toevoegen was iets ingewikkelder. Tijdens het maken van de donut heb ik in Blender gebruik gemaakt van Geometry nodes. Dit zijn notities die berekeningen en extra informatie aan een object toevoegen. Voor het maken van de spikkels heb ik hier gebruik van gemaakt. Dit was een makkelijker manier op dezelfde spikkel op meerdere plekken op de donut te laten voorkomen. Deze zouden dan op willekeurige plekken komen. Die spikkels waren 'echte' objecten. Ik heb ze omgezet naar een mesh (object). Hierdoor worden ze wel gezien door Three.js en dus wel ingeladen.

Goede kleur spikkels
Na het toevoegen van de spikkels zag het er zo uit.



Figuur 12: Donut met zwarte spikkels

De spikkels zijn in Blender wit.
Ik heb een paar aanpassingen gedaan met het licht. Ik wou een licht punt(een soort van lamp). Nu is het overal hetzelfde licht.



Figuur 13: Donut met grijze/witte spikkels

Ik denk dat ik de spikkels nooit een kleur heb gegeven en ze daarom zwart zijn in de figuur 12 en grijs- wit in figuur 13. In mijn geval is niet ook niet heel erg, want ik heb deze donut gemaakt om te oefenen met Blender en om te oefenen met three.js. Voor mijn eigen objecten is het wel handig om elk object een kleur te geven en om de Geometry nodes toe te passen op het object.

Online zetten
Tijdens het online zetten liep ik tegen een paar problemen, namelijk het pad in index.html naar main.js. Hiervoor moest geen / te staan, omdat het in dezelfde map staat, wat ik eerst wel had. Hieronder staat hoe het wel moet.

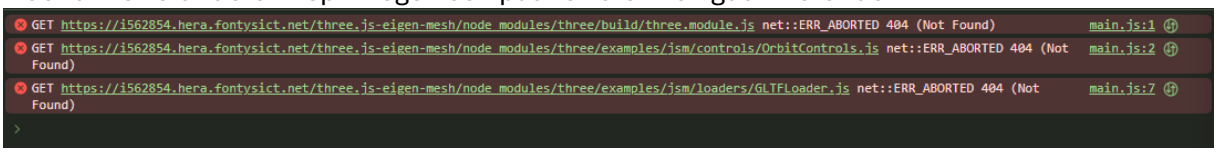
```html
<script type="module" src="main.js"></script>
```

Voor het online zetten waren er ook een paar bestanden die niet mee moesten, namelijk:

- node_modules
- package.json

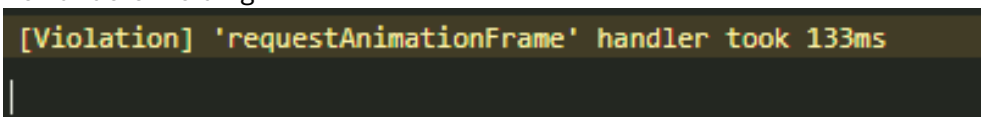
- package-lock.json

Door dit te veranderen liep ik tegen een paar errors. Zie figuur hieronder.



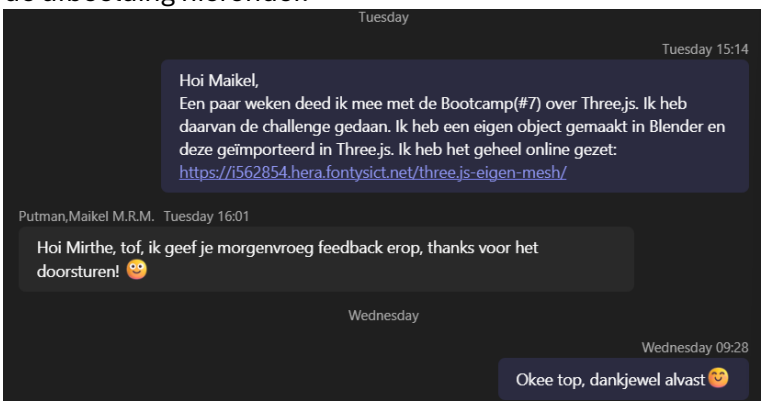
Hij kon node_modules map niet vinden. Dat klopt ook, want die moest niet mee online worden gezet. Daarom moest het vervangen worden voor een link die Three.js vanaf het internet laadt.

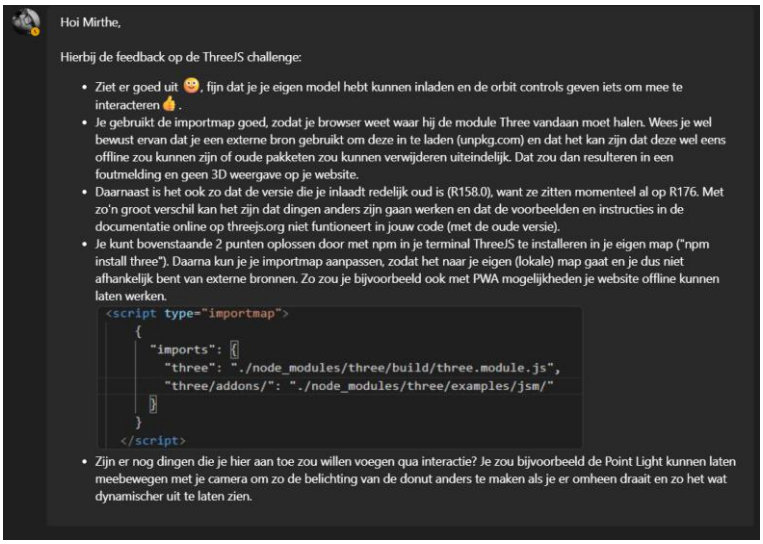
Een andere melding



De browser zegt:
“Hé, je animatie-frame duurt te lang, dus je app draait mogelijk niet vloeiend.”
Het is geen fout, maar een teken dat je animatie of model optimalisatie kan gebruiken.
Ik denk dat het komt omdat ik veel vlakken en punten op heel die donut heb. Dit moet die allemaal laden. Ook de dingen die niet zichtbaar zijn, zoals het onderste van de donut moet de browser laden, wat onnodig is. Voor mijn Project X objecten ga ik hier rekening mee houden.

Feedback Maikel
In de challenge stond dat je de challenge online moet zetten en dan vervolgens in een teams berichtje sturen naar Maikel met de link. Dit heb ik gedaan zodat ik feedback kon ontvangen. Zie de afbeelding hieronder.





Hoi Mirthe,

Hierbij de feedback op de ThreeJS challenge:

- Ziet er goed uit 😊, fijn dat je je eigen model hebt kunnen inladen en de orbit controls geven iets om mee te interacteren 👍.
- Je gebruikt de importmap goed, zodat je browser weet waar hij de module Three vandaan moet halen. Wees je wel bewust ervan dat je een externe bron gebruikt om deze in te laden (unpkg.com) en dat het kan zijn dat deze wel eens offline zou kunnen zijn of oude pakketen zou kunnen verwijderen uiteindelijk. Dat zou dan resulteren in een foutmelding en geen 3D weergave op je website.
- Daarnaast is het ook zo dat de versie die je inlaadt redelijk oud is (R158.0), want ze zitten momenteel al op R176. Met zo'n groot verschil kan het zijn dat dingen anders zijn gaan werken en dat de voorbeelden en instructies in de documentatie online op threejs.org niet funtioneert in jouw code (met de oude versie).
- Je kunt bovenstaande 2 punten oplossen door met npm in je terminal ThreeJS te installeren in je eigen map ("npm install three"). Daarna kun je je importmap aanpassen, zodat het naar je eigen (lokale) map gaat en je dus niet afhankelijk bent van externe bronnen. Zo zou je bijvoorbeeld ook met PWA mogelijkheden je website offline kunnen laten werken.

```
<script type="importmap">
    {
      "imports": {
        "three": "./node_modules/three/build/three.module.js",
        "three/addons/": "./node_modules/three/examples/jsm/"
      }
    }
</script>
```

- Zijn er nog dingen die je hier aan toe zou willen voegen qua interactie? Je zou bijvoorbeeld de Point Light kunnen laten meebewegen met je camera om zo de belichting van de donut anders te maken als je er omheen draait en zo het wat dynamischer uit te laten zien.

Samenvatting van de feedback (Feedpulse)

Na het volgen van de Three.js-bootcamp van Maikel ben ik aan de slag gegaan met de challenge. Hierbij heb ik in Blender een eigen 3D-model gemaakt en dit succesvol ingeladen in mijn Three.js-project.

<u>Positieve punten:</u>
-Ik heb met succes een eigen 3D-model in de scène geplaatst.
-Door het gebruik van OrbitControls is de scene interactief geworden.
-Ik maak correct gebruik van een importmap om Three.js in te laden.

<u>Aandachtspunten:</u>
-Ik laad Three.js momenteel in via unpkg.com, een externe bron.
Als deze bron offline gaat of verandert, werkt mijn app mogelijk niet meer.
Oplossing: Three.js lokaal installeren met npm install three om onafhankelijk te zijn van externe bronnen.

Ik gebruik momenteel Three.js versie R158, terwijl de huidige versie al R176 is.
Probleem: Dit kan leiden tot incompatibiliteit met recente documentatie of voorbeelden.
Oplossing: Updaten naar een recentere versie van Three.js.

<u>Suggestie voor verbetering:</u>
Om mijn scène dynamischer te maken, kan ik overwegen om het licht (bijv. PointLight) mee te laten bewegen met de camera, zodat de belichting verandert terwijl de gebruiker rond het object beweegt.

Verbetering na feedback

Na de waardevolle feedback van Maikel zijn er een paar dingen die ik moet gaan veranderen. Deze aandachtspunten ga ik verwerken en verbeteren. Daarnaast wil ik de suggestie om de scene dynamischer te maken en het licht mee te laten bewegen met de camera toepassen. Voor mijn persoonlijke project wil ik ook objecten animeren en wanneer je er over hovert dat die dan iets groter word.

<u>Lijst met aanpassingen</u>

- Three.js lokaal installeren (niet via een externe bron)
- Huidige versie van Three.js gebruiken
- Dynamischer maken met licht dat mee beweegt met camera
- Hover animatie (groter worden van het object)

Three.js versie en lokaal

Voor de versie van Three.js te updaten en lokaal te installeren heb ik hulp gevraagd aan ChatGPT. ChatGPT gaf me vervolgens een handig stappenplan.


Feedback van vorige keer Hierbij de feedback op de ThreeJS challenge:
Ziet er goed uit , fijn dat je je eigen model hebt kunnen inladen en de orbit controls geven iets om mee te interacteren.
Je gebruikt de importmap goed, zodat je browser weet waar hij de module Three vandaan moet halen. Wees je wel bewust ervan dat je een externe bron gebruikt om deze in te laden (unpkg.com) en dat het kan zijn dat deze wel eens offline zou kunnen zijn of oude pakketen zou kunnen verwijderen uiteindelijk. Dat zou dan resulteren in een foutmelding en geen 3D weergave op je website.
Daarnaast is het ook zo dat de versie die je inlaadt redelijk oud is (R158.0), want ze zitten momenteel al op R176. Met zo'n groot verschil kan het zijn dat dingen anders zijn gaan werken en dat de voorbeelden en instructies in de documentatie online op threejs.org niet funtioneert in jouw code (met de oude versie).
Je kunt bovenstaande 2 punten oplossen door met npm in je terminal ThreeJS te installeren in je eigen map ("npm install three"). Daarna kun je je importmap aanpassen, zodat het naar je eigen (lokale) map gaat en je dus niet afhankelijk bent van externe bronnen. Zo zou je bijvoorbeeld ook met PWA mogelijkheden je website offline kunnen laten werken.
Zijn er nog dingen die je hier aan toe zou willen voegen qua interactie? Je zou bijvoorbeeld de Point Light kunnen laten meebewegen met je camera om zo de belichting van de donut anders te maken als je er omheen draait en zo het wat dynamischer uit te laten zien.

Stappen	

Installeer Three.js lokaal met npm	bash
	`npm init -y`
	`npm install three`

Maak een mapstructuur zoals:	/project ├── /node_modules ├── /src │ ├── main.js └── index.html

In main.js:	js Kopiëren
	`import * as THREE from 'three';`
	`import { OrbitControls } from 'three/examples/jsm/controls/OrbitControls.js';`

Gebruik een bundler om je main.js te bundelen en te serveren. Bijvoorbeeld met Vite:	`npm install vite` `npx vite`

Licht meebewegen en animatie object

Voor het licht laten meebewegen met de camera en het groter maken van een object bij een hover heb ik hulp gevraagd aan ChatGPT. ChatGPT gaf me vervolgens een handige tips.

Ik maak een website met een 3D model, dit doe ik met het framework Three.js. Hoe zou ik dit voorwerp kunnen animeren dan wanneer je over het object hoverd die dan een klein beetje groter word? En hoe doe ik het dat de camera en licht met elkaar bewegen zodat je altijd goed licht hebt van de donut?

Groter worden van objecten wanneer je er over hoverd:

Stappen animatie Object vergroten	
Installeer GSAP	bash `npm install gsap`
Toevoegen import	`import { Raycaster, Vector2 } from 'three';` `import { gsap } from 'gsap';`

Voeg deze variabelen bovenaan toe	`const raycaster = new Raycaster();`
	`const mouse = new Vector2();`
	`let hoveredObject = null;`

Voeg deze eventlistener toe	`window.addEventListener('mousemove', (event) => {`
	`  mouse.x = (event.clientX / window.innerWidth) * 2 - 1;`
	`  mouse.y = -(event.clientY / window.innerHeight) * 2 + 1;`
	`});`

Pas je loader.load() functie aan om naar het model te verwijzen in animate():	`let donutModel;`
	
	`loader.load(`
	`  'donut4.glb',`
	`  function (gltf) {`
	`  donutModel = gltf.scene;`
	`  donutModel.position.set(0, 0, 0);`
	`  donutModel.scale.set(2, 2, 2);`
	`  scene.add(donutModel);`
	`  },`
	`  undefined,`
	`  function (error) {`
	`  console.error('Fout bij het laden van model:', error);`
	`  }`
	`);`

Toevoegen klik event:

Ik wou het nog meer uitbreiden met een klik event toe te voegen. Dat wanneer je op de donut klikt je een pop-up krijgt en deze dan weer kan sluiten. Dit vond ik heel makkelijk op de maken, want dit is heb ik al eens eerder gemaakt. De code kun je hieronder vinden.

```
<body>
  <div id="infoBox">
    <p>Je hebt de donut aangeklikt!</p>
    <button id="closeInfo">Sluiten</button>
  </div>
</body>
```

```
//klik event op donut
window.addEventListener('click', () => {
  if (hoveredObject) {
    document.getElementById('infoBox').style.display = 'block';
  }
});

document.getElementById('closeInfo').addEventListener('click', () => {
  document.getElementById('infoBox').style.display = 'none';
});

```

Licht met de camera:

```
const camera = new THREE.PerspectiveCamera(75, window.
camera.position.set(0, 0, 5);

const pointLight = new THREE.PointLight(0xffffff, 1);
pointLight.position.set(0, 0, 0); // Zet 'm op de camer
camera.add(pointLight); // Licht volgt nu de camera

scene.add(camera); // Vergeet niet de camera zelf toe t
```

Eerst voegde ik pointLight toe aan scene. Maar voor het volgen van de camera moest ik hem toevoegen aan camera dus camera.add(pointLight).

Na al dit hebben toegevoegd werkte de website lokaal. Toen ik hem online ging zetten laden hij niet goed in. De donut laden niet goed in. Ik heb hulp gevraagd aan Stan.

Hulp Stan

‘npx vite’ is hetzelfde principe als ‘npm run dev’. De server wordt dan lokaal gehost. Voor vite is dat http://localhost:5173/

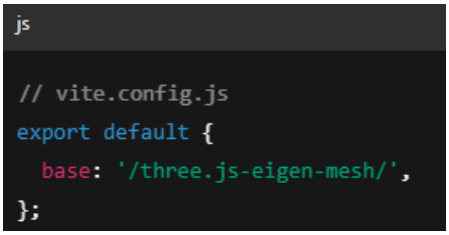
Dan draait die lokaal. Voor heb op een server te zetten moet je het bouwen. Dit doe je met ‘npx vite build’ of ‘npm run build’. Dan maakt die een map aan genaamd dist. Hierin is mijn hele code op de meest efficiënte manier.

Stan heeft in mijn code in package.json. het stukje onder debug toegevoegd. Voor het melden van een error en omdat mijn webite voor de live server nog in een mapje staat, namelijk three.js-eigen-mesh. Dat kan voor problemen zorgen. Daarom heeft hij via het opzoeken op stackoverflow de base toegevoegd in de code.

Weer online zetten

Nu ik een goede versie van three.js had en dat ik three.js kon meegeven in mijn code probeerde ik hem terug online te zetten. Vervolgens ging ik kijken of alles goed geladen was en dat was niet zo. Hij kon de donut niet vinden. Dit kwam omdat ik de donut importen met een verkeerd naam voor filezilla. Zo stond de donut in /public map en had ik dit eerst ook bij het pad gezet, maar dat moest niet. Het moest wel in /public map te staan, maar niet vernoemt in het pad om dat file te vinden.

Ik vroeg Chatgpt ook nog naar mijn foutmeldingen destijds en die zei dat ik de base ook nog moest toevoegen in mijn javascript.



Nadat ik het bovenstaande had gedaan en de website opnieuw in FileZilla had gezet werkt de website compleet. Hierna heb ik weer een update gegeven en feedback gevraagd aan Maikel

Feedback Maikel

Hallo Mirthe, bedankt voor de update en top dat je die dingen allemaal verbeterd hebt. Omdat je nu Vite gebruikt kan ik niet goed de broncode van de website meer beoordelen, aangezien deze wordt geminified (wat prima is, maar niet handig voor het bekijken ervan).

Het feit dat het werkt en de functionaliteiten ook goed te zien zijn, is dat in ieder geval goed gelukt 😄.

De melding die je krijgt, krijg ik zelf niet (ook niet na het langer open laten staan van de website. Het kan zijn dat het wellicht een issue is op je laptop (wellicht als je hem op een zuinige energiestand hebt staan). De requestAnimationFrame functie vraagt namelijk aan de browser om dit uit te voeren elke keer dat er een frame gemaakt moet worden, als dit langer duurt de het opbouwen van het volgende frame, dan kan het zijn dat er dingen nog niet goed verwerkt zijn voordat het frame wordt getekend. Vandaar de violation (vermoed ik). Het kan zijn dat door een lagere setting in de prestaties van je laptop (of je browser) dit zo is. Je zou hier eens met Stan van Oers naar kunnen kijken, hij is heel goed in het optimaliseren van deze omgeving.