EE3L11 Bachelor afstudeerproject

Project SUNRISE: Gebruikersinterface voor het e-bike oplaadstation

Groep D, Team HTML

Daniël Brouwer, 4288297
David Veselka, 4280199

15 juni 2016
Version 1.2

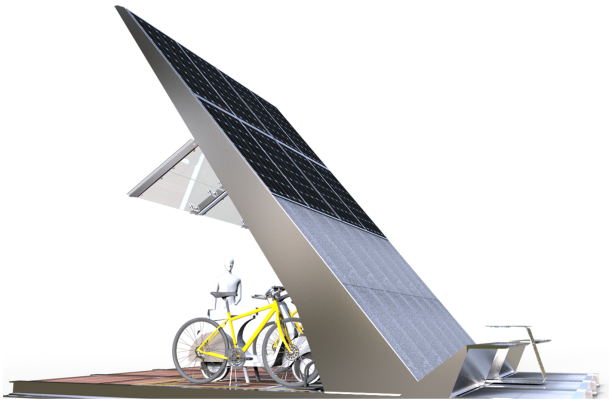
1 Samenvatting

De opkomst van e-bikes doet de vraag naar oplaadpunten sterk toenemen. Hierbij speelt duurzaamheid een grote rol in de stroomvoorziening van deze oplaadpunten. Hierom wordt er op de TU Delft campus een oplaadstation gebouwd die zijn energie voornamelijk van zonnepanelen ontvangt. Dit zal een proeftuin zijn voor gelijksoortige toekomstige systemen. Er is echter niet altijd de mogelijkheid om ter plekke een plaats in het oplaadstation te kunnen reserveren en gegevens te verkrijgen over het opladen van de fiets. Het doel van dit project is om een gebruikersvriendelijke interface te maken waarmee de gebruiker een oplaadplek kan reserveren en de laadstatus kan bijhouden. Er wordt gebruik gemaakt van HTML, CSS en JavaScript om een webpagina hiervoor te maken. In combinatie met een PHP-server die connectie heeft met het oplaadstation, kan via het gebruikersinterface statistieken worden uitgelezen over de laadstatus van de eigen fiets en van het lokale weer. Er werden verschillende webpagina's geïmplementeerd om de informatie overzichtelijk te maken. Op de server werden scripts geschreven om de webapplicatie te beveiligen en ervoor te zorgen dat alleen geauthoriseerde gebruikers van de oplaadservice gebruik kunnen maken. Hiermee is een systeem geïmplementeerd waarmee gebruikers op een gebruiksvriendelijke manier een oplaadplek kunnen reserveren en statistieken over het oplaadstation kunnen ontvangen.

Inhoudsopgave

2 Introductie

Dit project is een belangrijk deel van het e-bike laadstation dat op de TU Delft campus gebouwd gaat worden. Dit station bestaat uit vermogenselektronica om vermogen tussen de zonnepanelen, buffer-accu's, het lichtnet en de aangesloten tweewielers te verdelen. Het station is in staat om zowel bedraad als draadloos e-bikes op te laden. Ook kunnen er e-scooters worden opgeladen. Een weerstation is inbegrepen om het weer te monitoren. Ook wordt met temperatuursensoren de temperatuur van de zonnepanelen in de gaten gehouden. Het concept van het laadstation is te zien in figuur 1.



Figuur 1: Concept van het laadstation

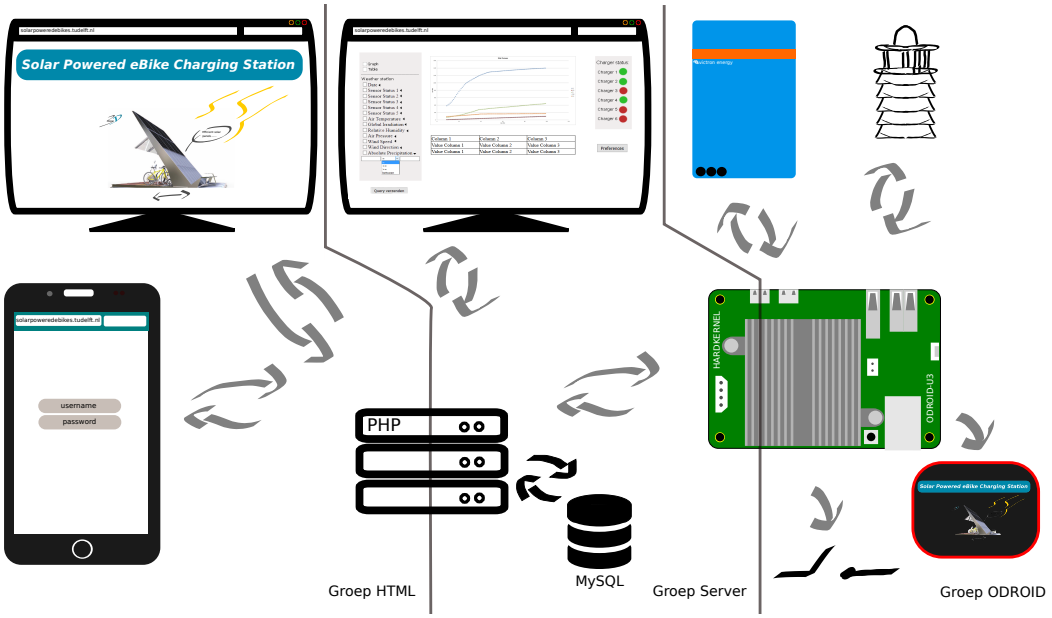
Het SUNRISE project (Smart Unified Networking Rig for an Integrated Solar Ebike charger) was voorgesteld als bachelor eindproject. Het eindresultaat van dit project zal gebruikers de mogelijkheid geven een laadplek voor de elektrische tweewieler te reserveren. De geschreven software zal communiceren met de server die de gebruikersaccounts en de beschikbare oplaadplekken beheert. In dit project zal ingegaan worden op het ontwerpen van een gebruikersinterface welke gerealiseerd wordt door een webapplicatie en een website waarop algemene informatie beschikbaar komt over het gehele project. Een overzicht van het volledige project is te zien in figuur 2.

2.1 State-of-the-art analysis

Al wordt in dit project een geheel niewe software ontwikkeld, er zijn een aantal dingen die beschikbaar zijn gesteld en waarop voorgebouwd wordt. Het gehele oplaadstation met aansturingselektronica is ontwikkeld en wordt beschikbaar gesteld. Ook is een webserver beschikbaar gesteld die een PHP programmeeromgeving en een phpMyAdmin database bevat. Verder is er gebruikgemaakt van de jQuery bibliotheek die JavaScript functies bevat die het programmeren in JavaScript vergemakkelijkt [1]. Om interactieve driedimensionale beelden te kunnen tonen is gebruik gemaakt van WebGL [2]. Hierbij worden matrixbewerkingen op objecten toegepast om ze te laten roteren, schalen en transleren. Om dit te vergemakkelijken is gebruik gemaakt van een bibliotheek die deze matrixoperaties uitvoert door simpele functies aan te roepen [3]. De objecten die weergegeven worden in de WebGL omgeving worden ontworpen in Blender. Hiermee is het mogelijk om 3D-modellen te modelleren en te voorzien van textures die gebruikt worden in de WebGL omgeving [4]. Om door middel van een kaart op de informatiewebsite de locatie van het oplaadstation weer te geven, wordt Open Streetmap in combinatie met Leaflet gebruikt [5].

2.2 Probleemdefinitie

Oplaadsystemen zijn al beschikbaar, zowel bij mensen thuis als in het openbaar. Wat een probleem vormt is de software interface voor de gebruiker: er wordt bij de huidige oplossingen geen informatie



Figuur 2: Systeemoverzicht van het gehele project

naar de gebruiker gecommuniceerd over de laadstatus van de elektrische fiets en de verwachtte actieradius. Ook kunnen er geen statistieken worden bijgehouden over het totale geladen vermogen over verschillende laadbeurten. Verder is er geen simpele manier om een laadplek ter plekke te reserveren via de smartphone. Kort geformuleerd gaat er in dit project een oplossing gevonden worden voor dit probleem: Hoe kan er een gebruikersvriendelijke oplossing gevonden worden om bezitters van elektrische fietsen gebruik te laten maken van het duurzame oplaadstation?

2.3 Overzicht

Deze thesis zal starten met het programma van eisen (3). Daarna zal in sectie 4 ingegaan worden op het ontwerpproces en zullen de verschillende componenten in afzonderlijke secties worden besproken: de WebGL modelviewer, de reserveerapplicatie en de mediawebsite (sectie 4.1, 4.2 en 4.3). Hierop volgt de implementatie in sectie 5 waarin de methoden worden uitgelicht waarmee het ontwerp geimplementeerd is. Ook hier zullen de verschillende onderdelen apart toegelicht worden (sectie 5.1, 5.2 en 5.3). Hierna zullen de resultaten worden besproken van de aparte onderdelen in sectie 6 en zal worden afgesloten met de conclusie en evaluatie in sectie 7 en 8.

3 Programma van eisen

Voor het e-bike oplaadstation moet een end-user interface gemaakt worden. Het interface moet bestaan uit twee delen: een HTML5 reserveerapplicatie en een algemene mediawebsite. De reserveerapplicatie moet het voor de gebruiker mogelijk maken om een plek in het oplaadstation te kunnen reserveren en de status van het opladen te volgen. De mediawebsite is bedoeld voor promotiedoeleinden en zal relevante informatie bevatten over het project. Voor de end-user interface zijn een aantal eisen gesteld waaraan voldaan dient te worden. De eisen zijn op te delen in, eisen voor de HTML5 applicatie en de algemene website. Vervolgens zijn deze eisen weer opgedeeld in functionele eisen en systeem eisen. De functionele eisen bepalen welke functionaliteiten het systeem dient te hebben gebaseerd op de probleemdefinitie. De systeemeisen zijn extra eisen die niet direct bijdragen aan de algemene functionaliteit van het systeem, maar desalniettemin benodigd zijn voor een gebruiksvriendelijke werking van het systeem.

Functionele eisen van de HTML5 applicatie

[1.1.1] Met de applicatie moet het mogelijk zijn om een plek te reserveren in het oplaadstation.

[1.1.2] De applicatie moet de opladers uit en aan kunnen zetten om aan eis [1.1.1] te kunnen voldoen.

[1.1.3] De applicatie moet middels een inlogsysteem geautoriseerden gebruikers toelaten en onbevoegden weren.

[1.1.4] De applicatie moet algemene data kunnen laten zien van het oplaadstation zoals de temperatuur.

Systeemeisen van de HTML5 applicatie

[1.2.1] De applicatie dient deels als promotie van de oplaadpaal en moet daarom gebruiksvriendelijk zijn en er goed uitzien.

[1.2.2] De applicatie moet op zowel smartphones en tablets als een PC kunnen draaien en dus goed schalen.

[1.2.3] De applicatie moet ook op een simpele netbook of eenvoudige smartphone kunnen draaien.

Functionele eisen van de mediawebsite

[2.1.1] Op de hoofdpagina met een driedimensionaal model komen van de oplaadpaal ter promotie van de paal.

[2.1.2] op de website moet de locatiegegevens komen van de oplaadpaal.

[2.1.3] De algemene website moet interessante informatie bevatten van het oplaadstation dat ook te begrijpen is voor niet-technische mensen.

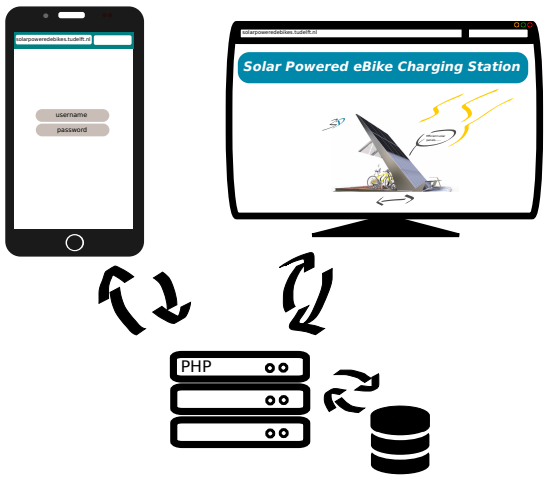
Systeemeisen van de mediawebsite

[2.2.1] De mediawebsite dient er aantrekkelijk uit te zien. Het doel van de website is namelijk op de oplaadpaal te promoten.

[2.2.2] De mediawebsite moet snel te overzien zijn zonder al te veel tekst zodat de gebruiker geïnteresseerd blijft.

4 Het ontwerpproces

Om te voldoen aan de eisen zal er een ontwerp gemaakt moeten worden. De eerste stap is een globaal overzicht van het systeem. In figuur 32 in appendix A is het systeem te zien. Centraal in het systeem staat de server die zal worden gebruikt om de HTML5 pagina's te personaliseren en te vullen met relevante data. Hierna wordt de pagina verzonden naar de gebruiker.



Figuur 3: Globaal overzicht van het systeem

Voor de opbouw van de pagina's zal HTML5 gebruikt worden. Tegenwoordig hebben alle bekende browsers support voor HTML5 en ook smartphones zijn grotendeels compatibel met het formaat. Desalniettemin zullen niet alle onderdelen (even goed) ondersteund worden en zal hiermee rekening gehouden moeten worden.

Om de communicatie tussen de eindgebruiker en de server te bewerkstelligen zal gebruik gemaakt worden van PHP. De PHP-code die geschreven wordt draait op de server en is ook alleen daar zichtbaar. Hier wordt een loginsysteem mee gemaakt waardoor het voor de gebruiker mogelijk is een oplaadplek te reserveren en voldoet het systeem aan de eisen: [1.1.1], [1.1.2] en [1.1.3]. De gebruiker registreert van tevoren een gebruikersnaam en wachtwoord welke in de SQL database van de server komen te staan. Hierna kan worden ingelogd met de toegewezen naam en wachtwoord en een oplaadplek worden gereserveerd. Eis [1.1.4] specificeert om enkele statistieken over het weer in de app te tonen. Dit wordt gedaan met een PHP-script die deze statistieken uitleest uit de SQL database. De uitgelezen waarden worden vervolgens getoond op de pagina.

De interface moet goed schalen op verschillende resoluties en schermgroottes om te voldoen aan eis [1.2.2]. Hiervoor wordt gebruik gemaakt van CSS 3. Met CSS is het mogelijk om de schaling van de verschillende webelementen aan te passen aan de grootte en resolutie van het scherm. Ook kan hiermee de indeling en opmaak van de pagina worden aangepast.

Om aan de eisen [2.1.1] en [2.2.1] te voldoen is er gekozen om gebruik te maken van een HTML5 canvas om grafische content weer te geven. In het HTML5 canvas kan gekozen worden voor twee verschillende opties voor het laten zien van grafische elementen: 2D canvas of een 3D webGL canvas. Beide opties hebben voordelen en nadelen.

2D canvas
Voordelen:

- Vrij gemakkelijk te implementeren

- Werkt op vrijwel alle apparaten

Nadelen:

- Gelimiteerde mogelijkheden, voor driedimensionale graphics is het niet geschikt

WebGL canvas
Voordelen:

- Mogelijkheid tot driedimensionale graphics
- Support is redelijk goed voor huidige browsers

Nadelen:

- Lastig te implementeren
- Zal niet op elk apparaat werken (bijvoorbeeld Microsoft Internet Explorer 8 of lager)

Omdat een 2D canvas niet toereikend is om aan de eis [2.1.1], met het 3D model van de oplaadpaal, te voldoen zal voor de mediawebsite gebruik gemaakt worden van WebGL.

Voor de eis [1.2.2], van de reserveerapplicatie, zal een 2D canvas het meest toereikend zijn omdat deze een grotere support heeft voor meerdere apparaten (bijvoorbeeld bij Android al vanaf versie 4.2 in de standaard browser). WebGL daarentegen is voor Android bijvoorbeeld pas na versie 5.0 standaard ondersteund. Daarbij moet wel gezegd worden dat voor versies voor 5.0, er wel alternatieve browsers te downloaden zijn die het wel ondersteunen. Voor iOS maakt het niet uit of er gebruik gemaakt wordt van een 2D canvas of webGL omdat beide mogelijkheden sinds iOS 8.1 te gebruiken zijn.

Ondanks dat een 2D canvas voor de mobiele applicatie voldoende zou zijn, zal hiervoor ook gebruik gemaakt worden van WebGL omdat de functies van het WebGL canvas op de mediasite dan weer hergebruikt kunnen worden in de app. Tevens zal er met driedimensionale grafische mogelijkheden, de mobiele app er nog mooier uitzien en dit is in overeenstemming met eis [1.2.1]. Bovendien maakt WebGL intensief gebruik van de grafische processor en daardoor zal het ook goed werken op tragere apparaten wat eis [1.2.3] ten goede doet.

4.1 Ontwerp van de modelviewer

Voor de WebGL implementatie is gekozen voor een apart zogenoemd Modelviewer script zodat deze gemakkelijk voor zowel de mediawebsite als de app te gebruiken is. Om er voor te zorgen dat de modelviewer de juiste functionaliteiten bezit voor zowel de app als de mediawebsite, zijn er voor de modelviewer een aantal eisen opgesteld:

[1.1] Modellen gemaakt in een 3D programma moeten kunnen worden geïmporteerd inclusief de textures.

[1.2] Modellen moeten geroteerd, getransleerd en geanimeerd kunnen worden.

[1.3] De viewer moet in staat zijn om ook text op het scherm te kunnen printen.

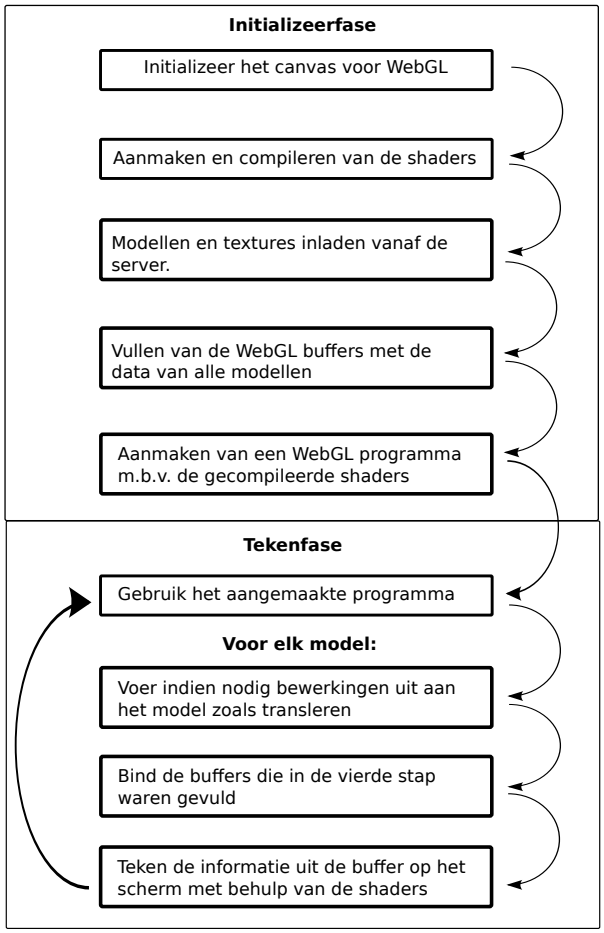
[1.4] Het script moet onafhankelijk van de media of app pagina te gebruiken zijn.

[1.5] Modellen moeten shading (bijvoorbeeld Phong) krijgen.

Om met behulp van WebGL een ontwerp te maken die aan de bovenstaande eisen voldoet moet er eerst een begrip worden gevormd van wat WebGL precies is en hoe het gebruikt moet worden.

4.1.1 WebGL

WebGL maakt het mogelijk om via OpenGL in de webbrowser grafische content weer te geven. WebGL maakt gebruik van de videokaart waardoor graphics snel weergegeven kunnen worden met weinig overhead voor de processor. Om content te tekenen maakt WebGL gebruik van een vertex(4.1.2) en fragment(4.1.3) shader. WebGL bestaat uit een verzameling van OpenGL functies en het is aan de ontwikkelaar welke functies er worden geïmplementeerd of gebruikt. Wel is het handig om bijvoorbeeld de buffers niet te vaak met nieuwe data te vullen omdat dit een dure operatie is. In figuur 4 staat een ontwerp van de werkwijze die zal worden gebruikt voor dit project.



Figuur 4: WebGL werkwijze

4.1.2 Vertexshader

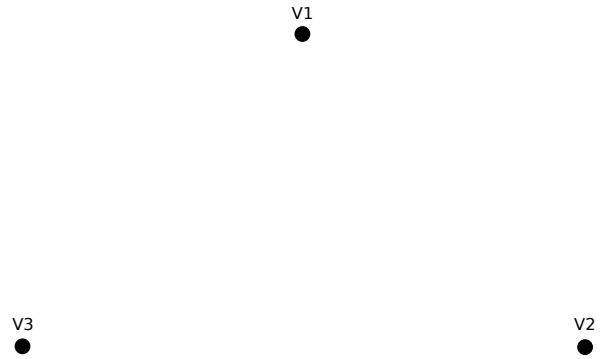
De vertexshader wordt gebruikt om punten / vertices te tekenen. Een vertexshader bestand kan er als volgt uitzien: (zie script 1).

Script 1: Voorbeeld van een vertexshader

```
1 attribute vec3 position;
2 varying vec3 fragPosition;
3 uniform mat4 worldMatrix;
4 uniform mat4 viewMatrix;
```

```
uniform mat4 projMatrix;
void main()
{
  fragPosition = (worldMatrix * vec4(position,1.0)).xyz ;
  gl_Position = projMatrix * viewMatrix * worldMatrix * vec4 (position,1.0) ;
}
```

Elke vertex passeert de vertexshader waarin een aantal berekeningen gedaan kunnen worden. In het voorbeeld is er één "attribute" genaamd "position" waarmee de positie van een vertex aangegeven wordt. De waarde van een "attribute" is gebonden aan een vertex en verschilt daarom ook per vertex. Een "varying" is een variabele die aan de fragmentshader(4.1.3) doorgegeven kan worden, na eventuele bewerkingen in de vertexshader. "Uniforms" zijn variabelen die alleen kunnen veranderen per getekend object. In dit voorbeeld zijn er drie "uniforms" van het type mat4: een worldmatrix, viewmatrix en projmatrix. Deze "uniforms" stellen 4x4 matrices voor die nodig zijn om de positie, schaal en rotatie van objecten en de camera te definiëren meer hierover in subsectie 4.1.7. Voor iedere vertex die de vertexshader passeert zal de fragmentshader aangeroepen worden die de kleur van de vertex bepaald.



Figuur 5: Resultaat na de vertexshader

4.1.3 Fragmentshader

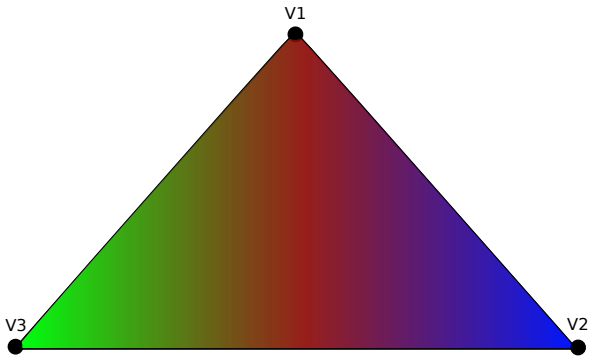
De fragmentshader wordt gebruikt om kleur te geven aan objecten. Een fragmentshader bestand kan er als volgt uitzien: (zie figuur 2).

Script 2: Voorbeeld van een fragmentshader

```
precision mediump float;
varying vec2 fragTextureCoord;
varying vec3 fragNormal;
uniform sampler2D sampler;

void main() {
//ambient parameters
vec3 ambientIntensity = vec3(0.8,0.8,0.8);
float ambientReflectionConstant = 1.0;

//diffuse parameters
vec3 sunIntensity = vec3(0.384,0.382,0.346);
vec3 sunDirection = normalize(vec3(1.0,5.0,0.0));
float diffuseReflectionConstant = 1.0;


vec4 texel = texture2D(sampler,fragTextureCoord);

//phong shading:
vec3 lightIntensity = ambientIntensity*ambientReflectionConstant + ↩
    diffuseReflectionConstant*(dot(fragNormal,sunDirection)*sunIntensity);
```

```
21
22 //resulting fragcolor:
23 gl_FragColor = vec4(texel.rgb * lightIntensity,texel.a);
24 }
25
26 }
```

In de fragmentshader kunnen net als in de vertexshader berekeningen gedaan worden alleen nu zijn het geen berekeningen op de positie van een vertex, maar de kleur van een vertex. In het voorbeeld wordt een texture gemapt op het model en daarbij word ook nog eens phongshading gebruikt om de licht weerkaatsing afhankelijk te maken van materiaaleigenschappen en lichtbronnen in de scene. De eerste regel in het voorbeeld geeft aan met hoeveel precisie de videokaart moet rekenen. In dit voorbeeld wordt medium precisie gebruikt. De twee vec3 vectoren van het type varying komen vanuit de vertexshader en geven respectievelijk het texturecoördinaat (4.1.5) aan en de richting van de normaal. Met behulp van de normaal en de ambient- en diffusieparameters kan phong shading toegepast worden om de lightintensiteit van elke vertex te berekenen. In het voorbeeld wordt de uitkomst van de berekening in een vec3 gestopt en hierin bevinden zich de rgb waardes van de kleur. Als laatst wordt de texture vermenigvuldigt met de berekende lichtintensiteit zodat nu het object een texture bevat met ambient en diffusie reflectie.



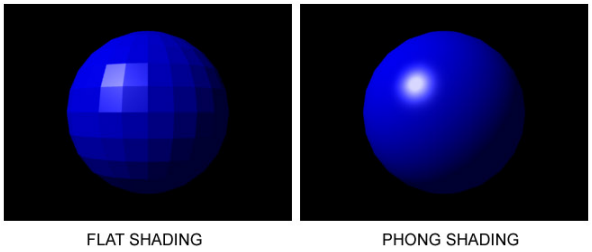
Figuur 6: Resultaat na de fragmentshader

4.1.4 Phong Shading

Om licht reflecties te kunnen simuleren kan gebruik gemaakt worden van een aantal functies waaronder: BRDF(Bidirectional Reflectance Distribution Function) of de BSSRDF (Bidirectional scattering-surface reflectance distribution function). In het kort kijkt BRDF alleen naar de hoek van het invallende licht en golflengte en daaruit wordt de gereflecteerde lichtstraal berekent. De BSSRDF is veel uitgebreider en houd ook rekening met scattering, waarbij licht een object deels of geheel binnendringt waardoor het invallende licht meerdere reflecties tot gevolg kan hebben. Omdat de tweede optie veel rekenkracht vergt en eigenlijk geen extra toegevoegde waarde geeft zal voor de eerste optie gekozen worden.

De BRDF functie is op verschillende manieren geïmplementeerd, de simpelste implementaties zijn:

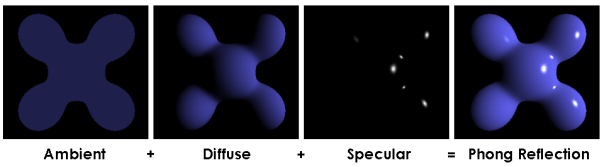
- Flat shading
- Gouraud shading
- Phong shading



Figuur 7: Verschil tussen flat en phong shading. Uit Wikimedia Commons [6]

Voor het project is gekozen voor het Phong shading model omdat de berekening ervoor relatief weinig rekenkracht vergt en het resultaat beter is dan Gouraud of flat shading (zie figuur 7).

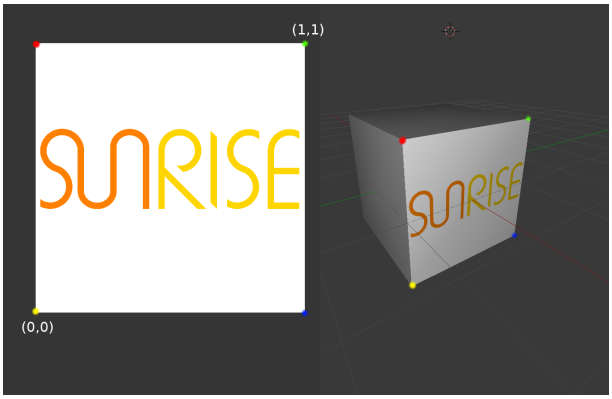
Phong-shading bepaald de kleur door interpolatie van de normaalvectoren aan een vlak en het Phong reflectie model (zie figuur 8). Voor het project zal alleen de Diffusie en Ambient term worden meegenomen in de berekening en zal Specular reflectie niet worden geïmplementeerd. Het is namelijk niet het doel om fotorealistische graphics te creëren, zeker gezien eis [1.2.3] zal dit onnodig extra rekenkracht vergen.



Figuur 8: Phong shading met Ambient, Specular en Diffusie term. Uit Wikimedia Commons [7]

4.1.5 Textures

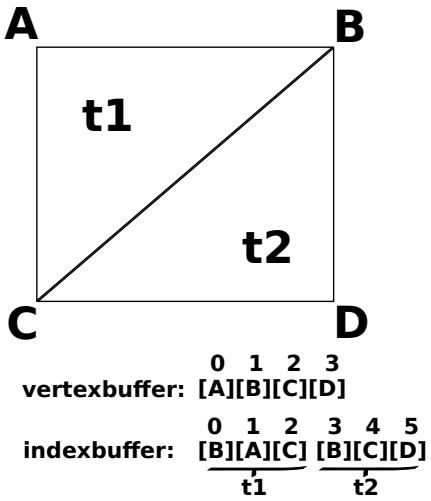
Textures zijn afbeeldingen die geprojecteerd worden op een (3D) model. Om te kunnen projecteren is een assenstelsel nodig om de positie op de texture te kunnen bepalen. WebGL gebruikt een cartesisch coördinatenstelsel dat loopt van (0,0) tot (1,1). In figuur 9 staat een voorbeeld waarbij een afbeelding wordt geprojecteerd met behulp van de vier vertices waaruit een vlak van de kubus bestaat.



Figuur 9: Het projecteren van een afbeelding op een kubus met behulp van de vertices

4.1.6 Vertices, indices en normals

De grafische modellen zullen bestaan uit vertices, indices en normals. De vertices specificeren de locaties van elk punt, de indices welke vertices een driehoek vormen en de normals, de normalen. Figuur 10 laat dit met een voorbeeld zien.



Figuur 10: Voorbeeld van vertices en indices bij een rechthoek bestaande uit twee driehoeken

4.1.7 World, view en projection matrices

Voor het weergeven van 3D modellen moeten de posities van de modellen en de camera goed gedefinieerd zijn. Met behulp van de drie matrices: world, view en projection wordt dit behaald.

De vertices van een model bestaan uit een lijst van coördinaten. Deze coördinaten specificeren de vertices in relatie tot het model. De worldmatrix transformeert deze coördinaten naar de zogenoemde "world space". De worldmatrix specificeert dus de positie van het model in de driedimensionale omgeving.

Na de worldmatrix komt de view matrix transformatie. De view matrix specificeert waar het model zich bevindt ten opzichte van het oog van de camera.

Als laatste is er nog een projectiematrix. De projectiematrix specificeert hoe de driedimensionale scene op het tweedimensionale beeld van de camera geprojecteerd moet worden en specificeert daarmee dus in wezen het type camera.

Alledrie de matrices zijn 4x4 groot. Dit is nodig omdat een 3x3 matrix niet in staat is om driedimensionale translaties mee voor te stellen. Translatie is namelijk anders dan rotatie en schaling geen lineaire operatie. Dit concept valt beter te begrijpen met behulp van een voorbeeld. Als men een punt/vertex gedefinieerd door de vector (1) wilt transleren en schalen met behulp van de 3x3 worldmatrix uit (2) zal alleen schaling mogelijk zijn. Vergelijking (3) laat het resulterende punt zien nadat de vector uit (1) is vermenigvuldigd met de matrix uit (2). Te zien is dat de schaling operatie op x_1, y_1 en z_1 gedaan worden door de elementen a_1, b_2 en c_3 uit de matrix. Om translatie mogelijk te maken zou vergelijking (3) uitgebreid moeten worden naar vergelijking (6). Om tot deze vergelijking te komen moeten er twee veranderingen gemaakt worden. Aan iedere driedimensionale vector moet een vierde element worden toegevoegd met de waarde 1 (4) en de matrix uit (2) moet worden uitgebreid tot een 4x4 matrix als in (5). Wanneer men nu de vector (4) vermenigvuldigd met de matrix (5) zal dit leiden tot het resultaat in vergelijking (6) en

daarmee is het dus mogelijk om een punt te schalen en te transleren.

$$v_1 = \begin{pmatrix} x_1 \\ y_1 \\ z_1 \end{pmatrix} \tag{1}$$

$$m_1 = \begin{pmatrix} a_1 & a_2 & a_3 \\ b_1 & b_2 & b_3 \\ c_1 & c_2 & c_3 \end{pmatrix} \tag{2}$$

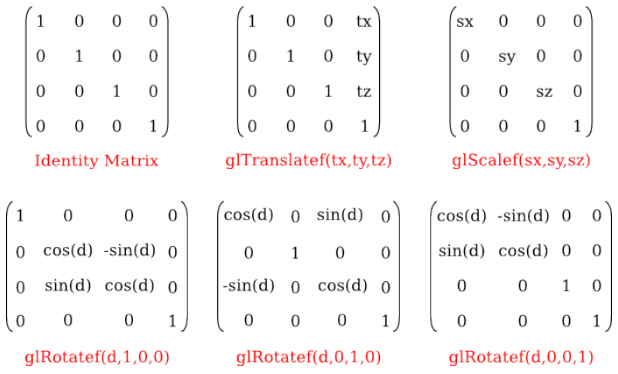
$$\begin{aligned} x_2 &= a_1 * x_1 + a_2 * y_1 + a_3 * z_1 \\ y_2 &= b_1 * x_1 + b_2 * y_1 + b_3 * z_1 \\ z_2 &= c_1 * x_1 + c_2 * y_1 + c_3 * z_1 \end{aligned} \tag{3}$$

$$v_2 = \begin{pmatrix} x_1 \\ y_1 \\ z_1 \\ 1 \end{pmatrix} \tag{4}$$

$$m_2 = \begin{pmatrix} a_1 & a_2 & a_3 & d_1 \\ b_1 & b_2 & b_3 & d_2 \\ c_1 & c_2 & c_3 & d_3 \\ 0 & 0 & 0 & 1 \end{pmatrix} \tag{5}$$

$$\begin{aligned} x_2 &= a_1 * x_1 + a_2 * y_1 + a_3 * z_1 + d_1 \\ y_2 &= b_1 * x_1 + b_2 * y_1 + b_3 * z_1 + d_2 \\ z_2 &= c_1 * x_1 + c_2 * y_1 + c_3 * z_1 + d_3 \end{aligned} \tag{6}$$

Rotaties gaan op een vergelijkbare wijze, alleen zullen er dan ook andere elementen dan de schaal operatoren: a_1, b_2 en c_3 uit de matrix worden gebruikt. Figuur 11 geeft de mogelijke transformaties weer.



Figuur 11: Mogelijke transformaties voor de 4x4 matrices. Uit het boek: Introduction to Computer Graphics [8]

4.2 Ontwerp van de reserveer applicatie

Als de gebruiker zijn fiets wil opladen in het oplaadstation, kan via een webapplicatie een oplaadplek gereserveerd worden. De gebruiker dient in te loggen met de vooraf geregistreerde login en een plek te kiezen in het station om de fiets op te laden. Deze app is via de webbrowser op een smartphone te bezoeken en wordt geschreven met HTML, CSS, PHP en JavaScript.

4.2.1 PHP

Om ervoor te zorgen dat de eindgebruiker de juiste informatie op zijn scherm krijgt, wordt gebruik gemaakt van een server met behulp van PHP scripts in combinatie met een MySQL database. Er zijn een aantal eisen gesteld waaraan het loginsysteem moet voldoen. Hieronder vallen ook eisen die zelf zijn gesteld om het programma van eisen concreter te maken:

[1.1] De server moet gebruikersinvoer kunnen verwerken die ingevoerd wordt via de gebruikersinterface.

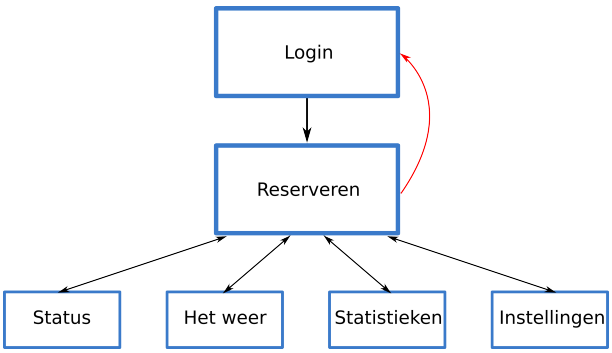
[1.2] De server moet kunnen reageren op gebruikersinvoer en de gewenste informatie terugsturen.

[1.3] De server moet het mogelijk maken voor de gebruiker om een oplaadplek te reserveren.

[1.4] Het systeem moet beveiligd zijn tegen aanvallers en hackers.

Om het mogelijk te maken dat de eindgebruiker een oplaadplek kan reserveren zonder dat kwaadwillenden hier misbruik van kunnen maken, gaat gebruik gemaakt worden van een loginsysteem. Hiermee kan voorkomen worden dat niet-geauthoriseerde personen een oplaadplek kunnen reserveren en wordt rekening gehouden met eis [1.3] en [1.4].

De richting van het uitvoeren van scripts is te zien in figuur 12.



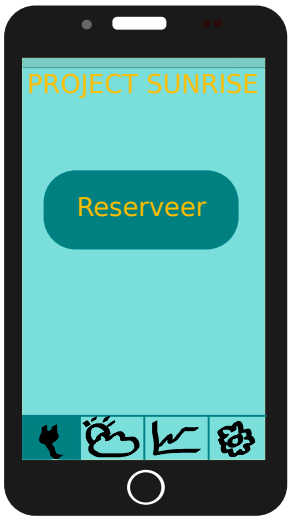
Figuur 12: structuur van het PHP inlogsysteem

Als de gebruiker voor het eerst het interface bezoekt komt deze terecht bij het loginscherm. Hier kan worden ingelogd met de vooraf geregistreerde gegevens. Nadat er is ingelogd heeft de gebruiker de mogelijkheid om een plek in het oplaadstation te reserveren. Indien er met een administratoraccount wordt ingelogd, wordt doorgelinkt naar het administrator dashboard. Verder zijn er vier pagina's die vanaf deze pagina kunnen worden opgevraagd. Op de statuspagina wordt weergegeven of de gebruiker een elektrische tweewieler aan het opladen is en wat het laadniveau is. Op de weerpagina worden statisieken over het weer getoond welke uit de MySQL database worden gehaald. Statistieken over vorige laadbeurten zijn weer te geven op de statistiekenpagina. In de instellingen kunnen persoonlijke voorkeuren worden aangegeven.

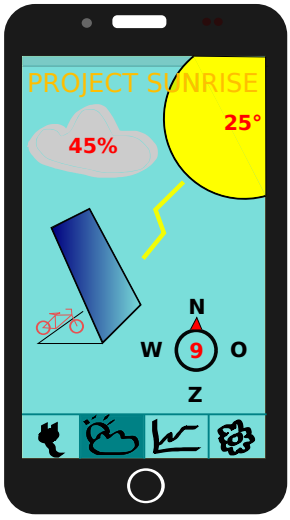
4.2.2 Visueel ontwerp

Voor het ontwerp van het visuele interface met de gebruiker is gekozen voor een centraal inlogscherm als de gebruiker voor het eerst de pagina bezoekt. Na het inloggen zijn er centraal op het scherm knoppen om een oplaadplek te reserveren. Deze staan midden in het scherm zodat het voor de gebruiker duidelijk te zien is. Het ontwerp is te zien in figuur 13. Dit is namelijk ook de primaire functie van de app. Onderin is een menu met vier knoppen voor vier verschillende pagina's. De eerste pagina is de pagina die de gebruiker direct na het inloggen te zien krijgt. Als er al een oplaadplek gereserveerd is kan de gebruiker de status van het opladen zien. Hier is te zien hoe vol de batterij inmiddels is en hoe lang het duurt voordat de gebruiker weer weg kan

rijden met een volle batterij. De tweede pagina geeft toegang tot een weeroverzicht. Hier kunnen bepaalde weergegevens op een gebruikersvriendelijke manier bekeken worden, zoals windsnelheid, temperatuur en relatieve luchtvochtigheid. Om het overzicht er levendig te laten uitzien wordt gebruik gemaakt van WebGL. Het ontwerp is te zien in figuur 14. In de derde pagina staan gegevens over het oplaadgedrag van de gebruiker. Hier kan deze onder andere zien hoeveel kilometer er gefietst kan worden op de huidige acculading en hoe vaak er al van het oplaadstation gebruikgemaakt is. De data, die afkomstig is van de server, wordt hier in grafieken weergegeven zoals te zien is in het ontwerp in figuur 15. Als laatste is er een pagina met instellingen, waaronder de optie om statistieken te ontvangen en uit te loggen.

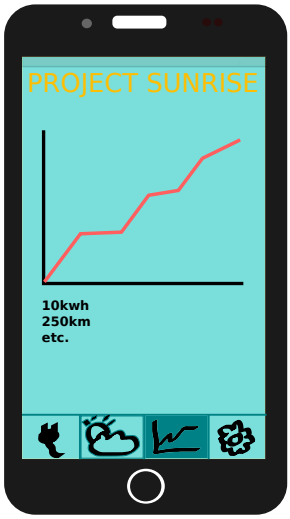


Figuur 13: Ontwerp van de reserveer pagina



Figuur 14: Ontwerp van de weer pagina

Om de pagina's goed en overzichtelijk eruit te laten zien wordt gebruik gemaakt van CSS om de HTML op te maken. Omdat de app bedoeld is om zowel op PCs met grote schermen als op mobiele apparaten, die veelal over een kleiner scherm beschikken, te draaien is het handig als de



Figuur 15: Ontwerp van de statistieken pagina

Figuur 16: Ontwerp van de instellingen pagina

app meeschaalt met de grootte en pixeldichtheid van het scherm. Om dit te realiseren gaat de app responsief gemaakt worden zoals het voorbeeld in figuur 17. Responsiviteit houdt in dat de webpagina goed te bekijken is zonder onnodig te hoeven inzoomen of scrollen. Dit is te realiseren door in CSS puur relatieve afmetingen te gebruiken.

4.2.3 PHP login

Als de gebruiker inlogt, moet de server verifiëren of de gebruiker de goede gebruikersnaam en wachtwoord heeft ingevuld. Verder moet er een beveiliging worden ingebouwd zodat ongeautho-riseerde gebruikers geen pagina's kunnen bezoeken die normaal alleen bezocht kunnen worden na ingelogd te zijn. Deze beveiliging kan gerealiseerd worden met PHP-sessies. PHP-sessies zorgen ervoor dat de gebruiker een sessie-ID toegewezen krijgt. Dit sessie-ID geeft toegang tot een set variabelen op de server die gebruikt kan worden om bijvoorbeeld persoonlijke instellingen op te

Figuur 17: Responsief ontwerp webpagina

slaan en of de gebruiker al is ingelogd. Omdat bijna alle data op de server staat is deze relatief lastig te hacken. De enige data die bij de gebruiker wordt opgeslagen is de sessie-ID. Deze is op twee manieren aan de gebruiker mee te geven:

[1.1] Via de URL. Hierbij wordt er achter het webadres een unieke sessie-ID geplakt waaraan de server kan herkennen welke data het kan vrijgeven.

[1.2] Via een cookie. Dit heeft hetzelfde effect als via een URL alleen wordt de ID nu in een cookie bewaard.

Het voordeel van een sessie-ID in de URL is dat het ook werkt bij browsers die geen cookies toestaan. Een nadeel is dat de URL er minder mooi uitziet en dat opgepast moet worden dat de volledige URL niet in handen komt van kwaadwillenden. Er is gekozen voor een sessie-ID in een cookie omdat hierbij de URL er mooier uitziet. Normaal moet er toestemming gevraagd worden om een cookie op te slaan, maar in het geval van een sessie-ID cookie is dit niet nodig.

4.3 Ontwerp van de media website

De algemene media website is bedoeld om de oplaadpaal te promoten en moet daarom aantrekkelijk en gemakkelijk te gebruiken zijn voor de gebruiker. Om aan de hoofdeisen uit hoofdstuk 3 te voldoen zijn er voor de media website een aantal specifiekere eisen opgesteld:

[1.1] Op de hoofdpagina komt een door de gebruiker te roteren model van het station met een aantal bezienswaardigheden die rondom de paal te vinden zijn (zoals de faculteit EWI) zodat goed te zien is waar de paal zich bevindt.

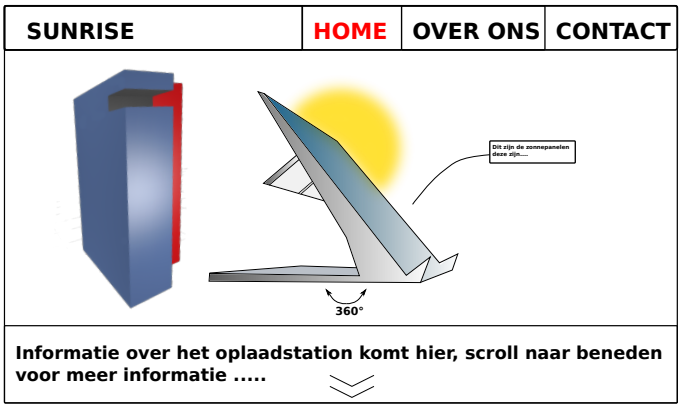
[1.2] Er moet een pagina komen met een kaart waarop de locatie te vinden is van de oplaadpaal.

[1.3] Er moet een pagina komen met alle personen die aan de ontwikkeling van de oplaadpaal hebben bijgedragen.

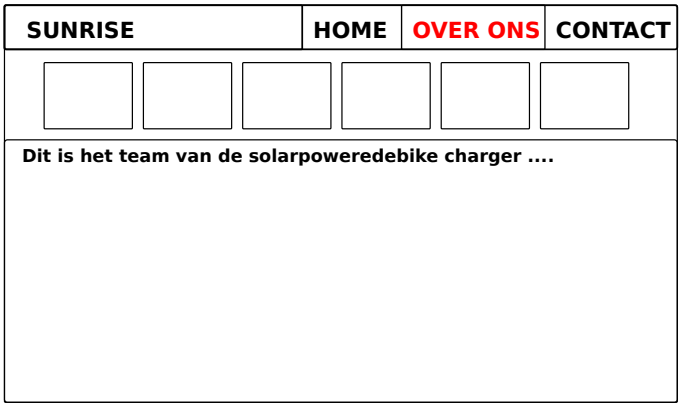
4.3.1 Visuele ontwerp

Voor de mediawebsite zijn een drietal designs gemaakt voor de home pagina (zie figuur 18), de over ons pagina (zie figuur 19) en een contact pagina (zie figuur 20).

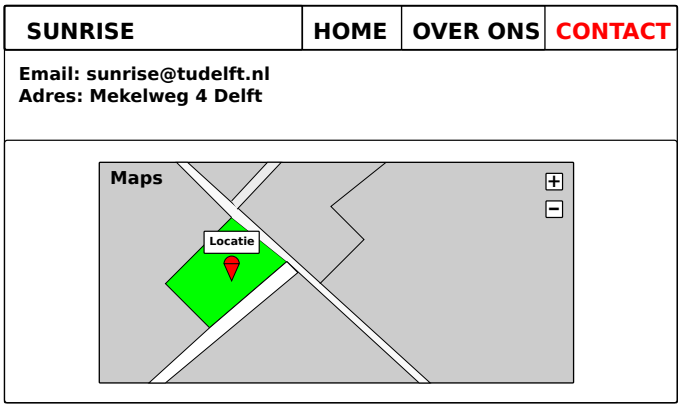
Bovenaan komt een menu bar om te navigeren naar de drie pagina's toe. Op elke pagina zal dit menu zichtbaar zijn. Op de hoofdpagina komt een mogelijkheid om naar beneden te scrollen waar meer informatie over de paal te vinden zal zijn.



Figuur 18: Ontwerp van de hoofdpagina



Figuur 19: Ontwerp van de over ons pagina



Figuur 20: Ontwerp van de contact ons pagina

5 Implementatie

Het ontwerp van alle onderdelen uit hoofdstuk 4 is geïmplementeerd. In dit hoofdstuk zal per onderdeel het implementatie proces worden beschreven.

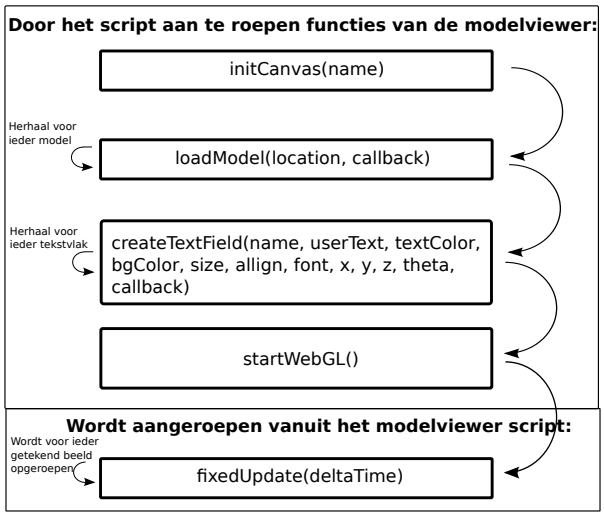
5.1 Implementatie van de modelviewer

De modelviewer is geïmplementeerd met JavaScript in combinatie met WebGL. Figuur 32 in appendix A geeft een overzicht van het modelviewer script met alle functies en klassen. Het modelviewer script is een losstaand script wat voor meerdere pagina's gebruikt kan worden, zolang er maar een canvas element meegegeven wordt. Hier is voor gekozen omdat het modelviewer script dan voor zowel de media site als de reserveer applicatie te gebruiken is.

5.1.1 Werkwijze van de modelviewer

De volledige werking van het systeem (te zien in figuur 32) zal in dit document niet worden uitgelegd, hiervoor zal de gebruikershandleiding geraadpleegd moeten worden. Wel zullen enkele belangrijke functies worden belicht.

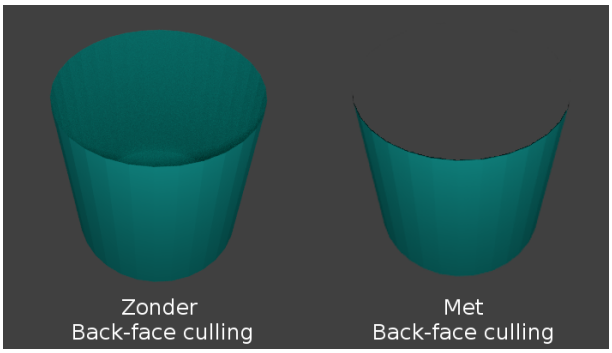
De modelviewer kan gebruikt worden door zijn functies aan te roepen vanuit een apart script. Voor het initialiseren en opbouwen van een simpele scene dienen de volgende functies aangeroepen te worden (zie figuur 21).



Figuur 21: Functies die aangeroepen dienen te worden om een simpele scene op te kunnen starten

In de opvolgende paragrafen zullen de functies kort worden besproken.

initCanvas functie Deze functie zal het canvaselement zoeken op de HTML pagina met de meegegeven naam "name" en hierna zal het canvas worden geïnitialiseerd waaronder de afmetingen aanpassen en het toevoegen van een WebGL element. Verder worden hier een aantal WebGL parameters ingesteld waaronder de dieptebuffer en back-face culling. De dieptebuffer wordt gebruikt om te bepalen welke delen van het object/model getekend moeten worden indien meerdere objecten elkaar overlappen. De dieptebuffer slaat voor elke pixel op wat de afstand is naar de camera toe. Indien twee of meerdere pixels overlap hebben zal de meest dichtstbijzijnde pixel getekend worden. Back-face culling is een techniek om vlakken niet te tekenen indien de normaal bij deze vlakken van de camera afwijst. Door deze techniek hoeven er minder vlakken getekend te worden en zal de prestatie worden verhoogd. Figuur 22 laat het verschil zien zonder en met back-face culling bij een cylinder waarbij het bovenste vlak is weggehaald.



Figuur 22: Een voorbeeld van back-face culling

loadModel functie Deze functie heeft 2 parameters: het pad + naam van het in te laden model en een callbackfunctie. De callbackfunctie zal worden uitgevoerd direct nadat het model ingeladen is. Dit is nodig omdat het laden over het internet enige tijd kan kosten en het inefficiënt is om hierop te wachten. Deze functie moet voor elk model dat ingeladen dient te worden, aangeroepen worden. Voor ieder model wordt een drietal bestanden ingeladen: een .json, .png en een .ez bestand. Het .json bestand bevat informatie over de vertices, indices en normals (zie 4.1.6) van het model. Alle modellen die zijn gemaakt voor de mediawebsite en reserveerapplicatie, zijn gemaakt met behulp van het programma Blender [4] en daarna naar het .json formaat omgezet met behulp van het computerprogramma assimp2json [9]. De modelviewer leest de informatie over de vertices, indices en normals vanuit het .json bestand in en stopt vervolgens deze informatie in een WebGL buffer zodat het model getekend kan worden in de renderloop (5.1.5). Naast het inlezen van het model, wordt er ook een texture (4.1.5) ingelezen die geprojecteerd zal worden op het model. De texture heeft het bestandsformaat .png. Het .ez bestandsformaat bevat een lijst met parameters die specifiek zijn voor het model. Het gaat hierbij om parameters zoals de diffusie intensiteit, ambient intensiteit (zie 4.1.4) en positie van het model in cartesische coördinaten. Al deze informatie zal in elke instantie van de klasse Model opgeslagen worden als ezData (zie ook het systeem overzicht 32 in appendix A).

createTextField functie Deze functie is optioneel en is gemaakt om tekst en nummers te kunnen weergeven op een tekstveld via WebGL. Omdat WebGL normaal gesproken geen ondersteuning biedt voor tekstvelden, is deze functie gemaakt die een rechthoekig vlak maakt en een texture creëerd via een 2D canvas, om deze vervolgens in de buffers te stoppen zodat ze bij de eerst volgende renderloop(5.1.5) iteratie op het beeld worden getekend. Net als de loadModel functie heeft deze functie ook een callbackfunctie als parameter die zal worden aangeroepen wanneer het tekstveld is gecreëerd.

startWebGL functie Nadat alle modellen en tekstvelden zijn ingeladen en de buffers zijn gevuld, zal deze functie aangeroepen worden. Deze functie laad en compileert de vertex(4.1.2) en fragment(4.1.3) shaders om vervolgens de renderloop(5.1.5) te starten waarin er getekend gaat worden.

fixedUpdate functie Deze functie is niet om aan te roepen, maar zal worden aangeroepen vanuit het modelviewer script en moet dus worden geïmplementeer in het huidige script. In deze functie kunnen naar wens, modellen worden geanimeerd, geroteerd, getransleerd en geschaald worden. De parameter “deltaTime"geeft aan hoeveel tijd er tussen ieder getekend beeld zit zodat bijvoorbeeld animatiesnelheden kunnen worden geïmplementeerd die onafhankelijk zijn van het aantal beelden per seconde die de computer aan kan. Hierdoor zal iedere animatie met dezelfde snelheid afspelen ongeacht of het door een snelle of langzame computer wordt getekend.

5.1.2 De klasse model

Via de loadmodel functie(5.1.1) wordt een model ingeladen. Bij het laden wordt er een object geïnstantieerd van de klasse "model". De klasse "model" bevat de volgende attributen:

name	de naam van het model.
position	een driedimensionale vector met de positie van het model in cartesische coördinaten.
rotation	een driedimensionale vector met de rotatie van het model in graden.
vertices	een Array met alle coördinaten van alle vertices.
indices	een Array met alle indices.
normals	een Array met alle normals van elk vlak in het model.
texCoords	een Array met coördinaten om de texture te kunnen projecteren op het model
texture	de texture zelf.
buffers	de WebGL buffers die zijn gevuld met de vertices, indices en normals.
ezData	bevat de parameters van het model.
worldMatrix	de world matrix van het model.
active	een Boolean die aangeeft of het model wel of niet getekend dient te worden.
childs	de "kinderen" van het model.
parent	de "ouder" van het model.

De klasse "model" bevat ook nog functies die aangeroepen kunnen worden op ieder model waarmee modellen getransleerd, geroteerd, geschaald en geanimeerd kunnen worden. Het animeren gebeurd altijd in de "fixedUpdate" functie (5.1.1) waarin iedere iteratie van de renderloop het model geroteerd, getransleerd en geschaald kan worden. Al deze operaties worden gedaan met behulp van de worldmatrix van het model. In paragraaf 5.1.4 wordt uitgelegd hoe dit in zijn werking gaat.

Naast de transformatie functies bevat een model ook functies om "childs" en een "parent" toe te voegen. Deze functie is geïmplementeerd zodat "child" modellen kunnen roteren om de as van hun "parent" en ook getransleerd kunnen worden met de "parent" mee. Deze functie is bijvoorbeeld handig als op de media website, de fietsen mee moeten draaien met de oplaadpaal zonder dat de originele rotatie of positie van de fiets wordt aangepast.

5.1.3 Afgeleiden van de klasse model

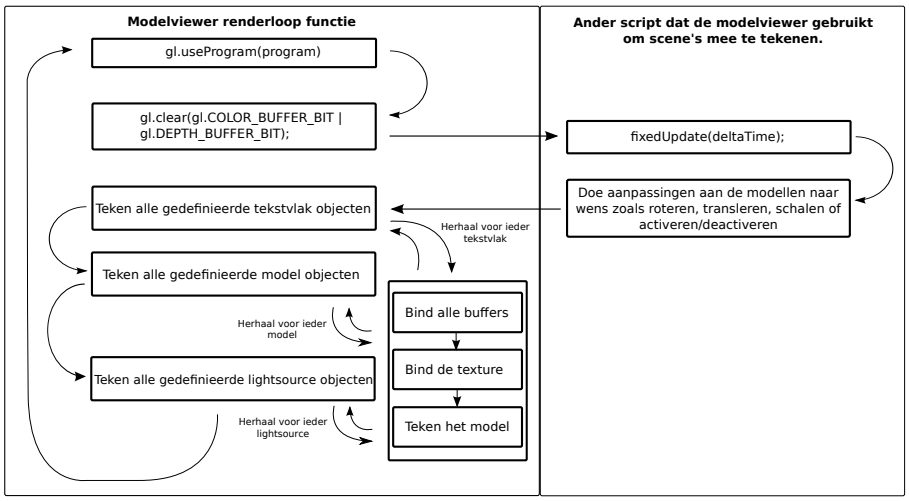
De klasse "model" bevat ook nog afgeleide klassen waaronder "text" en "lightSource". De klasse "text" is voor het maken van tekstvlakken en de klasse "lightSource" is vrijwel exact aan klasse "model" met het enige verschil dat een "lightSource" licht geeft en daarom een extra attribuut heeft die de lichtintensiteit specificeert.

5.1.4 Roteren, transleren en schalen van modellen

Om te kunnen roteren, transleren en schalen zullen er matrix operaties gedaan moeten worden op de worldmatrix van het te animeren "model". In hoofdstuk 4.1.7 staat uitgelegd wat deze operaties inhouden en hoe deze operaties moeten worden uitgevoerd. Voor de modelviewer is gekozen om een script genaamd glMatrix [3] te gebruiken om matrix en vector operaties uit te voeren. glMatrix is een uitgebreid script dat is gericht en geoptimaliseerd voor het gebruik in samenhang met WebGL.

5.1.5 De renderloop

Nadat alles is geïnitialiseerd, de shaders zijn gecompileerd en de modellen zijn ingeladen zal de renderloop opgeroepen worden. In de renderloop wordt alle data die in de buffers zijn gestopt, getekend op het scherm. Figuur 23 laat de werkwijze zien van de renderloop.



Figuur 23: De werkwijze van de renderloop

De eerste WebGL functie gl.useProgram(program) specificeert welk WebGL programma gebruikt moet worden om te tekenen. WebGL heeft de mogelijkheid om meerdere programma's te gebruiken en daarbij dus verschillende shaders te gebruiken. De modelviewer bevat maar één programma dat gebruik maakt van Phong Shading shaders en deze wordt geselecteerd met de gl.useProgram functie.

De volgende stap (gl.clear(gl.COLOR_BUFFER_BIT |gl.DEPTH_BUFFER_BIT)) is om de kleur en diepte buffer te resetten. Dit is nodig als bijvoorbeeld een model aan het roteren is om een ander model heen en het model zich daardoor voor het andere model schuift. In deze situatie moet de diepte buffer worden geüpdatet omdat anders het verkeerde object wordt getekend dat zich achter het eigenlijk te tekenen model bevind. In paragraaf 5.1.1 staat uitgelegd wat de diepte buffer precies inhoud.

Als derde stap wordt de fixedupdate functie opgeroepen (zie paragraaf 5.1.1).

Nadat alle gewenste transformaties gedaan zijn in de fixedupdate functie, kunnen alle objecten worden getekend. Eerst worden alle tekstvlakken getekend, daarna de lightsources en vervolgens alle modellen. Het tekenen van al deze objecten gebeurd op dezelfde manier. Eerst worden de buffers "gebind" die tijdens de initialiseer fase gevuld zijn met informatie over het object zoals de vertices, indices en normals. Daarna worden de textures "gebind" en vervolgens zal het object getekend worden op het scherm.

Als de renderloop klaar is met tekenen, wordt de functie opnieuw aangeroepen om het volgende beeld te tekenen.

5.2 Implementatie van de app

De app bestaat uit een aantal onderdelen. Allereerst is er het loginsysteem om ervoor te zorgen dat alleen geauthoriseerde gebruikers gebruik maken van de app. Verder wordt toegelicht hoe de HTML structuur is opgebouwd en deze gestijld is met CSS.

5.2.1 HTML inlogfomulier

Het inloggen gebeurt door aan de gebruikerszijde de gebruikersnaam en het wachtwoord in te vullen en op te sturen naar de server. Deze worden in een HTML formulier ingevuld en met een HTTP POST request opgestuurd naar de server. Een manier hoe dit geïmplementeerd kan worden is te zien in script 3.

Script 3: HTML inlogformulier

```
<form name="login" method=POST>
  <input type="username" placeholder="Gebruikersnaam" name="user_input_field" pattern="↩
      "[A-Za-z]{1,8}" />
    <input type="password" placeholder="Wachtwoord" name="password_input_field" size="↩
        20" />
        <input type="submit" placeholder="Verder" name="submit_button"/>
</form>
```

Een loginformulier is voor een kwaadwillende een uitstekende plek om SQL injecties te proberen. Wat dit inhoudt is dat de gebruiker SQL commando's gaat invoeren in plaats van een gebruikersnaam en een wachtwoord. Hiermee kan op een slecht beveiligde server gevoelige informatie openbaar gemaakt worden. Ook zou er HTML code ingevoerd kunnen worden om gebruikers over te halen gevoelige informatie in te voeren die niet naar de server gaat maar omgeleid wordt naar de server van de kwaadwillende. Om dit te voorkomen is de gebruikersnaam gelimiteerd tot alfanumerieke karakters zodat er geen code ingevoerd kan worden.

5.2.2 PHP loginsysteem

Als de gebruiker op Submit heeft geklikt worden de logingegevens opgestuurd naar de server en komen aan in de varabele $_POST. Dit is een array waar alle variabelen in komen te staan die via HTTP naar de server worden verzonden. Op de server wordt gezocht of er een account bestaat met dezelfde gebruikersnaam als ingevoerd. Als dit het geval is wordt gekeken of het wachtwoord juist is en of de gebruiker als administrator staat geregistreerd. PHP-code die hiervoor verantwoordelijk is is hieronder te zien in script 5.2.1.

Script 4: Loginsysteem PHP

```

<?php

        $username = $_POST["username"];
            $password = $_POST["password"];

        //check database for submitted username
        $query = getQuery(false,"SELECT * FROM Accounts WHERE name = ".$username."");

        $name = $query["name"];
        $pass = $query["password"];
        $admin = $query["admin"];

        if($name != $username) {//check if username is registered
          echo "User not known.";
        }
        else if($pass != $password) {//check if passwords match
          echo "Wrong password. Try again.";
        }
        else if($admin){//check for admin permissions
          header("Location: http://sunrise.org/admin");
        }
        else {//user is logging in with a normal user account
          header("Location: http://sunrise.org/user");
        }

?>
```

Het inloggen kan nu echter makkelijk worden omzeild door direct naar de reserveerpagina te gaan in de browser. Hierdoor kunnen er onbeperkt oplaadplekken worden gereserveerd zonder

authenticatie via het loginscherm wat natuurlijk niet de bedoeling is. Om dit te verhelpen gaat gebruik gemaakt worden van PHP-sessies. Om beter te begrijpen hoe hiermee een veilige login gerealiseerd kan worden volgen enkele voorbeelden in sectie 5.2.3.

5.2.3 PHP sessies

In PHP kan gebruik gemaakt worden van sessies om data alleen aan een specifiek persoon te laten zien. Hierbij wordt een sessie gestart door de functie urlsession_start() aan te roepen. Hiermee wordt een sessie-ID aangemaakt en als cookie opgeslagen in de browser. Dit ID geeft toegang tot specifieke variabelen op de server, opgeslagen en bereikbaar onder het array $_SESSION. Totdat de browser gesloten wordt zijn deze variabelen beschikbaar, alleen als men over het juiste session ID beschikt. In het geval van inloggen worden de gebruikersnaam en of de persoon een administrator is opgeslagen. Een administrator komt na het inloggen op het admin dashboard en een normale gebruiker bij het gebruikersmenu. Een voorbeeld van het toepassen van sessies is hieronder te zien in script 5.

Script 5: Sessies PHP

```
1
2  <?php
3          session_start();
4          if($name != $username) {//check if username is registered
5            echo "User not known.";
6          }
7          else if($pass != $password) {//check if passwords match
8            echo "Wrong password. Try again.";
9          }
10         else if($admin){//check for admin permissions
11           $_SESSION["username"] = $name;
12           $_SESSION["admin"] = true;
13           header("Location: http://sunrise.org/admin");
14         }
15         else {//user is logging in with a normal user account
16           $_SESSION["username"] = $name;
17           $_SESSION["admin"] = false;
18           header("Location: http://sunrise.org/user");
19         }
20 ?>
```

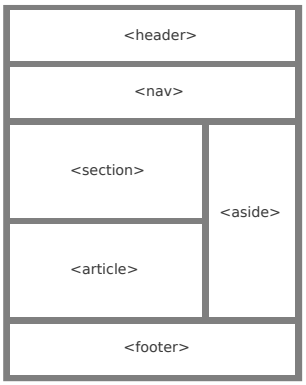
De code ziet er bijna hetzelfde uit als de code in script 5.2.1 met een paar uitzonderingen: eerst wordt een sessie gestart met urlsession_start() waar later de gebruikersnaam wordt opgeslagen en opgeslagen wordt of de gebruiker administratorrechten heeft. Dit is van belang zodat verhinderd kan worden dat een normale gebruiker bij het admin dashboard kan komen.

5.2.4 HTML

Als HTML opbouw is gekozen voor semantische HTML [10]. Waar gebruikelijk voor bijna elk element een tag wordt gebruikt, is dat in deze app anders. Een element zegt niet veel over de inhoud. Het kan een element aan de zijkant zijn maar net zo goed de hele pagina beslaan. Het voordeel van semantische HTML is dat zoekmachines de webpagina beter kunnen indexeren omdat de HTML tags een betekenis hebben. Voor de titelbalk bovenaan wordt een <header> tag gebruikt en voor de navigatiebalk onderin een <footer> tag. Verder wordt de rest van de pagina ingepakt in <main> tags. Het inlogformulier wordt aangegeven met <form> tags. De indeling van een webpagina met sematische tags zoals in de HTML5 standaard gespecificeerd is te zien in figuur 24.

5.2.5 CSS

Op alle pagina's van de app is er boven een header en onderaan een footer geplaatst met CSS. De header bevat de titel en in de footer worden knoppen voor navigatie naar verschillende pagina's weergegeven. Om ervoor te zorgen dat deze elementen altijd boven- en onderaan de pagina verschijnen wordt gebruikgemaakt van een stukje CSS code te zien in 6



Figuur 24: de indeling van een semantische HTML pagina

Script 6: header en footer in CSS

```
header, footer {
  position: fixed;
  height: 7%;
  background: blue;
  width: 100%;
  min-height: 40px;
}

header {
  top: 0;
}

footer {
  bottom: 0;
}
```

Beide elementen worden met de eerste stijlregel gefixeerd op hun positie. Deze zal dus niet veranderen als er gescrolld wordt door de pagina. Verder wordt er met de background parameter een kleur ingesteld voor de achtergrond van de oppervlakte die deze elementen innemen, in dit geval blauw. Om ervoor te zorgen dat de balken boven en onder over de hele breedte van het scherm te zien zijn, wordt gebruik gemaakt van de width parameter. De hoogte van de header en footer worden bepaald door de height parameter en is 7% van de schermhoogte. In het geval van een kleine schermhoogte is er een minimale hoogte ingesteld van 40 pixels zodat de elementen niet verdwijnen. Omdat er met relatieve afmetingen wordt gewerkt is een responsief ontwerp gerealiseerd waarbij dezelfde webpagina zowel op een PC als op een mobiel apparaat er goed uit ziet.

Voor de animaties is ook CSS gebruikt. Een veelvoorkomend effect dat bijdraagt aan het uiterlijk van webpagina's zijn subtiele transities, bijvoorbeeld een kleurverandering van een element als hier de muis boven wordt gehouden. Dit is ook in de app toegepast. De CSS code die hiervoor nodig is, is te zien in script 7.

Script 7: CSS transitie bij het houden van de muis boven het respectievelijke element

```
#option {
    display:inline-block;
    height:100%;
    width: 17%;
  background-color: blue;
    transition: background-color 0.4s;
}

#option:hover{
        background-color: lightgray;
}
```

Met deze CSS stijlregels wordt allereerst ervoor gezorgd dat alle elementen met id “option” naast elkaar worden gezet door de parameter display:inline-block. Verder wordt de achtergrondkleur en de transitietijd ingesteld. De transitietijd met parameter transition bepaalt hoe lang de overgang duurt wanneer de muis boven het element gehouden wordt, in dit geval 0.4 seconde. Met de stijlregel #option:hover wordt daadwerkelijk aangegeven naar welke kleur de transitie gaat plaatsvinden: lichtgrijs. Als de muis niet meer boven het element gehouden wordt gebeurt de omgekeerde transitie: de kleur verandert van lichtgrijs naar blauw.

Voor de reserveerpagina is gekozen voor een animatie waarbij de reserveerknoppen van bovenaf komen inschuiven. Aangezien er elke keer een ander aantal plekken vrij kan zijn vraagt dit ook om een dynamische manier van inladen. Als elke van de zes knoppen naar dezelfde plek zou schuiven zouden er gaten tussen vallen die afbreuk doen aan het uiterlijk van de app. Het is immers overzichtelijker deze knoppen direct onder elkaar te hebben. Om dit probleem te verhelpen wordt er naast CSS animaties gebruik gemaakt van een stukje jQuery [1]. Dit is te zien in script 8.

Script 8: jQuery voor het dynamisch inladen van reserveerknoppen

```
1
2  //aantal knoppen is input geregeld vanuit de server
3  //De CSS die toegevoegd wordt zal eruit komen te zien als:
4  //  .reserve_form button:nth-child(2) {
5  //    animation-name: button1;
6  //    animation-duration:0.5s;
7  //    animation-timing-function:ease-in-out;
8  //    animation-iteration-count:1;
9  //    animation-fill-mode:forwards;
10 //  }
11 //dit geval is code voor de tweede button
12 function loadButtons(buttoncount) {
13 for(var i=0;i<buttoncount;i++) {
14   var buttonId = ".reserve_form button:nth-child(" + (i+1) + ")";
15   var buttonchild = "button" + (i+1);
16   $(buttonId).css({"animation-name":buttonchild,"animation-duration":"0.5s","animation↩
         -timing-function":"ease-in-out","animation-iteration-count":"1","animation-fill-↩
         mode":"forwards"});
17 }
18
19 }
```

Vanuit de server wordt er via een PHP script in de MySQL database gekeken hoeveel en welke plekken er vrij zijn en hierop wordt de HTML aangepast aan het aantal te laden knoppen. Bovenstaand jQuery script voegt de juiste CSS code in om de knoppen op de goede plekken terecht te laten komen. De CSS die gegenereerd wordt roept de juiste stijlregel aan die al aanwezig is. Zo’n stijlregel is te zien in script 9.

Script 9: CSS voor het transleren van de reserveerknoppen

```
1
2  @keyframes button2 {
3  0%    { transform:translate(0,0);
4      }
5  100%  { transform:translate(0,200%);
6      }
7  }
```

De animatie is zo gedefiniëerd dat deze in 0.4s van de beginpositie naar de eindpositie schuift.In dit geval betreft het de tweede knop die dus twee knophoogten omlaag wordt getransleerd. Dit betreft niet altijd het reserveren van het tweede oplaadpunt: als deze al bezet is schuift de volgende niet bezette knop naar deze positie.

5.3 Implementatie van de media website

5.3.1 De homepagina

Op de hoofdpagina wordt het modelviewer script gebruikt om het oplaadpaal en enkele bezienswaardigheden te tekenen. De modellen zijn eerst via het programma Blender getekend [4] en daarna omgezet naar het juiste formaat voor de modelviewer (zie paragraaf 5.1.1). Via de werkwijze die in figuur 21 staat beschreven is de scène opgebouwd. In figuur 10 is een stukje code van de hoofd pagina te zien. Er worden drie modellen ingeladen via de "loadModel" functie van de modelviewer: een zon, fiets en het station. Hierna worden een aantal transformaties uitgevoerd op de drie modellen en wordt de fiets als "child" toegevoegd aan het station, zodat bij het draaien van het station de fiets meegaat. Vervolgens wordt de "startWebGL()" functie opgeroepen waardoor de modelviewer gaat starten met het tekenproces. In dit geval staat er niets geschreven in de "fixedUpdate()" functie omdat er geen animaties op de hoofdpagina worden weergegeven. In de scène zal de oplaadpaal en de fiets langzaam draaien. Ook is er de mogelijkheid voor de gebruiker om met de muis de twee modellen te draaien. De handmatige en automatische draaifuncties zijn een onderdeel van de modelviewer en zijn in de .ez bestanden, die automatisch worden ingeladen bij ieder model, gespecificeerd. Figuur 11 laat het .ez bestand zien van het station.

Script 10: Een stukje code van de media website

```
1
2  //initiate the canvas
3  var canvas = initCanvas("webgl_canvas");
4
5  //variables to hold the modelinformation
6  var station = new model();
7  var bike = new model();
8  var sun = new model();
9
10 //load all models
11 loadModel("./Models/station", function (model) {
12     station = model;
13
14     loadModel("./Models/sun", function (model) {
15         bike = model;
16
17               loadModel("./Models/sun", function (model) {
18             sun = model;
19
20         //translate the sun
21             sun.translate(4, 4, -5);
22
23             //scale the sun
24             sun.scaleLocal(0.9,0.9,0.9);
25
26             //rotate the camera a bit
27             cameraViewer.rotateLocal(-10, 0, 0);
28
29             //translate the station
30             station.translate(-5,0,0);
31
32             //translate the bike
33             bike.translate(-3,0,0);
34
35             //rotate the bike a bit
36             bike.rotateLocal(0,20,0);
37
38             //add the bike as a child to the station
39             station.addChild(bike);
40
41             //initiate the modelviewer
42             startWebGL();
43
44          });
45     });
46 });
47
48 //function gets called every renderloopiteration
49 function fixedUpdate(deltaTime) {
50 }
```

Script 11: Parameters van het station model

```
#worldPosition = {0,0,0}
#ambientReflCoef = {0.9}
#diffuseReflCoef = {0.03}
#initialRotation = {0,0,0}
#shininessCoef = {150}
#autoRotate = {true}
#rotationSpeed = {10}
#meshNo = {0}
#userRotation = {true}
```

5.3.2 De 'over ons' pagina

De over ons pagina geeft een overzicht van alle mensen die hebben meegewerkt aan het project. In HTML5 is een rij met " " elementen gemaakt met foto's van ieder teamlid. Als de muis boven één van de foto's gehouden wordt, zal er meer informatie over de desbetreffende persoon verschijnen. Voor deze functionaliteit is een klein script geschreven dat gebruik maakt van jQuery [1]. In het kort gebruikt dit script "events" die getriggerd worden wanneer de muis op één van de foto's terecht komt. Vervolgens wordt met behulp van het 'id' nummer van het " " element, de juiste biografische informatie getoond.

5.3.3 Contactpagina

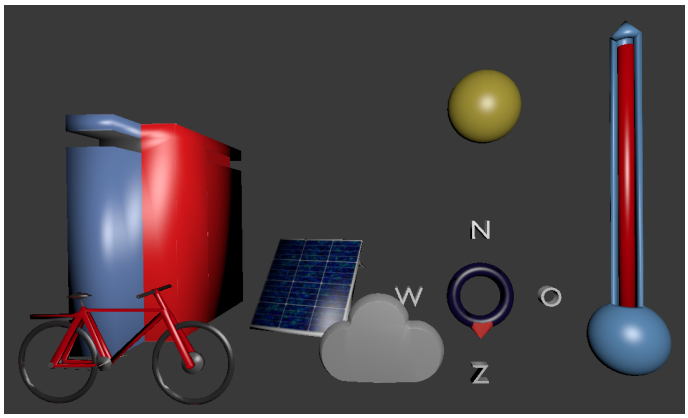
De contact pagina bevat een kaart met de positie van het laadstation. Voor de kaart is gebruik gemaakt van een script genaamd Leaflet [5]. Leaflet is een open-source, mobiel-vriendelijk script voor het tonen van interactieve kaarten. Er is voor Leaflet gekozen omdat het script klein (64KB) is en toch alle benodigde functionaliteiten bevat. De kaart die wordt gebruikt is van OpenStreetMap [11]. Naast de map bevind zich op de pagina ook een e-mailadres waar gebruikers vragen of opmerkingen kunnen stellen.

5.3.4 CSS

Voor de opmaak van de media website is net als bij de reserveer applicatie gebruik gemaakt van CSS. Bovendien is bij de media site ook de strategie van het "responsieve" design toegepast zodat ieder formaat scherm, van telefoon tot desktop computer, dezelfde site voorgeschoteld krijgt alleen dan met een verschillende indeling en of schaalfactor. Er is voor gekozen om de menubalk bovenaan altijd zichtbaar te laten zijn, ook als er naar beneden wordt gescrolld. Dit om de gebruiker extra scrollen naar boven te besparen als deze snel wil navigeren naar een andere pagina.

5.3.5 Blender

Met het programma Blender [4] zijn alle modellen gemaakt die op zowel de media website als de reserveer applicatie te vinden zijn. Figuur 25 laat alle modellen zien die zijn gemaakt.

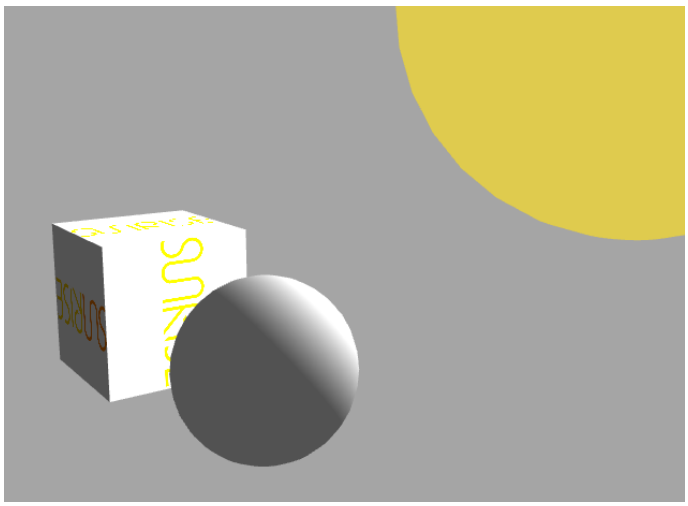


Figuur 25: De modellen die met het programma Blender [4] zijn gemaakt

6 Resultaten

6.1 Resultaat van de modelviewer

Om de modelviewer op zijn werking te testen, is er een simpele scene opgebouwd met twee modellen: een kubus en een bol. Daarnaast is er ook één lichtbron ingeladen die de zon moet voorstellen. Figuur 26 laat het resultaat zien van deze scene. De kubus laat zien dat het projecteren van textures werkt. Daarnaast is ook te zien dat het diffusie en ambient licht naar behoren werkt. Het vlak van de kubus dat zich niet in het directe licht van de zon bevind is donker van kleur en reflecteert alleen het ambient licht in de scene. De bol laat goed zien dat Phong Shading de normalen interpoleert waardoor er een zachte reflectie over de bol heen valt waardoor het lijkt of dat de bol ook echt rond is, terwijl in de werkelijkheid de bol uit driehoeken bestaat. Aan de bol is wel te zien dat de overgang van kleur vrij drastisch is. Dit komt omdat in de fragmentshader gebruik wordt gemaakt van medium precisie (zie paragraaf 4.1.3). High precisie zal een nog beter resultaat opleveren maar hier is niet voor gekozen omdat niet elk apparaat dit ondersteund. De lichtbron/zon straalt alleen licht uit en omdat er geen tweede lichtbron in de scene aanwezig is zal er op het oppervlak van de zon geen diffusiereflectie zijn. Dit is in overeenstemming met wat er in figuur 26 afgebeeld staat.

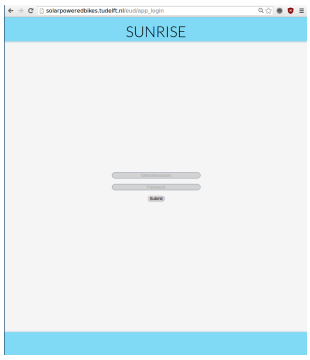


Figuur 26: Een kubus, een bol en een lichtbron (zon) getekend door de modelviewer

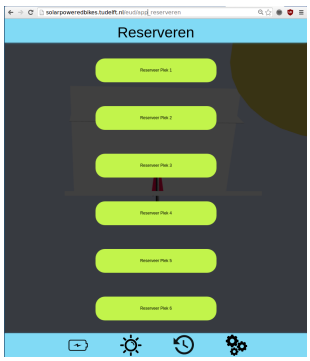
6.2 Resultaat van de app

De PHP-sessies werden op verschillende manieren getest. Allereerst werd er uitgegaan van een gewone gebruiker die inlogt en een plek reserveert en weer uitlogt. Dit verliep naar behoren. Ook werd gekeken of de reserveerpagina bereikt kon worden zonder in te loggen door direct naar het juiste webadres te gaan. Doordat de sessie bijhoudt of een gebruiker al heeft ingelogd was dit niet mogelijk en bleek de beveiliging goed te werken. Wanneer de gebruiker echter een oplaadplek reserveert en ingelogd blijft, is het nog mogelijk om onbeperkt oplaadplekken te reserveren. Deze fout gaat nog opgelost worden voordat de app uitgebracht wordt.

De app schaalt goed op grote en kleine schermen omdat er gebruik is gemaakt van relatieve afmetingen in CSS. Wat nog wel een probleem vormt is het schalen van tekst. Deze schaalt niet op dezelfde manier als de elementen waarin deze tekst zich bevindt. Hier gaat ook nog naar gekeken worden voordat de app wordt uitgebracht. Hieronder (figuur 27 en 28) zijn het reserveer- en loginscherm in hun huidige staat te zien.



Figuur 27: het loginscherm van de app

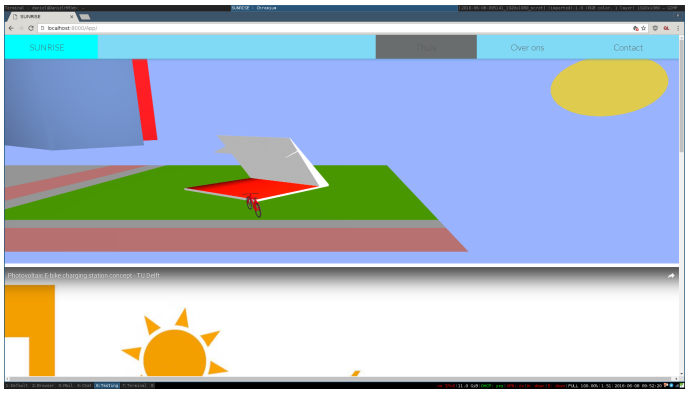


Figuur 28: het reserveerscherm van de app

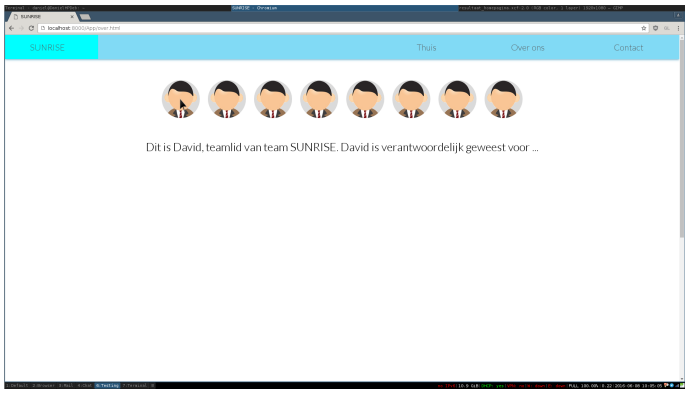
De huidige versie van de app is te vinden via `http://solarpoweredbikes.tudelft.nl/eud/app_login`. Inloggen kan met de volgende inloggegevens:
Gebruikersnaam: testaccount Wachtwoord: BAP

6.3 Resultaat van de mediawebsite

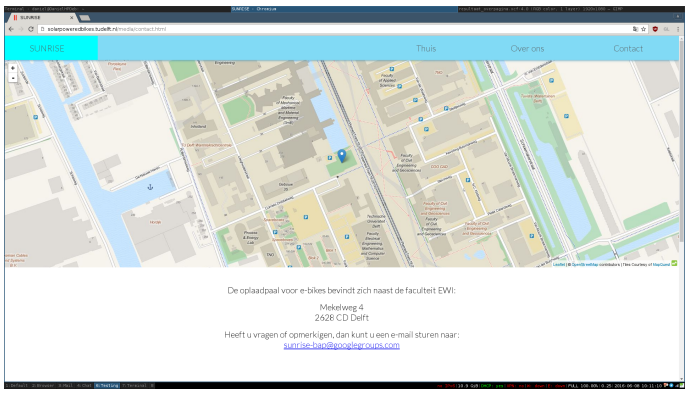
Ten tijde van het schrijven van deze thesis is de mediawebsite nog niet helemaal af. Technisch gezien werkt alles naar behoren en zullen er nog weinig aanpassingen gedaan worden aan scripts. Het enige wat nog mist is de informatie die nog op de pagina moet komen. Figuur 29 tot en met 31 laat de huidige staat zien van de mediawebsite. Via de volgende link: `http://solarpoweredbikes.tudelft.nl/media/index` kan de laatste versie van de website worden bekeken.



Figuur 29: De hoofdpagina



Figuur 30: De over ons pagina



Figuur 31: De contact pagina

7 Discussie

Bijna alle onderdelen zijn met succes geïmplementeerd en wordt er voldaan aan alle eisen die aan het systeem gesteld zijn. Desondanks zijn er een aantal zaken waar niet aan toegekomen is. Er lag een idee om ook een Android en iPhone applicatie te produceren die dezelfde functie vervult als de reserveerpagina alleen van dit idee is uiteindelijk afgeweken omdat de overige onderdelen een hogere prioriteit hadden. Het idee is uiteindelijk opgenomen in de lijst van aanbevelingen 8.2 zodat deze wellicht in de toekomst alsnog worden geïmplementeerd. Ook was het de bedoeling om de oplaadstatus te kunnen bijhouden van de eigen fiets in het laadstation. Dit is echter niet gelukt omdat er veel verschillende typen accu's in de verschillende e-bikes zitten. Er is hierdoor geen mogelijkheid om het exacte batterijpercentage te kunnen uitlezen.

Voor het modelviewscript had ook een voorgeprogrammeerde library gebruikt kunnen worden zoals three.js [12]. Dit zou een grote tijdwinst opleveren voor de implementatie. Hier is expres vanaf geweken omdat met een opbouw vanuit de basis een beter begrip kan worden gevormd over wat WebGL is en hoe het werkt. Zeker gezien de beperkte kennis over computergraphics vanuit de elektrotechniek, zal de opbouw vanuit de basis het leerproces vergemakkelijken.

Tijdens het testen van de WebGL omgeving bleek dat op computers met een verse installatie van Windows 7 de WebGL functionaliteit ontbrak. Dit kwam omdat op de desbetreffende computers nog niet alle updates waren geïnstalleerd. In de meeste gevallen zijn deze updates wel geïnstalleerd en zal het probleem zich niet voordoen. Om toch met de eerste groep rekening te houden wordt er teruggevallen op een geanimeerde afbeelding. Hoewel deze op alle computers kan laden, onbreekt hierbij wel de interactiviteit.

8 Conclusie en aanbeveling voor de toekomst

8.1 Conclusie

De bedoeling van het project was om een end-user interface te maken voor de gebruiker dat bestaat uit twee onderdelen: een mediawebsite en een reserveerapplicatie. In beide onderdelen is gebruik gemaakt van WebGL voor het weergeven van grafische content en een PHP-server voor de uitwisseling van gegevens. Alle onderdelen zijn met succes geïmplementeerd.

8.1.1 De modelviewer

De modelviewer is een script dat is geschreven voor het weergeven van driedimensionale content en werkt met behulp van WebGL. De modelviewer komt niet voor in het programma van eisen maar is wel nodig om bijvoorbeeld aan eis [2.1.1] te kunnen voldoen. De modelviewer is een losstaand script dat voor zowel de applicatie als de mediawebsite te gebruiken is. De modelviewer is met succes geïmplementeerd en kan driedimensionale modellen importeren, transformeren en vervolgens tekenen op het scherm. Ook bied de modelviewer de mogelijkheid om animaties toe te voegen. Daarnaast is er gebruik gemaakt van Phong shading die de driehoekige aard van de modellen moet onderdrukken.

8.1.2 De reserveerapplicatie

In de HTML5 reserveerapplicatie is het nu als gebruiker mogelijk om in te loggen en een plek te reserveren. De PHP server zorgt ervoor dat de juiste opladers aan of uit gezet worden waardoor aan eis [1.1.1] tot en met eis [1.1.3] is voldaan. Hiernaast bevat de applicatie een weerpagina en is er een statistieken pagina gemaakt waarmee aan eis [1.1.4] is voldaan. De applicatie is met succes getest op verschillende apparaten en heeft een gebruiksvriendelijke interface wat eis [1.2.1] tot en met [1.2.3] ten goede doet. Naast de gestelde eisen zijn er ook nog extra functies toegevoegd voor de verbetering van de applicatie, waaronder: HTTPS beveiligde logins, een instellingenpagina en driedimensionale animaties.

8.1.3 De mediawebsite

De mediawebsite heeft op de hoofdpagina een driedimensionaal model van de oplaadpaal dat door de gebruiker te draaien is en voldoet hiermee aan eis [2.1.1] en [2.2.1]. Verder heeft de mediawebsite een simplistisch design dat gemakkelijk te overzien is voor de gebruiker. Op de contact is de locatie te vinden van de oplaadpaal. Dit is conform eis [2.1.1]. Naast de gestelde eisen zijn er ook nog extra functies toegevoegd voor de verbetering van de mediawebsite waaronder: een kaart met marker die de locatie van de paal visueel weergeeft en op de hoofdpagina zijn er naast de oplaadpaal ook nog enkele bezienswaardigheden te zien zoals de faculteit EWI.

8.2 Aanbeveling voor de toekomst

De oplaadpaal is op dit moment in een experimentele fase en zal voorlopig alleen toegankelijk zijn voor medewerkers van de faculteit EWI. Indien het experiment succesvol wordt en de oplaadpaal toegankelijker wordt zal het end-user interface op een aantal punten kunnen worden uitgebreid. Van de HTML5 reserveer applicatie zou een iOS, Android of Windows Phone variant gemaakt kunnen worden waarmee push-meldingen kunnen worden ontvangen als de fiets volledig is opgeladen. Verder kunnen de statistieken worden uitgebreid met bijvoorbeeld een functie die automatisch de actieradius berekend aan de hand van het type e-bike. Ook kan de oplaadstatus in de app laten zien gezien worden als een uitbreiding. Tot slot zou de media website aangevuld kunnen worden met nieuwe features zoals bijvoorbeeld de mogelijkheid om de pagina te delen.

Referenties

[1] M. authors, “jquery.” [Online]. Available: http://jquery.com

[2] K. Group, “Webgl specification.” [Online]. Available: https://www.khronos.org/registry/webgl/specs/1.0/

[3] B. Jones and C. MacKenzie IV, “glmatrix: Javascript matrix and vector library for high performance webgl apps.” [Online]. Available: http://glmatrix.net/

[4] Blender Foundation, “Blender.” [Online]. Available: https://www.blender.org

[5] V. Agafonkin., “Leaflet: an open-source javascript library for mobile-friendly interactive maps.” [Online]. Available: http://leafletjs.com/

[6] Wikimedia Commons User:T-tus. (2005) Phong shading sample. [Online]. Available: https://upload.wikimedia.org/wikipedia/commons/8/84/Phong-shading-sample.jpg

[7] B. Smith. (2006) Illustration of the components of the phong reflection model (ambient, diffuse and specular reflection). [Online]. Available: https://upload.wikimedia.org/wikipedia/commons/6/6b/Phong_components_version_4.png

[8] D. J. Eck, “Introduction to computer graphics,” pp. 117–119,263–272. [Online]. Available: http://math.hws.edu/eck/cs424/downloads/graphicsbook-linked.pdf

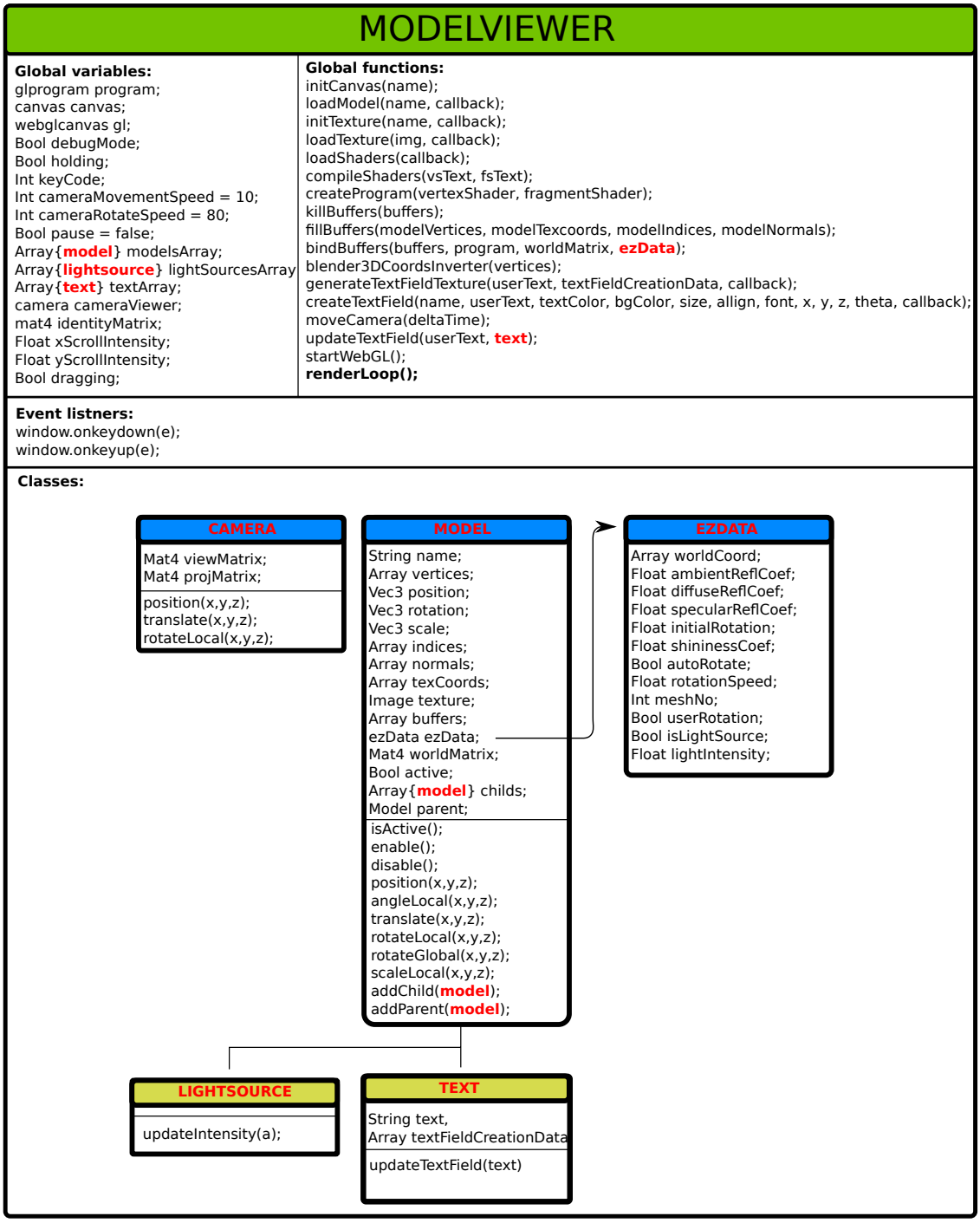
[9] A. Gessler, “Assimp2json: Json exporter for open asset import library.” [Online]. Available: https://github.com/acgessler/assimp2json

[10] M. authors, “Html5 semantic elements.” [Online]. Available: http://www.w3schools.com/html/html5_semantic_elements.asp

[11] ——, “Openstreetmap.” [Online]. Available: http://www.openstreetmap.org

[12] ——, “Three.js.” [Online]. Available: http://threejs.org

A Overzicht van alle klassen en functies van de modelviewer



Figuur 32: Overzicht van alle klassen en functies van de modelviewer