FACULTEIT INGENIEURSWETENSCHAPPEN
EN ARCHITECTUUR

TELETIJDSMACHINE GENT

GROUP 16 – PROJECT 3

Groepsleden:
Felix De Mûelenaere
Simon Adams

Faculteit Ingenieurswetenschappen en Architectuur
Afdeling Elektronica/ICT

Campus Ardoyen
Technologiepark-Zwijnaarde 125, 9052 Gent

www.ugent.be

UNIVERSITEIT
GENT

INHOUDSOPGAVE

inhoudsopgave

1 INLEIDING

1.1 Gent in 3D

"Gent in 3D" heeft tot doel de uitbouw van een volwaardig digitaal en geanimeerd 3D-model van Stad Gent in micro en in macro.

Zoveel mogelijk partners betrekken en gebruik laten maken van dit virtueel stadsmodel hoort bij de taakstelling. De digitale 3D-maquette heeft ook een tijdsdimensie, "Gent in 4D" dus als teletijdsmachine. Meer dimensies leiden tot Multi-D (zoals daar zijn: geluid, budget, ...).

Concrete taakstellingen van "Gent in 3D" zijn :

- Inwinnen van 3D-data over Stad Gent, onder meer door inzet van diverse 3D-scantechnieken
- Ondersteunen van 3D-projecten van stadsdiensten
- 3D-expertise inzetten bij uitwisseling van 3D-data tussen diverse partners
- Kennisdiffusie in samenwerkingsverbanden inzake onderzoek, onderwijs, ontwikkeling in 3D
- Promoten van inzet van 3D-communicatie
- De factor tijd als 4de dimensie meenemen: hoe zag Gent er uit in het verleden, hoe vandaag en hoe zal het er in de toekomst uit zien?
- Demonstreren van VR-apparatuur aan inwoners en bezoekers van Stad Gent
- Conversies van 3D-datasets voor gebruik in verschillende softwareomgevingen
- Inzet van 3D-printing i.f.v. toepassingen binnen de stadsdiensten
- Coördineren van deelprojecten die 3D-gebruik optimaliseren
- Inzetten op innovatie inzake de smart-city aspecten van de virtuele stad
- Initiatie-sessies organiseren (inzake 3D-tekenen, programmeren i.f.v. bewegende 3D-modellen, storytelling i.f.v. game development)
- Ontsluiten van 3D-data ondermeer door het beschikbaar stellen van Open 3D-data
- Door de hoeveelheid data die 3D-modellen vergen, hoort hier ook de problematiek van big-data bij
- Synergieën stimuleren tussen diverse beleidsdomeinen (beleidsvoorbereidend, ondersteunend en evaluerend) gebruik makende van 3D-technologie
- Inzet van 3D-game engines om animaties en leven in het 3D-model te brengen
- ...

"Gent in 3D" wordt aangestuurd door het 3D-team van Dienst Data en Informatie binnen het Departement Bedrijfsvoering van Stad Gent.

https://stad.gent/nl/over-gent-en-het-stadsbestuur/stadsbestuur/wat-doet-het-bestuur/gent-digitale-stad/gent-3d-virtuele-realiteit-vrarmr

1.2 Teletijdsmachine Gent

Het digitale 3D-model van Stad Gent is basis voor de "Teletijdsmachine Gent" bij middel van 4D, de tijdsdimensie.
Met een computer is mogelijk waartoe we in de echte fysieke wereld vandaag ogenschijnlijk nog niet toe in staat zijn, namelijk door de tijd reizen.
Een digitale copy in 3D van je eigen straat, buurt, wijk of stad laat toe om ook het verleden virtueel te reconstrueren of een prognose te maken van hoe de ontwikkeling zich in de toekomst zal aandienen en realiseren.
Een reconstructie van het oude Zuid-station en het wintercircus 150 jaar geleden, of het hart van de Stad in de 13de eeuw ? Het zijn projecten die al in de Gentse Teletijdsmachine bestaan. Maar ook het oude Prinsenhof of het Spaans kasteel 5 eeuwen geleden, zijn zo al terug tot leven gewekt.

1.3 VR (Virtual Reality), AR (Augmented Reality) en MR (Mixed Reality)

Stad Gent gebruikt VR (Virtual Reality), AR (Augmented Reality) en MR (Mixed Reality).

1.3.1 VR= Virtual Reality

Bij VR word je volledig ondergedompeld in een digitale 3D-wereld, zonder dat je nog de werkelijke wereld ziet.

1.3.2 AR= Augmented Reality

Voor AR heb je niet altijd een digitale 3D-omgeving nodig. Je projecteert je digitale (2D-)beelden op transparante media voor je ogen terwijl je dus ook nog de werkelijke wereld ziet.

1.3.3 MR= Mixed Reality

Bij MR combineer je de 2 voorgaande technieken. Een virtuele wereld kan bovenop de werkelijke wereld worden geprojecteerd.

1.3.4 Augmented Virtuality

Naast Augmented Reality is er ook nog Augmented Virtuality waarbij je werkelijke objecten integreert in de virtuele wereld.
Deze technieken worden ingezet bij interactieve communicatie bij ruimtelijke ontwikkelingsprojecten in de Stad Gent maar ook om terug te kijken naar het verleden. Een toepassing waarbij een timelapse video van de bouw van de Portus Ganda sluis geïntegreerd wordt in de digitale 3D-maquette van deze site is één van de tijd-ruimte voorbeelden.

2 METHODOLOGIE

In dit project worden historische gebouwen via ‘’*augmented virtuality”* bekeken. Het project werd opgedeeld in verschillende delen, het spreekt voor zich dat de verschillende delen individueel zijn uitgewerkt. De verschillende delen zullen nu kort beschreven worden. Het eerste deel bestond eruit om via locatie bepaling de relatieve afstand tot het gebouw te berekenen, alsook de hoek waaronder men het gebouw moest kunnen bekijken. Op papier lijkt dit eenvoudig, maar dit brengt toch enige complexiteit met zich mee tijdens de uitwerking hiervan. In het tweede deel en tevens ook laatste deel moest aan de hand van de gyroscoop het gebouw op de correcte manier bekeken worden.

Het algemene idee achter dit project is dat indien binnen- en of buitenlandse toeristen Gent komen bezoeken, deze terug in de tijd kunnen gaan en specifieke gebouwen van onze prachtige Arteveldestad kunnen bekijken in hun oorspronkelijke glorie. Door middel van de nieuwe technologie kunnen we het toerisme eenvoudig naar een hoger niveau tillen. Indien het systeem stabiel draait kunnen later ook andere steden volgen. Omdat dit louter over een ‘proof of concept’ gaat, zijn er verschillende extra features nog niet geïmplementeerd omdat deze relatief eenvoudig later kunnen worden toegevoegd. Onze eerste zorg was een stabiel systeem produceren. Dit systeem hield in dat men naar één voorbeeld gebouw kan kijken. Volgende basisideeën zijn wel geïmplementeerd, zo zal de gebruiker via de gyroscoop van het toestel wel de hoek van de camera kunnen aan passen én indien de gebruiker rondwandelt, zal zonder al te veel sprongen ook het beeld verplaatsen. In de onderstaande afbeelding ziet u het basisidee maar dit toegepast op timesquare in New York.

Figuur 1: Timesquare in New York

Bron: https://geoargames.blogspot.com/2019/01/?view=sidebar
Deel één aan de hand van latitude en longitude de locatie t.o.v. het gebouw vastleggen.

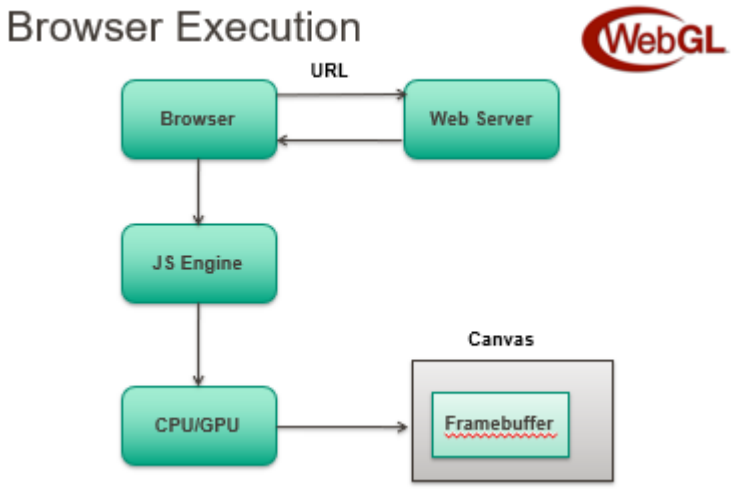
3 FRAMEWORK

3.1 OpenGL & WebGL

OpenGL is een ''*application programming interface*'' (API) die geschreven is om grafische modellen te kunnen weergeven op een computer door de grafische kaart aan te spreken. Dit maakt het mogelijk om 3D applicaties te schrijven die op een 2D computerscherm weergegeven worden. De implementatie is besturingssysteemonafhankelijk.

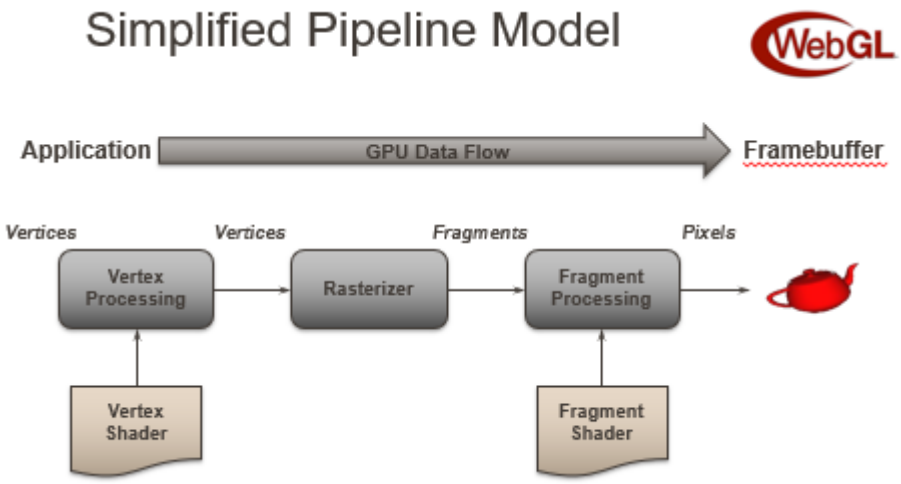
WebGL is een JavaScript implementatie van OpenGL ES 2.0. Het draait in alle recente browsers (zoals Chrome, Firefox, IE, Safari,…) en het is ook besturingssysteemonafhankelijk. De applicaties kunnen draaien op een externe server. Het grafisch weergeven wordt gerealiseerd binnenin de browser, gebruik makende van de lokale hardware.

WebGL gebruikt het HTML5 canvas element om grafische elementen in weer te geven. Het werkt uiteraard ook hand in hand met andere Web pakketten zoals CSS en jQuery.



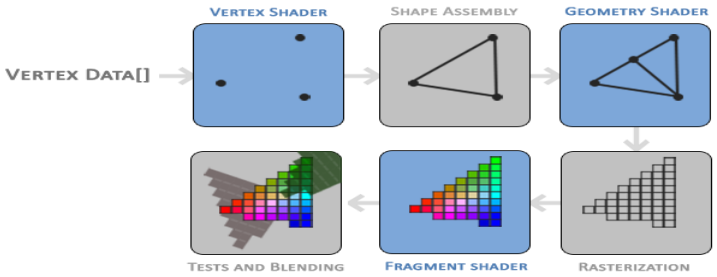
Figuur 2: Blokschema uitvoering in een browser

In bovenstaande blokschema van de uitvoering van een WebGL applicatie in een browser, is te zien dat een typische WebGL applicatie een mix is van HTML5, JavaScript en het ''*OpenGL Shading Language*'' (GLSL). De applicatie is toegankelijk via zijn url, alle moderne browsers kunnen JS draaien en ondersteunen HTML5. Het grafische ''*rendering*'' geschreven in JS, geeft het grafische weer in een HTML5 canvas. De WebGL code wordt van de server gehaald maar wordt gecompileerd in de JavaScript ''*engine*'' en draait op de lokale processor (CPU) en grafische kaart (GPU).



Figuur 3: Vereenvoudigde grafische "pipeline"

In bovenstaande figuur is een beknopte versie van de stappen die doorlopen worden om een grafisch 3D model op een 2D scherm weer te geven. Grafische modellen zijn hier opgebouwd uit verbonden punten en randen.

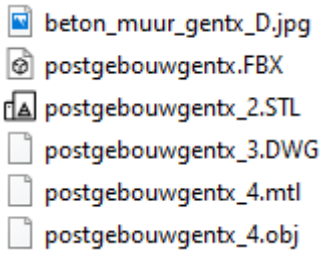


Figuur 4: Visuele voorstelling van het Stappenplan

Het principe is gemakkelijker te verstaan a.d.h.v. figuur 4. Om 3D om te zetten naar 2D (3D voxels naar 2D pixels), zodat dit weergegeven kan worden op een computerscherm, wordt er een raster over het model geplaatst en worden de voxels gemapt naar 2D.

3.2 Three.js

Three.js is een JavaScript ‘’*library*’’ die het gebruik van WebGL vereenvoudigt en de functionaliteit uitbreidt. Bij aanvang van het project werd er een 3D model van een bib in Gent aan ons ter beschikking gesteld (zie figuur 5).



Om gebruik te maken van dit 3D model maken we gebruik van verschillende ‘’*loaders*’’ van Three.js (zie figuur 6).

Meer bepaald worden de OBJLoader2 en de MTLLLoader gebruikt om respectievelijk het .obj en .mtl bestand te laden. Beton_muur wordt gebruikt als textuur.

Figuur 5: Bestanden van het 3D-model van het gebruikte postgebouw

```
<script src="js/loaders/MTLLoader.js"></script>
<script src="js/loaders/LoaderSupport.js"></script>
<script src="js/loaders/OBJLoader2.js"></script>
<script src="js/loaders/obj2/bridge/MtlObjBridge.js"></script>
```

Figuur 6: Gebruikte loaders van Three.js (screenshot van index.html bestand)

De website van Three.js https://threejs.org/docs/ bevat een duidelijk documentatie en veel voorbeelden om te leren werken met de library.

Er is dus uiteraard inspiratie gehaald uit het voorbeeld om een .obj bestand te laden dat te vinden is op https://threejs.org/examples/#webgl_loader_obj2.

Gelieve hoofdstuk 4 raad te plegen voor een uitgebreide technische uitleg over hoe onze applicatie werkt.

3.3 Motion sensors

Een belangrijk onderdeel van het project is om het 3D object te verdraaien wanneer de website geraadpleegd wordt met een smartphone. Aan de hand van oriëntatiesensoren van de smartphone verdraait het object als de gebruiker zijn smartphone beweegt. Zo kan men een bovenaanzicht verkrijgen van de bib en deze onder willekeurige hoeken bekijken.

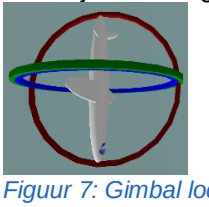
Om dit te verwezenlijken wordt er gebruik gemaakt van het bestand motion-sensors.js dat geschreven is door W3C (zie https://www.w3.org/TR/generic-sensor/). Intel® heeft handige voorbeelden voor het gebruik van deze sensor API op hun ‘’*github repository*’’ staan (zie https://github.com/intel/generic-sensor-demos).

3.3.1 Quaternionen

De quaternionen zijn een uitbreiding van de complexe getallen. Zoals de complexe getallen een tweedimensionale uitbreiding zijn van de reële getallen, zo zijn de quaternionen een tweedimensionale uitbreiding van de complexe getallen, en aldus een vierdimensionale uitbreiding van de reële getallen.

Quaternionen worden onder andere gebruikt in volgende toepassingen; robotica, computer grafiek, computer visie, navigatie,…

In deze applicatie worden quaternionen gebruikt om rotaties toe te passen in 3D. Ze hebben als voordeel dat ze niet leiden aan het probleem ‘’*gimbal lock*’’ dat voorkomt bij rotaties uit te voeren met Euler-hoeken. Gimbal lock komt voor als 2 van de 3 rotatie-assen in hetzelfde vlak terechtkomen, waarbij er in dat geval nog maar 1 vrijheidgraad over is (zie figuur 7).



Figuur 7: Gimbal lock probleem

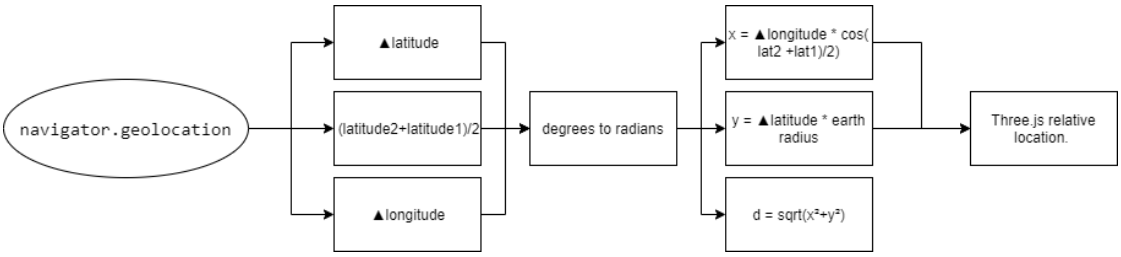
Een quaternion wordt in ons geval dus vanuit de sensoren van de smartphone gehaald en vervolgens wordt deze rotatie toegepast op de bib.

De quaternion wordt toegepast op de bib zodat het lijkt alsof de camera beweegt. Het is handig dat de bib beweegt en niet de camera want zo kan de plaatsing van de camera in de 3D ruimte gescheiden worden van de verdraaiing van de bib (dus vermijden dat 2 verschillende functies de camera aanpassen).

4 ALGORITMES

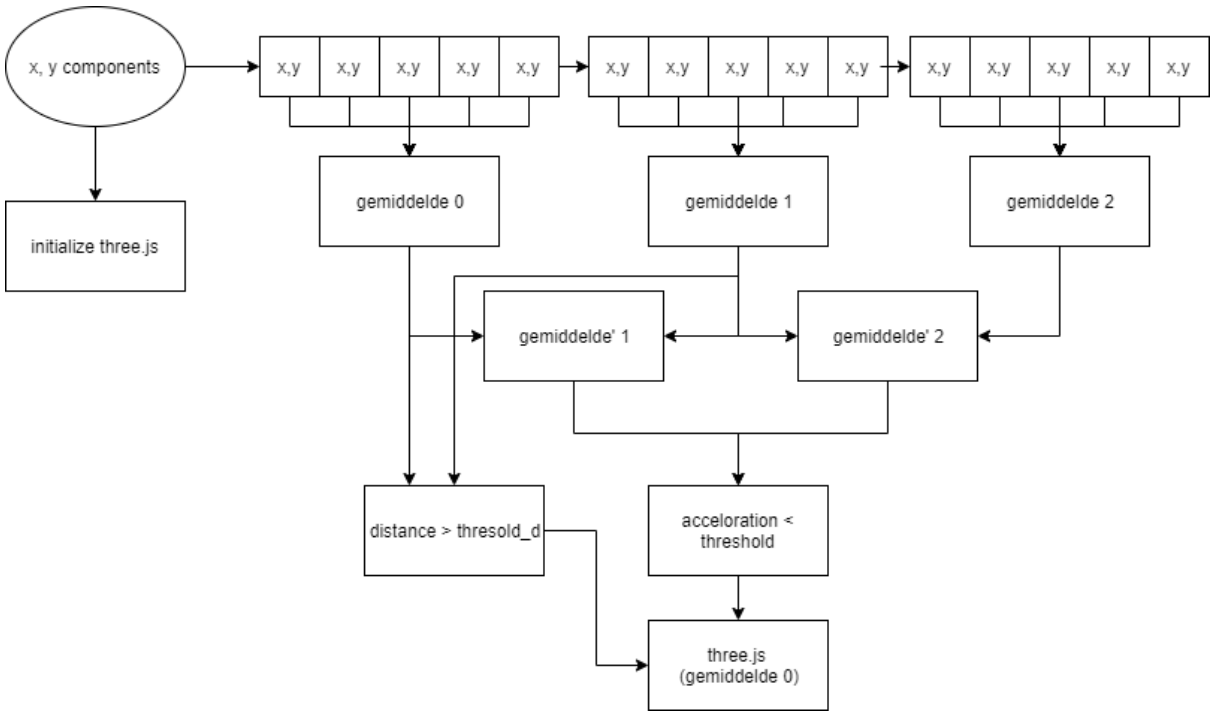
4.1 Locatie bepaling

Locatie bepaling:
Volgende algoritmes werden in javascript uitgewerkt, er werd gebruik gemaakt van Three.js om de visualisatie van het gebouw te realiseren. Om de relatieve locatie ten opzichte van het gebouw te controleren wordt via javascript de locatie van het toestel opgevraagd. Eens de gebruiker toestemming heeft gegeven om zijn locatie te bezichtigen, wordt deze op vaste intervallen gelogd. De locatie komt onder de vorm van longitude en latitude binnen (graden). Omdat de afstand moet worden bepaald tussen ons toestel én het gebouw wordt er een vereenvoudiging gebruikt. Hier wordt namelijk gewerkt met erg korte afstanden. Dit heeft tot gevolg dat, indien men ervan uit zou gaan dat de aarde een mooie sfeer is, hier met simpele driehoeksmeetkunde kan gewerkt worden.



Figuur 8: flowchart geolocatie

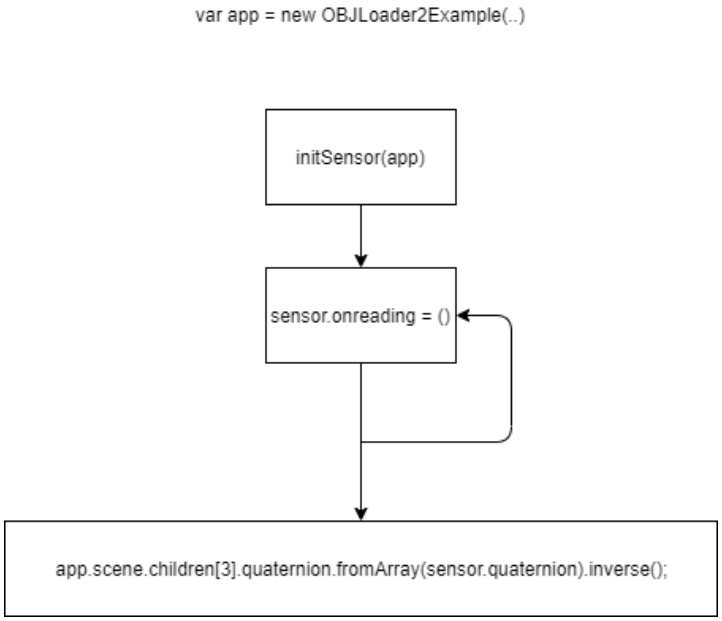
In bovenstaande figuur kan men de flowchart zien die alles omtrent de locatie bepaling in javascript uit de doeken doet. Als eerst wordt via de functie navigator.geolocation de huidige locatie van de gebruiker opgevraagd. Indien deze functie geen locatie biedt, zal een error worden terug gegeven. Eens de gegevens verwerkt zijn en men vanuit de longitude en de latitude respectievelijk de x en y component bepaald heeft, is het nodig om deze te filteren. De filter zal er voor zorgen dat er geen mogelijkheid tot drift ontstaat wanneer de gebruiker met zijn telefoon rondwandelt om het gebouw te bekijken.
Het algoritme om drift te vermijden kan men bekijken in figuur 9. Er wordt gewerkt met een array waar steeds een nieuw element vooraan wordt toegevoegd. Aan het einde van iedere cyclus wordt het laatste element van de array verwijderd. Om de 5 cycli worden de verschillende gemiddeldes opnieuw bepaald. Dit zodat er minder rekening gehouden wordt met uitschieters tijdens het bepalen van de locatie. Het is efficiënter om enkel de laatste 3 gemiddeldes bij te houden, op deze manier moeten er minder dubbele berekeningen plaatsvinden. Indien de afstand tussen de eerste twee gemiddeldes te groot is zal het nieuwe gemiddelde worden doorgegeven aan het three.js systeem. Er is ook nog een andere mogelijkheid, indien de persoon aan een ongeveer constante gemiddelde snelheid zichzelf verplaatst zal op dit moment ook het nieuwe gemiddelde worden ingesteld als het centrale punt.



Figuur 9: algoritme tegen drift.

4.2 Bepaling van de oriëntatie

Zoals in 3.3 wordt besproken, gaat men de oriëntatie verwezenlijken door het 3D model van de bib te verdraaien gebruik makende van de quaternion die men uit de sensor haalt. Voor een beknopte werking zie onderstaande flowchart (figuur 10).



Nadat het 3D object ingeladen werd, zal de Sensor geïnitialiseerd worden. Hierna wordt op regelmatige basis (60 Hz) de informatie gelezen met de sensor.onreading functie en kan men instructies uitvoeren telkens wanneer nieuwe sensorinformatie binnenkomt.

Het is dan ook in die functie dat de bib verdraaid wordt volgens de gelezen quaternion van de sensor (afkomstig van de smartphone van de gebruiker).

Figuur 10: flowchart ophalen van de sensorinformatie en toepassen op het model. (app.scene.children[3] stelt het ingeladen 3D object voor).

5 CONCLUSIE

In dit project zijn we begonnen met het zoeken naar een geschikte framework om het project te realiseren, hiervoor hebben we ons verdiept in de werking van OpenGL en WebGL. Three.js biedt veel mogelijkheden en bevat veel voorbeelden om mensen op weg te helpen met het schrijven van grafische applicaties voor browsers.
Het object verdraaien a.d.h.v. quaternions werkt nog niet onder alle omstandigheden waardoor dit als toekomstige stap nog verder onderzocht moet worden. Er wordt gewerkt met een RelativeOrientationSensor({referenceFrame: "screen"}) waardoor er geen mapping van het coördinatenstelsel van het toestel naar het coördinatenstelsel van het scherm hoeft te gebeuren. Dit werkt echter nog niet volledig zoals gewenst. Voor dit "proof of concept" hebben we geopteerd voor een stabiel systeem zonder extra's, het mooier maken van de applicatie kan dus als volgende stap gebeuren. De plaatsing van de camera volgens de coördinaten van de gebruiker, om een correcte kijkhoek te bekomen van de bib, werkt wel zoals gewenst.

REFERENTIES

1. https://stad.gent/nl/over-gent-en-het-stadsbestuur/stadsbestuur/wat-doet-het-bestuur/gent-digitale-stad/gent-3d-virtuele-realiteit-vrarmr
2. https://www.w3.org/TR/generic-sensor/
3. https://github.com/intel/generic-sensor-demos
4. https://geoargames.blogspot.com/2019/01/?view=sidebar
5. https://threejs.org/docs/
6. https://threejs.org/examples/#webgl_loader_obj2
7. https://threejs.org/docs/#api/en/math/Quaternion
8. https://en.wikipedia.org/wiki/Quaternion
9. https://en.wikipedia.org/wiki/Quaternions_and_spatial_rotation
10. https://en.wikipedia.org/wiki/Gimbal_lock
11. https://www.cs.unm.edu/~angel/WebGL/